

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіоелектроніки

Факультет комп'ютерних наук і технологій

(повне найменування інституту, факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ
РОЗПІЗНАВАННЯ МЕДИЧНИХ УПАКОВОК
RESEARCH AND SOFTWARE IMPLEMENTATION OF MEDICAL
PACKAGING RECOGNITION METHODS

Виконав: студент(ка) 2 курсу, групи КНТ-220м

Спеціальності 122 Комп'ютерні науки

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Системи штучного інтелекту

Гончаренко Д.А.

(прізвище та ініціали)

Керівник Олійник А.О.

(прізвище та ініціали)

Рецензент Скрупський С.Ю.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Інститут , факультет ІРЕ, ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти магістр
Спеціальність 122 Комп'ютерні науки
(код і найменування)
Освітня програма (спеціалізація) Системи штучного інтелекту
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С. О. Субботін
“ ” 2021 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Гончаренка Дмитра Андрійовича

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація методів розпізнавання медичних упаковок

Research and Software Implementation of Medical Packaging Recognition Methods

керівник проєкту (роботи) Олійник Андрій Олександрович, д.т.н., доцент

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “05” листопада 2021 року № 435

2. Строк подання студентом проєкту (роботи) 07 грудня 2021 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити 1. Аналіз предметної області. 2. Вибір інструментів розробки програмного забезпечення. 3. Основні рішення щодо реалізації компонентів системи. 4. Керівництво програміста. 5. Керівництво користувача.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділи	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	Олійник А. О., професор		
Нормоконтроль	Липовець М. В., асистент		

7. Дата видачі завдання “02” вересня 2021 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області	2-4 тижні	Розділ 1
3	Вибір засобів розробки програмного продукту	5 тиждень	Розділ 2
	Розробка архітектури програми	6 тиждень	Розділ 3
4	Розробка програмного продукту	7-9 тижні	Розділ 4-5
5	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	10 тиждень	Додатки
6	Захист роботи	11-16 тижні	

Студент(ка)

(підпис) Гончаренко Д. А.
(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис) Олійник А. О.
(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
106 с., 33 табл., 63 рис., 4 додатки, 28 джерел.

ГЕНЕТИЧНИЙ АЛГОРИТМ, КЛАСИФІКАЦІЯ, НЕЙРОННА
МЕРЕЖА, РОЗПІЗНАВАННЯ МЕДИЧНИХ УПАКОВОК, KERAS, NUMPY,
MATPLOTLIB, JUPYTER NOTEBOOK.

Об'єкт дослідження – процес розпізнавання медичних упаковок.

Предмет дослідження – методи для розпізнавання медичних упаковок
та класифікації їх за назвою препарату та дозуванням.

Мета роботи – дослідження, модифікація та програмна реалізація
методів розпізнавання медичних упаковок.

Метод дослідження – традиційні моделі, нейронні мережі,
комбіновані методи (нейронні мережі і генетичні алгоритми).

Матеріали, методи та технічні засоби: структурне та об'єктно-
орієнтоване програмування, мова програмування Python, середовище
виконання PyCharm та jupyter notebook, відеокарта NVIDIA, персональний
комп'ютер з процесором Intel Core 5 під управлінням операційної системи
Microsoft Windows 10.

Результати. Створено програмний продукт для розпізнавання
медичних упаковок та класифікації їх за назвою препарату та дозуванням.

Висновки. Розроблено програмний продукт, який дозволяє
розпізнавати медичні упаковки та класифікувати їх за назвою препарату та
дозуванням.

Основні положення і результати роботи були представлені та
обговорювалися на міжнародних конференціях (котрі входять до міжнародної
наукометричної бази Scopus):

- IEEE International Scientific-Practical Conference: Problems of Infocommunications Science and Technology, PIC S and T (Kharkiv, 2018, 2020, 2021, Kyiv, 2019);
- 15th International Conference on the Experience of Designing and Application of CAD Systems, CADSM 2019 (Polyana, Ukraine, 2019);
- 10th International Conference on Advanced Computer Information Technologies, ACIT 2020 (Deggendorf, Germany, 2020 p.);
- International Workshop: Informatics and Data-Driven Medicine ((IDDM), Lviv, 2019, Sweden, 2020);
- Information Technology and Interactions (Kyiv, 2020).

Маю розділ у колективній монографії *Lecture Notes on Data Engineering and Communications Technologies*[this link is disabled](#) (Germany, 2021), входить до міжнародної наукометричної бази Scopus.

Галузь використання – розпізнавання медичних упаковок та класифікація їх за назвою препарату та дозуванням у аптеках провізорами.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 106 pages, 33 tables, 63 figures, 4 appendices, 28 sources.

GENETIC AGGLORITHM, CLASSIFICATION, NEURAL NETWORK, MEDICAL PACKAGING RECOGNITION, KERAS, NUMPY, MATPLOTLIB, JUPYTER NOTEBOOK.

The object of research is the process of recognizing medical packaging.

The subject of research - methods for recognizing medical packaging and classifying them by drug name and dosage.

The purpose of the work - research, modification and software implementation of methods for recognizing medical packaging.

Research method - traditional models, neural networks, combined methods (neural networks and genetic algorithms).

Materials, methods and technical means: structural and object-oriented programming, Python programming language, PyCharme and jupyter notebook runtime, NVIDIA video card, personal computer with Intel Core 5 processor running Microsoft Windows 10 operating system.

Results. A software product has been created for the recognition of medical packaging and their classification by drug name and dosage.

Conclusions. A software product has been developed that allows to recognize medical packages and classify them by drug name and dosage.

The main provisions and results of the work were presented and discussed at international conferences (which are part of the international scientometric database Scopus):

- IEEE International Scientific-Practical Conference: Problems of Infocommunications Science and Technology, PIC S and T (Kharkiv, 2018, 2020, 2021, Kyiv, 2019);

- 15th International Conference on the Experience of Designing and Application of CAD Systems, CADSM 2019 (Polyana,, Ukraine, 2019);
- 10th International Conference on Advanced Computer Information Technologies, ACIT 2020 (Deggendorf, Germany, 2020);
- International Workshop: Informatics and Data-Driven Medicine (IDDM), Lviv, 2019, Sweden, 2020);
- Information Technology and Interactions (Kyiv, 2020).

I have a section in the collective monograph Lecture Notes on Data Engineering and Communications Technologies[this link is disabled](#) (Germany, 2021), part of the international scientometric database Scopus.

Area of use - recognition of medical packaging and their classification by drug name and dosage in pharmacies by pharmacists.

ЗМІСТ

	С.
Перелік скорочень та умовних позначень	11
Вступ	12
1 Аналіз предметної області.....	13
1.1 Завдання розпізнавання зображень	13
1.2 Вибір математичного апарату програмного забезпечення	16
1.2.1 Метод Віюли-Джонса.....	16
1.2.2 Багатошарова нейронна мережа	21
1.2.3 Рекурсивна нейромережа	22
1.2.4 Рекурентна нейромережа	23
1.2.5 Inception	23
1.2.6 Нейронна мережа VGG	24
1.2.7 YOLO	25
1.2.8 Згорткова нейронна мережа (CNN)	26
1.2.9 LSTM.....	28
1.3 Аналіз функцій активацій	30
1.3.1 Лінійна функція.....	30
1.3.2 Ступінчаста функція.....	31
1.3.3 Сігмоїда.....	32
1.3.4 Функція гіперболічного тангенсу	34
1.3.5 Функція ReLu	34
1.4 Аналіз методів для розпізнавання	36
1.5 Аналіз галузі штучного інтелекту для розпізнавання образів	37
1.6 Постановка завдання	38

1.7	Висновки за розділом 1	42
2	Вибір інструментів розробки програмного забезпечення	43
2.1	Вибір інструментальних засобів розробки	43
2.2	Середовище розробки Jupyter Notebooks	46
2.3	Фреймворк Keras	47
2.4	Фреймворк для роботи з масивами NumPy	49
2.5	Фреймворк TensorFlow	49
2.6	Бібліотека Pandas	50
2.7	Бібліотека Matplotlib	50
2.8	Висновки за розділом	51
3	Основні рішення щодо реалізації компонентів системи	52
3.1	Структура програмного забезпечення	52
3.2	Модифікація згорткової нейромережі	53
3.2.1	Архітектура	55
3.2.2	Модифікований генетичний алгоритм	58
3.3	Висновки за розділом 3	62
4	Керівництво програміста	63
4.1	Призначення та умови застосування програми	63
4.1.1	Призначення програми	63
4.1.2	Умови застосування програми	63
4.2	Звернення до програми	64
4.3	Вхідні та вихідні дані	65
4.4	Повідомлення програмісту	66
4.5	Висновки за розділом	67
5	Керівництво користувача	68

	10
5.1 Призначення та умови застосування програми	68
5.1.1 Призначення програми.....	68
5.1.2 Умови застосування програми	68
5.2 Виконання програми.....	69
5.2.1 Екран входу	69
5.2.2 Екран роботи алгоритму	69
5.3 Порівняння роботи згорткової нейронної мережі	70
5.4 Висновки за розділом	72
Висновки	73
Перелік джерел посилання.....	75
Додаток А Технічне завдання.....	79
Додаток Б Опис програми.....	85
Додаток В Текст програми.....	93
Додаток Д Слайди презентації.....	100

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

CNN – convolutional neural networks;
DMTL – deep multi-task learning;
GBM – gradient boosting machine;
HRBR – hybrid rule-based reasoning;
IDPA – improved diabetes prediction algorithm;
LSTM – long short-term memory;
MLP – multilayer perceptron;
RBM – restricted Boltzmann machine;
RSR – rough set reasoning;
SKCNN – sigmoid kernel convolutional neural network;
STL – single task learning;
SVM – support vector machine;
БШП – багатошаровий персептрон;
ЗНМ – згорткова нейронна мережа;
ПЗ – програмне забезпечення;
РО – розпізнавання образів;
РНМ – рекурентна нейронна мережа;
ШІ – штучний інтелект;
ШНМ – штучна нейронна мережа.

ВСТУП

Розробка, дослідження та реалізація методів вирішення завдань аналізу, розпізнавання та оцінювання зображень є одним із провідних напрямів інформатики [1]. Результати відповідних фундаментальних та прикладних досліджень безпосередньо використовуються в технічній діагностиці, неруйнівний контроль, дистанційне зондування, екологічний моніторинг, прогнозування та діагностику в медицині, планування, пошук у геології, прогнозування в хімії, автоматизацію наукових досліджень. Ця тематика приваблює значну кількість дослідників, проте, переважна більшість робіт у цій галузі має емпіричний характер [1].

До цих пір здебільшого застосовуються методи обробки зображень, запозичені із цифрової обробки сигналів, та традиційні математичні методи (головним чином, статистичні) [1]. Відомі публікації, присвячені можливостям розширення використовуваного під час аналізу зображень математичного апарату (наприклад, вейвлетів, фракталів, методу регуляризації) та застосування методів та апарату математичної теорії розпізнавання [1].

На сьогоднішній день методи розпізнавання образів (РО) знаходять практичне застосування в різних галузях людської діяльності скрізь, де одним із основних завдань, що стоять перед людиною, є класифікація деяких об'єктів або явищ, яким можна зіставити їх формалізований опис. Однією зі сфер застосування РО є аптечна справа. Аптечна справа передбачає високу відповідальність за безпеку, тому необхідно надійно відстежувати правильність визначення медичних упаковок по прибуттю на аптеку (за назвою та дозуванням) для встановлення їх в аптечному роботі. Дане рішення дозволить спростити роботу для першостольників і надати більше уваги покупцям для уникнення черг [2].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Завдання розпізнавання зображень

Комп'ютерна обробка зображень передбачає обробку цифрових зображень за допомогою комп'ютерів або спеціалізованих пристроїв, які побудовані на цифрових сигнальних процесорах. У цьому під обробкою зображень розуміється як поліпшення зорового сприйняття зображень, а й класифікація об'єктів, виконувана під час аналізу зображень.

Завдання розпізнавання - це завдання аналізу та сортування даних , що повині складатись з наступних етапів:

- перетворення вихідних даних до виду, зручного для розпізнавання;
- власне розпізнавання (вказівка власності об'єкта певному класу).

Для такого типу завдань слід використовувати поняття аналогії або подібності об'єктів і формулювати правила, на підставі яких об'єкт зараховується в один і той же клас або в різні класи [3]. А також можна оперувати набором прецедентів - прикладів, класифікація яких відома і які у вигляді формалізованих описів можуть бути пред'явлені алгоритму розпізнавання для налаштування завдання в процесі навчання [3].

Часто недоступна інформація для точних математичних моделей або виграш від використання моделі та математичних методів невідповідно до витрат [4].Через це для них важко будувати теорії та використовувати класичні математичні методи [4].

В основному існують наступні типи завдань для розпізнавання:

- завдання - віднесення існуючого, пред'явленого унікального об'єкта за описом до одного із існуючих класів (навчання з учителем) [3];
- завдання автоматичної класифікації - розбиття усіх об'єктів, ситуацій, явищ за їх описами на систему класів, що не перетинаються (таксономія, кластерний аналіз, самонавчання) [3];

- завдання відбору найбільш інформативного набору ознак під час розпізнавання [3];
 - завдання приведення усіх вихідних даних до виду, зручного для подальшого розпізнавання [3];
 - динамічне розпізнавання або динамічна класифікація - [4].
- Розпізнавання образів, це є науковий підхід яке в свій час пов'язане з розробкою принципів, ідей та побудовою систем, призначених для визначення належності певного(унікального) об'єкта до одного із заздалегідь зазначених(або ні) класів об'єктів [3]. Під об'єктами розпізнавання образів розуміють пролейблені(або ні) предмети, явища, процеси, об'єкти, ситуації, сигнали [3].

Кожен об'єкт описується набіром виділяючихся основних характеристик (ознак, властивостей) $X = x_1, \dots, x_i, \dots, x_m$, де i -а координата вектора, X визначає значення i -ї характеристики, та додатковою характеристикою, A що вказує на належність об'єкта до деякого класу (образу) [4].

Набір заздалегідь пролейблених об'єктів, тобто таких, у яких завчасно відомі характеристики X і A , використовується виявлення закономірних зв'язків між значеннями цих характеристик, і тому називаються тестовоїювибіркою. Певні, або зайві об'єкти тестової та валідаційної вибірок називаються реалізаціями [4].

Одне з основних завдань Розпізнавання образів - вибір правила (розв'язуючої функції) D , відповідно до якого за значенням контрольної реалізації X встановлюється її приналежність до одного з образів, тобто вказуються «найбільш відносної» змінної характеристики A для цього X [4].

Успіх у розв'язанні задачі Розпізнавання образів залежить в значній мірі від того, наскільки професійно та правильно були обрані ознаки X . Вихідний набір характеристик часто буває занадто перевантажений(великим), найбільш принципівих на відміну одного образу

від іншого [5]. Так, для завдань медичної спрямованості(діагностики) необхідно визначити, усі симптоми та їх поєднання (синдроми) слід використовувати при постановці діагнозу цього захворювання [5].- Тому проблема вибору інформативних ознак є важливою складовою проблеми розпізнавання образів [5].

Проблема розпізнавання образів тісно пов'язана із завданням попередньої класифікації або таксономією [5].

В головному завданні для розпізнавання образів побудови вирішальних функцій B використовуються рішачі зв'язки з поміж характеристик X і A , що виявляються на навчальній вибірці, та певні додаткові спеціальні припущення, наприклад як гіпотеза: характеристики X для реалізацій образів є випадкові вибірки з генеральних сукупностей з нормальним розподілом; реалізації одного образу розташовані «компактно» (у певному сенсі); ознаки у наборі X незалежні і т.д [4].

У тематиці розпізнавання образів зазвичай використовуються ідеї та результати багатьох інших наукових напрямів - математики, кібернетики, психології тощо [4].

Коли кажуть або згадують про системи розпізнавання образів то, зазвичай мають на увазі комплекси засобів, призначених до вирішення описаних далі, завдань [4].

Методи розпізнавання образів, в свою чергу, використовуються у процесі машинної діагностики усіх захворювань, для прогнозування корисних копалин у геології, психології, криміналістиці, лінгвістиці, медицині .хімії, ядерній та космічній фізиці, в автоматизованих системах управління тощо [4].

Одним із способів розв'язання задачі розпізнавання образів є використання спектральних та еволюційних методів [3].

1.2 Вибір математичного апарату програмного забезпечення

1.2.1 Метод Віюли-Джонса

Даний метод був запропонований Полом Віолою та Майклом Джонсом у 2001 році для розпізнавання облич, але даний метод також можна застосовувати для розпізнавання різного роду об'єктів, оскільки даний метод характеризується достатньо високою точністю пошуку та високою стійкістю до викривлень зображень [4]-[6]. Результати використання методу Віюли-Джонса на прикладі розпізнавання облич та пішоходів розглянуті у [7].

В методі Віюли-Джонса використовуються наступні ідеї:

- ознаки Хаара [8];
- інтегральне представлення зображення [9];
- метод AdaBoost [10]-[12];
- принцип скануючого вікна [13];
- каскадна структура класифікатору.

Ознака Хаара – відображення $f: X \rightarrow D_f$, де D_f – множина допустимих значень ознаки. Якщо задані ознаки $f_1, \dots, f_n$, то вектор ознак $x \in (f_1(x), \dots, f_n(x))$ називається ознаковим описом об'єкту $x \in X$ [14]. Ознакові описи допустимо ототожнювати з самими об'єктами [14]. При цьому множину $X = D_{f_1} * \dots * D_{f_n}$ називають ознаковим простором [14].

Графічне представлення ознак Хаара представлено на рисунку 1.1.

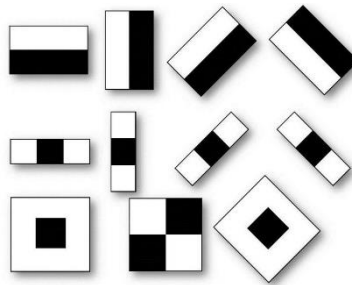


Рисунок 1.1 – Ознаки Хаара

Для вікна фіксованого розміру ознака Хаара являє собою множину прямокутних областей білого і чорного кольорів. Значення ознаки для аналізованої області зображення обчислюється за формулою (1.1):

$$F = W - B, \quad (1.1)$$

де W – це сума значень пікселів, які знаходяться в білій області ознаки,

B – сума значень пікселів які знаходяться в чорній області.

Таким чином, при обчисленні значення ознаки Хаару, буде виконано $n*m$ операцій, де n та m – ширина і висота аналізованої області, що вимагає багато часу. Прискорити процес обчислень дозволяє інтегральне представлення зображень.

Інтегральне представлення зображень [14] є матрицею, яка збігається з розмірами вхідного зображення. Значення кожної комірки даної матриці це сума інтенсивностей пікселів, що знаходяться лівіше і вище даної комірки [14]. Елементи матриці обчислюються за формулою(1.2):

$$L[x, y] = \sum_{i=0}^y \sum_{j=0}^x I[i, j], \quad (1.2)$$

де $I[i, j]$ – яскравість пікселя вихідного зображення.

Кожен елемент матриці $L[x, y]$ – сума пікселів в прямокутнику від $(0,0)$ до (x,y) . Розрахунок можливо провести за формулою(1.3):

$$L(x, y) = I(x, y) - L(x-1, y-1) + L(x, y-1) + L(x-1, y). \quad (1.3)$$

Побудувавши інтегральну матрицю можна досить швидко обчислювати суму пікселів довільного прямокутника довільної площі [14].

Значення ознаки Хаара можна обчислити використовуючи кілька операцій, кількість яких залежить від кількості прямокутників в ознаці і не залежить від площі аналізованої області.

Метод AdaBoost є розвитком методу Boosting, що полягає в комбінації слабких класифікаторів, щоб отримати сильний класифікатор (рис. 1.2). Сильним називається класифікатор, що допускає мало помилок класифікації. Слабкий класифікатор, навпаки, не дасть надійно розділити на класи або давати точні прогнози, в процесі видає багато помилок, що не дозволяє надійно розділяти класи, а також давати точні прогнози [32].



Рисунок 1.2 – Концепція AdaBoost

Як правило, застосовується один з алгоритмів AdaBoost – Discrete AdaBoost, Real AdaBoost, LogitBoost, Gentle AdaBoost і може використовувати довільне число класифікаторів та виконувати навчання на одному наборі прикладів, по черзі застосовуючи їх на різних етапах [10] - [14].

Алгоритм AdaBoost наступний:

а) дані об'єкти вибірки x і класи y для розподілу об'єктів:

$$(x_1, y_1), \dots, (x_m, y_m), \text{ де } x_i \in X, y_i \in Y = \{-1, 1\}$$

б) ініціалізуються ваги об'єктів:

$$D_1(i) = \frac{1}{m}, i = 1, \dots, m ;$$

в) для кожного простого класифікатору $t = 1, \dots, T$:

1) знаходиться класифікатор $h_t : X \rightarrow \{-1, 1\}$, який зменшує зважену помилку класифікації: яка розраховується по формулі (1.4):

$$h_t = \arg \min_{h_j \in H} e_j, \quad (1.4)$$

де e_j – зважена помилка класифікації h_j :

$$e_j = \sum_{i=1}^m D_t(i) [y_i \neq h_j(x_i)] ;$$

2) якщо для простого класифікатора t , виконується нерівність $e_t \geq 0.5$, то процес навчання зупиняється;

3) обчислюється $a_t \in \mathbb{R}$:

$$a_t = \frac{1}{2} \ln \frac{1 - e_t}{e_t}, ;$$

4) оновлюються ваги об'єктів:

$$D_{t+1}(i) = \frac{D_t(i) e^{-a_t y_i h_t(x_i)}}{Z_t},$$

де Z_t – нормалізуючий параметр, такий що:

$$\sum_{i=1}^m D_{t+1}(i) = 1;$$

5) виконується побудова результуючого класифікатора:

$$H(x) = \text{sign} \left(\sum_{t=1}^T a_t h_t(x) \right).$$

Для винесення рішення, чи присутній шуканий об'єкт на зображенні, необхідно виконання умови сильного класифікатора:

$$H(x) = \begin{cases} 1, \text{ якщо } \sum_{t=1}^T a^{(t)} h_j^{(t)} \geq \frac{1}{2} \sum_{t=1}^T a^{(t)} \\ 0, \text{ інакше} \end{cases}$$

Для пошуку об'єкта використовується принцип скануючого вікна: зображення сканується вікном пошуку, а потім до кожного положення вікна пошуку застосовується класифікатор (рис. 1.3) [15].

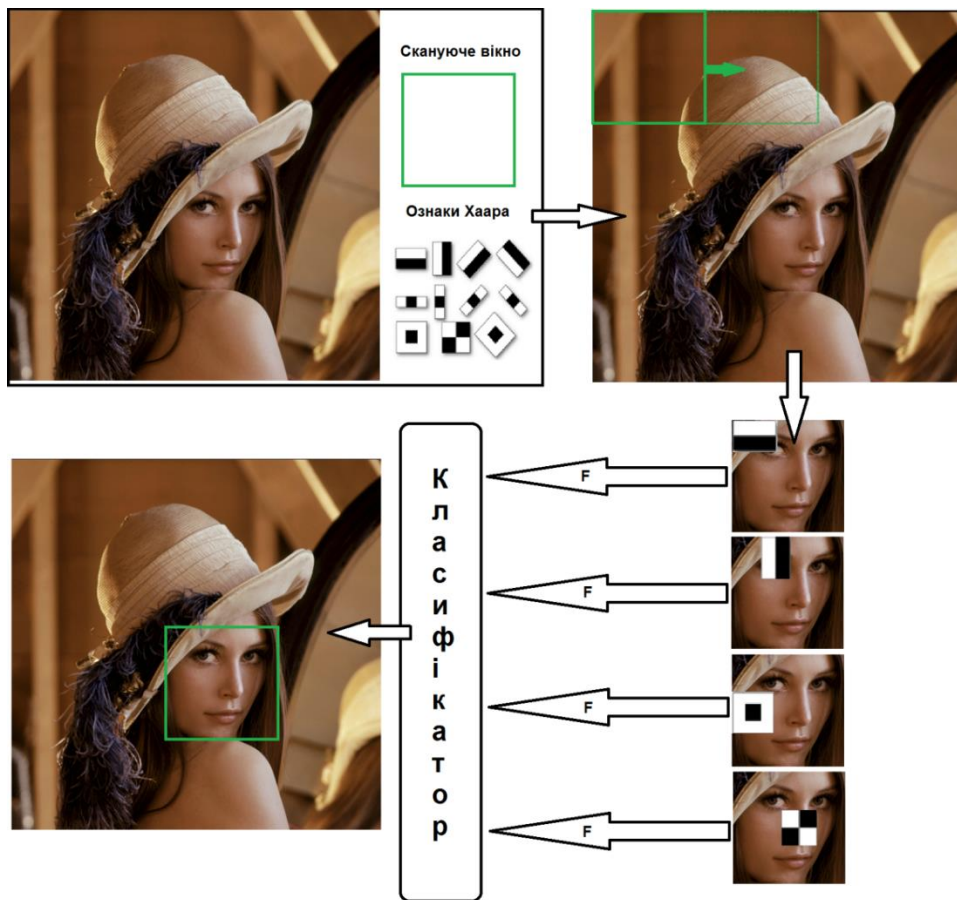


Рисунок 1.3 – Принцип роботи скануючого вікна

Каскадна модель сильних класифікаторів – дерево прийняття рішень, де кожен вузол дерева побудований таким чином, щоб детектувати образи і відхиляти регіони, що не є шуканими образами [13].

Складність навчання каскадних класифікаторів залежить від кількості етапів, кількості прикладів і кількості ознак. Після навчання класифікатор можливо застосувати до зображення:

- спочатку виконується прохід «простими» класифікаторами, причому «негативні» вікна відкидаються;
- позитивний результат першого класифікатора запускає другий класифікатор, другий запускає третій, і так далі;
- негативний результат класифікатора призводить до переходу до наступного скануючого вікна;
- послідовність класифікаторів стає складнішою, тож, зменшується кількість помилок.

1.2.2 Багатошарова нейронна мережа

Багатошаровими нейроні мережі – це згруповані мережі в яких нейрони згруповані у шари [14] -[19]. Виходячи з цього, кожен нейрон попереднього шару пов'язаний з усіма наступними нейронами послідуючих шарів, а всередині шарів зв'язки між нейронами відсутні, рисунку 1.5.

Шари нумеруються зліва направо. Перший шар називають вхідним чи розподільним. Його нейрони (які також називають вхідними) приймають елементи вектора ознак і розподіляють їх нейронами наступного шару. При цьому обробка даних у вхідному шарі не провадиться.

Останній шар називається вихідним. На виходах його нейронів (вони називаються вихідними) формується результат роботи мережі елементи вихідного вектора.

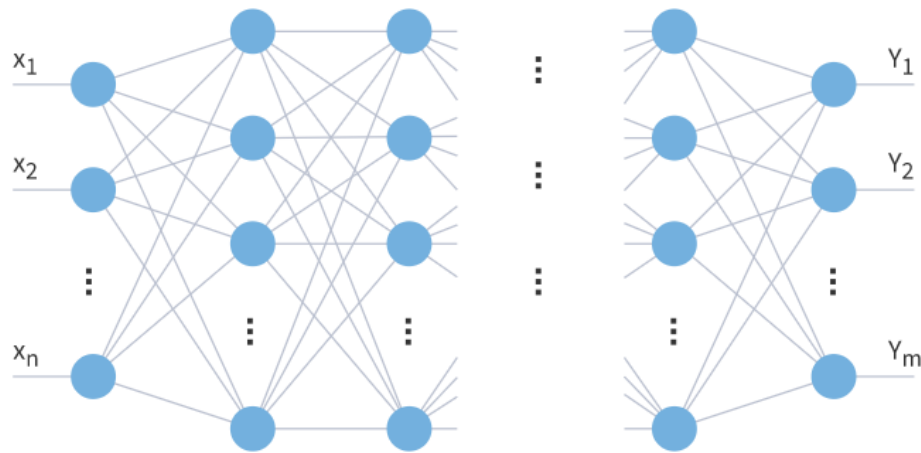


Рисунок 1.5 – Багатошарова нейронна мережа

Між вхідним та вихідним шаром розташовуються один або кілька проміжних або прихованих шарів. Прихованими вони називаються тому, що їх входи та виходи невідомі для зовнішніх по відношенню до нейронної мережі програм і користувача.

Для навчання багатошарових нейронних мереж використовують навчання з учителем. Найбільш популярним алгоритмом навчання їм є алгоритм зворотного поширення помилки та її різновиду.

В даний час багатошарова архітектура нейронних мереж є найбільш популярною та добре розробленою. Багатошарова нейронна мережа може моделювати функцію практично будь-якого ступеня складності, причому кількість шарів та число нейронів у кожному шарі визначають складність функції [19].

1.2.3 Рекурсивна нейромережа

Рекурсивна нейронна мережа - тип глибокої нейронної мережі, сформований при застосуванні тих самих наборів терезів рекурсивно над структурою, щоб зробити скалярне чи структуроване передбачення над вхідною структурою змінного розміру через активацію структури в топологічному порядку. У найпростішій архітектурі нелінійність, така як

тангенціальна функція активації, і матриця терезів, що розділяється всією мережею, використовуються для об'єднання вузлів батьківських об'єктів [20].

1.2.4 Рекурентна нейромережа

Рекурентні нейронні мережі – мережі з циклами, які добре підходять для обробки послідовностей на рисунку 1.6.

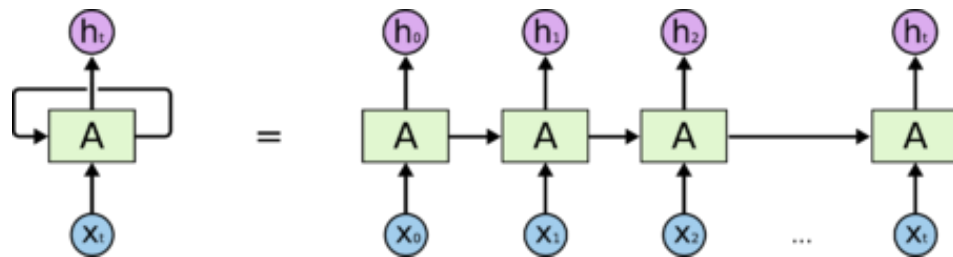


Рисунок 1.6 – Загальна схема рекурентної нейромережі [20]

Навчання RNN походить на навчання звичайної нейромережі. Зазвичай використовують алгоритм зворотного поширення помилки (англ. Backpropagation), та з невеликою модифікацією. Так як, одні й самі змінні, параметри зазвичай використовуються на кожному тимчасовому етапі у навчанні мережі, градієнт на кожному виході залежить не тільки від розрахунків поточного кроку, а й від результату виконань попередніх тимчасових кроків [18]. Якщо наводити приклад то , щоб вичислити градієнт для п'ятого елемента послідовності, нам потрібно було б «поширити помилку» на 4 кроки та підсумовувати градієнти [18]. Такий алгоритм зазвичай називається «алгоритмом зворотного поширення помилки крізь час» [18] - [20].

1.2.5 Inception

Пре-підготовлена неймережа Inception була створена Google .Вона аналізує кожен об'єкт на котрому вона була навчена. Для людини визначити що перед ним знаходиться яблуко дуже легко,але якщо перед людиною знаходиться камінець і потрібно виділити його похідність стає важко, в ході визначення легко помилитися або пропустити щось важливе. Пре-підготовлена неймережа Inception забезпечує точність приблизно 92% проти 72% у порівнянні з людиною на певних датасетах. Неймережа не упустить з уваги кожную деталь і кожен піксель на зображенні, хоча і час від часу виникають помилкові спрацьовування, які пізніше може перевірити експерт в даній області, рисунок 1.7, [18]-[20].

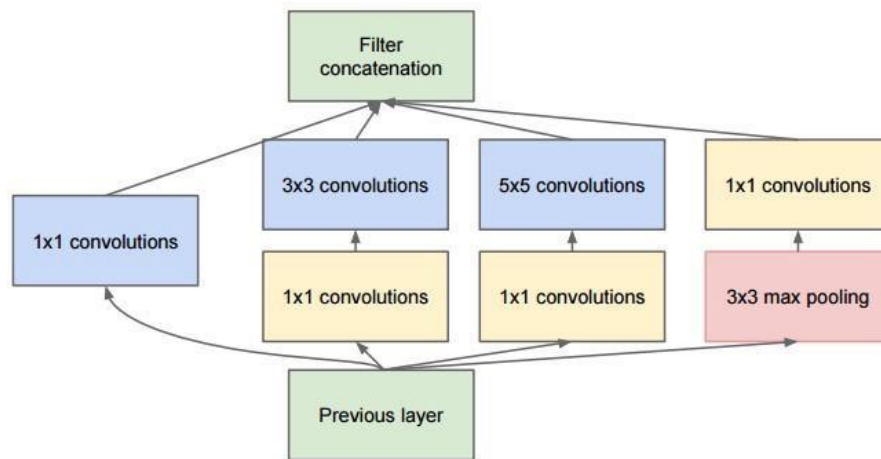


Рисунок 1.7 – Загальна схема Inception [18]-[20]

1.2.6 Нейронна мережа VGG

Нейронні мережі VGG. VGG звертається до іншого дуже важливого аспекту CNN: глибини. Давайте розглянемо архітектуру VGG:

- вхідні дані. VGG отримує зображення RGB розміром 224x224 пікселя. Для конкурсу ImageNet автори вирізали центральний патч розміром 224x224 на кожному зображенні, щоб зберегти розмір вхідного зображення однаковим;

- згорткові шари. Згорткові шари у VGG використовують дуже мале сприйнятливое поле (3x3, найменший можливий розмір, який все ще захоплює вліво/вправо та вгору/вниз). Існують також фільтри згортки 1x1, які діють як лінійне перетворення входу, за яким слідує блок ReLU. Крок згортки фіксується на 1 піксель, щоб просторова роздільна здатність зберігалася після згортки;
- повністю пов'язані шари. VGG має три повністю підключених шари: перші два мають 4096 каналів кожен, а третій має 1000 каналів, по 1 для кожного класу;
- приховані шари. Усі приховані шари VGG використовують ReLU. VGG зазвичай не використовує локальну нормалізацію відповіді (LRN), оскільки LRN збільшує споживання пам'яті та час навчання без особливого збільшення точності.

Невеликі згорткові фільтри дозволяють VGG мати велику кількість вагових шарів; звичайно, більше шарів призводить до покращення продуктивності [18] - [20].

1.2.7 YOLO

Оновлена версія YOLO, state-of-the-art моделі для розпізнавання об'єктів у реальному часі. YOLOv5 видає передбачення зі швидкістю 140 кадрів за секунду. YOLO (You Only Look Once) - це сімейство моделей, які стали популярні завдяки легковажності та якості пророкувань. Такі характеристики дозволяють використовувати YOLO для задач розпізнавання об'єктів у реальному часі та на портативних пристроях.

YOLOv2 та YOLOv3 версії YOLO послідовно покращували якість та швидкість передбачень. У квітні 2020 року представили четверту версію моделі YOLOv4. Поліпшенню якості YOLOv4 сприяли зміни в аугментації даних та архітектурі моделі.

Має швидкий час навчання, гарне розпізнавання на відео, і так само займає хорошу позицію між існуючими НР порівняно з стандартними наборами навчання.

Має середню точність і може використовуватися не у всіх галузях [18] - [20].

1.2.8 Згорткова нейронна мережа (CNN)

Згорткова нейронна мережа – це тип нейронних мереж, які використовують особливості згорткових шарів, та ідеї таких нейронних мереж полягає у чередуванні згорткових шарів та виділяючих шарів (рис. 1.8). Цей тип нейромереж вузько спеціалізована, багатошарова. Для навчання слід почати за стандартних методів, частіше за все – метод зворотнього поширення помилки, а далі продовжувати через відгалуження області використання. Функція активації нейронів – може бути будь-яка, або змінюватись від результатів. Кожний фрагмент зображення згортається (помножується на матрицю згортки поелементно), а результати підсумовуються та записуються в наступну позицію вихідного зображення. Загальну схему згорткової нейромержі можна побачити на рис.1.8 [21]-[22].

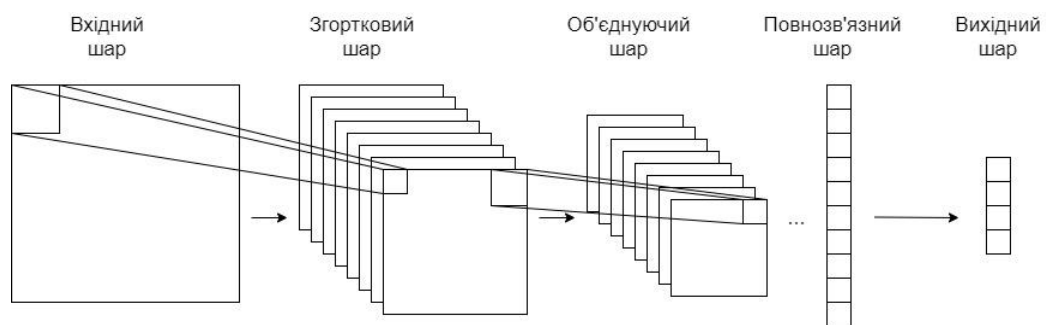


Рисунок 1.8 – Загальна схема згорткової нейромержі

CNN добре використовувати для обробки візуальних і інших n-мірних даних. Якщо провести порівняння з іншими нейромережами або

архітектурами розпізнавання то, CNN показують в більшості часу кращі результати в обробці та розпізнавання зображень, а також мовних застосувань, аніж конкуренти. Вони додатково можуть бути навчені за допомогою звичайного зворотного поширення [19]. Згорткові нейромережі легше навчаються, ніж інші звичайні, глибокі нейронні мережі і мають менше параметрів для оцінки, що робить їх навчання швидше, якісніше та результативніше, що здається дуже привабливими для багатьох розробників [19].

Шар згортки містить окремий фільтр для кожного каналу (ознаки), ядро згортки якого обробляє попередній шар по фрагментам. Вагові коефіцієнти ядра згортки невідомі та встановлюються в ході навчання. Особливістю даного шару є відносно невелика кількість параметрів, що встановлюються при навчанні, на відміну від кількості параметрів повнозв'язної нейронної мережі. Скалярний результат кожної згортки оброблюється функцією активації, якою може бути будь-яка. Традиційно в якості функції активації використовували функцію гіперболічного тангенса або сигмоїди, але у 2000-х роках була запропонована функція активації ReLU (rectified linear unit), яка дозволила значно прискорити процес навчання та спростити обчислення [21]. Саме вона, та її модифікації, є найчастіше використовуваними функціями активації в нейронних мережах [21].

Шар підвиборки є нелінійним ущільненням карти ознак: група пікселів непересічних прямокутників чи квадратів (зазвичай розміру 2×2) ущільнюється до одного пікселя, після нелінійного перетворення, для ущільнення найчастіше використовується функція максимуму. Така операція дозволяє значно зменшити об'єм зображення: якщо ознаки були виявлені на етапі згортки, далі вони вже не потрібні, тому вони ущільнюються. Фільтрація не потрібних деталей допомагає нейронній мережі не перевчитися [22].

Після декількох обробок зображення на шарах згортки та підвибірки, нейронна мережа працює не з сіткою пікселів, а абстрактними картами ознак.

З кожним шаром кількість ознак збільшується, а розмір зображення зменшується. Зрештою, отримується великий набір ознак, що зберігають невеликий об'єм даних. Ці дані об'єднуються та передаються повнозв'язній нейронній мережі, де просторова структура пікселів втрачається [22].

Також, для покращення роботи мережі, підвищення стійкості та запобігання перенавчання застосовують метод виключення (dropout), який в процесі навчання випадковим чином виключає окремі нейрони [22].

Згортова нейронна мережа є розвитком нейронних мереж когнітрону та неокогнітрону та має найкращі показники при застосуванні для розпізнавання образів, завдяки здатності враховувати двовимірну матрицю зображення.

На поточний час використання даної нейронної мережі вважається найкращим рішенням для розпізнавання образів, завдяки точності та швидкості розпізнавання.

1.2.9LSTM

LSTM – це один з варіантів рекурентних нейромереж (у подальшому РНМ), що здатні навчатись основуючись на довготривалих залежностях [22]. Мережі довгої-короткочасної-пам'яті спочатку були запропоновані Хохрайтером і Шмидхубером, а потім вдосконалені багатьма послідовними розробниками і дослідниками. Вони певним чином підходять для об'ємного кола задач і є одним із популярних типів РНМ [22].

У стандартних рекурентних нейромережах для реалізації рекурентності зазвичай використовують комбінацію прихованого шару на попередньому кроці і наступних вхідних даних в шарі з активаційною функцією [23]. В LSTM-мережі рекурентність зроблена аналогічним чином, але $\tanh$ -шарів налічують не один, а цілих чотири, і взаємодіють вони певним чином [23].

LSTM-мережа – це штучна нейронна мережа, що складається з LSTM-модулів [23]. LSTM-модуль – це рекурентний модуль мережі, що здатний запам'ятовувати значення як на короткі, та і на довгі проміжки часу. LSTM-модулі часто групуються в LSTM-блоки [24].

Блоки рекурентної нейромережі вміщують три або чотири так звані вентиля, котрі використовуються для спецефічного контролю потоків інформації на входах та виходах пам'яті даних цих блоків [23]. Дані вентиля реалізовані у вигляді логічних функцій для обчислення перемінх на діапазоні $[0;1]$ [23]. Дане множення на отримане значення потрібно використати для не повного допуску або повної заборони усього потоку інформації всередину і назовні пам'яті [23]. Якщо приводити приклад то можна уявити наступну ситуацію, перший або «вхідний вентиль» оцінює міру входження кожного нового значення до пам'яті, а наступний «вентиль забуття» оцінює міру збереження значення в пам'яті [23]. Кінцевий або «Вихідний вентиль» оцінює міру того, в якій стані знаходиться значення, що розташоване у пам'яті, далі використовується для розрахунку вихідної активаційної функції блоку [23]. Схема роботи мережі LSTM зображена на рисунку 1.9.

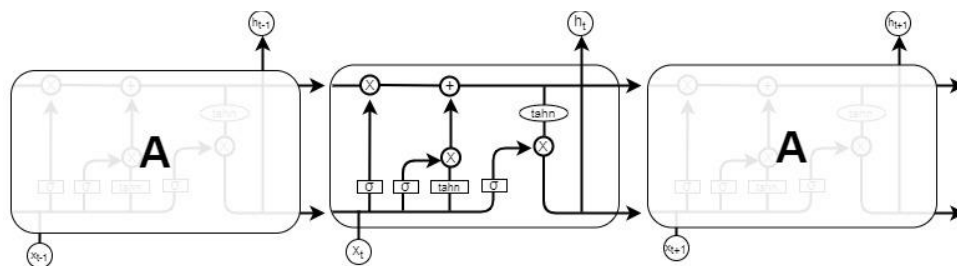


Рисунок 1.9 – Схема роботи LSTM [23]

В блоці нейромережі LSTM усі ваги ініціалізуються та використовуються для оперування вентилями. Вони Зазначені для виразів, які у наступному будуть подаватись на слідуючі блоки (включаючи x_t і вихід з попереднього часового кроку h_{t-1}) на кожному з вентилів. Через це, дані блоки нейромережі вирішує, як оптимально використовувати свою пам'ять

як однією з головних функцій з цими значеннями, а також тренування ваг дозволяє блокам нейромережі навчити функцію, для мінімізації функції втрат. Блоки нейромережі зазвичай навчають з допомогою методу зворотного поширення помилки [24].

1.3 Аналіз функцій активацій

У наших моделях які ми аналізуємо та тестуємо використовуються активаційні функції, які ми описуємо далі. Зважена сума з входів – активаційний сигнал – проходить через функцію активації для виведення даних з нейрона. Є кілька видів функції активації: лінійна, ступінчаста, сигмоїдна, тангенціальна, випрямна (Rectified linear unit, ReLu).

1.3.1 Лінійна функція

Лінійна функція є прямою лінією і пропорційна входу:

$$f(x) = ax$$

Такий вибір активаційної функції дозволяє отримувати спектр значень, а не лише бінарну відповідь [13]. Можна з'єднати кілька нейронів разом і якщо більше одного нейрона активовано, рішення приймається на основі застосування операції max (або softmax) [13].

Похідна від $A = c_x \cdot x$ дорівнює c . Це означає, що градієнт ніяк не пов'язаний з X . Градієнт є постійним вектором, а спуск здійснюється за постійним градієнтом. Якщо виробляється помилкове передбачення, зміни, зроблені зворотним поширенням помилки, теж постійні і залежить від зміни на вході $\text{delta}(x)$ [13].

Це викликає певні труднощі (не завжди, але в більшості випадків). Але є й інша проблема. Розглядаючи варіант з пов'язаними шарами кожен з котрих ,активується лінійною функцією [13]. Значення з цієї функції йде в

наступний шар як вхід, другий шар вважає зважену суму на своїх входах і, у свою чергу, включає нейрони в залежності від іншої лінійної активаційної функції.[13]

Незважаючи на кількість шарів, при своєму стандартному значенні їх лінійності, у фінальному варіанті в останньому шарі функція завжди буде лінійною.

Це означає, що два шари (або шарів N) можуть бути замінені одним шаром. Ми втратили можливість робити набори із шарів [13]. Не важливо, як ми стікаємо, вся нейронна мережа все одно буде подібна до одного шару з лінійною функцією активації (комбінація лінійних функцій лінійним чином - інша лінійна функція) [13].

1.3.2 Ступінчаста функція

Функція приймає значення 1 (активована), коли $Y > 0$ (кордон), і значення 0 (не активована) інакше.

Ми створили активаційну функцію нейрону. Це простий спосіб, проте є недоліки. Розглянемо таку ситуацію:

$$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$$

Розглянувши активаційну формулу нейрону, ми отримали бінарний класифікатор — модель, яка має говорити так чи ні (активований чи ні). Ступінчаста функція виводить 1 або 0.

Тепер змодельюємо випадок, коли потрібно більше нейронів для класифікації багатьох класів: клас1, клас2, клас3 тощо [13]. Усі нейрони з функції активації виведуть 1. У разі виникають питання у тому, який клас повинен у результаті вийти для заданого об'єкта. [13]

Головна мета, щоб активувався тільки один нейрон, а функції активації інших нейронів дорівнювали нулю (тільки в цьому випадку можна бути впевненим, що мережа правильно визначає клас) [13]. Таку мережу

важче навчати та домагатися збіжності. Якщо активаційна функція не бінарна, то можливі значення активовано на 50%, активовано на 20% і так далі [13]. Якщо активовано кілька нейронів, можна знайти нейрон з найбільшим значенням активаційної функції (краще, звичайно, щоб це була softmax функція, а не max. [13]

Але в такому випадку, як і раніше, якщо більше одного нейрона говорять "активований на 100%", проблема залишається [13]. Так як існують проміжні значення на виході нейрона, процес навчання проходить більш гладко і швидко, а ймовірність появи кількох повністю активованих нейронів під час тренування знижується порівняно зі ступінчастою функцією активації (хоча це залежить від того, що ви навчаєте та на яких даних). [13]

Проаналізував все вище зазначене було вирішено отримувати проміжні значення активаційної функції (аналогова функція), а не бінарна функція.

1.3.3 Сігмоїда

Сігмоїда виглядає гладкою і подібна до ступінчастої функції. Розглянемо її переваги:

$$f(x) = \tanh(x) = \frac{2}{1+e^{-x}}.$$

По-перше, сигмоїда — нелінійна за своєю природою, а комбінація таких функцій робить також нелінійну функцію [13]. Тепер ми можемо стікати шари [13].

Ще одна перевага такої функції - вона не бінарна, що робить активацію аналоговою, на відміну від ступінчастої функції. Для сигмоїди також характерний гладкий градієнт. [13]

Якщо ви помітили, в діапазоні значень X від -2 до 2 значення Y дуже швидко змінюється [13]. Це означає, що будь-яке мале зміна значення X у цій

галузі тягне за собою істотне зміна значення Y [13]. Така поведінка функції свідчить про те, що Y має тенденцію притискатися одного з країв кривої [13].

Сігмоїда дійсно виглядає підходящою функцією для класифікаційних завдань. Вона прагне привести значення однієї зі сторін кривої (наприклад, до верхнього при $x=2$ і нижньому при $x=-2$) [13]. Така поведінка дозволяє знаходити чіткі межі під час передбачення [14].

Друга перевага функції полягає у наступному:

- фіксований діапазон значень перемінних функції $[0,1]$ (лінійна функція змінюється в межах $(-\infty, \infty)$) [13] - [14];
- при великих значеннях функції активації не призводить до помилок у разі помилкового спрацьовування [13] - [14].

Сьогодні сигмоїда є однією з найчастіших активаційних функцій у нейромережах [13]. Але й вона має недоліки, на які варто звернути увагу [13].

Ви вже могли помітити, що при наближенні до кінців сигмоїди значення Y мають тенденцію слабо реагувати на зміни в X [13]. Це означає, що градієнт у таких областях набуває маленьких значень [13]. А це, у свою чергу, призводить до проблем із градієнтом зникнення. Докладно розглянемо, що відбувається при наближенні активаційної функції до майже горизонтальної частини кривої на обох сторонах. [13]

У такому разі значення градієнта мало або зникає (не може зробити істотних змін через надзвичайно малого значення) [13]. Нейросеть відмовляється навчатися далі або робить це вкрай повільно (залежно від способу використання або доти, доки градієнт/обчислення не почне страждати від обмежень на значення з плаваючою точкою) [13]. Існують варіанти роботи над цими проблемами, а сигмоїда все ще дуже популярна для класифікаційних завдань [13].

1.3.4 Функція гіперболічного тангенсу

Гіперболічний тангенс дуже схожий на сигмоїду [13]. Це скоригована сигмоїдна функція. Тому така функція має самі характеристики, як і в сигмоїди, розглянутої раніше. Розглянемо її:

$$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1.$$

Функція гіперболічного тангенсу нелінійна, вона підходить для комбінації шарів, у діапазоні значень функції – (-1, 1) та активаційна функція не перевантажується від величезних значень [12].

1.3.5 Функція ReLu

Користуючись визначенням, стає зрозуміло, що ReLu повертає значення x , якщо x позитивно, і 0 інакше [13]. Розглянемо її:

$$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$$

На перший погляд здається, що ReLu має ті самі проблеми, що й лінійна функція, оскільки ReLu лінійна в першому квадранті [14]. Але насправді ReLu нелінійна за своєю природою, а комбінація ReLu також нелінійна! (Насправді, така функція є хорошим апроксиматором, оскільки будь-яка функція може бути апроксимована комбінацією ReLu) [14]. Це означає, що ми можемо стікати шари. Область допустимих значень ReLu - $[0, \infty)$, тобто активація може "вибухнути" [14].

Наступний пункт – розрідженість активації. Уявимо велику нейронну мережу з безліччю нейронів [14]. Використання сигмоїди або гіперболічного тангенсу спричинятиме активацію всіх нейронів аналоговим способом [14]. Це означає, що майже всі активації мають бути оброблені для опису виходу мережі [14]. Інакше кажучи, активація щільна, але це затратно.

В ідеалі ми хочемо, щоб деякі нейрони не були активовані, це зробило б активації розрідженими та ефективними.

ReLU дозволяє це зробити. Уявімо мережу з випадково ініціалізованими вагами (або нормалізованими), в якій приблизно 50% активацій дорівнюють 0 через характеристики ReLU (повертає 0 для негативних значень x) [14]. У такій мережі включається менша кількість нейронів (розріджена активація), а мережа стає легше [14]. Відмінно здається, що ReLU підходить нам за всіма параметрами [14]. Але ніщо не ідеальне, зокрема і ReLU.

Через те, що частина ReLU є горизонтальною лінією (для негативних значень X), градієнт на цій частині дорівнює 0 [14]. Через рівність нулю градієнта, ваги не будуть коригуватися під час спуску [14]. Це означає, що нейрони, що знаходяться в такому стані, не будуть реагувати на зміни в помилці/вхідних даних (просто тому, що градієнт дорівнює нулю, нічого не змінюватиметься) [14]. Таке явище називається проблемою вмираючого ReLU (Dying ReLU problem) [14]. Через цю проблему деякі нейрони просто вимикаються і не відповідатимуть, роблячи значну частину пасивної нейромережі [14]. Однак є варіації ReLU, які допомагають цю проблему уникнути [14]. Наприклад, є сенс замінити горизонтальну частину функції на лінійну [14]. Якщо вираз лінійної функції задається виразом $y = 0.01x$ для області $x < 0$, лінія злегка відхиляється від горизонтального положення [14]. Існують інші способи уникнути нульового градієнта [14]. Основна ідея тут – зробити градієнт нерівним нулю та поступово відновлювати його під час тренування [14].

ReLU менш вимогливо до обчислювальних ресурсів, ніж гіперболічний тангенс або сигмоїда, оскільки робить простіші математичні операції [14]. Тому є сенс використовувати ReLU при створенні глибоких нейронних мереж [14].

1.4 Аналіз методів для розпізнавання

Через безліч мінімальних відмінностей між препаратами НС не відрізняють їх між собою, що робить дане завдання досить складним, так само посилюють дане завдання моменти з технічним обладнанням, а саме:

- погані камери (Нечітке зображення з камер, що дає зайві перешкоди та шуми);
- погане освітлення;
- невірний, не зафіксований ракурс;
- старе устаткування робочої машини яке уповільнює навчання, а й робить його продуктивним.

У сукупності дані фактори і до них відносяться дають відчутну похибку в розпізнаванні образу об'єкта (упаковки, ампули, флакони), а за рахунок того, що в НС існує поняття перенавчання, то шахи на успіх падають до при нульових значень які існують на даний момент дають величезну похибку в ній даної задачі. В ході роботи експериментальним шляхом було з'ясовано що для даного завдання підходить згортова нейронна мережа, а також варіації поданих моделей для подальшої роботи, що прискорило робочий процес за рахунок вже пройдених етапів проб та помилок. Надалі розглядатимемо проведену нами модернізацію мереж.

У роботі запропоновано модифікацію нейромережі, яка використовує базову архітектуру згортових нейромереж, а також деякі стандарти (активаційні функції та глибокі кулі) з моделі VGG16.

Для вирішення поставленого завдання було прийнято рішення розробити модифікацію згортової мережі, що дозволить підвищити точність нейронної мережі. Завдання роботи зводиться до налаштування архітектури нейромережі та таких параметрів як: кількість нейронів, швидкість завантаження даних та кількість епох. Дане припущення впливає з того, що на початкових стадіях роботи мережі немає жодної інформації про вхідні параметри для налаштування вагових параметрів нейронної мережі для

певної задачі. У розробленій модифікації проведено зміну архітектури самої мережі

Принципи вони формулюють наступні:

- велика кількість параметрів, які близькі один до одного в просторі (тобто в сусідніх "пікселях"), і цим можна користуватися, щоб робити згортковий шар (convolution) меншого розміру. Через те, що сусідні сигнали часто скорелювані, то можна зменшити розмірність перед згортковим шаром (convolution) без втрати інформації;
- для ефективного використання комп'ютерних ресурсів потрібно збільшувати і ширину, і глибину мережі;
- "широкі" кулі швидше навчаються, що особливо важливо на високих рівнях шарів (але локально, тобто цілком можна після них зменшувати розмірність).

1.5 Аналіз галузі штучного інтелекту для розпізнавання образів

Для автоматизації процесу сортування тільки прибулих ліків на аптечне відділення було прийнято рішення про створення робота-сортувальника, який на основі модифікованою НС буде, після аналізу упаковки за назвою препарату та дозуванням буде видавати точну назву та дозування, що скоротить час заповнення накладної.

Проблеми розпізнавання образу полягають у упаковках близнюках які різняться трохи друг від друга. Це можуть бути як різні препарати, так і упаковки одного препарату, але з різним дозуванням.

Для порівняння точності прогнозування було проаналізовано декілька різних методів, опис та порівняння яких наведено далі [16].

В якості даних використано фотографії препаратів з різних сторін. Набір на кожний екземпляр нараховує 36. Порівняння нейромереж зображено в таблиці 1.1.

Таблиця 1.1 – Порівняння неймереж

Назва	Час навчання, ч	Точність розрахункова, %	Точність практична, %	Швидкість праці на робочому місці, с
Віола - Джонсон	-	63	53	7
CNN	6.3	87	80	7
Inceptionv2	12.8	94	84	6
RNN	7	65	63	5
LSTM	8	63	54	8
YOLO5	12	89	67	5
VGG16	10	92	73	6
Inceptionv3	21.1	96	90	7
VGG19	15.5	95	89	6

1.6 Постановка завдання

Мета роботи – дослідження, модифікація та програмна реалізація методів розпізнавання медичних упаковок.

Для автоматизації процесу сортування тільки прибулих лікарів на аптечне відділення було прийнято рішення про створення робота-сортирувальника, який на основі згорткової буде після аналізу упаковки видавати точну назву, що скоротить час заповнення накладної.

Проблеми завдання розпізнавання образу полягають у упаковках близнюках які різняться трохи друг від друга. Це можуть бути різні препарати (рис 1.10 та рис. 1.11).



Рисунок 1.10 – Медична упаковка Гніпрал

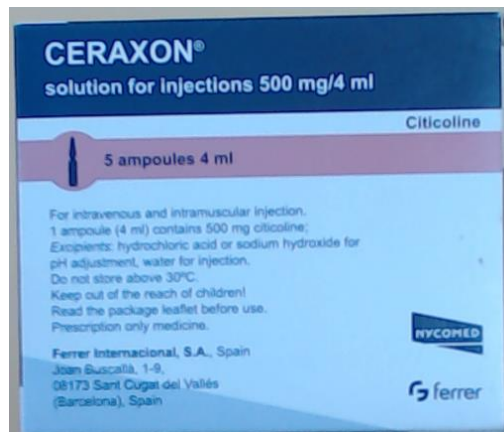


Рисунок 1.11 – Медична упаковка Цераксон

Так і упаковки одного препарату але з різним дозуванням (рис. 1.12 та рис. 1.13).



Рисунок 1.12 – Медична упаковка Афлубін (20 мл.)



Рисунок 1.13 – Медична упаковка Афлубін (50 мл.)

Через безліч мінімальних відмінностей між препаратами не відрізняють їх між собою, що робить дане завдання досить складним, так само посилюють дане завдання моменти з технічним обладнанням, а саме:

- погані камери (нечітке зображення з камер, що дає зайві перешкоди та шуми);
- погане освітлення;

- невірний, не зафіксований ракурс;
- старе устаткування робочої машини яке уповільнює навчання, а й робить його продуктивним.

У сукупності дані фактори і до них відносяться дають відчутну похибку в розпізнаванні образу об'єкта (упаковки, ампули, флакони), а за рахунок того, що в НС існує поняття перенавчання, то шайи на успіх падають до при нульових значень. Через це модель залежності якості передбачення від якості набору даних можна уявити в вигляді наступної формули (1.9):

$$A_{fin} = f(x_{light}, x_{distance}, x_{quality}, x_{foreshortening}, x_{quantity}, x_{focus}, x_{background}), (1.9)$$

де A_{fin} — показник якості передбачення нейромережі;

x_{light} — вплив світла;

$x_{distance}$ — вплив дистанції до об'єкту;

$x_{quality}$ — вплив якості фотографії;

$x_{foreshortening}$ — вплив ракурса з якого було зроблено фото;

$x_{quantity}$ — вплив кількості фотографій;

x_{focus} — вплив фокусування на об'єкті;

$x_{background}$ — вплив заднього фону.

В ході роботи експериментальним шляхом було з'ясовано що для даного завдання підходить згорткова нейронна мережа, а також варіації поданих моделей для подальшої роботи, що прискорило робочий процес за рахунок вже пройдених етапів проб та помилок. Надалі розглядатимемо проведену нами модернізацію мереж.

1.7 Висновки за розділом 1

Проведено детальне дослідження предметної області, визначені основні поняття і терміни, пов'язані с предметною областю і об'єктом розробки. Проведений аналіз проблем розпізнавання образів. Розглянуті основні нейромережі для праці з розпізнаванням образів. Були розглянуті основні активаційні функції для дослідження та використання у нейромережах. Було сформульовано і поставлено завдання до роботи.

2 ВИБІР ІТРУМЕНТІВ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Вибір інструментальних засобів розробки

Дипломна кваліфікаційна робота магістра виконана використовуючи мову програмування Python у парі з фреймворками які полегшують створення та візуалізацію нейромереж. Вибір був зроблений завдяки багатьом перевагам як самої мови так і доступних та зрозумілих у використанні фреймворків. Ці інструменти вважаються найбільш легкими та популярними у світі Deep Learning, розпізнавання образів, текстів тощо, що в свою чергу виробляє появу великої кількості готових рішень. Python – високорівнева мова програмування, орієнтована на підвищення продуктивності розробника за рахунок легкої та зрозумілої структури і читання коду [21]. Стандартні бібліотеки включають значний обсяг корисних функцій які можна використати однією функцією. Python використовує наступні парадигми програмування:

- Структурне [21];
- Функціональне [21];
- об'єктно – орієнтоване [21];
- імперативне [21];
- аспектноорієнтоване [21].

До основних властивостей мови належить наступні параметри:

- динамічна типізація [21];
- автоматичне звільнення пам'яті [21];
- зручні стандартні типи даних [21];
- обробка виключень [21].

Код в Python зазвичай збирається у функції та класи, які потім можна об'єднати в модулі, які в свою чергу вже при бажанні об'єднуються в пакети [24].

Однією з переваг цієї мови програмування є читабельність і лояльність до початківців. Вона полягає в легкості читання, ясності і більш високій якості, що відрізняють Python від інших інструментів в світі мов сценаріїв [21]. Це означає, що багаторазове його використання і обслуговування виконується набагато простіше, ніж використання програмного коду на інших мовах сценаріїв [21]. З вище описаного випливає ще один вагомий плюс Python – висока швидкість розробки [21].

У порівнянні з компілюючими або строго типізованими мовами, такими як C, C++ і Java, Python за рахунок не навантаженої сфери та алгоритмів, підвищує продуктивність розробника [21]. За загальним обсягом програмний код на мові Python часто половину або менше від того самого програмного коду на мові C ++ або Java [21]. Це означає швидший час написання коду, менша кількість часу на дебагінг і менший обсяг затрат програміста на підтримку та оптимізацію продукту. [22]

Окрім того, програми, створені на мові Python, з консолі, минаючи тривалі етапи компіляції, необхідні в деяких інших мовах програмування [21]. Велика частина програм на мові Python виконується без змін на всіх основних платформах [21]. Для перенесення файлів та коду з операційної системи Linux в Windows не рідко полягає в простому копіюванні файлів програм з однієї машини на іншу, потрібно лише встановити в систему один із інтерпретаторів Python.[24]

Окрім того, Python надає масу можливостей по створенню переносних інтерфейсів для глядачів, програм доступу до баз даних, веб– додатків і багатьох інших важливих програм [21]. Це робить його зручним і гнучким інструментом для розробки кросплатформних програм [21]. У складі Python поставляється велика кількість зібраних і переносних функціональних можливостей, відомих як стандартна бібліотека, а також безліч інших корисних бібліотек[24].

Стандартна бібліотека надає безліч можливостей для програміста, потрібних в створенні будь-яких програмах, вона дає програмісту змогу

виконувати пошук починаючи від звичайного тексту у шаблоні і закінчуючи нейромережевими функціями [21]. Python не забороняє розширювати свій функціонал як за рахунок ваших самостійно написаних бібліотек, так і за рахунок переписання старих, які були створені сторонніми розробниками [21]. Наприклад, розширення NumPy, що часто використовується розробниками, позиціонується як вільний і більш потужний еквівалент системи програмування математичних обчислень Matlab.[24]

Коди на Python легко будуть інтегруватись та взаємодіяти з іншими частинами програми написаними на інших мовах програмування [21]. Ця інтеграція дозволяє використовувати Python для налаштування і розширення функціональних можливостей програмних продуктів [21]. Також не рідко бувають ситуації коли різні частини коду потрібно робити за допомогою двох різних мов програмування [21]. Під час аналізу мов програмування було створено порівняльну характеристику середовищ розробки, яка наведена у табл. 2.1. [25]

Таблиця 2.1 - Порівняльна характеристика середовищ розробки

Критерії порівняння	C++	Java	Python
1	2	3	4
Статична типізація	+	+	-
Динамічна типізація	-	-	+
Явна типізація	+	+	+/-
Неявна типізація	+/-	-	+
Неявне приведення типів без втрати даних	+	-	-
Неявне приведення типів із втратою даних	+	-	-
Неявне приведення типів у неоднозначних ситуаціях	+	-	-
Аліаси типів	+	-	x
Висновок типів змінних з ініціалізатора	+/-	-	x
Висновок типів змінних з використання	+/-	-	x

Продовження таблиці 2.1

1	2	3	4
Висновок типів-аргументів при виклику методу	+	+	x
Кросплатформеність	+	+	+
Наявність автодоповнення	+	+	—
Наявність автоматичного закриття дужок, бракетів	+	+	—
Підтримка репозиторіїв	+	+	—
Наявність «plugin manager»	+	+	+
Простота встановлення	+	+	+
Підтримка аналізу коду	+	+	+
Підтримка рефакторингу коду	+	+	+
Наявність інтерактивної консолі	+	+	+
Зручний користувацький інтерфейс	++	+	+
Швидкість модернізації	++	+	+

2.2 Середовище розробки Jupyter Notebooks

Jupyter-ноутбук - це середовище розробки, де можна бачити результат виконання коду та змінних не перезапускаючи увесь код. Також можна розсортувати код по блоках та запускати тільки певні блоки. Наприклад ви можете написати шматочок коду на серветці та сказати серветці: «Починай роботу» [25].

У такому середовищі розробки можна, наприклад, записати додаткову або модернізовану функцію і перевірити як вона функціонує, без запуску програми усього коду. Також можна змінити порядок виконання коду, що є важливим додатком. Можна окремо завантажити файл у пам'ять, окремо перевірити вміст, окремо обробити вміст.

А ще в jupyter-ноутбуках є висновок результату одразу після фрагмента коду. Наприклад, можна прямо в середині коду побачити побудований графік, отримати попередні цифри чи іншу візуалізацію.[25]

Найчастіше, коли говорять про jupyter-ноутбуки, мають на увазі роботу з Python. Також в додатку можна використовувати інші інструменти:

- Perl;
- bash-скрипти;

- Ruby;
- R.

Для цього використовують спецефічні команди, або магічні, які так і називаються – `magic`-команди. Вони дозволяють запускати код іншими мовами, а ще істотно розширюють можливості звичайного Python [25].

Jupyter-ноутбуки запускаються на комп'ютері або у хмарі. Найпростіше почати працювати з ноутбуком використовуючи інший інструмент у хмарі наприклад Google Colab. Тобто вам потрібно запустити браузер та відкрити необхідну веб-сторінку. Далі колаб виділить користувачу ресурс і дозволить виконувати код.

Плюс підходу в тому, що вам не потрібно нічого встановлювати на комп'ютер - хмара все бере на себе, а просто пишете і запускаєте код. Мінус – програма може працювати не так швидко, як на локальній машині. Якщо вам потрібні якісь екзотичні бібліотеки, у хмарі їх може не виявитися. [26].

2.3 Фреймворк Keras

Keras – це найпопулярніший та найбільш гнучкий API для неймереж будь-якого рівня, описаний та доданий у Python і здатний працювати у парі з також потрібними бібліотеками TensorFlow, CNTK або Theano [26]. Він був розроблений для швидкого запуску з експериментуванням [26]. Уміння переходити від ідеї до результату з найменшою можливою затримкою є ключовим для хорошого дослідження [26].

Спочатку Keras ідейно слугував задля дослідників щоб дати їм можливість швидко проводити аналіз та експерименти [26]. Keras володіє наступними ключовими характеристиками:

- дозволяє виконувати один і той же код на CPU або GPU [26];
- має дружній API, що спрощує розробку прототипів моделей глибокого навчання [26];

- включає в себе вбудовану підтримку згортальних мереж (для розпізнавання образів), рекурентних мереж (для обробки послідовностей) і всіляких їх комбінацій [26];
- включає в себе підтримку довільних мережевих архітектур: моделей з множинними входами або виходами, спільне використання шарів, спільне використання моделей і т. д. [26].

Це означає, що Keras підходить для створення практичних моделей будь-якого рівня для звичайного та глибокого навчання. Бібліотека Keras поширюється безкоштовно і може використовуватись в комерційних проектах. Ця бібліотека сумісна з усіма версіями Python, починаючи від 2.7 до 3.8.

Також, Keras є бібліотекою високого рівня моделі, яка надає високорівневі блоки для розробки та аналізу моделей глибокого навчання [26]. Вона не реалізує низькорівневі операції, такі як маніпуляції з тензорами і диференціювання, – для цього використовується спеціалізована і оптимізована бібліотека підтримки тензорів [26]. У додатку до цього Keras не покладається на певну бібліотеку для підтримки тензорів, а використовує більш глобальний, а саме модульний підхід, тобто до фреймворку Keras можна у парі під'єднати декілька різних бібліотек [26].

В даний час підтримуються три такі бібліотеки які можуть, та повинні працювати у парі, а саме: TensorFlow, Theano і Microsoft Cognitive Toolkit (CNTK) [26]. Будь-який метож, який використовує Keras, можна запускати з будь-якої з цих бібліотек без необхідності змінювати щось в коді, але потрібно вказувати маршрут та бібліотеку з якої обарете функції: можна швидко перемикатися між ними в процесі розробки, що часто допомагає програмуїту, наприклад, якщо одна з бібліотек показує більш високу продуктивність при вирішенні даної конкретної завдання то можна використовувати певно її [26].

2.4 Фреймворк для роботи з масивами NumPy

NumPy – це фреймворк (бібліотека) для мови програмування Python, що забезпечує підтримку складних, багатовимірних масивів і матриць. з великою колекцією математичних функцій [25]. Засновник те пероопрохідець для NumPy був Numeric, відпочатку він був створений Джимом Хугунінім за участі кількох інших розробників [25]. У 2005 році Травіс Оліфонт створив NumPy, включивши в додаток для усіх функції, функцію Numarray з числовими змінними та певними модифікаціями. NumPy – це бібліотека для Python з відкритим кодом та має велику аудиторію для продовження користування. NumPy націлений на реалізацію програмного продукту для рішень Python, яка є оптимізатором інтерфейсу байт-коду [25]. Основною функціональністю NumPy слугує її найбільш важлива функція "ndarray" – структура даних для n-мірних масивів. Такі масиви можуть також передавати, користуватись та змінювати буфери пам'яті, що виділяються за допомогою певних розширень, таким як наприклад C / C++, Cython та Fortran в інтерпретатор CPython без необхідності додатково копіювати дані, та надаючи ступінь сумісності з існуючими бібліотеками чисел [25]. NumPy має вбудовану підтримку для ndarrays, відображених на пам'ять [25].

2.5 Фреймворк TensorFlow

TensorFlow – це бібліотека для побудови нейронних мереж, які навчаються вирішувати завдання шляхом нейронавчання на основі праці справжнього мозку та обробляє дані з певною швидкістю на різних рівнях (вузлах), що допомагає знаходити вірний результат [25]. Відкритий та доступний код бібліотеки машинного навчання TensorFlow, в Google поліпшили процес побудови, навчання, а також розгортання складних нейронних мереж [25].

Бібліотеки TensorFlow помітно спрощенні для вбудовування в додатки певних самонавчальних елементів ,а також функцій штучного інтелекту, призначених для розпізнавання мови, організації комп'ютерного зору або обробки природної мови. [26]

2.6 Бібліотека Pandas

Бібліотека Pandas є незамінним помічником Python. Він тягне на собі аналіз даних, машинне навчання та наукові обчислення, а також суттєво полегшує обробку векторів та матриць. Деякі провідні пакети Python використовують Pandas як основний елемент своєї інфраструктури. До них відносяться scikit-learn, SciPy і tensorflow. Крім можливості розібрати по кісточках усі нумерні дані, вміння працювати з Pandas дає значну перевагу при налагодженні складніших сценаріїв бібліотек [26].

2.7 Бібліотека Matplotlib

Matplotlib — це повна бібліотека для створення статичних, анімованих та інтерактивних візуалізацій на Python. Matplotlib робить двовимірною (2D) та тривимірною (3D) графікою:

- створює графіки якості публікації;
- робить інтерактивні фігури, які можна масштабувати, панорамувати, оновлювати (контурні графіки, кругові діаграми, спектральні діаграми);
- налаштує візуальний стиль та макети.
- може експортувати в багато форматів файлів;
- може вбудовувати в JupyterLab графічні інтерфейси користувача.

2.8 Висновки за розділом

В другому розділі було розглянуто різні підходи до реалізації розпізнавання образів та їх особливості, було проведено аналіз та вибір мови програмування для реалізації програмного комплексу.

Також, було обрано середовище розробки і бібліотеки для розв'язання завдання проекту.

Були проаналізовані наявні бібліотеки, та її функціональність, і практична значимість у проектах з ідеєю використання нейронних мереж.

Були розглянуті як бібліотеки для першого аналізу даних, так і для побудови нейромереж, а також для візуалізації графіків навчання та графіків залежності параметрів один для одного.

3 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

3.1 Структура програмного забезпечення

Логічна структура програмного забезпечення висловлена його функціональним призначенням, через це на етапі проєктування було обрано рішення щодо об'єднання певних функціональних компонентів системи до логічних модулів, котрі в свою чергу поділені на функції та реалізують окремі функціональні можливості програм. Структура розробленого продукту зображена на рис.3.1. Потрібно зазначити, що подібна логічна організація програми дозволяє прискорити розроблення та налагодження програмного продукту, а в майбутньому розширювати функціональні можливості готового проєкту шляхом внесення змін або додавання нових модулів

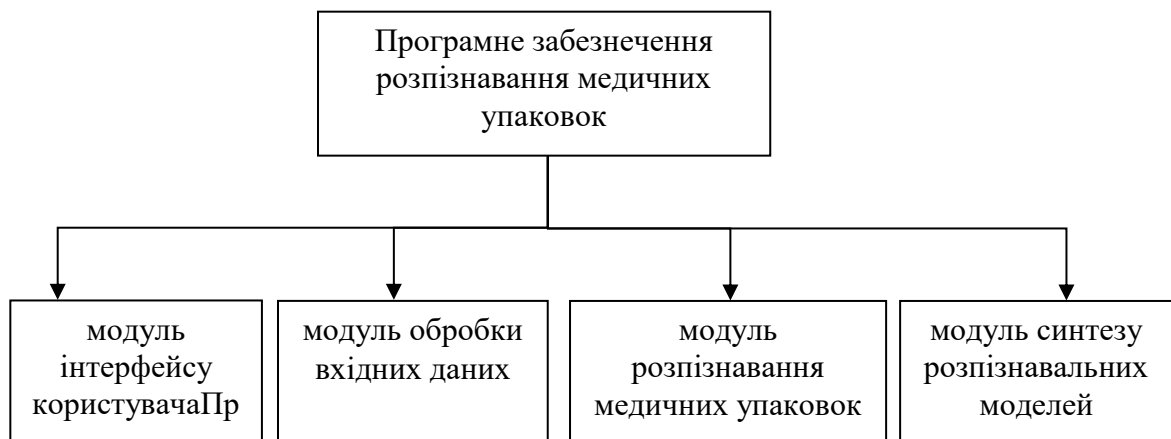


Рисунок 3.1 – Структурна схема програми

Модуль інтерфейсу користувача призначений для виконання наступних функцій:

- забезпечення зручного та інтуїтивно зрозумілого консольного інтерфейсу з користувачем системи;
- забезпечення внутрішньої взаємодії між іншими функціональними блоками системи.

Модуль обробки вхідних даних було призначено для роботи наступних функцій:

- Створення вибірки;
- Збереження вибірки даних
- Завантаження існуючої вибірки даних
- Перевірка коректності введених користувачем даних.

Модуль розпізнавання медичних упаковок призначений для розв'язання завдань розпізнавання медичних упаковок на основі еволюційного підходу. В даному модулі реалізовано різні варіанти еволюційних операторів ініціалізацій, відбору, схрещування та мутації.

3.2 Модифікація згорткової нейромережі

У згортковій НМ була змінена архітектура для детальнішого розбору фотографії та виділення унікальних точок на кожній фотографії, а особливо на близнюках. Були проведені також експерименти з аугментацією зображень і обраний штучний спосіб створення нових зображень для датасетів. Хорошим доповненням послужив використаний ГА який впровадив доповнення до вже зміненої архітектури інші нововведення, додаткові шари, інша послідовність - кількість нейронів для розпізнавання.

Для розробки та тестування моделей було використано власний датасет.

Датасет був зроблений на базі фотографій медичних упаковок. Але в умовах праці такого обсягу буде мало для навчання, через це була зроблена аугментація (штучне розширення датасету) і ми отримали 300 фотографій для унікального пакета, загальна кількість більше 10 тисяч упаковок. Також щомісяця з'являється приблизно 100 нових упаковок.

Даний датасет був поділений на чотири інші датамережі: один для тренування, другий для перевірки точності нейромереж у кінці кожної епохи,

третій для кінцевої перевірки після навчання нейромереж, а останній для практичної перевірки.

Важливим вимогою до даних для моделювання є їх кількість та якість. Якщо дані містять зайву зашумленість, не чітку послідовність та невірний формат – це негативно впливає на точність прогнозів, працю та якість моделей [28].

Також дані, призначені для використання в якості навчальних датасетів для класичних та еволюційних методів, повинні бути нормалізовані та приведені до одного типу, для зменшення помилки та покращення якості тренування.

Обробка початкових даних перед поданням моделі була пророблена в наступній послідовності:

- надані лейблі, номери класів усім фотографіям для розподілення на два класи, знімки з упаковками близнюками, та без близнюків;
- приведення даних до типів, що використовуватимуться в розрахунках.

Завдання, що вирішується нашою нейромережею, – класифікація зображень, унікальних медичних упаковок. Накладені обмеження на мережу - це швидкість відповіді - не більше 5 секунд і точність розпізнавання не менше 92% (без урахування високої точності під час перенавчання).

У якості вхідних даних для нейромережі слугують зображення з форматом JPEG, з розмірністю 150x150 пікселів. Превищення цього розміру підвищить обчислювальну складність і через це обмеження на швидкість відповіді будуть порушені, визначення розміру цього завдання вирішується методом підбору. В іншому випадку малого розміру, нейромережа не зможе виявляти важливі ключові ознаки упаковки.

Вхідні дані кожного конкретного значення пікселя нормалізуються в діапазон від 0 до 1 за формулою (3.1):

$$f(p, min, max) = \frac{p-min}{max-min}, \quad (3.1)$$

де f -функція нормалізації;

p – значення конкретного кольору пікселя від 0 до 255 [27];

min - мінімальне значення пікселя – 0 [27];

max - максимальне значення пікселя – 255 [27].

3.2.1 Архітектура

У таблиці 3.1 можна побачити архітектуру модифікованої CNN (MBDC - medical box detection system). Модифікація полягає у зміні архітектурі НМ: додані нові шари, у повнозв'язних шарах модифікована структура за допомогою додавання додаткових шарів згортки, збільшено кількість повнозв'язних шарів порівняно з оригіналом, використані ГА для оптимізації. У стандартних повнозв'язкових шарах використовують 1 або 2 згорткових шари, активаційну функцію та згортку (max_pooling) у кінці повнозв'язного шару. У модифікованому методі було запропоновано змінити архітектуру повнозв'язкових шарів від 2 до 4 згорткових шарів, при більше 4 згорткових шарів у повнозв'язному шарі час навчання збільшується, а точність падає. Так само було додано необхідну кількість нейронів для кожного шару у всіх повнозв'язних шарах, що було з'ясовано експериментальним шляхом. Було додано шар для розподілу параметрів. Від стандартного застосування наш шар має два шари активації, активаційну функцію. Також розглядається подальше використання технології у вирішенні завдання розпізнавання образів медичних упаковок.

Таблиця 3.1 – Архітектура MBDC

Model: "MBDC"		
Layer (type)	Output Shape	Param #
Вхідний шар		
conv2d (Conv2D)	(None, 222, 222, 64)	1792
conv2d (Conv2D)	(None, 220, 220, 64)	36928
activation (Activation)	(None, 220, 220, 64)	0
max_pooling2d (MaxPooling)	(None, 73, 73, 64)	0
Повнозв'язковий слой 1		
conv2d (Conv2D)	(None, 71, 71, 128)	73856
conv2d (Conv2D)	(None, 69, 69, 128)	147584
activation (Activation)	(None, 69, 69, 128)	0
max_pooling2d (MaxPooling)	(None, 23, 23, 128)	0
Повнозв'язковий слой 2		
conv2d (Conv2D)	(None, 21, 21, 256)	295168
conv2d (Conv2D)	(None, 19, 19, 256)	590080
conv2d (Conv2D)	(None, 17, 17, 256)	590080
conv2d (Conv2D)	(None, 15, 15, 256)	590080
activation (Activation)	(None, 15, 15, 256)	0
max_pooling2d (MaxPooling)	(None, 5, 5, 256)	0
Повнозв'язковий слой 3		
conv2d (Conv2D)	(None, 5, 5, 512)	1180160
conv2d (Conv2D)	(None, 5, 5, 512)	2359808
conv2d (Conv2D)	(None, 5, 5, 512)	2359808
conv2d (Conv2D)	(None, 5, 5, 512)	2359808
activation (Activation)	(None, 5, 5, 512)	0
max_pooling2d (MaxPooling)	(None, 2, 2, 512)	0
Повнозв'язковий слой 4		
conv2d (Conv2D)	(None, 2, 2, 512)	2359808

Продовження таблиці 3.1

Model: "MBDC"		
conv2d (Conv2D)	(None, 2, 2, 512)	2359808
conv2d (Conv2D)	(None, 2, 2, 512)	2359808
conv2d (Conv2D)	(None, 2, 2, 512)	2359808
activation (Activation)	(None, 2, 2, 512)	0
max_pooling2d (MaxPooling)	(None, 1, 1, 512)	0
Шар для розподілу параметрів, який не є згортковим.		
dense (Dense)	(None, 1, 1, 4096)	2101248
flatten (Flatten)	(None, 4096)	0
dense (Dense)	(None, 1000)	4097000
activation (Activation)	(None, 1000)	0
Шар «струсу» нейронів для виходу з локального мінімуму		
dropout (Dropout)	(None, 1000)	0
Вихідний шар		
dense (Dense)	(None, 215)	215215
activation (Activation)	(None, 215)	0
=====		
Total params: 26,437,847		
Trainable params: 26,437,847		

У таблиці 3.2 наведено шари архітектури MBDC з таблиці 3.1.

Згідно даної таблиці ми можемо побачити результат налаштування нейромережі. А також можна побачити усі згорткові і додаткові шари які були використання у розробці нейромережі. Як додаток можна звернути увагу на кількість параметрів на кожному шарі що буде свідчити про достатню, або багату кількість параметрів для навчання. Для реалізації нейромережі та

оптимізації було підключено генетичний алгоритм.

Таблиця 3.2 - Шари архітектури MBDC

Назва	Опис
activation	Шар з активаційною функцією
max_pooling2d	Шар збільшеної згортки
dense	Звичайний шар
flatten	Шар після сигналу якого виконується розрахунок вірогідності приналежності до класу
dropout	Шар котрий викидає певну долю нейронів на кінцевому шарі з вірогідністю від 20% до 40%.

3.2.2 Модифікований генетичний алгоритм

У роботі використано модифікований генетичний алгоритм, який як значення генів використовує дійсні, а не двійкові числа. Це пов'язано з тим, що гени служать кількістю нейронів, які будуть подаватися на вхід до шарів нейромережі. опишемо пізніше. У модифікованому ГА подається 5 генів (табл. 3.3).

Таблиця 3.3 – Гени у модифікованому ГА

P(X,Y)	X ₁	X ₂	X ₃
-----	Y ₁	Y ₂	Y ₃
	Z ₁	Z ₂	Z ₃
	K ₁	K ₂	K ₃
	A ₁	A ₂	A ₃

Опис параметрів з таблиці 3.3.

Де, P – хромосома;

X_n – ген котрий відповідає за параметр глибини нейромережі;

Y_n – ген котрий відповідає за кількість нейронів;

Z_n – ген котрий відповідає за кількість епох;

K_n – ген котрий відповідає за швидкість завантаження даних ;

A_n – ген котрий відповідає за кількість повнозв'язних шарів;

n - популяція.

Перший ген відповідає за кількість шарів в повнозв'язних шарах, а другий відповідає за кількість нейронів у шарі, при умовах що попередній шар має меншу або таку саму кількість нейронів яка не дорівнює 0, а також крок якого буде в два рази більше за попередній $N_n = N_{n-1} * 2$, $N_n \geq N_{n-1}$, де N - кількість нейронів у шарі.

Третій ген відповідає за кількість епох, скільки буде навчатись наш ген.

A_n четвертий ген відповідає за швидкість завантаження даних у, це важливо бо, якщо зміниться конфігурація на комп'ютері на якому навчається, зміниться і потужність і час навчання.

Останій п'ятий ген відповідає за кількість повнозв'язних шарів.

Для схрещування було вибрано метод Parallel GA.

Алгоритми для паралельного виконання генетичних операцій та відбору нових особин. Наприклад: Паралельний відбір у турнірі: ми запускаємо $N/2$ процесів, і кожен процес буде випадковим чином вибирати 4 особи в популяції, виконувати 2 обчислення та визначати переможців. Визволені особи будуть зареєстровані в новому поколінні. Отже, за робочий цикл процесу буде замінено все покоління. Для селекції хромосом, яка полягає у виборі на основі функції пристосованості кращих хромосом, які будуть приймати участь у створення нащадків (нового покоління).

Зрештою йде формування нової популяції хромосом при використанні генетичних операторів схрещування (crossover-65%) і мутації (mutation-40%) до відібраних батьківських хромосом [26]. Функції схрещування та мутації у

відсотковому співвідношенні були обрані в результаті тестувань. При схрещуванні нижче 50% якість параметрів залишається незмінною, також виходить при ситуації мутаціях нижче 50%.

В результаті перевірки умови закінчення ітерацій відбувається або зупинка алгоритму, або повертається до розрахунку функції пристосованості.

На останок вибір найкращої хромосоми з максимальним значенням $p(X_i)$ [26]. Суміщення зображень об'єктів програмно реалізовано в середовищі PyCharm [26]. Отримано задовільне суміщення зображень об'єктів при допустимих значеннях часу розрахунків [26]. Зазначену модифікацію методу було використано для оптимізації CNN нейронної мережі: кількості вузлів мережі, кількість нейронів для кожного шару окрім вхідного та вихідного, розміру підвибірки та кількості епох навчання [26].

Випадковим чином генерується кінцевий набір пробних рішень за формулою (3.2):

$$p^1 = \{p_1^1 \dots p_n^{k1}\}, p_n^1 \in X, \quad (3.2)$$

де p – покоління;

n - розмір популяції;

X - ген.

Оцінка пристосованості поточного покоління за формулою 3.3:

$$F^k = \{f_1^k \dots f_n^k\}, f_m^k = W(f_n^k), \quad (3.3)$$

де f - функція пристосованості;

k – номер покоління.

Генерація нового покоління за допомогою операторів селекції S , схрещування C і мутацій M за формулою 3.4:

$$p^k = M * C * S(p^k, F^k). \quad (3.4)$$

В процесі селекції відбирають тільки кілька кращих пробних рішень, інші далі не використовуються. На рисунку 3.2 представлено поділ хромосом з деякою ймовірністю дочірні хромосоми отримують від батьківських гени за допомогою схрещування і після цього відбувається розрахунок мутацій.

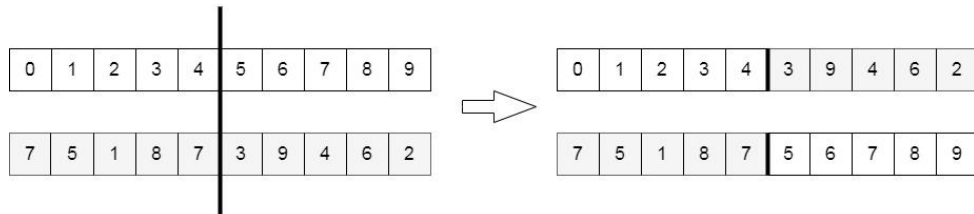


Рисунок 3.2 – Поділ хромосом

Схрещування замість пари рішень створює іншу, елементи якої перемішані якимось особливим чином. Мутація випадковим чином змінює якусь компоненту пробного рішення на іншу.

Прийmemo, що цільова функція (фітнес-функція (F)), функція якості, функція придатності, функція пристосованості) обчислює суму квадратів відхилень вихідних даних виразу $Y_i = f(X_i)$ від заданих еталонних значень Y_i для певних наборів безлічі вхідних даних за формулою (3.5):

$$X_i (1 \leq i \leq N): F = \sum_{i=1}^N (f(X_i) - Y_i)^2, \quad (3.5)$$

де N - число експериментальних даних.

Метою алгоритму є пошук мінімуму F . На практиці, якщо виходило кілька рішень з однаковим значенням цільової функції, вибиралося рішення з меншою оцінкою структурної складності, тобто з меншою загальною довжиною (сумою числа елементів) формул рішення.

Було проведено близько 50 експериментів, в ході яких було виявлено особливо важливі моменти, які наведені у таблиці 3.4 рекомендацій. На таблиці можна побачити які параметри були використані для модернізації CNN.

Таблиця 3.4 - Рекомендації щодо налаштування параметрів ГА модифікованого для модернізації CNN

Назва параметру	Значення
Кількість осіб	5-25(5-Мінімальне значення для «тестового» запуску)
Кількість популяцій	10(тут трохи більше 0.4 бо буде тривалий процес навчання що призводитиме до перенавчання на момент точності ~80%)
Кількість нейронів в першому шарі	32– 64(крок вибору у нейронів $x*2$)
Кількість нейронів в другому шарі	128– 4098(крок вибору у нейронів $x*2$)
Кількість нейронів в третьому шарі	128– 4098(шаг вибору кол-во нейронов $x*2$)
Кількість нейронів в четвертому шарі	128– 4098(крок вибору у нейронів $x*2$)
Кількість нейронів в п'ятому шарі	128– 4098(шаг вибору кол-во нейронов $x*2$)
Ша дропауту	0.2– 0.4(тут не більше 0.4 бо буде тривалий процес навчання що призводитиме до перенавчання на моменті точності ~80%)

3.3 Висновки за розділом 3

В результаті виконання розділу розглянуто спосіб модифікації CNN за допомогою модифікації ахітектури, аналізу набору даних та використання генетичного алгоритму. Розроблено та описано модифікації генетичного алгоритму, а саме спосіб вибору осіб до батьківських пар за рахунок обчислення їх неспорідненості та використанню паралельного генетичного алгоритму. Визначено та описано модель даних, на яких тренується модель та описано схему її функціонування.

4 КЕРІВНИЦТВО ПРОГРАМІСТА

4.1 Призначення та умови застосування програми

4.1.1 Призначення програми

Розроблений програмний продукт реалізовано відповідно до поставленого завдання та задовольняє всім необхідним вимогам.

Програмний продукт призначений для розпізнавання медичних упаковок та класифікації їх за назвою препарату та дозуванням, також спрощує задачу сортування на аптеці. Може бути використаний першостільниками для швидкого аналізу прибувшого товару.

4.1.2 Умови застосування програми

Дане ПЗ розроблене використовуючи мову програмування Python 3.7. Розробка ПЗ та його тестування проводилися з обладнанням на таких конфігураціях:

- процесор Intel Core i5 – 7400, 3.0 GHz;
- 32 GB оперативної пам'яті;
- операційна система Windows 10 (64);

Розроблене програмне забезпечення можна використовувати в операційних системах Windows і Linux. Програмні компоненти мають ряд залежностей, у разі недотримання яких робота пакета може бути некоректною та нестабільною. Ці основні залежності включають:

- мова програмування Python 3.7;
- бібліотека нейронної мережі Keras;
- бібліотека машинного навчання Scikit-learn;
- бібліотека для побудови та візуалізації даних Matplotlib;
- бібліотека, яка дозволяє ефективно працювати з матрицями, таблицями та числовими даними NumPy.

4.2 Звернення до програми

Для початку розпізнавання та запуску консольного застосунку необхідно упевнитися у відповідності характеристик.

ПП має файлову структуру зображену на рисунку 4.1.

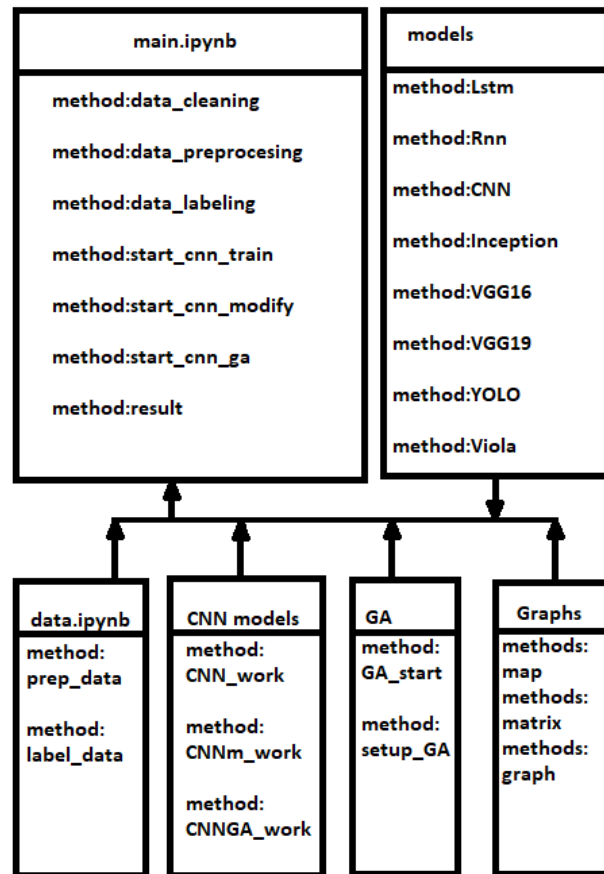


Рисунок 4.1 - Файлова структура ПП

Папка dataset містить набір тренувального набору даних у форматі .png, що використовувались під час тренування моделей.

Папка model призначена для зберігання файлів моделі.

Папка results призначена для зберігання результатів роботи програми, а саме графіків та матриці і мапи товарів.

Main – це папка, що містить файл CNN.py, який містить скрипт реалізації штучної нейронної мережі.

Data- це папка для підготовки даних.

Cnn-models – це папка з моделями згорткової, модификованої та модификованої генетичним методом нейромереж.

GA – це папка з модифікованим генетичним алгоритмом.

Graphs – це папка що містить графіки навчання.

4.3 Вхідні та вихідні дані

В якості вхідних даних використано датасет, який містить набір фотографій для кожної медичної упаковки. Упаковки мають свою папку зі своїми фотографіями в ньому.

Результат роботи моделі виводиться у консоль та має значення у відсотках. Результати роботи моделі наведені на рисунку 4.2.

```
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the constructor
Epoch 1/40
75/75 [=====] - 17s 221ms/step - loss: 2.8335 - acc: 0.0629 - val_loss: 2.8334 - val_acc: 0.0550
Epoch 2/40
75/75 [=====] - 8s 110ms/step - loss: 2.8334 - acc: 0.0680 - val_loss: 2.8334 - val_acc: 0.0550
Epoch 3/40
75/75 [=====] - 8s 106ms/step - loss: 2.8333 - acc: 0.0646 - val_loss: 2.8334 - val_acc: 0.0550
Epoch 4/40
75/75 [=====] - 8s 110ms/step - loss: 2.8336 - acc: 0.0456 - val_loss: 2.8334 - val_acc: 0.0550
Epoch 5/40
75/75 [=====] - 8s 106ms/step - loss: 2.8332 - acc: 0.0612 - val_loss: 2.8334 - val_acc: 0.0550
Epoch 6/40
75/75 [=====] - 8s 108ms/step - loss: 2.8333 - acc: 0.0680 - val_loss: 2.8334 - val_acc: 0.0550
...
```

Рисунок 4.2 – Приклад виведення результату виконання моделі у
jupyternotebook

Графічний вигляд результату наведено на рисунку 4.3. На графіку зображено графічне відображення результату тренування згорткової нейромережі, а саме графік точності та функції втрат.

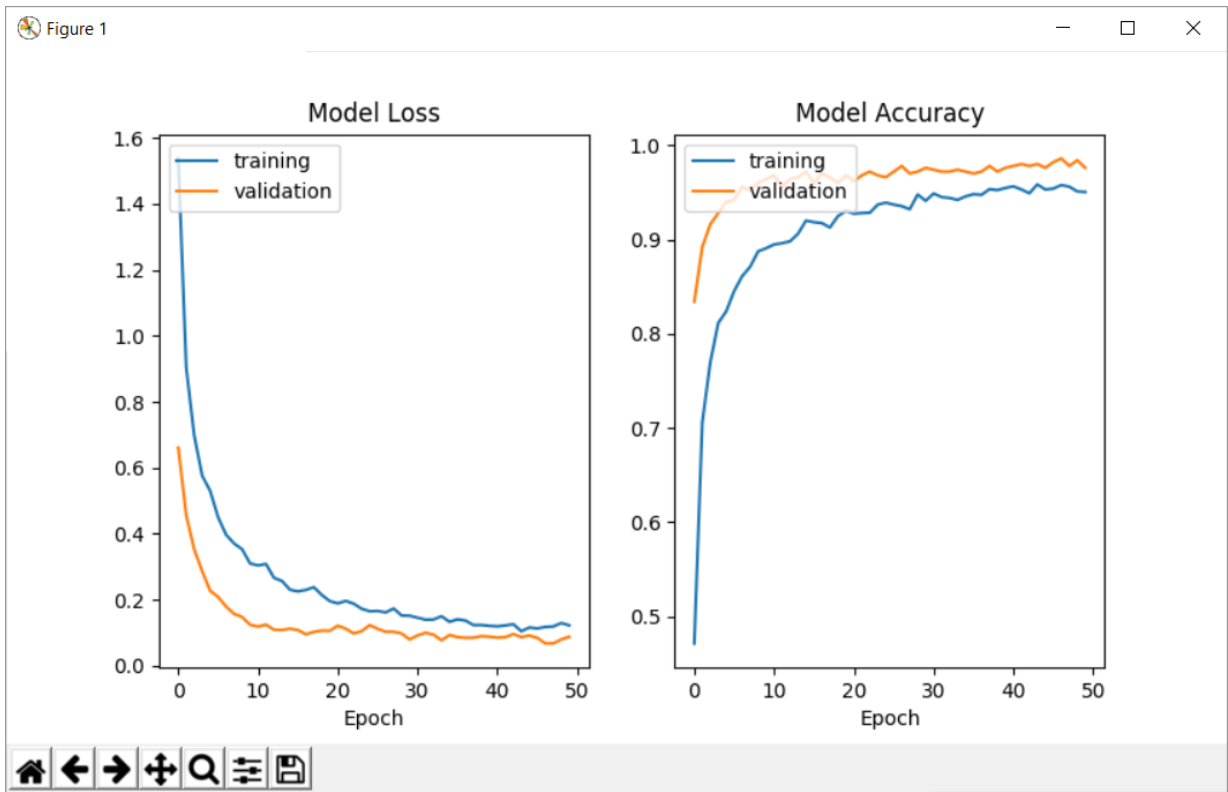


Рисунок 4.3 – Графічне представлення результату тренування згорткової неймережі

4.4 Повідомлення програмісту

Повідомлення табл. 4.1 виводяться в разі виникнення будь-яких помилок при спробі виконання запитів, а також в якості інформаційних повідомлень про успішність виконання необхідних дій.

Таблица 4.1 - Повідомлення та їх опис

Повідомлення	Опис
Дані завантажено успішно	Дані успішно завантажені та готові для навчання
Помилка завантаження даних	Виникла помилка під час завантаження даних
Навчання розпочато	Навчання неймережі розпочато
Навчання закінчено	Навчання неймережі закінчено

Продовження таблиці 4.1.

Повідомлення	Опис
Помилка системи або розрахунку	Під час навчання нейромережі виникли помилки які пов'язані або з системою, або з розрахунком
Необроблена помилка	Помилка яка не була помічена раніше

4.5 Висновки за розділом

Проведено детальний опис роботи з програмним продуктом. Розроблене програмне забезпечення можна використовувати в операційних системах Windows і Linux. Програмний продукт призначений для розпізнавання медичних упаковок та класифікації їх за назвою препарату та дозуванням, також спрощує задачу сортування на аптеці.

5 КЕРІВНИЦТВО КОРИСТУВАЧА

5.1 Призначення та умови застосування програми

5.1.1 Призначення програми

Розроблений ПП реалізовано відповідно до поставленого завдання та задовольняє всім необхідним вимогам.

Програмний продукт призначений для розпізнавання медичних упаковок та класифікації їх за назвою препарату та дозуванням, також спрощує задачу сортування на аптеці. Може бути використаний першостільниками для швидкого аналізу прибувшого товару.

5.1.2 Умови застосування програми

Дане ПЗ розроблене використовуючи мову програмування Python 3.7. Розробка ПЗ та його тестування проводилися з обладнанням на таких конфігураціях:

- процесор Intel Core i5 – 7400, 3.0 GHz;
- 32 GB оперативної пам'яті;
- Операційна система Windows 10 (64);

Розроблене програмне забезпечення можна використовувати в операційних системах Windows і Linux. Програмні компоненти мають ряд залежностей, у разі недотримання яких робота пакета може бути некоректною та нестабільною. Ці основні залежності включають:

мова програмування Python 3.7;

- нейромережева бібліотека Keras;
- бібліотека машинного навчання Scikit-learn;
- бібліотека для побудови графіків та візуалізації даних Matplotlib;
- бібліотека, що дозволяє ефективно працювати з матрицями, таблицями та числовими даними NumPy.

5.2 Виконання програми

5.2.1 Екран входу

Для початку розпізнавання та запуску консольного застосунку необхідно упевнитися у відповідності характеристик .

Для того, щоб запустити програму та виконати діагностування необхідно відкрити інтерпретатор командного рядка, у випадку операційної системи Windows – це cmd.exe та прописати шлях до необхідного файлу, а саме Cnn_arch.ipynb, як зображено на рисунку 5.1.

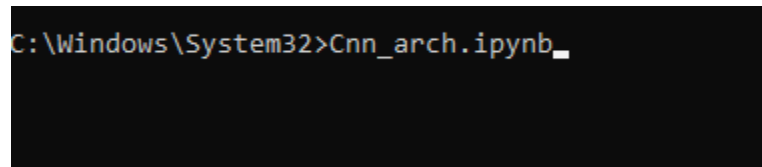


Рисунок 5.1 – Запуск програми через командний рядок

5.2.2 Екран роботи алгоритму

Якщо виконані всі вимоги до програмного продукту, йде виконання програми, яке супроводжується виводом певної інформації (рис. 5.2-5.3). В командному рядку виводиться інформація про кількість епох, значення функції втрат, точність.

```
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the constructor
Epoch 1/40
75/75 [=====] - 17s 221ms/step - loss: 2.8335 - acc: 0.0629 - val_loss: 2.8334 - val_acc: 0.0550
Epoch 2/40
75/75 [=====] - 8s 110ms/step - loss: 2.8334 - acc: 0.0680 - val_loss: 2.8334 - val_acc: 0.0550
Epoch 3/40
75/75 [=====] - 8s 106ms/step - loss: 2.8333 - acc: 0.0646 - val_loss: 2.8334 - val_acc: 0.0550
Epoch 4/40
75/75 [=====] - 8s 110ms/step - loss: 2.8336 - acc: 0.0456 - val_loss: 2.8334 - val_acc: 0.0550
Epoch 5/40
75/75 [=====] - 8s 106ms/step - loss: 2.8332 - acc: 0.0612 - val_loss: 2.8334 - val_acc: 0.0550
Epoch 6/40
75/75 [=====] - 8s 108ms/step - loss: 2.8333 - acc: 0.0680 - val_loss: 2.8334 - val_acc: 0.0550
```

Рисунок 5.2 – Виконання програми

Також виконання програми супроводжується виводом графічної інформації, яка може виглядати наступним чином.

На рисунку 5.3 відображено графік функції втрат протягом навчання згорткової нейромережі.

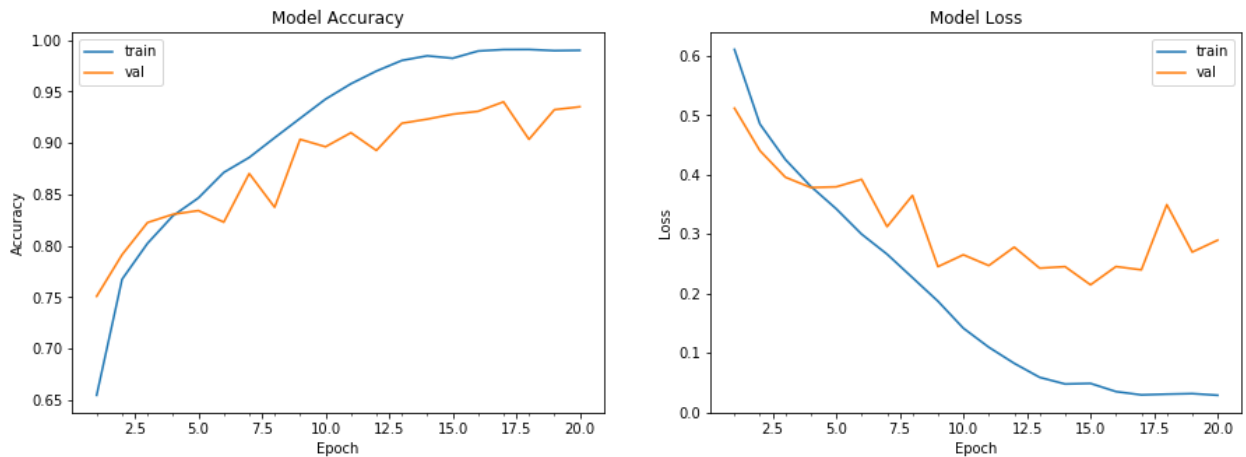


Рисунок 5.3 – Графік точності та функції втрат згорткової нейромережі

На рисунку 5.4 зображено графічне представлення результату тренування згорткової нейромережі, а саме графік точності та функції втрат.

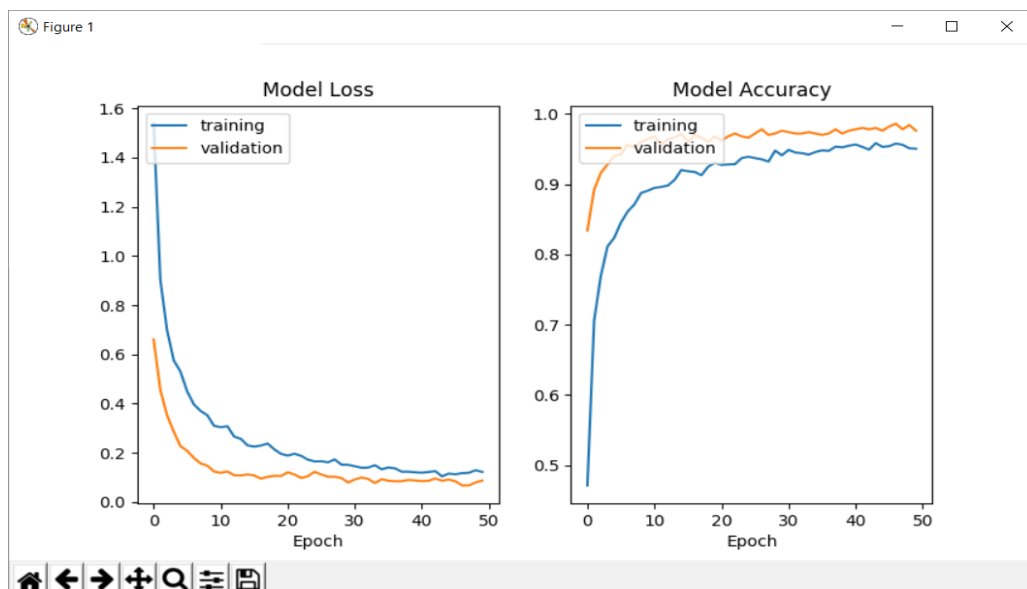


Рисунок 5.4 – Графічне представлення результату тренування згорткової нейромережі

5.3 Порівняння роботи згорткової нейронної мережі

Порівняння роботи згорткової нейронної мережі з вже існуючими моделями нейронних мереж та порівняння їх з модифікованою MBDC нейромереж зображено в таблиці 5.1.

Таблиця 5.1 – Порівняння нейромереж

Назва	Час навчання, ч	Точність розрахункова, %	Точність практична, %	Швидкість праці на робочому місці, с
Віола - Джонс он	-	63	37	7
CNN	6.3	87	80	7
MBDC	8.2	93	92	4
Incepti onv2	12.8	94	84	6
RNN	7	65	63	5
LSTM	8	63	54	8
YOLO 5	12	89	67	5
VGG1 6	10	92	73	6
Incepti onv3	21.1	96	90	7
VGG1 9	15.5	95	89	6

У таблиці 5.2 представлено порівняння між MBDC та MBDCGAM (з модернізованим ГА).

Таблиця 5.2 – Порівняння нейромережи MBDC та MBDCGAM

Назва	Точність розрахункова, %	Час, ч	Точність практична, %	Швидкість праці на робочому місці, с
MBDC	93	8.2	92	4
MBDCGAM	97	11.1	98	4

Згідно з таблицею 5.1 можна побачити, що в ході модифікації НС ми досягли 92% точності розпізнавання. Це було викликано тим, що при використанні НС і MBDC було недостатньо умов для розпізнавання упаковок-близнюків, що і призвело до такої низької точності в практичному розпізнаванні. У таблиці 6.2 можемо побачити що ми так само досягли MBDCGAM 98% точності на реальній вибірці даних, з чого випливає, що ми досягли успіху і ГА виправдав свої очікування

5.4 Висновки за розділом

Розділ містить опис роботи оператора з розробленим програмним продуктом. Вказуються параметри програмно-апаратних компонентів. Описано необхідні програмні продукти, які потребують програмного забезпечення.

У розділі описано, як працювати з програмним продуктом, як з ним працювати, що у нашій роботі отримує користувач під час роботи. Наведено приклади відображення графічної інформації.

ВИСНОВКИ

Певною ,та мабуть головною перевагою нейронних мереж є їх здатність до навчання на основі лімітованих даних навколишнього середовища та у результаті навчання підвищити свою продуктивність. Посилення якості праці та навчання відбувається з часом у відповідності до певних правил. В цілому для навчання нейромереж потрібно створити шари з певною кількістю нейронів для початку навчання та перерозподілу параметрів,та вагових коефіцієнтів. .У даній роботі розглянуто переваги та якості основних методів,типи нейромереж, а також технік навчання нейронних мереж, детально описано складові частини обраних згорткових нейромереж та аргументовано їх взаємозв'язок з нашою задачею. Серед запропонованої кількості архітектур згорткових мереж критерії оцінки якості навчання виражаються на графіках через показники, а саме такі, як точність, число втрат (обраховує функція втрат) та інші. Програмна реалізація модифікованих нейромереж та продукту базується на використанні технологій глибинного навчання представлених у вигляді бібліотек та спеціальних пакетів, що дозволяють програмістам модифікувати,розроблювати та керувати архітектурами нейромереж та налаштовувати параметри усі параметри за допомогою гнучкого налаштування.

Результатом роботи стала розроблена модернізована модель, на основі базової архітектури, згорткової мережі для розпізнавання медичних упаковок. У модифікованому методі було запропоновано змінити архітектуру повнозв'язкових шарів від 2 до 4 згорткових шарів, при більше 4 згорткових шарів у повнозв'язному шарі час навчання збільшується, а точність падає. Так само було додано необхідну кількість нейронів для кожного шару у всіх повнозв'язних шарах, що було з'ясовано експериментальним шляхом. Було додано шар для розподілу параметрів. Від стандартного застосування наш шар має два шари активації, активаційну функцію.. Значне підвищення

значення відсотку точності розпізнавання майже для усіх запропонованих для тестування наборах даних.

Можна побачити, що в ході модифікації НС ми досягли 92% точності розпізнавання. Це було викликано тим, що при використанні НС і MBDC було недостатньо умов для розпізнавання упаковок-близнюків, що і призвело до такої низької точності в практичному розпізнаванні. Модифікація MBDCGAM досягла 98% точності на реальній вибірці даних, з чого випливає, що ми досягли успіху і ГА виправдав свої очікування.

Основні положення і результати роботи були представлені та обговорювалися на міжнародних конференціях (котрі входять до міжнародної наукометричної бази Scopus):

- IEEE International Scientific-Practical Conference: Problems of Infocommunications Science and Technology, PIC S and T (Kharkiv, 2018, 2020, 2021, Kyiv, 2019);
- 15th International Conference on the Experience of Designing and Application of CAD Systems, CADSM 2019 (Polyana, , Ukraine, 2019);
- 10th International Conference on Advanced Computer Information Technologies, ACIT 2020 (Deggendorf, Germany, 2020 p.);
- International Workshop : Informatics and Data-Driven Medicine ((IDDM), Lviv, 2019, Sweden, 2020);
- Information Technology and Interactions (Kyiv, 2020).

Опубліковано розділ у колективній монографії Lecture Notes on Data Engineering and Communications Technologies [this link is disabled](#) (Germany, 2021), входить до міжнародної наукометричної бази Scopus.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Genetic Algorithm for Solution of the Problem of Optimal Location of the Distributed Electrical Networks / [D. Goncharenko, A. Oliinyk, I. Fedorchenko, S. Korniienko, O. O. Stepanenko, A. S. Kharchenko, Y. Fedorchenko] 2020 10th International Conference on Advanced Computer Information Technologies (ACIT) – Deggendorf, Germany, 2020. – P. 380-385. (Входить до міжнародної наукометричної бази Scopus). Doi: 10.1109/ACIT49673.2020.9208888.
2. Evolutionary method for solving the traveling salesman problem / [A. Oliinyk, I. Fedorchenko, O. Stepanenko M. Rud, D. Goncharenko] // Problems of Infocommunications. Science and Technology: 5th International Scientific-Practical Conference PICST2018, Kharkiv, 9–12 October 2018 : proceedings. – Kharkiv: KNURE, 2018. – P. 331-339. (Входить до міжнародної наукометричної бази Scopus). Doi: 10.1109/INFOCOMMST.2018.8632033.
3. Development of genetic methods for predicting the incidence of volumes of emissions of pollutants in air / [A. Oliinyk, I. Fedorchenko, O. Stepanenko A. Katschan, Y. Fedorchenko, D. Goncharenko] // 2nd International Workshop : Informatics and Data-Driven Medicine IDDM 2019, Lviv, 11-13 November 2019 : proceedings. – Lviv: LPNU, 2019. – P. 340-353. (Входить до міжнародної наукометричної бази Scopus).
4. Genetic method of image processing for motor vehicle recognition / [I. M. Fedorchenko, A. O. Oliinyk, O. O. Stepanenko, T. Zaiko, A. Svyrydenko, D. Goncharenko] // CEUR Workshop Proceedings. – 2019. – Vol.2353. – P. 211-226. (Входить до міжнародної наукометричної бази Scopus).
5. Combinatorial optimization problems solving based on evolutionary approach / [I. M. Fedorchenko, A. O. Oliinyk, O. O. Stepanenko, M. Rud, D. Goncharenko] // 2019 15th International Conference : Experience of Designing and Application of CAD Systems (CADSM), Polyana, 26 Feb.-2 March 2019 :

proceedings. – Los Alamitos: IEEE, 2019. – P. 41-45. (Входить до міжнародної наукометричної бази Scopus).

6. Краткая медицинская энциклопедия: диагностика [Электрон. ресурс]. – Режим доступа: <http://www.golkom.ru/kme/05/1-408-3-1.html>.

7. Практика: Распознавание объектов на изображениях с помощью Keras [Электрон. ресурс]. – Режим доступа: <https://www.asozykin.ru/courses/nnpython-lab2>.

8. Практика: Распознавание рукописных цифр из набора данных MNIST на Keras [Электрон. ресурс]. – Режим доступа: <https://www.asozykin.ru/courses/nnpython-lab1>.

9. Schapire R.E. A decision-theoretic generalization of on-line learning and an application to boosting [Text] / R. E. Schapire, Y. Freund // Journal of Computer and System Sciences. – 1997. – Vol. 55. – P. 119-139.

10. Schapire R.E. Improved Boosting Algorithms Using Confidence-rated Predictions [Text] / R. E. Schapire, Y. Singer // Machine Learning. – 1999. – Vol. 37. – P. 297-336.

11. Wei Y. Efficient Histogram-Based Sliding Window [Text] / Y. Wei, L. Tao // IEEE CVPR. – 2010. – P. 3003-3010.

12. Yolo v3 Object Detection in Tensorflow [Electronic resource]. – Access mode: <https://www.kaggle.com/aruchomu/yolo-v3-object-detection-in-tensorflow>.

13. Arthropod Taxonomy Orders Object Detection Dataset [Electronic resource]. – Access mode: <https://www.kaggle.com/mistag/arthropod-taxonomy-orders-object-detection-dataset>.

14. Abiyev, R., Ma'aitah, M.: Deep Convolutional Neural Networks for Chest Diseases Detection. Journal of Healthcare Engineering, vol. 2018, pp. 1-11 (2018). doi: 10.1155/2018/4168538.

15. Pattrapisetwong, P., Chiracharit, W.: Automatic lung segmentation in chest radiographs using shadow filter and multilevel thresholding. 2016

International Computer Science and Engineering Conference (ICSEC), (2016). doi: 10.1109/icsec.2016.7859887.

16. Liu, Y., Wang, H., Gu, Y., Lv, X.: Image classification toward lung cancer recognition by learning deep quality model. *Journal of Visual Communication and Image Representation*, vol. 63, p. 102570, (2019). doi: 10.1016/j.jvcir.2019.06.012.

17. Allaouzi, I., Ben Ahmed, M.: A Novel Approach for Multi-Label Chest X-Ray Classification of Common Thorax Diseases. *IEEE Access*, vol. 7, pp. 64279-64288, (2019). doi: 10.1109/access.2019.2916849.

18. Candemir, S., Antani, S.: A review on lung boundary detection in chest X-rays. *International Journal of Computer Assisted Radiology and Surgery*, vol. 14, no. 4, pp. 563-576, (2019). doi: 10.1007/s11548-019-01917-1.

19. Ke, Q. et al.: A neuro-heuristic approach for recognition of lung diseases from X-ray images. *Expert Systems with Applications*, vol. 126, pp. 218-232, (2019). Doi: 10.1016/j.eswa.2019.01.060.

20. Souza, J., Bandeira Diniz, J., Ferreira, J., França da Silva, G., Correa Silva, A., de Paiva, A.: An automatic method for lung segmentation and reconstruction in chest X-ray using deep neural networks. *Computer Methods and Programs in Biomedicine*, vol. 177, pp. 285-296, (2019). doi: 10.1016/j.cmpb.2019.06.005.

21. Sirazitdinov, I., Kholiavchenko, M., Mustafaev, T., Yixuan, Y., Kuleev, R., Ibragimov, B.: Deep neural network ensemble for pneumonia localization from a large-scale chest x-ray database. *Computers & Electrical Engineering*, vol. 78, pp. 388-399, (2019). doi: 10.1016/j.compeleceng.2019.08.004.

22. Jain, R., Nagrath, P., Kataria, G., Sirish Kaushik, V., Jude Hemanth, D.: Pneumonia detection in chest X-ray images using convolutional neural networks and transfer learning. *Measurement*, vol. 165, p. 108046, (2020). doi: 10.1016/j.measurement.2020.108046.

23. S.K., L., Mohanty, S., K., S., N., A., Ramirez, G.: Optimal deep learning model for classification of lung cancer on CT images. *Future Generation Computer Systems*, vol. 92, pp. 374-382, (2019). doi: 10.1016/j.future.2018.10.009.
24. Gupta, N., Gupta, D., Khanna, A., Rebouças Filho, P., de Albuquerque, V.: Evolutionary algorithms for automatic lung disease detection. *Measurement*, vol. 140, pp. 590-608, (2019). doi: 10.1016/j.measurement.2019.02.042.
25. Shakeel, P., Burhanuddin, M., Desa, M.: Lung cancer detection from CT image using improved profuse clustering and deep learning instantaneously trained neural networks. *Measurement*, vol. 145, pp. 702-712, (2019). doi: 10.1016/j.measurement.2019.05.027.
26. Tang, P. Adaptive directed mutation for real-coded genetic algorithms / P. Tang, M. Tseng // *Applied Soft Computing*. – 2013. – Vol. 13. – P. 600-614.
27. Qin, C., Yao, D., Shi, Y., Song, Z.: Computer-aided detection in chest radiography based on artificial intelligence: a survey. *BioMedical Engineering OnLine*, vol. 17, no. 1, (2018). doi: 10.1186/s12938-018-0544-y.
28. Zeng, Z., Liang, N., Yang, X., Hoi, S.: Multi-target deep neural networks: Theoretical analysis and implementation", *Neurocomputing*, vol. 273, pp. 634-642, (2018). doi: 10.1016/j.neucom.2017.08.044.

ДОДАТОК А
Технічне завдання

Вступ

Розробка, дослідження та реалізація методів вирішення завдань аналізу, розпізнавання та оцінювання зображень є одним із провідних напрямів інформатики. Результати відповідних фундаментальних та прикладних досліджень безпосередньо використовуються в технічній діагностиці, неруйнівний контроль, дистанційне зондування, екологічний моніторинг, прогнозування та діагностику в медицині, планування, пошук у геології, прогнозування в хімії, автоматизацію наукових досліджень. Ця тематика приваблює значну кількість дослідників, проте, переважна більшість робіт у цій галузі має емпіричний характер.

А.1 Підстава для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Дослідження та програмна реалізація методів розпізнавання медичних упаковок», затверджене наказом Національного університету «Запорізька політехніка» № 435 від 05 листопада 2021 р.

А.2 Призначення розробки

Програмний продукт призначений для вирішення задачі розпізнавання медичних упаковок та класифікації їх за назвою препарату та дозуванням.

А.3 Основні вимоги до програми, що розробляється

А.3.1 Вимоги до функціональних характеристик

Дане ПЗ розроблене використовуючи мову програмування Python 3.7. Розробка ПЗ та його тестування проводилися з обладнанням на таких конфігураціях:

- відповідність темі проєкту;
- коректність роботи;
- можливість завантаження даних з файлу;
- можливість роботи у консольному режимі;
- можливість виводу вихідних даних у графічному та текстовому вигляді;
- можливість збереження графічної інформації.

А.3.2 Вимоги до програми

Для запуску програми необхідно упевнитися у відповідності характеристик комп'ютера. Впевнитися, що всі необхідні залежності попередньо встановлені на ЕОМ:

- папка dataset містить набір тренувального набору даних у форматі .png, що використовувались під час тренування моделей.
- папка model призначена для зберігання файлів моделі.
- папка results призначена для зберігання результатів роботи програми, а саме графіків та матриці і мапи товарів.
- main – це папка, що містить файл CNN.py, який містить скрипт реалізації штучної нейронної мережі
- Data- це папка для підготовки даних;

- Cnn-models – це папка з моделями згорткової, модифікованої та модификованої генетичним методом нейромереж;
- GA – це папка з модифікованим генетичним алгоритмом;
- Graphs – це папка що містить графіки навчання.

A.3.3 Вимоги до надійності

Програма повинна забезпечувати достатній рівень надійності. В процесі роботи можуть виникати збійні ситуації, що мають відповідні повідомлення, що описані в керівництві користувача та програміста. Програмний продукт повинен забезпечувати досягнення відносно якісного результату за умов слідування користувачем правил керівництва.

A.3.4 Умови експлуатації

Експлуатація програмного забезпечення розпізнавання медичних упаковок здійснюватися відповідно до експлуатаційної документації на розроблене програмне забезпечення.

A.3.5 Вимоги до складу та параметрів технічний засобів

Зазначене програмне забезпечення написане мовою Python 3.7. Розробка і тестування проводилися на обладнанні з такими конфігураціями:

- процесор Intel Core i5 – 7400, 3.0 GHz;
- 32 GB оперативної пам'яті;
- Операційна система Windows 10 (64).

Розроблене ПЗ можна використовувати в операційних системах Windows і Linux. Програмні компоненти мають ряд залежностей, у разі

недотримання яких робота пакета може бути некоректною та нестабільною.

Ці основні залежності включають:

- мова програмування Python 3.7;
- бібліотека нейронної мережі Keras;
- бібліотека машинного навчання Scikit-learn;
- бібліотека для побудови та візуалізації даних Matplotlib;
- бібліотека, яка дозволяє ефективно працювати з матрицями, таблицями та числовими даними NumPy.

A.3.6 Вимоги до маркування і пакування

Програмне забезпечення розпізнавання медичних упаковок може бути записана на флеш-носії.

A.3.7 Вимоги до транспортування й зберігання

Вимоги до транспортування й зберігання програмного забезпечення розпізнавання медичних упаковок аналогічні вимогам, пропонованим до носія, на якому записана програма.

A.4 Вхідні та вихідні дані

В якості вхідних даних використано датасет, який містить набір фотографій для кожної медичної упаковки. Упаковки мають свою папку зі своїми фотографіями в ньому.

Результат роботи моделі виводиться у консоль та має значення у відсотках. Приклад виводу результату зображено на рисунку A.1.

```
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the constructor
Epoch 1/40
75/75 [=====] - 17s 221ms/step - loss: 2.8335 - acc: 0.0629 - val_loss: 2.8334 - val_acc: 0.0550
Epoch 2/40
75/75 [=====] - 8s 110ms/step - loss: 2.8334 - acc: 0.0680 - val_loss: 2.8334 - val_acc: 0.0550
Epoch 3/40
75/75 [=====] - 8s 106ms/step - loss: 2.8333 - acc: 0.0646 - val_loss: 2.8334 - val_acc: 0.0550
Epoch 4/40
75/75 [=====] - 8s 110ms/step - loss: 2.8336 - acc: 0.0456 - val_loss: 2.8334 - val_acc: 0.0550
Epoch 5/40
75/75 [=====] - 8s 106ms/step - loss: 2.8332 - acc: 0.0612 - val_loss: 2.8334 - val_acc: 0.0550
Epoch 6/40
75/75 [=====] - 8s 108ms/step - loss: 2.8333 - acc: 0.0680 - val_loss: 2.8334 - val_acc: 0.0550
...
```

Рисунок А.1 – Приклад виводу результату роботи моделі у консолі

Також результат виводиться у графічному вигляді рис. А.2. На рисунку А.2 зображено графічне представлення результату тренування згорткової нейромережі, а саме графік точності та функції втрат.

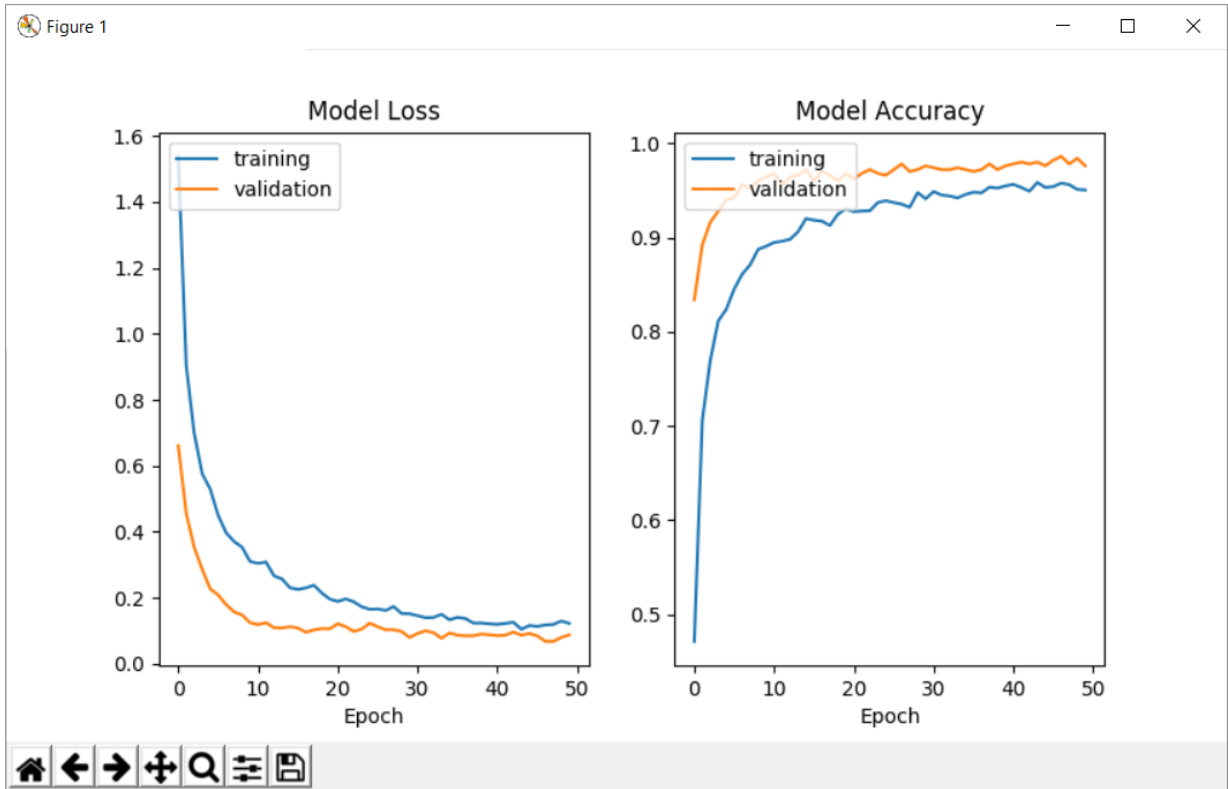


Рисунок А.2 – Графічне представлення результату тренування згорткової нейромережі

ДОДАТОК Б
Опис програми

Б.1 Загальні відомості

Програмне забезпечення розпізнавання медичних упаковок розроблена з використанням мови Python 3.7.

Б.2 Функціональне призначення

Функціональним призначенням програмного забезпечення є автоматизація процесу сортування тільки прибулих ліків на аптечне відділення було прийнято рішення про створення робота-сортувальника, який на основі модифікованою НС буде, після аналізу упаковки за назвою препарату та дозуванням буде видавати точну назву та дозування, що скоротить час заповнення накладної.

Проблеми розпізнавання образу полягають в упаковках близнюках які різняться трохи друг від друга. Це можуть бути як різні препарати, так і упаковки одного препарату, але з різним дозуванням (рис Б.1 та рис. Б.2).

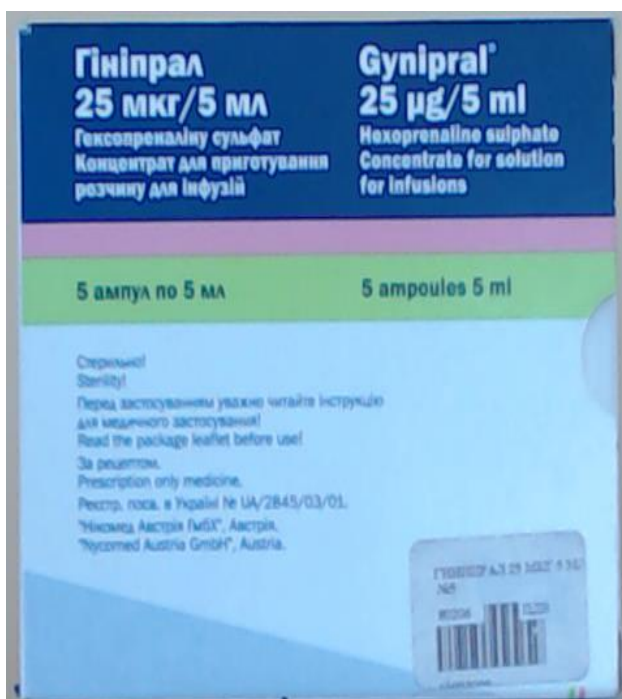


Рисунок Б.1 – Медична упаковка Гініпрал

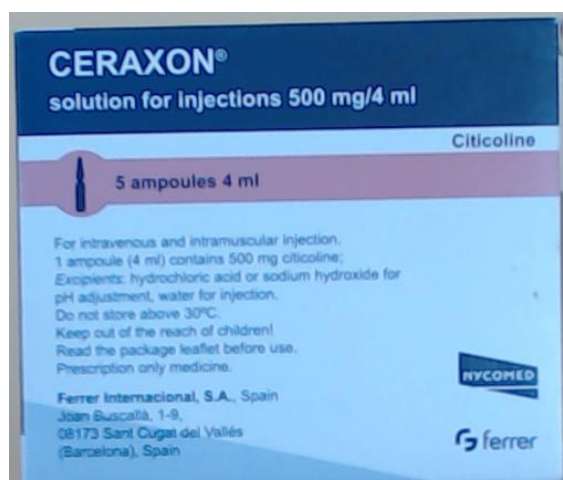


Рисунок Б.2 – Медична упаковка Цераксон

Так і упаковки одного препарату але з різним дозуванням (рис. Б.3 та рис. Б.4).



Рисунок Б.3 – Медична упаковка Афлубін (20 мл.)



Рисунок Б.4 – Медична упаковка Афлубін (50 мл.)

Через безліч мінімальних відмінностей між препаратами не відрізняють їх між собою, що робить дане завдання досить складним, так само посилюють дане завдання моменти з технічним обладнанням, а саме:

- погані камери (нечітке зображення з камер, що дає зайві перешкоди та шуми);
- погане освітлення;
- невірний, не зафіксований ракурс;
- старе устаткування робочої машини яке уповільнює навчання, а й робить його продуктивним.

У сукупності дані фактори і до них відносяться дають відчутну похибку в розпізнаванні образу об'єкта (упаковки, ампули, флакони), а за рахунок того, що в НС існує поняття перенавчання, то шайи на успіх падають до при нульових значень. Через це модель залежності якості передбачення від якості набору даних можна уявити в вигляді наступної формули (Б.1):

$$A_{fin} = f(x_{light}, x_{distance}, x_{quality}, x_{foreshortening}, x_{quantity}, x_{focus}, x_{background}), (Б.1)$$

де A_{fin} — показник якості передбачення нейромережі;

x_{light} — вплив світла;

$x_{distance}$ — вплив дистанції до об'єкту;

$x_{quality}$ — вплив якості фотографії;

$x_{foreshortening}$ — вплив ракурса з якого було зроблено фото;

$x_{quantity}$ — вплив кількості фотографій;

x_{focus} — вплив фокусування на об'єкті;

$x_{background}$ — вплив заднього фону .

В якості даних використано фотографії препаратів з різних сторін. Набір на кожний екземпляр нараховує 36 оригінальних.

Б.3 Опис логічної структури

Логічна структура програмного забезпечення висловлена його функціональним призначенням, через це на етапі проєктування було обрано рішення щодо об'єднання певних функціональних компонентів системи до логічних модулів, котрі в свою чергу поділені на функції та реалізують окремі функціональні можливості програм. Структура розробленого продукту зображена на рис.Б.5. Потрібно зазначити, що подібна логічна організація програми дозволяє прискорити розроблення та налагодження програмного продукту, а в майбутньому розширювати функціональні можливості готового проєкту шляхом внесення змін або додавання нових модулів

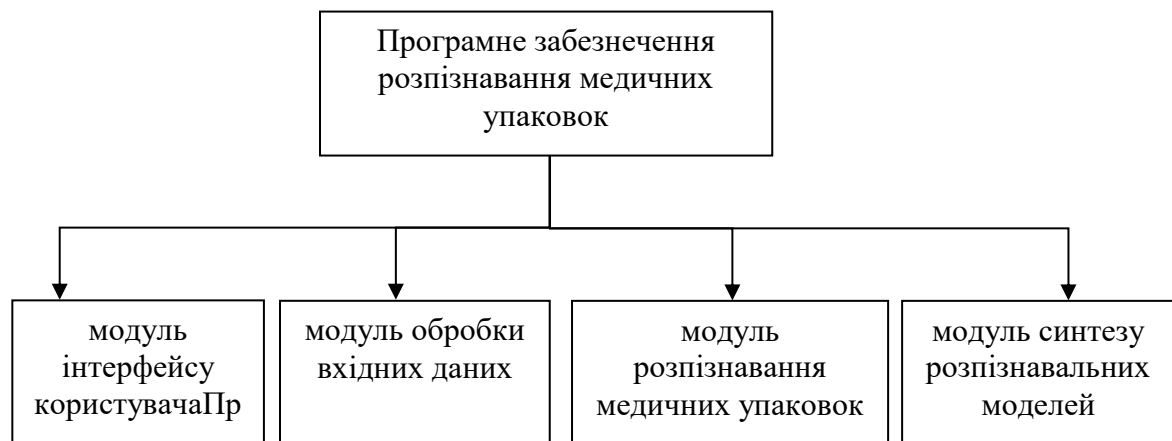


Рисунок Б.5 – Структурна схема програми

Модуль інтерфейсу користувача призначений для виконання наступних функцій:

- забезпечення зручного та інтуїтивно зрозумілого консольного інтерфейсу з користувачем системи;
- забезпечення внутрішньої взаємодії між іншими функціональними блоками системи.

Модуль обробки вхідних даних було призначено для роботи наступних функцій:

- Створення вибірки;
- Збереження вибірки даних
- Завантаження існуючої вибірки даних
- Перевірка коректності введених користувачем даних.

Модуль розпізнавання медичних упаковок призначений для розв'язання завдань розпізнавання медичних упаковок на основі еволюційного підходу. В даному модулі реалізовано різні варіанти еволюційних операторів ініціалізація, відбору, схрещування та мутації.

Б.4 Використані технічні засоби

Для експлуатації програми розпізнавання медичних упаковок необхідні такі програмно-технічні засоби:

- процесор Intel Core i5 – 7400, 3.0 GHz;
- 32 GB оперативної пам'яті;
- операційна система Windows 10 (64).

Розроблене програмне забезпечення можна використовувати в операційних системах Windows і Linux. Програмні компоненти мають ряд залежностей, у разі недотримання яких робота пакета може бути некоректною та нестабільною. Ці основні залежності включають:

- мова програмування Python 3.7;
- бібліотека нейронної мережі Keras;
- бібліотека машинного навчання Scikit-learn;
- бібліотека для побудови та візуалізації даних Matplotlib;
- бібліотека, яка дозволяє ефективно працювати з матрицями, таблицями та числовими даними NumPy.

Б.5 Виклик і завантаження

Для того, щоб запустити програму та виконати діагностування необхідно відкрити інтерпретатор командного рядка, у випадку операційної системи Windows – це cmd.exe та прописати шлях до необхідного файлу, а саме Cnn_arch.ipynb, як зображено на рисунку Б.6.

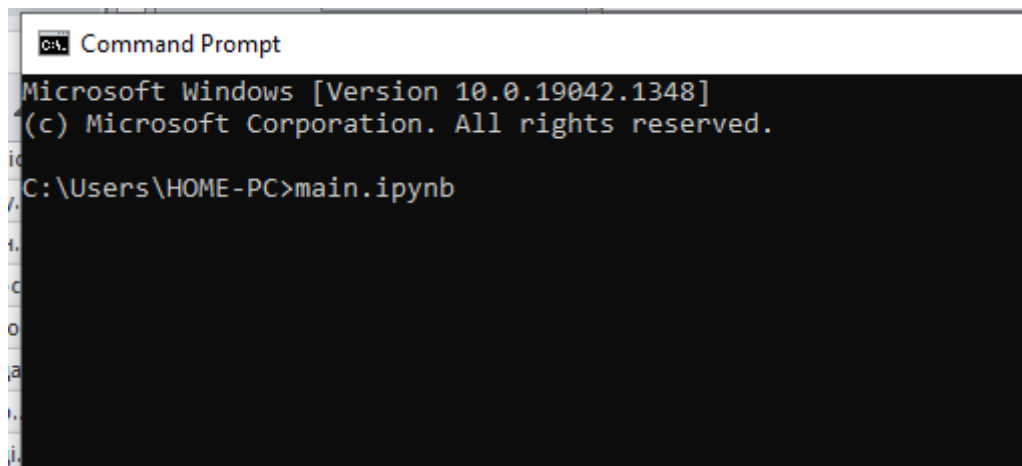


Рисунок Б.6 – Запуск програми через командний рядок.

Б.6 Вхідні та вихідні дані

В якості вхідних даних використано датасет, який містить набір фотографій для кожної медичної упаковки. Упаковки мають свою папку зі своїми фотографіями в ньому.

Результат роботи моделі виводиться у консоль та має значення у відсотках. Приклад виводу результату зображено на рисунку Б.7.

```
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the constructor
Epoch 1/40
75/75 [=====] - 17s 221ms/step - loss: 2.8335 - acc: 0.0629 - val_loss: 2.8334 - val_acc: 0.0550
Epoch 2/40
75/75 [=====] - 8s 110ms/step - loss: 2.8334 - acc: 0.0680 - val_loss: 2.8334 - val_acc: 0.0550
Epoch 3/40
75/75 [=====] - 8s 106ms/step - loss: 2.8333 - acc: 0.0646 - val_loss: 2.8334 - val_acc: 0.0550
Epoch 4/40
75/75 [=====] - 8s 110ms/step - loss: 2.8336 - acc: 0.0456 - val_loss: 2.8334 - val_acc: 0.0550
Epoch 5/40
75/75 [=====] - 8s 106ms/step - loss: 2.8332 - acc: 0.0612 - val_loss: 2.8334 - val_acc: 0.0550
Epoch 6/40
75/75 [=====] - 8s 108ms/step - loss: 2.8333 - acc: 0.0680 - val_loss: 2.8334 - val_acc: 0.0550
...
```

Рисунок Б.7 – Приклад виводу результату роботи моделі у консолі

Також результат виводиться у графічному вигляді рис. Б.8. На рисунку Б.8 зображено графічне представлення результату тренування згорткової нейромережі, а саме графік точності та функції втрат.

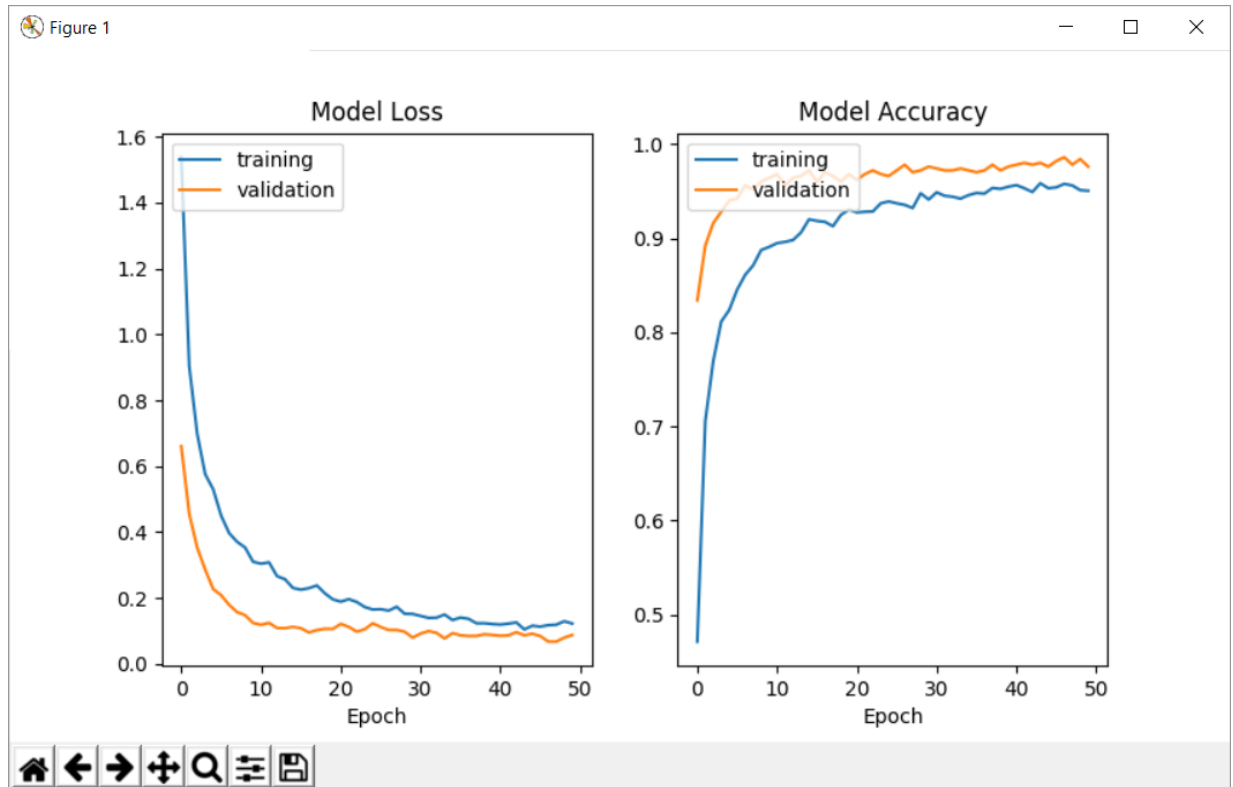


Рисунок Б.8 – Графічне представлення результату тренування згорткової нейромережі

ДОДАТОК В
Текст програми

// Файл Main.py

```

        from tensorflow.python.keras.preprocessing.image import
ImageDataGenerator
        from tensorflow.python.keras.models import Sequential
        from tensorflow.python.keras.layers import Conv2D,
MaxPooling2D, ZeroPadding2D, Convolution2D
        from tensorflow.python.keras.layers import Activation,
Dropout, Flatten, Dense

        from tensorflow.python.keras import optimizers
        from tensorflow.python.keras import models
        import tensorflow as tf
        from tensorflow.keras.applications import VGG16
        from tensorflow.keras.preprocessing import image
        from tensorflow.keras.models import Model, Sequential
        from tensorflow.keras.layers import Dense, Flatten, Dropout
        from tensorflow.keras import backend as K
        from tensorflow.keras.preprocessing.image import
ImageDataGenerator
        from tensorflow.keras.layers import Input
        from tensorflow.keras.callbacks import ModelCheckpoint ,
EarlyStopping
        # Каталог с данными для обучения
        train_dir = 'C:/nn/train'
        # Каталог с данными для проверки
        val_dir = 'C:/nn/val/'
        # Каталог с данными для тестирования
        test_dir = 'C:/nn/test'
        # Размеры изображения
        img_width, img_height = 224, 224
        # Размерность тензора на основе изображения для входных
данных в нейронную сеть
        # backend Tensorflow, channels_last
        input_shape = (img_width, img_height, 3)
        # Количество эпох
        epochs = 15
        # Размер мини-выборки
        batch_size = 8
        # Количество изображений для обучения
        nb_train_samples = 600
        # Количество изображений для проверки
        nb_validation_samples = 204
        # Количество изображений для тестирования
        nb_test_samples = 10

        datagen1 = ImageDataGenerator(rescale=1. / 255,
rotation_range=180,
width_shift_range=0.8,height_shift_range=0.8,
```

```

shear_range=0.8,zoom_range=0.8,
                                horizontal_flip=True)
datagen = ImageDataGenerator(rescale=1. / 255)
In [4]:
train_generator = datagen.flow_from_directory(
    train_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode='categorical')
Found 204 images belonging to 17 classes.
In [5]:
val_generator = datagen.flow_from_directory(
    val_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode='categorical')
Found 204 images belonging to 17 classes.
In [6]:
test_generator = datagen.flow_from_directory(
    test_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode='categorical')
Found 204 images belonging to 17 classes.
In [7]:
#old_own
model = Sequential()
model.add(Conv2D(64, (3, 3), input_shape=input_shape))

model.add(Conv2D(64, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(3, 3)))

model.add(Conv2D(128, (3, 3)))
model.add(Conv2D(128, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(3, 3)))

model.add(Conv2D(256, (3, 3)))
model.add(Conv2D(256, (3, 3)))
model.add(Conv2D(256, (3, 3)))
model.add(Conv2D(256, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(3, 3)))

model.add(Conv2D(512, (3, 3),padding='same'))
model.add(Conv2D(512, (3, 3),padding='same'))
model.add(Conv2D(512, (3, 3),padding='same'))
model.add(Conv2D(512, (3, 3),padding='same'))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(3, 3),padding='same'))

```

```

model.add(Conv2D(512, (3, 3),padding='same'))
model.add(Conv2D(512, (3, 3),padding='same'))
model.add(Conv2D(512, (3, 3),padding='same'))
model.add(Conv2D(512, (3, 3),padding='same'))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(3, 3),padding='same'))

model.add(Dense(4096))
model.add(Flatten())
model.add(Dense(1000))
model.add(Activation('relu'))
model.add(Dropout(0.4))
model.add(Dense(17))
model.add(Activation('softmax'))

model.compile(optimizer=optimizers.RMSprop(lr=1e-10),
loss='categorical_crossentropy', metrics=['accuracy'])

history=model.fit_generator(
    train_generator,
    steps_per_epoch=nb_train_samples // batch_size,
    epochs=40,
    validation_data=val_generator,
    validation_steps=nb_validation_samples // batch_size)

scores = model.evaluate_generator(test_generator,
nb_test_samples // batch_size)
WARNING:tensorflow:From C:\ProgramData\Anaconda3\lib\site-
packages\tensorflow\python\ops\init_ops.py:1251: calling
VarianceScaling.__init__ (from tensorflow.python.ops.init_ops)
with dtype is deprecated and will be removed in a future
version.
Instructions for updating:
Call initializer instance with the dtype argument instead
of passing it to the constructor
Epoch 1/40
75/75 [=====] - 17s 221ms/step -
loss: 2.8335 - acc: 0.0629 - val_loss: 2.8334 - val_acc: 0.0550
Epoch 2/40
75/75 [=====] - 8s 110ms/step -
loss: 2.8334 - acc: 0.0680 - val_loss: 2.8334 - val_acc: 0.0550
Epoch 3/40
75/75 [=====] - 8s 106ms/step -
loss: 2.8333 - acc: 0.0646 - val_loss: 2.8334 - val_acc: 0.0550
Epoch 4/40
75/75 [=====] - 8s 110ms/step -
loss: 2.8336 - acc: 0.0456 - val_loss: 2.8334 - val_acc: 0.0550
Epoch 5/40
75/75 [=====] - 8s 106ms/step -
loss: 2.8332 - acc: 0.0612 - val_loss: 2.8334 - val_acc: 0.0550

```

```

Epoch 6/40
75/75 [=====] - 8s 108ms/step -
loss: 2.8333 - acc: 0.0680 - val_loss: 2.8334 - val_acc: 0.0550
Epoch 7/40
75/75 [=====] - 8s 109ms/step -
loss: 2.8332 - acc: 0.0527 - val_loss: 2.8334 - val_acc: 0.0550
Epoch 8/40
75/75 [=====] - 8s 109ms/step -
loss: 2.8334 - acc: 0.0646 - val_loss: 2.8334 - val_acc: 0.0550
Epoch 9/40
75/75 [=====] - 8s 107ms/step -
loss: 2.8333 - acc: 0.0578 - val_loss: 2.8334 - val_acc: 0.0550
Epoch 10/40
75/75 [=====] - 8s 108ms/step -
loss: 2.8334 - acc: 0.0544 - val_loss: 2.8334 - val_acc: 0.0550
Epoch 11/40
75/75 [=====] - 8s 109ms/step -
loss: 2.8334 - acc: 0.0561 - val_loss: 2.8334 - val_acc: 0.0550
Epoch 12/40
75/75 [=====] - 8s 112ms/step -
loss: 2.8336 - acc: 0.0510 - val_loss: 2.8334 - val_acc: 0.0550
Epoch 13/40
75/75 [=====] - 8s 107ms/step -
loss: 2.8332 - acc: 0.0663 - val_loss: 2.8334 - val_acc: 0.0550
Epoch 14/40
75/75 [=====] - 8s 107ms/step -
loss: 2.8332 - acc: 0.0629 - val_loss: 2.8334 - val_acc: 0.0550
Epoch 15/40
75/75 [=====] - 8s 107ms/step -
loss: 2.8335 - acc: 0.0527 - val_loss: 2.8334 - val_acc: 0.0550
Epoch 16/40
75/75 [=====] - 8s 107ms/step -
loss: 2.8333 - acc: 0.0507 - val_loss: 2.8334 - val_acc: 0.0550
Epoch 17/40
75/75 [=====] - 8s 107ms/step -
loss: 2.8333 - acc: 0.0561 - val_loss: 2.8334 - val_acc: 0.0550
Epoch 18/40
75/75 [=====] - 9s 124ms/step -
loss: 2.8331 - acc: 0.0629 - val_loss: 2.8334 - val_acc: 0.0550
Epoch 19/40
75/75 [=====] - 8s 107ms/step -
loss: 2.8336 - acc: 0.0561 - val_loss: 2.8334 - val_acc: 0.0550
Epoch 20/40
75/75 [=====] - 8s 108ms/step -
loss: 2.8333 - acc: 0.0578 - val_loss: 2.8334 - val_acc: 0.0550
Epoch 21/40
75/75 [=====] - 8s 105ms/step -
loss: 2.8332 - acc: 0.0629 - val_loss: 2.8334 - val_acc: 0.0550
Epoch 22/40
75/75 [=====] - 8s 107ms/step -
loss: 2.8333 - acc: 0.0595 - val_loss: 2.8334 - val_acc: 0.0550
Epoch 23/40

```

```

75/75 [=====] - 8s 107ms/step -
loss: 2.8331 - acc: 0.0612 - val_loss: 2.8334 - val_acc: 0.0550
Epoch 24/40
75/75 [=====] - 8s 108ms/step -
loss: 2.8333 - acc: 0.0659 - val_loss: 2.8334 - val_acc: 0.0550
Epoch 25/40
75/75 [=====] - 8s 107ms/step -
loss: 2.8334 - acc: 0.0595 - val_loss: 2.8334 - val_acc: 0.0550
Epoch 26/40
75/75 [=====] - 8s 105ms/step -
loss: 2.8331 - acc: 0.0476 - val_loss: 2.8334 - val_acc: 0.0550
Epoch 27/40
75/75 [=====] - 8s 108ms/step -
loss: 2.8334 - acc: 0.0676 - val_loss: 2.8334 - val_acc: 0.0550
Epoch 28/40
75/75 [=====] - 8s 107ms/step -
loss: 2.8333 - acc: 0.0599 - val_loss: 2.8334 - val_acc: 0.0550
Epoch 29/40
75/75 [=====] - 8s 103ms/step -
loss: 2.8332 - acc: 0.0676 - val_loss: 2.8334 - val_acc: 0.0550
Epoch 30/40
75/75 [=====] - 8s 104ms/step -
loss: 2.8334 - acc: 0.0616 - val_loss: 2.8334 - val_acc: 0.0550
Epoch 31/40
75/75 [=====] - 8s 105ms/step -
loss: 2.8334 - acc: 0.0625 - val_loss: 2.8334 - val_acc: 0.0550
Epoch 32/40
75/75 [=====] - 8s 103ms/step -
loss: 2.8335 - acc: 0.0629 - val_loss: 2.8334 - val_acc: 0.0550
Epoch 33/40
75/75 [=====] - 8s 104ms/step -
loss: 2.8334 - acc: 0.0612 - val_loss: 2.8334 - val_acc: 0.0550
Epoch 34/40
75/75 [=====] - 8s 105ms/step -
loss: 2.8332 - acc: 0.0510 - val_loss: 2.8334 - val_acc: 0.0550
Epoch 35/40
75/75 [=====] - 8s 103ms/step -
loss: 2.8330 - acc: 0.0714 - val_loss: 2.8334 - val_acc: 0.0550
Epoch 36/40
75/75 [=====] - 8s 108ms/step -
loss: 2.8333 - acc: 0.0561 - val_loss: 2.8334 - val_acc: 0.0550
Epoch 37/40
75/75 [=====] - 8s 109ms/step -
loss: 2.8333 - acc: 0.0557 - val_loss: 2.8334 - val_acc: 0.0550
Epoch 38/40
75/75 [=====] - 8s 110ms/step -
loss: 2.8335 - acc: 0.0582 - val_loss: 2.8334 - val_acc: 0.0550
Epoch 39/40
75/75 [=====] - 8s 112ms/step -
loss: 2.8332 - acc: 0.0709 - val_loss: 2.8334 - val_acc: 0.0550
Epoch 40/40

```

```

75/75 [=====] - 8s 107ms/step -
loss: 2.8335 - acc: 0.0595 - val_loss: 2.8334 - val_acc: 0.0550
print(scores)

test_work_dir = 'C:/nn/test_work_dir'
#test_work_dir = test_work_dir(rescale=1. / 255)

work_generator = datagen.flow_from_directory(
    test_work_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode='categorical')

import numpy as np
a=(np.argmax(model.predict(work_generator),axis=-1))

print(a)

#model.save('testsavel2'+str(epochs)+'.h5')
model.save('save_own_dataaug6'+str(epochs)+'-vgg_like.h5')

label_map=(train_generator.class_indices)
print (label_map)

```

ДОДАТОК Д
Слайди презентації

Національний університет «Запорізька політехніка»

**ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ РОЗПІЗНАВАННЯ МЕДИЧНИХ
УПАКОВОК**

RESEARCH AND SOFTWARE IMPLEMENTATION OF MEDICAL PACKAGING RECOGNITION METHODS

Виконала:
студент групи КНТ-220М

Д.А. Гончаренко

Керівник:
професор, д.т.н.

А.О. Олійник

Рисунок Д.1 – Слайд 1

Мета, об'єкт, предмет дослідження

Мета роботи – дослідження, модифікація та програмна реалізація методів розпізнавання медичних упаковок.

Об'єкт дослідження – процес розпізнавання медичних упаковок.

Предмет дослідження – методи для розпізнавання медичних упаковок та класифікації їх за назвою препарату та дозуванням.

Рисунок Д.2 – Слайд 2

Постановка завдання



$$A_{fin} = f(x_{light}, x_{distance}, x_{quality}, x_{foreshortening}, x_{quantity}, x_{focus}, x_{background}),$$

де A_{fin} – показник якості передбачення нейромережі;

x_{light} – вплив світла;

$x_{distance}$ – вплив дистанції до об'єкту;

$x_{quality}$ – вплив якості фотографії;

$x_{foreshortening}$ – вплив ракурса з якого було зроблено фото;

$x_{quantity}$ – вплив кількості фотографій;

x_{focus} – вплив фокусування на об'єкті;

$x_{background}$ – вплив заднього фону.

Рисунок Д.3 – Слайд 3

Порівняльна характеристика середовищ розробки

Критерій порівняння	C++	Java	Python
Статична типізація	+	+	-
Динамічна типізація	-	-	+
Явна типізація	+	+	+/-
Неявна типізація	+/-	-	+
Неявне приведення типів без втрати даних	+	-	-
Неявне приведення типів із втратою даних	+	-	-
Неявне приведення типів у неоднозначних ситуаціях	+	-	-
Аліаси типів	+	-	X
Висновок типів змінних з ініціалізатора	+/-	-	X
Висновок типів змінних з використання	+/-	-	X
Висновок типів-аргументів при виклику методу	+	+	X
Параметричний поліморфізм	-	-	X
Інформація про типи в runtime	+	+	+
Кросплатформеність	+	+	+
Наявність автодоповнення	+	+	-
Наявність автоматичного закриття дужок, брекєтів	+	+	-
Підтримка репозиторіїв	+	+	-
Наявність «plugin manager»	+	+	+
Простота встановлення	+	+	+
Підтримка аналізу коду	+	+	+
Підтримка рефакторингу коду	+	+	+
Наявність інтерактивної консолі	+	+	+
Зручний користувацький інтерфейс	++	+	+
Швидкість модернізації	++	+	+

Рисунок Д.4 – Слайд 4

Порівняльна характеристика середовищ розробки

Критерій порівняння	C++	Java	Python
Статична типізація	+	+	-
Динамічна типізація	-	-	+
Явна типізація	+	+	+/-
Неявна типізація	+/-	-	+
Неявне приведення типів без втрати даних	+	-	-
Неявне приведення типів із втратою даних	+	-	-
Неявне приведення типів у неоднозначних ситуаціях	+	-	-
Адапти типів	+	-	X
Висновок типів змінних з ініціалізатора	+/-	-	X
Висновок типів змінних з використання	+/-	-	X
Висновок типів-аргументів при виклику методу	+	+	X
Параметричний поліморфізм	-	-	X
Інформація про типи в <i>runtime</i>	+	+	+
Кросплатформеність	+	+	+
Наявність автодоповнення	+	+	-
Наявність автоматичного закриття дужок, брекєтів	+	+	-
Підтримка репозиторіїв	+	+	-
Наявність «plugin manager»	+	+	+
Простота встановлення	+	+	+
Підтримка аналізу коду	+	+	+
Підтримка рефакторингу коду	+	+	+
Наявність інтерактивної консолі	+	+	+
Зручний користувацький інтерфейс	++	+	+
Швидкість модернізації	++	+	+

Рисунок Д.5 – Слайд 5

Структурна схема програми



Рисунок Д.6 – Слайд 6

Аналіз методів розпізнавання

Назва	Час навчання, ч	Точність розрахункова, %	Точність практична, %	Швидкість праці на робочому місці, с
1	2	3	4	5
Віола - Джонсон	-	63	37	7
CNN	6.3	87	60	7
Inceptionv2	12.8	94	84	6
RNN	7	65	63	5
LSTM	8	63	54	8
YOLO5	12	89	67	5
VGG16	10	92	73	6
Inceptionv3	21.1	96	90	7
VGG19	15.5	95	89	6

Рисунок Д.7 – Слайд 7

Аналіз методів розпізнавання

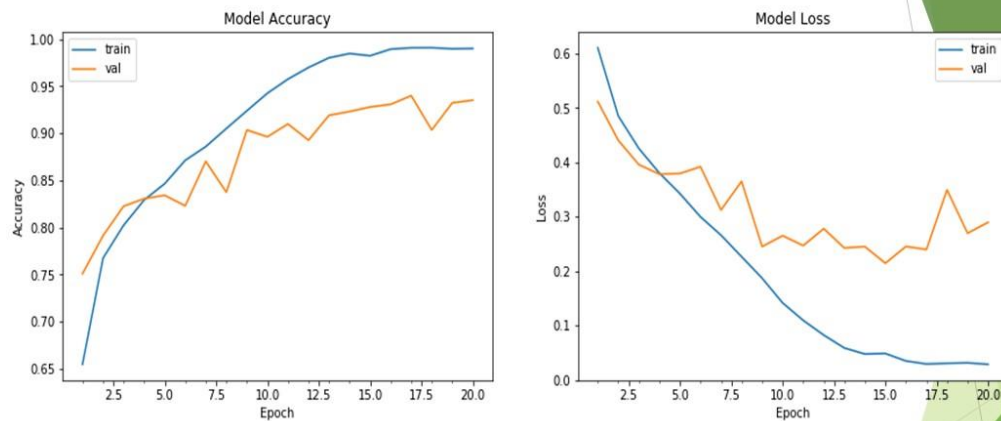


Рисунок П.1 – Графік точності та функції втрат згорткової нейромережі

Рисунок Д.8 – Слайд 8

Архітектура MBDC

Model: "MBDC"		
Layer (type)	Output Shape	Param #
Вхідний шар		
conv2d (Conv2D)	(None, 222, 222, 64)	1792
conv2d (Conv2D)	(None, 220, 220, 64)	36928
activation (Activation)	(None, 220, 220, 64)	0
max_pooling2d (MaxPooling)	(None, 73, 73, 64)	0
Повнозв'язковий шар 1		
conv2d (Conv2D)	(None, 71, 71, 128)	73856
conv2d (Conv2D)	(None, 69, 69, 128)	147584
activation (Activation)	(None, 69, 69, 128)	0
max_pooling2d (MaxPooling)	(None, 23, 23, 128)	0
Повнозв'язковий шар 2		
conv2d (Conv2D)	(None, 21, 21, 256)	295168
conv2d (Conv2D)	(None, 19, 19, 256)	590080
conv2d (Conv2D)	(None, 17, 17, 256)	590080
conv2d (Conv2D)	(None, 15, 15, 256)	590080
activation (Activation)	(None, 15, 15, 256)	0
max_pooling2d (MaxPooling)	(None, 5, 5, 256)	0

Повнозв'язковий шар 3		
conv2d (Conv2D)	(None, 5, 5, 512)	1180160
conv2d (Conv2D)	(None, 5, 5, 512)	2359808
conv2d (Conv2D)	(None, 5, 5, 512)	2359808
conv2d (Conv2D)	(None, 5, 5, 512)	2359808
activation (Activation)	(None, 5, 5, 512)	0
max_pooling2d (MaxPooling)	(None, 2, 2, 512)	0
Повнозв'язковий шар 4		
conv2d (Conv2D)	(None, 2, 2, 512)	2359808
conv2d (Conv2D)	(None, 2, 2, 512)	2359808
conv2d (Conv2D)	(None, 2, 2, 512)	2359808
conv2d (Conv2D)	(None, 2, 2, 512)	2359808
activation (Activation)	(None, 2, 2, 512)	0
max_pooling2d (MaxPooling)	(None, 1, 1, 512)	0
Шар для розподілу параметрів, який не є розрахунковим		
dense (Dense)	(None, 1, 1, 4096)	2101248
flatten (Flatten)	(None, 4096)	0
dense (Dense)	(None, 1000)	4097000
activation (Activation)	(None, 1000)	0
Шар «струсу» нейронів для виходу з локального мінімуму		
dense (Dense)	(None, 1000)	0
Вихідний шар		
dense (Dense)	(None, 215)	215215
activation (Activation)	(None, 215)	0
Total params: 26,437,847		
Trainable params: 26,437,847		

Рисунок Д.9 – Слайд 9

Порівняння нейромережи MBDC та MBDCGAM

Назва	Точність розрахункова, %	Час, ч	Точність практична, %	Швидкість праці на робочому місці, с
MBDC	93	8.2	92	4
MBDCGAM	97	11.1	98	4

Рисунок Д.10 – Слайд 10

Висновок

12

Результатом роботи стала розроблена модернізована модель, на основі базової архітектури, згорткової мережі для розпізнавання медичних упаковок. У модифікованому методі було запропоновано змінити архітектуру повнозв'язкових шарів від 2 до 4 згорткових шарів, при більше 4 згорткових шарів у повнозв'язному шарі час навчання збільшується, а точність падає. Так само було додано необхідну кількість нейронів для кожного шару у всіх повнозв'язних шарах, що було з'ясовано експериментальним шляхом. Було додано шар для розподілу параметрів. Від стандартного застосування наш шар має два шари активації, активаційну функцію.. Значне підвищення значення відсотку точності розпізнавання майже для усіх запропонованих для тестування наборах даних.

Можна побачити, що в ході модифікації НС ми досягли 92% точності розпізнавання. Це було викликано тим, що при використанні НС і MBDC було недостатньо умов для розпізнавання упаковок-близнюків, що і призвело до такої низької точності в практичному розпізнаванні. Модифікація MBDCGAM досягла 98% точності на реальній вибірці даних, з чого випливає, що ми досягли успіху і ГА виправдав свої очікування.

Рисунок Д.11 – Слайд 11

Апробація роботи.

3

Основні положення і результати роботи були представлені та обговорювалися на міжнародних конференціях (котрі входять до міжнародної наукометричної бази **Scopus**) 11 - праць:

- IEEE International Scientific-Practical Conference: Problems of Infocommunications Science and Technology, PIC S and T (Kharkiv, 2018, 2020, 2021, Kyiv, 2019));
- 15th International Conference on the Experience of Designing and Application of CAD Systems, CADSM 2019 (Polyana, , Ukraine, 2019);
- 10th International Conference on Advanced Computer Information Technologies, ACIT 2020 (Deggendorf, Germany, 2020 p.);
- International Workshop : Informatics and Data-Driven Medicine ((IDDM), Lviv, 2019, Sweden, 2020);
- Information Technology and Interactions (Kyiv, 2020).

Маю розділ у колективній монографії Lecture Notes on Data Engineering and Communications Technologies [this link is disabled](#) (Germany, 2021), входить до міжнародної наукометричної бази Scopus.

Рисунок Д.12 – Слайд 12