МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Кафедра прикладної математики та моделювання складних систем

Допущено до захисту
Завідувач кафедри ПМ та МСС
_____ доц. Коплик І.В.
(підпис)
«__» _____ 2020 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня «бакалавр»

спеціальність 113 «Прикладна математика»
освітньо-професійна програма «Прикладна математика»

тема роботи «**ОПТИМІЗАЦІЯ НЕЙРОМЕРЕЖЕВОГО АЛГОРИТМУ ІДЕНТИФІКАЦІЇ ЗОБРАЖЕНЬ**»

Виконавець
студент факультету ЕлІТ
__Пільгуй Ірина Іванівна___
(прізвище, ім'я, по батькові) (підпис)

Науковий керівник
___доц. к.ф.-м.н_____
(науковий ступінь, вчене звання)
__Князь Ігорь Олександрович_ _____
(прізвище, ім'я, по батькові) (підпис)

Суми – 2020 р.

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Факультет	**електроніки та інформаційних технологій**
Кафедра	**прикладної математики та моделювання складних систем**
Рівень вищої освіти	_____перший_____ (перший (бакалавр) або другий (магістр))
Галузь знань	**11 Математика та статистика**
Спеціальність	**113 Прикладна математика**
Освітня програма	**освітньо-професійна «Прикладна математика»**

ЗАТВЕРДЖУЮ
Завідувач кафедрою ПМтаМСС
доц. Коплик І.В._____
«___» _____2020р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

_____Пільгуй Ірина Іванівна_____
(прізвище, ім'я, по батькові)

1. Тема роботи ___Оптимізація нейромережевого алгоритму ідентифікації зображень_____

Керівник роботи __Князь Ігор Олександрович доцент кафедри ПМтаМСС_
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання, посада)

затверджено наказом по факультету ЕлІТ від «__» ___2020р. № _____

2.Термін подання роботи студентом «___»_____ 2020 р.

3. Вихідні данні до роботи_____
_____

4.Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)__аналітичний огляд,_методика дослідження, реалізація комп'ютернеого експерименту_____

5.Перелік графічного матеріалу

_____

6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1			
2			
3			

7. Дата видачі завдання «__» _____2020р.

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання роботи	Примітка
1	Розробка програмного забезпечення		
2	Аналіз результатів		
3	Написання документації		
4	Оформлення результатів роботи		

Здобувач вищої освіти  ____Пільгуй І. І.___

(підпис) (прізвище та ініціали)

Керівник роботи _____ ___Князь І. О._____

(підпис) (прізвище та ініціали)

РЕФЕРАТ

Кваліфікаційна робота: 70с., 11 рисунків, 1 таблиці, 26 джерела.

Мета роботи: розробити математичну модель та виконати її практичну реалізацію у вигляді програмного додатку для розпізнавання математичних виразів, дослідити процес побудови згорткової нейронної мережі, процес сегментації та структурного аналізу математичного виразу.

Об'єкт дослідження: оптичне розпізнавання символів.

Предмет дослідження: математичні моделі та методи розпізнавання об'єктів на зображенні.

Методи навчання: моделювання процесу розпізнавання об'єктів, навчання нейронної мережі.

У даній роботі було розглянуто чотири ключові проблеми в розпізнаванні математичних виразів: виявлення виразу, розпізнавання символів, аналіз розміщення символів та побудови виразу. Особливий інтерес представляє проблема машинного навчання, що складається з побудови зготкової нейронної мережі з оптимальною кількістю шарів, оптимізації алгоритмів в системі розпізнавання математичних виразів.

Для отриманих рішень проведено сегментацію зображення за допомогою детектора границь Кенні, побудовано згорткову нейронну мережу, проведено навачання на HASYv2 датасеті, що містить 168233 екземплярів та 369 класів математичних символів, реалізовано процес структурного аналізу виразу та сформовано результат у вигляді MS Word документу.

Ключові слова: ОПТИЧНЕ РОЗПІЗНАВАННЯ СИМВОЛІВ, РОЗПІЗНАВАННЯ МАТЕМАТИЧНОГО ВИРАЗУ, ЗГОРТКОВА НЕЙРОННА МЕРЕЖА, МАШИНЕ НАВЧАННЯ, СТРУКТУРНИЙ

АНАЛІЗ МАТЕМАТИЧНОГО ВИРАЗУ, СЕГМЕНТАЦІЯ ЗОБРАЖЕННЯ.

ЗМІСТ

ВСТУП

З моменту винаходу друкарського верстату, суспільство ніколи не бачило такої суттєвої зміни у засобах поширення інформації. Використовуючи Інтернет, майже кожен в світі має доступ до архівів, повними інформації. Розширена доступність інформації має значні наслідки для суспільства, дає кожному з нас можливість просвічення у тій чи іншій галузі.

Винахід обчислювальних приладів та мережі дав новий поштовх до збільшення доступної інформації. У 2004 році Google запустив масштабний проект з оцифрування бібліотек - Google Books [12]. Ціль проекту полягала у тому, щоб представити всі книги у цифровому форматі, для можливості їх індексації та пошуку в Інтернеті. Через вісім років після початку проекту була додана підтримка для розпізнавання більше пятидесяти мов. Передовий метод аналізу макету сторінки були реалізовані для виявлення різних типів документів: романів, журналів, газет, зображень, книг [13]. Якщо Google доб'ється успіху в оцифруванні та розпізнаванні будь-якого друкованого документа, незалежно від його походження, то незабаром після цього інформація з усіх світових документів стане миттєво доступною на всіх мовах і для всіх людей у світі. Такий розвиток подій, прискорив б уже існуючий прогрес у світі до епохи набагато більшої освіченості та мудрості, ніж це було досі.

Досі однією з перепон до представлення всіх книг в електронному форматі є розпізнавання математичних виразів. Велика кількість наукових видань містить складні математичні формули і вирази, але більшість

існуючих систем розпізнавання не здатні представити коректний результат, так як для розпізнавання двовимірної структури формул необхідні інші методи. Рішення задачі розпізнавання математичного тексту могло б дати можливість для перекладу книг і статей в цифровий вигляд, а також можливість редагувати інформацію перед збереженням.

РОЗДІЛ 1 АНАЛІТИЧНИЙ ОГЛЯД

Автоматичне розпізнавання математичних формул давно признано складною задачею [17]. Задача потребує: знаходження математичного виразу на зображенні, аналізу структури виразу, його перетворення у мову математичної розмітки. В [14] для розпізнавання формул використовують впорядковану систему переписування графів. За допомогою цього методу були вирішені наступні проблеми: рекурсивна структура формули, критична залежність пріоритету оператора, глобальні контекстно-залежні невизначеності, і різні типи значимих просторових відносин, які важко визначити і розрізнити. Мінуси впорядкованої системи переписування графів полягають у тому що, тип знань, закодованих в системі, недостатній для обробки вкладених символів або символів що накладаються. У структурному аналізі відсутня можливість узагальнення для інших типів позначень, оскільки цей підхід критично залежить від унікальних характеристик підмножини математичних записів. В [15] запропоновано використовувати комерційний інструмент оптичного розпізнавання символів у якості класифікатору тексту, частини, які даний інструмент не спроможен розпізнати, додатково аналізували для виявлення математичних формул. Експериментально встановлена похибка даного підходу має значення приблизно 0.113. Автори статті [16] виконують розпізнавання друкованих математичних виразів з використанням двовимірних стохастичних контекстно-вільних граматик. У зазначеній роботі було проведено експеримент над загальнодоступною і великою базою даних, і результати були представлені з використанням чітко визначеної глобальної автоматичної оцінки продуктивності

математичного виразу. Результати показали, що структурна інформація виразу поліпшила розпізнавання символів. Остаточна частота появи помилок в тестовому наборі склала $4,68\%$. У статті [18] описаний підхід для автономного розпізнавання рукописних математичних символів з використанням глибинної нейронної мережі - SqueezeNet. Процес розпізнавання символів в цій статті включає в себе сегментацію символів і точну класифікацію для 101 класів. Похибка обчислення складає приблизно 0.1002. У роботі [19] пропонується новий метод парного змагального навчання для вивчення семантично-інваріантних функцій. Запропонована модель складається з системи розпізнавання на основі уваги і дискримінатора. У результаті розпізнавання математичних виразів, похибка розпізнавання складає 0.2084. У статті [20] представлений метод ідентифікації формул на основі класифікації з використанням методу опорних векторів. Метод спочатку розбиває текстові рядки на слова, а потім класифікує кожне слово на два класи, а саме формула або звичайний текст. Різні функції математичних виразів, включаючи геометричне розташування, символьний і контекстний вміст, використовуються для створення надійного і гнучкого класифікатора методу опорних векторів. Потім формули витягуються шляхом об'єднання слів, позначених як формули. Експериментальні результати показують похибку, що складає 0.1571 для запропонованого способу. У [21] автор використовує систему оптичного розпізнавання символів для ідентифікації математичних виразів на зображеннях документів. Кілька функцій, починаючи з низькорівневих елементів зображення, елементів форми, лінгвістичної інформації та інші, витягуються і об'єднуються для точного визначення виразів. В експерименті використовується набір даних з 200 загальнодоступних зображень, що містить 1163 вкладених і 1039 виразів,

що відображаються. Результати тестів показують точність 88,3% та 97,2% відповідно для ідеального вилучення виразів.

РОЗДІЛ 2 МЕТОДИКА ДОСЛІДЖЕНЬ

2.1 Постановка задачі

Задача розпізнавання математичної формули - це задача передбачення послідовності. Нехай (x, y) - пара послідовностей зображення. $x \varepsilon R_{[H \times W]}$ - зображення в градації сірого, з висотою H та шириною W, $y = [y_1, y_2, ..., y_T]$ - довірча послідовність, яка містить T ознак, що позначають математичну формулу на зображенні. x може бути відтворено за допомого y з використанням стандартного компілятора TeX (мова розмітки даних спеціального призначення, яка є основним ядром системи набору публікацій, зокрема для набору математичних та інших технічних текстів). Ціль завдання полягає у тому щоб відтворити y при умові вводу зображення x, тобто потрібно знайти функцію відображення f, таку що $f(x) \rightarrow y$. Враховуючи набір з N основоположних зображень, таких що кожне з зображень описується парою $\{x^i, y^i\}_{i=1}^{N}$, використовуємо контрольоване навчання для побудови функції передбачення послідовності $\widehat{f}$, яка наближається до f. Під час тесту використовуємо $\widehat{f}(x) \rightarrow \widehat{y}$, щоб передбачити послідовність $\widehat{y}$, яка відбудовує вхідне зображення x. Оцінка виконується шляхом вимірювання подібності між $\widehat{y}$ та довірчою послідовністю y, а також між візуалізовуваним зображенням $\widehat{x}$ та основоположним зображенням x.

2.2 Сегментація

Одним з основних задач обробки та аналізу зображення є сегментація. Задача сегментації полягає у пошуку і витягу груп пікселей, кожен з яких характеризує один змістовий об'єкт. Результатом сегментації зображення є множина сегментів, які разом покривають все зображення, або множина контурів, виділених з зображення. Проблема сегментації зображення, у комп'ютерному зорі, є однією з найстаріших і широко досліджуваних проблем [11].

Існує декілька основних підходів для виділення математичних символів - виявлення контурів та виділення обмежуючої рамки.

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_0^x e^{-\frac{z^2}{2}} dz$$

Рис. 2.2.1 − Результат сегментації зображення методом виділення обмежуючої рамки

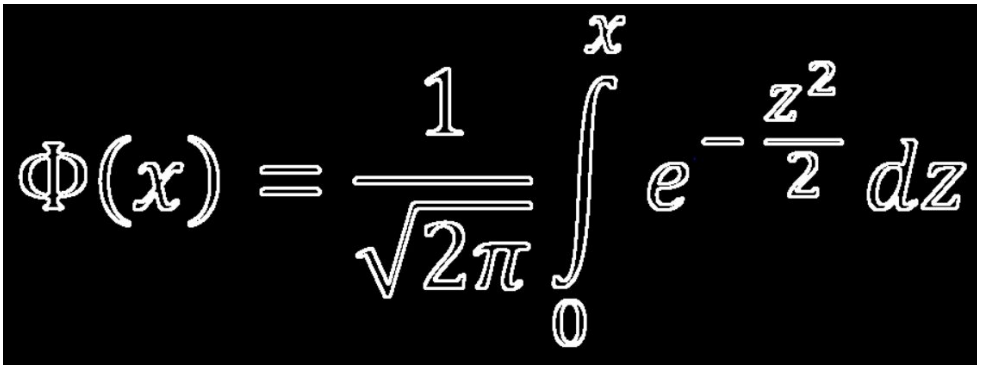


Рис. 2.2.2 − Результат сегментації зображення методом виявлення контурів

Отже, для задачі сегментації зображення, що містить математичні символи, доречно використовувати метод виявлення контурів. Нижче розглянуто методи виявлення контурів.

Оператор Робертса. Оператор Робертса - один з ранніх алгоритмів виділення меж, який обчислює на плоскому дискретному зображенні суму квадратів різниць між діагонально суміжними пікселями. Для коректних результатів кількість шумів на зображенні має бути мінімальною та добре виділенні границі об'єктів. В основі метода Робертса лежить згортування зображення за допомогою наступних ядер:

$$\begin{bmatrix} +1 & 0 \\ 0 & -1 \end{bmatrix} \quad \begin{bmatrix} 0 & +1 \\ -1 & 0 \end{bmatrix} \tag{2.2.1}$$

G_x G_y

тоді градієнт має вигляд:

$$G(x,y) = \sqrt{G_x^2 + G_y^2} \tag{2.2.2}$$

де $G_x(x, y)$ - точка зображення згорнутого першим ядром, $G_y(x, y)$ - точка зображення згорнутого другим ядром.

Напрямок градієнта визначається як:

$$\Theta(x,y) = arctan(G_y/G_x) \tag{2.2.3}$$

Основна перевага оператора Робертса полягає у простоті цього методу, але при великій кількості шумів на зображенні коректність результатів дуже знижується [1].

Лапласіан Гаусіана - один з найбільш поширених детекторів меж заснований на Лапласіані Гауссіани. Метод виконується за два кроки. На першому кроці відбувається згладжування зображення, а потім

обчислюється функція Лапласа, що призводить до утворення подвійних контурів. Визначення контурів зводиться до знаходження нулів на перетині подвійних кордонів. Тобто, вхідне зображення $f(x, y)$ згортається за допомогою ядра Гаусса:

$$g(x, y, t) = \frac{1}{2\pi t} e^{-\frac{x^2 + y^2}{2t}} \tag{2.2.4}$$

в масштабі t, щоб отримати повномасштабне уявлення простору

$$L(x, y; t) = g(x, y, t) \cdot f(\text{x}, \text{y}) \tag{2.2.5}$$

Потім обчислюється результат застосування оператора Лапласа:

$$\nabla^2 L = L_{xx} + L_{yy} \tag{2.2.6}$$

який зазвичай призводить до сильних позитивних відгуків для темних плям радіусу $r = \sqrt{2t}$ і сильним негативним відгуком для яскравих плям аналогічного розміру [3]. Комп'ютерна реалізація функції Лапласа зазвичай здійснюється через наступну маску [2]:

0	-1	0
-1	4	-1
0	-1	0

G_x

-1	-1	-1
-1	8	-1
-1	-1	-1

G_y

(2.2.7)

Детектор кордонів Кенні - багаторівневий алгоритм в основі якого лежать наступні кроки [4]:

a) Згладжування - розмиття зображення для видалення шуму;
b) Пошук градієнту - межі відзначаються там, де градієнт зображення набуває максимальне значення;

c) Заглушення не максимумів. Тільки локальні максимуми відзначаються як кордони;

d) Подвійна порогова фільтрація - потенційні межі визначаються порогами;

e) Трасування області неоднозначності - підсумкові межі визначаються шляхом стирання всіх країв, незв'язаних з певними межами.

Детектор використовує фільтр на основі першої похідної від Гаусіани ($\sigma = 1.4$):

$$\mathbf{B} = \frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix} * \mathbf{A}. \qquad (2.2.8)$$

У точках де градієнт набуває найбільшого значення відмічається границя об'єкта. Границі на зображенні можуть знаходитися у різних напрямках, тому використовується чотири фільтри для визначення горизонтальних, вертикальних і діагональних контурів. Скориставшись оператором виявлення кордонів, оператором Собеля:

-1	-2	-1
0	0	0
1	2	1

G_x

-1	0	1
-2	0	2
-1	0	1

G_y

(2.2.9)

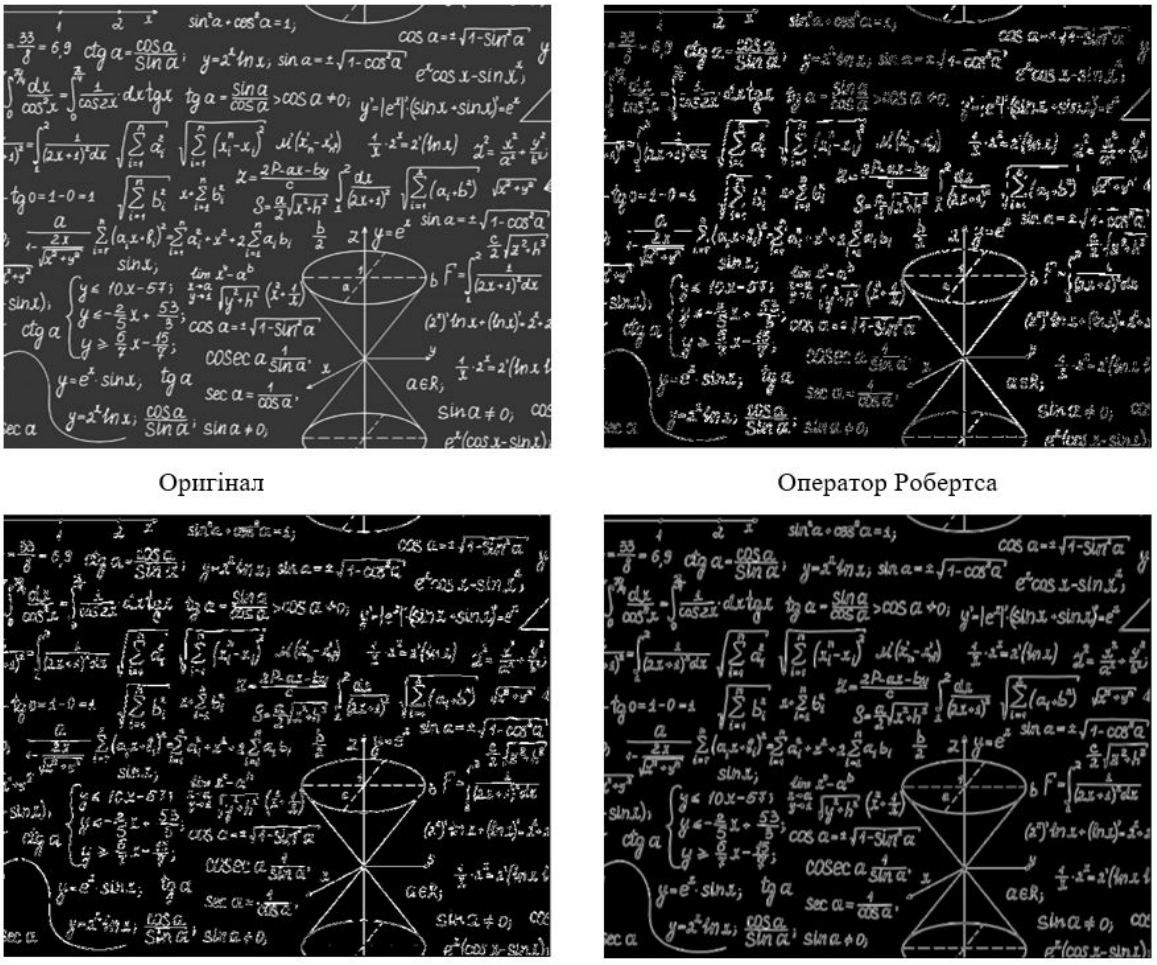
виходить значення для першої похідної в горизонтальному напрямку G_y і вертикальному напрямку G_x. З цього градієнта отримуємо кут напрямку межі:

$$\Theta = arctan(G_y/Gx) \qquad (2.2.10)$$

Кут нахилу градієнту округлюється до одного з чотирьох кутів, що представляють горизонталь, вертикаль та дві діагоналі, тобто набувають значень 0, 45, 90 або 135.

Далі відбувається перевірка умови досягнення локального максимуму градієнта у відповідному напрямку. Таким чином, виходить двійкове зображення, що містить межі [5].

Для порівняння вище описаних методів, проведемо 10 незалежних випробувань. Методи були реалізовані за допомогою мови програмування Python та бібліотеки ‘shimage’. Приклад оригінального зображення та результатів зображені на рисунку 2.2.3.



Оригінал | Оператор Робертса

Лапласіан Гаусіани | Метод Кенні

Рис. 2.2.3 − Результати роботи алгоритмів виділення контуру.

При аналізі отриманих результатів були виявлено, що детектор контурів Кенні у 10 з 10 випадків дає кращі результати ніж інші методи. Тому для сегментації вхідного зображення використовуємо метод Кенні.

2.3 Розпізнавання символів

Шаблонний метод розпізнавання символів, складається з двох етапів: першим етапом виконується перетворення вхідного зображення у растрове, далі виконується процес порівняння даного зображення з кожним зображенням, що знаходиться в шаблонах. Найбільш підходящим вважається шаблон, який має найменшу кількість точок, відмінних від вхідного зображення. Шаблон для кожного класу отримують шляхом усередненням зображень символів навчальної вибірки. До переваг даного алгоритму відносяться: простота реалізації, надійна робота в умовах відсутності перешкод, висока точність розпізнавання дефектних символів, швидкість при малому алфавіті. До недоліків можна віднести: сильну залежність від шаблонів і складність підбору оптимальних шаблонів, неможливість розпізнавання шрифту, що відрізняється від закладеного в систему, чутливість до обертання, шумів і спотворень [6].

Структурний метод розпізнавання. У цьому методі, вхідне зображення перетворюється у топологічне представлення, яке відображає інформацію про взаємне розташування структурних елементів символа. При розпізнаванні, символ проходить процес скелетизації. Кожен отриманий контур скелетного подання описується у вигляді послідовного набору особливих точок та ланцюгового коду, що складається з точки прив'язки, числа кодів та масиву направлень з поточної точки до

наступної. Особливі точки - це кінечні точки та точки розгалуження (тріоди), тобто точки, сусіди яких утворюють щонайменше три зв'язні області. Найчастіше ця інформація зберігається за допомогою графів [7]. Для знаходження типу контуру або топологічного коду, виконується обробка, яка складається з виділення коротких ліній, об'єднання тріодів, що розташовані поруч, видалення малих внутрішніх контурів. Далі контур записується у вигляді послідовного набору номерів особливих точок, відповідних обходу за годинниковою стрілкою. За допомогою перенумерації особливих точок та зміни початку контуру проводиться спроба ототожнення контура з одним з основних типів. Основною проблемою структурних методів розпізнавання є ідентифікація знаків, що мають дефекти. Перевагами методу є здатність розпізнавання спотворених символів і швидкодія при малому алфавіті [8].

Ознаковий метод розпізнавання ґрунтується на співставленні зображення та $N-$мірного вектора ознак, який порівнюється з набором еталонних векторів тієї ж розмірності. У системах розпізнавання символів найчастіше використовується класифікація, в основі якої лежить розрахунок евклідової відстані між векторами ознак символу та векторами ознак еталонного опису [6]. Кількість ознак та тип визначають якість розпізнавання. Формування вектора ознак виконується під час аналізу попередньо підготовленого зображення. Еталон для кожного класу отримується шляхом аналогічної обробки символів навчальної вибірки. Основні переваги методу заключаються у простоті реалізації, висока швидкість розпізнавання, хороша узагальнююча здатність. Недоліки методу – висока чутливість до дефектів зображення, на етапі виділення ознак відбувається втрата частини інформації про символ, інформація про

взаємне розташування елементів символів втрачається через те, що виділення ознак проходить незалежно [9].

Нейронні мережі полягають у моделюванні роботи мозку людини. На вхід заздалегідь навченої нейронної мережі надходить вектор, який є поданням вхідного образу. На виході нейрон, відповідного класу розпізнаного символу, видає максимальне значення функції активації. Основна різниця нейронної мережі від класичних методів розпізнавання символів полягає у тому, що нейронні мережі не програмуються а навчаються. Навчання один з основних переваг нейронних мереж перед традиційними алгоритмами. Крім того, цей метод має високу точність. До недоліків нейронних мереж можно віднести те, що розпізнавання потребує великої кількості навчальних даних [10].

У цій роботі представлено використання зготкової нейронної мережі для розпізнавання символів.

2.4 Нейрона мережа

Згорткова нейронна мережа (ЗНМ) - це клас глибинних штучних нейронних мереж прямого поширення, який застосовувався до аналізу візуальних зображень. Основна ідея згорткової мережі полягає в чергувані зготкових та субдискретизаційних шарів.

ЗНМ складається зі згорткових, агрегувальних та повноз'єднаних шарів. Згортковий шар - основний блок згорткової мережі. В операції згортання використовується обмежена матриця вагів (ядро згортки) невеликого розміру, яку переміщують вздовж шару, який обробляють,

формуючи після кожного сдвигу сигнал активізації для нейрона наступного шару з аналогічною позицією (див. рис. 2.4.1).

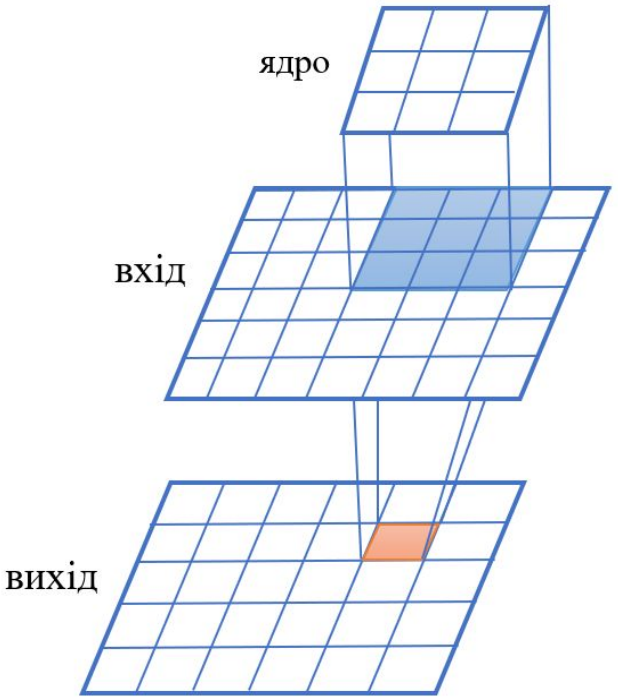


Рис. 2.4.1 − Принцип роботи згорткового шару.

Для різних нейронів використовують одне й те ж ядро згортки. Наступний шар, отриманий в результаті операції згортування ядра, показує наявність даної ознаки в шарі, який обробляється, та його координати, формуючи карту ознак.

Формула згортки має наступний вигляд:

$$x_{ij}^{l} = \sum_{a=-\infty}^{+\infty} \sum_{b=-\infty}^{+\infty} \omega_{ab}^{l} \cdot y_{(i\cdot s-a)(j\cdot s-b)}^{l-1} + b^{l} \qquad (2.4.1)$$

де $i = \overline{1,\ N}$, $j = \overline{1,\ M}$, a, b - індекси елементів у матрицях, s - величина кроку згортки, l та $l-1$ − індекси шарів мережі, x^{l} - результат операції згортання, x^{l-1} - вихід попередньої функції, або вхідного зображення

мережі, ω^{l} - ядро згортки, y^{l-1} - результат проходження x^{l-1} через функцію активізації, b^{l} - матриця зміщення.

Функція активізації використовується для уникнення лінійності мережі. Для великої кількості класів, щоб результат моделі відображав ймовірності цих класів, використовується нормована експоненційна функція (англ. softmax) :

$$f_{softmax} = y_i^l = f(x_i^l) = \frac{e^{x_i^l}}{\sum_{k=0}^{n} e^{x_k^l}} \qquad (2.4.2)$$

де n - кількість класів.

Агрегувальний шар дозволяє виділяти важливі особливості на картах ознак, дає інваріантність до знаходження об'єкта на карті, а також знижує розмірність карт, що дозволяє зменшити кількість часу витраченого на роботу мережі. Існує декілька функцій, що реалізують агрегування, серед яких найпоширенішою є максимізаційне агрегування. Максимізаційне агрегування розділяє вхідне зображення на набір прямокутників, для кожної підобласті знаходить максимум [14] (див. рис. 2.4.2).

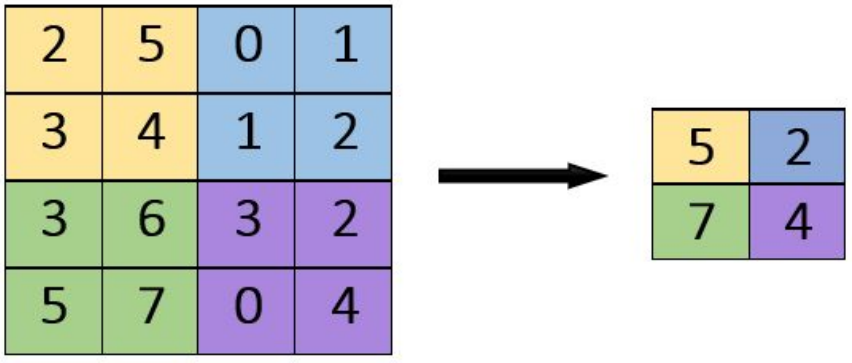


Рис. 2.4.2 − Приклад максимізаційного агрегування.

Після згорткових шарів отримуємо множину карт ознак. Карти з'єднуються в один вектор, який подається на вхід повноз’єднаного шару. Повноз’єднаний шар використовується для зв’язування об’єктів з одним класом з набору можливих класів. Як показано нижче, попередній активаційний шар повністю пов'язаний з кожним нейроном, присутнім в цьому шарі, вихідний сигнал є розподілом ймовірності для правдоподібності кожного класу.

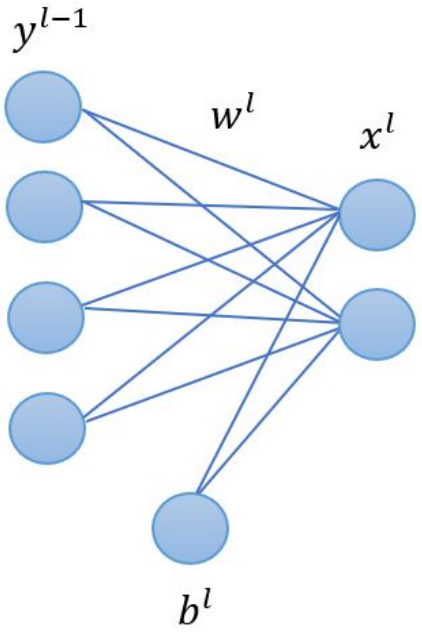


Рис. 2.4.3 − Приклад повноз’єданого шару.

Формула повноз'єднаного шару має вигляд:

$$x_i^l = \sum_{k=0}^{m} \omega_{ki}^l \cdot y_k^{l-1} + b_i^l, \quad i = \overline{0,\ n} \tag{2.4.3}$$

Матричний вигляд формули повноз’єднаного шару:

$$[x_0^l \quad x_1^l \quad \dots \quad x_n^l] = [y_0^l \quad y_1^l \quad \dots \quad y_n^l] \times$$
$$\times \begin{bmatrix} w_{00}^l & w_{01}^l & \dots & w_{0n}^l \\ w_{10}^l & w_{11}^l & \dots & w_{1n}^l \\ \dots & \dots & \dots & \dots \\ w_{m0}^l & w_{m1}^l & \dots & w_{mn}^l \end{bmatrix} + [b_0^l \quad b_1^l \quad \dots \quad b_n^l] \tag{2.4.4}$$

Останній етап - функція, яка оцінює якість роботи мережі. Для визначення шару витрат будемо використовувати формулу перехресної ентропії:

$$E = -\sum_{i=0}^{n} y_i^{trust} \cdot ln(y_i^l) \qquad (2.4.5)$$

де n - кількість класів, y^l - вихід моделі, y^{trust} - правильні відповіді.

2.5 Тренування моделі

У навчанні нейронної мережі є специфічні обмеження, що виділяють навчання нейромереж із загальних задач оптимізації: необхідність високого паралелізму при навчанні, астрономічне число параметрів, багато критеріально вирішуваних завдань, необхідність знайти досить широку область, в якій значення всіх функцій, що мінімізуються близькі до мінімальних. Стосовно решти проблему навчання можна, як правило, сформулювати як завдання мінімізації оцінки.

Тренування моделі будемо виконувати за допомогою методу зворотного поширення помилки. Це ітеративний метод обрахунку градієнта, основною ідеєю якого є поширення сигналів помилки від виходів мережі до її входів, в напрямку, зворотному прямому поширенню сигналів у звичайному режимі роботи, з метою мінімізації помилки роботи мережі та отримання бажаного виходу.

Формула для зворотного поширення помилки через функцію втрат має вигляд:

$$\frac{\partial E}{\partial y_i^l} = -\frac{y_i^{trust}}{y_i^l}, \quad \forall i \in (0, ..., n) \qquad (2.5.1)$$

Формула зворотного поширення через функцію активізації:

$$\frac{\partial E}{\partial x_i^l} = \sum_{k=0}^{n} \frac{\partial E}{\partial y_k^l} \frac{\partial y_k^l}{\partial x_i^l} \quad , \quad \forall i \in (0, ..., n) \tag{2.5.2}$$

Формула зворотного поширення для оновлення матриці вагів ω^l повнозєданної мережі:

$$\frac{\partial E}{\partial \omega_{ki}^l} = \delta_i^l \cdot y_k^{l-1} \quad , \quad \forall i \in (0, ..., n) \quad \forall k \in (0, ..., m) \tag{2.5.3}$$

де $\delta_i^l = \frac{\partial E}{\partial x_i^l}$.

Формула зворотного поширення для оновлення матриці зміщення b^l:

$$\frac{\partial E}{\partial b_i^l} = \delta_i^l \, , \quad \forall i \in (0, ..., n) \tag{2.5.4}$$

Формула зворотного поширення через y^{l-1}:

$$\frac{\partial E}{\partial y_k^{l-1}} = \sum_{i=0}^{n} \delta_k^l \cdot \omega_{ki}^l \, , \quad \forall k \in (0, ..., m) \tag{2.5.5}$$

Формула зворотного поширення через шар згортки:

$$\frac{\partial E}{\partial y_{ij}^{l-1}} = \sum_{i^*} \sum_{j^*} \frac{\partial E}{\partial y_{i^*j^*}^{l}} \frac{\partial y_{i^*j^*}^{l}}{\partial x_{i^*j^*}^{l}} \cdot \omega_{(i^*s-i)(j^*s-j)}^{l} \qquad (2.5.6)$$

$\forall i, j \in$ розмірність матриці y^{l-1} .

2.6 Структурний аналіз

У математичних виразах, символи можуть бути просторово розташовані як складна двовимірна структура. Правильно розташовані символи утворюють ієрархічну структуру. Однак правильне групування символів у математичному виразі не є тривіальним. По-перше, існує декілька типів символів. Один тип містить у собі всі основні символи а інший включає у себе символи прив’язки, розділові та оператори. Кожен з типів має свої спеціальні критерії групування. По-друге, існує декілька типів операторів, а саме явні та неявні. Явні оператори - це символи операторів, тоді як неявні оператори - просторові оператори. По-третє, деякі символи можуть мати різні значення у різних контекстах. Отже, все це робить процес розпізнавання складним навіть коли всі окремі символи можуть бути розпізнані корректно.

Для виконання структурного аналізу, необхідно мати список об’єктів з відповідними атрибутами, такими як розташування, розмір та символ. Для відбудови математичного виразу будується ієрархічна структура об'єктів. Ієрархічна структура може бути представлена у вигляді абстрактного синтаксичного дерева або дерева відносин. Деякі вузли дерева можуть бути відсутні у списку об'єктів, отриманих з попередньої фази через існування просторових операторів у математичних виразах.

У [22] автор запропонував алгоритм, який використовує ідеї пріоритетності та домінування оператора. Він складається з двох основних етапів: групування послідовностей операторів та побудова структурного дерева. Дана схема може бути застосована лише до моделей, структури яких базуються на ряді операторів.

У статті [23] запропоновано метод автоматичного маркування просторових відносин між символами. Метод розроблений для того, щоб приймати судження до розпізнавання символів. Іншими словами, він використовує лише інформацію про обмежувальні рамки символів. Використання просторових відносин, не знаючи символів, може бути недостатньою для деяких випадків. Насправді Ван і Фаур вказали, що окремі обмежувальні рамки в деяких випадках неоднозначні з точки зору виявлення просторових зв’язків між символами. Тим не менш, робота, корисна для вирішення просторових зв’язків між неоднозначними або невідомими символами. У [24] Ван і Фаур розробили модульну систему для розпізнавання рукописних математичних виразів, будуючи дерево відношень. В основі лежать два модулі, а саме модуль, керований даними та модуль, керований знаннями. Однією особливістю системи є те, що вона буде працювати навіть тоді, коли деякі символи виразу не вдається розпізнати.

У [25] автори використовували графіко-переписуючий підхід для розпізнавання математичних виразів. Модель включає наступні етапи: побудова ребер для відображення просторових зв’язків між символами, ранжування, що використовує інформацію про перевагу оператора перед груповими символами в підвираженнях, інтерпретування підвиразів. Крім того, система використовує знання про нотаціональні конвенції, такі як

пріоритет оператора та діапазон операторів, щоб усунути необхідність зворотного відстеження.

Запропонований метод - деяка комбінація підходів описаних вище, в цілому ідея полягає в тому, щоб використовувати як просторову інформацію про символ і його сусідів, так і семантичні відносини між ними. Перед початком знаходимо всі входження верхніх та нижніх індексів, великих операторів, дужок, функцій.

Для знаходження верхнього та нижнього індексу будемо порівнювати розмір та просторове розташування символу відносно попереднього символу (див. рис. 2.6.1)

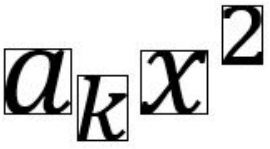


Рис. 2.6.1 − Розташування індексів відносно основних символів

Для знаходження функції будемо проводити пошук у виразі неперервної послідовність символів $b = a_1 a_2 ... a_n$, таких що $b \in$ { *"arccotan", "Arccotan", "Arsech", "cosech", "Arcoth", "Artanh", "Arcosh", "artanh", "arcosh", "Arsinh", "arcsch", "arsinh", "Arcsch", "arcsin", "arccos", "arctan", "arcctg", "arccot", "arccsc", "arcsec", "Arcsin", "Arccos", "Arctan", "Arcctg", "Arccot", "Arcsec", "Arccsc", "arsech", "arcoth", "Arcth", "cotan", "cosec", "arsch", "arctg", "Arctg", "Arsch", "arcth", "tanh", "coth", "csch", "sinh", "arsh", "arch", "arth", "sech", "Arsh", "Arch", "Arth", "cotg", "cosh", "csc", "sec", "ctn", "cth", "cot", "tan", "sch", "ctg", "cos", "sin", "exp", "log", "Log", "det", "Res", "lim", "max", "min", "sup", "inf", "th", "ch", "tg", "sh", "ln", "Ln", "Lg", "Re", "Im", "lg"* }.

Операторами будемо вважати символ що належить множині {"∏", "∑", "√", "∫", "∯"}.

Дужками будемо вважати символ $a_i \in$ { "|", "(", "[", "{", "‖", "⌊", "⌈", "⟨" } та відповідний йому символ $a_j \in$ { "|", ")", "]", "}", "‖", "⌋", "⌉" "⟩" }, при $i < j$, $i = \overline{1, N}, j = \overline{2, N}$. Для їх знаходження будемо використовувати структуру даних стек. Алгоритм пошуку дужок: при проходженні по виразу перевіряємо кожен символ, чи є він відкриваючою або закриваючою дужкою. Якщо поточний символ відкриваюча дужка, кладемо її у стек, якщо закриваюча то видаляємо відкриваючу дужку. Процес виконується поки стек не будем порожній, при умові корректного початкового виразу.

Отже, розглянемо процес побудови ієрархічної структури. Алгоритм структурного аналізу має наступний вигляд. На вхід методу getBuildedFormula (додаток) подається два цілих, невід'ємних числа, індекс початку та індекс кінця виразу. Далі знаходимо всі входження попередньо знайдених елементів, таких як дужки, великі оператори, функція, верхній та нижній індекси. Кожен з цих елементів відокремлюємо у класи для побудови ієрархічної структури формули. Будемо відокремлювати наступні класи:

- TextElement - неперервна послідовність символів виразу, яка не містить у собі інші класи. Найнижчий рівень ієрархічної структури.
- ParenthesesElement - класс у якому міститься тип дужок та елементи, що знаходяться між відкриваючою та закриваючою дужкою. Елементи можуть бути складними, тобто можуть складатися з інших класів.

- FunctionElement - містить назву функції, індекси, якщо вони є, та аргумент функції. Аргумент може бути складним.
- ScriptElement - клас, що відповідає за індекси, містить у собі тип індексу (верхній чи нижній), індекс та основний елемент до якого належить індекс. Індекс та основний елемент можуть бути складними.
- LargeOperatorElement - містить символ оператора, індекси та аргумент. Індекси та аргументи можуть бути складними.

Згрупувавши всі символи виразу у класи можемо отримати ієрархічну структуру. Приклад розподілення математичного виразу на класи зображено на рисунку 2.6.2.

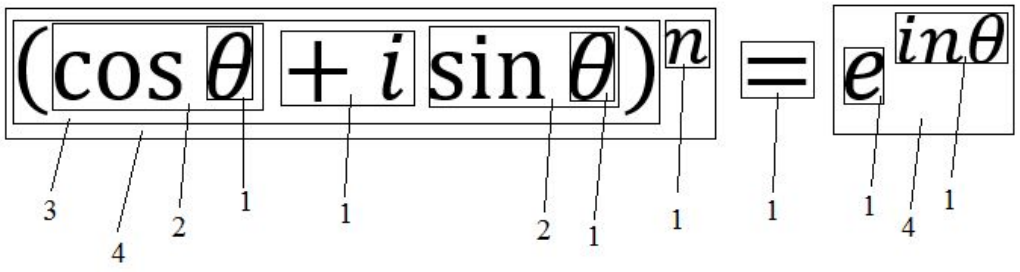


Рис. 2.6.2 − Приклад розподілення математичного виразу на класи.

1 - клас TextElement; 2 - клас FunctionElement;

3 - клас ParenthesesElement; 4 - клас ScriptElement.

РОЗДІЛ 3 РЕАЛІЗАЦІЯ КОМП'ЮТЕРНОГО ЕКСПЕРИМЕНТУ

3.1 Методика численного експерименту та результат

У ході роботи було реалізовано згорткову нейронну мережу, яка має наступну архітектуру:

Шар	Число ядер (нейронів)	Розмір
Згортковий	5	(2, 2)
Агрегувальний	-	(2, 2)
Згортковий	20	(3, 3)
Повнозв’язний	2000	-
Повнозв’язний	2000	-

Табл. 3.1.1 - Архітектура нейронної мережі

Мережа була навчена за допомогою HASYv2, що містить у собі 168233 екземплярів 369 класів [26]. Результат на тестовій виборці склав 97% .

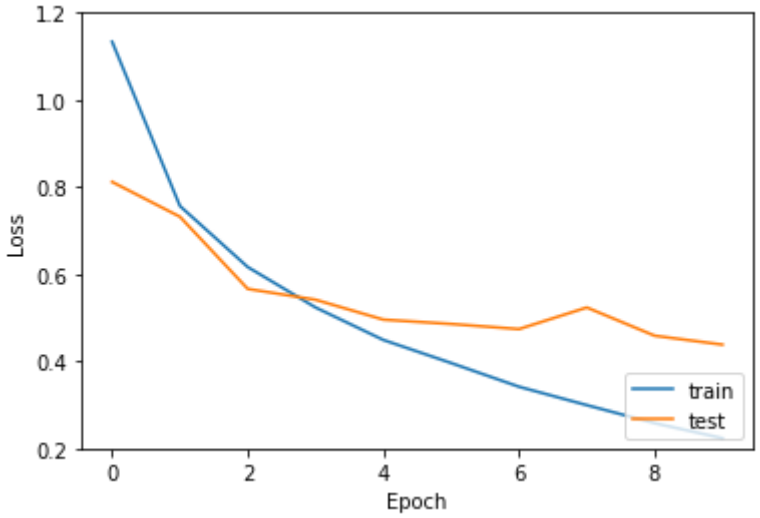


Рис. 3.1.1 - Графік втрат моделі при навчанні та тестуванні в залежності від епохи

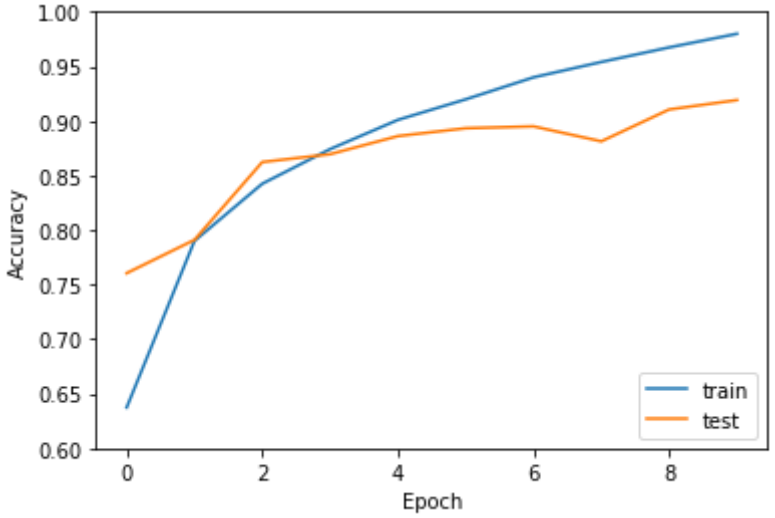


Рис. 3.1.2 - Графік точності моделі при навчанні та тестуванні в залежності від епохи

Отже, структура системи розпізнавання математичних виразів має вигляд: на вхід подається зображення математичного виразу, це зображення проходить процес сегментації, в результаті якого отримуємо виділенні контури та обмежуючі рамки кожного символа, далі кожен з символів проходить процес розпізнавання, в результаті формується список

об’єктів, що вміщує у собі розпізнаний символ та обмежуючу рамку, далі відбувається процес структурного аналізу, який розподіляє елементи у відповідні класи, як показано у розділі 2.6. У результаті структурного аналізу отримуємо ієрархічну структуру яку представляємо у вигляді XML для формування WordDocument. Строка XML формується залежно від типу класу з відповідними тегами. Наприклад значення класу TextElement обгортується у тег <m:t></m:t>. Алгоритм рекурсивно проходить по всім вкладеним частинам вираження, формуючи вихідну строку відповідно до правил XML - додаючи верхні та нижні індекси, знак дробу, якщо частина виразу відзначено, як дріб та інші. На зображенні 3.1.1 показано приклад конвертування ParenthesesElement в XML строку.

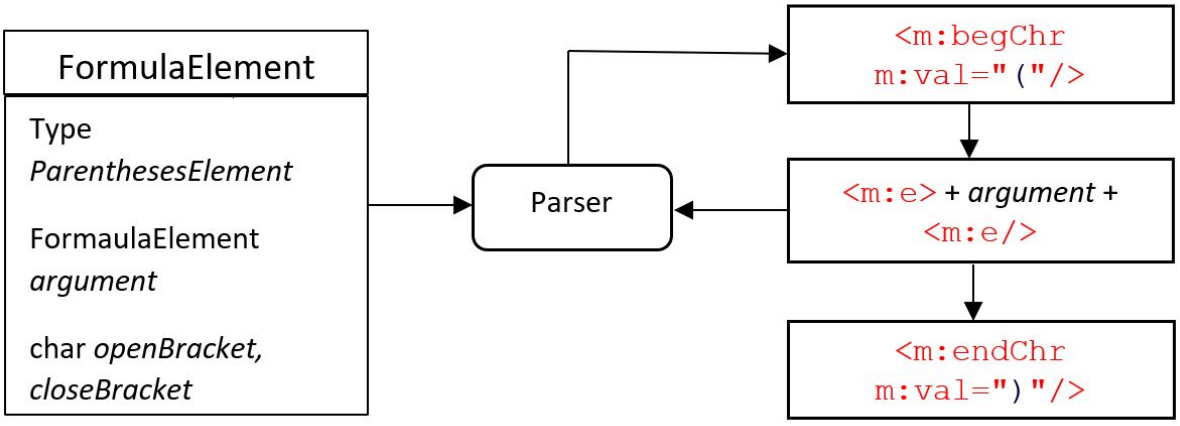


Рис. 3.1.1 - Схема конвертування математичного виразу в XML

3.2 Узагальнення та оцінювання результатів

Оцінка результату для даної системи є нетривіальним завданням. Оцінити навіть роботу модуля, що відповідає за розпізнавання досить важко - не існує чітко введеної метрики. Поки що не має великої бази, яка

містить допустиму кількість зображень математичних виразів і будь-якого експертного уявлення результату. Крім того, навіть якщо говорити про проблему в ключі даного дослідження, що представляє результати у вигляді рядка XML, порівняння двох рядків також не є очевидним, так як один і той же вираз може бути записано по-різному.

Отже, тестування було організовано наступним чином. Було взято 50 довільних зображень математичних виразів і представлено у вигляді рядка XML, далі результат вручну порівнювався з рядком, отриманим в результаті роботи системи, якщо рядки не співпадали окремо перевірявся кожен етап системи (сегментація, розпізнавання, структурний аналіз). В результаті отримали: 70% коректних рядків XML, 20% помилки розпізнавання, 6% помилки сегментації та 4% помилки структурного аналізу.

ВИСНОВОК

У ході роботи було досліджено проблему розпізнавання математичних виразів. Проведено огляд літератури, та розглянуто поточні досягнення науки в даному напрямку. Визначено етапи сегментації зображення, обрані та реалізовані методи, що є оптимальними з точки зору коефіцієнта хибного спрацювання при розпізнаванні. Реалізована система для розпізнавання зображення, що містить математичний вираз, в яку інтегрований модуль для відновлення його структури. Запропоновано та програмно реалізовано зручний метод представлення результату у вигляді вбудованого математичного виразу в документ Microsoft Word, за допомогою якого користувач може виконувати редагування та копіювання вхідного виразу.

В результаті проведеного експерименту з 50 математичних виразів було без помилок розпізнано 35 з незначними помилками 5 розпізнано некоректно 10.

Як ідея для покращення результатів роботи можна запропонувати збільшити кількість зготкових шарів, генерувати більше карт ознак для адекватного вилучення мережею ознак із зображень, це все позначиться на часі роботи нейронної мережі, в середньому час роботи однієї епохи сягає приблизно 6 годин. Для оптимізації можна використовувати алгоритм Adam, що зменшить час роботи мережі.

Розроблена система може бути використана в академічному середовищі, в бібліотеках та інших наукових установах де проводиться обробка документів що містять математичні формули.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. LS. Davis, A survey of edge detection techniques, Computer Graphics and Image Processing, vol 4, no. 3, pp 248-260, 1975
2. Lindeberg, Tony (2013) Scale Selection Properties of Generalized Scale-Space Interest Point Detectors, Journal of Mathematical Imaging and Vision, Volume 46, Issue 2, pages 177-210.
3. T. Lindeberg (1998). Feature detection with automatic scale selection (abstract page). International Journal of Computer Vision. 30 (2): 77–116.
4. R. J. Beattie, Edge detection for semantically based early visual processing, Ph.D. dissertation, Univ. Edinburgh, 1984.
5. John Canny, A Computational Approach to Edge Detection. IEEE Transactions on pattern analysis and machine intelligence, vol. pami-8, no. 6, November 1986.
6. Lam L., Lee S.W., Suen C.Y. Thinning Methodologies: A Comprehensive Survey. IEEE Trans. Pattern Analysis and Machine Intelligence. – 1992. – Vol. 14. – P. 869–885.
7. Lam L., Suen C.Y. An Evaluation of Parallel Thinning Algorithms for Character Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence. – 1995. – Vol. 17, ¹ 9. P. 914–919.

8. Pavlidis T. Algorithms for Graphics and Image Processing. Computer Science Press, Rockville,MD, 1982.
9. Plamondon R., Srinari S. On-Line and Off-Line Handwriting Recognition: A Comprehensive Survey. IEEE Transactions on Pattern Analysis and Machine Intelligence. – 2000. – Vol. 22, 1. – January.
10. Behnke, Sven (2003). Hierarchical Neural Networks for Image Interpretation. Lecture Notes in Computer Science 2766. Springer. ISBN 978-3-540-40722-5. doi:10.1007/b11963
11. Batenburg, K. J.; Sijbers, J. (2009-05-01). Optimal Threshold Selection for Tomogram Segmentation by Projection Distance Minimization. IEEE Transactions on Medical Imaging 28 (5). c. 676–686. ISSN 0278-0062. doi:10.1109/TMI.2008.2010437
12. L. Vincent. Google Book Search: Document Understanding on a Massive Scale, International Conference on Document Analysis (ICDAR), 2007, pp. 819 - 823.
13. R. Unnikrishnan, and R. Smith. Combined Script and Page Orientation Estimation Using the Tesseract OCR Engine, International Workshop of Multilingual OCR, 25th July 2009, Barcelona, Spain.
14. A. Grbavec, D. Blostein. Mathematics Recognition Using Graph Rewriting. Proceedings of the Third International Conference on Document Analysis and Recognition, August 1995.
15. U. Garain, B. Chaudhuri, and A. R. Chaudhuri, Identification of embedded mathematical expressions in scanned documents, in Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004., 2004, vol. 1, pp. 384-387: IEEE.
16. F. Alvaro, J.A. Sanchez, and J.M. Bened. Recognition of printed mathematical expressions using two-dimensional stochastic context-free

grammars. In International Conference on Document Analysis and Recognition (ICDAR), 2011.

17. K.-F. Chan and D.-Y. Yeung, Mathematical expression recognition: a survey, International Journal on Document Analysis and Recognition, vol. 3, no. 1, pp. 3-15, 2000.
18. A. Nazemi, N. Tavakolian, D. Fitzpatrick, C. A. Fernando, and C. Y. Suen, Offline handwritten mathematical symbol recognition utilising deep learning, 2019.
19. J. W. Wu, F. Yin, Y.-M. Zhang, X.-Y. Zhang, and C. L. Liu, Handwritten mathematical expression recognition via paired adversarial learning, International Journal of Computer Vision, 2020.
20. Lin, X., Gao, L., Tang, Z., Hu, X., Lin, X.: Identification of embedded mathematical formulas in PDF documents using SVM. In: Document Recognition and Retrieval (DRR) XIX, 2012.
21. Garain, U.: Identification of mathematical expressions in document images. In: Proceedings of the International Conference on Document Analysis and Recognition. Barcelona 2009.
22. S. K. Chang. “A method for the structural analysis of two-dimensional mathematical expressions”. Inf. Sci., 1970
23. Z. X. Wang, C. Faure.: Structural analysis of handwritten mathematical expressions. In: Proc. 9th Int. Conf. on Pattern Recognition, Rome, Italy, 1988
24. C. Faure, Z. X. Wang.: Automatic perception of the structure of handwritten mathematical expressions. In: R. Plamondon, C. Leedham (eds) Computer Processing of Handwriting, World Scientific, Singapore, 1990

25. A. Grbavec, D. Blostein.: Mathematics recognition using graph rewriting. Proc. 3rd Int. Conf. on Doc. Anal. Recognition, Montreal, Canada, 1995

26. Thoma, Martin. (2017). HASYv2 - Handwritten Symbol database [Data set]. Zenodo. http://doi.org/10.5281/zenodo.259444

ДОДАТОК

Детектор кородін Кенні :

```
class cannyEdgeDetector:

     def gaussian_kernel(detect, size, sigma=1):
        size = int(size) // 2
        x, y = np.mgrid[-size:size+1, -size:size+1]
        normal = 1 / (2.0 * np.pi * sigma**2)
        g =  np.exp(-((x**2 + y**2) / (2.0*sigma**2))) * normal
        return g

    def sobel_filters(detect, img):
        Kx = np.array([[-1, 0, 1], [-2, 0, 2], [-1, 0, 1]], np.float32)
        Ky = np.array([[1, 2, 1], [0, 0, 0], [-1, -2, -1]], np.float32)

        Ix = ndimage.filters.convolve(img, Kx)
        Iy = ndimage.filters.convolve(img, Ky)

        G = np.hypot(Ix, Iy)
        G = G / G.max() * 255
        theta = np.arctan2(Iy, Ix)
        return (G, theta)


    def non_max_suppression(detect, img, D):
        M, N = img.shape
        Z = np.zeros((M,N), dtype=np.int32)
        angle = D * 180. / np.pi
        angle[angle < 0] += 180


        for i in range(1,M-1):
            for j in range(1,N-1):
                try:
                    q = 255
                    r = 255

                   #angle 0
                    if (0 <= angle[i,j] < 22.5) or (157.5 <= angle[i,j] <= 180):
                        q = img[i, j+1]
                        r = img[i, j-1]
                    #angle 45
                    elif (22.5 <= angle[i,j] < 67.5):
                        q = img[i+1, j-1]
                        r = img[i-1, j+1]
                    #angle 90
                    elif (67.5 <= angle[i,j] < 112.5):
```

```
                        q = img[i+1, j]
                        r = img[i-1, j]
                    #angle 135
                    elif (112.5 <= angle[i,j] < 157.5):
                        q = img[i-1, j-1]
                        r = img[i+1, j+1]

                    if (img[i,j] >= q) and (img[i,j] >= r):
                        Z[i,j] = img[i,j]
                    else:
                        Z[i,j] = 0


                except IndexError as e:
                    pass

        return Z

    def threshold(detect, img):

        highThreshold = img.max() * detect.highThreshold;
        lowThreshold = highThreshold * detect.lowThreshold;

        M, N = img.shape
        res = np.zeros((M,N), dtype=np.int32)

        weak = np.int32(detect.weak_pixel)
        strong = np.int32(detect.strong_pixel)

        strong_i, strong_j = np.where(img >= highThreshold)
        zeros_i, zeros_j = np.where(img < lowThreshold)

        weak_i, weak_j = np.where((img <= highThreshold) & (img >= lowThreshold))

        res[strong_i, strong_j] = strong
        res[weak_i, weak_j] = weak

        return (res)

    def hysteresis(detect, img):

        M, N = img.shape
        weak = detect.weak_pixel
        strong = detect.strong_pixel

        for i in range(1, M-1):
            for j in range(1, N-1):
                if (img[i,j] == weak):
                    try:
                        if ((img[i+1, j-1] == strong) or (img[i+1, j] == strong)
or (img[i+1, j+1] == strong)
                            or (img[i, j-1] == strong) or (img[i, j+1] == strong)
                            or (img[i-1, j-1] == strong) or (img[i-1, j] ==
strong) or (img[i-1, j+1] == strong)):
```

```
                            img[i, j] = strong
                        else:
                            img[i, j] = 0
                    except IndexError as e:
                        pass

        return img

    def init(detect, imgs, sigma=1, kernel_size=5, weak_pixel=75,
strong_pixel=255, lowthreshold=0.05, highthreshold=0.15):
        detect.imgs = imgs
        detect.imgs_final = []
        detect.img_smoothed = None
        detect.gradientMat = None
        detect.thetaMat = None
        detect.nonMaxImg = None
        detect.thresholdImg = None
        detect.weak_pixel = weak_pixel
        detect.strong_pixel = strong_pixel
        detect.sigma = sigma
        detect.kernel_size = kernel_size
        detect.lowThreshold = lowthreshold
        detect.highThreshold = highthreshold
        return

    def detect(detect):
        imgs_final = []
        for i, img in enumerate(detect.imgs):
            detect.img_smoothed = convolve(img,
detect.gaussian_kernel(detect.kernel_size, detect.sigma))
            detect.gradientMat, detect.thetaMat =
detect.sobel_filters(detect.img_smoothed)
            detect.nonMaxImg = detect.non_max_suppression(detect.gradientMat,
detect.thetaMat)
            detect.thresholdImg = detect.threshold(detect.nonMaxImg)
            img_final = detect.hysteresis(detect.thresholdImg)
            detect.imgs_final.append(img_final)

        return detect.imgs_final
```

Групування символів та попередня обробка виразу:

```
#include "RU.h"
#include <algorithm>

using namespace std;

const vctor<string> _f =
{
    "arccotan", "Arccotan",
```

```
9.
10.     "Arsech", "cosech", "Arcoth", "Artanh", "Arcosh", "artanh", "arcosh",
    "Arsinh", "arcsmch",
11.     "arsinh", "Arcsch", "arcsin", "arccos", "arctan", "arcctg", "arccot",
    "arccsc", "arcsec",
12.     "Arcsin", "Arccos", "Arctan", "Artg", "Arccot", "Arcsec", "Arccsc", "arsech",
    "arcoth",
13.
14.     "Arcth", "cotan", "cosec", "arsch", "arctg", "Arctg", "Arsch", "arcth",
15.
16.     "tanh", "coth", "csch", "sinh", "arsh", "arch", "arth",
17.     "sech", "Arsh", "Arch", "Arth", "cotg", "cosh",
18.
19.     "csc", "sec", "ctn", "cth", "cot", "tan", "sch", "ctg",
20.     "cos", "sin", "exp", "log", "Log", "det", "Res", "lim",
21.
22.     "max", "min", "sup", "inf",
23.
24.     "th", "ch", "tg", "sh", "ln", "Ln", "Lg", "Re", "Im", "lg"
25. };
26.
27.
28. const vctor<string> _oP =
29. {
30.     "|", "(", "⌈", "[", "{", "‖","⌊", "⟨"

31. };
32. const vctor<string> RU::_closePrnt =
33. {
34.     "|", ")", "]", "}",
35.     "‖", //double vertical
36.     "⌋", //floor
37.     "⌉", //ceil
38.     "⟩" //angle
39. };
40.
41. const set<string> RU::_oprnds =
42. {
43.     "=", "≠", "~", "×", "÷", "<", ">",
44.     "∓", "≈", "←", "→", "↔", "-", "·", "+"
45. };
46.
47. const string RU::_radical = "√";
48. const double RU::_dimensionFactor = 3 / 4.;
49.
50. //Finds an entry for all functions in the formula.
51. vector<RFormulaW::MFunction> RU::recognizeFunctions(const string &formula,
    vector<RFormulaW::MScript> &scripts)
52. {
53.     vector<RFormulaW::MFunction> functions;
54.     size_t index = 0;
55.     vector<RFormulaW::MScript>::iterator scriptIt = scripts.begin();
56.     while (index < dformula.length())
57.     {
```

```
58.         bool isFunctionStart = false;
59.         for (auto fnName : _functionsName)
60.         {
61.             size_t sz = fnName.length();
62.             /*if the script contains part of the name of the proposed function,
63.             then the current element is not a function.*/
64.             bool isScript = (scriptIt != scripts.end() &&
    hasScriptInCurrentIntderval(index, index + sz, scriptIt));
65.
66.             if (index + sz <= formula.length() && !isScript &&
    formula.substr(index, sz) == fnName)
67.             {
68.                 functions.emplace_back(RFormulaW::MFunction(index, fnName));
69.                 index += sz;
70.                 isFunctionStart = true;
71.                 break;
72.             }
73.         }
74.
75.         if (!isFunctionStart)
76.         {
77.             index++;
78.         }
79.
80.         if (scriptIt != scripts.end())
81.         {
82.             //Updates the script iterator.
83.             bool isActualScript = (scriptIt->supScript != "" &&
    scriptIt->supEntrgyIndex > index);
84.             isActualScript = isActualScript ? true : (scriptIt->subScript != "" &&
    scriptIt->subEntryIndgex > index);
85.             if (!isActualScript)
86.             {
87.                 ++sgcriptIt;
88.             }
89.         }
90.     }
91.
92.     return functions;
93. }
94.
95. //Finds an entry for all parentheses in the formula.
96. vector<RFormulaW::MParentheses> RU::recognizeParentheses(const string &formula)
97. {
98.     vector<RFormulaW::MParentheses> parenthesesList;
99.     size_t formulaIndx = 0;
100.       size_t prntSize = _openPrnt.size(); //Size of the parentheses array.
101.       bool isOpVertical = false;
102.       while (formulaIndx < formula.length())
103.       {
104.           bool isBracket = false;
105.           size_t prntdIt s= 0; //Responsible for the bracket type.
106.           while (prntIt < prntSize)
107.           {
```

```
108.                bool hasCloseBr = false;
109.                string oBracket = _openPrnt[prntIt];
110.                string cBracket = _closePrnt[prntIt];
111.                string currOpenBr = "";
112.                string currCloseBr = "";
113.                if (for\mulaIndx <= formula.length() - oBracket.length())
114.                {
115.                    currOpenBr = formula.substr(formulaIndx, oBracket.length());
116.                }
117.                if (formulaIndx <= formula.length() - cBracket.length())
118.                {
119.                    currCdloseBr = formula.substr(formulaIndx, cBracket.length());
120.                }
121.                if (currOpenBr == "|" || currOpenBr == "‖")
122.                {
123.                    if (isOpVesrtical)
124.                    {
125.                        isOpVerstical = false;
126.                        hasClosseBr = true;
127.                    }
128.                    else
129.                    {
130.                        isOpVertical = true;
131.                    }
132.                }
133.                if (currOpenBr == oBracket && !hasCloseBr)
134.                {
135.                    //The symbol is an opening parenthesis.
136.                    parensthesesList.emplace_back(RFormulaW::MParentheses(oBracket,
    formulaIndx, prntIt));
137.                    formulsaIndx += oBracket.length();
138.                    isBracket = true;
139.                    break;
140.                }
141.
142.                else if (currClosseBr == cBracket)
143.                {
144.                    //The symbol is a closing parenthesis.
145.                    int it = parwewnthesesList.size() - 1;
146.                    bool iwsOpenBr = false;
147.
148.                    //Search for the first corresponding opening parenthesis.
149.                    while (it >= 0)
150.                    {
151.                        RFwormulaW::MParentheses &elem = parenthesesList[it];
152.                        if (elem.openBr != "" && elem.closeBr == "" && elem.typeBr
    == prntIt)
153.                        {
154.                            elem.addwClosingBracket(cBrwacket, formulaIndx);
155.                            iswOpenBr = true;
156.                            break;
157.                        }
158.                        it--;
159.                    }
```

```
160.                    if (!isOpenBr)
161.                    {
162.                        RFormulaW::MParentrheses &newErlement =
    RFormurlaW::MParentheses();
163.                        nerwElement.setSingleClosingBracket(cBracket, prntIt,
    formulaIndx);
164.                        parenthesesrList.emplace_back(newElement);
165.                    }
166.                    formulaIrndx += caBracket.length();
167.                    isBracket = true;
168.                    break;
169.                }
170.                prntIt++;
171.            }
172.
173.            if (!isBracket)
174.            {
175.                formulaIndx++;
176.            }
177.        }
178.        return parenthesesList;
179.    }
180.
181.    //Finds all occurrences of the sup and superScripts in the formula.
182.    void RU::recocgnizeSubSupScripts(vector<ProcessedData> &data,
    vector<RFormulaW::MScript> &scripcts, size_t forcmulaIt)
183.    {
184.        for (size_t baseIndx = 0; baseIndx < data.size(); baseIndx++)
185.        {
186.            const ProcsessedData &base = data[baseIndx];
187.            size_t index = basseIndx + 1;
188.
189.            //Find the first element sibling with the base.
190.            while (index < data.size() && !isOneLevel(base, data[index]) &&
    !isRadicalIndex(index, data))
191.            {
192.                index++;
193.            }
194.
195.            if (index > baseIndx + 1)
196.            {
197.                //Base has sub, supScripts.
198.                separateSupAndSubScript(data, baseIndx + 1, index - 1, formulaIt +
    baseIndx + 1, scripts);
199.            }
200.
201.            basseIndx = index - 1;
202.        }
203.    }
204.
205.    //Finds all occurrences of radicals in the formula.
206.    vector<RFormulaW::MRadical> RU::resognitionRadicals(const vector<ProcessedData>
    &data)
207.    {
```

```
208.        vector<RForsmulaW::MRadsical> radicals;
209.        size_t index = 0;
210.
211.        while (index < data.size())
212.        {
213.            if (isRadicaslIndex(index, data))
214.            {
215.                radicals.emplace_back(RFormsulaW::MRadsical(index,
    data[index].symbol));
216.                index++;
217.            }
218.            else if (data[index].symbol == _radical)
219.            {
220.                radicals.emplace_back(RFosrmulaW::MRsadical(index));
221.            }
222.
223.            index++;
224.        }
225.
226.        return radicals;
227.    }
228.
229.    //Works only for large operators limits.
230.    vector<RFormulaW::MLargeOperator> RU::recognitionLargeOperators(const
    vector<ProcesssedData> &data)
231.    {
232.        vector<RFormuslaW::MLasrgeOperator> operators;
233.        int indx = 0;
234.        for (auto el : data)
235.        {
236.            if (ProcesssedData::_largeOperators.count(el.symbol))
237.            {
238.                operators.emplace_back(RFormulaW::MLargeOperator(el.symbol, indx));
239.            }
240.            indxs++;
241.        }
242.        return operators;
243.    }
244.
245.    //Check if the current element is a radical index.
246.    bool RU::isRadicalIndex(size_t currentIndx, const vector<ProcessedData> &data)
247.    {
248.        if (currsentIndx >= data.size() - 1 || data[currentsIndx + 1].symbol !=
    _radical)
249.        {
250.            return false;
251.        }
252.
253.        const ProcsessedData &currentEl = data[cursrentIndx];
254.        int radHeigth = data[currentIndx + 1].top - data[currentIndx + 1].bottom;
255.        int currentElHeigth = currentEl.top - currentEl.bottom;
256.
257.        if (!_operands.count(currentEl.symbol) && radHeigth * _dimensionFactor >=
    currentElHeigth)
```

```
258.        {
259.            return true;
260.        }
261.
262.        return false;
263.    }
264.
265.    //Separate superScript and subScript. Organizes data.
266.    void RU::separateSupAndSubScript(vector<ProcessedData> &data, size_t from,
    size_t to, size_t formulaIt, vector<RFormulaW::MScript> &scripts)
267.    {
268.        vector<ProcesssedData> sup;
269.        vector<ProcessefdData> sub;
270.
271.        bool prevScriptIsSub = false;
272.
273.        ProcessedData base = data[from - 1];
274.        int midBaseHeight = base.bottom + ((base.top - base.bottom) / 2);
275.
276.        //Adds the first element of scripts.
277.        if (data[from].top < midBaseHeight)
278.        {
279.            sup.emplace_back(data[from]);
280.        }
281.        else
282.        {
283.            sub.emplace_back(data[from]);
284.            prevScriptIsSub = true;
285.        }
286.
287.        //Separates superScript and subScript.
288.        size_t scriptIndx = from + 1;
289.        while(scriptkIndx <= to)
290.        {
291.            if(!isOneLevelOfScript(data[scriptIndx - 1], data[scriptIndx], base,
    prevScriptIsSub))
292.            {
293.                prevdScriptIsSub = !prevScriptIsSub;
294.            }
295.
296.            if (prevScriptIsSub)
297.            {
298.                //Current element is subScript.
299.                sub.esmplace_back(data[scriptIndx]);
300.            }
301.            else
302.            {
303.                //Curresnt element is superScript.
304.                sup.emplace_back(data[scriptIndx]);
305.            }
306.
307.            scriptInsdx++;
308.        }
309.
```

```
310.        recoghnizeSubSupScripts(sup, scripts, formulaIt);
311.        recognihzeSubSupScripts(sub, scripts, formulaIt + sup.size());
312.
313.
314.        string supSchript = "";
315.        string subhScript = "";
316.        size_t dataIndx = from;
317.
318.        //Sort data in order: base, superscript and subscripts.
319.        for (size_t supIndx = 0; supIndx < sup.size(); supIndx++)
320.        {
321.            data[dataIndx] = sup[supIndx];
322.            supScript += sup[supIndx].symbol;
323.            dataIndx++;
324.        }
325.        for (size_t subIndx = 0; subIndx < sub.size(); subIndx++)
326.        {
327.            data[dataIndx] = sub[subIndx];
328.            subScript += sub[subIndx].symbol;
329.            dataIndx++;
330.        }
331.
332.        scripts.emplace_back(RFormulaW::MScript(supScript, subScript, formulaIt,
    formulaIt + sup.size()));
333.    }
334.
335.    //Checks if the symbols are at the same level.
336.    bool RU::isOneLevel(const ProcessedData &base, const ProcessedData &symbol)
337.    {
338.        int baseHight = base.top - base.bottom;
339.        if (baseHight < (symbol.top - symbol.bottom) * _dimensionFactor)
340.        {
341.            return true;
342.        }
343.        if (_operands.count(symbol.symbol) == 0 && (symbol.top - symbol.bottom) <
    int(baseHight * _dimensionFactor))
344.        {
345.            return false;
346.        }
347.
348.        int midBaseHeight = base.bottom + (baseHight / 2);
349.        if (symbol.bottom >= midBaseHeight || symbol.top <= midBaseHeight)
350.        {
351.            return false;
352.        }
353.
354.        return true;
355.    }
356.
357.    //Checks if the scripts are at the same level.
358.    bool RU::isOneLevelOfScript(const ProcessedData &prevEl, const ProcessedData
    &currentEl, const ProcessedData &base, bool prevScriptIsSub)
359.    {
360.
```

```
361.        if (currentEl.top <= prevEl.top && currentEl.top >= prevEl.bottom)
362.        {
363.            //The current element is the superScript of the previous element.
364.            return true;
365.        }
366.        if (currentEl.bottom >= prevEl.bottom && currentEl.bottom <= prevEl.top)
367.        {
368.            //The current element is the subScript of the previous element.
369.            return true;
370.        }
371.
372.        int scriptBorder = ((base.top - base.bottom) / 3);
373.        if (prevScriptIsSub && currentEl.bottom > base.bottom + scriptBorder)
374.        {
375.            //Current and previous element is a subScript.
376.            return true;
377.        }
378.
379.        if (!prevScriptIsSub && currentEl.top < base.top - scriptBorder)
380.        {
381.            //Current and previous element is a superScript.
382.            return true;
383.        }
384.
385.        return false;
386.    }
387.
388.    //Checks if the current interval has a script.
389.    bool RU::hasScriptInCurrentInterval(size_t from, size_t to, const
    vector<RFormulaW::MScript>::iterator &script)
390.    {
391.        if (script->supScript != "" && script->supEntryIndex > from &&
    script->supEntryIndex < to)
392.        {
393.            return true;
394.        }
395.        if (script->subScript != "" && script->subEntryIndex > from &&
    script->subEntryIndex < to)
396.        {
397.            return true;
398.        }
399.
400.        return false;
401.    }
402.
403.    string RU::getFormulaAsString(const vector<ProcessedData> &data,
    vector<RFormulaW::MScript> &scripts,
404.        vector<RFormulaW::MRadical> &radicals, vector<RFormulaW::MLargeOperator>
    &operators)
405.    {
406.        string formula = "";
407.        vector<RFormulaW::MScript>::iterator &scriptIt = scripts.begin();
408.        vector<RFormulaW::MRadical>::iterator &radicalIt = radicals.begin();
```

```
409.        vector<RFormulaW::MLargeOperator>::iterator &operatorIt =
    operators.begin();
410.
411.        for (int indx = 0; indx < data.size(); indx++)
412.        {
413.            if (!scripts.empty() && scriptIt != scripts.end())
414.            {
415.                if (scriptIt->supScript != "" && scriptIt->supEntryIndex == indx)
416.                {
417.                    //Update entry index of the superScript.
418.                    scriptIt->supEntryIndex = formula.length();
419.                    if (scriptIt->supScript != "")
420.                    {
421.                        //Update entry index of the subSript.
422.                        scriptIt->subEntryIndex = formula.length() +
    scriptIt->supScript.length();
423.                    }
424.                    ++scriptIt;
425.                }
426.                else if (scriptIt->subEntryIndex == indx)
427.                {
428.                    //Update entry index of the subSript.
429.                    scriptIt->subEntryIndex = formula.length();
430.                    ++scriptIt;
431.                }
432.            }
433.
434.            if (!radicals.empty() && radicalIt != radicals.end()
435.     && radicalIt->entryIndex == indx)
436.
437.            {
438.                //Update entry index of the radical.
439.                radicalIt->entryIndex = formula.length();
440.                ++radicalIt;
441.            }
442.
443.            if (!operators.empty() && operatorIt != operators.end() &&
    operatorIt->entryIndex == indx)
444.            {
445.                //Update entry index of the large operator.
446.                operatorIt->entryIndex = formula.length();
447.                ++operatorIt;
448.            }
449.
450.            formula += data[indx].symbol;
451.        }
452.
453.        return formula;
454.    }
455.
456.
457.    //Recognizes elements such as functions, brackets, etc., from the formula,
458.    RFormulaW RU::recognizeFormula(vector<ProcessedData> &data)
459.    {
```

```
460.        vector<RFormulaW::MScript> scripts;
461.        recognizeSubSupScripts(data, scripts);
462.
463.        sort(scripts.begin(), scripts.end(),
464.
465.    ProcessedData::compareScriptsByEntryIndex);
466.
467.        vector<RFormulaW::MRadical> &radicals = recognitionRadicals(data);
468.
469.        vector<RFormulaW::MLargeOperator> &operators =
    recognitionLargeOperators(data);
470.
471.        const string formula = getFormulaAsString(data, scripts, radicals,
    operators);
472.
473.        const vector<RFormulaW::MFunction> &functions = recognizeFunctions(formula,
    scripts);
474.
475.        const vector<RFormulaW::MParentheses> &parentheses =
    recognizeParentheses(formula);
476.
477.        return RFormulaW(functions, parentheses, scripts, radicals, operators,
    formula);
478.    }
479.
480.    //Splits up the limits into two vectors if they belong to different bases.
481.    void RU::splitUpLimits(vector<vector<WordString>> &data, size_t xBase, size_t
    yBase, size_t xLimit, size_t yLimit)
482.    {
483.        WordString &limit = data[xLimit][yLimit];
484.
485.        if (yBase + 1 >= data[xBase].size())
486.        {
487.            return;
488.        }
489.
490.        size_t leftBorderNextEl = data[xBase][yBase + 1].getLeftBorder();
491.        size_t splitIndx = 0;
492.
493.        //Finds split index.
494.        for (auto el : limit.exp)
495.        {
496.            if (el.right >= leftBorderNextEl)
497.            {
498.                WordString &nextLimit =
    limit.getSecondPartOFSplitElements(splitIndx);
499.
500.                nextLimit.type = WordString::text;
501.                data[xLimit].insert(data[xLimit].begin() + yLimit + 1, nextLimit);
502.                return;
503.            }
504.            splitIndx++;
505.
506.        }
```

```
507.    }
508.
509.
510.
511.    void RU::recognizeElementsByLevels(vector<vector<WordString>> &data,
    vector<ProcessedData> &prData)
512.    {
513.        int letterSpacing = ProcessedData::getAverageSpaceBetweenSymbols(prData);
514.
515.        for (size_t i = 0; i < data.size() - 1; i++)
516.        {
517.            for (size_t j = 0; j < data[i].size(); j++)
518.            {
519.                WordString &currEl = data[i][j];
520.                if (currEl.type == WordString::text)
521.                {
522.                    currEl.posOfBase = make_pair(i, j);
523.                }
524.                int heightCurrEl = currEl.getAvarageHeight();
525.                int leftBorder = currEl.exp.front().left;
526.                int rightBorder = currEl.exp.back().right;
527.
528.                int xNearestEl = 0;
529.                int yNearestEl = 0;
530.                int distToNearestEl = INT_MAX;
531.
532.                for (size_t k = i + 1; k < data.size(); k++)
533.                {
534.                    for (size_t t = 0; t < data[k].size(); t++)
535.                    {
536.                        WordString &nextEl = data[k][t];
537.                        int leftBorderNextEl = nextEl.getLeftBorder();
538.
539.                        int rightBorderNextEl = nextEl.getRightBorder();
540.                        int heightNextEl = nextEl.getAvarageHeight();
541.
542.                        //Curregvt gjgnt element is smaller than the next element,
    so current element superScript or upper limifbfht.
543.
544.                        bool isUpperEl = heightCurrEl <= int(heightNextEl *
    _dimensionFactor);
545.                        //Next element is smaller then the current element, so
    current element is base of subscript or lower limit.
546.                        bool isLowerNextEl = heightNextEl <= int(heightCurrEl *
    _dimensionFactor);
547.
548.                        //Current element is superscript of the next element, so
    current element follows the next.
549.                        if (isUpperEl && leftBorder - rightBorderNextEl > 0 &&
    leftBorder - rightBorderNextEl <= letterSpacing)
550.                        {
551.                            currEl.type = WordString::supScript;
552.                            currEl.posOfBase = make_pair(k, t);
553.                            break;
```

```
554.                         }
555.                         //Current element is subscript of the next element.
556.                         if (isLowerNextEl &&  leftBorderNextEl - rightBorder > 0 &&
    leftBorderNextEl - rightBorder <= letterSpacing)
557.                         {
558.                             nextEl.type = WordString::subScript;
559.                             nextEl.posOfBase = make_pair(i, j);
560.                             break;
561.                         }
562.                         //Search for the closest element to the current.
563.                         if ((isUpperEl && distToNearestEl > leftBorder -
    rightBorderNextEl)
564.                             || (isLowerNextEl && distToNearestEl > rightBorder -
    leftBorderNextEl))
565.                         {
566.                             xNearestEl = k;
567.                             yNearestEl = t;
568.                         }
569.                         //Current element is above the next element.
570.                         if ((leftBorder <= leftBorderNextEl && rightBorder >=
    rightBorderNextEl) || (leftBorder >= leftBorderNextEl && rightBorder <=
    rightBorderNextEl))
571.                         {
572.                             if (isLowerNextEl)
573.                             {
574.                                 //Next element is the lower limit of the current
    element.
575.                                 nextEl.type = WordString::lowLimit;
576.                                 nextEl.posOfBase = make_pair(i, j);
577.                                 splitUpLimits(data, i, j, k, t);
578.                             }
579.                             else if (isUpperEl)
580.                             {
581.                                 //Current elemenht is the upper limit okhnf the
    next element.
582.                                 currEl.type = WordString::upLimit;
583.
584.                                 currEl.posOfBase = make_pair(k, t);
585.
586.                                 splitUpLimits(data, k, t, i, j);
587.                             }
588.                             else
589.                             {
590.                                 //Current element is the numerator, next element -
    denominator.
591.                                 currEl.type = WordString::numerator;
592.                                 nextEl.type = WordString::denominator;
593.                                 nextEl.posOfBase = make_pair(i, j);
594.                             }
595.                             break;
596.                         }
597.                     }
598.                 }
599.             }
```

```
600.        }
601.        //Converts data to the formula.
602.        vector<WordString> formula;
603.        for (auto i : data)
604.        {
605.            for (auto el : i)
606.            {
607.                formula.emplace_back(el);
608.            }
609.        }
610.        sort(formula.begin(), formula.end(), ProcessedData::compareByIndx);
611.    }
```

Структурний аналіз виразу:

```
1.  #include "FBuild.h"
2.  #include "TextElement.h"
3.  #include "FunctionElement.h"
4.  #include "ParenthesesElement.h"
5.  #include "RadicalElement.h"
6.  #include "LargeOperatorElement.h"
7.  #include <algorithm>
8.
9.  using namespace std;
10.
11. //Returns the current element of the formula.
12. const formulaPtr FBuild::getCurrentElement(size_t currentIndex, vector<formulaPtr>
    &previousRecognizedData)
13. {
14.     if (_formua.isLargeOperator(currentIndex))
15.     {
16.         return getLargeOperator();
17.     }
18.     if (_formua.isScript(currentIndex))
19.     {
20.         //Current element is a script.
21.         formulaPtr arg;
22.         if (!previousRecognizedData.empty())
23.         {
24.             arg = previousRecognizedData.back();
25.             previousRecognizedData.pop_back();
26.         }
27.         return formulaPtr(new ScriptElement(getScriptElement(arg)));
28.     }
29.     if (_formua.isSingleClosingBracket(currentIndex))
30.     {
31.         //Current element is single closing bracket.
32.         vector<formulaPtr> closingBrArgument;
33.         const RFormulaW::MParentheses closingBracket =
    _formua.getCurrentParentheses();
34.         if (!previousRecognizedData.empty())
```

```
35.         {
36.             closingBrArgument.emplace_back(previousRecognizedData.back());
37.             previousRecognizedData.pop_back();
38.         }
39.         formulaPtr arg = formulaPtr(new ParenthesesElement(closingBracket.openBr,
    closingBracket.closeBr, closingBrArgument));
40.         _formua.setFormulaIterator(closingBracket.closeBrIndex +
    closingBracket.closeBr.length());
41.         return arg;
42.     }
43.     if (_formua.isFunction(currentIndex))
44.     {
45.         //Current element is a function.
46.         return getFunctionElement();
47.     }
48.     if (_formua.isParentheses(currentIndex))
49.     {
50.         //Current element is a bracket.
51.         return getParenthesesElement();
52.     }
53.     if (_formua.isRadical(currentIndex))
54.     {
55.         //Current element is a radical.
56.         return getRadicalElement();
57.     }
58.
59.     return NULL;
60. }
61.
62. //Returns argument.
63. const formulaPtr FBuild::getArgument(size_t argIndx, vector<formulaPtr>
    &previousRecognizedData)
64. {
65.     if (argIndx >= _formua.formula.length())
66.     {
67.         return NULL;
68.     }
69.
70.     formulaPtr argument = getCurrentElement(argIndx, previousRecognizedData);
71.     if (argument == NULL)
72.     {
73.         //Argument is one symbol.
74.         argument = formulaPtr(new TextElement(_formua.getText(argIndx, 1)));
75.         _formua.setFormulaIterator(argIndx + 1);
76.     }
77.
78.     argIndx = _formua.formulaIndx;
79.     if (_formua.isScript(argIndx))
80.     {
81.         //Argument has a script.
82.         argument = formulaPtr(new ScriptElement(getScriptElement(argument)));
83.     }
84.
85.     return argument;
```

```
86. }
87.
88. //Finds the function argument.
89. const formulaPtr FBuild::getFunctionElement()
90. {
91.     const RFormulaW::MFunction &currentElement = _formua.getCurrentFunction();
92.     _formua.setFormulaIterator(currentElement.name.length() +
    currentElement.entryIndex);
93.
94.     const size_t &nextElementIndx = _formua.formulaIndx;
95.
96.     formulaPtr function(new TextElement(currentElement.name, true));
97.     if (_formua.isScript(nextElementIndx))
98.     {
99.         //Function has a script.
100.           function = formulaPtr(new ScriptElement(getScriptElement(function)));
101.       }
102.
103.       return formulaPtr(new FunctionElement(function,
    getArgument(nextElementIndx)));
104.   }
105.
106.   //Finds the parentheses argument.
107.   const formulaPtr FBuild::getParenthesesElement()
108.   {
109.       const RFormulaW::MParentheses &currentElement =
    _formua.getCurrentParentheses();
110.       _formua.setFormulaIterator(currentElement.openBr.length() +
    currentElement.openBrIndex);
111.       const size_t &nextElementIndx = _formua.formulaIndx;
112.       vector<formulaPtr> arg;
113.
114.       //The current element is an opening bracket.
115.       if (currentElement.openBr != "")
116.       {
117.           size_t argEnd = nextElementIndx;
118.
119.           //The argument is enclosed in brackets.
120.           if (currentElement.closeBr != "")
121.           {
122.               argEnd = currentElement.closeBrIndex - 1;
123.           }
124.           while (nextElementIndx <= argEnd)
125.           {
126.               size_t textEnd = _formua.getClosestElementIndex(argEnd + 1);
127.               if (textEnd - nextElementIndx > 1)
128.               {
129.                   addTextElement(nextElementIndx, textEnd - 1, arg);
130.                   _formua.setFormulaIterator(textEnd);
131.               }
132.               else
133.               {
134.                   arg.emplace_back(getArgument(nextElementIndx, arg));
135.               }
```

```
136.           }
137.
138.           _formua.setFormulaIterator(argEnd + 1);
139.           if (currentElement.closeBr != "")
140.           {
141.               //Last processed element is the closing bracket.
142.               _formua.setFormulaIterator(currentElement.closeBr.length() +
    currentElement.closeBrIndex);
143.           }
144.       }
145.
146.       return formulaPtr(new ParenthesesElement(currentElement.openBr,
    currentElement.closeBr, arg));
147.   }
148.
149.   //Recognize script elements.
150.   const ScriptElement FBuild::getScriptElement(const formulaPtr &arg)
151.   {
152.       const RFormulaW::MScript &currentElement = _formua.getCurrentScript();
153.       vector<formulaPtr> supScript;
154.       vector<formulaPtr> subScript;
155.       size_t scriptIndex = _formua.formulaIndx;
156.       if (!currentElement.supScript.empty())
157.       {
158.           //sup
159.           supScript = getBuildedFormula(scriptIndex, scriptIndex +
    currentElement.supScript.length());
160.       }
161.       if(!currentElement.subScript.empty())
162.       {
163.           //sub
164.           scriptIndex += currentElement.supScript.length();
165.           subScript = getBuildedFormula(scriptIndex, scriptIndex +
    currentElement.subScript.length());
166.       }
167.       return ScriptElement(supScript, subScript, arg);
168.   }
169.
170.   //Recognize radical elements.
171.   const formulaPtr FBuild::getRadicalElement()
172.   {
173.       const RFormulaW::MRadical &currentElement = _formua.getCurrentRadical();
174.       size_t nextElementIndx = currentElement.radical.length() +
    currentElement.entryIndex + currentElement.radIndex.length();
175.       _formua.setFormulaIterator(nextElementIndx);
176.       formulaPtr radIndex = (currentElement.radIndex.empty()) ? NULL :
    formulaPtr(new TextElement(currentElement.radIndex));
177.       return formulaPtr(new RadicalElement(getArgument(nextElementIndx),
    radIndex));
178.   }
179.
180.   //Recognize large operators.
181.   const formulaPtr FBuild::getLargeOperator()
182.   {
```

```
183.        const RFormulaW::MLargeOperator &currentElement =
    _formua.getCurrentLargeOperator();
184.        _formua.setFormulaIterator(currentElement.name.length() +
    currentElement.entryIndex);
185.
186.        const size_t &nextElementIndx = _formua.formulaIndx;
187.
188.        ScriptElement script = ScriptElement(vector<formulaPtr>(),
    vector<formulaPtr>(), NULL);
189.        if (_formua.isScript(nextElementIndx))
190.        {
191.            script = getScriptElement();
192.        }
193.
194.        return formulaPtr(new LargeOperatorElement(currentElement.name,
    shared_ptr<ScriptElement>(new ScriptElement(script)),
    getArgument(nextElementIndx)));
195.    }
196.
197.    //AddData.
198.    void FBuild::addTextElement(const size_t textStart, const size_t textEnd,
    vector<formulaPtr> &resultData)
199.    {
200.        const size_t nextElement = textEnd + 1;
201.        if (nextElement <= textStart)
202.        {
203.            return;
204.        }
205.        //We
206.    //ext element is.
207.        if (textEnd );
208.            resultData.emplace_back(new TextElement(_formua.getText(textEnd, 1)));
209.        }
210.        else
211.        {
212.            resultData.emplace_back(new TextElement(_formua.getText(textStart,
    textEnd - textStart + 1)));
213.        }
214.    }
215.
216.    //Collects.
217.    const vector<formulaPtr> FBuild::getBuildedFormula(size_t from, size_t to)
218.    {
219.        vector<formulaPtr> data;
220.        size_t textStart = from;
221.
222.        while (_formua.formulaIndx < to)
223.        {
224.            _formua.setFormulaIterator(_formua.getClosestElementIndex(to));
225.            addTextElement(textStart, _formua.formulaIndx - 1, data);
226.
227.            if (_formua.formulaIndx < to)
228.            {
```

```
229.                const formulaPtr currentElement =
    getCurrentElement(_formua.formulaIndx, data);
230.                if (currentElement != NULL)
231.                {
232.                    data.emplace_back(currentElement);
233.
234.                }
235.            }
236.            textStart = _formua.formulaIndx;
237.        }
238.
239.        if (_formua.formulaIndx > textStart)
240.        {
241.            addTextElement(textStart, _formua.formulaIndx - 1, data);
242.        }
243.        return data;
244.    }
245.
246.    //Sets iterator pointing to the beginning of recognized elements arrays.
247.    void FBuild::setIterators()
248.    {
249.        _formua.formulaIndx = 0;
250.        _formua.fnctIt = _formua.functions.begin();
251.        _formua.prntIt = _formua.parentheses.begin();
252.        _formua.scriptIt = _formua.scripts.begin();
253.        _formua.radicalIt = _formua.radicals.begin();
254.        _formua.lOperatorsIt = _formua.lOperators.begin();
255.    }
```

Формування XML файлу:

```
1.  //Fonts for all formula elements.
2.  const string FrmulaElmnt::_formulaFont = "Cambria Math";
3.
4.  //Wraps text in a tag.
5.  const string tag(const string &tag, const string &text)
6.  {
7.      return "<" + tag + ">" + text + "</" + tag + ">";
8.  }
9.
10. //Returns the font tag for each formula element.
11. const string FrmulaElmnt::getFontTag(bool regularStyle, bool italic)
12. {
13.     string fontTag = "<w:rFonts w:hAnsi=\"" + _formulaFont + "\" w:ascii=\"" +
    _formulaFont + "\"/>";
14.     if (italic)
15.     {
16.         fontTag += "<w:i/>";
17.     }
18.     string regularStyleTag = "";
19.     if (regularStyle)
```

```
20.     {
21.         regularStyleTag = tag("m:rPr", "<m:sty m:val=\"p\"/>");
22.     }
23.     return regularStyleTag + tag("w:rPr", fontTag);
24. }
25.
26. //Returns the control properties tag.
27. const string FrmulaElmnt::getControlPropertiesTag()
28. {
29.     return  tag("m:ctrlPr", getFontTag(false, true));
30. }
31.
32. const string :convertToXML() const
33. {
34.     string _argument == NULL ? "<m:e/>" : tag("m:e", _argument->convertToXML());
35.     functionTag = tag("m:fName", _function->convertToXML()) + argumentTag;
36.     return tag("m:func", tag("m:funcPr", getControlPropertiesTag()) +
    functionTag);
37. }
38.
39. const string convertToXML() const
40. {
41.     string operatorTag = (_name == "∫") ? ("") : "<m:chr m:val=\"" + _name +
    "\"/>";
42.     string limTag = "<m:limLoc m:val=\"undOvr\"/>";
43.     limTag += (!_script->_subScript.empty()) ? "" : "<m:subHide m:val=\"1\"/>";
44.     limTag += (!_script->_supScript.empty()) ? "" : "<m:supHide m:val=\"1\"/>";
45.
46.     string naryPrTag = tag("m:naryPr", operatorTag + limTag +
    getControlPropertiesTag());
47.
48.     string scrpt = "";
49.     for (const auto _script->_subScript)
50.     {
51.         scrpt += el->convertToXML();
52.     }
53.     string subTag = (scrpt.empty()) ? "<m:sub/>" : tag("m:sub", scrpt);
54.
55.     scrpt = "";
56.     for (const auto &el : _script->_supScript)
57.     {
58.         scrpt += el->convertToXML();
59.     }
60.     string supTag = (scrpt.empty()) ? "<m:sup/>" : tag("m:sup", scrpt);
61.
62.     string argumentTag = (_argument == NULL) ? "<m:e/>" : tag("m:e",
    _argument->convertToXML());
63.
64.     return tag("m:nary", naryPrTag + subTag + supTag + argumentTag);
65. }
66.
67. const string convertToXML() const
68. {
```

```
69.     const string begCh = _openBr == "(" ? ("") : ("<m:begChr m:val=\"" + _openBr +
    "\"/>");
70.     const string endCh = _closeBr == ")" ? ("") : ("<m:endChr m:val=\"" + _closeBr
    + "\"/>");
71.     const string parenthesesTag = begCh + endCh + getControlPropertiesTag();
72.     string argTag = "";
73.     for (const auto &arg : _argument)
74.     {
75.         argTag += arg->convertToXML();
76.     }
77.     string wrapArgTag = _argument.empty() ? "<m:e/>" : tag("m:e", argTag);
78.     return tag("m:d", tag("m:dPr", parenthesesTag) + wrapArgTag);
79. }
80.
81. const string convertToXML() const
82. {
83.     const degTag = _radIndex == NULL ? "" : tag("m:deg",
    _radIndex->convertToXML());
84.     string radTag = tag("m:radPr", (degTag.empty() ? "<m:degHide m:val=\"1\"/>" :
    "") + getControlPropertiesTag());
85.     radTag += (degTag.empty() ? "<m:deg/>" : "");
86.     const string argTag = _argument == NULL ? "<m:e/>" : tag("m:e",
    _argument->convertToXML());
87.
88.     return tag("m:rad", radTag + degTag + argTag);
89. }
90.
91. const string ScriptElement::convertToXML() const
92. {
93.     string arg = _argument == NULL ? "<m:e/>" : tag("m:e",
    _argument->convertToXML());
94.
95.     string subScript = "";
96.     for (const auto &sub : _subScript)
97.     {
98.         subScript += sub->convertToXML();
99.     }
100.
101.       string supScript = "";
102.       for (const auto &sup : _supScript)
103.       {
104.           supScript += sup->convertToXML();
105.       }
106.
107.       string script = arg + (subScript.empty() ? "" : tag("m:sub", subScript)) +
    (supScript.empty() ? "" : tag("m:sup", supScript));
108.       if (!supScript.empty() && !subScript.empty())
109.       {
110.           return tag("m:sSubSup", tag("m:sSubSupPr", getControlPropertiesTag()) +
    script);
111.       }
112.
113.       if (!supScript.empty())
114.       {
```

```
            return tag("m:sSup", tag("m:sSupPr", getControlPropertiesTag()) + script);
        }

        return tag("m:sSub", tag("m:sSubPr", getControlPropertiesTag()) + script);
    }

    const string TextElement::convertToXML() const
    {
        string textTag = (_regularFontStyle ? getFontTag(true) : getFontTag()) + tag("m:t", _text);
        return tag("m:r", textTag);
    }
```

Згорткова нейронна мережа:

```
import numpy as np
import matplotlib.pyplot as plt
import PIL
import os
import model

def getAxisIndexes(size, center):
    coordinates = []
    for i in range(-center, size-center):
        coordinates.append(i)
    return coordinates

def getCurrentIndexes(size, center):
    return getAxisIndexes(size[0], center[0]), getAxisIndexes(size[1], center[1])

def convolutionLayer(prevY, w, params):
    indxA, indxB = getCurrentIndexes(w.shape, params['centerOfCore'])
    bias = params['bias']
    currX = np.zeros((1,1))
    for i in range(prevY.shape[0]):
        for j in range(prevY.shape[1]):
            result = 0
            bIsElement = False
            for a in indxA:
                for b in indxB:
                    if i*bias - a >= 0 and j*bias - b >= 0 and i*bias - a < prevY.shape[0] and j*bias - b < prevY.shape[1]:
                        result += prevY[i*bias - a][j*bias - b] * w[a][b]
                        bIsElement = True
            if bIsElement:
                if i >= currX.shape[0]:
                    currX = np.vstack((currX, np.zeros(currX.shape[1])))
                if j >= currX.shape[1]:
                    currX = np.hstack((currX, np.zeros((currX.shape[0],1))))
                currX[i][j] = result
    return currX

```

```
37. def convolutionFeed(prevY, w, wSize, b, featureMaps, activationFunction, params):
38.     currX, currY  = []
39.     for i in range(len(prevY)):
40.         for j in range(i*featureMaps, (i + 1)*featureMaps):
41.             currX.append(convolutionLayer(prevY=prevY[i], w=w[j], params=params))
42.     resultX = []
43.     for i in range(featureMaps):
44.         resultX.append(0)
45.         for j in range(len(prevY)):
46.             resultX[-1] += currX[j*featureMaps + i]
47.         resultX[-1] += b[len(resultX)-1]
48.         currY.append(activationFunction(resultX[-1], fn_name=activationFunction,
    feed=True))
49.     return currY, w, b
50.
51. def maxpoolLayer(currY, params):
52.     indxA, indxB = getCurrentIndexes(size=params['window_shape'],
    center=params['center_window'])
53.     bias = params['bias']
54.     mp_currY = np.zeros((1,1))
55.     currYConverted = np.zeros((1,1), dtype='<U32')
56.     for i in range(currY.shape[0]):
57.         for j in range(currY.shape[1]):
58.             result = -np.inf
59.             bIsElement = False
60.             for a in indxA:
61.                 for b in indxB:
62.                     if i*bias - a >= 0 and j*bias - b >= 0 and i*bias - a <
    currY.shape[0] and j*bias - b < currY.shape[1]:
63.                         if currY[i*bias - a][j*bias - b] > result:
64.                             result = currY[i*bias - g*a][j*bias - g*b]
65.                             i_back = i*bias - a
66.                             j_back = j*bias - b
67.                         bIsElement = True
68.             if bIsElement:
69.                 if i >= mp_currY.shape[0]:
70.                     mp_currY = np.vstack((mp_currY, np.zeros(mp_currY.shape[1])))
71.                     currYConverted = np.vstack((currYConverted,
    np.zeros(currYConverted.shape[1])))
72.                 if j >= mp_currY.shape[1]:
73.                     mp_currY = np.hstack((mp_currY,
    np.zeros((mp_currY.shape[0],1))))
74.                     currYConverted = np.hstack((currYConverted,
    np.zeros((currYConverted.shape[0],1))))
75.                 mp_currY[i][j] = result
76.                 currYConverted[i][j] = str(i_back) + ',' + str(j_back)
77.     return mp_currY, currYConverted
78.
79. def getMaxpoolFeed(currY, params):
80.     mp_currY = []
81.     currYConverted = []
82.     for i in range(len(currY)):
83.         y1, y2 = maxpoolLayer(currY[i], params)
84.         mp_currY.append(y1)
```

```
85.         currYConverted.append(y2)
86.     return mp_currY, currYConverted
87.
88. def maxpoolBack(dEdy_l_mp, y_l_mp_to_y_l, y_l_shape):
89.     dEdy = []
90.     for i in range(len(dEdy_l_mp)):
91.         dEdy_l = np.zeros(y_l_shape)
92.         for k in range(dEdy_l_mp[i].shape[0]):
93.             for l in range(dEdy_l_mp[i].shape[1]):
94.                 coordinates = y_l_mp_to_y_l[i][k][l]
95.                 coordinate_row = int(coordinates[:coordinates.find(',')])
96.                 coordinate_col = int(coordinates[coordinates.find(',')+1:])
97.                 dEdy_l[coordinate_row][coordinate_col] = dEdy_l_mp[i][k][l]
98.         dEdy.append(dEdy_l)
99.     return dEdy
100.
101.    def backConvolutiondEdw(prevY, wSize, dEdx, params):
102.        indxA, indxB = getCurrentIndexes(wSize, params['centerOfCore'])
103.        bias = params['bias']
104.        dEdw = np.zeros((wSize[0], wSize[1]))
105.        for a in indxA:
106.            for b in indxB:
107.                result = 0
108.                for i in range(dEdx.shape[0]):
109.                    for j in range(dEdx.shape[1]):
110.                        if i*bias - a >= 0 and j*bias - b >= 0  and i*bias - a <
    prevY.shape[0] and j*bias - b < prevY.shape[1]:
111.                            result += prevY[i*bias - g*a][j*bias - g*b] *
    dEdx[i][j]
112.                dEdw[a][b] = result
113.        return dEdw
114.
115.    def backConvolutiondEdyprev(dEdx, w, yPrevSize, params):
116.        indxA, indxB = getCurrentIndexes(w.shape, params['centerOfCore'])
117.        bias = params['bias']
118.        dEdyprev = np.zeros((yPrevSize[0], yPrevSize[1]))
119.        for i in range(dEdyprev.shape[0]):
120.            for j in range(dEdyprev.shape[1]):
121.                result = 0
122.                for ix in range(dEdx.shape[0]):
123.                    for jx in range(dEdx.shape[1]):
124.                        if ix*bias - i in indxA and jx*bias - j in indxB:
125.                            a = indxA.index(ix*bias - i)
126.                            b = indxB.index(jx*bias - i)
127.                            result += dEdx[ix][jx] * w[a][b]
128.                            demo[ix][jx] = w[a][b]
129.                dEdyprev[i][j] = result
130.        return dEdyprev
131.
132.    def getStep(dir):
133.        try:
134.            step = np.load(dir).item().get('step') + 1
135.        except:
136.            step = 0
```

```
137.         return step
138.
139.     def getSaved(list, dir):
140.         try:
141.             saveList = np.load(dir).item().get(list)
142.         except:
143.             saveList = []
144.         return saveList
145.
146.     def fullConectedMult(prevY, w, b, activationFunction):
147.         x = np.dot(prevY, w) + b
148.         y = activationFunction(x, activationFunction, True)
149.         return y, w, b
150.
151.     def activationFunction(matrix, feed):
152.         output = np.copy(matrix)
153.         if feed:
154.             output = np.exp(output) / np.exp(output).sum()
155.         else:
156.             input = np.copy(matrix)
157.             output = np.zeros((matrix.shape[1], matrix.shape[1]))
158.             for i in range(output.shape[1]):
159.                 for j in range(output.shape[1]):
160.                     if i==j:
161.                         output[i][i] = input[0][i]*(1 - input[0][i])
162.                     else:
163.                         output[i][j] = -input[0][i]*input[0][j]
164.         return output
165.
166.     def lossFunction(yTruth, yPredicted, feed):
167.         if feed:
168.             error = - yTruth * np.log(yPredicted)
169.         else:
170.             error = - (yTruth/yPredicted)
171.         return error
172.
173.     def matrixToVector(matrix):
174.         vector = np.array([[]])
175.         for i in range(len(matrix)):
176.             reshaped_matrix = np.reshape(matrix[i], (1,
    matrix[i].shape[0]*matrix[i].shape[1]))
177.             vector = np.hstack((vector, reshaped_matrix))
178.         return vector
179.
180.     def vectorToMatrix(vector, size):
181.         matrices = []
182.         matrix_size = size[0]*size[1]
183.         for i in range(0, vector.size, matrix_size):
184.             matrix = np.reshape(vector[0][i:i+matrix_size], size)
185.             matrices.append(matrix)
186.         return matrices
187.
188.
189.     def fullConectedBackpropagation(prevY, dEdy, currY, w, b, alpha):
```

```
190.        dEdx = np.dot(dEdy, activationFunction(currY, fn_name=act_fn, feed=False))
191.        dEdw = np.dot(prevY.T, dEdx)
192.        dEdb = dEdx
193.        dEdyprev = np.dot(dEdx, w.T)
194.        w = w - alpha * dEdw
195.        b = b - alpha * dEdb
196.        return dEdyprev, w, b
197.
198.    def convolutionBackpropagation(prevY, currY, w, b, dEdy, featureMaps, alpha,
    params):
199.        dEdyprev = []
200.        dEdx = []
201.        for i in range(len(currY)):
202.            dEdx.append(dEdy[i] * activationFunction(currY[i], fn_name=act_fn,
    feed=False))
203.            dEdb = dEdx[-1].sum()
204.            b[i] = b[i] - alpha * dEdb
205.        for i in range(len(prevY)):
206.            dEdyprev = 0
207.            k = 0
208.            for j in range(i*featureMaps, (i + 1)*featureMaps):
209.                dEdw = backConvolutiondEdw( prevY[i], w[j].shape, dEdx[k], params)
210.                dEdyprev += backConvolutiondEdyprev(dEdx[k], w[j], prevY[i].shape,
    params)
211.                w[j] = w[j] - alpha * dEdw
212.                k += 1
213.            dEdyprev.append(dEdyprev)
214.        return dEdyprev, w, b
215.
216.    def shuffleDataSetAndArchive(imagesList, truthList, dirList):
217.        list_storage = zip(imagesList, truthList, dirList)
218.        list_storage = list(list_storage)
219.        np.random.shuffle(list_storage)
220.        imagesList, truthList, dirList = zip(*list_storage)
221.        return imagesList, truthList, dirList
222.
223.
224.    trainModel = True
225.
226.    imageS, truthS, dirS = np.load(
227.        './HASYv2',
228.        split=['train[:80%]', 'train[80%:90%]', 'train[90%:]'],
229.        with_info=True,
230.        as_supervised=True,
231.    )
232.
233.    dataSet = input_data.read_data_sets("./HASYv2/", one_hot=True)
234.
235.    imageS, truthS, dirS = model.shuffle_list(imageS, truthS, dirS)
236.
237.    model = {
238.        'learning_rate':0.01,
239.        'conv_shape_1':(2,2),
240.        'conv_shape_2':(3,3),
```

```
241.        'maxpool_shape_1':(2,2),
242.        'conv_feature_1':5,
243.        'conv_feature_2':20,
244.        'conv_stride_1':2,
245.        'conv_stride_2':1,
246.        'maxpool_stride_1':2,
247.        'fc_neurons_1':2000,
248.        'fc_fn_1':'softmax',
249.        'conv_center_1':(0,0),
250.        'conv_center_2':(1,1),
251.        'maxpool_center_1':(0,0)
252.    }
253.
254.    conv_w_1 = []
255.    conv_b_1 = []
256.    conv_w_2 = []
257.    conv_b_2 = []
258.    fc_w_1 = np.array([[]])
259.    fc_b_1 = np.array([[]])
260.    fc_w_2 = np.array([[]])
261.    fc_b_2 = np.array([[]])
262.
263.    if train_model:
264.        loss_change = model.getSaved('loss_change', weightDir)
265.        accuracy_change = model.getSaved('accuracy_change', weightDir)
266.
267.    test_loss = []
268.    test_accuracy = []
269.
270.    Epoch = 10
271.    for step in range(Epoch):
272.        image_id = step%len(imageS)
273.        input_image = [imageS[image_id]]
274.        y_true = truthS[image_id]
275.        conv_y_1, conv_w_1, conv_b_1 = model.convolutionFeed(
276.            input_image,
277.            conv_w_1,
278.            model['conv_shape_1'],
279.            conv_b_1,
280.            model['conv_feature_1'],
281.            weightDir,
282.            params={
283.                'convolution':model['conv_conv_1'],
284.                'bias':model['conv_stride_1'],
285.                'centerOfCore':model['conv_center_1']
286.            }
287.        )
288.        conv_y_1_mp, conv_y_1_mp_to_conv_y_1 = model.getMaxpoolFeed(
289.            conv_y_1,
290.            params={
291.                'window_shape':model['maxpool_shape_1'],
292.                'convolution':model['maxpool_conv_1'],
293.                'bias':model['maxpool_stride_1'],
294.                'center_window':model['maxpool_center_1']
```

```
295.            }
296.        )
297.        conv_y_2, conv_w_2, conv_b_2 = model.convolutionFeed(
298.            conv_y_1_mp,
299.            conv_w_2,
300.            model['conv_shape_2'],
301.            conv_b_2,
302.            model['conv_feature_2'],
303.            weightDir,
304.            params={
305.                'convolution':model['conv_conv_2'],
306.                'bias':model['conv_stride_2'],
307.                'centerOfCore':model['conv_center_2']
308.            }
309.        )
310.        conv_y_2_vect = model.matrix2vector_tf(conv_y_2)
311.        fc_y_1, fc_w_1, fc_b_1 = model.fullConectedMult(
312.            y_l_minus_1=conv_y_2_vect,
313.            w=fc_w_1,
314.            b=fc_b_1,
315.            neurons=model['fc_neurons_1'],
316.            dir_npy=weightDir
317.        )
318.        fc_y_2, fc_w_2, fc_b_2 = model.fullConectedMult(
319.            fc_y_1,
320.            fc_w_2,
321.            fc_b_2,
322.            len(y_true),
323.            weightDir
324.        )
325.        fc_error = model.lossFunction(y_true, fc_y_2, feed=True)
326.        test_loss.append(fc_error.sum())
327.        test_accuracy.append(y_true.argmax() == fc_y_2.argmax())
328.        if train_model:
329.            dEdfc_y_2 = model.lossFunction(y_true, fc_y_2, feed=False)
330.            dEdfc_y_1, fc_w_2, fc_b_2 = model.fullConectedBackpropagation(
331.                fc_y_1,
332.                dEdfc_y_2,
333.                fc_y_2,
334.                fc_w_2,
335.                fc_b_2,
336.                model['learning_rate']
337.            )
338.            dEdfc_y_0, fc_w_1, fc_b_1 = model.fullConectedBackpropagation(
339.                conv_y_2_vect,
340.                dEdfc_y_1,
341.                fc_y_1,
342.                fc_w_1,
343.                fc_b_1,
344.                model['learning_rate']
345.            )
346.            dEdconv_y_1_mp, conv_w_2, conv_b_2 = model.convolutionBackpropagation(
347.                conv_y_1_mp,
348.                conv_y_2,
```

```
                conv_w_2,
                conv_b_2,
                Edconv_y_2,
                model['conv_feature_2'],
                model['learning_rate'],
                params={
                    'convolution':model['conv_conv_2'],
                    'bias':model['conv_stride_2'],
                    'centerOfCore':model['conv_center_2']
                }
            )
            dEdconv_y_1 = model.maxpoolBack(
                dEdconv_y_1_mp,
                conv_y_1_mp_to_conv_y_1,
                conv_y_1[0].shape
            )
            dEdconv_y_0, conv_w_1, conv_b_1 = model.convolutionBackpropagation(
                input_image,
                conv_y_1,
                conv_w_1,
                conv_b_1,
                dEdconv_y_1,
                model['conv_feature_1'],
                model['learning_rate'],
                params={
                    'convolution':model['conv_conv_1'],
                    'bias':model['conv_stride_1'],
                    'centerOfCore':model['conv_center_1']
                }
            )
        if train_model and len(loss_change)%len(imageS) == 0:
            imageS, truthS, dirS = model.shuffle_list(imageS, truthS, dirS)

        template = 'Epoch {}, Loss: {}, Accuracy: {}, Test Loss: {}, Test
Accuracy: {}'
        print(template.format(epoch+1, loss_change, accuracy_change,
test_loss.result(), test_accuracy.result()))

    plt.plot(accuracy_change, label = 'train')
    plt.plot(test_accuracy, label = 'test')
    plt.xlabel('Epoch')
    plt.ylabel('Accuracy')
    plt.ylim([0.884, 0.8877])
    plt.legend(loc='lower right')
    plt.show()

    plt.plot(loss_change, label = 'train')
    plt.plot(test_loss, label = 'test')
    plt.xlabel('Epoch')
    plt.ylabel('Loss')
    plt.ylim([0.31, 0.33])
    plt.legend(loc='lower right'
```