МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Тернопільський національний економічний університет
Факультет комп’ютерних інформаційних технологій
Кафедра комп'ютерної інженерії

Мишківський Олександр Валерійович

Програмний модуль розпізнавання текстової інформації на основі штучних нейронних мереж/ The software module for textual information recognizing based on artificial neural networks

напрям підготовки: 6.050102 - Комп’ютерна інженерія
фахове спрямування - Комп’ютерні системи та мережі
Бакалаврська робота

Виконав студент групи КСМз-41/2
О.В. Мишківський

Науковий керівник:
Якименко І.З.

Тернопіль - 2018

ЗМІСТ

ВСТУП

Завдання розпізнавання тексту в комп'ютерному зорі є однією з найважливіших задач в машинному навчанні. За тривалий період часу було проведено безліч досліджень на цю тему, які в подальшому вплинули не тільки на розпізнавання тексту, а й на аналіз тексту в цілому.

У машинному навчанні завдання розпізнавання тексту є завданням класифікації, яка може бути вирішена різними підходами. Один з таких методів є вирішальні дерева, як запропоновано в [1]. Однак даний метод має труднощі при класифікації об'єктів, що мають велику кількість параметрів.

В [2] зроблена спроба порівняти різні методи машинного навчання. Найбільш цікавими серед порівнюваних методів є метод найближчих сусідів і штучна нейронна мережа. У результатах видно, що обидва методи показують непогані результати, проте метод найближчих сусідів також зазнає певних труднощів при класифікації об'єктів з великою кількістю атрибутів, що особливо важливо при класифікації зображень.

Для порівняння роботи нейронної мережі і методу найближчих сусідів в [3] використовували набір даних, що складаються з зображень цифр. Кількість навчальних об'єктів становило одинадцяти тисяч. В результаті експериментально було показано, що нейронна мережа краще підходить для вирішення завдань класифікації зображень. Також в роботі був запропонований кілька покращений метод, який підвищив точність розпізнавання.

Так як перераховані методи могли частково впоратися із завданням класифікації, цього все одно було недостатньо. З відкриттям згортальних нейронних мереж, завдання класифікації зображень отримала новий розвиток, і теоретична точність розпізнавання могла досягати до 99%, однак на практиці таке вдавалося далеко не завжди. тоді в [4] було запропоновано використовувати

згорткові нейронні мережі з деякими поліпшеннями. В результаті, точність досягала 70%, що є не найкращим результатом.

У підсумку, можна зробити висновок про те, що, незважаючи на численні дослідження, статті, нові методи, поліпшення існуючих методів, все одно класифікувати такі складні об'єкти, як зображення, вкрай важко з 100% точністю. Також, незважаючи на той факт, що в теорії точність класифікації може досягати 99,9%, на практиці це практично недосяжно. Середній показник точності в реальних задачах становить близько 80-90%, а іноді і не доходить до 70%. Виходячи з цього факту, основною метою даної роботи є поліпшення точності розпізнавання в вузьконаправленої завдання класифікації шляхом створення цілісного алгоритму.

Постановка задачі

Для виконання поставленої мети необхідно виконати наступні завдання:

– проаналізувати існуючі методи і алгоритми обробки зображень;

– проаналізувати методи машинного навчання;

– проаналізувати методи оптимізації та прискорення процесу навчання;

– розробити алгоритм обробки зображень, що дозволяє сегментувати текстову область на зображення на окремі символи;

– спроектувати нейронну мережу;

– підготувати навчальний набір даних;

– експериментальним шляхом підібрати гіперпараметри нейронної мережі;

– навчити нейронну мережу і протестувати мережу і модуль обробки зображень на тестових даних і даних, наближених до реальних;

– за результатами тестів зробити висновки і запропонувати подальші способи поліпшення алгоритму.

Критеріями та обмеженнями поставлених завдань є:

– обробляються тільки зображення, що містять друкований текст;

– текст може складатися тільки з цифр і букв англійського алфавіту в верхньому і нижньому регістрах;

– мінімальна допустима точність розпізнавання на тестових даних 80%;

– розроблений алгоритм повинен бути стійкий як до поворотів, так і до деяких дефектів друку.

1 АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ

1.1 Огляд і аналіз аналогів

В якості основних аналогів виступають такі сервіси, як:

- ABBY FineReader (http://finereaderonline.com/ru-ru);
- Free Online OCR service (https://www.onlineocr.net/);
- Convertio ().

Для прикладу взято тестові зображення, які показані на рисунку 1.1.

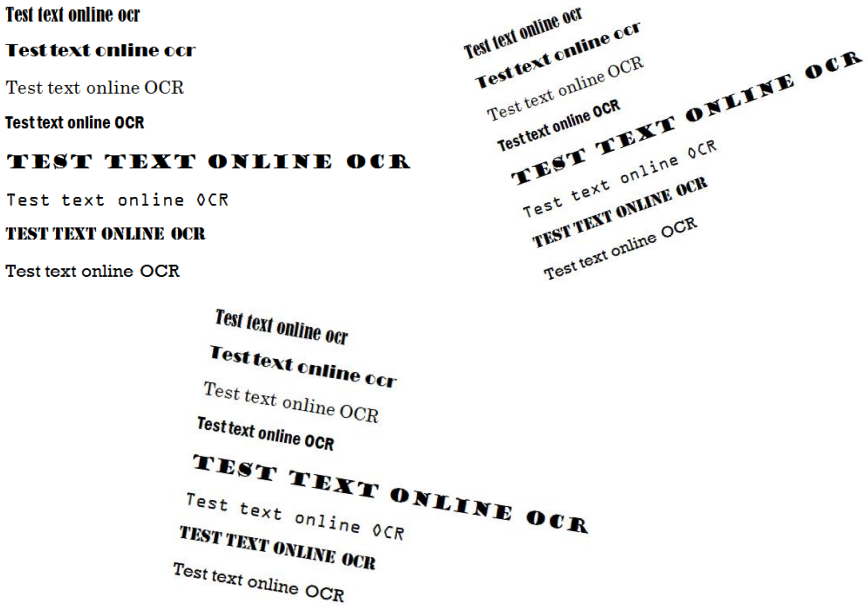


Рисунок 1.1 – Тестові зображення

Результати тестування кожного з сервісів представлено на рисунках 1.2-1.4:

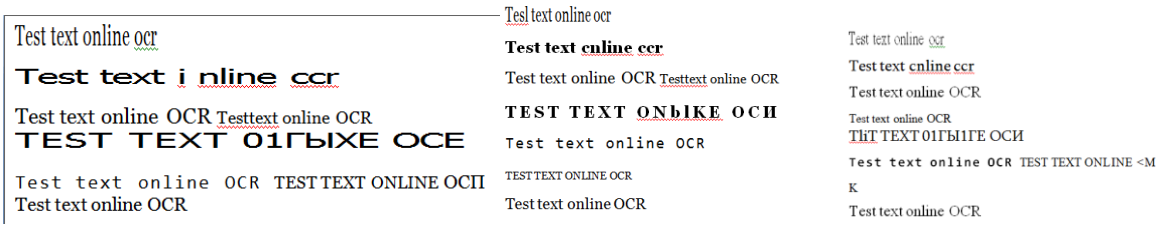


Рисунок 1.2 – Результати тестування Abby FineReader

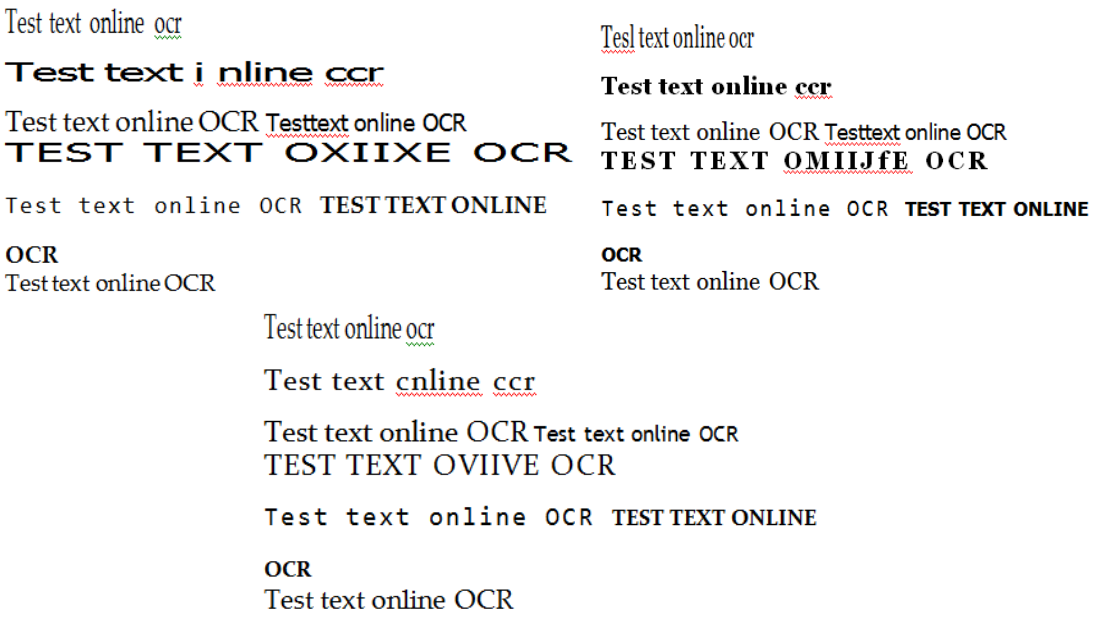


Рисунок 1.3 – Результати тестування сервісу Convertio

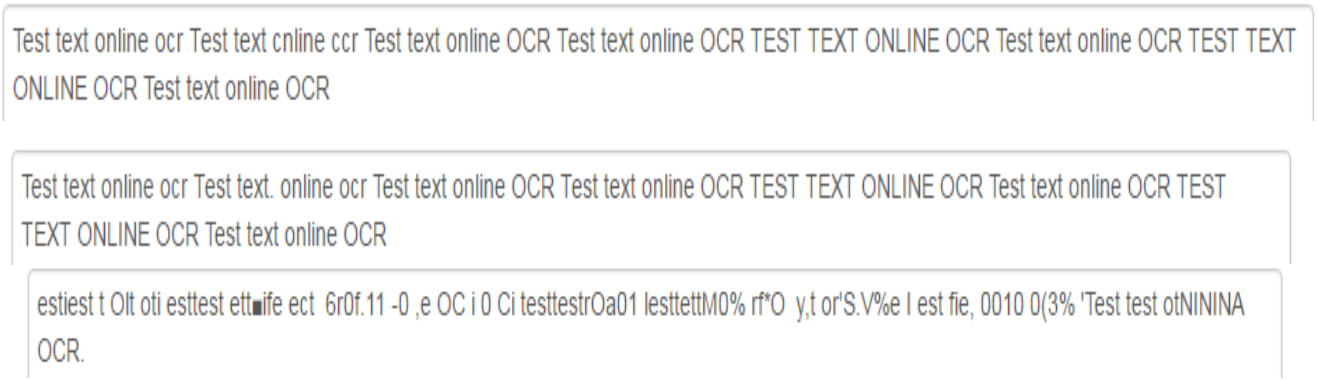


Рисунок 1.4 – Результати тестування сервісу Free online OCR service

Крім тестів на зображеннях з поворотами, були проведені тести з зображеннями з деякими дефектами друку. Всі сервіси, крім Abby FineReader змогли розпізнати даний документ з деякою кількістю помилок, яких було більше, ніж при простому повороті. Abby FineReader не зміг повністю розпізнати зображення через те, що сервіс нарахував велику кількість помилок і їм було прийнято рішення не повертати результат у вигляді документа, а повідомити про помилку. Порівняння сервісів розпізнавання тексту представлено в таблиці 1.1.

Таблиця 1.1 - Порівняльна таблиця сервісів розпізнавання тексту

	Стійкість до поворотів (якість розпізнавання)	Стійкість до дефектів друку	Збереження порядку символів і рядків
Abby FineReader	часткова	При виявленні великої кількості дефектів розпізнавання неможливо	Часткове збереження порядку рядків (якщо є поворот)
Free online OCR Service	часткова	практично відсутня	Часткове збереження порядку рядків (якщо є поворот)
Convertio	є	часткова	Часткове збереження порядку рядків (якщо є поворот)

Як видно з таблиці 1.1, у більшості сервісів присутні проблеми, пов'язані зі стійкістю до поворотів зображення і деяким дефектів друку. Через відсутність стійкості до поворотів, в деяких випадках порушується порядок проходження рядків. Також з-за повороту зображення деякі сервіси некоректно розпізнають деякі символи. Особливо ця проблема виражена при повороті зображення проти годинникової стрілки.

Виходячи з перерахованих недоліків, важливо, щоб розроблений алгоритм був максимально стійкий до цих ефектів, щоб він був здатний конкурувати з перерахованими аналогами.

1.1.1 Обгрунтування вибору програмних засобів

В якості основного мови програмування був обраний Python 2.7. Причин такого вибору кілька:

- простий синтаксис;
- велика кількість бібліотек, необхідних для наукових обчислень;

- наявність бібліотеки OpenCV для обробки зображень;
- можливість використання стороннього ПЗ для забезпечення оптимальної продуктивності, необхідної для машинного навчання.

Головним конкурентом у виборі мови програмування був C ++. Вибір був зроблений не на його користь лише через відсутність фреймворка Keras, який був обраний в якості основного для розробки нейронної мережі.

Програмне забезпечення, яке необхідного для забезпечення прискорення процесу навчання з використанням потужностей графічного процесора, було обрано модуль CUDA ToolKit від компанії Nvidia. Причиною цього стало те, що на використовуваному комп'ютері встановлена відеокарта GeForce GTX 770 від компанії Nvidia, тому використання модулів від інших виробників може викликати труднощі в подальшому.

Крім цього у розділі будуть проаналізовані класичні методи розв'язання задачі розпізнавання тексту, які включають в себе загальноприйняті методи попередньої обробки зображення і сучасні методи в машинному навчанні.

В кінці глави будуть підведені підсумки, в яких буде відображено поточний стан в рішенні поставленого завдання і основні проблеми, пов'язані з її рішенням.

1.2 Аналіз методів обробки зображень

Незважаючи на те, що розпізнавання тексту є приватним завданням класифікації в машинному навчанні, саме по собі завдання знаходження тексту на зображенні є завданням комп'ютерного зору. Зображення, що надходять на вхід, в переважній більшості випадків піддаються хоча б найменшій попередній обробці [5], щоб в подальшому виконати поставлене завдання.

Часто потрібно просто отримати якусь інформацію про зображення, наприклад, статистичні дані про насиченість кольору, або розкласти зображення на колірні складові.

Далі в цьому розділі буде представлений аналіз основних методів обробки зображень, які використовуються в класичному підході в завданні розпізнаванні тексту.

1.2.1 Кольорова палітра

Підбір кольорів (колірна модель) - це модель представлення кольору в системі колірних координат. Кольорова палітра дозволяє задати будь-який колір конкретної точкою в системі координат [6].

На даний момент існує безліч колірних моделей різних розмірностей. Найбільш популярними є RGB, CMY (K), Grayscale, HSV і HSI. Крім цих моделей існують і інші, але їх використання більш вузьконаправлено і не отримало широкого поширення. В алгоритмі попередньої обробки вхідних зображень найбільший інтерес викликають колірні моделі RGB і Grayscale.

Модель RGB - це адитивна модель, тобто кольори виходять шляхом складання трьох основних кольорів (червоний, синій і зелений). Адитивний синтез кольору, в якості способу отримання кольорових зображень, був запропонований Джеймсом Максвеллом в 1861 році.

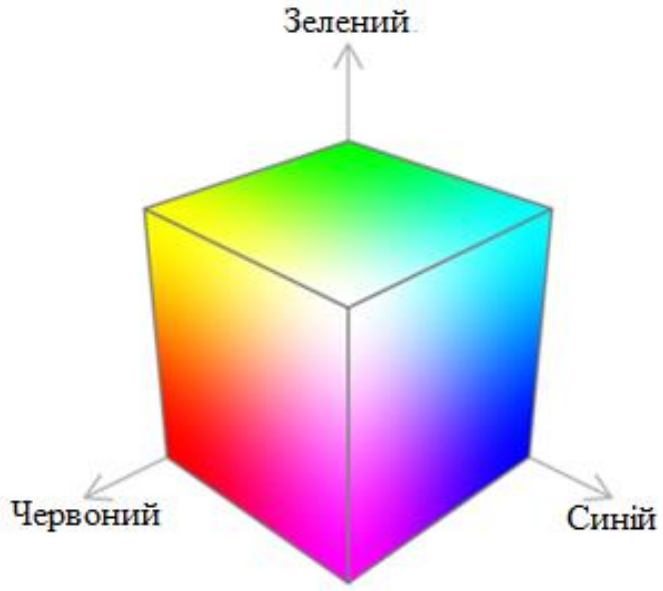


Рисунок 1.5 –Модель RGB

Модель Grayscale також відома, як «Сіра шкала» використовується для передачі чорно-білих зображень. Дана модель здатна передати 256 градацій сірого. Існує безліч способів перекладу кольорового зображення в чорно-біле[7][8]. Найпростіший метод - це взяття середнього значення (1.1):

$$\frac{(R+G+B)}{3}. \tag{1.1}$$

Даний метод є досить простим як в реалізації, так і в обчисленнях, проте все одно має серйозний недолік. Недолік полягає в тому, що такий підхід погано передає відтінок сірого щодо того, як людське око сприймає сірий. Позбутися від цього недоліку дозволяє зважена сума (1.2):

$$(0{,}3 \cdot R + 0{,}59 \cdot G + 0{,}11 \cdot B). \tag{1.2}$$

Є деякі модифікації даного підходу (1.3):

$$(0{,}299 \cdot R + 0{,}587 \cdot G + 0{,}114 \cdot B). \tag{1.3}$$

Ще одним методом є пошук середнього значення між найбільш насиченим кольором і найменш насиченим кольором (1.4):

$$\frac{((Max(R,G,B) + Min(R,G,B))}{2}. \tag{1.4}$$

Існує також і метод розкладання зображення (1.5), (1.6):

$$Max(R,G,B), \tag{1.5}$$

$$Min(R,G,B). \tag{1.6}$$

Максимальне розкладання зробить чорно-біле зображення яскравіше, а мінімальне, навпаки - темніше. Такий підхід часто застосовують для отримання художнього ефекту, наприклад, в обробці відео або фото.

Цікавий ефект виходить при перетворенні кольорового зображення в чорно-біле, використовуючи метод розкладання за кольором (1.7):

$$R = G = B. \tag{1.7}$$

Певна кількість цифрових камер використовувало цей алгоритм для отримання чорно-білих фотографій.

1.3 Трансформація зображень

Трансформація зображення - це операція деформації піксельної сітки вихідного зображення, не змінюючи вихідного вмісту, і відображення зміненої піксельної сітки на цільове зображення.

У разі попередньої обробки зображення з метою подальшої його передачі на вхід нейронної мережі, зображення також буде потрібно збільшити в розмірах. Як і багато інших трансформації, збільшення зображення відбувається за рахунок застосування трансформуючої матриці. Для того, щоб отримати дану матрицю, необхідно вибрати чотири точки на оригінальному документі і 4 точки на цільовому зображенні. У підсумку, знаходження цієї матриці зводиться до вирішення наступної системи рівнянь (1.8) [9]:

$$\begin{bmatrix} x_0^s & y_0^s & 1 & 0 & 0 & 0 & -x_0^s * x_0^d & -x_0^s * x_0^d \\ x_1^s & y_1^s & 1 & 0 & 0 & 0 & -x_1^s * x_1^d & -x_1^s * x_1^d \\ x_2^s & y_2^s & 1 & 0 & 0 & 0 & -x_2^s * x_2^d & -x_2^s * x_2^d \\ x_3^s & y_3^s & 1 & 0 & 0 & 0 & -x_3^s * x_3^d & -x_3^s * x_3^d \\ 0 & 0 & 0 & x_0^s & y_0^s & 1 & -x_0^s * y_0^d & -x_0^s * y_0^d \\ 0 & 0 & 0 & x_1^s & y_1^s & 1 & -x_1^s * y_1^d & -x_1^s * y_1^d \\ 0 & 0 & 0 & x_2^s & y_2^s & 1 & -x_2^s * y_2^d & -x_2^s * y_2^d \\ 0 & 0 & 0 & x_3^s & y_3^s & 1 & -x_3^s * y_3^d & -x_3^s * y_3^d \end{bmatrix} * \begin{bmatrix} k_{00} \\ k_{01} \\ k_{02} \\ k_{10} \\ k_{11} \\ k_{12} \\ k_{20} \\ k_{21} \end{bmatrix} = \begin{bmatrix} x_0^d \\ x_1^d \\ x_2^d \\ x_3^d \\ y_0^d \\ y_1^d \\ y_2^d \\ y_3^d \end{bmatrix} \quad (1.8)$$

де x_i^sіy_i^s - це координати точок вихідного зображення;

x_i^d і y_i^d - результуючого.

Необхідно знайти вектор коефіцієнтів *k*. Після цього необхідно отримати сингулярне розкладання, яке і буде шуканою матрицею[10].

Після того, як була отримана трансформуюча матриця, до вихідного зображення застосовується перспективна трансформація за такою формулою (1.9) [11]:

$$dst(x,y) = src\left(\frac{M_{11}x + M_{12}y + M_{13}}{M_{31}x + M_{32}y + M_{33}}, \frac{M_{21}x + M_{22}y + M_{23}}{M_{31}x + M_{32}y + M_{33}}\right). \quad (1.9)$$

Завдяки такому підходу, можна зручним чином маніпулювати розмірами вихідного зображення, так як перспективна трансформація дозволяє змінювати піксельну сітку, не спотворюючи вміст.

Поточний крок попередньої обробки зображення полягає в тому, щоб описаним раніше методом збільшити вхідне зображення в два рази.

1.3.1 Граничний поділ

Граничий поділ є однією з найпростіших операцій сегментації зображення. Суть порогового поділу грунтується на ідеї того, що пікселі ключових об'єктів і фонових об'єктів на зображенні мають різну інтенсивність. З огляду на цей факт,

можна відокремити фон від ключових об'єктів, використовуючи певний порогове значення (рисунок 1.6). Порівнюючи кожен піксель з граничним значенням, в залежності від результатів порівняння, можна виставити нове порогове значення пікселя [12, 13].

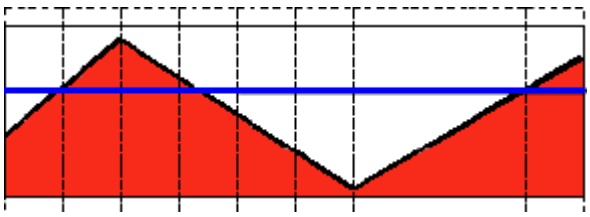


Рисунок 1.6– Графічне представлення роботи порогового поділу

Змінюючи порогове значення, можна домогтися різних результатів сегментації.

Існує безліч методів порогового поділу. Деякі з них дозволяють отримати двокольорове зображення, і такі методи є методами бінаризації. Інші дозволяють видалити з зображення лише певний фон, залишивши при цьому всю решту колірну гаму. Всі ці методи відносяться до класу методів з глобальним порогом, так як граничне значення фіксоване [14].

Бінаризація є однією з операцій порогового поділу. Результатом бінаризації є бінарне зображення. Під бінарним зображенням найчастіше мається на увазі чорно-біле зображення, проте можна домогтися і іншого результату. Основним критерієм бінаризації є порогове значення, яке є критерієм інтенсивності окремо взятої точки зображення (рисунок 1.7).

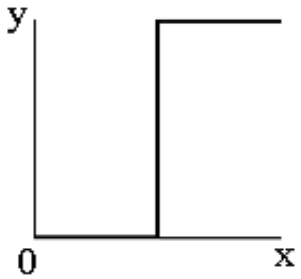


Рисунок 1.7 – Загальний випадок роботи функції бінаризації

Існує деякий безліч методів бінаризації, кожен з яких підходить під ту чи іншу задачу [15].

Одним з найбільш простих методів отримання бінарного зображення є метод бінаризації з нижнім порогом (1.10).

$$f'(x,y) = \begin{cases} \max Value, \text{ якщо } f(x,y) < threshold \\ 0, \text{ якщо } f(x,y) > threshold \end{cases}. \quad (1.10)$$

В даному випадку, якщо піксель з координатами x і y має інтенсивність більше порогового значення, то нове значення буде дорівнює 0, що еквівалентно чорному кольору, інакше нове значення стане рівним $\max Value$, яке еквівалентно білому кольору (рисунок1.8.).

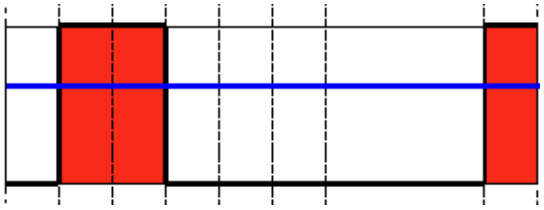


Рисунок 1.8 – Бінаризація з нижнім порогом

Для отримання, так званого, негативу зображення слід використовувати бінаризацію з верхнім порогом (1.11).

$$f'(x,y) = \begin{cases} \max Value, \text{ якщо } f(x,y) > threshold \\ 0, \text{ якщо } f(x,y) < threshold \end{cases}. \quad (1.11)$$

Результатом застосування даного методу до вихідного зображення стане його негатив.

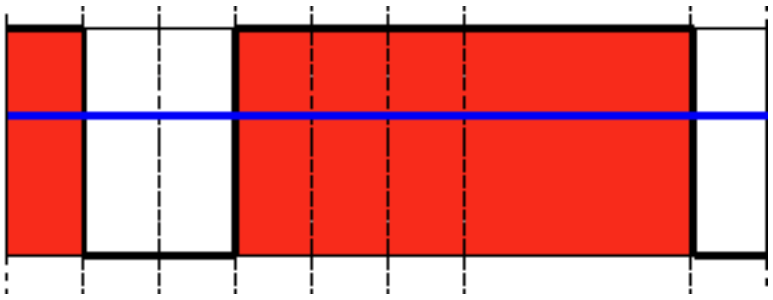


Рисунок 1.9 - Бінарна функція для отримання негативу зображення

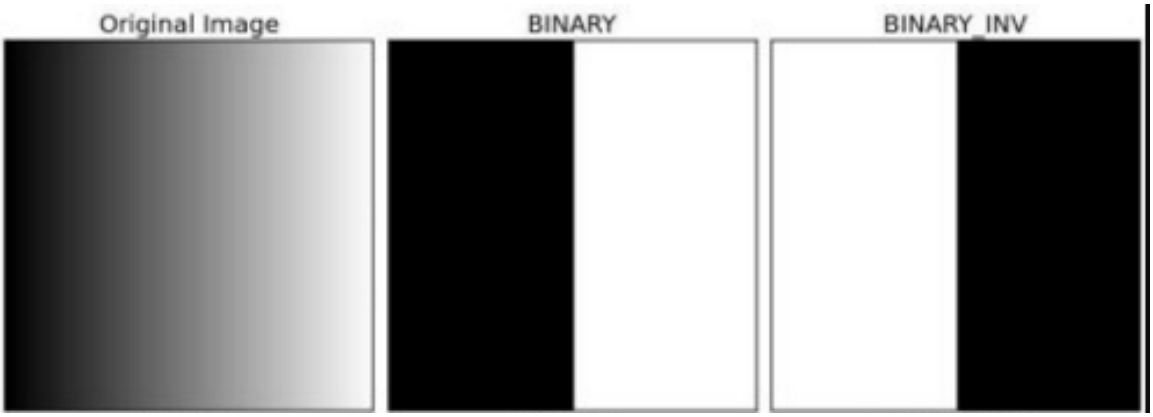


Рисунок 1.10- Приклад роботи методів бінаризації. Binary - метод з нижнім порогом; Binary_inv - метод з верхнім порогом

Крім методів, що реалізують операцію бінаризації, існують методи, здатні відсікати лише деякі кольори.

Метод порогового поділу з урізанням (1.12) дозволяє отримати зображення, у якого дуже високі значення усічені до порогового значення. Це дозволяє, наприклад, зменшити яскравість деяких об'єктів на зображенні.

$$f'(x,y) = \begin{cases} threshold, \text{ якщо } f(x,y) > threshold \\ f(x,y), \text{ якщо } f(x,y) < threshold \end{cases}. \quad (1.12)$$

У підсумку, графік значень інтенсивностей буде виглядати так (рисунок 1.11):

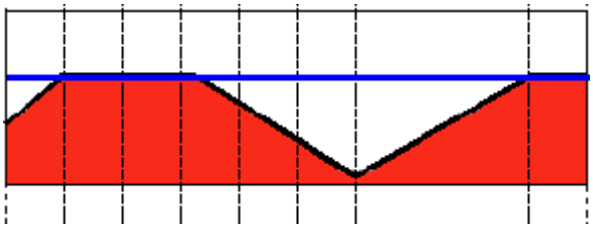


Рисунок 1.114 -. Бінаризація з урізанням

Двома іншими операціями порогового поділу є операція прямого видалення фону (1.13) (рисунок 1.12) і її зворотня версія (1. 14) (рисунок 1.13), що дозволяє, навпаки, залишити фон і видалити інші об'єкти.

$$f'(x,y) = \begin{cases} 0, \textit{якщо } f(x,y) < threshold \\ f(x,y), \textit{якщо } f(x,y) > threshold \end{cases}. \quad (1.13)$$

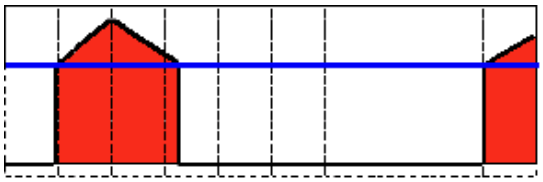


Рисунок 1.12 - Пряме видалення фону

$$f'(x,y) = \begin{cases} 0, \textit{якщо } f(x,y) > threshold \\ f(x,y), \textit{якщо } f(x,y) < threshold \end{cases}. \quad (1.14)$$

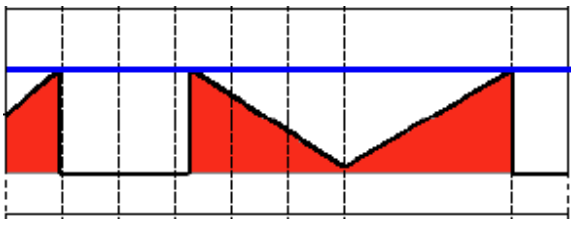


Рисунок 1.13 - Зворотнє видалення фону

Результат роботи перерахованих методів, а саме: Trunc - усічення верхніх меж інтенсивності; Tozero - пряме видалення фону; Tozero_inv - зворотне видалення фону представлено на рисунку 1.14.

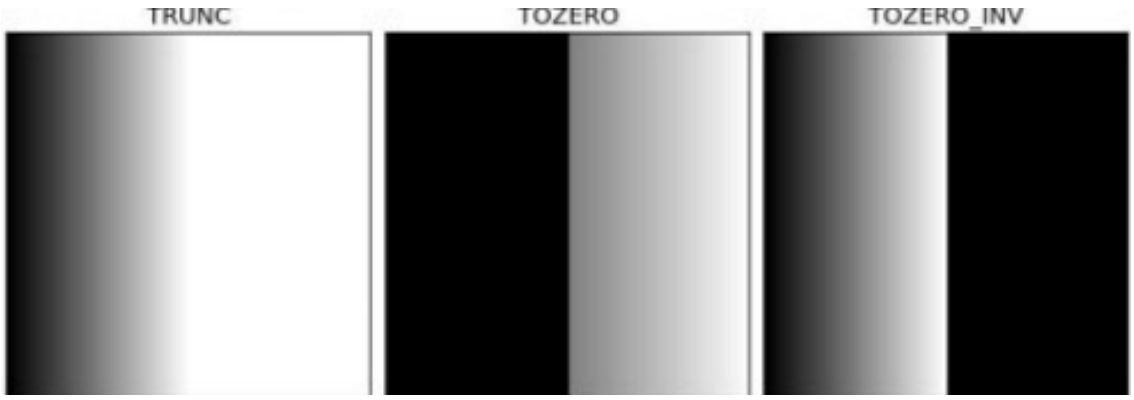


Рисунок 1.14- Результат роботи перерахованих методів. Trunc - усічення верхніх меж інтенсивності; Tozero - пряме видалення фону; Tozero_inv - зворотне видалення фону

1.3.2 Адаптивний пороговий поділ

Незважаючи на те, що для більшості завдань цілком підходять звичайні методи порогового поділу з сталим граничним значенням, існують такі завдання, для яких фіксований поріг не підходить. Такі завдання вирішуються за допомогою методів, які здатні змінювати поріг динамічно. Такі методи називаються методами адаптивного порогового поділу [16].

Суть методів адаптивного порогового поділу полягає в адаптації порога до поточного оточенню (групі пікселів) [17]. Особлива перевага таких методів помітно у випадках сильного ефекту градієнта через висвітлення або відображення [18].

Одним з подібних методів є метод, в результаті якого окремо взятому пікселю присвоюється значення рівне середньому значенню (1.15) між максимальною інтенсивністю і мінімальної в групі сусідніх пікселів [19].

$$M = \frac{MAX + MIN}{2}. \qquad (1.15)$$

Важливим параметром даного методу є розмір «вікна», яке задається розміром матриці. Пікселі, що потрапили в це вікно, будуть вважатися сусідніми і

щодо них буде розраховано середнє значення інтенсивності. Якщо вікно буде занадто великим, то негативні ефекти, такі як ефект градієнта від освітлення або відображення, будуть сильніше впливати на результат. Порівняння методів бінаризації зліва-направо: оригінал зображення, звичайна бінаризація, адаптивний метод представлено на рисунку 1.15.

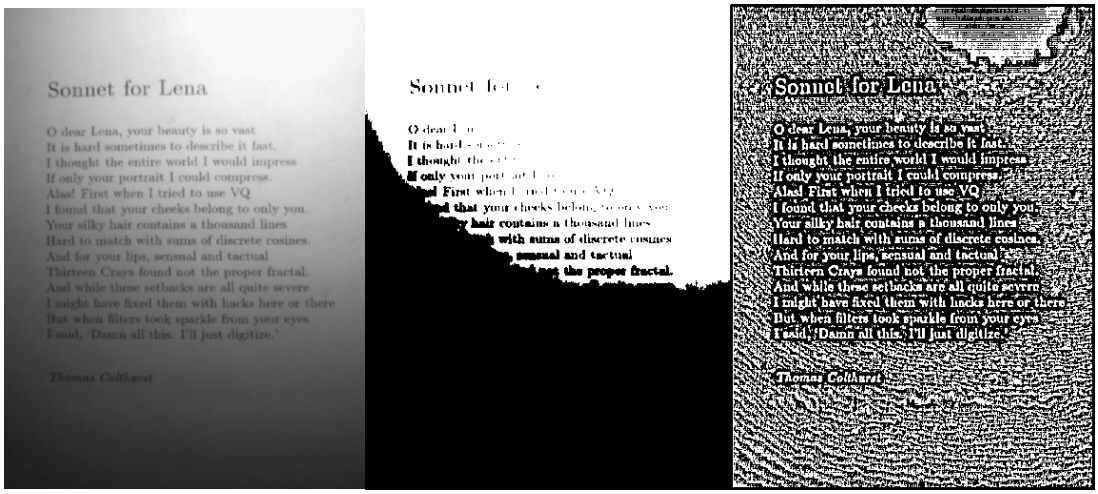
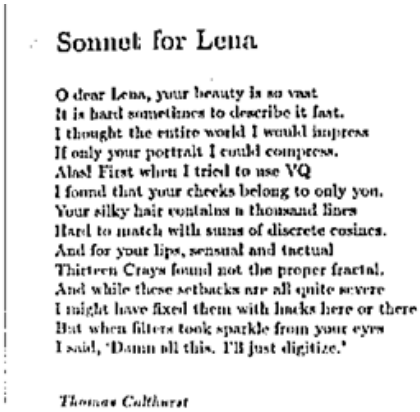


Рисунок 1.15- Порівняння методів бінаризації. Зліва-направо: оригінал зображення; звичайна бінаризація; адаптивний метод

Як видно з результатів, метод бінаризації з сталим значенням показав результат набагато гірший, ніж адаптивний метод, так як втрачена велика інформативна частина і безліч дрібних деталей. Незважаючи на те, що метод з використанням середнього значення зберіг основні деталі зображення, результат все одно не є прийнятним для таких завдань, як задача розпізнавання тексту. Для того щоб поліпшити результат, було запропоновано обчислити деяку константу з отриманого середнього значення [19].

Приклад роботи адаптивного методу з використанням додаткової константи представлено на рисунку 1.16.

Інший метод адаптивного порогового поділу використовує розподіл Гаусса. В даному випадку, також розраховується середнє значення, але всі пікселі в заданому вікні мають певну вагу, який залежить від віддаленості даного пікселя від центру [20].



Sonnet for Lena

O dear Lena, your beauty is so vast
It is hard sometimes to describe it fast.
I thought the entire world I would impress
If only your portrait I could compress.
Alas! First when I tried to use VQ
I found that your cheeks belong to only you.
Your silky hair contains a thousand lines
Hard to match with sums of discrete cosines.
And for your lips, sensual and tactual
Thirteen Crays found not the proper fractal.
And while these setbacks are all quite severe
I might have fixed them with hacks here or there
But when filters took sparkle from your eyes
I said, 'Damn all this. I'll just digitize.'

Thomas Colthurst

Рисунок 1.16 - Приклад роботи адаптивного методу з використанням додаткової константи

Порівняння трьох методів: Global thresholding - звичайна бінаризація, Adaptive Mean - адаптивний метод з пошуком середнього значення, Adaptive Gaussian - адаптивний метод з розрахунком розподілу Гаусса представлено на рисунку 1.17.

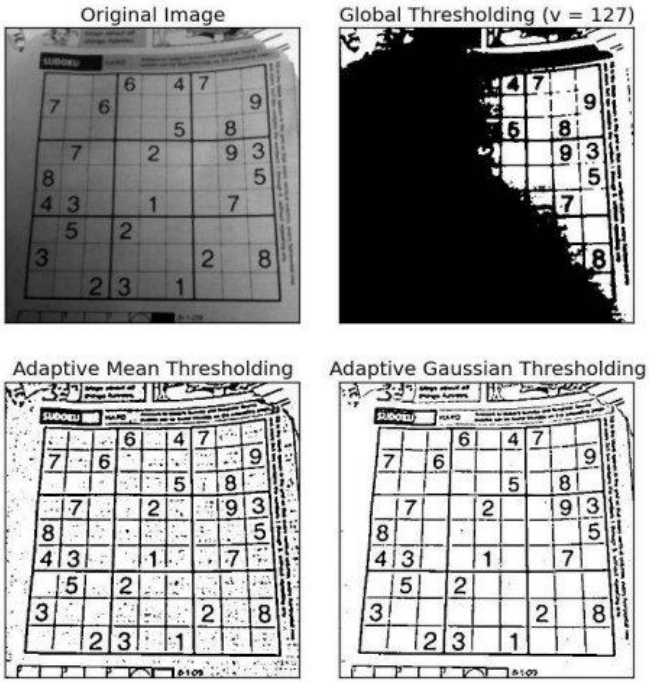


Рисунок 1.17- Порівняння трьох методів. Global thresholding - звичайна бінаризація; Adaptive Mean - адаптивний метод з пошуком середнього значення; Adaptive Gaussian - адаптивний метод з розрахунком розподілу Гаусса.

1.3.3 Пошук контурів символів

Заключна частина обробки зображення полягає в пошуку контурів об'єктів, які в даному випадку є символами. Більшість методів пошуку контурів працюють тільки з бінарними зображеннями, в яких об'єкти представлені чорним кольором, а фон - білим [21].

Одним з методів пошуку контурів є Pixel-Following Method [22], який проходить по межах досліджуваного об'єкта, дотримуючись певних правил руху. По ходу проходження по контурах досліджуваного об'єкта, пікселі контурів записуються у вигляді пар значень координат *x* і *y*. Незважаючи на його простоту, він має кілька недоліків:

- необхідність в окремому буфері пам'яті з розмірами *Xw* * *Yh*, де *Xw* - ширина зображення, *Yh* - висота зображення;
- нестійкість до змін розміру зображення, так як після змін необхідно заново знаходити контури.

У разі розпізнавання тексту недолік, пов'язаний зі змінами розміру вихідного зображення, не критичний, так як вихідне зображення передбачається змінювати в розмірі одного разу. Однак недолік, пов'язаний з витратою пам'яті, є істотним, тому що не завжди текст буде займати весь піксельний простір, тому частина виділеної пам'яті залишиться невикористаною. Приклад роботи Pixel-Following Method представлено на рисунку 1.18.

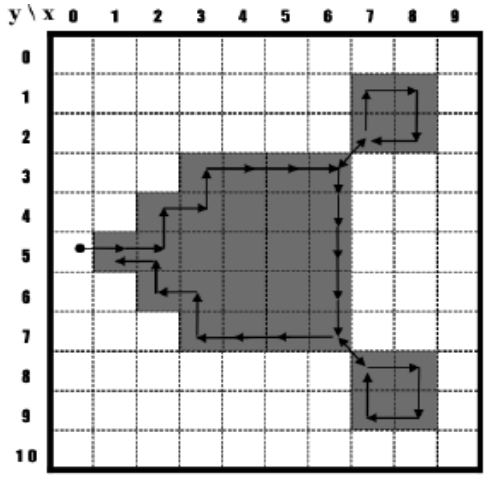


Рисунок 1.18 - Приклад роботи Pixel-Following Method

Покращенням описаного методу є метод Vertex-Following Method [23], який в деякій мірі вирішує проблему, пов'язану зі змінами розміру зображення, так як метод оперує координатами кутів. Однак проблема з нераціональним використанням пам'яті невирішене, так як даний метод використовує стільки ж пам'яті, скільки і Pixel-Following Method. Приклад роботи Vertex-Following Method представлено на рисунку 1.19.

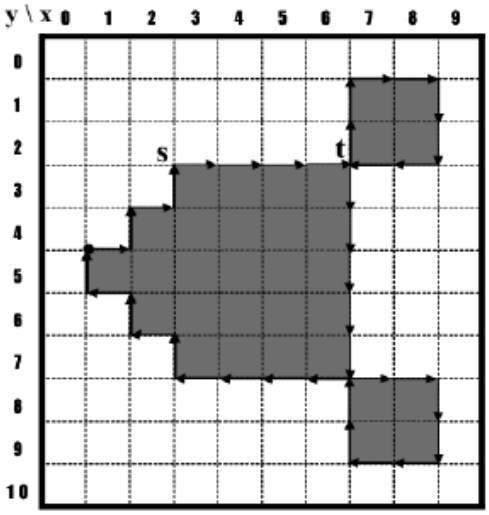


Рисунок 1.5 – Приклад роботи Vertex-Following Method

Найбільш вдалим методом в плані витрати пам'яті є метод Run-Data-Based Following Method [24], [25]. Принцип роботи даного методу полягає в послідовному скануванні горизонтальних ліній зображення зліва направо. При знаходженні певного контуру, в пам'яті записуються його лівий і правий кордони.

У подальшому аналізі, записані пари також використовуються для формування більш складних зв'язків, які, в кінцевому підсумку, утворюють контури об'єкта. Приклад роботи Run-Data-Based Following Method представлено на рисунку 1.20.

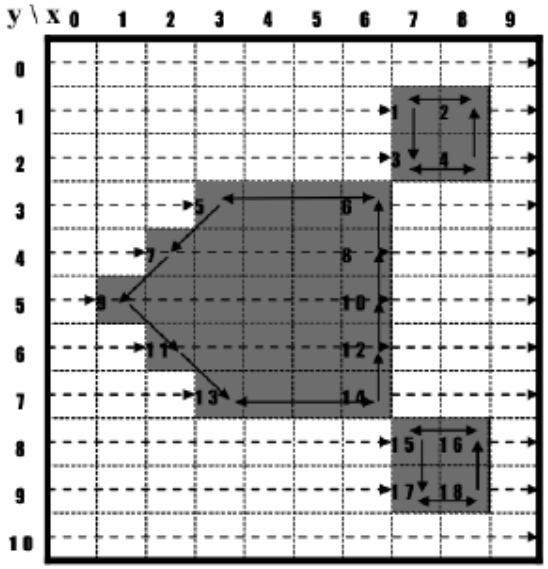


Рисунок 1.20 - Приклад роботи Run-Data-Based Following Method

Отже, в даному розділі було проведено аналіз методів обробки зображень, машинного навчання та оптимізації процесу навчання. На основі чого були встановлені переваги і недоліки на основі яких обґрунтовано вибір підходів.

2 НЕЙРОМЕРЕЖЕВІ ТЕХНОЛОГІЇ ДЛЯ ОБРОБКИ ЗОБРАЖЕНЬ

2.1 Вихідний алгоритм пошуку тексту на зображенні

Розглянутий вихідний алгоритм пошуку тексту на зображенні має наступну послідовність дій (рисунок 2.1):

Рисунок 2.1 - Блок схема алгоритму пошуку тексту на зображенні

Переваги даного алгоритму:

- висока швидкість виконання через невеликої кількості операцій;
- простота реалізації.

До недоліків даного алгоритму можна віднести наступне:

- відсутня впорядкованість знайдених контурів. Для обробки тексту це критично, так як отримані символи вихідного тексту будуть оброблені в хаотичному порядку. У підсумку, отриманий текст вийде нечитабельним;
- якщо символ має дефект у вигляді розриву, то він може детектуватися як кілька символів;
- деякі пари символів детектируются як один символ.

Таким чином, для правильної обробки зображення для подальшого розпізнавання тексту необхідно усунути перераховані недоліки.

2.1.1 Аналіз методів і алгоритмів машинного навчання

Машинне навчання - це процес, в результаті якого комп'ютер здатний показувати поведінку, якої не було закладено в нього явно [26]. Дане визначення не є єдино вірним, проте воно досить точно характеризує суть машинного навчання. Розробками в машинному навчанні почали займатися ще в середині 20 століття. Однією з причин необхідності розробок методів машинного навчання був той факт, що деякі завдання не можна було вирішити, запрограмувавши їх одного разу, тому що вхідні дані могли змінюватися. Машинне навчання вирішує цю проблему таким чином, що деякі методи дозволяють перенавчитися на нових вхідних даних, або якимось іншим чином підлаштуватися під них, щоб в результаті отримати вірний результат [27]. За весь час досліджень було розроблено безліч алгоритмів, кожен з яких найбільш вдало підходить до тієї чи іншої області застосування. Машинне навчання включає в себе використання серйозного математичного апарату, а саме: математичний аналіз, лінійну алгебру, дискретну математику, методи оптимізації, теорію ймовірностей і інші. З тих пір машинне навчання знайшло своє застосування в багатьох областях:

- комп'ютерне зір;
- розпізнавання мови;
- комп'ютерна лінгвістика;
- медицина;

- інтелектуальні ігри;
- фінансові додатки;
- та інші.

Машинне навчання дозволяє вирішити такі завдання, як [28]:

- класифікація;
- регресія;
- прогнозування;
- прийняття рішень;
- ранжування;
- кластеризація;
- і деякі інші.

У разі розпізнавання тексту таким завданням є класифікація.

Машинне навчання має на увазі наявність будь-яких навчальних даних. Залежно від характеру вихідних даних, машинне навчання можна розділити на три основні типи[29]:

- дедуктивне навчання - це навчання на основі знань, правил або інструкцій, які спеціальним чином формалізовані і з яких в подальшому виводяться нові правила;
- індуктивне навчання - це навчання на основі емпіричних даних, використовуючи які, машина створює якесь загальне правило, яке частіше виражено у вигляді якоїсь функції. Індуктивне навчання поділяється на деякий безліч типів;
- комбіноване навчання.

Індуктивне навчання включає в себе навчання з учителем і навчання без учителя. Завдання класифікації відноситься до навчання з учителем.

Навчання з учителем передбачає наявність певної кількості об'єктів *X*, які представлені рядом ознак, і безлічі відповідей або результатів *Y*, які також називаються мітками об'єктів. Між цими множинами існує якась залежність. Залежність відповідей від об'єктів відома тільки на об'єктах з навчальної вибірки.

Завдання навчання з учителем полягає в тому, щоб відновити залежність, тобто навчитися передбачати відповіді $y \in Y$ по нових об'єктах $x \in X$ [30].

Як уже було згадано, об'єкти з безлічі X описуються рядом ознак. Кожна ознака об'єкта також має деякі властивості: $x \in X = Q_1 \ldots Q_n$:

- якщо $Qj = R$, то ознака кількісна;
- якщо Qj звичайна, то ознака є номінальною (при цьому, якщо $|Qj|=2$, то така ознака називається бінарною);
- якщо Qj звичайна і впорядкована, то така ознака називається порядкованою.

Існує кілька способів створення простору ознак опису об'єктів:

- відбір значимих ознак;
- синтез нових ознак з уже існуючих.

Якщо перший спосіб цілком можна виконати вручну, за допомогою фахівців в конкретній предметній області, то другий краще довірити машині, щоб вона сама вирішила, які ознаки необхідно синтезувати для того, щоб отримати більш точний результат.

Інформацію про об'єкти зручно записувати у вигляді матриці, кожен рядок якої - це об'єкт, а кожен стовпець - це ознака або атрибут. Очікувані відповіді також записують у вигляді матриці. Виходячи з безлічі Y, можна виділити кілька типів завдання навчання:

- якщо Y звичайне, то це завдання класифікації. Часто використовується, наприклад, в задачах розпізнавання. У підсумку, завдання машини зводиться до того, щоб визначити, до якого класу належить об'єкт x;
- якщо $Y = R$, то це завдання відновлення регресії. Таке завдання зводиться до визначення функції f з певного класу, яка апроксимує невідому залежність.

Метод навчання без вчителя відрізняється від методу навчання з учителем тим, що на вході відомі тільки описи безлічі об'єктів. Під час навчання потрібно відновити взаємозв'язок об'єктів між собою [31].

Даний метод часто протиставляється методу навчання з учителем з точки зору людської фізіології, так як багато методів у своїй основі намагаються повторити процеси, що протікають в мозку. Підхід навчання з учителем піддавався критиці з-за того, що був далекий від реального процесу навчання людини, так як важко було б уявити те, що мозок людини під час навчання порівнював передбачуваний результат з дійсним. Саме ця відмінність вигідно відрізняє метод навчання без учителя, так як цей метод є найбільш правдоподібною моделлю навчання, з точки зору процесів, що протікають в мозку.

У разі розпізнавання тексту основними розглянутими методами машинного навчання будуть методи навчання з учителем.

2.1.2 Логістична регресія

Логістична регресія - метод побудова лінійного класифікатора. Дана модель описує апостеріорну ймовірність належності об'єкта до певного класу [32]. Для отримання ймовірності належності об'єкта до класу в межах від 0 до 1 використовується сигмоїдальна функція (рисунок 2.2) (1.16).

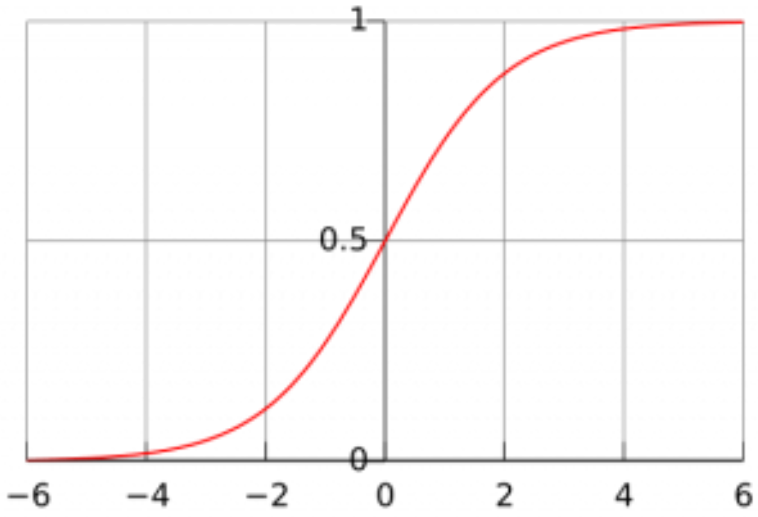


Рисунок 2.2 - Вид сигмодальної функції

$$g(Z) = \frac{1}{1+e^{-z}}\,. \tag{2.1}$$

Навчена модель має певний набір коефіцієнтів, кількість яких залежить від кількості ознак об'єктів з навчальної вибірки. У підсумку, модель передбачення класу можна представити у вигляді:

$$h_\theta(X) = g\left(Q_1 x_1 + Q_2 x_2 + \ldots + Q_n x_n\right), \tag{2.2}$$

де - i-ий коефіцієнт, θ_i;

x_i - i-та ознака вхідного об'єкта.

Незважаючи на те, що у визначенні згадується про лінійний класифікатор, дана модель дозволяє будувати і нелінійні класифікатори. Нелінійність досягається за рахунок змін у функції $h_\theta(X)$, наприклад:

$$h_\theta(X) = g\left(\theta_1 x_1 + \theta_2 x_1 x_2 + \ldots + \theta_n x_n^n\right). \tag{2.3}$$

Функція втрат у логістичній регресії має вигляд [33]:

$$J(\theta) = \sum_{i=1}^{m}\left(y^{(i)} \log h_\theta\left(x^{(i)}\right) + \left(1 - y^{(i)}\right)\log\left(1 - h_\theta\left(x^{(i)}\right)\right)\right), \tag{2.4}$$

де m - загальна кількість обєктів навчальної вібірки;

$y^{(i)}$ - клас, якому належить i – тий об’єкт;

$h_\theta\left(x^{(i)}\right)$ - ймовірність належності об’єкта i до класу $y^{(i)}$.

Використання логарифмічних функцій до функцій втрат дозволяє досягти кращої збіжності алгоритму. Основним методом навчання є метод градієнтного спуску.

Описана раніше модель здатна класифікувати об'єкт серед двох класів. Для класифікації об'єкта серед більшої кількості об'єктів використовується підхід «один проти всіх» (one-vs-all). Даний підхід полягає в тому, що модель навчається окремо для кожного класу. При класифікації об'єкта на виході передбачення є масив значень, які характеризують ймовірність приналежності даного об'єкту до одного з відомих моделі класів.

Переваги методу:

- простота реалізації;
- нескладний в розумінні;
- показує хороші результати на лінійно–роздільних даних.

Недоліки:

- незважаючи на можливість будувати нелінійну класифікаційну площину, все одно має не найвищу точність класифікації на лінійно нероздільних об'єктах;
- необхідність у використанні додаткових підходів, при кількості класів> 2, що збільшує час навчання;
- сильно залежить від вхідних ознак, особливо в разі побудови нелінійного класифікатора;
- піддається ефекту перенавчання.

2.1.3 Метод найближчих сусідів

Метод найближчих сусідів побудований на ідеї схожості об'єктів, які знаходяться близько один до одного. Близькість одного об'єкта до іншого виражається через відстань, яку можна порахувати наступним чином:

$$d(x, x') = \sqrt{\sum_{i=1}^{n} (x_i - x_i')^2} \,. \qquad (2.5)$$

Після того, як були пораховані всі відстані між досліджуваним об'єктом і об'єктами з навчальної вибірки, вибирається *k*-найближчих об'єктів. Після цього, виходячи з того, до якого класу належить більше число обраних об'єктів, робиться висновок про належність досліджуваного об'єкта до певного класу.

Ключовим параметром методу є кількість сусідів, які будуть враховуватися для визначення приналежності досліджуваного об'єкта до певного класу. Чим більше число *k*, тим більш рівномірною буде гіперплощина. Слід відмітити, що коли *k* дорівнює кількості об'єктів з навчальної вибірки, результат вироджується в константу.

Крім класичної реалізації з використанням числа сусідів, існують і деякі інші, які здатні оптимізувати процес вибору числа сусідів:

- класичний метод з фіксованим числом сусідів;
- метод зважених найближчих сусідів, суть якого полягає в тому, що чим ближче об'єкт, тим більш істотний внесок він вносить в класифікацію досліджуваного об'єкта;
- метод Парзеновского вікна фіксованої ширини;
- метод Парзеновского вікна змінної ширини;
- метод потенційних функцій.

Важливою особливістю даного методу є те, що він не вимагає навчання. З одного боку, це може здатися плюсом, проте, через відсутність процесу навчання, з'являються проблеми пов'язані з оптимізацією, так як всю навчальну вибірку доводиться зберігати в пам'яті постійно. Крім постійного зберігання в пам'яті всієї навчальної вибірки, також необхідно постійно перераховувати відстані між новим досліджуваним об'єктом і об'єктами з навчальної вибірки, що є трудомістким завданням через можливого обсягу навчальної вибірки [31]. Ще одним недоліком є той факт, що при занадто великій кількості ознак, суми великого числа відхилень з великою ймовірністю мають дуже близькі значення. Це означає, що більшість об'єктів будуть рівновіддалені один від одного і класифікація стане фактично випадковою. Цей недолік вирішується шляхом вибору оптимальної кількості ознак і відсівом найбільш неінформативних ознак.

Основні переваги методу:

- простий в реалізації;
- в середньому показує високу точність класифікації;
- не вимагає навчання.

Основні недоліки:

- чутливий до кількості ознак;
- необхідність в оптимальному виборі числа сусідів;
- великі витрати пам'яті;
- додаткові обчислювальні витрати під час обчислення відстаней і пошуку k-найближчих сусідів;
- чутливий до, так званим, викидам в навчальній вибірці.

2.1.4 Машина опорних векторів

Один з найбільш популярних методів навчання з учителем для задач класифікації. Метод був запропонований в 1963 році. Авторами даного методу є Вапник В.Н. і Червоненкис А.Я .. Спочатку даний метод виконував роль лінійного класифікатора, однак пізніше був запропонований поліпшений варіант даного методу з можливістю будувати нелінійні класифікатори гиперплощини [34].

Якщо об'єкти з навчальної вибірки лінійно нероздільні, то між ними можна провести дві паралельні гіперплощини, які будуть розділяти об'єкти на класи.

Рівняння паралельних гіперплощин:

$$\vec{w} \cdot \vec{x} - b = 1; \vec{w} \cdot \vec{x} - b = -1. \quad (2.6)$$

Відстань між двома паралельними гіперплощинами називається роздільною смугою. Ширина смуги дорівнює $\frac{2}{|w|v}$.

Завданням методу є максимізувати ширину роздільної смуги, що досягається мінімізацією параметра $\| w \|$.

Для того, щоб об'єкти «Не провалювалися» в роздільну смугу, необхідно додати деякі обмеження (рисунок 2.3):

- $\vec{w} \cdot \vec{x} - b \geq 1$, якщо об'єкт належить класу 1;
- $\vec{w} \cdot \vec{x} - b < 1$, якщо об'єкт належить класу 0.

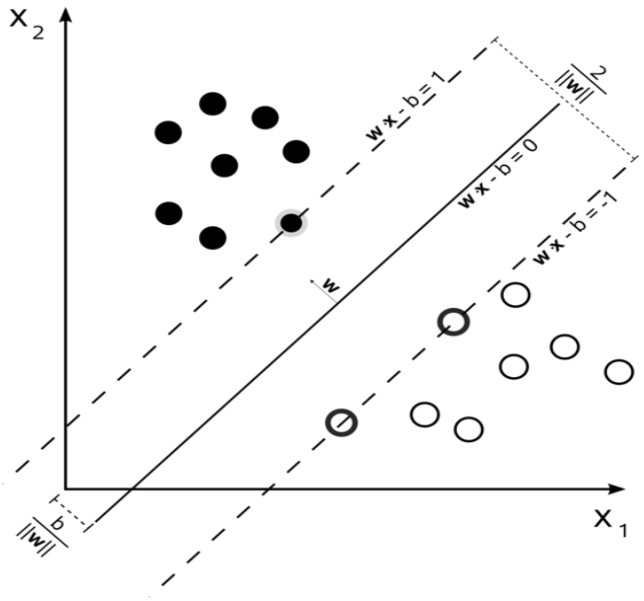


Рисунок 2.3 - Результат роботи методу опорних векторів для поділу об'єктів двох класів

Для лінійно нероздільних даних використовується ядерна функція [35], що дозволяє побудувати нелінійну гіперплощину. Завдання при цьому залишається тим самим: максимізувати відстань $\frac{2}{|w|v}$. Приклад лінійно нероздільних даних представлено на рисунку 2.4.

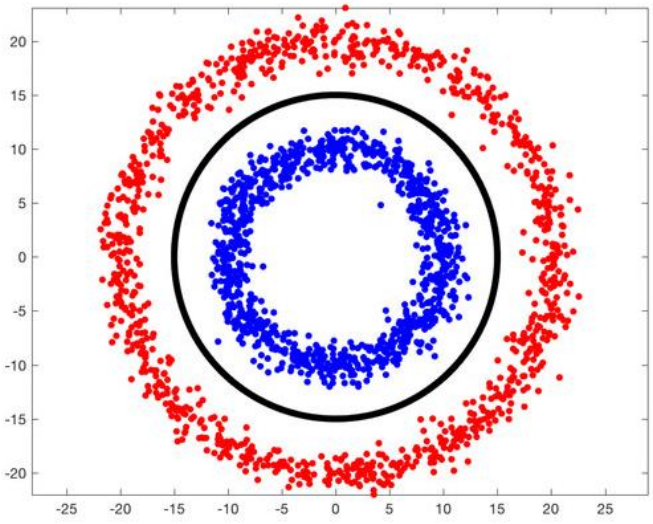


Рисунок 2.4 - Приклад лінійно нероздільних даних

Для того щоб мати можливість класифікувати об'єкти серед більше двох класів, використовуючи даний метод, також, як і для логістичної регресії, необхідно використовувати додаткові методи [37], наприклад, «one-vs-all».

Переваги використання даного методу:

- найбільш швидкий метод знаходження вирішальних функцій;
- метод знаходить розділяючу смугу з максимальною шириною, що дозволяє виробляти більш впевнену класифікацію.

Недоліки:

- метод чутливий до шумових даних і до стандартизації даних;
- немає загального підходу до вибору ядра.

2.2 Нейронні мережі

Нейронна мережа складається з елементарних одиниць - нейронів. Кожен нейрон має кілька входів і один вихід. На рисунку 2.5 представлена модель нейрона, яка була запропонована Маккалоком-Піттсом в 1943 році [35] [36] [37]. На рисунку 2.6 представлена модель біологічного нейрона.

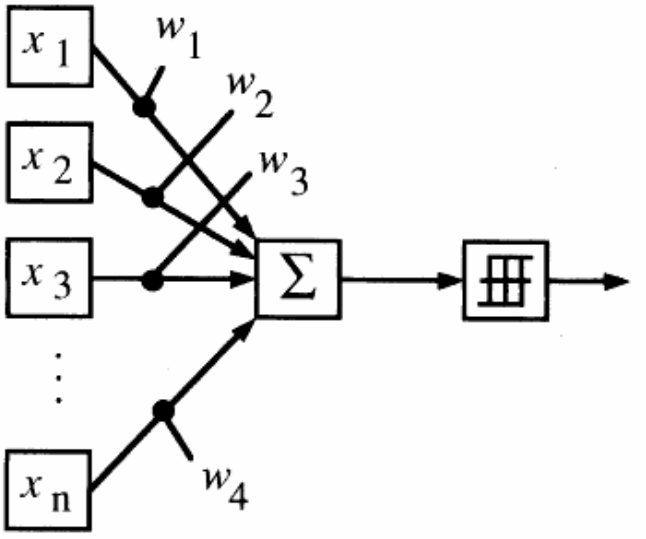


Рисунок 2.5 - Модель штучного нейрона

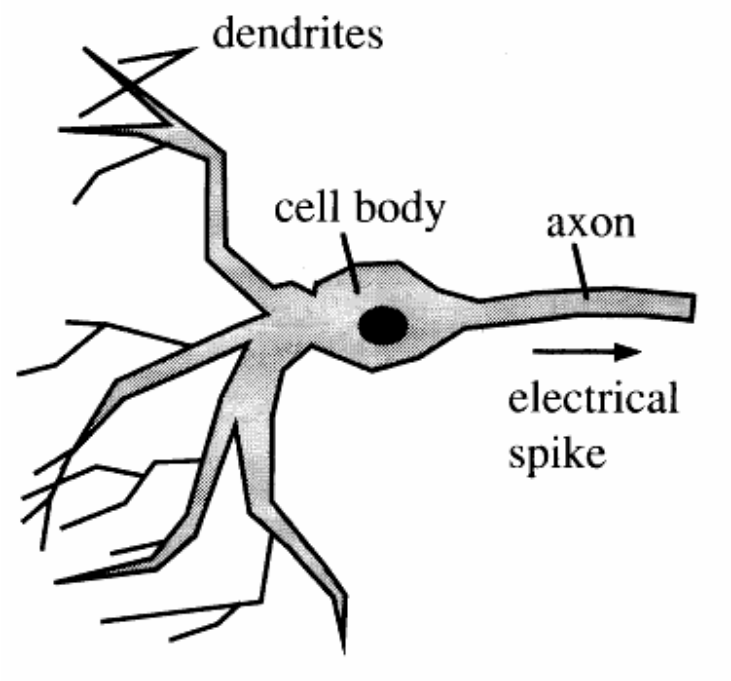


Рисунок 2.6 - Модель біологічного нейрона

Основними параметрами нейрона є ваги і функція активації. Зазвичай ваги представляють у вигляді деякого вектора $\begin{bmatrix} w_1 \\ \vdots \\ w_n \end{bmatrix}$.

Сам по собі принцип роботи спрощеної моделі нейрона простий:

– на вхід подається набір параметрів $[x_1, \ldots, x_n]$;

– параметри підсумовуються з урахуванням представлених ваг: $\sum_{i}^{n} w_i x_i$;

– отримане значення передається в якості аргументу функції активації, яка повертає підсумкове значення $f\left(\sum_{i}^{n} w_i x_i\right)$.

Істотним недоліком моделі Маккалок-Піттса було те, що функція активації була пороговою. Це призводило до того, що якщо сума вхідних значень з урахуванням вагових коефіцієнтів була трохи менше порогового значення, тобто нейрон був не активований, тобто губився сигнал.

Пізніше в 1949 році Дональдом Хеббом був запропонований перший алгоритм навчання і їм же сформульовані правила [36], [38]:

– якщо вихідний сигнал (результат) є невірним і дорівнює нулю, то необхідно збільшити значення синоптичних зв'язків;

– якщо вихідний сигнал (результат) є невірним і дорівнює одиниці, то необхідно зменшити значення синоптичних зв'язків.

Незважаючи на те, що технології пішли далеко вперед, дані правила все ще актуальні і лежать в основі багатьох проектованих нейронних мережах.

Важливою подією в нейромережевих технологіях стала поява персептрона Розенблатта в 1958 році, який складався з елементів трьох типів:

– *S*-елементи - це шар рецепторів;

– *A*-елементи - з'єднані з *S*-елементами. Є суматорами з деякою пороговою функцією. Фактично кожен *A*-елемент являє собою нейрон. При перетині порогового значення, сигнал з *A*-елемента передається на вхід R-елемента з урахуванням вагового коефіцієнта;

– *R*-елементи - це суматори, які підсумовують значення вхідних сигналів з урахуванням ваг і також передають знайдене значення порогової функції. Від результату функції залежить вихідне значення.

Персептрон Розенблатта вже був здатний вирішувати такі завдання, як класифікація, розпізнавання і узагальнення. Також у Розенблатта була описана модель багатошарового персептрона з декількома шарами *A*-елементів.

Для навчання як одношарового, так і багатошарового перспетрона, Розенблатт користувався правилами Хебба. Навчання багатошарового персептрона зажадало деякої модифікації функції активації, а саме використання диференціювання, для того, щоб було можливо використовувати метод градієнтного спуску. Найбільш частим прикладом такої функції є сигмоїда. Важливою особливістю в навчанні персептрона було те, що навчатися могли не всі верстви, а лише деякі з них, в той час як ваги решти шарів були жорстко фіксовані.

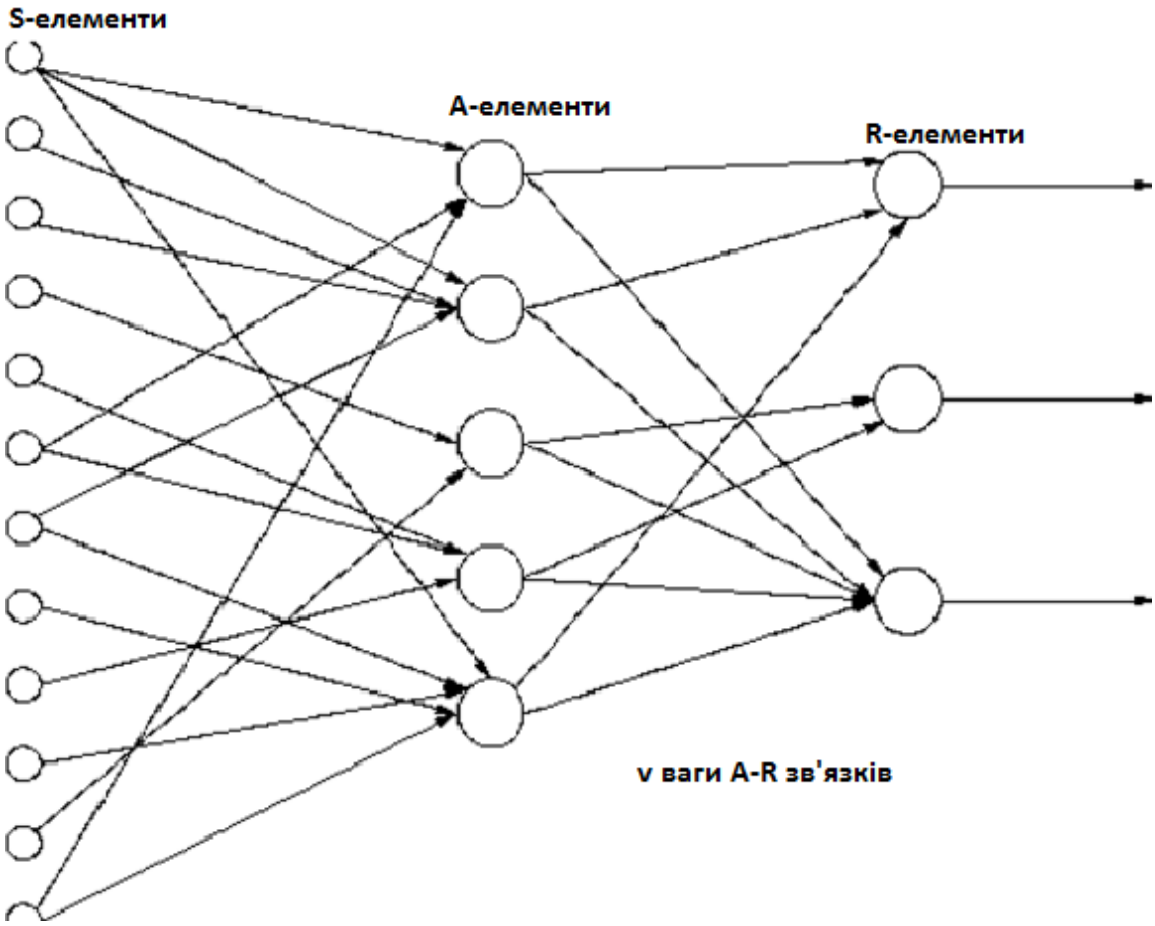


Рисунок 2.7 – Одношаровий персептрон

У 1989 році групою вчених, серед яких були Девід І. Румельхарт, Дж. Е. Хінтон і Рональд Дж. Вільямс, був розроблений метод навчання, який знайшов назву «метод зворотного поширення помилки» [35]. До розробки даного методу

було досить-таки складно налаштовувати ваги нейронних мереж з одними і більш прихованими шарами.

Метод зворотного поширення помилки складається з двох основних частин:

- обчислення поточного результату. На цьому кроці на вхід нейронної мережі подається набір ознак одного об'єкта. Далі відбувається підрахунок вихідних значень поточного шару. Після цього вихідні значення шару L стають вхідними значеннями шару $L + 1$. Так відбувається до досягнення останнього, вихідного шару, виходом якого є рішення. Дана процедура носить назву прямого поширення.
- після отримання результату, відбувається розрахунок деякої помилки $\delta = \text{predicted} - \text{expected}$, де predicted - отримане значення; expected - очікуване. Після цього дана помилка пропускається через всі шари нейронної мережі з урахуванням розрахунку часткових похідних. Також помилка шару L підсумовується з помилкою шару L-1. Даний процес повторюється до вхідного шару, помилка на якому не обчислюється. Після цього відбувається оновлення матриці ваг (ДП.КСМ.111857/16.00.00.000.С2), причому помилки $\delta^{(3)}$ – вихідного шару, $\delta^{(2)}$ - скритого шару, $\left(W^{(2)}\right)^T$ – транспонована матриця прихованого шару до вихідного, $\frac{\partial g(z^{(2)})}{\partial z^{(2)}}$ – похідна функції активації.

Так як в основі методу зворотного поширення помилки використовується метод градієнтного спуску, то є кілька варіацій даного алгоритму:

- матриця ваг оновлюється після кожного примірника з навчальної вибірки;
- матриця ваг оновлюється після проходження всіх примірників з навчальної вибірки;
- матриця ваг оновлюється після проходження якоїсь частини об'єктів з навчальної вибірки.

З появою методу зворотного поширення помилки також з'явилося поняття багатошаровий персептрон Румельхарта. Багатошаровий персептрон Румельхарта

відрізнявся від багатошарового персептрона Розенблатта тим, що, по-перше, навчалися завжди всі верстви, а, по-друге, в якості методу навчання використовувався метод зворотного поширення помилки. Також багатошаровий персептрон Румельхарта є окремим випадком багатошарового персептрона Розенблатта.

Подальший розвиток нейронних мереж і комп'ютерних технологій в цілому призвело до появи багатьох інших моделей, серед яких є згорткові нейронні мережі (рисунок 2.9).

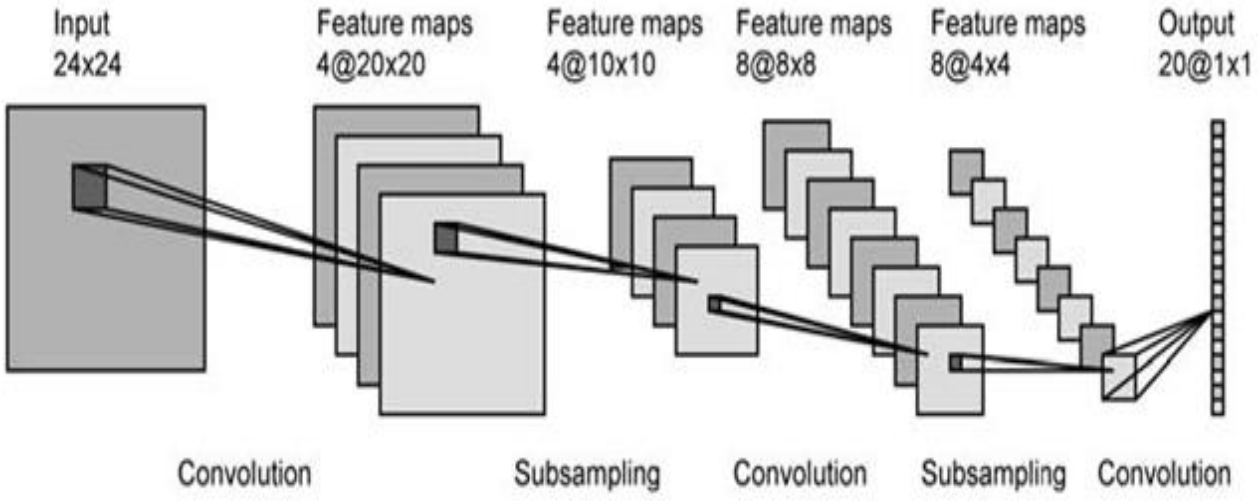


Рисунок 2.9 – Класична структура згорткової нейронної мережі

Ключовою операцією таких мереж є операція згортки, яка дозволяє істотно скоротити кількість операцій і кількість ваг, яке необхідно зберігати в пам'яті. Операція згортки виконується в згортковому шарі. Також для даної операції потрібно ядро, яке являє собою якусь матрицю. Згорткові нейронні мережі привабливі тим, що, завдяки операції згортки зображення, розглядаються не окремі пікселі в якості ознак об'єкта, а деяка група пікселів, яка характеризує собою деяку ознаку об'єкта. Такий підхід також дозволяє домогтися ефекту, через якого мережі неважливо, в якій частині зображення знаходиться та чи інша ознака, а важливо лише його наявність. У задачах класифікації та іншої обробки зображень ця особливість відіграє важливу роль,

Навчання згортальних нейронних мереж здійснюється за допомогою методу зворотного поширення помилки. Для прискорення навчання часто застосовуються додаткові оптимізації. В основному подібні оптимізатори впливають на такий параметр, як крок градієнта, динамічно змінюючи його в ході навчання (рисунок 2.10).

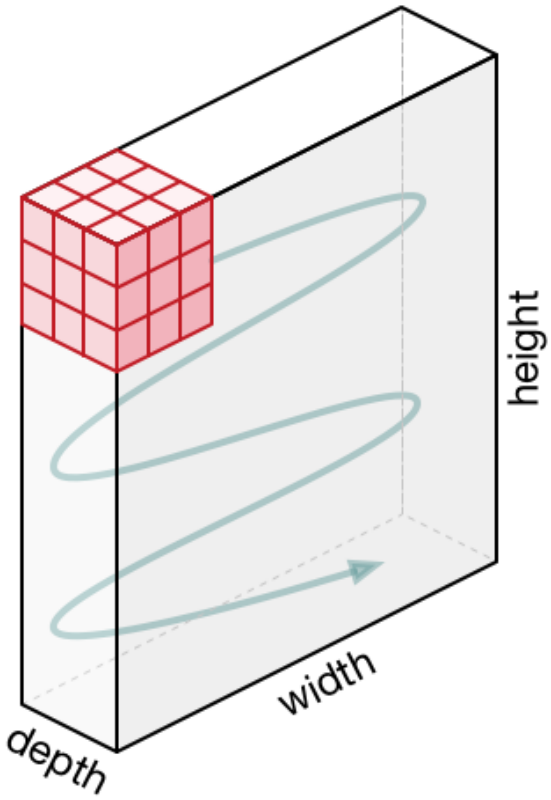


Рисунок 2.10 - Схематичне представлення алгоритму роботи згортки

Як вже раніше згадувалося, ключовим шаром згортальних нейронних мереж є згортковий шар. Кожен такий шар може складатися з різної кількості нейронів. Кількість нейронів в згортковому шарі фактично є кількістю видобутих ознак з вхідного об'єкта на цьому шарі. Також варто відзначити другий чимало важливий шар згорткових мереж - шар субдискретизації. Даний шар виконує важливу операцію ущільнення отриманого набору ознак. Часто в якості операції ущільнення використовується максимальне значення. У деяких випадках перевагу віддають середнього значення або L2-нормуванню (рисунок 2.11).

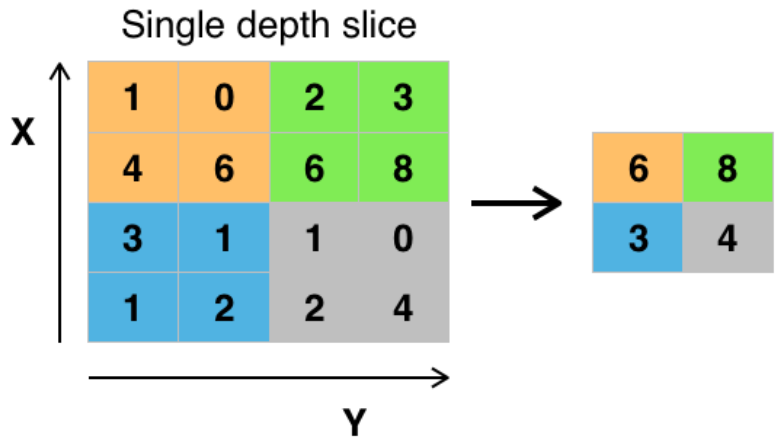


Рисунок 2.11 –Субдискретизація – операція максимуму

Даний шар дозволяє перетворити вихідний набір ознак з попереднього згорткового шару в матрицю менших розмірів. Це дозволяє як економити пам'ять і час на навчання, так і сприяє пошуку ще більш складних ознак на згортковому шарі, що знаходиться після шару субдискретизації.

2.3 Обгрунтування вибору методу машинного навчання

В даному розділі було розглянуто чотири методи машинного навчання:

- логістична регресія;
- метод найближчих сусідів;
- машина опорних векторів;
- нейронна мережа.

Крім розглянутих методів існують також і інші методи, які також використовуються в задачах класифікації, але з урахуванням того, що необхідно розпізнавати зображення, вони не були включені до цього переліку.

Кожен з розглянутих методів має свої переваги і недоліки. З урахуванням переваг і недоліків кожного, необхідно вибрати найбільш оптимальний метод для вирішення задачі розпізнавання зображень символів.

Порівняння методів буде здійснюватися на випадково згенерованих зразках. Генеруватися будуть лінійно нероздільні вибірки, так як завдання класифікації символів має на увазі класифікацію лінійно нероздільних об'єктів (рисунок 2.12).

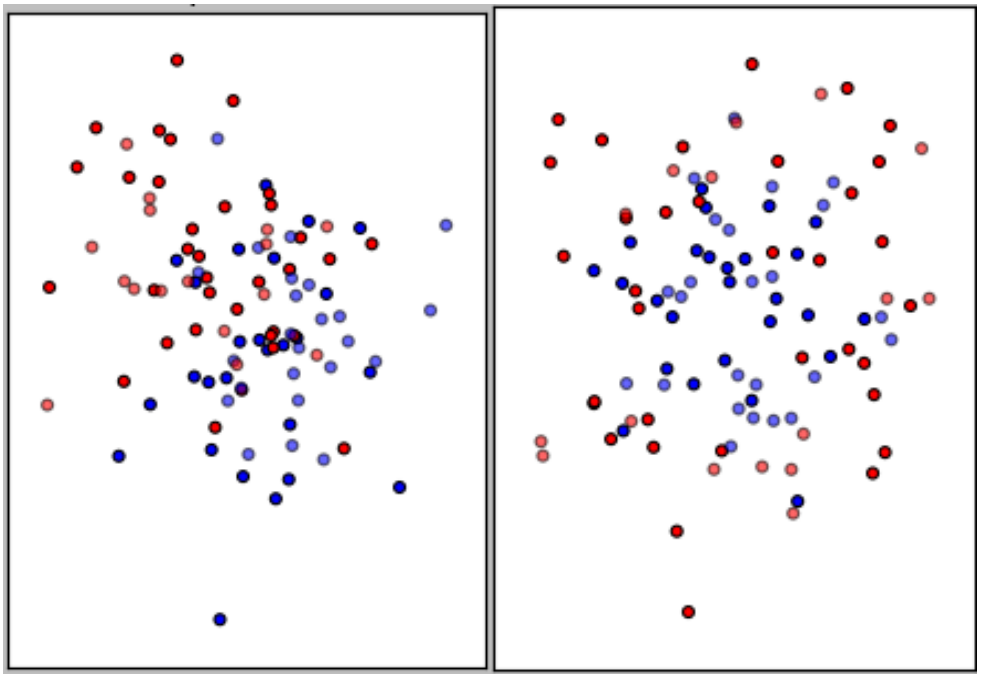


Рисунок 2.12- Згенеровані безліч об'єктів. Зліва-направо: лінійно роздільні об'єкти з безліччю шумових об'єктів; лінійно нероздільні об'єкти

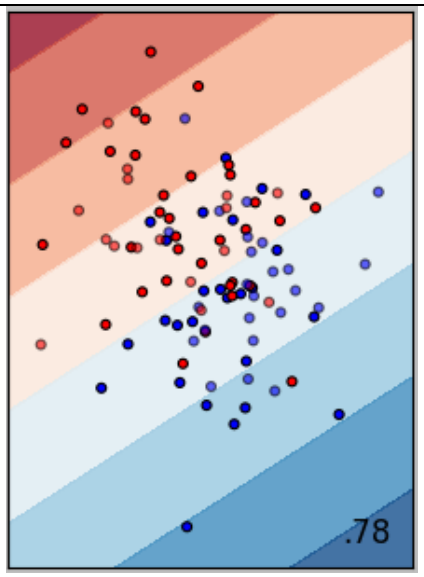
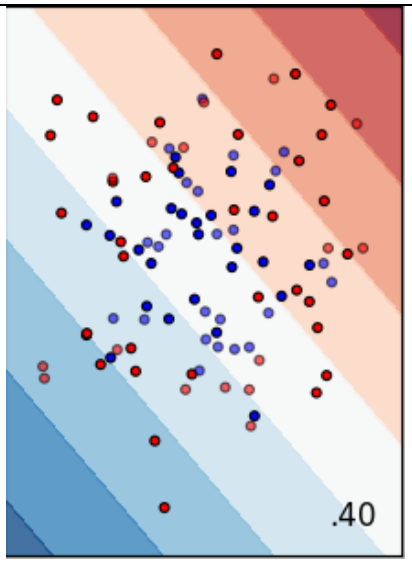
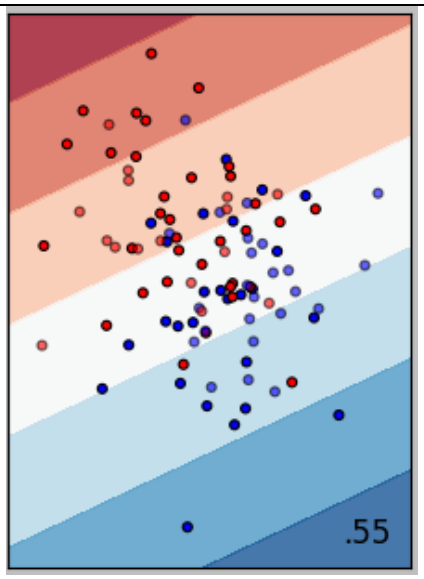
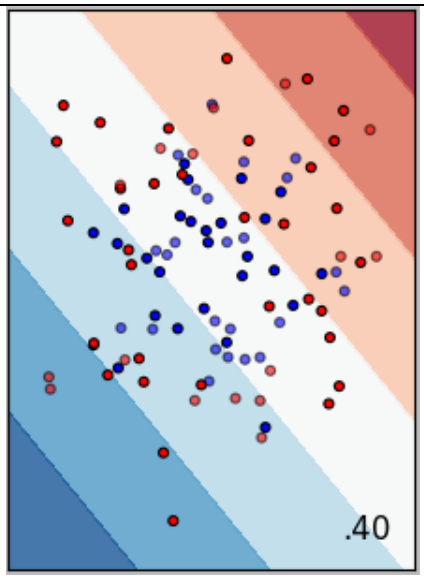
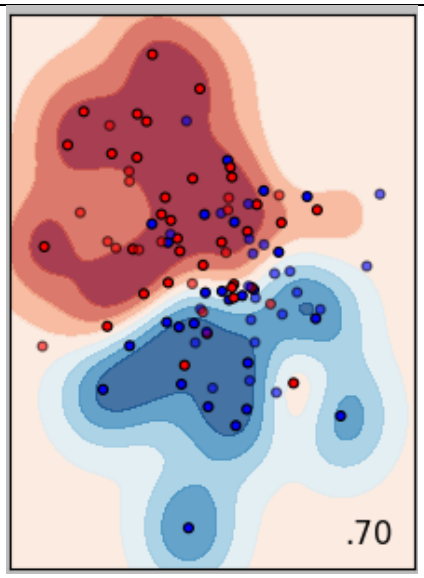
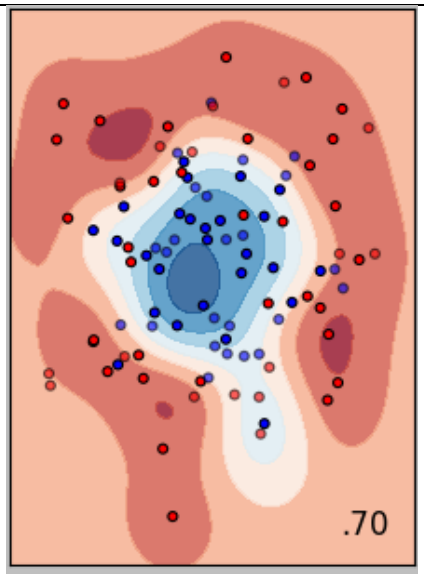
Обидва згенерованих безлічі об’єктів є лінійно нероздільними з деякою кількістю шумових об'єктів.

Тестувалися такі методи:

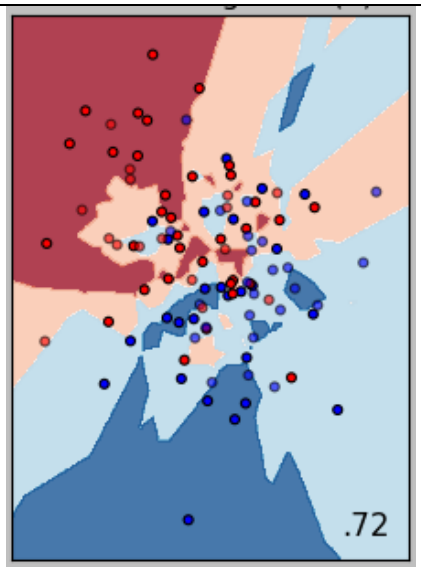
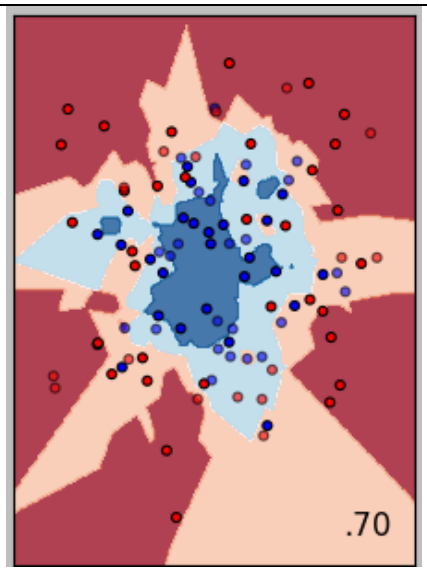
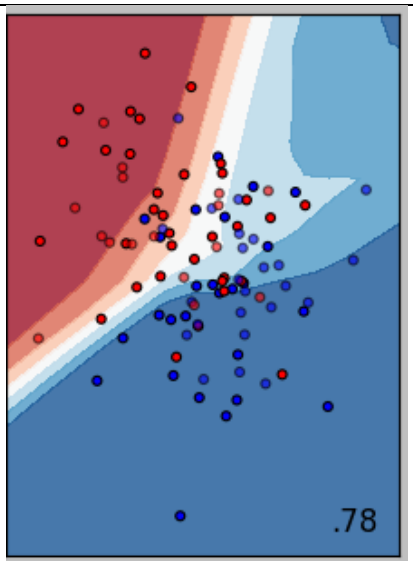
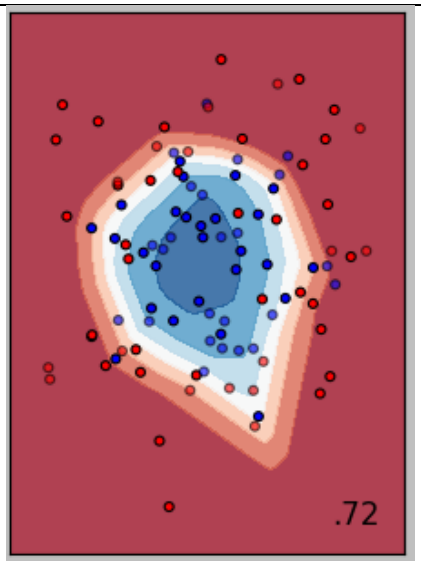
- логістична регресія;
- класичний метод опорних векторів;
- машина опорних векторів з радіальним ядром;
- машина найближчих сусідів з числом сусідів рівним трьом;
- багатошаровий персептрон.

Результати тестування наведені в таблиці 2.1.

Таблиця 2.1 - Таблиця з результатами роботи різних методів класифікації

1	2	3
Логістична регресія		
Машина опорних векторів		
Машина опорних векторів з радіальним ядром		

Продовження таблиці 2.1

1	2	3
Метод найближчих сусідів		
багатошаровий персептрон		

В результаті тестування були зроблені наступні висновки:

– логістична регресія і метод опорних векторів з лінійним ядром погано підходять для вирішення завдань класифікації лінійно нероздільних об'єктів;

– метод найближчих сусідів в результаті тестування показав непогану точність класифікації. Однак залишається проблема вибору числа сусідів особливо з урахуванням того, що в розв'язуваної задачі не два класи, а 62. Також було виявлено сильний негативний вплив шумових об'єктів на кінцевий результат роботи алгоритму;

– машина опорних векторів з радіальним ядром і багатошаровий персептрон також показали високу точність класифікації, але багатошаровий персептрон показав точність класифікації в середньому більше.

Відмінності в точності класифікації в таких методах, як машина опорних векторів, метод найближчих сусідів і багатошаровий персептрон невеликі. Кількість допустимих помилок залежить від розв'язуваної задачі, тому в деяких завданнях можна пожертвувати точністю розпізнавання заради збільшення швидкості роботи або навчання алгоритму. Однак в разі завдання розпізнавання символів точність розпізнавання має найвищий пріоритет, тому навіть втрата пари відсотків точності може сильно позначитися на підсумковому результаті.

Після проведення даного тестування було прийнято рішення використовувати нейронну мережу в якості методу машинного навчання для вирішення задачі розпізнавання символів.

2.3 Дослідження технологій оптимізації процесу навчання

Процес навчання моделі пов'язаний з певними часовими витратами. Основною складністю процесу навчання є велика кількість матричних операцій, а саме множення матриць. Ще однією складністю є велика кількість навчальних зразків. В сумі, два цих чинника є серйозною проблемою для процесорної обробки.

Для вирішення цієї проблеми були розроблені спеціальні оптимізують методи, які дозволяли навчити модель швидше, проте незабаром і цього стало не вистачати. У підсумку, були розроблені методи, що дозволяють використовувати потужності графічного процесора [39].

Графічні процесори дозволяють прискорити процес навчання за рахунок того, що використовується архітектура SIMD (рисунок 2.13) [40]. Дана архітектура дозволяє забезпечити паралелізм на рівні даних, так як має на увазі

один потік команд і кілька потоків даних. При використанні такої архітектури, можна домогтися істотного прискорення обчислювального процесу в задачах, в яких необхідно проводити однакові обчислювальні операції над великим об'ємом даних. Машинне навчання якраз і є одним з таких типів завдань.

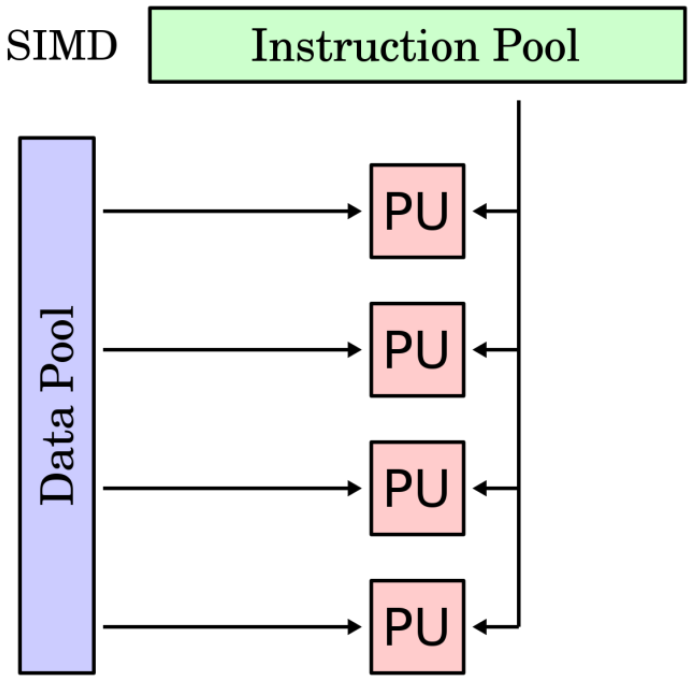


Рисунок 2.13 - Схематичне представлення SIMD архітектури

Однією з технологій, що підтримує дані методи організації обчислювального процесу, є технологія CUDA [41] від компанії Nvidia. У відкритих Nvidia одним з ключових параметрів продуктивності в паралельних обчисленнях є кількість ядер CUDA. Чим більше ядер, тим, відповідно, швидше будуть проводитися обчислення.

На рисунку 2.14 представлена схема виконання завдань з використанням технології CUDA. В даному випадку Host - це центральний процесор, який посилає команди графічного процесора; Kernel - завдання, що їх посилають центральним процесором; Device - графічний процесор, який виконує команди; Grid - мережа з певними розмірами, яка групує блоки; Block - компонент, що

характеризує верхній рівень ядра графічного процесора. Кожен блок може містити в собі певну кількість паралельних потоків.

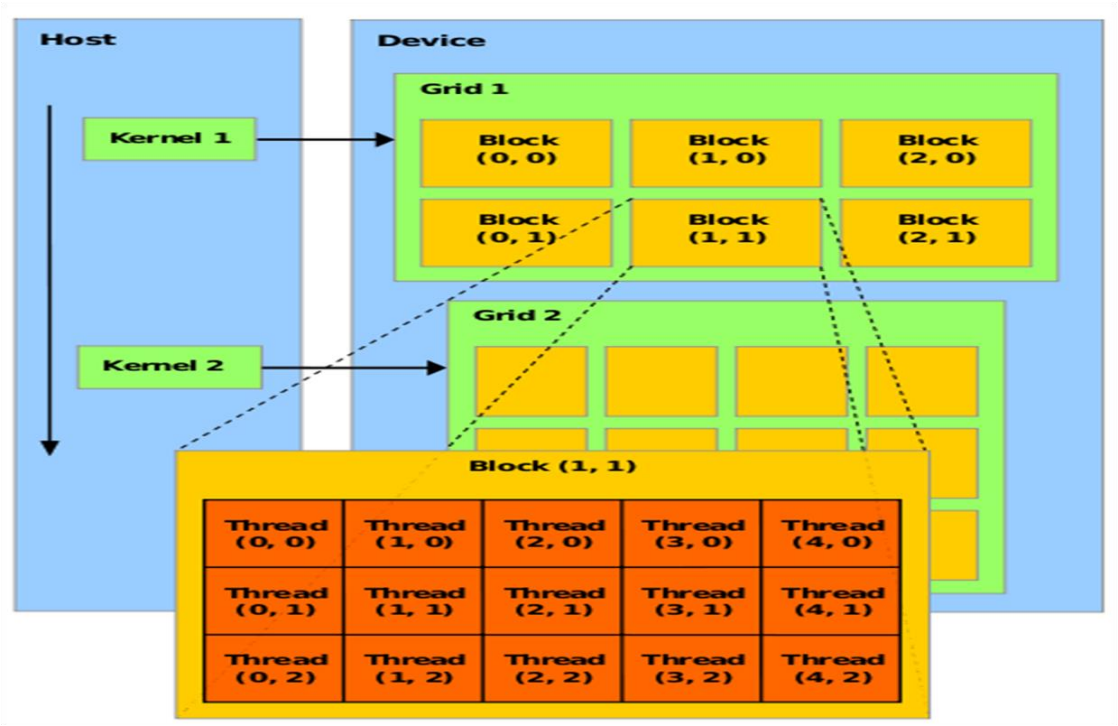


Рисунок 2.14 – Схема розподілу завдань, розбиття їх на окремі блоки і виділення певної кількості потоків

В даному розділі були проаналізовані методи обробки зображень, існуючі методи машинного навчання і технології оптимізації навчання.

Аналіз методів обробки зображень показав, що необхідно вирішити такі недоліки як невпорядкованість знайдених контурів і виділення деяких пар символів як одного символу.

Аналіз методів машинного навчання показав, що найбільш оптимальним підходом є використання нейронної мережі для класифікації лінійно нероздільних об'єктів. Як архітектури нейронної мережі була обрана сверточное нейронна мережа. Однак основною проблемою, пов'язане з реалізацією нейронної мережі, є те, що необхідно не тільки спроектувати мережу, а й оптимально вибрати набір гіперпараметров, від яких значною мірою залежить кінцевий результат.

При аналізі методів оптимізації машинного навчання було виявлено, що для прискорення процесу навчання можна скористатися наявними можливостями графічного процесора.

3 РОЗРОБКА МЕТОДІВ І АЛГОРИТМІВ РОЗПІЗНАВАННЯ ТЕКСТУ З ДОПОМОГУ НЕЙРОННОЇ МЕРЕЖІ

3.1 Розробка алгоритму обробки зображень

Необхідність в розробці нового алгоритму обробки зображень полягає в тому, що проаналізований раніше алгоритм містить в собі два суттєвих недоліки:

- некоректне виділення символів з дефектами;
- некоректне виділення деяких пар символів;
- невпорядкованість виявлених контурів.

Покращена версія вихідного алгоритму повинна усунути ці недоліки.

3.1.1 Алгоритм обробки зображень

Проблема з некоректним виділенням деяких пар символів пов'язана з тим, що розглянуті пари символів перетинаються в деякій точці і алгоритм пошуку контурів розцінює їх як один символ. До такої помилку може привести навіть незначне перетин одного символу з іншим всього в парі точках (рисунок 3.1).

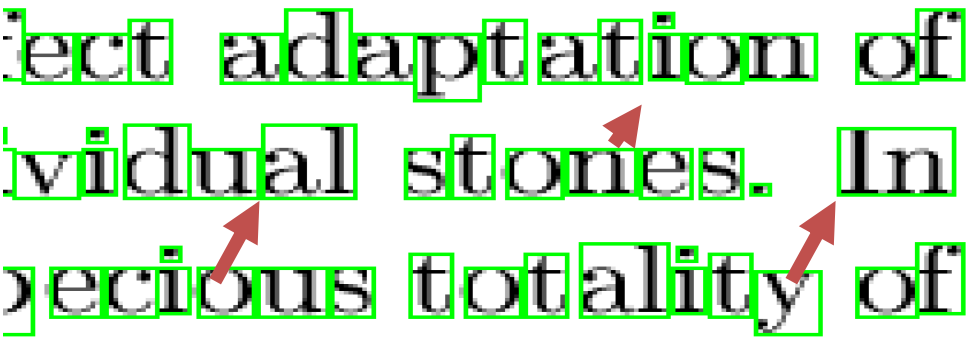


Рисунок 3.1 – Результат роботи вихідного методу обробки зображень. Деякі пари символів виділені як єдиний символ

Як видно з рисунка 3.1, такі пари символів як, наприклад, «In», «al» були розпізнані як єдиний символ. Причиною цього є те, що вони перетинаються в деяких точках і обраний алгоритм пошуку контурів визначає їх як єдиний символ.

В якості одного з рішень проблеми може послужити зменшення товщини контурів. Для цього можна скористатися однією з морфологічних операцій, а саме операцією ерозії [42]. Дану операцію застосовують до бінарних зображень. Основним параметром даної операції є ядро, яке задається матрицею. Потім дане ядро застосовують послідовно до всього зображення. Якщо уявити, що всі пікселі зображення мають в якості свого значення або 0, або 1, то, після проходження ядра, піксель залишить значення 1 в тому випадку, якщо інші пікселі, які покриває ядро, також мають значення 1. Якщо хоча б один піксель в якості свого значення має 0, то і розглянутий піксель прийме 0. Результат роботи «ерозії» показано на рисунку 3.2.

Рисунок 3.2 – Результат роботи методу «ерозії»

Операція ерозії також добре справляється з видаленням шумів. Істотним недоліком даної операції є те, що іноді в результаті виходять проміжки не тільки між символами, а й в самих символах.

Більш «м'яким» варіантом ерозії є операція розмикання [43]. Використовується ядро яке має наступний вигляд: $\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$. Після додавання операції розмикання з описаним ядром, результат вийшов таким (рисунок 3.3):

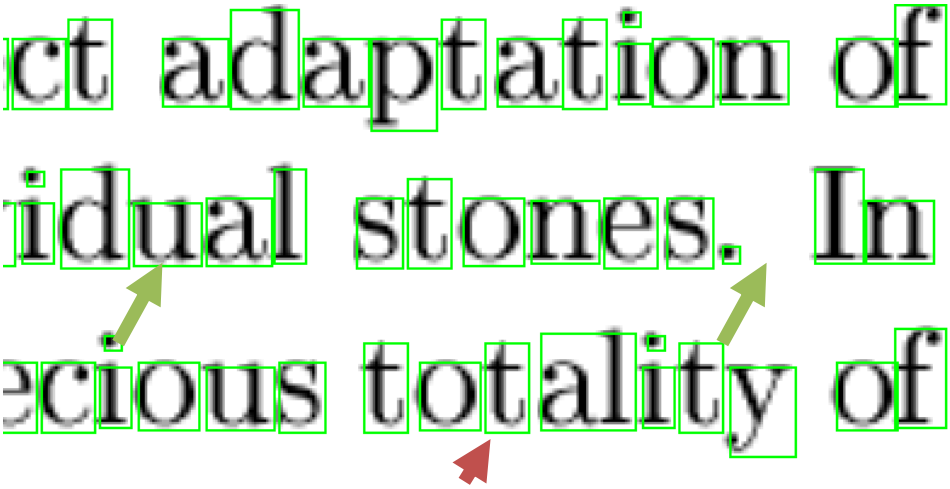


Рисунок 3.3 – Результат роботи поліпшеного методу обробки зображень

Дана операція дозволила «роз'єднати» деякі пари символів, що збільшило загальний відсоток правильно виділених символів.

Крім помилки, пов'язаної з пересічними символами, також є помилка, пов'язана з дефектами символів (рисунок 3.4):

Рисунок 3.4 – Приклад символу з дефектом друку, який привів до помилки виділення

Як видно з рисунка 3.4, літера «w» має невеликий розрив в центральній області, що призводить до виділення даного символу, як двох символів. Рішенням цієї проблеми може послужити операція дилатації (рисунок 3.5) [44].

Рисунок 3.5 – Приклад роботи методу «дилатації»

Недоліком даного методу є те, що він збільшує контури надто сильно. Збільшення контурів призводить до того, що, по-перше, це прибирає дію операції розмикання, по-друге, з'єднує ще більше символів, що призведе до появи попередньої помилки.

Як і для операції ерозії, так і для операції дилатації існує більш «м'який» її варіант. Цей варіант називається операцією замикання (рисунок 3.6) [43].

Рисунок 3.6 – Приклад роботи методу «замикання»

Після застосування операції замикання, підсумкова кількість коректно виділених символів зросла і використовується ядро яке має такий вигляд: [1 1] (рисунок 3.7).

Рисунок 3.7 – Результат роботи поліпшеного методу обробки зображень

Також для збільшення загального відсотка правильно виділених символів була застосована операція трансформації для збільшення зображення в 2 рази.

Для вирішення проблеми невпорядкованого списку виділених символів необхідна спеціальне сортування. Методи сортування, які будуть сортувати контури лише за координатами x і y не вирішують цю проблему, так як символи, що мають однакові координати x або y будуть виводитися поруч незалежно від того, чи знаходяться вони на одному рядку чи ні. У підсумку, перед сортуванням символів кожному символу присвоювався певний номер рядка, який збільшується, якщо різниця в y координатах між поточним символом і попереднім була більше порогової. Порогові величини міжрядкового інтервалу були обрані як

максимальна висота серед усіх знайдених символів. Підсумковий алгоритм обробки зображень наведено на рисунку 3.8.

Рисунок 3.8 – Підсумковий алгоритм обробки зображень. Синім кольором виділені вихідні методи; Зеленим - додані методи для обробки зображень

3.1.2 Тестування запропонованих алгоритмів

Для оцінки розробленого алгоритму необхідно провести порівняльний аналіз обох алгоритмів.

Перший і найголовніший тест, який був пройдений обома алгоритмами, це тест на загальну точність виділення символів (рисунок 3.9).

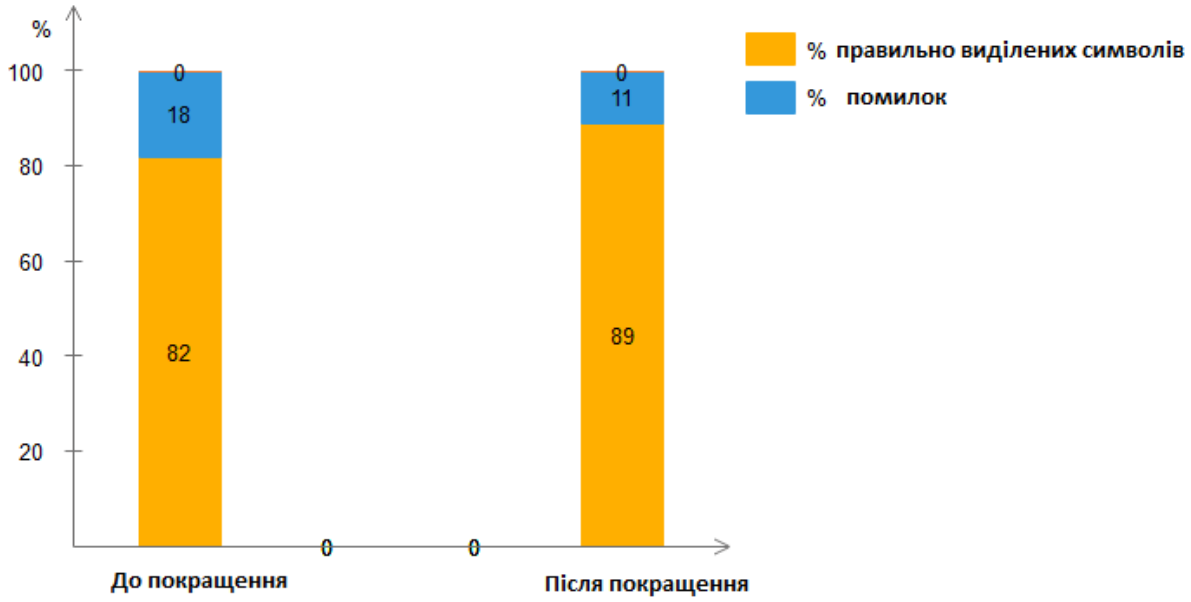


Рисунок 3.9 – Результати порівняння вихідного алгоритму і поліпшеного

Як видно з результатів, розроблений алгоритм має на 7% більше правильно виділених символів, ніж початковий алгоритм. Однак тести також показали, що деякі символи все також виділяються некоректно. В основному це пари символів, які знаходяться дуже близько один до одного і на яких слабо подіяв метод розмикання.

Також були проведені тести на швидкість обробки. Було виконано 100 ітерацій як першого алгоритму, так і другого. Тести проводилися без урахування операції читання і запису зображень. Результати представлені на рисунку 3.10.

Параметр	Значення
Початковий алгоритм	3,64 с
Покращений алгоритм	12,7 с

Рисунок 3.10 – Результати проведених тестів на швидкість обробки

Результати тестів швидкості виконання алгоритму показали, що розроблений алгоритм вимагає майже в 3 рази більше часу, ніж початковий, але це компенсується підвищеною точністю виділення.

В ході розробки алгоритму обробки зображень вдалося досягти наступних результатів:

- вирішено проблему впорядкованості виділених символів;
- практично повністю усунена проблема з символами, що знаходяться дуже близько один до одного;
- вирішено проблему з символами, що мають деякі дефекти.

Загальна кількість правильно виділених символів зросла на 7%.

Основний недолік розробленого методу полягає в довшому виконанні в порівнянні з початковим алгоритмом.

Вихідний код розробленого алгоритму представлений в додатку А.

3.2 Проектування нейронної мережі

Проаналізувавши методи машинного навчання, було прийнято рішення використовувати нейронні мережі для розпізнавання тексту. Також проаналізувавши наукові роботи в області розпізнавання і класифікації з використанням нейронних мереж, був зроблений висновок про те, що найкращим чином для даної задачі підходять згорткові нейронний мережі.

В якості початкової топології нейронної мережі була обрана топологія, представлена на рисунку 3.11.

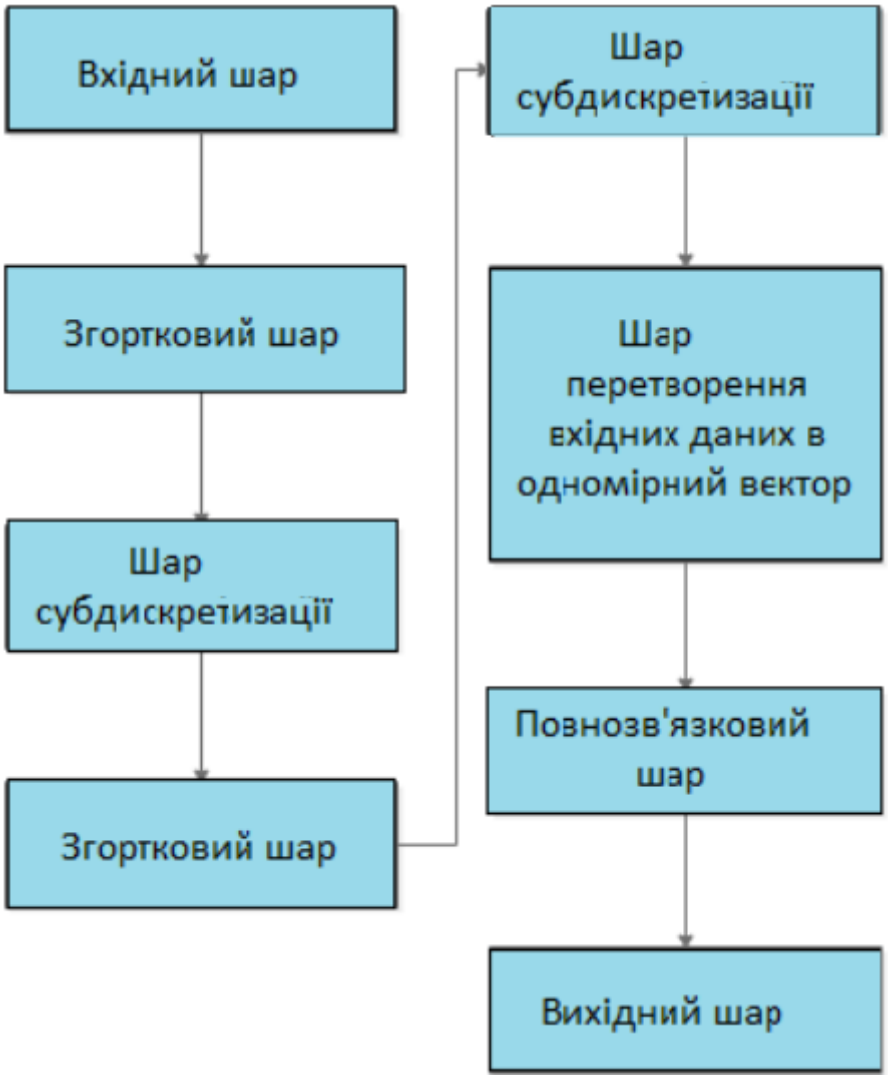


Рисунок 3.11 – Початкова топологія нейронної мережі

3.2.1 Підготовка навчального набору даних

Для навчання нейронної мережі необхідний набір навчальних даних. В якості початкового набору навчальних даних виступала вибірка, що складається з цифр і букв англійського алфавіту в обох регістрах. Всього представлено 62 класи. Кожен клас складається з 1024 об'єктів. Всього представлено 63488 об'єктів. Кожен об'єкт являє собою зображення розмірами 128 * 128 пікселів.

Першим кроком при підготовці навчального набору даних була фільтрація навчального набору від шумових символів. Приклад шумових даних представлений на рисунку 3.12.

Рисунок 3.12 – Об'єкти буква «Н» в верхньому регістрі

Після фільтрації в навчальній вибірці залишилося 31496 об'єктів.

Другим кроком була відцентровка вихідних символів і видалення неінформативних областей у вигляді фону. Отриманий результат представлений на рисунку 3.13.

Рисунок 3.13 – Результат центрування символу навчальної вибірки і видалення зайвих країв

Третім кроком було зменшення розміру вихідних зображень до 28 * 28 пікселів. Даний крок був необхідний як для прискорення процесу навчання, так і для зменшення витрат на зберігання отриманої інформації.

У підсумку, після пророблених кроків навчальна вибірка прийняла такий вигляд:

- розмір одного зображення: 28 * 28 пікселів;
- об'єктів кожного класу: 508;
- всі символи знаходяться строго по центру зображення.

3.2.2 Підбір оптимального набору гіперпараметрів

Початковий набір гіперпараметрів представлений на рисунку 3.14.

Параметр	Значення
Розмір ядра згортку	5
Розмір ядра субдескрітізації	2
Кількість нейронів в повнозв'язковому шарі	512
Кількість об'єктів,які беруть участь в градієнтоному спуску за одну зміну ваги	32
Початкове значення ваги	0
Кількість епох	50
Кількість нейронів в першому згортковому шарі	32
Кількість нейронів в другому згортковому шарі	64

Рисунок 3.14 – Вихідний набір гіперпараметров нейронної мережі

Методів, які могли б найбільш оптимально підібрати набір гіперпараметрів, не існує, тому всі гіперпараметри підбиралися експериментально.

Було проведено кілька експериментів. Результати експериментів представлені на рисунку 3.15.

Емпіричним шляхом були виявлені найбільш значущі гіперпараметри, а саме:

- розмір ядра згортки;
- кількість нейронів в повнозв'язному шарі;
- кількість об'єктів, що беруть участь в градієнтному спуску за один прохід.

Розмір ядра згортки відповідає за вилучення ознак із зображень. Після декількох експериментів було прийнято рішення використовувати ядро розміром 3 * 3.

Параметр	Набір №1	Набір №2	Набір №3	Набір №4
Розмір ядра згортку	5	5	3	3
Розмір ядра субдіскретізації	2	2	2	2
Кількість нейронів в повнозв'язкоому шарі	512	512	512	1024
Кількість об'єктів,які беруть участь в градієнтоному спуску за одну зміну ваги	32	256	512	1024
Початкове значення ваги	-e < random < e	-e < random < e	-e < random < e	-e < random < e
Кількість епох	50	50	50	50
Кількість нейронів в першому згортковому шарі	32	32	32	32
Кількість нейронів в другому згортковому шарі	64	64	64	64
Точність розпізнавання,%	77	79	83	87

Рисунок 3.15 – Результати тестування різних наборів гіперпараметрів

Кількість нейронів в повнозв'язному шарі також в якомусь сенсі характеризує додатковий набір ознак досліджуваного об'єкта. Можна було б припустити, що чим більше нейронів в цьому шарі, тим краще, однак це не так. При занадто великій кількості нейронів, модель не зможе узагальнити навчальну вибірку і фактично запам'ятає її. Цей ефект називається перенавчанням. Найбільш помітною ознакою перенавчання є висока точність класифікації на навчальній вибірці і низька точність класифікації на тестовій вибірці.

Третій параметр також є важливим, так як впливає на збіжність алгоритму.

У підсумку, з таким набором гіперпараметрів і на основі вихідної топології була отримана точність класифікації рівна 87%, що є непоганим показником, але в той же час недостатнім для розв'язуваної задачі.

3.2.3 Топологія нейронної мережі

Для подальшого підвищення якості розпізнавання були зроблені спроби змінити вихідну топологію нейронної мережі.

В ході експериментів були зроблені спроби додати додаткові пари шарів згортки і субдискретизацію, але це не дало бажаного результату, а в деяких випадках навіть істотно знизило точність розпізнавання і збільшило час навчання через повільну збіжність алгоритму (рисунок 3.16).

В одному з експериментів була зроблена спроба додати ще трохи повнозв'язних шарів зі збільшеним числом нейронів. На спеціально підготовленій тестовій вибірці об'єктів точність розпізнавання нейронної мережі зі зміненою топологією склала 89%.

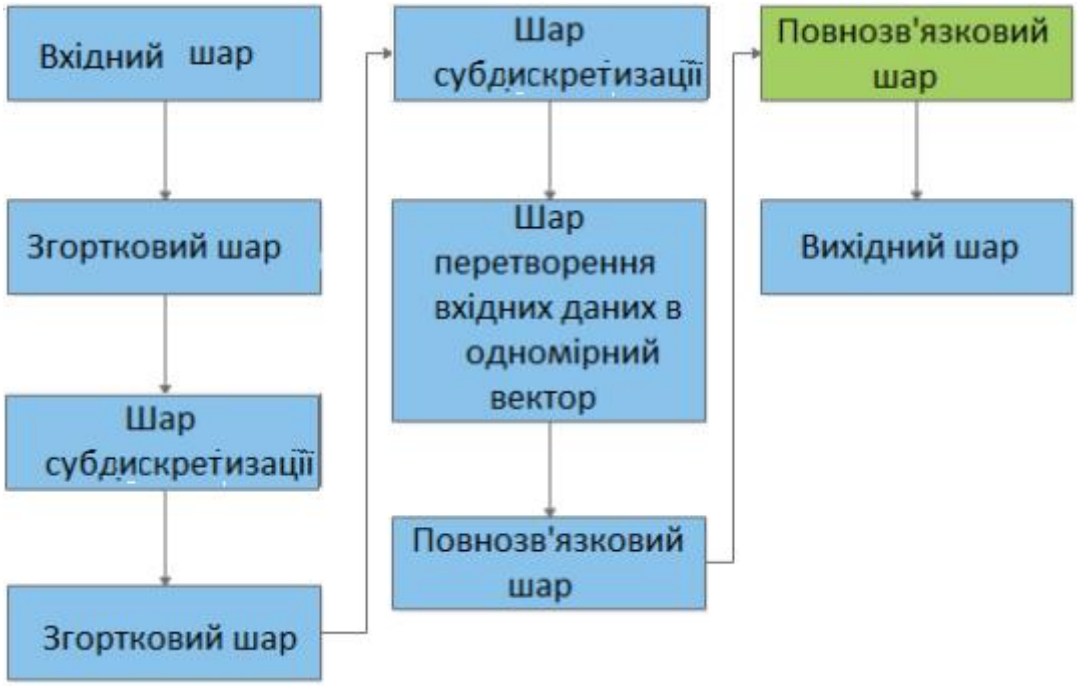


Рисунок 3.16 – Покращена топологія нейронної мережі. Червоним позначено новий шар

Кількість прихованих нейронів в новому повнозв'язному шарі становить 2048. Дане число нейронів було отримано емпіричним шляхом.

Варто відзначити той факт, що з ускладненням нейронної мережі також зростає ймовірність її перенавчання. Для усунення ефекту перенавчання були додані L1 і L2 регуляризації.

3.2.4 Тестування спроектованої нейронної мережі

Тестування спроектованої нейронної мережі вироблялося на заздалегідь підготовленому зображенні. Вхідне зображення являє собою зразок ідеального вхідного зображення, так як не містить в собі будь-яких дефектів в написанні символів, поворотів самого зображення або окремих його символів і інших видозмін (рисунок 3.17).

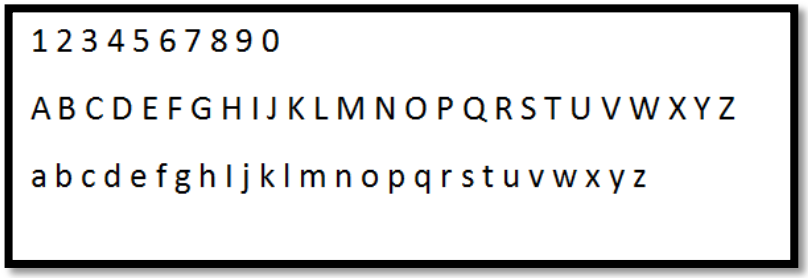


Рисунок 3.17 – Приклад вхідного зображення

В результаті було отримано наступний результат (рисунок 3.18):

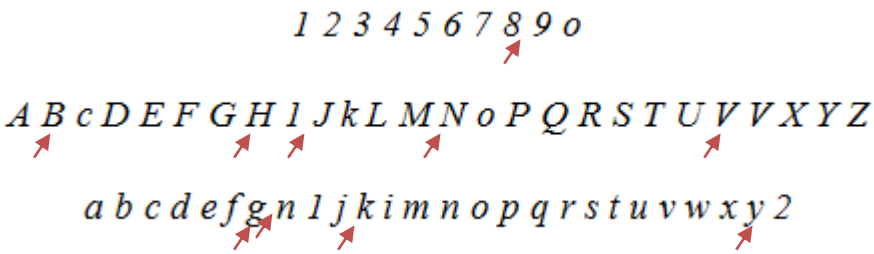


Рисунок 3.18 – Результат роботи алгоритму розпізнавання. Стрілки, що показують невірно розпізнані символи

Як видно з результату нейронна мережа помиляється на схожих об'єктах, через які з'являються описані раніше неоднозначності при розпізнаванні. Якщо помилки при розпізнаванні деяких символів можна не враховувати, так як навіть людина не завжди може побачити різницю між цифрою «1» і буквою «l», то помилки «2-z», «nh» і «il» вже істотніше.

Також мережа була протестована на зображенні, яке було з деяким кутом повороту (рисунок 3.19).

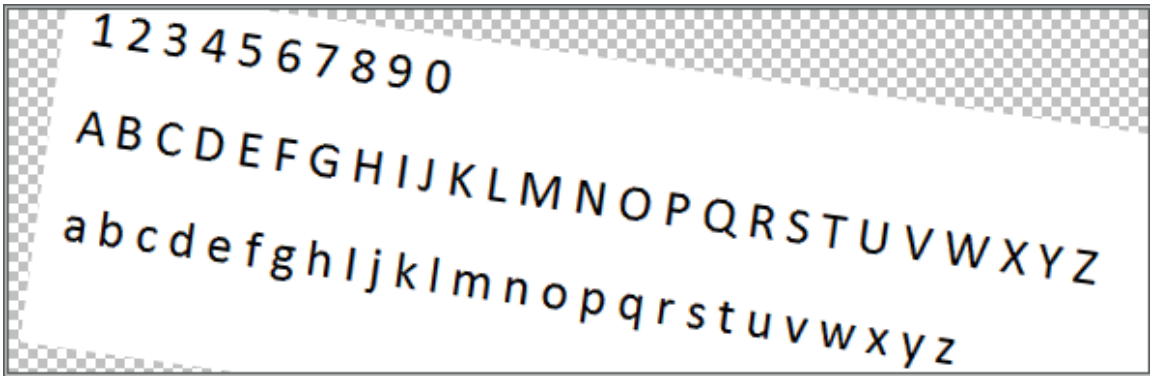


Рисунок 3.19 – Приклад вхідного зображення з поворотом

В результаті тестування мережі на такому зображенні були отримані результати з дуже низьким відсотком правильно розпізнаних символів. При детальному аналізі виявлено проблеми, тобто основним джерелом низької точності розпізнавання було виділення символів модулем обробки зображень, яке не враховувало повороту. Іншим джерелом проблеми була сама мережа, яка має проблеми при розпізнаванні символів з поворотом більше порогового значення.

В ході проектування нейронної мережі були зроблені наступні зміни вихідного набору параметрів мережі:

- емпіричним шляхом були отримані оптимальні набори гіперпараметрів для даного завдання;
- топологія нейронної мережі також зазнала деяких змін, а саме додавання додаткового повнозв'язного шару;
- експериментальним шляхом було знайдено оптимальне число прихованих нейронів в новому шарі.

В ході тестування спроектованої нейронної мережі разом з розробленим раніше модулем обробки зображень були виявлені такі недоліки:

- модуль обробки зображень показує низькі результати, якщо вхідне зображення сильно повернуте;
- при сильному повороті зображення також порушується порядок символів;
- із-за повороту вхідного зображення і наявності в ньому схожих пар символів нейронна мережа частіше помиляється при класифікації і, таким чином, утворюються неоднозначності.

Вихідний код моделі нейронної мережі представлений в додатку Б. Код підготовки навчального набору даних представлений в додатку С.

3.3 Розробка допоміжних методів підвищення точності розпізнавання

У попередньому розділі при тестуванні розробленої нейронної мережі і модуля обробки зображень були виявлені такі недоліки, як:

- чутливість до поворотів зображення;
- порушення порядку символів при поворотах зображення;
- помилки при розпізнаванні схожих символів і деяких інших.

Для усунення перерахованих недоліків необхідно вдосконалити розроблені модулі.

3.3.1 Детектування і усунення повороту зображення

Перше, що необхідно усунути у вхідного зображення, це поворот. Усунення повороту зображення виконується в кілька кроків:

- знайти об'єкт на зображенні, який залишився після порогової обробки;

- застосувати операцію виділення знайденого об'єкта прямокутником мінімального розміру;
- знайти кут отриманого прямокутника з попередньої операції і, з огляду на кут повороту прямокутника, знайти кут, на який потрібно повернути зображення для усунення повороту;
- застосувати афінне перетворення [45].

Пошук об'єкта на зображенні не складає труднощів, і застосувати можна будь-яку зручну операцію пошуку. Після знаходження об'єкта його необхідно виділити прямокутником з мінімальною площею (рисунок 3.20).

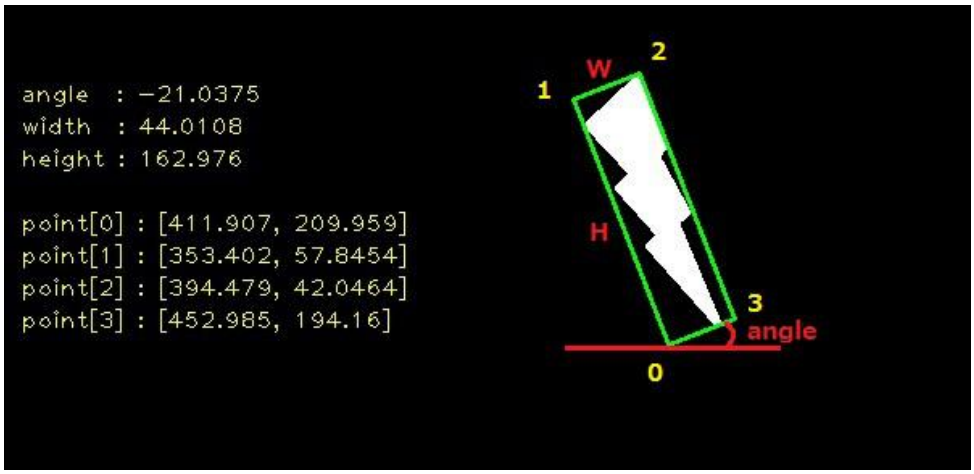


Рисунок 3.20 – Результат виділення об'єкта найменшим прямокутником

Після того, як знайдений кут повороту об'єкта і кут, на який необхідно повернути все зображення для усунення ефекту повороту, тобто застосувати афінне перетворення. Афінне перетворення - це перетворення виду $f : R^n \to R^n$. Функція перетворення виглядає наступним чином:

$$f(x) = M \cdot x + v, \tag{3.1}$$

де M - матриця перетворення;

v - новий базис.

Матриця перетворення отримується у такий спосіб:

$$M = \begin{bmatrix} \alpha & \beta & (1-\alpha)\cdot x - \beta \cdot y \\ -\beta & -\alpha & \beta \cdot x - (1-\alpha)\cdot y \end{bmatrix}, \quad (3.2)$$

$$\alpha = N \cdot \cos A; \ \beta = N \cdot \sin A.$$

Координати *x* і *y* - це координати центру вихідного зображення. Параметр *N* - це ізотропний масштабний коефіцієнт. Кут *A* - це той кут, на який необхідно повернути зображення.

Результат роботи методу детектування повороту представлений на рисунку 3.21.

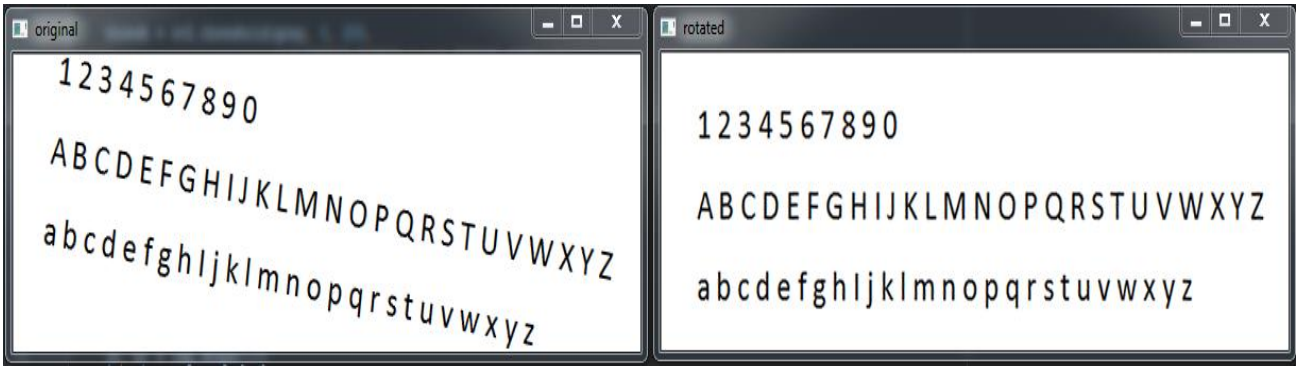


Рисунок 3.21 – Результат роботи методу детектування і усунення повороту зображення

3.3.2 Усунення неоднозначностей при розпізнаванні

Як згадувалося раніше, в даній задачі є проблема, пов'язана з неоднозначним розпізнаванням. Причиною цієї проблеми є досить схожі пари символів. Для усунення цієї проблеми можна скористатися додатковим методом порівняння зображень [46].

Ідея всього алгоритму представлена на схемі (рисунок 3.22).

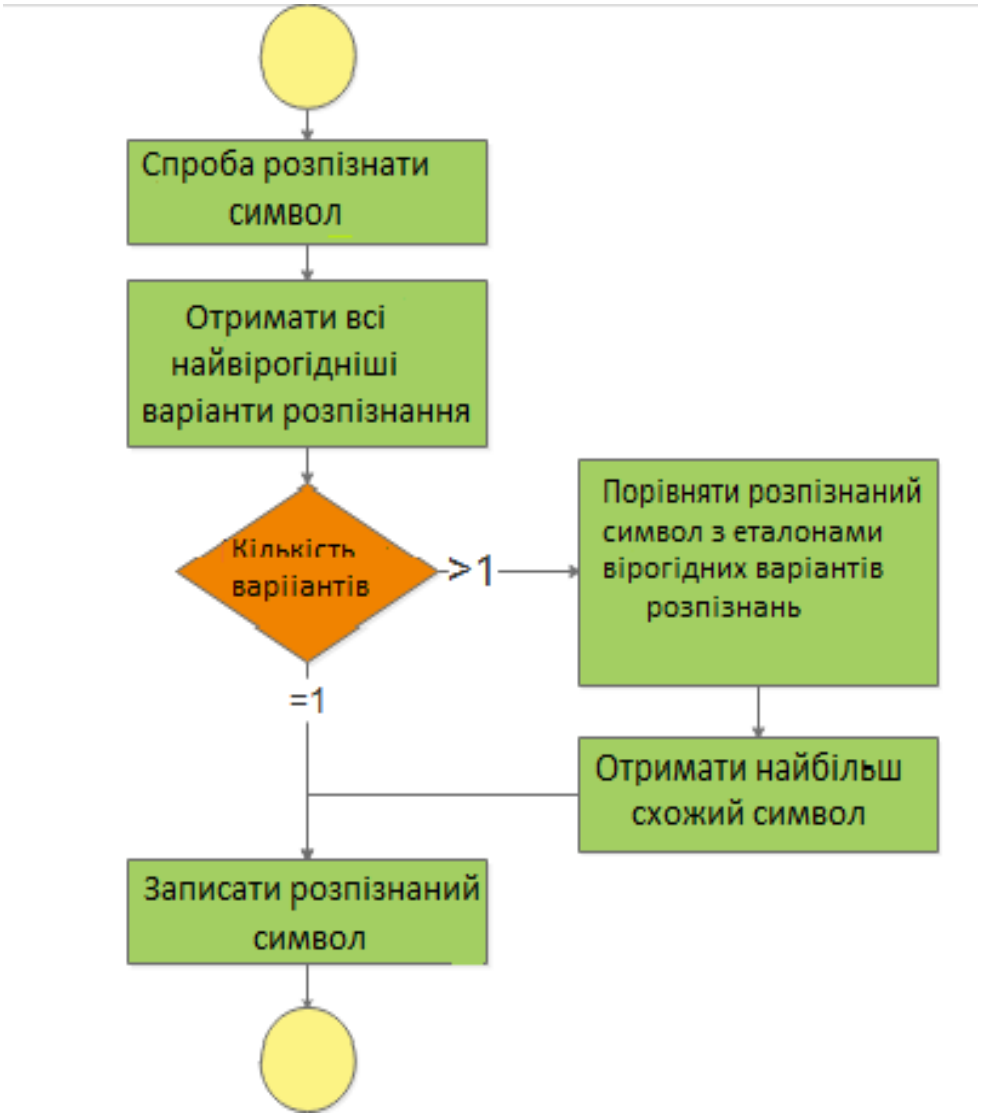


Рисунок 3.22 – Алгоритм усунення неоднозначностей

В якості вихідного методу порівняння зображень був обраний метод середньоквадратичної помилки:

$$MSE = \frac{1}{l \cdot w} \sum_{i=0}^{l-1} \sum_{j=0}^{w-1} (Letter(i, j) - \mathrm{Re}\ ference(i, j))^2 . \qquad (3.3)$$

Незважаючи на те, що метод в середньому показує хороші результати порівняння, даний метод не враховує деякі особливості зображень.

Для більш точного порівняння зображень було прийнято рішення використовувати індекс структурного подібності [30].

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}. \quad (3.4)$$

На основі розроблених теоретичних положень запропоновано алгоритми, результат роботи яких наведено на рисунку 3.23.

Рисунок 3.23 – Результат роботи розглянутих алгоритмів.

На правому рисунку видно, що алгоритм детектування і усунення повороту зображення показує меншу помилку, що означає велику схожість, однак зображення відрізняються сильніше, ніж на лівому рисунку. У той же час алгоритм усунення неоднозначностей повернув вірний результат.

3.3.3 Використання ансамблів з нейронних мереж

В якості ще одного методу, що підвищує точність розпізнавання, був обраний метод використання ансамблю з нейронних мереж. Суть методу полягає в тому, що рішення приймає не одна нейронна мережа, а кілька. У підсумку, всі ймовірності підсумовуються, потім вираховується середнє, і, нарешті, отримується відповідь, які значення дорівнюють максимальному значенню ймовірності.

Ідея використання декількох нейронних мереж заснована на тому, що кожна мережа «голосує» за той чи інший варіант і припущення кожної мережі містить якусь помилку щодо правильного варіанта. З ростом числа голосуючих мереж, ця помилка в середньому зменшується.

В даному випадку було прийнято рішення використовувати вісім нейронних мереж. Для досягнення найкращого результату мережі були навчені не на одному наборі даних, а на чотирьох. Відмінностями в навчальних даних був кут повороту символу: 0, 90, 180 і 270 градусів відповідно. Таким чином, вісім нейронних мереж були розбиті на пари, які навчалися на одному з перерахованих навчальних наборах.

Пари нейронних мереж також мали внутрішні відмінності. Одна з нейронних мереж навчалася без застосування регуляризації, інша навчалася із застосуванням регуляризації. Таке рішення засноване на тому, що в більшості випадках розпізнаватися будуть стандартні комп'ютерні шрифти, тому, якщо одна з мереж перевчитися на навчальних даних, в які входить більшість стандартних шрифтів, то результат від цього не погіршиться.

3.3.4 Тестування ансамблів з нейронних мереж

Перше тестування проводилося на зображенні, представленому на рисунку 3.24.

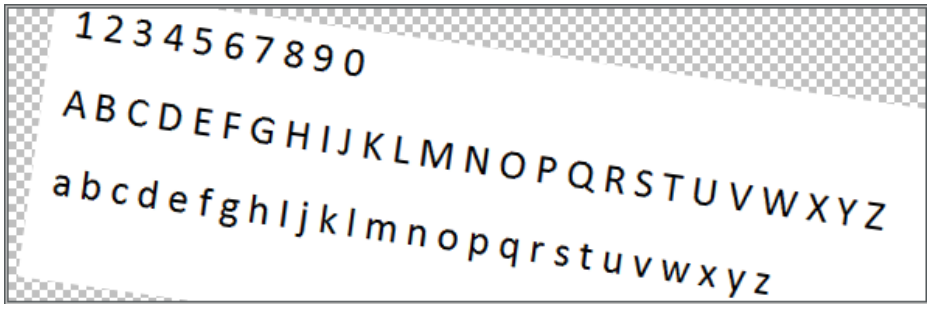


Рисунок 3.24 – Вхідний зображення з поворотом

При використанні ансамблів нейронних мереж було отримано наступний результат представлений на рисунку 3.25:

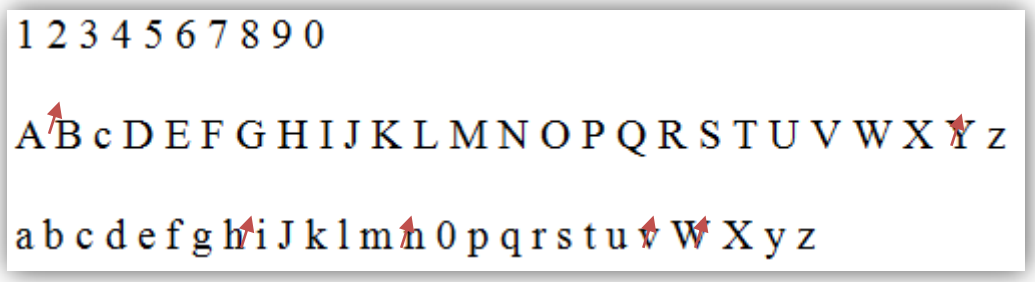


Рисунок 3.25 – Результат роботи алгоритму з додатковими методами підвищення точності розпізнавання.

Стрілки, що показують неправильно розпізнані символи. Як видно з результатів, серйозних помилок немає, проте помилки, пов'язані з нижнім і верхнім регістрами залишилися. Також є деяка кількість помилок, пов'язаних з описаними раніше неоднозначно, але в порівнянні з попереднім результатом, їх стало менше.

На тестовій вибірці алгоритм розпізнавання показав точність 92%.

Отже, в ході розробки додаткових методів, що підвищують точність класифікації, були досягнуті наступні результати:

- за допомогою ансамблю нейронних мереж вдалося підвищити загальний відсоток правильно розпізнаних символів;
- за допомогою методу усунення повороту зображення вдалося усунути негативний ефект від повороту і також підвищити стійкість алгоритму до такого ефекту;
- за допомогою методу порівняння зображень вдалося лише трохи усунути проблему неоднозначності.

В цілому, результати розробки позитивні, проте необхідна доробка деяких деталей алгоритму.

Вихідний код кінцевого варіанту модуля розпізнавання представлений в додатку Д.

Для створення програми, за допомогою якої можна перевести символи з зображення в текст, на наш погляд, оптимально використовувати мову Delphi - мова високого рівня. Для реалізації програми використовувалася система програмування Delphi 6 форми Enterprise (Borland), оскільки вона дає широкий набір можливостей при програмуванні додатків ОС Windows. В основі Delphi використовується мова Object Pascal - розширення об'єктно-орієнтованої мови Pascal. До складу Delphi ще входять локальний SQL-сервер, бібліотека візуальних компонентів, генератори звітів і ін. Delphi створює маленькі за розміром високоефективні модулі (такі як .dll .exe), це означає, що вимоги до робочих місць клієнтів значно знижуються. Переваги Delphi:

- хороша опрацювання ієрархії об'єктів; - підтримка всіх вимог, які пред'являються до об'єктно-орієнтованого мови програмування;
- швидке створення інтерфейсу;
- зручна IDE;
- висока продуктивність розробленого додатка;
- хороші засоби налагодження;
- компоненти доступу до даних: BDE, ODBC, ADO вже вбудовані;
- підтримка багатоланкової технології (multi-tiered) доступу до даних;
- можливість створення нових компонентів і інструментів засобами Delphi;
- доступ до візуальних компонентів freeware, shareware і комерційних фірм;
- низькі вимоги до ресурсів комп'ютера.

Інтерфейс програми представлений на рисунку 3.26.

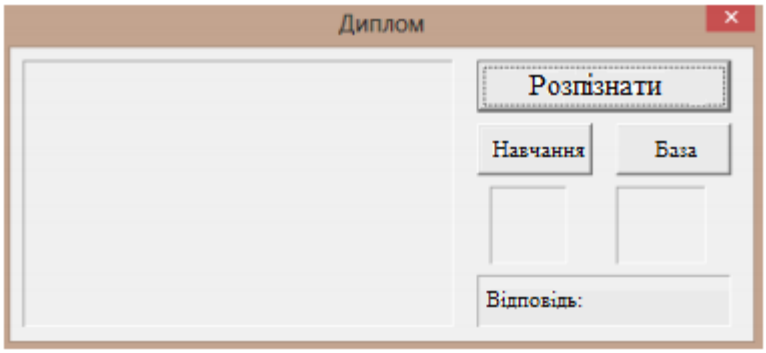


Рисунок 3.26 – Інтерфейс програми

У вікні відображається завантажене зображення. Кнопка "Розпізнати" - завантажуємо зображення для розпізнавання. Кнопка "Навчання" - завантажується зображення, яке хочемо навчити мережу (зображення або число). Після завантаження мережу запитує, чи вірно вона розпізнала образ (приклад на рисунку 3.27). Якщо так, то нічого не потрібно робити, якщо не вірно, то вказуємо, яким повинен бути результат розпізнавання (рисунки 3.28 і 3.29).

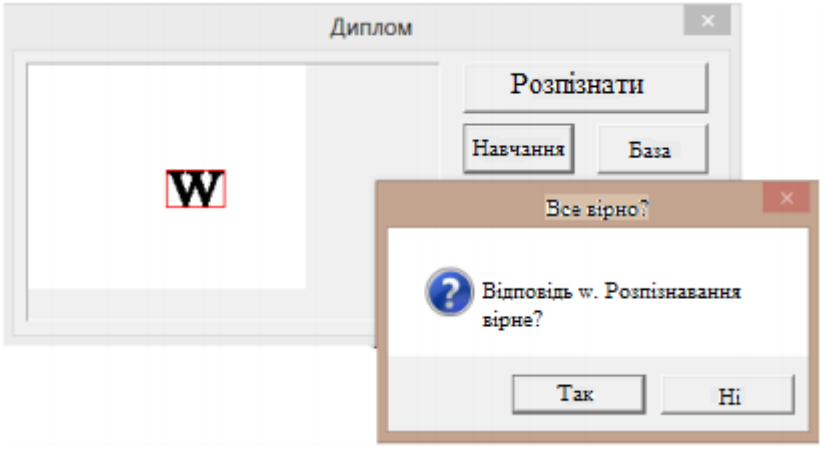


Рисунок 3.27 - Вірне розпізнавання прикладу для навчання

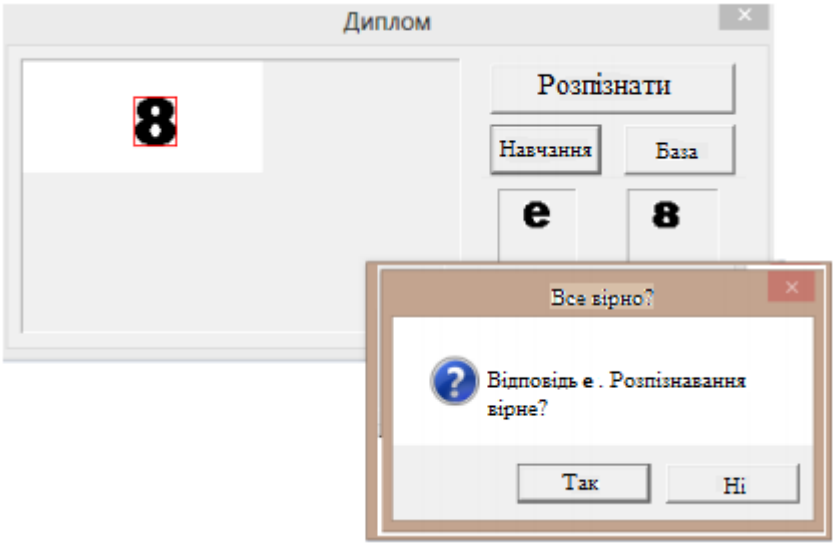


Рисунок 3.28 - Невірне розпізнання прикладу для навчання

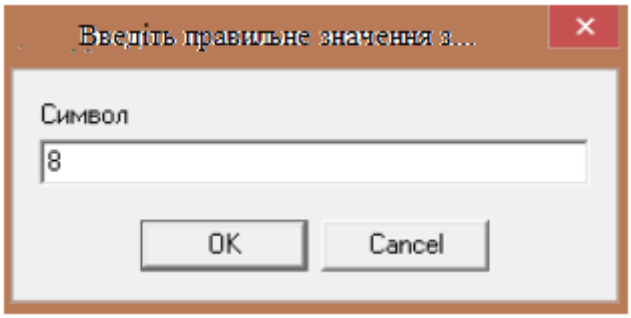


Рисунок 3.29 – Навчання нейронної мережі

Кнопка "База" – здійснює вибір бази, по якій проводиться навчання та розпізнавання. Для простоти взята база шрифтів, так як в основному проводиться розпізнавання тексту. Також є два маленьких вікна, в першому показаний розпізнаний образ з бази, а в другому - розпізнаний образ з зображення. В останньому вікні виводиться результат розпізнавання програми - зображений текст. За допомогою функції function TForm1.PreParse1 (Pic: TPicture): string; знаходимо всі символи на зображенні. Для початку переведемо зображення в чорно-білий колір за допомогою процедури procedure TForm1.Mono (Bmp: TBitmap). Кожен знайдений символ обводимо червоною рамкою. Розпізнавання відбувається за допомогою функції function TForm1.Compare (b1, b2: TBitmap): integer. Далі складаємо базу з зображеннями символів і цифр. Приклад розпізнавання зображення вже навченої мережею представлений на рисунку 3.29.

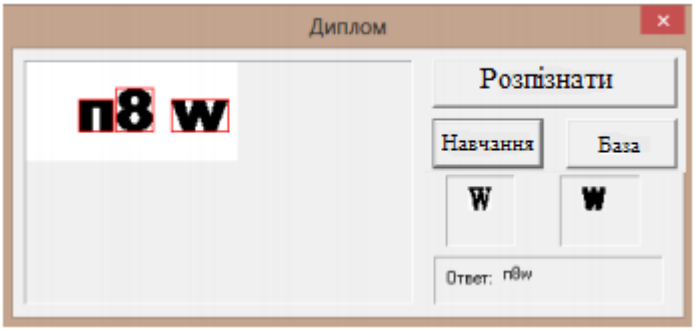


Рисунок 3.30 – Розпізнавання

Слід зазначити у програми недоліки, такі як:

- нездатність розпізнавати великі тексти і пропозиції;

- некоректне розпізнавання при поворотах і шуми. Дана програма добре підходить для розпізнавання простої капчі.

Справедливості заради зазначимо, вона не може конкурувати в розпізнаванні текстів з FineReader, але зі своїм завданням справляється цілком. Тепер розглянемо застосування нейромереж для відновлення зображень, тобто необхідна програма, яка в зіпсованому зображенні розпізнає еталонний образ. Для даного завдання краще використовувати нейронну мережу Хопфілда, яка складається з одного шару нейронів (де їх число - це і кількість входів і виходів мережі). Всі нейрони пов'язані один з одним синапсис. Є і єдиний вхідний синапсис, за допомогою якого вводиться сигнал. Вихідні сигнали же утворюються на аксонах. Для роботи мережі необхідно надходження на вхід набору двійкових сигналів, які будуть вважатися зразковими. Мережа повинна виділити відповідний зразок з довільного неідеального сигналу, який подається на вхід, або знайти самий схожий образ. Для нейронної мережі потрібна бінарна послідовність з -1 і +1. "-1" - піксель чорного кольору, "-1" - піксель білого кольору. Тепер ми можемо уявити зображення як послідовність. Щоб відновлення не займало багато часу і було досить точним, скористаємося наступною схемою: візьмемо зображення 100x100 пікселів, для кожного пікселя є свій нейрон, отримуємо мережу з 10000 нейронів.

При розробці програми використовувалася програма Microsoft Visual Studio - лінійка продуктів компанії Майкрософт, який включає в себе інтегровану середу розробки програмного забезпечення, а також багато інших інструментальні засоби. Ці продукти допомагають розробляти консольні додатки, додатки з використанням графічного інтерфейсу, веб-сайти, веб-додатки, При запуску програми з'являється вікно, яке представлено на рисунку 3.31.

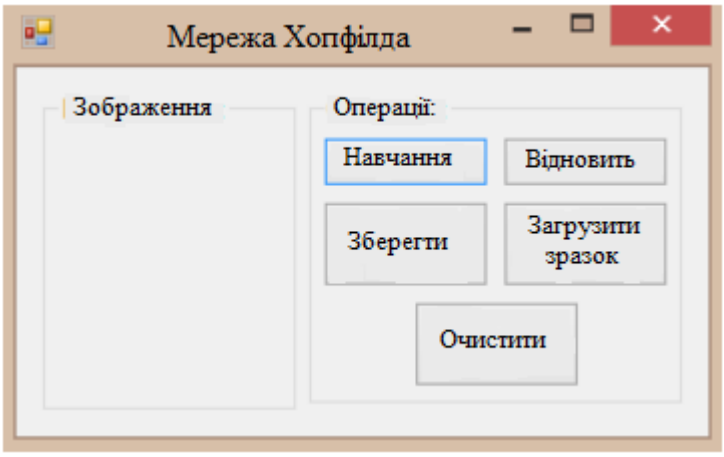


Рисунок 3.31 - Основне вікно програми

Є вікно, в якому показано завантажене зображення. Натискання на кнопку "Навчання" - зображення заноситься в базу. Кнопка "Відновити" - відновлюється завантажене зображення, пошук найбільш схожого зразка і заміна його замість спотвореного зображення. Кнопка "Зберегти" - зберігається зображення, завантажене в програму, на комп'ютер. Кнопка "Завантажити зразок" - завантажується зображення для відновлення або навчання мережі. Кнопка "Очистити" - очищається база. Навчимо програму двом зображень, показаним на рисунку 3.32.

Рисунок 3.32 – Вихідні зображення

Завантажуємо в програму спотворене зображення (рисунок 3.33), натискаємо кнопку "Відновити" і отримуємо той малюнок, на який спотворене зображення більше схоже (рисунок 3.34)

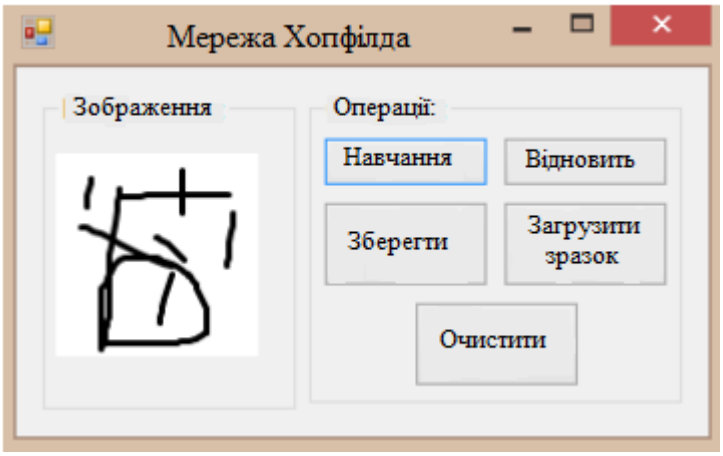


Рисунок 3.33 - Завантажене зображення для відновлення

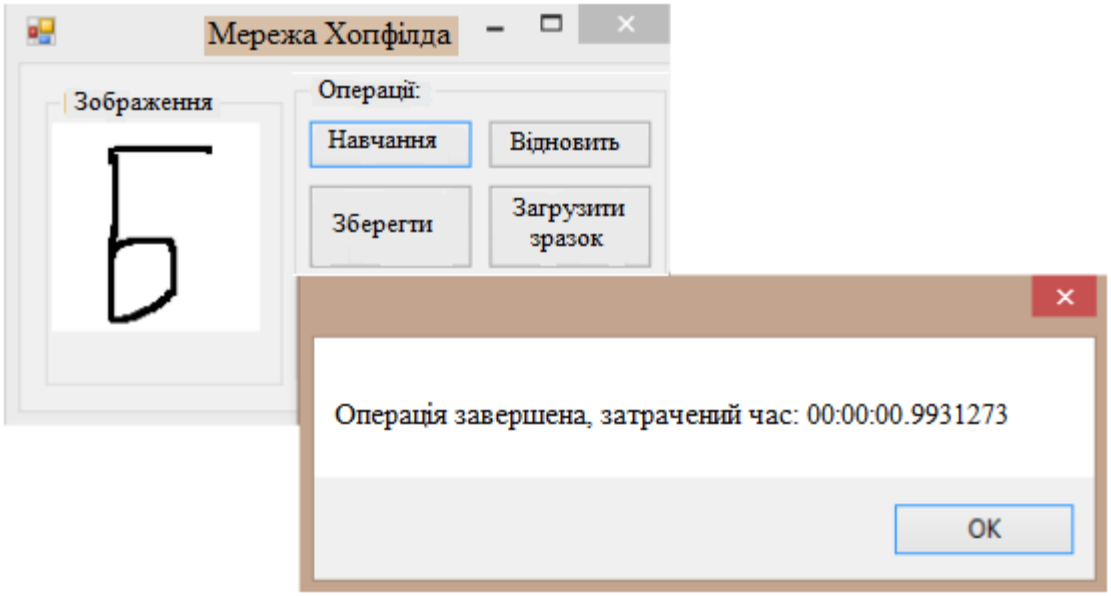


Рисунок 3.34 – Відновлене зображення

Дана програма успішно відновлює зіпсовані зображення, але іноді може невірно розпізнати і вивести неіснуючі зображення. Це пов'язано з обмеженістю можливостей мережі. Однією з причин може бути схожість вихідних зображень, що буде викликати у мережі перехресні асоціації. Програма може також використовуватися для розпізнавання капчі, але для цього мережу потрібно успішно навчити. Також можна її використовувати для усунення помилок в словах і ін. Недоліком програми є те, що вона працює тільки з зображеннями розміром 100x100 пікселів. На основі проведених досліджень був розроблений програмний продукт для обробки зображень, здатний за допомогою штучних

нейронних мереж розпізнавати ці зображення. Створено дві штучних нейронних мереж з учителем, які обробляють і модифікують еталонну базу образів.

4 ТЕХНІКО-ЕКОНОМІЧНИЙ РОЗДІЛ

Метою техніко – економічного розділу дипломної роботи є здійснення економічних розрахунків, спрямованих на визначення економічної ефективності програмного модуля розпізнавання текстової інформації на основі штучних нейронних мереж та прийняття рішення про його подальший розвиток і впровадження або ж недоцільність проведення відповідної розробки. Для проведення даного дослідження необхідно провести ряд розразунків.

4.1 Розрахунок витрат на розробку програмного модуля

Витрати на розробку і впровадження програмного модуля для маршрутизації запитів у комп'ютерній мережі ($К$) включають:

$$К = К_1 + К_2,$$

де $К_1$ - витрати на розробку апаратного та програмного забезпечення грн.;

$К_2$ - витрати на відлагодження і дослідну експлуатацію програми рішення задачі на комп'ютері, грн.

Витрати на розробку апаратних та програмних засобів включають:

- витрати на оплату праці розробників ($В_{ОП}$);
- витрати на відрахування у спеціальні державні фонди ($В_ф$);
- витрати на матеріали та комплектуючі ($П_в$);
- накладні витрати ($Н$);
- інші витрати ($І_в$)
- витрати на використання комп'ютерної техніки($В_{КТ}$) .

Витрати на оплату праці включають заробітну плату (ЗП) всіх категорій працівників, безпосередньо зайнятих на всіх етапах проектування. Розмір ЗП обчислюється на основі трудоємності відповідних робіт у людино-днях та середньої ЗП відповідних категорій працівників.

У розробці проектного рішення задіяні наступні спеціалісти - розробники, а саме: керівник проекту; студент-дипломант; консультант техніко-економічного розділу (таблиця 4.1).

Таблиця 4.1 - Вихідні дані для розрахунку витрат на оплату праці

№п/п	Посада виконавців	Місячний оклад, грн.
1	Керівник ДП, викладач	6026
2	Консультант техніко-економічного розділу, доцент	6026
3	Студент	1100

Витрати на оплату праці розробників проекту визначаються за наступною формулою:

$$B_{ОП} = \sum_{i=1}^{N} \sum_{j=1}^{M} n_{ij} \cdot t_{ij} \cdot C_{ij} \ , \qquad (4.1)$$

де n_{ij} – чисельність розробників i-ої спеціальності j-го тарифного розряду, осіб;

t_{ij} – затрачений час на розробку проекту співробітником i-ої спеціальності j-го тарифного розряду, год;

C_{ij} – годинна ставка працівника i-ої спеціальності j-го тарифного розряду, грн.,

Середньогодинна ставка працівника може бути розрахована за такою формулою:

$$C_{ij} = \frac{C_{ij}^{0}(1+h)}{РЧ_i}, \qquad (4.2)$$

де C_{ij} – основна місячна заробітна плата розробника *i*-ої спеціальності *j*-го тарифного розряду, грн.;

h – коефіцієнт, що визначає розмір додаткової заробітної плати (при умові наявності доплат);

$РЧ_i$ - місячний фонд робочого часу працівника *i*-ої спеціальності *j*-го тарифного розряду, год. (приймаємо 168 год.).

Коефіцієнт *h*, який визначає розмір додаткової заробітної плати, для керівника та консультанта техніко-економічного розділу дорівнює 0,47.

Результати розрахунку записують до таблиці 4.2.

Таблиця 4.2 - Розрахунок витрат на оплату праці

№ п/п	Посада виконавців	Час розробки, год	Погодинна заробітна плата, грн/год.	Витрати на розробку, грн
1	Керівник ДП, доцент	16	88,6	1417,6
2	Консультант техніко-економічного розділу, доцент	2	88,6	177,2
3	Студент	144	6,55	943,2
Разом				2538

Відрахування на соціальні заходи. Величну відрахувань у спеціальні державні фонди визначають у відсотковому співвідношенні від суми основної та додаткової заробітних плат. Згідно діючого нормативного законодавства сума відрахувань у спеціальні державні фонди складає 20,5% від суми заробітної плати: $B_Ф = \frac{20,5}{100} \cdot 2538 = 520,29$ грн.

Загальна сума витрат на матеріальні ресурси (B_M) визначається за формулою:

$$B_M = \sum_{i=1}^{n} K_i \cdot Ц_i, \qquad (4.3)$$

де K_i - витрата *i*-го типу матеріалу, натуральні одиниці вимірювання;

$Ц_i$ - ціна за одиницю *i*-го типу матеріалу, грн.;

i - тип матеріального ресурсу;

n - кількість типів матеріальних ресурсів.

Таблиця 4.3 - Зведені розрахунки матеріальних витрат

№ п/п	Найменування матеріальних ресурсів	Од. виміру	Факт. витрачено матеріалів	Ціна за одиницю, грн.	Сума, грн	Транспортні витрати (10% від суми)	Загальна сума, грн
	Допоміжна література	шт	1	600	600	60	660
	Папір (формат А4)	уп	2	80	160	16	176
	Ручка кулькова	шт	2	10	20	2	22
	Олівець простий	шт	2	10	20	2	22
	Диски CD-R	шт	2	15	30	3	33
	Зошит, 96 арк	шт	1	50	50	5	55
	Тонер для принтера	уп	1	90	90	9	99
	Канцелярскі маркери (червоний, зелений)	шт	2	20	40	4	44
	Р а з о м						1111,00

Витрати на використання комп'ютерної техніки ($В_{КТ}$) включають витрати на амортизацію комп'ютерної техніки, витрати на користування програмним забезпеченням, витрати на електроенергію, що споживається комп'ютером. За даними обчислювального центру ТНЕУ для комп'ютера типу IBM PC/ATX вартість години роботи становить 6 грн. Середній щоденний час роботи на комп'ютері – 2 години. Розрахунок витрат на використання комп'ютерної техніки приведений в таблиці 4.4.

Таблиця 4.4- Розрахунок витрат на використання комп'ютерної техніки

№ п/п	Назва етапів робіт, при виконанні яких використовується комп'ютер	Час використання комп'ютера, год.	Витрати на використання комп'ютера грн.
1	Проведення досліджень та оформлення їх результатів	60	360
2	Оформлення техніко-економічного розділу	8	48
3	Оформлення ДП	12	72
Разом		80	480

Накладні витрати проектних організацій включають три групи видатків: витрати на управління, загальногосподарські витрати, невиробничі витрати. Вони розраховуються за встановленими відсотками до витрат на оплату праці. Середньостатистичний відсоток накладних витрат приймемо 150% від заробітної плати: $H = 1{,}5 \cdot 1860{,}4 = 2790{,}6$ (грн).

Інші витрати є витратами, які не враховані в попередніх статтях. Вони становлять 10% від заробітної плати: $I_B = 1860{,}4 \cdot 0{,}1 = 186{,}04$ (грн).

Витрати на розробку програмного забезпечення складають:

$$K_1 = B_{ОП} + B_Ф + B_M + H + I_B + B_{КТ},$$

$$K_1 = 1860{,}4+381{,}38+1111{,}00+2790{,}6+186{,}04+480{,}00=6809{,}42 \text{ (грн)} .$$

Витрати на відлагодження і дослідну експлуатацію програмного продукту визначаємо за формулою:

$$K_2 = S_{м.г.} \cdot t_{від} \qquad (4.4)$$

де $S_{м.г.}$ - вартість однієї машино-години роботи ПК, грн./год;

$t_{від}$ - комп'ютерний час, витрачений на відлагодження і дослідну експлуатацію створеного програмного продукту, год.

Загальна кількість днів роботи на комп'ютері дорівнює 30 днів. Середній щоденний час роботи на комп'ютері – 2 години. Вартість години роботи комп'ютера дорівнює 6 грн., тому K_2= 6·60 = 360 грн.

4.2 Визначення експлуатаційних витрат

Для оцінки економічної ефективності розроблювальної системи моніторингу слід порівняти її з аналогом, тобто існуючим програмним забезпеченням ідентичного функціонального призначення.

Експлуатаційні одноразові витрати по програмному забезпеченню і аналогу включають вартість підготовки даних і вартість роботи комп'ютера (за час дії програми):

$$E_{П} = E_{1П} + E_{2П},$$

де $E_{П}$ - одноразові експлуатаційні витрати на ПЗ (аналог), грн.;

$E_{1П}$ - вартість підготовки даних для експлуатації ПЗ (аналогу), грн.;

$E_{2П}$ - вартість роботи комп'ютера для виконання проектного рішення (аналогу), грн.

Річні експлуатаційні витрати $B_{ЕП}$ визначаються за формулою:

$$B_{ЕП} = E_{П} * N_{П},$$

де $N_{П}$ - періодичність експлуатації ПЗ (аналогу), раз/рік.

Вартість підготовки даних для роботи на комп'ютері визначається за формулою:

$$E_{1П} = \sum_{l=1}^{n} n_i t_i c_i ,$$

де *i* - категорії працівників, які приймають участь у підготовці даних (i=1,2,...n);

n_i - кількість працівників *i*-ої категорії, осіб.;

t_i - трудомісткість роботи співробітників *i*-ої категорії по підготовці даних, год.;

c_i - середнього годинна ставка працівника *i*-ої категорії з врахуванням додаткової заробітної плати, що знаходиться із співвідношення:

$$c_i = \frac{c_i^0 (1+b)}{m},$$

де c_i^0 - основна місячна заробітна плата працівника *i*-ої категорії, грн.;

b - коефіцієнт, який враховує додаткову заробітну плату (приймемо 0,57);

m - кількість робочих годин у місяці, год.

Для роботи з даними як для проектного рішення так і аналогу потрібен один працівник, основна місячна заробітна плата якого складає: *c* = 3723 грн. Тоді:

$$c_1 = \frac{3723(1+0{,}57)}{22*8} = 33{,}21 \text{грн/год}$$

Трудомісткість підготовки даних для проектного рішення складає 1 год., для аналога 1,5 год.

Таблиця 4.5- Розрахунок витрат на підготовку даних та реалізацію проектного рішення на комп'ютері

№	Час роботи співробітників, год.	Середньогодинна заробітна плата, грн./год.	Витрати , грн.
Проектне рішення			
1	1	33,21	33,21
Аналог			
2	1,5	33,21	66,42

Витрати на експлуатацію комп’ютера визначається за формулою:

$$E_{2П} = t * S_{МГ}$$

де t - витрати машинного часу для реалізації рішення (аналогу), год.;

$S_{МГ}$ - вартість однієї години роботи комп’ютера, грн./год.

Далі:

$E_{2П}$ = 1* 6 = 6 грн.; $E_{2А}$=1,5*6 = 9 грн.

$E_{ІП}$=33,21+6=39,21 грн. ; $E_{А}$=66,42+9 = 75,42 грн.

$B_{ЕП}$= 39,21*252=9880,92 грн. ; $B_{ЕА}$=75,42 *252=19005,84 грн.

Накладні витрати пов’язані з обслуговуванням виробництва, утриманням апарату управління підприємства (фірми) та створення необхідних умов праці.

В залежності від організаційно-правової форми діяльності господарюючого суб’єкта, накладні витрати можуть становити 60–100 % від суми основної та додаткової заробітної плати працівників.

$$H_B = 0{,}7 * B_{ОП}, \qquad (4.5)$$

де H_B – накладні витрати.

H_B=0,7*5845,11=4091,58 грн.

Результати проведених розрахунків зведемо у таблицю 4.6.

Таблиця 4.6 - Кошторис витрат

№ п/п	Найменування витрат	Сума витрат, грн.
1	2	3
1	Витрати на оплату праці	1860,4

Продовження таблиці 4.6

1	2	3
2	Відрахування у спеціальні державні фонди	381,38
3	Витрати на матеріали та комплектуючі	1111,00
4	Накладні витрати на розробку	2790,6
5	Інші витрати	186,04
6	Витрати на відлагодження і дослідну експлуатацію програмного продукту	360
7	Накладні витрати експлуатацію	4091,58
8	Річні експлуатаційні витрати	19005,84
Разом		29786,09

Договірна ціна ($Ц_Д$) для проектних рішень розраховується за формулою:

$$Ц_Д = B_{КС} \cdot \left(1 + \frac{p}{100}\right), \quad (4.6)$$

де $B_{КС}$ – кошторисна вартість, грн.;

p - середній рівень рентабельності, % (приймаємо 26% за погодженням з керівником): $Ц_Д$ =29786,09·(1+0,26)=37530,47 грн.

4.3 Визначення економічної ефективності і терміну окупності капітальних вкладень

Економічна ефективність ($E_ф$) полягає у відношенні результату виробництва до затрачених ресурсів:

$$E_ф = \frac{П}{B_{КС}}, \quad (4.7)$$

де $П$ – прибуток, грн.;

$B_{КС}$ – кошторисна вартість, грн..

$$E_{ф}=7812,27 \text{ грн.} / 29786,09 \text{ грн.} = 0,25.$$

Поряд із економічною ефективністю розраховують термін окупності капітальних вкладень (Tp):

$$T_P = \frac{1}{E_P} \ . \qquad (4.8)$$

Тобто: T_P =1/0,25 = 4р.

Прийнятним вважається термін окупності, близький до 7 років.

Розраховані економічні показники проекту занесемо до таблиці 4.7.

Таблиця 4.7 - Економічні показники розробки

№ п/п	Показник	Значення
1.	Собівартість, грн.	29786,09
2.	Плановий прибуток, грн.	7744,38в
3.	Ціна, грн.	37530,47
4.	Економічна ефективність	0,25
5.	Термін окупності, рік	4

Враховуючи основі економічні показники з таблиці 4.7, можна зробити висновок, що при економічній ефективності 0,25 та терміні окупності 4 роки проводити роботи по впровадженню даного програмного модуля є доцільним та економічно вигідним.

ВИСНОВКИ

1. В ході розробки методів і алгоритмів розпізнавання тексту за допомогою нейронних мереж було проведено аналіз методів обробки зображень, машинного навчання та оптимізації процесу навчання. На основі чого були встановлені переваги і недоліки на основі яких обґрунтовано вибір підходів.

2. Створені алгоритми обробки зображень і спроектована нейронна мережа, на основі якої були протестовані як на окремих тестових вибірках, так і на наближеним до реальних зображеннях. В результаті чого встановлено, що найкращою точністю 92 % володіє метод на основі нейронної мережі зі зміненою топологією.

3. Створена власна система детектування окремих символів, яка дозволяє з допустимою точністю встановлювати символи.

Незважаючи на досить високий показник точності, в деяких випадках цього буває недостатньо, тому дане рішення може бути в подальшому модифіковано. Як деяких модифікацій можна застосувати наступне:

1. Для сегментації зображення на окремі символи можна використовувати рекурентні нейронні мережі;
2. Більш тривале навчання згортальних нейронних мереж;
3. Збільшення навчальної вибірки;
4. Використання нейронної мережі для розпізнавання не окремих символів, а цілих слів. Така мережа може бути використана або як доповнення до вже навченої мережі, або як повна її заміна;
5. Поліпшення методів обробки зображення для усунення решти негативних ефектів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Flach P. Beyond binary classification // P. Flach / In: Machine Learning: The Art and Science of Algorithms that Make Sense of Data. – Glasgow: Cambridge University Press, 2012. – P. 81-104.
2. Buisson L. Machine learning classification of SDSS transient survey images // L. Buisson, N. Sivandam, B. Basett, M. Smith / Monthly Notices of the Royal Astronomical Society, Vol. 2, No. 454, 2015. – P. 2026-2038.
3. Cao J., al E. Extreme learning machine and adaptive sparse representation for image classification // Neural networks, No. 81, 2016. – P. 91-102.
4. Xiao T. The application of two-level attention models in deep convolutional neural network for fine-grained image classification // In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015. – P. 842-850.
5. Гонсалес Р. Основные стадии цифровой обработки изображений // Р. Гонсалес, Р. Вудс/ Цифровая обработка изображений. – Москва: Техносфера, 2005. –Р. 56-60.
6. Гонсалес Р. Цветовые модели // Р. Гонсалес, Р. Вудс/ Цифровая обработка изображений. –Москва: Техносфера, 2005.– С 426-439.
7. Helland T. Seven grayscale conversion algorithms 2011. [Електронний ресурс] / URL: http:// www.tannerhelland.com/3643/grayscale-image-algorithm-vb6/ (дата звертання: 17.03.2018).
8. Cook J.D. Three algorithms for converting color to grayscale 2009. [Електронний ресурс] / URL: https://www.johndcook.com/blog/2009/08/24/ algorithms-convert-color-grayscale/ (дата звертання: 17.03.2018).
9. OpenCV. Geometric transformations URL: [Електронний ресурс] / http://docs.opencv.org/2.4/modules/imgproc/doc/geometric_transformations.html#getperspectivetransform (дата звертання: 17.03.2018).

10. OpenCV. Operations on arrays URL: [Електронний ресурс] / http:// docs.opencv.org/2.4/modules/core/doc/operations_on_arrays.html#SVD (дата звертання: 20.04.2018).

11. OpenCV. Geometric transformations URL: [Електронний ресурс] / http://docs.opencv.org/2.4/modules/imgproc/doc/geometric_transformations.html#warp perspective (дата звертання: 19.04.2018).

12. Гонсалес Р. Пороговая обработка // Р. Гонсалес, Р. Вудс/ Цифровая обработка изображений. – Москва: Техносфера, 2005. – С. 850-874.

13. Gary Bradski A.K. Threshold // In: Learning OpenCV. – Sebastopol: O'Reilly Media, Inc, 2008. – P. 135-138.

14. Гонсалес Р., Вудс Р. Обработка с глобальным порогом // Р. Гонсалес, Р. Вудс/ Цифровая обработка изображений.– Москва: Техносфера, 2005. – С. 855-858.

15. OpenCV. Basic Thresholding Operations URL: [Електронний ресурс] / http://docs.opencv.org/2.4/doc/tutorials/imgproc/threshold/threshold.html (дата звертання: 21.04.2018).

16. Гонсалес Р. Обработка с адаптивным порогом // Р. Гонсалес, Р. Вудс/ Цифровая обработка изображений. – Москва: Техносфера, 2005. – С. 858-861.

17. Hanzra B.S. Adaptive Thresholding 2015. //[Електронний ресурс] / URL: http://hanzratech.in/2015/01/21/adaptive-thresholding.html (дата звертання: 25.04.2018).

18. Gary Bradski A.K. Adaptive Threshold // In: Learning OpenCV. – Sebastopol: O'Reilly Media, Inc, 2008. – P. 138-140.

19. Fisher R., Perkins S., Walker A., Wolfart E. Adaptive Thresholding //[Електронний ресурс] / URL: http://homepages.inf.ed.ac.uk/rbf/HIPR2/adpthrsh.htm (дата звертання: 26.04.2018).

20. OpenCV. Image Thresholding \ Adaptive Thresholding [Електронний ресурс] / URL: http://docs.opencv.org/trunk/d7/d4d/tutorial_py_thresholding.html (дата звертання: 26.04.2018).

21. Seo J. Fast Contour-Tracing Algorithm Based on // J. Seo, S. Chae, J. Shim, D. Kim / – Sensors. 2015. – P. 1-27.

22. Samuel A.L. Some Studies in Machine Learning Using the Game of Checkers // A.L. Samuel/ IBM Journal, 1959. – P. 210-229.

23. MachineLearning.ru. Основные стандартные типы задач 2016. [Електронний ресурс]/ URL: http://www.machinelearning.ru/wiki/index.php?title=%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B5 (дата звертання: 6.04.2017).

24. Hastie T. Fitting Logistic Regression Models // T. Hastie, R. Tibshirani, J. Friedman/ The Elements of Statistical Learning. Data Mining, Inference, and Prediction.– Springer, 2004. – P. 120-122.

25. Hastie T. k-Nearest-Neighbor Classifiers // T. Hastie, R. Tibshirani, J. Friedman/ The Elements of Statistical Learning. Data Mining, Inference, and Prediction. – Springer, 2006. – P. 463-468.

26. Duan K.B., Keerthi S.S. Which Is the Best Multiclass SVM Method? An Empirical Study // K.B. Duan, S.S. Keerthi / Multiple Classifier Systems. – Springer, 2005. – P. 278-285.

27. Барский А. Б. Нейронные сети: распознавание, управление, принятие решений./ А. Б. Барский – М.: Финансы и статистика, 2004. – 176 с.

28. Большакова Е.И. Автоматическая обработка текстов на естественном языке и компьютерная лингвистика : учеб. пособие/ Е.И. Большакова и др. — М.: МИЭМ, 2011. – 272 с.

29. Галушкин А.И. Нейронные сети: основы теории //А.И. Галушкин/ Издательство «Горячая линия – Телеком», М., 2010. – 480 с.

30. Круг П.Г. Нейронные сети и нейрокомпьютеры: Учебное пособие по курсу "Микропроцессоры" по направлению "Информатика и вычислительная техника" / П. Г. Круг, Моск. энерг. ин-т (МЭИ ТУ) . – М. : Изд-во МЭИ, 2002 . – 176 с.

31. Оганезов А.Л. Применение нейронных сетей в задачах распознавания образов: дис. на соискание ученой степени канд. физ.-мат. наук. Тбилиси. 2006. – 149 с.

32. Саймон Хайкин. Нейронные сети: полный курс.// Саймон Хайкин/ 2-е изд. Пер. с англ. – М.: Издательский дом "Вильямс", 2006. – 1104 с.

33. Фролов А. Синтез и распознавание речи. Современные решения [Електронний ресурс] / А. Фролов, Г. Фролов. – Электрон. журн. – 2003. URL: http://www.frolov-lib.ru (дата звертання 20.03.2018).

34. Методичні рекомендації до виконання дипломного проекту з освітньо-кваліфікаційного рівня “Бакалавр” напряму підготовки 6.050102 «Комп’ютерна інженерія» фахового спрямування «Комп’ютерні системи та мережі» / О.М. Березький, Л.О.Дубчак, Р.Б. Трембач, Г.М. Мельник, Ю.М. Батько, С.В. Івасьєв / Під ред. О.М. Березького. - Тернопіль: ТНЕУ, 2016.–65 с.

35. Методичні вказівки до написання техніко-економічного розділу для дипломних проектів на здобуття освітньо-кваліфікаційного рівня “Бакалавр” напряму підготовки 6.050102 «Комп’ютерна інженерія» / І.Р.Паздрій. - Тернопіль: ТНЕУ, 2015.– 36 с.