Міністерство освіти і науки України
Дніпровський національний університет імені Олеся Гончара
Факультет прикладної математики
Кафедра комп'ютерних технологій

ЛАБОРАТОРНА РОБОТА №3

Виконавець: студент групи ПК-21м-1
Панасенко Єгор
Сергійович

Постановка задачі

Тема: «Побудова та налаштування персептрону. Розв’язання задачі розпізнавання за допомогою персептрону»

Мета роботи: закріпити навички застосування математичного апарата нейронних мереж для розв'язання задачі розпізнавання образів, навички розробки алгоритму навчання персептрона та застосування його до розпізнавання літер.

За результатами виконання лабораторної роботи студент повинен:

знати:

- алгоритм навчання персептрона;
- можливості й межі застосування персептронів;
- види активаційних функцій, які застосовуються у нейронних мережах;

вміти:

- розуміти постановку задач розпізнання образів;
- проектувати нейронну мережу й працювати з нею;
- проектувати й навчати персептрони;
- описати принцип дії персептрона Розенблатта.

Хід роботи

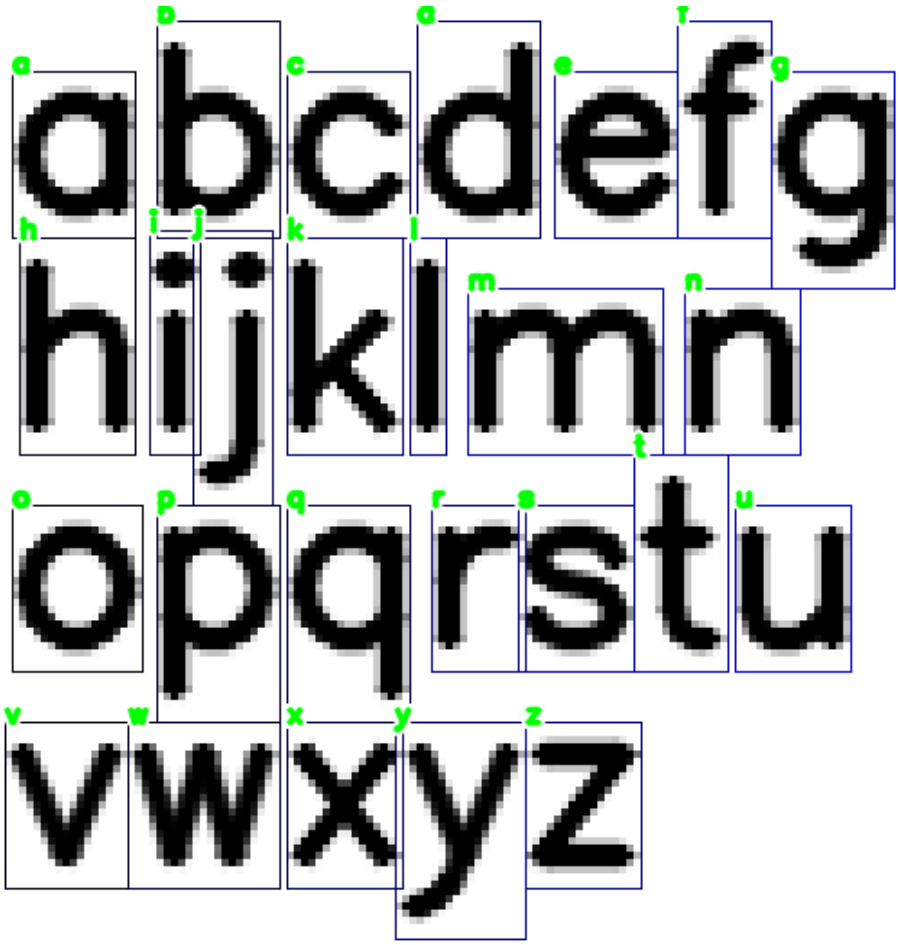
Для лабораторної роботи було розглянуто задачу розпізнавання образів друкованих літер. Було розроблено программу на мові Python з використанням бібліотеки OpenCV та Numpy для роботи із зображенням та для виконання математичних операцій.

Спочатку програма генерує зображення 140x140 із алфавітом маленьких літер англійської мови. На цьому зображенні, за допомогою вбудованого у OpenCV алгоритму пошуку контурів, знаходиться позиція кожної літери.

Загалом виникали проблеми із літерами “i” та “j” тому що вони за замовчанням знаходилися як два окремих контури, тому перед знаходженням контурів було прийнято рішення використовувати розмивання по вертикалі (по осі y).

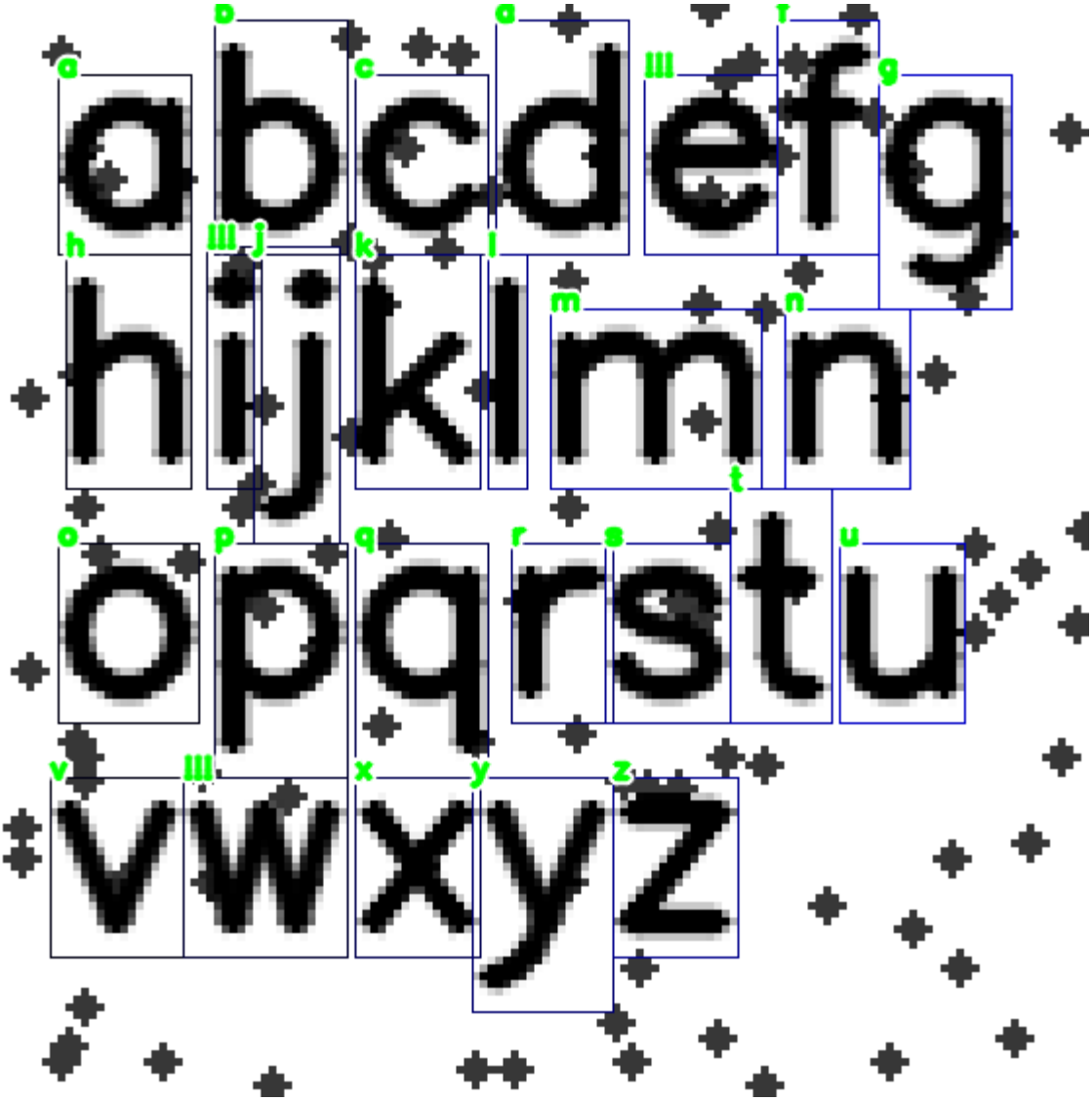
Далі після знаходження позиції кожної літери, використовується функція яка інтерпретує зображення літери у зрозумілій нейронній мережі данні, так як літери мають різний розмір, знайдену літеру розміщували на новому пустому зображенні розміром 40x30, далі зменшували до 12x9, таким чином отримували 108 значень від 0 до 255 у градації сірого. Ці значення використовуються як образ для нейронної мережі.

Далі програма навчає на основі цих образів літер найпростіший одношаровий персептрон за допомогою ітераційного алгоритму коректування синаптичних ваг. Після навчання програма використовує те ж саме зображення для розпізнавання другованих літер. Таким чином було отримане таке зображення:



На цьому зображенні прямокутниками виділено літеру для розпізнавання, та зеленою літерою позначено яка літера розпізналась за допомогою персептрона.

Далі на оригінальне зображення було додано артефакти для тестування персептрона у незвичайних ситуаціях і знову за допомогою нього було розпізнано літери. Результат розпізнавання бачимо на цьому зображенні:



Як бачимо персептрон у більшості випадків правильно розпізнав літери правильно, але для трьох випадків отримав помилку для літер “e”, “i”, “w”.

Код алгоритму:

```
import numpy as np
import cv2

width = 140
height = 140
lw = 12
lh = 9

alphabet = "".join([chr(ord("a")+i) for i in range(26)])
letters = {}
lettersw = {i: np.zeros((lw*lh,), dtype=int) for i in alphabet}

def prep(img):
    out = np.zeros((40, 30), np.uint8)+255
    h, w = img.shape
    out[:h, :w] = img
    out = cv2.resize(out, (lw,lh), interpolation=cv2.INTER_CUBIC)
    out = 255 - out.reshape((lw*lh,))
    return out

def calc(arr, l):
    S = sum([yi * wi for yi, wi in zip(arr, lettersw[l])])
    return 1 if S > 0 else 0

def calc_res(arr):
    res = "".join([i for i in alphabet if calc(arr, i)])
    return res or "!!!"

def test(img, letter):
    return calc(img, letter)


img = np.zeros((height,width), np.uint8)
img[:,:] = 255
font = cv2.FONT_HERSHEY_SIMPLEX
for i in range(4):
    cv2.putText(img, alphabet[i*7:i*7+7], (7,27+i*30), font, 1, (0, 255, 0), 2,
cv2.LINE_AA)

img2 = img.copy()

def find_text(img):
    kernel = cv2.getStructuringElement(cv2.MORPH_CROSS, (1, 3))
    img_er = cv2.erode(img, kernel, iterations=2)
    contours, hierarchy = cv2.findContours(img_er, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)
    cnts = [c for c, h in zip(contours, hierarchy[0]) if h[3] == 0]
    cnts = sorted(cnts, key=lambda i: (i[0][0][1]//20, i[0][0][0]))
    return [cv2.boundingRect(i) for i in cnts]

data = find_text(img)
print(len(data))
for i, (x,y,w,h) in enumerate(data):
    cv2.rectangle(img2,(x,y),(x+w,y+h),20+(i%7)*30,2)
    letters[alphabet[i]] = prep(img[y:y+h,x:x+w])

letters2 = {k: v.copy() for k, v in letters.items()}
```

```
for lj in alphabet:
    for _ in range(100):
        for lk in alphabet:
            while True:
                c = calc(letters[lk], lj)
                if c == int(lj == lk):
                    break
                elif c == 0:
                    lettersw[lj] += letters[lk]
                elif c == 1:
                    lettersw[lj] -= letters[lk]
        if all([calc(letters[x], lj) == int(lj == x) for x in alphabet]):
            break
    print([calc(letters[lj], i) for i in alphabet])

cv2.imshow("in", img2)

def recognize(img):
    img_bg = cv2.resize(img, (140*4, 140*4), interpolation=cv2.INTER_NEAREST)
    img_bg = cv2.cvtColor(img_bg, cv2.COLOR_GRAY2RGB)

    data = find_text(img)
    print(len(data))
    for i, (x,y,w,h) in enumerate(data):
        letr = prep(img[y:y+h,x:x+w])
        cv2.rectangle(img_bg,(4*x,4*y),(4*(x+w),4*(y+h)),20+(i%7)*30,1)
        res = calc_res(letr)
        cv2.putText(img_bg, res, (x*4, y*4), font, 0.5, (255, 255, 255), 6,
cv2.LINE_AA)
        cv2.putText(img_bg, res, (x*4, y*4), font, 0.5, (0, 255, 0), 2,
cv2.LINE_AA)
    return img_bg

cv2.imshow("orig", recognize(img))

noise = np.zeros((height,width), np.uint8)
cv2.randn(noise, 0, 50)
noise = 255 - (noise // 128 * 200)
kernel = cv2.getStructuringElement(cv2.MORPH_CROSS, (3, 3))
noise = cv2.erode(noise, kernel, iterations=2)
img = cv2.bitwise_and(img, noise, mask=noise)


cv2.imshow("out", recognize(img))

cv2.waitKey(0)
cv2.destroyAllWindows()
```