УДК 004.93'2

РОЗПІЗНАВАННЯ ТА КОРЕКЦІЯ ЗОБРАЖЕНЬ ДЕСЯТКОВОГО СКЛАДАННЯ У СТОВПЧИК ЗА ДОПОМОГОЮ CNN ТА АЛГОРИТМУ КОКА-ЯНҐЕРА-КАСАМІ

П. П. Пеліх[1, а], В. М. Кригін[2, б]

[1] *Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», НН Фізико-технічний інститут*

[2] *Міжнародний науково-навчальний центр інформаційних технологій та систем НАН України та МОН України*

Анотація

У даній роботі розглянуто метод розпізнавання та корекції зображень складання десяткових виразів у стовпчик. Метод використовує згорткову нейронну мережу та двовимірне узагальнення алгоритму Кока-Янґера-Касамі для зважених двовимірних контекстно-вільних граматик. Описано процес побудови мережі та її навчання, правила граматики для розпізнавання складання десяткових чисел у стовпчик, а також опис комп'ютерної програми, де реалізовано запропонований метод.

Ключові слова: згорткові нейронні мережі, алгоритм Кока-Янґера-Касамі, аналіз зображень математичних виразів

Вступ

Структурне розпізнавання візуальних образів — це сукупність методів, які аналізують зв'язки між об'єктами на зображенні для їх розпізнавання. Воно знайшло своє застосування в обробці документів, а також різноманітних рукописних виразів, написання яких чітко визначено деякими правилами, до яких відносяться й математичні вирази, що є об'єктом нашого дослідження.

У цій роботі продемонстровано метод, що використовує класифікатор рукописних цифр та узагальнення алгоритму Кока-Янґера-Касамі. Основна відмінність від роботи [1] полягає у використанні згорткової нейронної мережі, що є більш сучасним підходом для класифікації об'єктів на зображенні.

Структура роботи наступна: у першому розділі міститься високорівневий опис методу, що містить три основні етапи, наступні два розділи містять більш детальний опис цих етапів, а в останньому розділі наведено результати експериментів на модельних та реальних даних.

1. Опис програми

1.1. Вхідні дані

На вхід програмі подається одноканальне 8-бітне зображення $\mathbb{I}$ розміру $H \times W$, на якому міститься складання десяткових чисел у стовпчик без знаку «+» та без риски під доданками. Приклад вхідного зображення наведений на рис. 1.

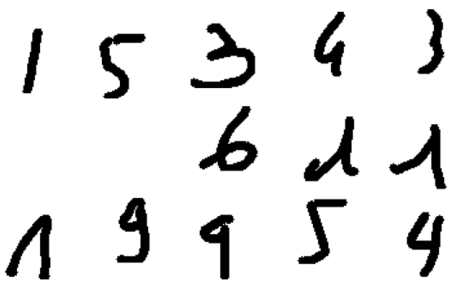


Рис. 1. Приклад вхідного зображення

1.2. Кроки програми

Щоб визначити, чи є коректним вираз на вхідному малюнку, зображення проходить через етапи, що описано у даному підрозділі.

Пошук та розпізнавання цифр. На першому кроці виконується пошук цифр. Зображенням проходиться ковзне вікно, вміст якого подається на вхід класифікатору. Результат класифікації ставиться у відповідність поточному ковзному вікну.

Розпізнавання дробового виразу. Наступним етапом є структурний аналіз зображення, який проводиться узагальненим алгоритмом Кока-Янґера-Касамі для роботи зі зваженими двовимірними контекстно-вільними граматиками, результатом якого стане чисельний показник некоректності виразу на зображенні.

Корекція вхідного зображення. Корекцією ми називаємо таку мінімально можливу заміну цифр на малюнку, щоб вираз став коректним. Зображення цифр, на які будуть замінятися вхідні, беруться з деякого набору еталонів.

[а] pavpel-ipt22@lll.kpi.ua
[б] valeriy.krygin@gmail.com

Якщо на зображені розпізнано коректний вираз, воно не модифікується. Якщо ж помічена помилка, алгоритм корегує зображення та віддає його користувачу програми.

2. Пошук та розпізнавання цифр

2.1. Прохід по зображенню ковзним вікном

Як уже було зазначено, пошук цифр на зображенні відбувається за допомогою ковзного вікна. Вміст кожної позиції вікна проходить через класифікатор, який для кожної цифри вказує ступінь упевненості у тому, що в поточній позиції знаходиться певна цифра. На цьому етапі потрібно обрати

- розмір ковзного вікна,
- розмір горизонтального кроку,
- розмір вертикального кроку.

Зазначимо, що проходів може бути декілька і, можливо, з різними параметрами, бо цифри на зображенні можуть мати різні розміри, що потребує різних розмірів ковзного вікна.

2.2. Класифікація вмісту ковзного вікна

Кожному ковзному вікну для кожного можливого класу треба надати чисельну характеристику схожості вмісту на відповідну цифру або щось інше. Для цього було використано класифікатор, який повертає числа з проміжку $[0, 1]$ для кожної цифри від 0 до 9 і для відсутності цифри. У якості останнього класу виступають підзображення, які містять або фон, або тільки частину цифри, що не може бути віднесено до класів цифр. Останній класс будемо позначати символом ε. Сума характеристик схожості для усіх цифр та відсутності цифри у кожному віконці дорівнює одиниці. Надалі цю характеристику будемо скорочено називати ймовірністю. У якості класифікатору було вирішено використовувати згорткові нейронні мережі (англ. Convolutional Neural Networks, CNN).

2.3. Підготування даних

Для тренування нейронної мережі були вибрані зображення цифр з набору математичних виразів CROHME [2]. Для доповнення (англ. augmentation) набору були взяті наступні послідовно застосовані перетворення до зображень з оригінального набору:

Масштабування. Розмір вхідного символу випадковим чином масштабується з коефіцієнтом масштабування у діапазоні $[0.66, 1]$.

Зсув. Відмасштабований вхідний символ зсувається випадковим чином по вертикалі та горизонталі необмеженим чином, тобто можливі ситуації, коли частина символу або навіть він увесь виходить за межі зображення.

Зашумлення. До зображення випадковим чином додається на вибір шум Ґаусса або шум «солі та перцю» (англ. salt and pepper noise) різних інтенсивностей.

Перше перетворення обумовлено тим, що на реальному зображенні можуть зустрічатися цифри різних розмірів. Друге перетворення створює не тільки такі навчальні приклади, у яких цифри відцентровані на зображенні, але й такі, на яких присутні тільки частини від цифр, які у даній задачі мають клас ε, тобто не класифіковані як цифри. Третє перетворення має створити наближення до реальних умов, коли на зображенні присутній шум різного походження.

Отримані на попередньому етапі зображення нормуються за допомогою перетворень, що описано нижче.

Нормалізація. Значення у кожному пікселі нормалізуються таким чином, щоб усі вони лежали у діапазоні $[0, 1]$.

Інверсія. Значення у кожному пікселі змінюється відповідно до формули

$$p_{new} = 1 - p_{old},$$

де p_{new} — нове значення у пікселі, p_{old} — старе значення.

2.4. Архітектура згорткової нейромережі

Штучна нейромережа побудована з декількох складних блоків, кожний з яких складається з

1) шару пакетної нормалізації (англ. Batch Normalization); у [3] було показано, що додавання цього шару суттєво прискорює навчання моделі та підвищує її точність;
2) шару згортки;
3) активація ReLU.

Два таких складних блоки ідуть підряд, маючи різні параметри для згорткового шару, а після кожної пари іде шар Max Pooling з розміром вікна 2×2 та кроком 2. Після декількох таких послідовностей іде повнозв'язний шар з функцією *softmax* в кінці. Архітектура CNN наведена на рис. 2.

На вхід мережа приймає одноканальні зображення розміру 50×50, причому значення у кожному пікселі лежать у діапазоні $[0, 1]$.

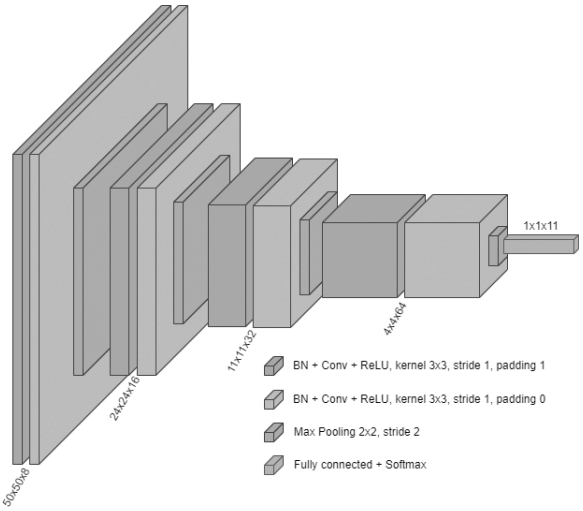


Рис. 2. Архітектура обраної згорткової нейромережі

2.5. Обробка отриманих результатів

Результати роботи класифікатора для кожного ковзного вікна зберігаються у матриці $P(x_1, y_1, x_2, y_2, \ell)$, де

- (x_1, y_1) — координати верхнього лівого кута ковзного вікна;
- (x_2, y_2) — координати нижнього правого кута ковзного вікна;
- ℓ — клас, для якого зберігається значення у матриці, $\ell \in \{0, ..., 9, \varepsilon\}$.

Надалі буде використовуватись скорочений запис $\boldsymbol{p}_i = (x_i, y_i)$, де i залежатиме від контексту. Для кожного ковзного вікна $w(\boldsymbol{p}_1, \boldsymbol{p}_2)$ у клітинці $P(\boldsymbol{p}_1, \boldsymbol{p}_2, \ell)$ зберігається $1 - P_\ell(w)$, де $P_\ell(w)$ — ймовірність того, що вікно w містить об'єкт класу ℓ. Інші значення матриці заповнюються значеннями $+\infty$.

3. Розпізнавання виразу

Подальший етап проводить структурний аналіз зображення та надає певну оцінку коректності зображення.

3.1. Узагальнений алгоритм Кока-Янґера-Касамі

Алгоритм Кока-Янґера-Касамі (англ. Cocke-Younger-Kasami algorithm, CYK algorithm) — це алгоритм, який дозволяє встановити, чи можна у мові, породженою даною контекстно-вільною граматикою, вивести задане слово. Якщо так, можна також вивести спосіб отримання цього слова з набору правил граматики. Цей алгоритм було відкрито незалежно різними вченими у 60-х роках минулого століття, за що він і отримав таку назву.

Наведений для одновимірних контекстно-вільних граматик алгоритм природнім чином узагальнюється і для роботи з двовимірними, а також до більших розмірностей. Ідея узагальнення одновимірного алгоритму на двовимірний випадок була наведена у [4], де також наведена його складність — $\mathcal{O}(m^2n^2(m+n))$. Окрім узагальненя на більші розмірності граматик він також може бути модифікований для роботи зі зваженими граматиками, де правила не бінарні, а вказують ступінь можливості конкатенації символів цими правилами для отримання нетермінальних символів.

Нехай $X = \{1, ..., W\} \times \{1, ..., H\} \times \{1, ..., W\} \times \{1, ..., H\}$, а $G = \langle T, N, g^V, g^H, g, P\rangle$ — граматика, де T — множина термінальних символів, N — множина нетермінальних символів, $g^V : N^3 \to \{0, 1\}$ — правила вертикальної конкатенації, $g^H : N^3 \to \{0, 1\}$ — правила горизонтальної конкатенації, $g : N \times T \to \{0, 1\}$ — правила перейменування нетермінальних символів на термінальні, $P : X \times T \to \mathbb{R}$ — матриця ваг термінальних символів. Модифікація алгоритму для зважених двовимірних контекстно-вільних граматик наведена у алг. 1.

Algorithm 1 Узагальнений алгоритм Кока-Янґера-Касамі

Введемо допоміжні функції. Функція розрахунку штрафу за горизонтальне склеювання двох частин зображення

$$\begin{aligned} &S(\boldsymbol{p}_L, \boldsymbol{p}_R, n_U, n_D, y) \\ &= \mathbb{1}(g^H(n_U, n_D, n) = 0) \cdot \infty \\ &+P(\boldsymbol{p}_L, x_R, y, n_U) + P(x_L, y + 1, \boldsymbol{p}_R, n_D). \end{aligned} \tag{1}$$

Функція розрахунку штрафу найкращого правила горизонтального склеювання двох частин зображення

$$H(\boldsymbol{p}_L, \boldsymbol{p}_R) = \min_{\substack{n_U \in N \\ n_D \in N \\ y_L \le y < y_R}} S(\boldsymbol{p}_L, \boldsymbol{p}_R, n_U, n_D, y).$$

Функція розрахунку штрафу за вертикальне склеювання двох частин зображення

$$\begin{aligned} &S^V(\boldsymbol{p}_L, \boldsymbol{p}_R, n_L, n_R, y) \\ &= \mathbb{1}(g^V(n_L, n_R, n) = 0) \cdot \infty \\ &+P(\boldsymbol{p}_L, x_R, y, n_L) + P(x_L, y + 1, \boldsymbol{p}_R, n_R). \end{aligned} \tag{2}$$

Функція розрахунку штрафу найкращого правила вертикального склеючання двох частин зображення

$$V(\boldsymbol{p}_L, \boldsymbol{p}_R) = \min_{\substack{n_L \in N \\ n_R \in N \\ x_L \le x < x_R}} S^V(\boldsymbol{p}_L, \boldsymbol{p}_R, n_L, n_R, y).$$

Перейдемо до описання самого алгоритму.

Вхід: Граматика $G = \langle T, N, g^V, g^H, g, P\rangle$ з обчисленими вагами для термінальних символів.
Результат: Матриця P' — обчислені ваги для кожного підзображення і зображення в цілому.

for $t \in T$ **do**
  for $x_L = \overline{1, W-1}$, $y_L = \overline{1, H-1}$ **do**
    for $x_R = \overline{x_L + 1, W}$, $y_R = \overline{y_L + 1, H}$ **do**
      $P'(\boldsymbol{p}_L, \boldsymbol{p}_R, t) = P(\boldsymbol{p}_L, \boldsymbol{p}_R, t)$
    end for
  end for
end for
for $s = \overline{1, H \cdot W}$ **do**
  for $x_L = \overline{1, W}, y_L = \overline{1, H}$ **do**
    for $\boldsymbol{p}_R : (y_R + 1 - y_L) \cdot (x_R + 1 - x_L) = s$ **do**
      for $n \in T \cup N$ **do**
        $h \leftarrow H(\boldsymbol{p}_L, \boldsymbol{p}_R, n)$
        $v \leftarrow V(\boldsymbol{p}_L, \boldsymbol{p}_R, n)$
        $r \leftarrow \min_{t \in T}(P'(\boldsymbol{p}_L, \boldsymbol{p}_R, t) + \mathbb{1}(g(t, n) = 1) \cdot \infty)$
        $P'(\boldsymbol{p}_L, \boldsymbol{p}_R, n) \leftarrow \min(P'(\boldsymbol{p}_L, \boldsymbol{p}_R, n), h, v, r)$
      end for
    end for
  end for
end for
  return P'

3.2. Етап корекції

Після того, як отримали матрицю P', яка містить вагу кожного символа, а також для кожного підзображення та зображення в цілому, запускаємо алг. 2.

Algorithm 2 Алгоритм корекції зображення

Нехай $I \in N$ — нетермінальний символ, який відповідає коректному зображенню, $\mathbb{E}$ — множина еталонних зображень. Даний алгоритм використовує функції (1) та (2). Введемо функцію пошуку накращого горизонтального розбиття підзображення, на якому міститься символ n,

$$\hat{H}(\boldsymbol{p}_L, \boldsymbol{p}_R, n) = \operatorname*{argmin}_{\substack{n_U \in N \\ n_D \in N \\ y_L \le y < y_R}} S(\boldsymbol{p}_L, \boldsymbol{p}_R, n_U, n_D, y),$$

та функцію пошуку найкращого вертикального розбиття підзображення, на якому міститься символ n,

$$\hat{V}(\boldsymbol{p}_L, \boldsymbol{p}_R, n) = \operatorname*{argmin}_{\substack{n_L \in N \\ n_R \in N \\ x_L \le x < x_R}} S^V(\boldsymbol{p}_L, \boldsymbol{p}_R, n_L, n_R, y).$$

Перейдемо до описання самого алгоритму.

Вхід: Матриця P', зображення $\mathbb{I}$.
Результат: Зображення з коректним виразом, яке якомога більше схоже на $\mathbb{I}$.
$\boldsymbol{p}_L \leftarrow (1, 1)$
$\boldsymbol{p}_R \leftarrow (W, H)$
$s \leftarrow I$
$w \leftarrow P'(\boldsymbol{p}_L, \boldsymbol{p}_R, s)$
for $s \in N$ **do**
  $n_L, n_R, y \leftarrow \hat{H}(\boldsymbol{p}_L, \boldsymbol{p}_R, n)$
  $n_U, n_D, x \leftarrow \hat{V}(\boldsymbol{p}_L, \boldsymbol{p}_R, n)$
  $n_R \leftarrow \operatorname*{argmin}_{t \in T}(P'(\boldsymbol{p}_L, \boldsymbol{p}_R, t) + \mathbb{1}(g(t, n) = 1) \cdot \infty)$
  $D_0 \leftarrow \{(\boldsymbol{p}_L, \boldsymbol{p}_R, n_R)\}$
  $D_x \leftarrow \{(\boldsymbol{p}_L, x, y_R, n_L), (\boldsymbol{p}_L, x + 1, y_R, n_R)\}$
  $D_y \leftarrow \{(x_L, y, \boldsymbol{p}_R, n_U), (x_L, y, \boldsymbol{p}_R, n_D)\}$
  $D \leftarrow D_0 \cup D_x \cup D_y$
  $\boldsymbol{p}_L, \boldsymbol{p}_R, s \leftarrow \operatorname*{argmax}_{(\boldsymbol{p}'_L, \boldsymbol{p}'_R, s') \in D} P'(\boldsymbol{p}'_L, \boldsymbol{p}'_R, s')$
end for
if $\min_{t \in T} P'(\boldsymbol{p}_L, \boldsymbol{p}_R, t) \neq P'(\boldsymbol{p}_L, \boldsymbol{p}_R, s)$ **then**
  Підзображення в $(\boldsymbol{p}_L, \boldsymbol{p}_R)$ замінити на еталонне з $\mathbb{E}$, відповідне до s, на зображенні $\mathbb{I}$.
end if
  return $\mathbb{I}$

3.3. Набір правил складання у стовпчик

Для того, щоб можна було застосувати алгоритм Кока-Янґера-Касамі для аналізу зображення, треба побудувати граматику, з якою він буде працювати. Умовно правила можна розділити на три частини: розпізнавання цифр, розпізнавання стовпчиків та їх конкатенацію.

Знайти розташування цифр на зображенні недостатньо. Через те, що вони «стрибають» на зображенні, їх треба конкатенувати з частинами фону

$$\begin{aligned} 1 &\to \varepsilon|1 \quad \dots \quad 9 \to \varepsilon|9, \\ 1 &\to 1|\varepsilon \quad \dots \quad 9 \to 9|\varepsilon, \\ 1 &\to \frac{1}{\varepsilon} \quad \dots \quad 9 \to \frac{9}{\varepsilon}, \\ 1 &\to \frac{\varepsilon}{1} \quad \dots \quad 9 \to \frac{\varepsilon}{9}. \end{aligned}$$

Опишемо правила для двох цифр, які стоять в одному стовчику. Нам потрібно знати суму цих цифр за модулем 10, а також знати, чи ця сума є більшою за 9

$$\begin{aligned} A_{(m+n) \mod 10, (m+n) \operatorname{div} 10} &\to \frac{m}{n}, \ m, n = \overline{0, 9}, \\ A_{(m+n) \mod 10, (m+n) \operatorname{div} 10} &\to \frac{n}{m}, \ m, n = \overline{0, 9}. \end{aligned}$$

Важливо нагадати, що зазвичай у складанні в стовпчик приймають участь два числа необов'язково однакової довжини. У таких випадках перед коротшим числом можна поставити стільки нулів, скільки потрібно для компенсації його «недостатньої» довжини, але замість них на зображенні маємо фон

$$\begin{aligned} A_{m,0} &\to \frac{m}{\varepsilon}, \ m = \overline{0, 9}, \\ A_{m,0} &\to \frac{\varepsilon}{m}, \ m = \overline{0, 9}. \end{aligned}$$

Стовпчики можна поділяти за комбінацією двох ознак: породження зайвого десятку та поглинання зайвого десятку. Ці типи будемо позначати символом $V_{i,j}$, $i, j \in 0, 1$, де перший індекс дорівнює одиниці, якщо стовпчик поглинає десяток, та нулю у іншому випадку. Аналогічно, одиниця або нуль для другого індексу, якщо стовпчик генерує або не генерує десяток, відповідно. Опишемо ці правила формулами

$$\begin{aligned} V_{0,\ell} &\to \frac{A_{\ell,k}}{k}, \ k = \overline{0, 9}, \\ V_{1,\ell} &\to \frac{A_{\ell,k}}{k+1}, \ k = \overline{1, 9}, \\ V_{1,1} &\to \frac{A_{0,9}}{0}. \end{aligned}$$

Правила конкатенації самих стовпчиків виглядають наступним чином:

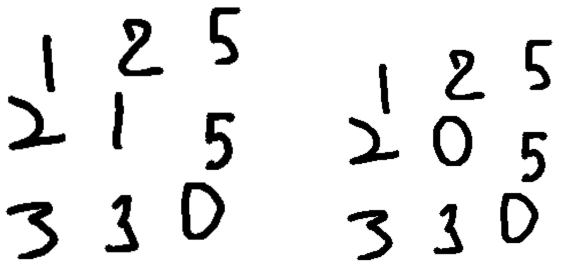
$$\begin{aligned} I &\to V_{0,0}, \\ I &\to V_{0,0}|V_{0,0}, \\ I &\to V_{1,0}|V_{0,1}, \\ I &\to I|V_{0,0}, \\ I &\to I_0|V_{0,1}, \\ I_0 &\to V_{1,0}, \\ I_0 &\to I|V_{1,0}, \\ I_0 &\to I_0|V_{1,1}. \end{aligned}$$

Символ I відповідає коректному виразу, I_0 — конкатенація стовпчиків, за якої поглинається десяток.

4. Опис результатів

У цьому підрозділі представлено результати експериментів, що було проведено за допомогою програми, яка реалізує описаний метод.

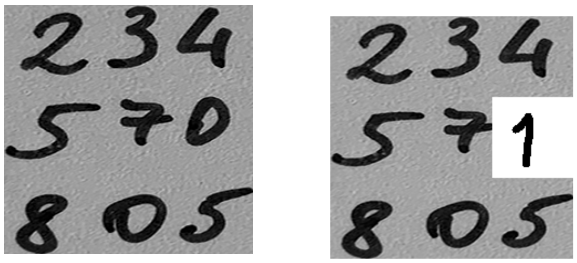
Спершу розглянемо тривіальний випадок, коли на вхідному зображенні представлено коректний вираз (рис. 1). Для символу I (коректний вираз) програма надає ступінь несхожості приблизно 0.05, що, враховуючи похибку у розпізнаванні цифр від класифікатора, інтерпретується як коректне зображення. Після етапу корекції було отримано те ж саме зображення, що підтвержує те, що на зображенні дійсно коректний вираз.



(а) Вхідне зображення, на якому наявна помилка у числовому виразі

(б) Відкориговане вхідне зображення

Рис. 3. Синтетичний приклад некоректного виразу



(а) Вхідне зображення

(б) Відкориговане зображення

Рис. 4. Реальне зображення некоректного виразу

Розглянемо випадок, коли на зображені є одна помилка (рис. 3(а)). Цього разу символ I має ступінь несхожості біля 1.01, що вже вказує на наявність помилки. Після етапу корекції на зображені була замінена одиниця, що знаходиться у центрі (рис. 3(б)), що є однією з можливих корекцій зображення.

На рис. 4(а) представлено реальне зображення з некоректним виразом, яке було також подано на вхід програмі. На виході було отримано відкориговане зображення (рис. 4(б)).

Висновки

У даній роботі розглянуто сумісне використання згорткової нейромережі та методу структурного аналізу зображень. На прикладі зображень складання у стовпчик продемонстрована ефективність сукупності цих методів. Слід зазначити, що цей підхід не обмежується розпізнаванням таких зображень, і його можна використовувати для аналізу інших структурованих зображень, якщо відповідним чином замінити класифікатор і граматику. Сучасні обчислювальні можливості відкривають нові перспективи використання подібних методів.

Перелік використаних джерел

1. Průša D., Hlavac V. 2D context-free grammars: Mathematical formulae recognition // Proceedings of the Prague Stringology Conference. — 2006. — 01. — P. 77–89.
2. ICDAR 2019 CROHME + TFD: Competition on Recognition of Handwritten Mathematical Expressions and Typeset Formula Detection / Mahdavi M., Zanibbi R., Mouchere H., Viard-Gaudin C., and Garain U. // 2019 International Conference on Document Analysis and Recognition (ICDAR). — 2019. — sep. — P. 1533–1538.
3. Ioffe S., Szegedy C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. — 2015. — Vol. 37. — P. 448–456. — Access mode: https://proceedings.mlr.press/v37/ioffe15.html.
4. Шлезингер М. И., Главач В. Десять лекций по статистическому и структурному распознаванию образов. — Наукова думка, 2004. — 545 с.