УДК 004.855.5
DOI 10.18372/2786-5487.1.15824

Балицька Ірина Андріївна
Python джуніор-розробник,
Adozi Inc., США, Маямі

НЕЙРОМЕРЕЖЕВИЙ ДОДАТОК РОЗПІЗНАВАННЯ ЗНАКІВ ДАКТИЛЬНОГО АЛФАВІТУ

Анотація. *У статті розглядається процес створення додатку для вирішення проблеми розпізнавання жестів рук – знаків алфавіту глухонімих. Використано штучні нейронні мережі та функції комп'ютерного зору.*

Ключові слова: *штучна нейронна мережа, згорткова нейронна мережа, дактильний алфавіт, PyTorch.*

Annotation. *The aspects of developing an application for solving the problem of recognition the hand gestures – deaf-mute alphabet signs – is considered. Artificial neural networks and computer vision are utilized.*

Key words: *artificial neural network, convolutional neural network, dactyl alphabet, PyTorch.*

З розвитком VR-технологій та методів взаємодії «людина-комп'ютер», задача розпізнавання жестів почала набирати популярність, про що свідчить велика кількість робіт в цій області, наприклад [1], [2].

Розпізнавання мови жестів може допомогти людям з вадами слуху або мови вирішити проблеми комунікації. Глухонімі люди використовують два види знакових систем – дактильний алфавіт і жестову мову. Дактильний алфавіт – це система положень або жестів рук, що відповідають літерам. Наприклад, рука, стиснута в кулак, позначає букву «а», а долоня з випрямленими стислими пальцями і відставленим убік великим – «в» [3].

Для розробки додатку була обрана мова програмування Python 3 та фреймворк глибинного навчання PyTorch.

В першу чергу був зібраний набір даних, що складається з 29 класів зображень. В набір не були включені літери «ґ», «ї», «й», «щ», оскільки вони майже нічим не відрізняються від літер «г», «і», «и», «ш» відповідно, окрім руху руки, який неможливо відобразити на одиночному фотознімку. Для того, щоб нейронна мережа змогла ефективно навчатися, необхідна велика кількість тренувальних даних, тому для кожного класу було зібрано по 25 зображень – усього 725.

Далі зібрані дані обробляються: змінюється розмір – бажано, щоб зображення мали однакові виміри; дані нормалізуються – значення входів зводяться до деякого обмеженого діапазону (наприклад, [0 ... 1] або [–1 ... 1]), що дозволяє моделям працювати з ними більш коректно; також зображення були перетворені в чорно-білі (а саме у відтінки сірого) – це потрібно для того, щоб можна було працювати з одним колірним каналом замість трьох.

Наведені вище дії можна виконати за допомогою PyTorch. Окрім них, до зображень були застосовані інші перетворення: випадкове віддзеркалення по вертикалі або горизонталі та поворот на випадковий градус. Це допомагає урізноманітнити маленький набір даних.

Після створення та налагодження тренувальних даних, настає черга розробки нейронної мережі. Для задач, пов'язаних із розпізнаванням зображень, підходять згорткові нейромережі. Мережа буде складатися з трьох згорткових шарів, трьох агрегувальних шарів та двох повноз'єднаних шарів.

Шар згортки – це основний шар згорткової нейронної мережі. Він містить в собі фільтр для кожного каналу, ядро згортки якого обробляє попередній шар за фрагментами (підсумовуючи результати матричного добутку для кожного фрагмента). Вагові коефіцієнти

ядра згортки невідомі і встановлюються в процесі навчання. Особливістю згорткового шару є порівняно невелика кількість параметрів, які встановлюються при навчанні [4, с. 43]. В PyTorch на вхід згорткового шару подаються такі параметри: кількість каналів зображення (в цьому випадку 1), кількість виходів із шару та ядро (розмір вікна згортки).

Агрегувальний (або пулінговий) шар – це нелінійне ущільнення карти ознак, при цьому група пікселів (зазвичай розміру 2×2) ущільнюється до одного пікселя, проходячи нелінійне перетворення. Перетворення зачіпають непересічні прямокутники або квадрати, кожен з яких скорочується в один піксель, при цьому вибирається піксель, що має максимальне значення. Пулінг дозволяє істотно зменшити просторовий обсяг зображення. Якщо на попередній операції згортки вже були виявлені деякі ознаки, то для подальшої обробки настільки детальне зображення вже не потрібно, і воно зменшується до менш детального. Фільтрація вже непотрібних деталей допомагає не перенавчатися. Пулінговий шар, як правило, вставляється після шару згортки перед шаром наступної згортки [4, с. 45].

На вхід нейронної мережі надходило зображення розміром 60×60. Для того, щоб визначити, яким буде розмір зображення після проходження згорточного шару, використовується формула:

деW_{out}– розмір вихідного зображення,

W_{in}– розмір вхідного зображення,

F– розмір ядра,

P– вирівнювання (за замовчуванням дорівнює 0),

S– крок (за замовчуванням дорівнює 1). [5]

Отже, якщо розмір зображення – 60×60, а ядра – 5, то розмір вихідного зображення:

$$W_{out} = \frac{60 - 5 + 2 \cdot 0}{1} + 1 = 56.$$

Після проходження агрегувального шару, зображення ділиться на розмір вікна згортки:

$$W_{out} = \frac{56}{2} = 28.$$

Після проходження згорткових та пулінгових шарів, зображення поступає на вхід повноз’єднаного шару. Для того, щоб визначити кількість вхідних нейронів шару, кількість вихідних нейронів попереднього шару множиться на добуток висоти і ширини зображення. Вихідний шар має містити стільки виходів, скільки класів в наборі даних, тобто 29.

В даній архітектурі використовується функція активації ReLU (Rectified Linear Unit). Функція повертає 0, якщо приймає від’ємний аргумент, в разі ж додатного аргументу, функція повертає саме число. ReLU може бути записана наступним чином [5]:

Функція активації ReLU застосовується до кожного шару, окрім вихідного. Для вихідного шару найкраще підходить функція Softmax. Функція перетворює векторxрозмірностіKв векторσтієї ж розмірності, де кожна координатаσ_iотриманого вектора представлена дійсним числом в інтервалі [0,1] і сума координат дорівнює 1. Softmax може бути представлена наступним чином [5]:

$$\sigma(x_j) = \frac{e^{(x_j)}}{\sum e^{(x_i)}}. \quad (3)$$

Після розробки мережі слід перейти до її навчання та тестування.

Для оптимізації обирається функція Adam зі швидкістю навчання 0,001. Функція Cross Entropy Loss (перехресна ентропія) застосовується для підрахунку втрат.

В тренуванні моделі застосовується метод зворотнього поширення помилки – метод обчислення градієнта, який використовується при оновленні ваг багатошарового перцептрона. Після тренування та перевірки модель зберігається.

Тепер на готову модель можна подавати зображення з вебкамери. Для цього використовується модуль OpenCV. Він містить в собі бібліотеки алгоритмів комп’ютерного зору та обробки зображень.

Висновки: у роботі розглянуто додаток, який дозволяє розпізнваати знаки жетової мови глухонімих (дактильного алфавіту). Використання такого додатку дозволить спростити комунікацію з людьми, що мають особливі потреби, зокрема даний додаток може знайти численні застосування в учбовому процесі. Проблема розпізнавання жестів вирішується шляхом створення згорткових нейронних мереж, великої кількості зображень для тренування та комп'ютерного зору.

Питанням подальшого дослідження є розширення та ускладнення роботи додатку. Набір даних може бути доповнений знаками інших мов, наприклад англійської. Також можна реалізувати розпізнавання тих знаків, які містять у собі рух руки, приміром, використовуючи для цього по 2 зображення замість одного.

Список використаних джерел

1. Dhawale P., Masoodian M., Rogers B. Bare-hand 3d gesture input to interactive systems //CHINZ'06: 7th ACM SIGCHI New Zealand Chapter's Conference (International) on Computer–Human Interaction: Design Centered HCI Proceedings. – New York, NY, USA: ACM, 2006. P. 25–32.

2. Aguiar R., Pereira J. M., Braz J. Gadevi – game development integrating tracking and visualization devices into virtools //GRAPP 2009: 4th Conference (International) on Computer Graphics Theory and Applications Proceedings. – INSTICC Press, 2009. P. 313–321.

3. Зайцева Г.Л. Жестовая речь. Дактилология: Учеб. для студ. высш. учеб. заведений. – М.: Гуманит. изд. центр ВЛАДОС, 2000. – 192 с

4. Гафаров Ф.М Искусственные нейронные сети и приложения: учеб. пособие / Ф.М. Гафаров, А.Ф. Галимянов. – Казань: Изд-во Казан. ун-та, 2018. – 121 с.

5. CS231n: Convolutional Neural Networks for Visual Recognition [Електронний ресурс] – Режим доступу: https://cs231n.github.io/convolutional-networks/ – Назва з екрану.