УДК 004.6

РОЗПІЗНАВАННЯ РУКОПИСНИХ СИМВОЛІВ ЗА ДОПОМОГОЮ JAVASCRIPT

Галушка А.С., Акбаш К.С.

Центральноукраїнський державний педагогічний університет імені Володимира Винниченка, м. Кропивницький, Україна

Розпізнавання рукописного введення уже давно використовується у портативних електронних пристроях. Станом на сьогодні для цього найчастіше використовуються рекурентні нейронні мережі та аналітичні алгоритми із сімейства $. У статті ми описуємо способи розпізнавання рукописних символів на основі приведення до ламаних ліній та до кривих Без'є. Описаний алгоритм дозволяє значно зменшити частку хибних спрацювань, що вигідно для використання у ігрових проектах.

Ключові слова: розпізнавання рукописного введення, нейронні мережі, криві Без'є.

Handwriting recognition using JavaScript

The Volodymyr Vynnychenko Central Ukrainian State Pedagogical University, Kropyvnytsky, Ukraine

Handwriting recognition has long been using in portable electronic devices. As of today, recurrent neural networks and analytical algorithms from the $ family are most often used for this purpose. In the article, we describe ways of recognizing handwritten characters based on bringing to broken lines or Bezier curves. The described algorithm allows to significantly reducing the proportion of false positives, which is beneficial for using in gaming projects.

Key words: handwriting recognition, neural networks, Bezier curves.

Постановка проблеми. Ще на початку 1980-х років з'явилися пристрої, у яких використовувалося рукописне введення тексту. Завдяки поширенню сенсорних екранів та тенденції відмови від фізичних елементів керування у портативних пристроях все частіше застосовується керування за допомогою рукописних жестів. Ми пропонуємо алгоритм розпізнавання рукописних жестів, адаптований для використання у ігрових проектах.

Аналіз останніх досліджень і публікацій. Станом на сьогодні більшість систем для розпізнавання рукописного тексту базуються на рекурентних нейронних мережах. У роботі [1] пропонується використовувати двосторонню короткотривалу пам'ять та асоціативний темпоральний класифікатор для

налаштування мережі під почерк користувача на льоту. Також пропонується застосовувати дистиляцію знань для того, щоб зменшити кількість необхідних ресурсів для роботи мережі з помірною втратою якості розпізнавання. У роботі [2] описано, як за допомогою рекурентних нейронних мереж та приведення до кривих Без'є реалізовано розпізнавання рукописного тексту у продукті Gboard. У роботі [3] описується метод швидкого навчання мережі з використанням ресурсів GPU. З 2007 року триває розробка систем для розпізнавання з родини $, до яких належать $1, $N, $P, $P+ та $Q. Кожна з цих систем має свої особливості, наприклад $Q, описана в роботі [4], орієнтована на використання у малопотужних портативних пристроях, таких як розумні годинники.

Як бачимо, станом на сьогодні для розпізнавання рукописного тексту використовуються рекурентні нейронні мережі. У випадках, коли необхідно розпізнавати введення «на льоту», особливо на малопотужних пристроях, можуть використовуватися спеціалізовані аналітичні алгоритми.

Постановка завдання. Мета статті полягає у висвітленні розробленого нами алгоритму розпізнавання рукописних жестів за допомогою приведення до примітивних фігур засобами мови програмування JavaScript.

Виклад основного матеріалу. Розпізнавання рукописного введення в основному поділяють на офлайн (робота зі статичним зображенням тексту) та онлайн (захоплення даних про процес введення та аналіз на їх основі). Онлайн-розпізнавання використовується в інтерактивних системах, тоді як офлайн-розпізнавання використовується для розпізнавання тексту, банківських чеків, адреси отримувача на поштових відправленнях, підтвердження підпису.

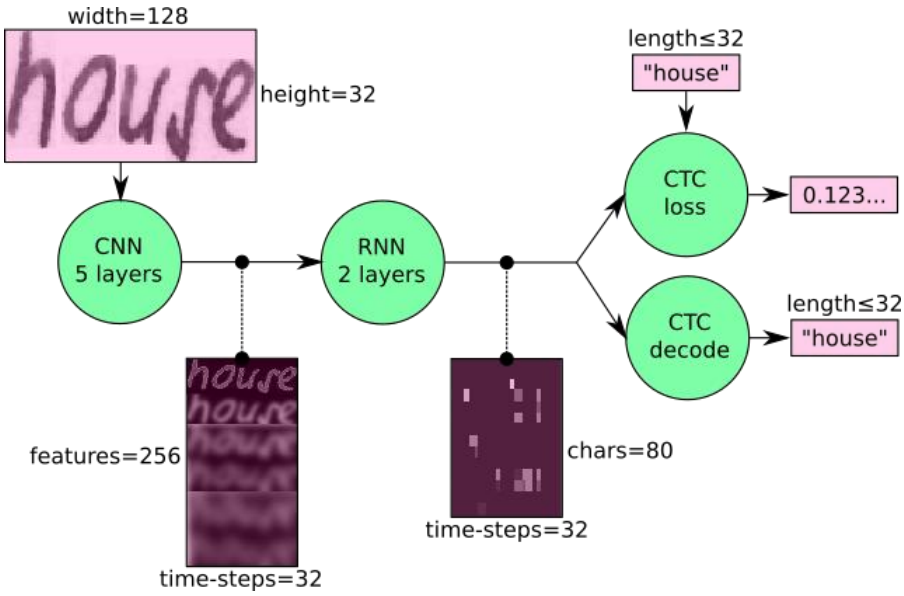

Рис. 1. Принцип роботи рекурентної нейронної мережі

На рис. 1 зображено принцип, за яким рекурентна нейронна мережа розпізнає текст на зображенні.

1. Спочатку згорткова нейронна мережа обробляє зображення, виділяючи відповідні риси та зменшуючи об’єм даних для наступних компонентів.
2. Потім рекурентна нейронна мережа аналізує виділені риси, визначаючи ймовірність того, що символ *x* знаходиться на позиції *i*. Короткотривала пам’ять, реалізована у цій мережі, дозволяє їй ефективно адаптуватися до почерку окремого користувача. Максимальний розмір слова та об’єм алфавіту обмежується розміром матриці рекурентної нейронної мережі.
3. Дані, отримані від рекурентної нейронної мережі, передаються до асоціативного темпорального класифікатора, який відтворює тест та визначає рівень помилки.

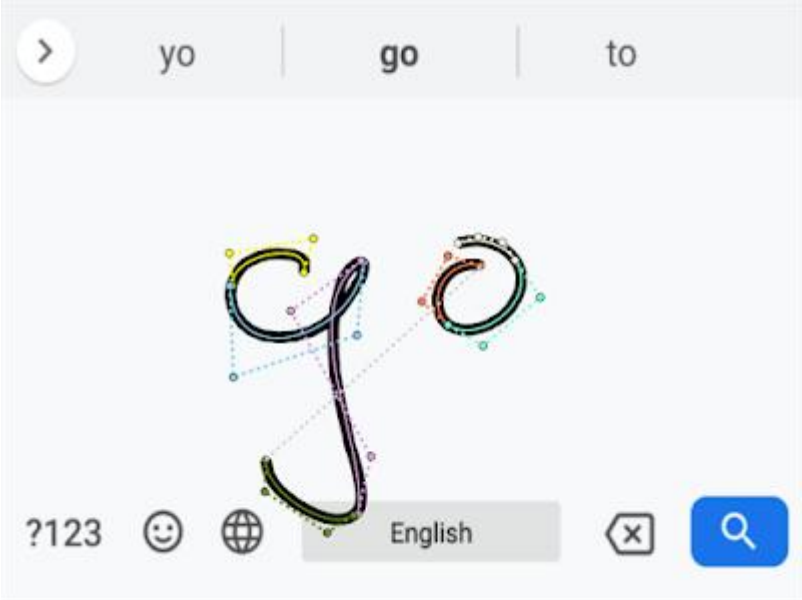

Рис. 2. Спрощення введення у додатку Gboard [2]

Наявність інформації про процес введення тексту дозволяє значно полегшити процес розпізнавання. На рисунку 2 зображено результат нормалізації користувацького введення у додатку Gboard. Спершу шлях вказівника спрощується шляхом видалення точок, які знаходяться занадто близько одна від одної, а потім він перетворюється на послідовність кубічних кривих Без'є. Це послідовність використовується для визначення введених символів.

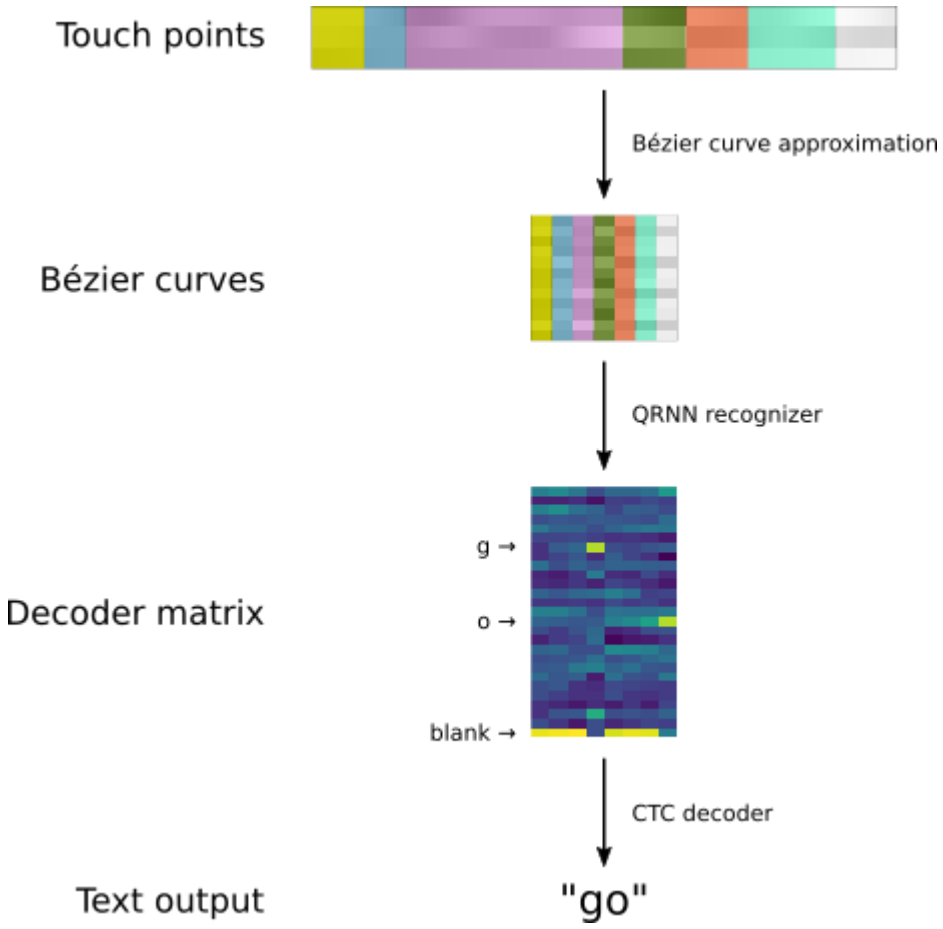

Рис. 3. Процес розпізнавання символів за допомогою кривих Без'є та рекурентної нейронної мережі [2]

Подальший процес розпізнавання схожий на той, що використовується при розпізнаванні тексту на зображенні – квазі-рекурентна нейронна мережа розпізнає символи на основі кривих Без'є, а асоціативний темпоральний класифікатор формує текст результату.

Один із головних недоліків розпізнавання на основі нейронних мереж – повільна робота на малопотужних пристроях. У цьому випадку вигідніше використовувати аналітичні алгоритми.

У 1993 році для роботи на КПК було розроблено систему для розпізнавання введення за допомогою стилусу під назвою Graffiti. На рисунку 4 зображений спеціальний алфавіт, розроблений для цієї системи. Кожен символ цього алфавіту мітить лише одну лінію, завдяки чому здійснювати розпізнавання значно легше.

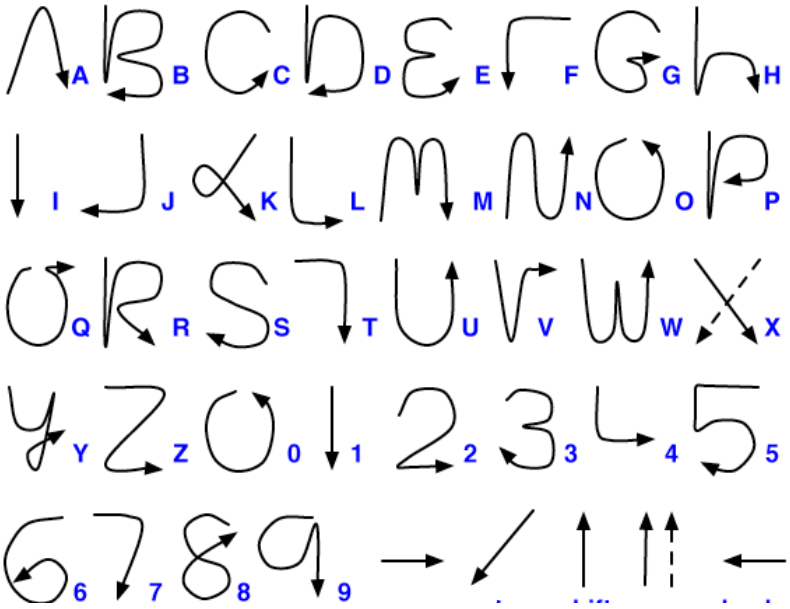

Рис. 4. Алфавіт для системи розпізнавання символів Graffiti [6]

Станом на сьогодні активно використовуються такі алгоритми, як $1, $N та $P [7]. $1 використовує метод найближчого сусіда для співставлення кривої з еталоном. $N працює за схожим принципом, але він допускає, щоб символ складався з кількох кривих. $P, на відміну від попередніх алгоритмів, представляє еталон не як упорядкований набір точок, а як неупорядковану хмару точок, завдяки чому розпізнавання фігури не залежить від кількості кривих, з яких вона складається.

На жаль, ці алгоритми часто розпізнають як правильні символи, які відсутні в алфавіті. Це може бути недопустимо в іграх, коли від гравця вимагається відтворення символу з певною точністю. Тому ми вирішили розробити власний алгоритм, який розпізнаватиме символи з однієї кривої. Він працює у два етапи:

1. Спрощення кривої, отриманої від користувача.
2. Розпізнавання символу на основі спрощеної кривої

Перший варіант спрощення, який ми розглянули – це розкладання кривої на набір прямих ліній. Для зменшення кількості точок кривої ми використали алгоритм Рамера-Дугласа-Пекера, суть якого полягає у тому, щоб на основі точок даної ламаної, яка апроксимує криву, побудувати ламану з меншою кількістю точок [8]. Отриману ламану ми спрощуємо знову, обмежуючи вісі *x*

та *y* значеннями {-1, 0, 1}. Далі ми порівнюємо ламану із шаблонами, щоб визначити, який символ вона зображує. Наші шаблони складаються з груп, кожна з яких може складатися з кількох ліній. Під час порівняння вимагається, щоб у ламаній була присутня хоча б одна лінія з групи. Існують також групи, для яких дозволено відсутність збігів. Не допускається наявність лінії, відсутньої у поточній або наступній групі. Один з таких шаблонів наведено на рисунку 5.

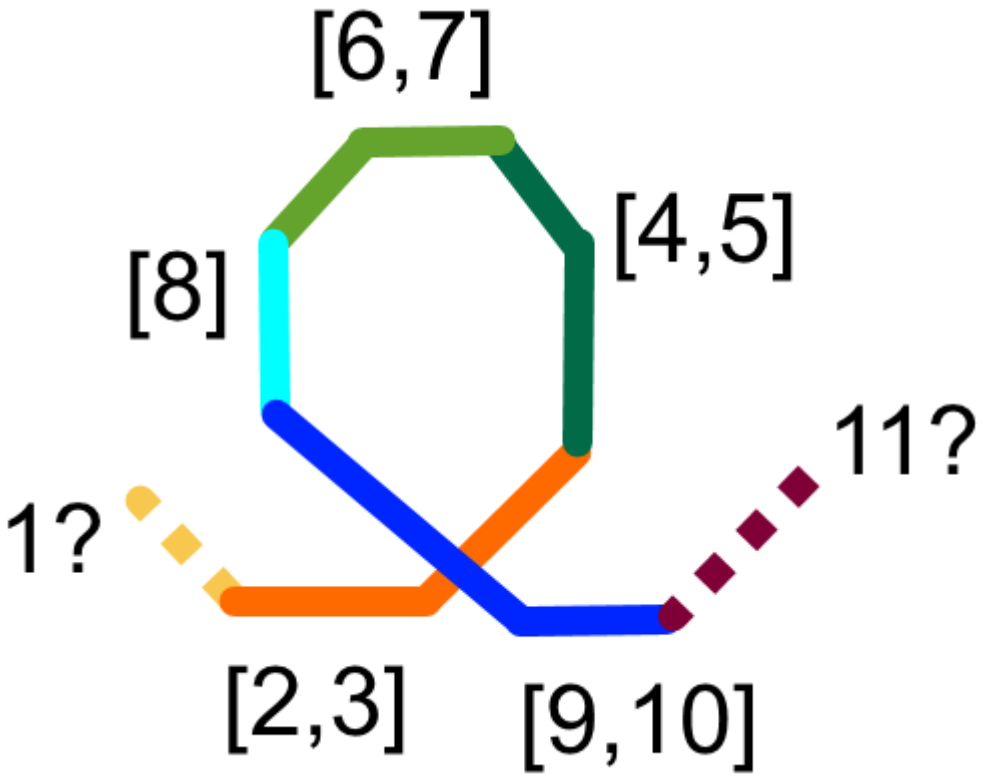

Рис. 5. Шаблон символу з ламаної лінії

Цей підхід добре працює, якщо символ не має округлих ліній. В іншому випадку наший алгоритм спрощення може поглинати важливі елементи символу, значно ускладнюючи процес розпізнавання.

Для того, щоб забезпечити розпізнавання округлих символів, ми замінили алгоритм спрощення на приведення до кубічних кривих Без'є. Для розпізнавання використовуються такі характеристики, як направленість контрольних точок та відношення їхньої віддаленості від відповідних точок кривої до довжини кривої. Приведення до кривих вимагає складніших обчислень, проте дає можливість розпізнавати складніші символи.

Розпізнавання на основі шаблонів дуже рідко допускає хибне спрацювання, через що наший алгоритм вигідно використовувати у ігрових програмах, де від користувача вимагається точність відтворення символу. До недоліків алгоритму можна віднести те, що він обробляє лише ті символи, що складаються з однієї лінії.

Для спрощення та розпізнавання ліній достатньо простих операцій над масивами, тому зручність поширення та незалежність від платформи стали ключовими критеріями при виборі мови програмування. Для реалізації ми обрали Javascript, оскільки ця мова дозволяє застосовувати алгоритм не лише на веб-сторінках, а і у мобільних додатках, сформованих на їх основі.

Висновки. У статті описано спосіб реалізації розпізнавання рукописних жестів за допомогою приведення до примітивних фігур засобами мови програмування JavaScript, розглянуто переваги та недоліки такого підходу. Акцент зроблено на оптимізації якості розпізнавання жестів.

Список літератури

1. Ren G. Recognition of Online Handwriting with Variability on Smart Devices / G. Ren, V. Ganapathy. // ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). – 2019. – pp. 7605–7609.
2. Feuz S. RNN-Based Handwriting Recognition in Gboard [Електронний ресурс] / S. Feuz, P. Gonnet // Google AI Blog. – 2019. – Режим доступу до ресурсу: https://ai.googleblog.com/2019/03/rnn-based-handwriting-recognition-in.html.
3. Doetsch P. Fast and robust training of recurrent neural networks for offline handwriting recognition / P. Doetsch, M. Kozielski, H. Ney. // 2014 14th International Conference on Frontiers in Handwriting Recognition. – 2014. – №14. – pp. 279–284.
4. Vatavu R. $Q: A super-quick, articulation-invariant stroke-gesture recognizer for low-resource devices [Електронний ресурс] / R. Vatavu, L. Anthony, J. Wobbrock // Proceedings of the ACM Conference on Human-Computer Interaction with Mobile Devices and Services. – 2018. – Режим доступу до ресурсу: https://faculty.washington.edu/wobbrock/pubs/mobilehci-18.pdf.
5. Scheidl H. Build a Handwritten Text Recognition System using TensorFlow [Електронний ресурс] / Harald Scheidl // Towards Data Science. – 2018. – Режим доступу до ресурсу: https://towardsdatascience.com/build-a-handwritten-text-recognition-system-using-tensorflow-2326a3487cd5.

6. Goldberg D. Touch-typing with a stylus / D. Goldberg, C. Richardson. // Proceedings of the INTERACT'93 and CHI'93 conference on Human factors in computing systems. – 1993. – pp. 80–87
7. Wobbrock J. Gestures without libraries, toolkits or training: A $1 recognizer for user interface prototypes / J. Wobbrock, A. Wilson, Y. Li. // Proceedings of the ACM Symposium on User Interface Software and Technology (UIST '07). – 2007. – pp. 159–168.
8. Hershberger J. Speeding up the Douglas-Peucker line-simplification algorithm / J. Hershberger, J. Snoeyink. // University of British Columbia, Department of Computer Science. – 1992. – pp. 134–143.