JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 7 (2018) 210-214
Wysłane: 2018-01-11
Przyjęte: 2018-01-15

Efektywność sztucznych sieci neuronowych w rozpoznawaniu znaków pisma odręcznego

Marek Miłosz*, Janusz Gazda
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Sztuczne sieci neuronowe są jednym z narzędzi współczesnych systemów odtwarzania z obrazów tekstów, w tym pisanych odręcznie. W artykule przedstawiono rezultaty eksperymentu obliczeniowego mającego na celu analizę jakości rozpoznawania cyfr pisanych odręcznie przez dwie sztuczne sieci neuronowe (SSN) o różnej architekturze i parametrach. Jako podstawowe kryterium jakości rozpoznawania znaków użyto wskaźnika poprawności. Poza tym analizie poddano liczbę neuronów i ich warstw oraz czas uczenia SSN. Do stworzenia SSN, oprogramowania algorytmów ich uczenia i testowania wykorzystano język Python i bibliotekę TensorFlow. Obydwie SSN uczono i testowano przy pomocy tych samych dużych zbiorów obrazów znaków pisanych odręcznie.

Słowa kluczowe: rozpoznawanie znaków; pismo odręczne; sztuczne sieci neuronowe

*Autor do korespondencji.

Adres e-mail: m.milosz@pollub.pl

Effectiveness of artificial neural networks in recognising handwriting characters

Marek Miłosz*, Janusz Gazda
Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. Artificial neural networks are one of the tools of modern text recognising systems from images, including handwritten ones. The article presents the results of a computational experiment aimed at analyzing the quality of recognition of handwritten digits by two artificial neural networks (ANNs) with different architecture and parameters. The correctness indicator was used as the basic criterion for the quality of character recognition. In addition, the number of neurons and their layers and the ANNs learning time were analyzed. The Python language and the TensorFlow library were used to create the ANNs, and software for their learning and testing. Both ANNs were learned and tested using the same big sets of images of handwritten characters.

Keywords: character recognition; handwriting; artificial neural networks

*Corresponding author.

E-mail address: m.milosz@pollub.pl

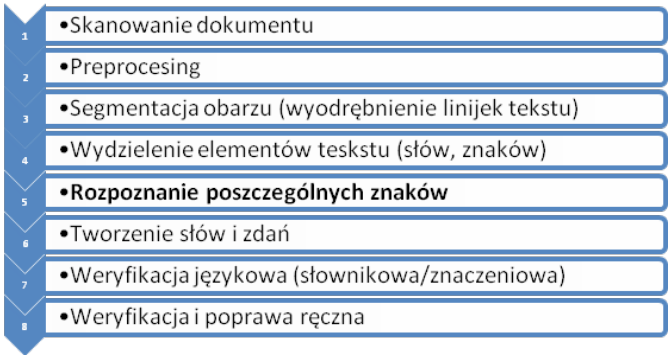
1. Wstęp

Optyczne rozpoznawanie znaków (ang. *Optical Character Recognition, OCR*) polega na przetworzeniu obrazu rastrowego tekstu w zakodowany cyfrowo tekst. Proces ten jest jednym z zadań cyfryzacji zbiorów dokumentów, w tym pisanych odręcznie – wówczas jest to inteligentne rozpoznawania tekstów (ang. *Intelligent Character Recognition, ICR*). Potrzeba takiego przekształcenia zeskanowanego automatycznie obrazu dokumentu w tekst występuje przede wszystkim w bibliotekach, w których zgromadzono olbrzymie zasoby zbiorów drukowanych i tworzonych odręcznie, oraz w muzeach. Oprócz tego potrzeby takie występują w obszarze biznesu i e-państwa. Systemy OCR są nieodzownymi elementami aplikacji biznesowych obiegu dokumentów, w których jeszcze przez jakiś czas będą pojawiać się dokumenty elektronicznie lub ręcznie tworzone/wypełniane, wydrukowane, podpisane i skanowane. W miarę cyfryzacji gospodarki wolumen dokumentów skanowanych, a szczególnie tych wypełnianych odręcznie, będzie się zmniejszać [1]. Tym niemniej w chwili obecnej w niektórych obszarach e-gospodarki i e-państwa pewna część dokumentów jest wypełniana odręcznie. W związku z czym ICR stanowią się nieodzownymi i alternatywnymi do ręcznego wprowadzania treści elementami systemów klasy ERP (ang. *Enterprise Resource Planning*), G2C (ang. *Government-to-Citizen*) czy też RPA (ang. *Robotic Process Automation*) [2].

2. Typowy schemat rozpoznawania pisma odręcznego

Systemy OCR (a także ICR) realizują cały szereg algorytmów i programów informatycznych w tym (rys. 1):

- skanowanie dokumentów,
- obróbka plików graficznych rastrowych pozyskanych w procesie skanowania i modyfikacji,
- analiza struktury zawartości plików rastrowych,
- rozpoznawanie pojedynczych znaków,
- tworzenie słów (jednostek leksykalnych) i zdań,
- weryfikacja leksykalna,
- weryfikacja ręczna poprawności tekstu i jego poprawa.



Rys. 1. Schemat rozpoznawania pisma odręcznego – stos metod i algorytmów (opracowanie własne na podstawie [3])

Część z tych algorytmów jest dość znana. Przykładowo są to algorytmy segmentacji obrazów [4] lub weryfikacji słownikowej rozpoznanego słowa.

Jednym z istotnym stosie metod, stosowanych w procesie OCR/ICR, jest algorytm rozpoznawania poszczególnych znaków (punkt 5 na rys. 1).

3. Metody rozpoznawania pojedynczych znaków pisma

Algorytmy rozpoznawania (klasyfikacji) poszczególnych znaków przekształcają dane rastrowe na tekstowe. Bazują one na następujących klasyfikatorach [5]:

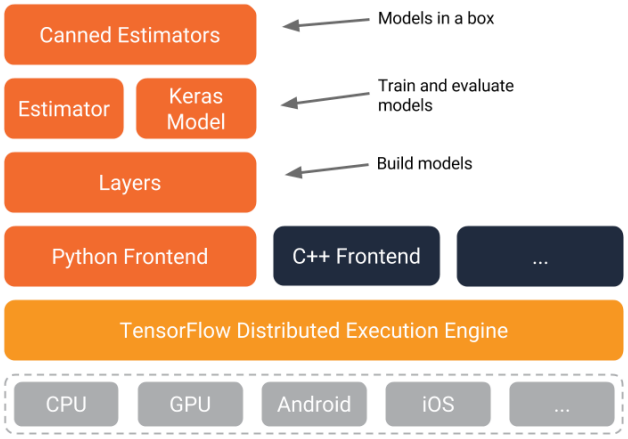
- k-NN – k-najbliższych sasiadów – algorytm centroidów,
- klasyfikator bayesowki,
- NN (ang. *Neural Network*) – sieci neuronowe,
- HMM (ang. *Hidden Markov Model*) – ukryte modele Markowa,
- SVM (ang. *Suport Vector Machine*) – maszyna wektorów wspierających (nośnych).

Wszystkie te metody bazują na podawaniu na wejściu wektora pozyskanego jako rezultat analizy obrazu (etap 4. z rys. 1), np. wartości pixeli obrazu zawierającego znak, i określeniu do której grupy klasyfikacyjnej (tj. do jakiego znaku) „pasuje" dany wektor. Podane powyżej klasyfikatory określają miarę „dopasowania" znaku do grupy, tj. klasyfikację znaku. Proces ten, wykorzystujący nauczoną SSN, przedstawiony został na rys. 2.

Rys. 2. Schemat klasyfikacji znaku przez SSN

4. Architektura platformy TensorFlow

Istnieje cały szereg bibliotek wykorzystujących uczenie maszynowe, w tym SSN. Do najbardziej znanych należą: TensorFlow, Theano, Keras i Scikit-learn [6]. Biblioteka TensorFlow została opracowana w laboratoriach firmy Google, jako biblioteka typu open source, i jest przeznaczona do tworzenia programów wykorzystujących uczenie maszynowe oraz do badania SSN [7]. TensorFlow obecnie jest całą platformą o wielopoziomowej strukturze (rys. 3).



Rys. 3. Wielopoziomowa struktura TensorFlow [7]

TensorFlow wykorzystuje w zasadzie dowolne, dostępne zasoby obliczeniowe, od pojedynczego mikroprocesora (CPU) czy też karty graficznej (GPU), poprzez ich klaster, aż po chmurę obliczeniową, a nawet platformy urządzeń mobilnych [7]. TensorFlow posiada API do typowych języków programowania, od C++, poprzez Python do Javy i Java Script.

TensorFlow jest od paru lat wykorzystywany do rozwiązywania wielu zadań sztucznej inteligencji, takich jak analiza wyrazu twarzy [8] i zachowań ludzkich [9], klasyfikacja danych [10] i obiektów [11] oraz prognozowanie przyszłości [12].

5. Analiza efektywności rozpoznawania przez sieci o różnej architekturze i parametrach

5.1. Opis problemu

Celem badania jest analiza wpływu architektury SNN, nauczonej rozpoznawania znaków (cyfr) pisanych odręcznie, na proces (czas) uczenia sieci oraz na jakość rozpoznawania przy różnych liczbach neuronów w warstwach.

Można sformułować następujące pytanie badawcze:

Czy dodanie drugiej warstwy ukrytej w SSN zwiększy jakość rozpoznawania znaków pisma odręcznego?

By dać odpowiedź na pytanie badawcze należy rozpatrzyć dwie SSN bez sprzężenia zwrotnego (ang. *Feedforward*) różniące się liczbą warstw ukrytych: jedna i dwie. Sieci, proces uczenia oraz testowania implementuje się w języku Python z wykorzystaniem biblioteki TensorFlow. Wszystkie dwa typy sieci należy uczyć i testować przy pomocy tych samych zbiorów znaków. Implementacja eksperymentu obliczeniowego powinna być zrealizowana w identycznym środowisku sprzętowo-programistycznym. Należy także

modyfikować liczbę neuronów w poszczególnych warstwach ukrytych kolejnych eksperymentach obliczeniowych.

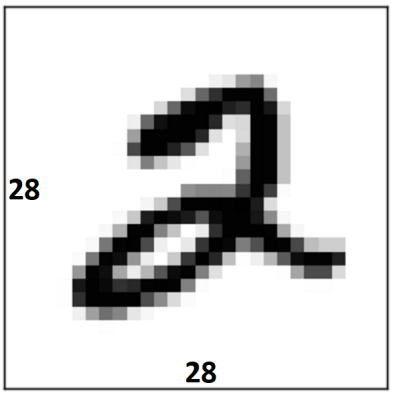
W rezultacie eksperymentu określony będzie wskaźnik poprawności nauczonej sieci (mierzony w % poprawnych rozpoznań znaków w zbiorze testowym) i czas uczenia sieci.

W związku z różną architekturą SSN do porównania obu wariantów wykorzystane zostanie porównanie łącznej (sumarycznej) liczby neuronów we wszystkich warstwach SSN i poprawności rozpoznawania znaków. Porównanie to zostanie dokonane przy wartościach wskaźnika poprawności 90% i 94%.

5.2. Zbiór danych uczących i testujących

W eksperymencie wykorzystano ogólnie dostępny zbiór obrazów cyfr pisanych odręcznie [13]. Zbiór ten składa się z dwóch podzbiorów: uczącego (60 tys. znaków) i testowego (10 tys. znaków). Cyfry pisane odręcznie są umieszczone w postaci obrazów o stałej wielkości i wyśrodkowane w prostokącie obrazu. Oprócz tego na zbiór składają się dwa pliki zawierające etykiety (tekstowe) każdego obrazu.

Obrazy cyfr były poddane operacji czyszczenia z szumów oraz poddane procesowi normalizacji, poprzez zmianę wielkości w taki sposób by obraz cyfry zmieścił się z kwadracie 20x20 punktów z zachowaniem proporcji [13]. Następnie kwadraty te umieszczono centralnie w obrazy o rozdzielczości 28x28 punktów. Cyfry odwzorowane są w szarej jedno-bajtowej skali kolorów (w zakresie od 0 do 255) – rys. 4.



Rys. 4. Przykład obrazu cyfry z bazy MNIST [13]

5.3. Plan eksperymentu

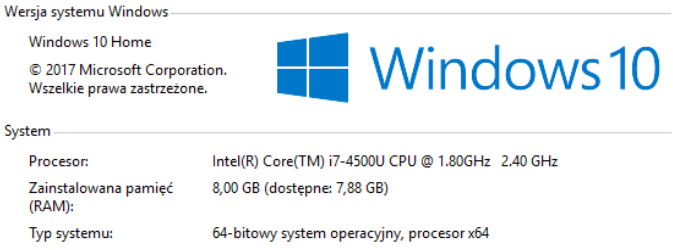
Obie sieci neuronowe trenowane będą przy pomocy tego samego zbioru uczącego, by następnie sprawdzić jakość rozpoznawania cyfr pisanych odręcznie przy pomocy zbioru testującego [13].

Przy pomocy biblioteki TensorFlow będą implementowane SSN określonej architektury i parametrach (liczbie neuronów w warstwach). Obie sieci będą miały 784 wejścia (28x28) i 10 wyjść, odpowiadających każdej z 10 cyfr. Warstwa wejściowa i wyjściowa będą miały dodatkowe wejście zwane bias, czyli zerowe. Wszystkie neurony będą miały funkcję aktywacji: max(0,z). Początkowe wartości wag wszystkich warstw będą generowane losowo i modyfikowane w trakcie uczenia sieci. Ze względu na ograniczenie pamięci RAM proces uczenia będzie podzielony na 10 kroków po 6000 znaków każdy.

Oprócz implementacji SSN należy zaimplementować proces ich uczenia i testowania. Wykorzystuje się do tego odpowiednie procedury biblioteki TensorFlow.

5.4. Realizacja eksperymentu

Eksperyment przeprowadzony został na komputerze o parametrach jak na rys. 5. W trakcie badania nie były uruchamiane żadne dodatkowe procesy, a procesor był wykorzystywany w 100% przez program uczenia sieci.



Rys. 5. Parametry systemu użytego do realizacji eksperymentu [6]

Na rys. 6 i 7 przedstawiono fragmenty programów realizujących sieci poszczególne. Przykładowy program uczenia sieci jednowarstwowej przedstawiony został na rys. 8. Trenowanie obydwu sieci odbyło się przy pomocy analogicznego algorytmu.

```
15  def neural_network_model(data):
16
17      hidden_layer1 = {'weights': tf.Variable(tf.random_normal([784, n_nodes_hl1])),
18                       'biases': tf.Variable(tf.random_normal([n_nodes_hl1]))}
19
20      output_layer = {'weights': tf.Variable(tf.random_normal([n_nodes_hl1, n_classes])),
21                      'biases': tf.Variable(tf.random_normal([n_classes]))}
22
23      l1 = tf.add(tf.matmul(data, hidden_layer1['weights']), hidden_layer1['biases'])
24      l1 = tf.nn.relu(l1)
25
26      output = tf.add(tf.matmul(l1, output_layer['weights']), output_layer['biases'])
27      return output
```

Rys. 6. Definicja sieci z jedną warstwą ukrytą [6]

```
19  def neural_network_model(data):
20      # input_data * weights + biases
21      hidden_l1 = {'weights': tf.Variable(tf.random_normal([784, n_nodes_hl1])),
22                   'biases': tf.Variable(tf.random_normal([n_nodes_hl1]))}
23
24      hidden_l2 = {'weights': tf.Variable(tf.random_normal([n_nodes_hl1, n_nodes_hl2])),
25                   'biases': tf.Variable(tf.random_normal([n_nodes_hl2]))}
26
27      output_l = {'weights': tf.Variable(tf.random_normal([n_nodes_hl2, n_classes])),
28                  'biases': tf.Variable(tf.random_normal([n_classes]))}
29
30      l1 = tf.add(tf.matmul(data, hidden_l1['weights']), hidden_l1['biases'])
31      l1 = tf.nn.relu(l1)
32
33      l2 = tf.add(tf.matmul(l1, hidden_l2['weights']), hidden_l2['biases'])
34      l2 = tf.nn.relu(l2)
35
36      output = tf.add(tf.matmul(l2, output_l['weights']), output_l['biases'])
37      return output
```

Rys. 7. Definicja sieci z dwoma warstwami ukrytymi [6]

```
def train_neural_network(x):
    start_time = time.time()
    prediction = neural_network_model(x)
    cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(logits=prediction, labels=y))
    # optimizer value = 0.001, Adam similar to SGD
    optimizer = tf.train.AdamOptimizer().minimize(cost)
    epochs_no = 10

    with tf.Session() as sess:
        sess.run(tf.global_variables_initializer())

        # training
        for epoch in range(epochs_no):
            epoch_loss = 0
            for _ in range(int(mnist.train.num_examples / batch_size)):
                epoch_x, epoch_y = mnist.train.next_batch(batch_size)
                _, c = sess.run([optimizer, cost], feed_dict={x: epoch_x, y: epoch_y})
                # code that optimizes the weights & biases
                epoch_loss += c
            print('Epoch', epoch, 'completed out of', epochs_no, 'loss:', epoch_loss)

        # testing
        correct = tf.equal(tf.argmax(prediction, 1), tf.argmax(y, 1))
        accuracy = tf.reduce_mean(tf.cast(correct, 'float'))
        print('Accuracy:', accuracy.eval({x: mnist.test.images, y: mnist.test.labels}))
    print("--- %s seconds ---" % (time.time() - start_time))
```

Rys. 8. Program uczenia sieci z jedną warstwą ukrytą [6]

6. Rezultaty eksperymentu

W tab. 1 i 2 przedstawiono rezultaty eksperymentu dla SSN z jedną i dwoma warstwami ukrytymi.

Tabela 1. Analiza poprawności rozpoznawania znaków i czasu uczenia sieci z jedną warstwą ukrytą [6]

Liczba neuronów w warstwie ukrytej	Poprawność (%)	Czas uczenia (s)
10	66,98	6,92
50	89,06	10,93
100	92,37	16,32
200	93,63	25,45
300	94,08	34,61
400	94,36	45,85
500	94,15	56,38
600	94,64	66,62
700	94,89	75,03
800	94,84	84,82

By odpowiedzieć na pytanie badawcze dokonano przekształcenia rezultatów badań poprzez zsumowanie liczby neuronów w sieci o dwóch warstwach ukrytych i uporządkowanie tych rezultatów wzrastająco według współczynnika poprawności rozpoznawania znaków pisma odręcznego.

Tab. 3 zawiera rezultaty porównania efektywności obydwu SSN (tj z jedną i dwoma warstwami ukrytymi) w rozpoznawaniu znaków pisma odręcznego.

Tabela 2. Analiza poprawności rozpoznawania znaków i czasu uczenia sieci z dwoma warstwami ukrytymi [6]

Liczba neuronów w pierwszej warstwie ukrytej	Liczba neuronów w drugiej warstwie ukrytej	Poprawność (%)	Czas uczenia (s)
10	10	47,16	7,47
10	50	43,84	8,15
10	100	45,49	8,51
10	200	32,16	9,83
10	300	43,3	11,01
10	400	36,89	11,77
10	500	33,36	12,64
50	10	40,29	11,34
50	50	85,9	12,13
50	100	89,6	12,97
50	200	88,18	14,94
50	300	89,18	15,86
100	10	28,46	15,63
100	50	78,8	17,3
100	100	91,82	18,08
100	200	92,14	21,1
100	300	92,2	23,71
100	400	93,12	23,46
100	500	92,33	26,23
200	10	33,84	26,01
200	50	79,16	28,09
200	100	92,35	29,5
200	200	93,38	32,15
200	300	93,53	36,37
200	400	93,58	37,71
200	500	93,86	42,08
300	10	14,21	36,2
300	50	74,12	37,32
300	100	93,54	39,5
300	200	94,21	45,11
300	300	94,06	48,09
400	10	35,46	46,93
400	50	92,33	48,73
400	100	93,66	50,7
400	200	94,21	57,33
400	300	94,6	60,18
400	400	94,7	66,54
400	500	94,96	71,23
500	10	17,8	57,51
500	50	68,69	59,22
500	100	93,21	63,37
500	200	94,82	68,17
500	300	94,86	73,57
500	400	94,81	80,93
500	500	94,89	87,45

Tabela 3. Porównanie efektywności rozpoznawania znaków przez SSN

Liczba warstw ukrytych	Poprawność, %	Łączna liczba neuronów	Czas uczenia, [s]
1	>90%	100	16
	>94%	300	34
2	>90%	200	18
	>94%	600	48

7. Podsumowanie

Rezultaty eksperymentu wskazują, że w warunkach normalizacji obrazów znaków pisanych do niezbyt dużego obszaru graficznego (20x20 punktów w 256 poziomach szarości), najlepszą SSN jest sieć o jednej warstwie ukrytej (identycznie nauczona jak sieć z dwoma warstwami). Liczba neuronów w takiej sieci jest dwa razy mniejsza w porównaniu z siecią z dwoma warstwami (tab. 3). Podobnie czas uczenia sieci z jedną warstwą ukrytą o analogicznej jakości w porównaniu z siecią o dwóch warstwach jest krótszy (tab. 3).

Odpowiedź na pytanie badawcze jest zatem **negatywna** – dodatkowa warstwa dla odpowiednio dobranej liczby neuronów SSN jednowarstwowej nie zwiększa efektywności rozpoznawania znaków pisma odręcznego.

Już przy niewielkiej liczbie neuronów SSN o jednej warstwie ukrytej ma bardzo dobrą jakość rozpoznawania cyfr pisanych odręcznie. Poprawność powyżej 90% osiągana jest już przy niecałych 100 neuronach. Jest to niewątpliwie rezultat normalizacji obrazów graficznych cyfr i dużego zbioru uczącego, a także małej liczbie wariantów rezultatu (10 cyfr).

Rezultat przedstawionego eksperymentu i wnioski z niego płynące może być wykorzystany w procesie doboru algorytmów (w tym przypadku: normalizacji znaku i struktury SSN) w procesie tworzenia systemu rozpoznawania pisma odręcznego.

Literatura

[1] ICR – czy warto skanować pismo odręczne?, http://ocrwdokumentach.pl/icr-rozpoznawanie-pisma-odrecznego/ [11.01.2018]

[2] S. Anagnoste, Robotic Automation Process - The next major revolution in terms of back office operations improvement, Proceedings of The International Conference on Business Excellence, vol 11(1) (2017), 676-686.

[3] W. Kacalak, M. Majewski, A New Method for Handwriting Recognition Using Artificial Neural Networks. Intelligent Engineering Systems Through Artificial Neural Networks, 16 (2006), 459-465.

[4] J. Smołka, M. Skublewska-Paszkowska, E. Łukasik, Algorithm for selecting optimal clustering parameters used for over-segmentation reduction. PRZEGLAD ELEKTROTECHNICZNY, 9, (2016), 250-256.

[5] Z. Gomółka, B. Twaróg, E. Żesławska, Rozpoznawanie pisma odręcznego za pomocą sztucznych sieci neuronowych, Technical News, (2013), 98-102.

[6] J. Gazda, Zastosowanie sztucznych sieci neuronowych do rozpoznawania tekstu. Praca dyplomowa pod kierunkiem M.Miłosza, Lublin, (2018), 42.

[7] What is the TensorFlow machine intelligence platform? https://opensource.com/article/17/11/intro-tensorflow [11.01.2018]

[8] HaoBiao, Dae-Seong Kang, The Research of Face Expression Recognition based on CNN using Tensorflow. Journal of Advanced Information Technology and Convergence, 7, (2017), 55-63.

[9] A. Ignatov, Real-time human activity recognition from accelerometer data using Convolutional Neural Networks. Applied Soft Computing, 62, (2018), 915-922.

[10] F. Ertam, G. Aydin, Data classification with deep learning using Tensorflow. 2017 International Conference on Computer Science and Engineering (UBMK), (2017), 755-758.

[11] N. Gavai, Y. Jakhade, S. Tribhuvan, R. Bhattad, MobileNets for flower classification using TensorFlow. 2017 International Conference on Big Data, IoT and Data Science (BID) Big Data, IoT and Data Science, (2017), 154-158.

[12] J. Evermann, J. R. Rehse, P. Fettke, XES tensorflow - Process prediction using the tensorflow deep-learning framework. Proceedings of the 29th International Conference on Advanced Information Systems Engineering, 1848, (2017), 41-48.

[13] The MNIST database of handwritten digits. http://yann.lecun.com/exdb/mnist/ [11.01.2018]