Rozpoznawanie pisma odręcznego za pomocą sztucznych sieci neuronowych

Z. Gomółka, B. Twaróg, E. Żesławska

Uniwersytet Rzeszowski, Interdyscyplinarne Centrum Modelowania Komputerowego, 35-959 Rzeszów, ul. Pigonia 1

Abstract. W artykule przedstawiono problematykę zastosowania sztucznych sieci neuronowych do odczytu pisma ręcznego. Automatyczne rozpoznawanie znaków pisma stanowi ważną gałąź badań związanych z przetwarzaniem języka naturalnego za pomocą komputera. OCR pozwala na zmianę dokumentów istniejących w postaci pisemnej lub drukowanej do odpowiadającej im postaci elektronicznej. Stworzono aplikację do analizy pisma odręcznego i przeprowadzono szereg symulacji. Wyniki i konkluzje zawarto w części końcowej pracy.

Wprowadzenie

Wykorzystuje się dużo rozmaitych metod rozpoznawania znaków i obrazu w zależności od typu identyfikowanych znaków. Wszystkie metody opierają się na terminie przestrzeni cech, to znaczy n-wymiarowej przestrzeni, w której każdy identyfikowany obiekt jest odwzorowywany. Wszystkim jest przydzielony specyficzny zestaw cech, znany jako wektor cech podzielony na trzy kategorie: rzeczywisty, całkowity i binarny. Przy czym należy odnotować, że wektory rzeczywiste zawierać mogą dużo więcej informacji, aczkolwiek czas potrzebny, żeby je przetworzyć i pamięć przechowywania są duże. Tymczasem wektory binarne są najmniej zasobo- i czasochłonne, ale ilość informacji, które niosą jest mniejsza. Trudno jest określić, który rodzaj wektora cech jest najodpowiedniejszy. Zależy to od typu identyfikowanych obiektów i użytych cech. Wektory cech podzielić można, ze względu na ich długość o zmiennej i stałej ilości elementów. Pierwszy typ jest kategorycznie lepszy, natomiast drugi prowadzi to do karkołomnego zadania jakim jest zestawianie wektorów o różnych iloś–ciach elementów.

Wektory cech

Znaki mogą być reprezentowane przez cechy strukturalne o wysokiej odporności na zakłócenia i zmiany stylu. Ten rodzaj reprezentacji koduję pewną wiedzę na temat struktury obiektu lub może dostarczyć pewnej wiedzy co do tego, z jakich rodzajów elementów składa się taki obiekt. Cechy strukturalne są oparte na topologicznych i geometrycznych właściwości znaku, takie jak proporcje, punkty krzyżowe, pętle, punkty gałęzi, pociągnięcia i ich kierunki, przegięcia między dwoma punktami, łuki poziome na górze lub na dole itp.[1]. Do globalnych transformacji i momentów zaliczyć można transformatę Fouriera (FT), obliczana z konturu obrazu. Jako że pierwsze n współczynników FT może być wykorzystywane w celu odbudowy konturu, to te n współczynników można uważać za n-wymiarowy wektor cech, który reprezentuje znak. Współczynniki momentów centralnego i Zernike'a mogą posłużyć do całkowitej rekonstrukcji oryginalnego obrazu, przy czym czynią proces rozpoznawani znaków odporny na skalę obiektu, jego translację, czy rotację[6,7]. Cechy receptorowe tworzone są przez utworzenie pewnej stałej ilości receptorów i sprawdzenie czy dany receptor lub jego część znajduje się w obrębie figury znaku.

Klasyfikacja znaków

Klasyfikacja, jako ostatni etap rozpoznawania, polega na przyporządkowaniu znaku do zdefiniowanej klasy. Wyróżnić można wiele różnych klasyfikatorów: k-NN – k-najbliższych sąsiadów, klasyfikator bayesowki, NN – sieci neuronowe, HMM – ukryte modele Markowa, SVM – maszyna wektorów wsparcia, itp. [1,3]. Wspólną cechą większości jest podawanie na wejście wektora cech rozpoznawanego obiektu i otrzymywanie na wyjściu numeru klasy, do której należy. Przydzielenie znaków do klas odbywa się na podstawie wiedzy zgromadzonej przez klasyfikator. Stanowią ją wektory cech znaków, których przynależność do klas jest znana. Niektóre klasyfikatory oparte na sieciach neuronowych nie korzystają z wektorów cech. Na ich wejście podawana jest mapa bitowa obiektu, na podstawie, której wykonywana jest klasyfikacja. Natomiast klasyfikatory oparte o logikę rozmytą nie dają na wyjściu numeru jednej klasy, do której dany znak należy, lecz rozmytą funkcję przynależności danego obiektu do każdej z klas.

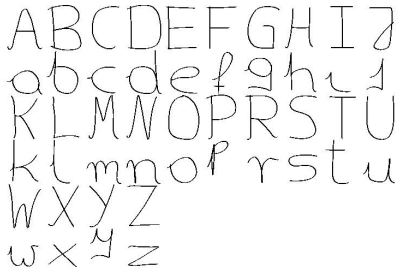
W celu zidentyfikowania znaku należy znaleźć wśród wektorów, których przynależność jest znana, wektor najbliższy do wektora cech tego znaku. W praktyce szuka się minimum normy różnicy wektorów. Ta metoda, choć stosunkowo szybka i prosta, może owocować dużą ilością błędów. Aby się ich ustrzec wykorzystuje się bardziej złożony algorytm rozpoznawania. Klasyfikator K-najbliższych sąsiadów zamiast jednego znajduje K najbliższych wektorów do przetwarzanego wektora [4,5]. Klasą przynależności identyfikowanego obiektu jest ta klasa, której reprezentantów jest najwięcej pośród K najbliższych wektorów. Jeśli liczba reprezentantów więcej niż jednej klasy jest największa i równa, wybierana jest ta klasa, której suma odległości wszystkich wektorów jest najmniejsza. Najlepsze wartości parametrów klasyfikatora wyznaczyć można jedynie eksperymentalnie, ponieważ zależna jest ona od wektorów cech.

Wstępne przetwarzanie obrazu

Obraz O(R,G,B) wyrażony w trzech kanałach RGB przekształcany jest w obraz $O(Y,C_b,C_r)$ wyrażony jednym kanałem luminancji i dwoma kanałami chrominancji za pomocą wzorów:

$$Y = 0.299R + 0.587G + 0.114B$$
$$C_b = 0.169R - 0.331G + 0.500B \quad (1)$$
$$C_r = 0.500\% - 0.419G - 0.081B$$

Przy założeniu, że analizowane będą tylko te obrazy, na których jest ciemny tekst na jasnym tle, to jedyne potrzebne informacje niesie kanał luminancji. Można więc odrzucić oba kanały chrominancji, co zmniejszy złożoność obliczeniową programu. Następnie należy przeprowadzić binaryzację obrazu za pomocą segmentacji przez progowanie, z progiem globalnym wyznaczonym za pomocą histogramu. Histogram obrazu obliczony jest prostym algorytmem przeglądającym obraz, zliczającym ilość pikseli o danym kolorze i zapisującym tą ilość w tablicy. Na podstawie histogramu wyznaczany jest próg binaryzacji jako pierwsze minimum ilości pikseli w danym kolorze najbliższe kolorowi czarnemu, po przynajmniej jednej niezerowej wartości w histogramie. W celu uniknięcia sytuacji, gdzie występują minima w niepożądanych miejscach, tj. między kolorami pikseli, które chcemy zachować, próg binaryzacji jest przesuwany o 20 kolorów w stronę koloru białego.



Rys. 1. Wzorce, efekt binaryzacji

W analizowanych obrazach może występować szum, stworzony przez urządzenia skanujące. W celu uzyskania jak najlepszych rezultatów w rozpoznawaniu, szum należy zredukować do minimum, aby tego dokonać można zastosować różne filtry, od filtrów dolnoprzepustowych po filtry statystyczne. Poniżej przedstawiono efekty działania filtrów na przykładowym zaszumionym obrazie binarnym.

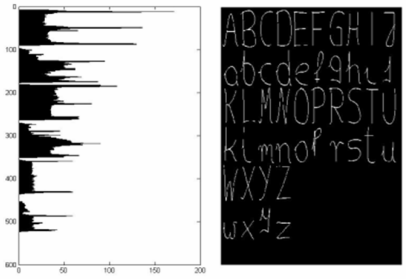
Rys.2. Obraz zaszumiony

Rys.3. Efekt działania filtru minmax

Analizując efekty działania filtrów widzimy, że filtr minmax nie usunął wszystkich szumów. Filtr dolno–przepustowy wymaga ponownej binaryzacji obrazu i nie tyle usunął szumy, co je rozmył. Najlepszy rezultat uzys–kano wykorzystując filtr medianowy, tak otrzymany obraz przygotowany jest do dalszej pracy. W celu pro–wadzenia kolejnych etapów kolory obrazów są odwracane, wszystkie czarne piksele obrazu (wartość 0) stają się bia–łymi (war–tość 1) i odwrotnie.

Wyodrębnienie linijek tekstu

W celu wyodrębnienia linijek tekstu należy policzyć białe piksele na obrazie powstałym po operacjach związanych z przetwarzaniem obrazu. Wzorce sum ilości białych pikseli pokazano na rys.4. Następnie należy znaleźć grupy niezerowych sąsiadujących ze sobą wartości tych sum. Najprawdopodobniej linijki tekstu znajdować się będą w rzędach obrazu odpowiadającym wyznaczonym grupom.



Rys. 4. Wzorce sumy ilości białych pikseli

Podzielenie linijek tekstu na wyrazy i normali–zacja wyrazów

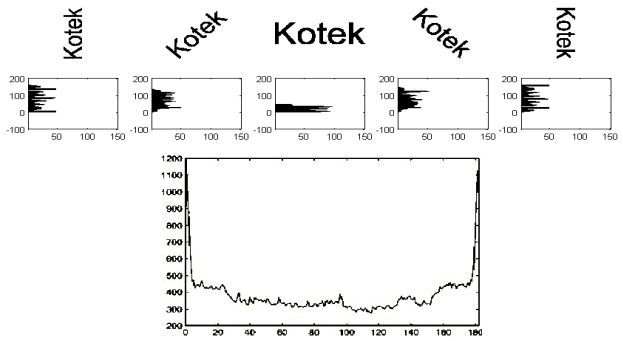
Poszczególne linijki tekstu podzielić można w ten sam sposób jak dzieliliśmy obraz na linijki, tylko zliczając piksele w poziomie zamiast w pionie. Program będzie używał innej metody, będzie znajdywany skrajny lewy biały piksel i analizował piksele sąsiadujące z nim w poszu–kiwaniu innych niezerowych punktów. Następnym krokiem będzie analizowanie sąsiedztwa tych znalezionych białych pikseli i tak aż nie będzie więcej niezanalizowanych po–łączonych ze sobą pikseli o wartości 1. Przewagą tej drugiej metody jest to, że poradzi sobie ona w sytuacji przed–stawionej na rys.5 , chociaż jest ona dużo wolniejsza.

Rys.5. Przykład sytuacji nie do rozwiązania dla segmentacji przez projekcje

Zwykle na tym etapie analizy obrazu przeprowadzane są korekty kąta pomiędzy kierunkiem poziomym a linią słowa oraz kąta nachylenia liter [2].

Rys. 6. Korygowane kąty

Najczęściej używanym algorytmem do korekcji skosu i nachylenia jest rozkład Wignera-Ville'a. Do korekcji skosu używane są poziome histogramy tego słowa w różnych kątach skosu i odpowiadające im wykresy maksymalnej intensywności, które są pokazane na rys. 7 dla kątów 0°, 45°, 90°, 135°, 180°. Należy zauważyć to, że wierzchołki są najbardziej rozproszone w 0° i w 180°, a najbardziej zwarte w 90°.



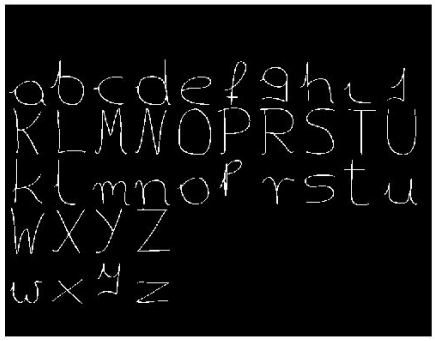
Rys. 7. Histogramy projekcji dla poszczególnych kątów skosu i wykres zwartości wierzchołków w stosunku do kąta skosu

Do korekcji nachylenia, koncepcja jest taka sama jak przy korekcji skosu. Jedyną różnicą jest to używane są pionowe histogramy. Inne techniki wykorzystywane to Profil projekcji, transformaty Fouriera i Hougha, techniki te mają jednak niewielki kąt pracy

Projekt modułu przygotowującego wzorce i uczą–cego sieć wzorców.

Przygotowanie programu do pracy odbywa się w kilku etapach. Pierwszym etapem jest wczytanie i wstępna obróbka obrazu. Następnie przeprowadzane jest wstępne przetwarzanie za pomocą funkcji bitlum. Wyznaczany jest kanał luminancji, obliczany histogram obrazu, po czym przeprowadzana jest segmentacja obrazu. W kolejnym etapie obraz poddawany jest filtracji za pomocą wybranego filtra. Drugi etap to podzielenie obrazu na linijki i odsepa–rowanie wzorców. Dzielimy obraz na linijki znaj–dując pierwszy i ostatni element pierwszej grupy nieze–ro–wych sum białych pikseli w liczonych w rzędach obrazu, następ–nie wycinamy fragment obrazu odpowiadającego tej grupie. Etap ten działa według następującego schematu:

1. Funkcja znajduje pierwszy górny piksel o wartości 1 po lewej stronie i jego współrzędne na obrazie dodawane są do kolejki;
2. Pobierane są współrzędne jednego piksela z kolejki i piksel kopiowany jest do tablicy literka, a jego wartość w linijce jest zerowana;
3. Sprawdzane są wartości sąsiednich pikseli w kolejności podanej poniżej, gdy badany piksel ma wartość 1, spraw–dzane jest czy informacje o tym pikselu znajdują się już w kolejce, jeśli nie to współrzędne tego punktu dodawane są do niej;
4. Sprawdzane jest to czy kolejka zawiera jeszcze jakieś elementy, jeśli tak to program powraca do punktu 2;
5. Zawartość tablicy literka kopiowana jest do tablicy litery, a następnie zerowana;
6. Algorytm sprawdza czy w linijce tekstu znajdują się jeszcze piksele o wartościach 1, jeśli tak to powraca do punktu pierwszego.



Rys. 8. Wzorce, efekt wycięcia pierwszej linii wzorców

Rys. 9. Pierwsza linijka wzorców

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & p & 5 \\ 6 & 7 & 8 \end{bmatrix}$$

Rys. 10. Kolejność analizowania sąsiedztwa punktu p

Rys.11. Zmienna słowo po analizie 200 pikseli

Rys.12. Zawartość macierzy lwy po analizie 200 pikseli

Rys. 13. Odseparowany pierwszy wzorzec

W fazie normalizacji wzorców przeprowadzane są norma–lizacje wielkości i położenia litery w macierzy wzorca. Najpierw funkcja umieszcza wzorzec w macierzy kwad–ratowej o wymiarach (x,x), gdzie x jest większą z wartości szerokości i wysokości wzorca. Litery, które mają większą wysokość niż szerokość, są wyśrodkowane w macierzy wyjściowej funkcji w poziomie i na odwrót. Następnie przeprowadzane jest skalowanie macierzy wyjściowej do wcześniej ustalonej wielkości. Kolejnym krokiem jest przy–gotowanie trzech różnych zestawów cech wzorców. Pier–wszy zestaw jest to cała macierz wzorca podana kolejno rzędami. Drugi zestaw to odpowiedzi wygenerowanych wcześniej receptorów. W tym przypadku są to odcinki lub punkty, a ich odpowiedź jest wynikiem sprawdzenia, czy któryś z pikseli składowych receptora znajduję się w kon–turze litery. Trzeci zestaw to cechy strukturalne, w któ–rych skład wchodzą m.in. pole powierzchni, współrzędne środka ciężkości, ekscentryczność, orientacja, rozmiar ob–szaru

wypukłego, długości osi głównej, itp. Następnie wek–tory cech są normalizowane według wzoru:

$$x_i(j) = \frac{x_i(j)}{\sum_{v=1}^{n} x_v(j)} \qquad (2)$$

Poniżej przedstawiono odpowiednio przygotowane wzorce oraz przykładowe zbiory receptorów.

Rys. 14. Odseparowany wzorzec przed normalizacją

Rys. 15. Wzorzec wyśrodkowany

Rys. 16. Wzorzec wyśrodkowany i zmniejszony

Rys. 17. Zbiór przykładowych receptorów

Rys. 18. Zbiór przykładowych receptorów po filtracji

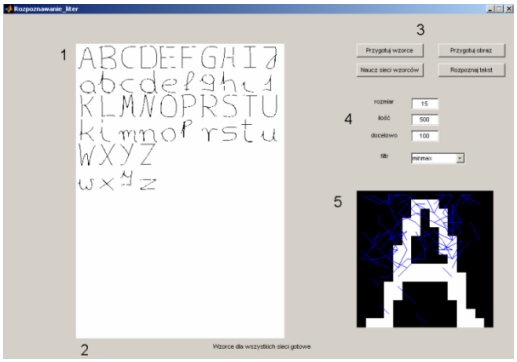
Kolejny, krok to utworzenie macierzy cech wzorców, które są wykorzystywane do nauczanie sieci rozpoznawania wzorców. Dla każdego zestawu wektorów cech wzorców tworzona jest sieć neuronowa, inicjuje się wagi synap–tyczne, a następnie sieć jest uczona zgodnie z algorytmem uczenia z rywalizacją.

Rozpoznawanie pisma

Pierwszy etap polega na wczytaniu i wstępnej obróbce obrazu, przebiega dokładnie tak jak poprzednio. Następnie wykonywana jest operacja wczytania, wstępnej obróbki obrazu, podzielenia obrazu na linijki tekstu i po–dzielenia ich na poszczególne słowa. Proces ten wykony–wany jest przy użyciu poprzednich funkcji. Dzielenie ob–razu wykorzystywane są przedstawione wcześnie, ze względu na sposób działania funkcje te świetnie radzą sobie także z tym zadaniem. Kolejny etap to dzielenie wyrazów na litery, ich rozpoznawanie oraz separacja znaków.

Interfejsu programu

Interfejs programu został zaprojektowany tak, aby był jak najprostszy i zrozumiały dla użytkownika. Wygląd gotowego interfejsu został przedstawiony na rys. 21, składa się z następujących sekcji:
1. Główne okno ilustrujące
2. Pasek stanu
3. Przyciski sterujące
4. Pola tekstowe zawierające parametry programu
5. Pomocnicze okno ilustrujące



Rys. 19. Interfejs programu do rozpoznawania liter

Testy aplikacji

W celu przeprowadzenia testów utworzono dwa zestawy wzorców oraz sześć próbek testowych. Próbki tes–towe różnią się od siebie m.in. sposobem stworzenia, pisa–nia, rozdzielczością, w której są zapisane oraz wyso–kością liter. Wszystkie obrazy uczestniczące w testach są w forma–cie JPEG, a znaki na nich są koloru czarnego na białym tle. W tabeli 1 zestawiono ilość liter wykorzystanych w po–szczególnych próbkach.

Nr próbki	1	2	3	4	5	6	Razem
Ilość znaków	25	21	30	27	29	26	158

Tabela 1: Ilość liter w poszczególnych próbkach

Testy przeprowadzone zostały w dwóch etapach:
I. Sieć neuronowa nauczona za pomocą wektorów cech powstałych osobno na podstawie wzorców, bez przepro–wadzanej szkieletyzacji w etapie preprocessingu;
II. Sieć neuronowa nauczona za pomocą wektorów cech powstałych osobno na podstawie wzorców, z przepro–wa–dzaną szkieletyzacją w fazie preprocesingu.
Wszystkie testy składają się z przygotowania wzorców dla sieci, nauczenia sieci rozpoznawania liter i próby rozpoznania liter z czterema różnymi rozmiarami prze–skalowanej macierzy znaku po normalizacji litery dla wszystkich sześciu próbek testowych. Łącznie zostało wykonanych 288 pojedynczych prób identyfikacji tekstu na obrazie. W tabeli 2 przedstawione są średnie arytmetyczne wyników otrzymanych łącznie w całym bloku testowym.

Rozmiar macierzy po skalowaniu	Test nr 1	Test nr 2	Test nr 3	Test nr 4	Średnia
11 x 11	20,04%	12,66%	11,60%	7,59%	12,97%
15 x 15	20,68%	13,29%	13,08%	11,81%	14,72%
19 x 19	16,88%	12,66%	21,31%	13,29%	16,03%
23 x 23	16,88%	14,98%	20,46%	12,24%	16,14%
Średnia z	18,62%	13,40%	16,61%	11,23%	14,97%

jednego testu					
				Średnia łączna	14,97%

Tabela. 2. Wyniki testów ze względu na rozmiar macierzy po skalowaniu

Otrzymane rezultaty można analizować ze względu na skuteczność identyfikacji z przeprowadzoną szkieletyzacją i bez, dla różnych rozmiarów macierzy znaku po przeska–lowaniu oraz dla dwóch różnych zestawów wzorców W wyniku przeprowadzenia badań okazało się, że skuteczność analizy z wprowadzoną szkieletyzacją jest mniejsza niż bez niej. Ważnym elementem rozpoznawania pisma jest norma–lizacja i odpowiedni dobór parametrów programu i dopaso–wanie zestawu wzorców do konkretnej sytuacji. Problem ten można rozwiązać zmieniając architekturę sieci neuro–nowej i/lub poprzez zwiększenie ilości wzorców przy–padających na jedną klasę rozpoznawanych obiektów. W tabeli 3 poniżej znajdują się wyniki rozpoznawania po–szczególnych próbek testowych.

	Test 1	Test 2	Test 3	Test 4
Próbka 1	41,67%	20,67%	38,00%	23,00%
Próbka 2	19,44%	20,63%	13,10%	8,73%
Próbka 3	12,22%	2,78%	10,83%	5,00%
Próbka 4	13,27%	17,90%	18,83%	14,51%
Próbka 5	12,93%	7,18%	9,48%	5,17%
Próbka 6	15,06%	15,06%	11,22%	12,50%

Tabela 3. Wyniki testów ze względu próbkę testową

W przypadku, gdy litery w próbce testowej są prawie identyczne w porównaniu ze wzorcem otrzymane wyniki były zadowalające. W procesie rozpoznawania tekstu nie uniknione są błędy, które można podzielić na dwie grupy: błędy segmentacji i identyfikacji. Większość błędów była spowodowana błędną segmentacją, przy czym są to tylko błędy segmentacji wyrazów na znaki. Podział obrazu na linie i linii na wyrazy w tych próbkach testowych został przeprowadzony ze 100%-ową skutecznością. Aby zmniej–szyć ilość błędów, należałoby ulepszyć lub zastosować inna metodę segmentacji wyrazów, chociaż nie wyelimi–no–wałoby to wszystkich błędów. Nawet przy znalezieniu op–tymalnego sposobu na podział słów, pozostałaby kwestia błędów klasyfikacji, którą można by rozwiązać poprzez dobór innej architektury sieci, poprawę sposobu ekstrakcji cech lub zmianę klasyfikatora.

Podsumowanie

W artykule przedstawiono problematykę zasto–so–wania sztucznych sieci neuronowych do odczytu pisma ręcznego. Zaprezentowano metody i techniki związane z optycznym rozpoznawaniem znaków oraz stworzono aplikację do odczytywania pisma ręcznego. Następnie prze–prowadzono szereg testów w celu sprawdzenia skuteczności rozpoznawania pisma przez aplikację. Ze względu na złożoność zagadnienia rozpoznawania pisma odręcznego stworzona aplikacja nie gwarantuje wysokiej skuteczności rozpoznawania pisma. Pomimo to opracowany program posiada szereg zalet. Przede wszystkim program ten można dostosować do charakteru pisma konkretnego użytkownika, za pomocą wzorców stworzonych przez niego. Ze względu na architekturę użytej sieci neuronowej i wykorzystanych w nim technik, program wykazuje się duża szybkością działania. Stosując się do pewnych zasad przy tworzeniu tekstów do rozpoznania można osiągnąć 100%-owo po–prawną segmentację obrazu na wyrazy. W przypadku stabilnego jednostajnego pisma program może osiągnąć zadowalające wyniki. Głównym powodem niepoprawnej klasyfikacji liter przez program jest błędna segmentacja wyrazów na znaki. Rozwiązaniem tego problemu jest ulep–szenie lub zmiana metody segmentacji. Nawet przy znale–zieniu optymalnego sposobu na podział słów, pozostaje kwestia błędów klasyfikacji. Tutaj być może rozwiązaniem jest dobór innej architektury sieci, zmiana sposobu ekstrak–cji cech, zmiana klasyfikatora lub zastosowanie kilku róż–nych klasyfikatorów jednocześnie.

Literatura

[1] J. Hull, "A database for handwritten text recognition research," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 16, no. 5, pp. 550-554, 1994.

[2] K.-H. Steinke and B. Mund, "Datamining in Herbar–belegen," in Wismarer Wirtschaftsinformatik –Tage, 2010.

[3] N. Das, J. M. Reddy, R. Sarkar, S. Basu, M. Kundu, M. Nasi–puri, and D. K. Basu, "A Statistical-Topological Feature Combination for Recognition of Handwritten Numerals," Applied Soft Computing, vol. 12, no. 8, pp. 2486-2495, 2012.

[4] N. Das, R. Sarkar, S. Basu, M. Kundu, M. Nasipuri, and D. K. Basu, "A Genetic Algorithm based Region Sampling for Selection of Local Features in Handwritten Digit Recognition Application," Applied Soft Computing, vol. 12, no. 5, pp. 1592-1606, 2012.

[5] N. Das, S. Pramanik, S. Basu, P. Saha, R. Sarkar, M. Kundu, and M. Nasipuri, "Recognition of Handwritten Bangla Basic Characters and Digits using Convex Hull based Feature Set," in International Conference on Artificial Intelligence and Pattern Recognition, 2009, pp. 380-386.

[6] N. Dalal and B. Triggs, "Histograms of oriented Gra–dients for Human Detection," in Conference on Computer Vision and Pattern Recognition, vol. 1, 2005, pp. 886-893.

[7] Tadeusiewicz R., Sieci neuronowe, Akademicka Ofi–cyna Wydawnicza RM, Warszawa 1993 (Wyd. 2).