Marek Kamiński, Piotr Kopniak

Wydział Elektrotechniki i Informatyki
Politechnika Lubelska

Wykorzystanie sztucznych sieci neuronowych do rozpoznawania ruchu gałek ocznych i wspomagania komunikacji osób sparaliżowanych

1. Wstęp

Współczesna medycyna, jak powszechnie wiadomo, jest wspomagana przez różnego rodzaju rozwiązania techniczne i informatyczne. Ośrodki zdrowia wykorzystują specjalistyczny sprzęt diagnostyczny i leczniczy oraz oprogramowanie wspomagające. Wśród tych różnorodnych rozwiązań technicznych znajdują się także takie, które umożliwiają redukcję skutków niepełnosprawności. Technika coraz lepiej uzupełnia braki i niedomagania organizmu ludzkiego. Coraz powszechniej wykorzystuje się samojezdne wózki inwalidzkie, zrobotyzowane protezy oraz różnego rodzaju systemy wspomagania widzenia i słuchu. Wymienione rozwiązania techniczne są wspomagane przez jednostki obliczeniowe, które wymagają odpowiedniego oprogramowania. Może być to oprogramowanie sterujące sprzętem lub np. oprogramowanie inteligentnie wspomagające działania osoby niepełnosprawnej. W tym ostatnim przypadku do budowy aplikacji wykorzystuje się zdobycze naukowe związane ze sztuczną inteligencją.

Wykorzystanie sztucznej inteligencji w znacznym stopniu przyspiesza i automatyzuje rozwiązywanie złożonych problemów z różnych dziedzin nauki oraz umożliwia autonomiczne podejmowanie decyzji przez systemy komputerowe, nawet przy braku dostatecznej liczby danych wejściowych. Jest to dynamicznie rozwijająca się dziedzina, która ciągle znajduje nowe zastosowania, dlatego stała się również przedmiotem zainteresowania twórców systemu informatycznego opisywanego w niniejszym artykule.

Celem przedstawionych w tekście prac jest stworzenie taniego i prostego oprogramowania ułatwiającego komunikację ze światem zewnętrznym osobom sparaliżowanym. Stworzony i opisany w artykule prototyp oprogramowania, wykorzystujący sztuczne sieci neuronowe, umożliwia osobie niepełnosprawnej wprowadzanie tekstu jedynie za pomocą ruchu gałek ocznych.

2. Wprowadzenie do zagadnienia sztucznej inteligencji w kontekście sztucznych sieci neuronowych

2.1. Pojęcie sztucznej inteligencji

Mianem sztucznej inteligencji określa się systemy starające się odwzorować inteligentny sposób działania, głównie w odniesieniu do człowieka. Systemy tego typu działają zazwyczaj w określonych warunkach (środowisku), w których podejmują stosowne akcje w zależności od napływających danych wejściowych[1].

W związku z mnogością interpretacji, trudno jest jednoznacznie określić techniki wchodzące w skład sztucznej inteligencji, jednakże do często wyszczególnianych elementów sztucznej inteligencji należą takie kategorie, jak[2]:

- sztuczne sieci neuronowe,
- systemy eksperckie,
- algorytmy genetyczne i adaptacyjne.

2.2. Sztuczne sieci neuronowe – zastosowania

W tworzonej aplikacji zastosowano zbiór technik określanych pojęciem sztucznych sieci neuronowych. W związku z użyciem tychże technik zostaną pokrótce przedstawione: zasada działania sieci neuronowych, ich budowa oraz możliwości. Posłuży to do budowy docelowej aplikacji umożliwiającej interakcję z użytkownikiem.

Sztuczne sieci neuronowe pozwalają na ustalenie nieznanej funkcji opisującej, dysponując danymi z zakresu określonej przestrzeni wejściowej i najczęściej

[1] J. Nilsson, *Artificial Intelligence: A New Synthesis*, Morgan Kaufmann Publishers, San Francisco 1998, s. 1–7.

[2] L. Rutkowski, *Metody i techniki sztucznej inteligencji*, Wydawnictwo Naukowe PWN, Warszawa 2005, s. 1–19.

przestrzeni wyjściowej (aczkolwiek można przeprowadzić proces szkolenia sieci, zasilając ją jedynie danymi wejściowymi – proces ten nosi nazwę „szkolenia bez nauczyciela"). Właściwość taka pozwala na zastosowanie sieci neuronowych wszędzie tam, gdzie odnalezienie zależności pomiędzy danymi wejściowymi i wyjściowymi jest problematyczne bądź nie daje się jej ustalić za pomocą tradycyjnych metod[3].

Do przykładowych zastosowań sieci neuronowych należą takie aspekty, jak[4]:

- robotyka (rozpoznawanie obrazu, odwzorowanie zarejestrowanych ruchów kończyn),
- systemy wczesnego wykrywania zagrożenia,
- identyfikacja obiektów w polu widzenia kamery,
- optymalizacja algorytmów,
- wyszukiwanie nowych leków,
- systemy prognozowania i przewidywania notowań na giełdzie.

Można zauważyć, że kilka z powyższych zastosowań łączy się bezpośrednio bądź pośrednio z medycyną. Zastosowania takie jak wyszukiwanie nowych leków, odwzorowanie ruchu kończyn czy użyta przy tworzonej aplikacji identyfikacja obiektów w polu widzenia kamery są to elementy, które nieustannie się rozwija i w których obrębie przeprowadza się wiele symulacji, badań i modelowania właśnie w medycynie[5].

2.3. Sztuczne sieci neuronowe – budowa i zasada działania

Aby w pełni zrozumieć przedstawiany temat, przydatna okazuje się wiedza na temat tworzenia i budowy sztucznych sieci neuronowych. Działanie sztucznych sieci neuronowych można w uproszczeniu określić jako operację dokonującą przekształcenia przestrzeni wejściowej na przestrzeń wyjściową, według pewnej wewnętrznej funkcji[6].

---

[3] R. Tadeusiewicz, *Sieci neuronowe*, Akademicka Oficyna Wydawnicza RM, Warszawa 1993, s. 12–13.

[4] R. Tadeusiewicz, *Elementarne wprowadzenie do techniki sieci neuronowych z przykładowymi programami*, Akademicka Oficyna Wydawnicza PLJ, Warszawa 1998, s. 18–20.

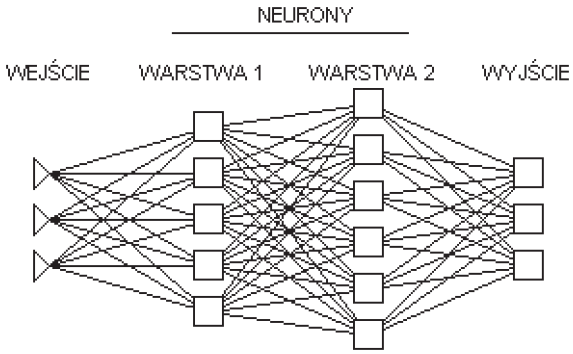
[5] S.M. Cesar, K. Sang-Woon, *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, Springer-Verlag, Berlin 2011.

[6] A.M. Arbib, *The Handbook of Brain Theory and Neural Networks*, The MIT Press, Cambridge 2003, s. 3–9.

Na sieć składają się zatem zawsze trzy warstwy logiczne: wejście, funkcja przetwarzania danych oraz wyjście, jednakże w praktyce najczęściej się mówi o warstwach fizycznych tworzących sieć, do których należą[7]:

- warstwa wejściowa,
- warstwa ukryta (w zależności od kontekstu określenie to może oznaczać ogół warstw wewnętrznych składających się na warstwę ukrytą bądź jedynie jedną z nich),
- warstwa wyjściowa.

Przykładowy szkic budowy sieci neuronowej przedstawia rysunek 1.



Rysunek 1. Przykład topologii sieci neuronowej jednokierunkowej o dwóch warstwach ukrytych

Źródło: http://www.statsoft.pl/textbook/graphics/neuralnw.gif.

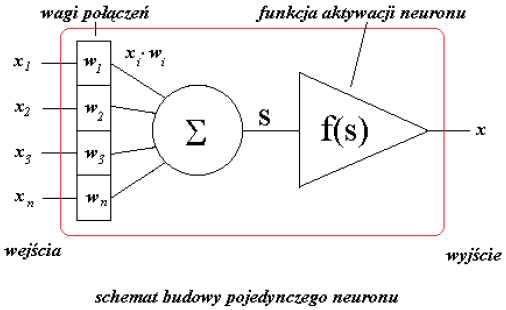
O ile warstwa wejściowa i wyjściowa występują zawsze, o tyle warstwy ukryte są opcjonalne. W zależności od rozpatrywanego przypadku może istnieć wiele warstw ukrytych bądź nie istnieć żadna (w takim przypadku warstwa wejściowa jest bezpośrednio połączona z warstwą wyjściową). Specyficznym przypadkiem są sieci rekurencyjne Hopfielda, w których nie istnieje warstwa ukryta, a warstwa wejściowa jest jednocześnie warstwą wyjściową[8].

Aby zrozumieć zasadę działania sieci, niezbędne jest zaprezentowanie zaimplementowanego w pamięci komputera uproszczonego modelu pojedynczego neuronu. Model ten przedstawia rysunek 2.

---

7 R. Tadeusiewicz, *Elementarne wprowadzenie*..., op.cit., s. 8–14.

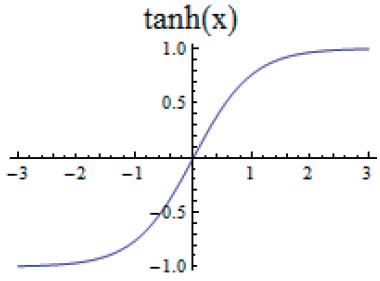
8 R. Tadeusiewicz, *Sieci neuronowe*..., op.cit., s. 10–13, 86–90.



Rysunek 2. Schemat budowy pojedynczego neuronu

Źródło: http://aragorn.pb.bialystok.pl/~radev/ai/se/zal/ready/krzyszkowski_pliki/image004.gif.

Każdy neuron posiada określoną liczbę wejść, zależną od liczby połączeń z neuronami poprzedniej warstwy. Każde z tych wejść posiada indywidualną wagę, określającą, jaka wartość trafi do bloku sumatora, a następnie na wejście funkcji aktywacji. Wartość wag poszczególnych neuronów w sieci jest ustalana podczas procesu szkolenia sieci, który zostanie szczegółowo opisany niżej. Po przeprowadzeniu procesu szkolenia sieci wszystkie wagi są ustalone i możliwa jest dalsza eksploatacja sieci[9].



Rysunek 3. Wykres funkcji tangens hiperboliczny

Źródło: http://arkadiusz-jadczyk.eu/images/tanh.png.

W kontekście pojedynczego neuronu korzystanie z sieci prowadzi do przekazania na jego wejścia określonych wartości, które są następnie wymnażane przez ustalone wcześniej wagi. Tak uzyskane wartości są sumowane i przekazywane funkcji aktywacji, która decyduje o ostatecznej odpowiedzi sieci (najczęściej

[9] R. Tadeusiewicz, *Elementarne wprowadzenie...*, op.cit., 14–18.

funkcję tę stanowi funkcja nieliniowa – w pracy została obrana funkcja tangens hiperboliczny, widoczna na rysunku 3)[10].

2.4. Szkolenie sieci

Na typową sieć składają się zazwyczaj setki, tysiące bądź w skrajnych przypadkach miliony neuronów – każdy z nich jest połączony z kolejnym tak, aby był zapewniony odpowiedni przepływ danych. W tym momencie rozważań możliwe jest przedstawienie sposobu szkolenia sieci neuronowych, który zostanie ograniczony do zaprezentowania szkolenia sieci jednokierunkowych z nauczycielem, ze względu na zastosowanie takowych metod w pracy. Standardowo proces szkolenia poprzedza zebranie wartości przestrzeni wejściowej w obrębie pojedynczych obiektów i określenie, jaka powinna być wartość na wyjściu sieci. Dane użyte do szkolenia sieci w tej fazie nazywa się wzorcami. Po zebraniu określonej liczby wzorców możliwe jest przejście do właściwego szkolenia sieci[11].

Jedną z najczęściej wykorzystywanych metod szkolenia jest metoda szkolenia z propagacją wsteczną. Proces ten polega na przekazaniu na wejście danych pochodzących z zebranych wzorców, by następnie odczytać odpowiedź sieci. Odpowiedź ta zestawiana jest ze wzorcowym wyjściem i obliczany jest błąd, rozsyłany następnie z powrotem do wszystkich neuronów, które złożyły się na powstanie tegoż błędu. W celu odpowiedniej propagacji funkcja aktywacji musi być funkcją posiadającą pochodną. Otrzymanie informacji o błędzie pozwala na wyznaczenie kierunku dopasowania wag neuronów sieci tak, aby otrzymywany błąd był coraz mniejszy. O satysfakcjonującej wartości błędu decyduje zazwyczaj programista[12].

W tym miejscu rozważań należy zwrócić uwagę na dwa istotne aspekty szkolenia sieci – kiedy błąd przyjmuje wartość zbyt wysoką bądź zbyt małą. Duży błąd sieci oznacza, że sieci nie udało się skutecznie odnaleźć zależności pomiędzy danymi wejściowymi a wyjściowymi – w rezultacie otrzymywane na wyjściu dane często mogą sprawiać wrażenie zupełnie przypadkowych. Drugi aspekt stanowi sieć, której uzyskany błąd jest bardzo mały. W przypadku rozbudowanych sieci może to oznaczać, że sieć wykorzystała posiadaną pamięć do opisu każdego wzorca z osobna, co określa się mianem uczenia „na pamięć"

[10] A.M. Arbib, op.cit., s. 3–9.
[11] R. Tadeusiewicz, *Sieci neuronowe*..., op.cit., s. 59–64.
[12] R. Tadeusiewicz, *Elementarne wprowadzenie*..., op.cit., s. 14–18.

(pozwala to zauważyć, jak ważny jest wybór odpowiedniej topologii). Sieć taka nie posiada zdolności generalizacji i zazwyczaj jest efektem niepożądanym[13].

3. Wprowadzenie do pracy z kamerą

Przed przystąpieniem do tworzenia aplikacji korzystających z kamery konieczne jest przeprowadzenie procesu odnalezienia oraz wyodrębnienia twarzy i oczu wewnątrz sekwencji obrazów pochodzących z urządzenia. Zadanie to znacznie upraszcza wykorzystanie funkcjonalności oferowanej przez bibliotekę OpenCV[14]. Przy pomocy biblioteki zostało zaimplementowanych wiele klas pozwalających na proces wykrywania i rejestrowania twarzy oraz oczu człowieka. Ważnym wymogiem jest możliwość wykonywania tejże czynności w czasie rzeczywistym, tj. określonej liczbie klatek pochodzących z kamery, tak aby zapewniona była płynność działania algorytmu oraz jego prawidłowa praca.

Stworzone algorytmy są tym skuteczniejsze, im obraz bazowy jest bardziej szczegółowy. Wiąże się to jednak z jednym, poważnym ograniczeniem – czasem wykonywania procesu. Im większej rozdzielczości jest obraz, tym większe są wymogi względem zasobów procesora. W przypadku obrazów statycznych nie jest to zazwyczaj problemem, jednakże w przypadku chęci zachowania płynności w przetwarzaniu całych sekwencji obrazów w określonym czasie konieczna stała się implementacja metody pozwalającej maksymalnie zredukować czas potrzebny na wykrycie twarzy, przy jednoczesnym zachowaniu zadowalającej płynności[15].

Optymalizacja metody wykrywania twarzy oraz oczu została zrealizowana przez zastosowanie procesu skalowania obrazu natywnego. Podejście takie pozwala na satysfakcjonującą redukcję czasu przetwarzania, przy jednoczesnym zachowaniu zdolności rozpoznawczych algorytmu (badania nad dobraniem obrazu o odpowiednim współczynniku skalowania zostaną zaprezentowane w dalszej części pracy). Metoda ta polega na pobraniu natywnego obrazu pochodzącego z kamery, a następnie zmniejszeniu jego rozmiarów.

---

[13] J. Żurada, M. Barski, W. Jędruch, *Sztuczne sieci neuronowe*, Wydawnictwo Naukowe PWN, Warszawa 1996.

[14] G. Bradski, A. Kaehler, *Learning OpenCV*, O'Reilly, Sebastopol 2008.

[15] R. Szeliski, *Computer Vision: Algorithms and Applications*, Springer-Verlag, Berlin 2010.

Proces wykrycia elementów jest przeprowadzany w następującej kolejności:

1. Pobranie obrazu natywnego.
2. Zmniejszenie obrazu natywnego o zadany współczynnik skalowania.
3. Odnalezienie twarzy na obrazie przeskalowanym.
4. Pobranie wycinka obrazu natywnego przedstawiającego twarz.
5. Przeskalowanie wycinka obrazu natywnego.
6. Odnalezienie oczu na przeskalowanym wycinku obrazu natywnego.
7. Odwzorowanie współrzędnych i wymiarów obiektów z płaszczyzny skalowania na płaszczyznę natywną.

4. Wykorzystanie sztucznych sieci neuronowych w procesie wykrywania pozycji oczu

4.1. Opis problematyki i zastosowane algorytmy

W trakcie tworzenia aplikacji zostało stworzonych wiele różnych modeli mających na celu wykrycie i klasyfikację pozycji oczu przez zastosowanie sieci neuronowych. Jedne z nich okazywały się bardziej skuteczne, inne mniej, aczkolwiek do momentu opracowania finalnego algorytmu każdy z nich okazywał się niewystarczająco satysfakcjonujący.

Założenia procesu są stosunkowo proste i polegają na odebraniu obrazu z kamery i przetworzeniu jego tak, aby niezależnie od odległości i warunków otoczenia można było określić, w jakim położeniu znajdują się w danym momencie oczy. Mimo precyzyjnego określenia zarówno przestrzeni wejściowej, jak i wyjściowej, sam proces określenia przez sieć funkcji przetwarzania, tj. funkcji określającej położenie oczu, dostarcza wielu trudności. Standardowe przekazanie obrazu na wejście sieci bezpośrednio z kamery wiąże się z następującymi problemami:

- Przestrzeń wejściowa, nawet przy precyzyjnym wyodrębnieniu obrazu oczu, jest bardzo duża, w kontekście przetwarzania przez sieć – praktycznie nieskończona. Przykładowo, w przypadku obrazu o rozdzielczości wynoszącej jedynie 40 na 30 pikseli tylko dla dwóch kolorów liczba wszystkich możliwych kombinacji na wejściu sieci wynosi 2^{1200}, co sprawia, że zapewnienie

pełnego pokrycia przestrzeni wejściowej nie jest możliwe (sytuacja wygląda jeszcze gorzej przy standardowej, 24-bitowej, głębi kolorów, dla której liczba kombinacji wynosi 16777216^{1200}).

- Wyniki działania algorytmów wykrycia oczu są niezwykle zmienne, podobnie jak odbierany obraz z kamery (za sprawą jakości oraz rozdzielczości przetwornika znajdującego się w kamerze), co sprawia, że nawet przy względnie statycznym obrazie współrzędne odnalezionych oczu również się przemieszczają w określonych granicach. Możliwe jest przygotowanie specjalnego algorytmu pozycjonującego raz wykryte oczy, jednakże jednym z wymogów aplikacji jest zapewnienie wykrycia oczu niezależnie od ruchu i pozycji osoby będącej przed kamerą. W podobny sposób została odrzucona metoda statycznego określenia na wstępie programu pozycji oczu.

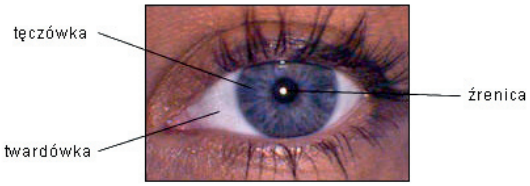
W różnych systemach mogą zostać użyte różne kamery, o zupełnie innych charakterystykach oraz specyfice pracy.

W związku z napotkanymi problemami postanowiono zawęzić operacje na obrazie z kamery jedynie do następujących obrazów:

- rozdzielczość wyjściowa wynosi 40 na 30 pikseli – wartość ta została obrana doświadczalnie na podstawie rozmiaru oczu przy standardowej odległości od ekranu monitora; standardowo wykryta przestrzeń jest bardziej nieregularna, aczkolwiek zazwyczaj większa od przyjętej rozdzielczości, co pozwoliło na efektywne zastosowanie skalowania obrazu oczu do żądanej rozdzielczości,
- głębia kolorów została zredukowana do odcieni szarości (tj. do głębi 8-bitowej), co zostało uwarunkowane ograniczeniem przetwarzania i analizy obrazu tylko do jednej składowej.

Uzyskany obraz zostaje poddany operacji rozciągnięcia na całą dostępną przestrzeń kolorów (zapewnione jest pełne pokrycie histogramu).

Po zastosowaniu powyższych instrukcji raz jeszcze rozpoczęto badania mające na celu satysfakcjonujące wyszkolenie sieci. Dobór odpowiednich algorytmów był uwarunkowany próbą odzwierciedlenia metod określania pozycji oczu przez człowieka, który radzi sobie z tym problemem bardzo dobrze, nawet w niezwykle zmiennych warunkach. Udało się ustalić, iż skuteczne może się okazać przekazanie na wejście sieci informacji na temat jasności kolorów w obrębie różnych stref obrazu. Spostrzeżenie to wynika z faktu, że istnieje wyraźna granica pomiędzy tęczówką a twardówką oka, co przedstawia rysunek 4.



Rysunek 4. Schemat budowy oka

Źródło: http://zdroow.pl/wp-content/uploads/2012/11/Budowa-oka-%C5%BArenica2.jpg.

Początkowo założono podział obrazu na kilkanaście bloków, a następnie określenie dla każdego z nich średniego odcienia szarości. Rozwiązanie takie pozwoliło na znaczne poprawienie efektywności działania sieci, jednakże – jak się później okazało – udało się opracować bardziej efektywną technikę. Polega ona na obliczeniu w obrębie oka dwóch parametrów głównych i czterech pomocniczych. Parametry główne stanowią wypadkowe odpowiednio czarnego i białego koloru. Parametry pomocnicze również stanowią wypadkowe koloru czarnego i białego (po dwie dla każdego z kolorów), ale w kontekście odległości od wypadkowych głównych. Przyjęcie takiego podejścia skutkuje podzieleniem przestrzeni na dwa podzbiory – znajdujący się w pobliżu wypadkowej głównej oraz znajdujący się w jej znacznej odległości, dla których następnie są obliczane wspomniane wyżej wypadkowe.

W związku z odmiennym kształtem i zachowaniem się lewego i prawego oka zostały utworzone dwie niezależne sieci neuronowe, których wyniki zdecydują o przyjętej pozycji oczu. Omówienie opracowanego algorytmu kończy przedstawienie możliwości sieci neuronowych w kontekście pracy z kamerą. Mimo olbrzymich możliwości sieci często zdarzają się sytuacje, w których brak dostępu do pełnej przestrzeni wejściowej skutecznie uniemożliwia prawidłowe dostosowanie się sieci do problemu – w takich sytuacjach niezbędne okazuje się wstępne przygotowanie danych wejściowych. Jednym z istotnych ograniczeń jest również prędkość współczesnych procesorów i ilość zasobów pamięci.

Utworzona aplikacja pozwoliła na zestawienie możliwości sztucznych sieci neuronowych z analizą i wykorzystaniem danych pochodzących z kamery w kontekście praktycznej aplikacji.

4.2. Wyniki badań

Przeprowadzone badania pozwoliły na dobranie zarówno odpowiedniej struktury sieci neuronowej, jak i metody określania pozycji oczu na podstawie obrazu uzyskanego z kamery. W procesie ustalania odpowiedniej metody do-

konano implementacji i sprawdzenia wielu różnych algorytmów – zestawienie to prezentuje tabela 1.

Tabela 1. Zestawienie przetestowanych technik rozpoznawania pozycji oczu

Lp.	Zastosowana metoda wyznaczania pozycji oczu	Rezultat działania	Efektywność względna
1.	przekazanie całego obrazu oka (odcienie szarości)	bardzo niska efektywność, chaotyczna klasyfikacja	–
2.	przekazanie całego obrazu oka (czarno-biały)	niska efektywność, chaotyczna klasyfikacja	–
3.	podzielenie obrazu oka na regiony kolorów	niska efektywność, widoczna pewna powtarzalność	30%
4.	obliczenie wypadkowych koloru czarnego i białego	dobra efektywność, widoczna duża powtarzalność	80%

Źródło: opracowanie własne.

W tabeli zostały wyszczególnione tylko niektóre spośród wszystkich zaimplementowanych metod wyznaczania pozycji oczu. W wyniku prowadzonych prac stwierdzono, że sieć neuronowa jednokierunkowa, niezależnie od struktury, nie potrafiła wyznaczyć pozycji oczu jedynie na podstawie dostarczonego obrazu – szkolenie sieci w tym kierunku prowadziło do otrzymania bardzo niskiej efektywności działania przy niezwykle chaotycznej klasyfikacji w kontekście wyznaczania pozycji. Wszystko to świadczy o ograniczonych zdolnościach sieci, które nie potrafią wyznaczyć zależności pomiędzy ogromną zmiennością przekazywanych im danych wejściowych.

Nieco lepsze rezultaty dała metoda podziału obrazu oka na kilka mniejszych regionów i obliczenia dla każdego z nich składowej wypadkowej kolorów, jednak również ta metoda nie zapewniała satysfakcjonującej efektywności działania, mimo występowania pewnej powtarzalności w procesie klasyfikacji.

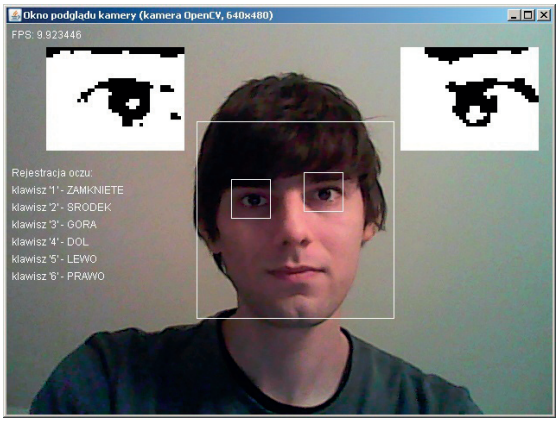
Ostatnią metodę stanowi wspomniana i opisana wcześniej metoda obliczania wypadkowych koloru białego i czarnego wewnątrz obrazu. Tego typu podejście pozwala na zastosowanie dość prostego obliczeniowo algorytmu i jednocześnie zapewnia szybkie działanie ze względu na niewielkie rozmiary sieci neuronowej. Po wielu testach metoda ta została wybrana jako główna metoda ustalania pozycji oczu w programie.

5. Opis aplikacji pozwalającej osobie sparaliżowanej na wprowadzanie tekstu

Możliwości sztucznych sieci neuronowych w zakresie komunikacji z użytkownikiem przedstawia aplikacja pozwalająca na wprowadzanie tekstu na ekran komputera jedynie przy pomocy ruchu gałek ocznych. Tego typu rozwiązanie pozwala na sterowanie komputerem bez kontaktu fizycznego, co jest szczególnie istotne w przypadku osób sparaliżowanych, których ruchy są często ograniczone jedynie do ruchu oczami.

5.1. Aplikacja rejestratora pozycji oczu

W celu stworzenia docelowej aplikacji rozpoznającej ruchy gałki ocznej niezbędne stało się stworzenie aplikacji pomocniczej, pozwalającej na zebranie danych o wychyleniu i kształcie oczu, a następnie zapisaniu ich na komputerze w postaci trwałej. Zebrane w ten sposób dane będą mogły następnie posłużyć sieci neuronowej, stanowiąc zbiór danych wejściowych (podczas gdy stan wyjścia jest wstępnie określany przez użytkownika). Podejście takie sprawia, że uruchomienie i kalibracja programu w obecnej postaci muszą być w początkowej fazie nadzorowane przez osobę mającą dostęp do klawiatury komputera. Po jednorazowej konfiguracji i zebraniu danych na temat oczu korzystanie z aplikacji rejestratora danych jest opcjonalne, jednak może się okazać pomocne w przypadku pracy w warunkach oświetlenia znacznie różniących się od tych, względem których zebrane zostały dane. Okno działającej aplikacji rejestrującej dane o pozycji oczu przedstawia rysunek 5.



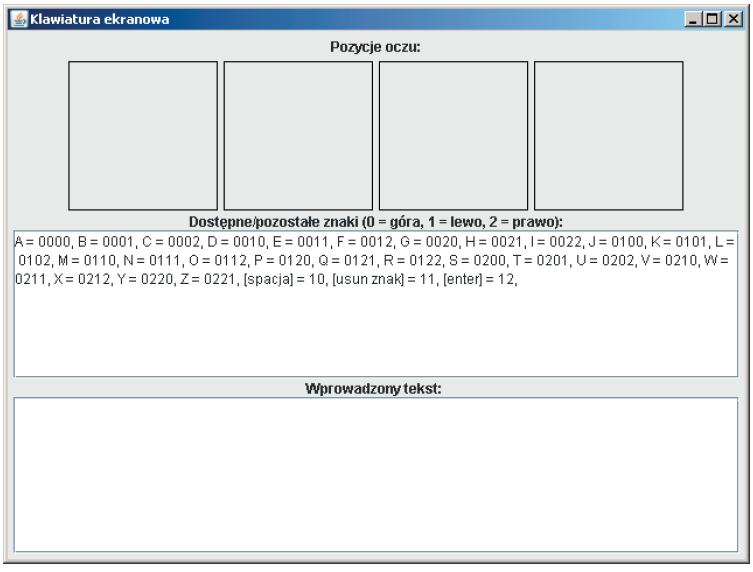
Rysunek 5. Okno aplikacji rejestratora pozycji oczu

Źródło: opracowanie własne.

Proces rejestracji poszczególnych pozycji oczu polega na spojrzeniu użytkownika w danym kierunku i naciśnięciu odpowiedniego przycisku z poziomu klawiatury komputera. W procesie rejestracji w każdej sekundzie działającego programu jest zazwyczaj tworzonych kilkanaście klatek przedstawiających obraz wykrytych oczu. Dane te są następnie zapisywane w oddzielnych plikach w pamięci trwałej komputera. Program ze względu na swoje bardziej uniwersalne działanie umożliwia rejestrację sześciu różnych pozycji oczu, natomiast w aplikacji docelowej będą jedynie wykorzystywane cztery z nich. Należy w tym miejscu również zaznaczyć, iż o ile w obecnej postaci przynajmniej jednokrotne uruchomienie programu rejestratora jest niezbędne do prawidłowego rozpoznawania oczu, o tyle przyszłe implementacje zakładają stworzenie w pełni zautomatyzowanego systemu rejestracji pozycji oczu, tak aby osoba sparaliżowana mogła dokonać kalibracji samodzielnie.

5.2. Aplikacja wirtualnej klawiatury

Po przeprowadzonym procesie rejestracji poszczególnych pozycji oczu możliwe jest przejście do omówienia aplikacji pozwalającej na praktyczne wykorzystanie tychże danych. Podobnie jak w przypadku aplikacji rejestratora, również na aplikację główną składa się jedynie jedno okno (tak aby udostępniony był jak najprostszy interfejs dla osób sparaliżowanych). Widok przykładowego okna aplikacji zaraz po uruchomieniu przedstawia rysunek 6.



Rysunek 6. Okno aplikacji klawiatury ekranowej zaraz po uruchomieniu

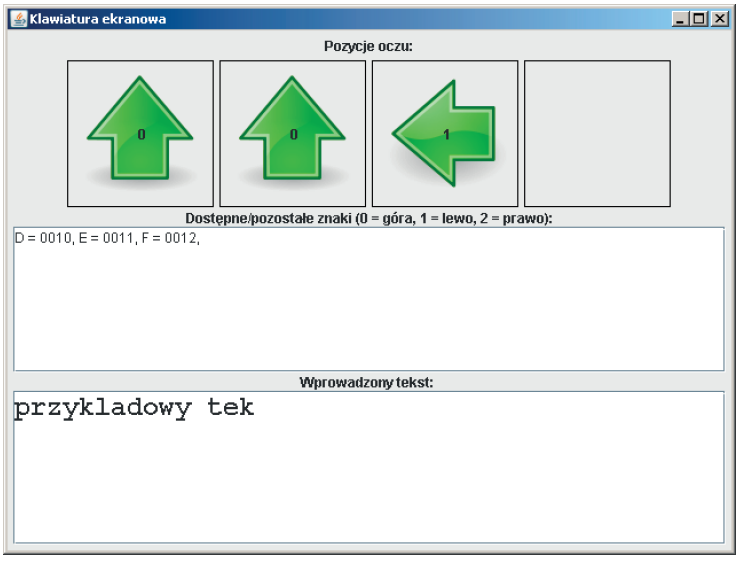
Źródło: opracowanie własne.

W aplikacji można wyróżnić trzy zasadnicze bloki, w skład których wchodzą:

- wykryte pozycje oczu,
- lista pozostałych znaków,
- pole wyświetlające wprowadzony przez użytkownika tekst.

W tym miejscu rozważań należy przedstawić sam sposób wprowadzania i modyfikacji znaków. Jak można zauważyć na rysunku 6, w momencie uruchomienia programu użytkownik otrzymuje do dyspozycji wszystkie znaki alfabetu oraz trzy akcje specjalne (wstawienie spacji, wstawienie nowej linii, usunięcie wprowadzonego znaku), co przedstawia pole określone etykietą „Dostępne/pozostałe znaki".

Aplikacja na bieżąco monitoruje pozycję oczu użytkownika – w momencie wykrycia zmiany pozycji oczu fakt ten jest rejestrowany i przedstawiany użytkownikowi w jednym z pól widniejących pod etykietą „Pozycje oczu". Każda kolejna zarejestrowana pozycja oczu prowadzi do zawężenia listy dostępnych znaków do wprowadzenia. Fakt ten ilustruje rysunek 7, który przedstawia okno programu poddanego dłuższemu użytkowaniu.



Rysunek 7. Okno aplikacji głównej poddanej dłuższemu użytkowaniu

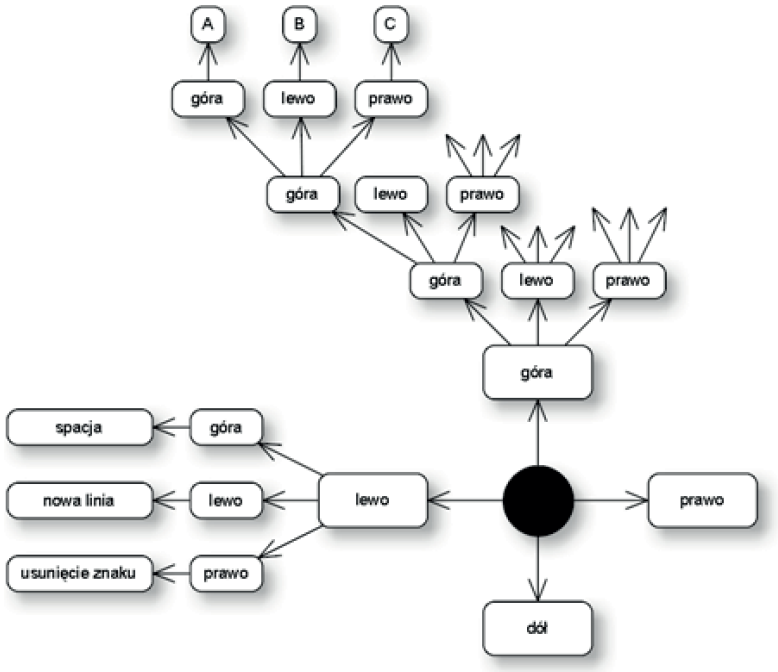
Źródło: opracowanie własne.

Na przedstawionym rysunku widać, że zbiór znaków został ograniczony do liter poprzedzonych prefiksem góra–góra–lewo – w tym przypadku są to trzy

kolejne litery: D, E oraz F. Wprowadzenie kolejnego znaku wieńczy sekwencję, co prowadzi do wprowadzenia znaku do odpowiedniego pola. W przypadku, gdy pozycja oka nie zostanie prawidłowo zinterpretowana bądź użytkownik będzie chciał przerwać proces wprowadzania określonego znaku, możliwe jest zamknięcie oczu bądź spojrzenie w dół, co skutkuje wyczyszczeniem wszystkich zarejestrowanych pozycji oczu (w związku z wystąpieniem niezdefiniowanego prefiksu).

W celu ułatwienia i ujednolicenia metody wprowadzania tekstu zdecydowano się na odpowiednie pogrupowanie poszczególnych znaków. W związku z tym, że zamknięcie oczu oraz spojrzenie w dół jest interpretowane jako anulowanie wprowadzania znaku, do dyspozycji pozostały trzy uwzględniane pozycje: spojrzenie w górę, w lewo oraz w prawo. Pewnego wyjaśnienia może wymagać również sama metoda przypisania odpowiednich kombinacji do poszczególnych znaków. W skład angielskiego alfabetu wchodzi 26 liter, co idealnie zawiera się w wartości 3^3, umożliwiając wpisanie każdej litery przy użyciu jedynie trzech ruchów oczami. Jednakże w celu umożliwienia wykonywania dodatkowych operacji i wprowadzania dodatkowych znaków specjalnych konieczne stało się rozszerzenie początkowych trzech kroków do kroków czterech (co daje liczbę 3^4, która pozwala na zdefiniowanie 81 różnych znaków/akcji). Tym sposobem wszystkie znaki alfabetu są poprzedzone w drzewie decyzyjnym prefiksem 0 (tj. rozpoczynają się od spojrzenia w górę), natomiast zdefiniowane wstępnie akcje specjalne są dostępne z poziomu prefiksu 1 (spojrzenie w lewo). Zastosowanie takiego podejścia umożliwiło również skrócenie dostępu do akcji specjalnych (wśród których znajduje się wstawienie klawisza spacji, nowej linii bądź usunięcie wprowadzonego wcześniej znaku), których częstotliwość wywoływania jest znacznie wyższa aniżeli poszczególnych znaków alfabetu. Poglądowy schemat wprowadzania poszczególnych znaków został przedstawiony na rysunku 8.

Jak można zauważyć, usunięcie znaku bądź wstawienie odstępu między znakami wymaga jedynie dwóch ruchów oczami, co znacznie ułatwia zarządzanie aplikacją.



Rysunek 8. Fragment drzewa wyboru dla kolejnych pozycji oczu
Źródło: opracowanie własne.

W ramach stworzonej aplikacji zostało przeprowadzonych wiele testów, które przy zastosowaniu zgromadzonych wcześniej danych wzorcowych pozwalały na jednoznaczne i precyzyjne wprowadzanie nowych znaków – po krótkiej pracy z programem udało się uzyskać procent błędów wahający się od 10% do 20%. Oznacza to, że w przeprowadzonym teście na 10 wprowadzanych znaków zazwyczaj jedynie jeden bądź dwa z nich zostały wprowadzone nieprawidłowo – w reszcie przypadków aplikacja skutecznie rozpoznała wykonywane przez użytkownika ruchy, umożliwiając wpisywanie tekstu, co potwierdza działanie programu.

5.3. Metoda doboru odpowiedniej struktury sieci neuronowej

Na potrzeby systemu został utworzony specjalny mechanizm odpowiedzialny za dobór struktury sieci jednokierunkowej adekwatny do postawionego problemu. Zasada działania polega na podzieleniu zadanej przestrzeni wejściowej na dwa odrębne zbiory, z których jeden posłuży do szkolenia sieci, drugi natomiast – badania odpowiedzi sieci. W myśl tej zasady są sprawdzane kolejne topologie

sieci – proces ten wieńczy wybranie struktury sieci zwracającej najmniejszy błąd dla danych weryfikacyjnych. Długotrwałe działanie mechanizmu pozwoliło ustalić, że satysfakcjonujący rezultat uzyskuje się dla sieci o strukturze 12–10–6, co oznacza, że na sieć składa się łącznie 28 neuronów przy zastosowaniu jednej warstwy ukrytej. Wstępne szkolenie sieci dało satysfakcjonujący rezultat błędu na poziomie 1–2%, co stanowi o skuteczności opracowanego algorytmu.

6. Podsumowanie i kierunki dalszych badań

Korzystając z możliwości sztucznych sieci neuronowych, utworzono praktyczną aplikację pozwalającą na interakcję użytkownika z komputerem przy pomocy standardowej kamery VGA – w tym celu zostały przeprowadzone badania w kontekście doboru odpowiednich algorytmów tak, aby efektywność wprowadzanych znaków była jak najwyższa. Do głównych zalet aplikacji należą:

- Niski koszt – aplikacja została stworzona z myślą o zastosowaniu standardowej kamery operującej na rozdzielczości VGA. Kamery tego typu są powszechnie dostępne (obecnie stają się nieodłącznym elementem laptopów), co sprawia, że koszt wytworzenia jest niski.
- Dostępność – w związku z niskim kosztem sprzętu i niskim wymaganiami sprzętowymi aplikacja może zostać uruchomiona na większości używanych dzisiaj komputerów (jak również na telefonach komórkowych czy tabletach).
- Przenośność – bezpośrednio związana z dostępnością. Oznacza, że sama aplikacja może zostać uruchomiona w różnych środowiskach i przy różnych konfiguracjach sprzętowych (za sprawą utworzenia aplikacji na platformie Java).

Wśród dalszych możliwości rozwoju aplikacji można wyróżnić m.in.:

- Zintegrowanie procesu kalibracji i zbierania danych na potrzeby szkolenia sztucznych sieci neuronowych z poziomu aplikacji głównej.
- Zaimplementowanie słownika, który pozwoli na skrócenie czasu potrzebnego do wprowadzenia kolejnych słów.
- Rozbudowę kolejnych gałęzi w drzewie decyzyjnym. Przykładowym rozwiązaniem jest utworzenie gałęzi, z której możliwy będzie dostęp do najczęściej używanych wyrazów.
- Dodanie możliwości przeczytania wprowadzonego tekstu przez syntezator systemowy.

Stworzony system – mimo swojej niezwykłej prostoty – skutecznie realizuje wymogi projektowe, zachowując przy tym wyjątkowo niski koszt wytworzenia oraz eksploatacji, co czyni go alternatywą do obecnie istniejących systemów, głównie w zakresie interakcji z osobami sparaliżowanymi.

Bibliografia

1. Arbib A.M., *The Handbook of Brain Theory and Neural Networks*, The MIT Press, Cambridge 2003.
2. Bradski G., Kaehler A., *Learning OpenCV*, O'Reilly, Sebastopol 2008.
3. Cesar S.M., Sang-Woon K., *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, Springer-Verlag, Berlin 2011.
4. Nilsson J., *Artificial Intelligence: A New Synthesis*, Morgan Kaufmann Publishers, San Francisco 1998.
5. Osowski S., *Sieci neuronowe*, Oficyna Wydawnicza Politechniki Warszawskiej, Warszawa 1994.
6. Rutkowski L., *Metody i techniki sztucznej inteligencji*, Wydawnictwo Naukowe PWN, Warszawa 2005.
7. Szeliski R., *Computer Vision: Algorithms and Applications*, Springer-Verlag, Berlin 2010.
8. Tadeusiewicz R., *Elementarne wprowadzenie do techniki sieci neuronowych z przykładowymi programami*, Akademicka Oficyna Wydawnicza PLJ, Warszawa 1998.
9. Tadeusiewicz R., *Sieci neuronowe*, Akademicka Oficyna Wydawnicza RM, Warszawa 1993.
10. Żurada J., Barski M., Jędruch W., *Sztuczne sieci neuronowe*, Wydawnictwo Naukowe PWN, Warszawa 1996.

Źródła sieciowe

1. http://aragorn.pb.bialystok.pl/~radev/ai/se/zal/ready/krzyszkowski_pliki/image004.gif.
2. http://arkadiusz-jadczyk.eu/images/tanh.png.
3. http://www.statsoft.pl/textbook/graphics/neuralnw.gif.
4. http://zdroow.pl/wp-content/uploads/2012/11/Budowa-oka-%C5%BArenica2.jpg.

* * *

Using artificial neural networks to identify eye movement and as a communication support for paralysed people

Summary

The paper describes the design and implementation of a system used to help paralysed people to interact with a personal computer via a VGA camera. The in-built software uses a virtual keyboard which is controlled only by the user's eye movement. The software resorts to artificial neural networks to analyse and process the data in the background. The purpose of the solution is to create a cheap, simple and functional device for human-machine interaction.

Keywords: eye movement, communication support, neural networks, paralysed people, human-machine interaction