POZNAN UNIVERSITY OF TECHNOLOGY ACADEMIC JOURNALS
No 91 Electrical Engineering 2017

DOI 10.21008/j.1897-0737.2017.91.0015

Paweł LEWANDOWSKI*
Mateusz PÓŁTORAK*
Mateusz WAGNER*
Janusz POCHMARA*
Andrzej RYBARCZYK*

SYSTEM WSPOMAGAJĄCY ROZPOZNAWANIE ZNAKÓW JĘZYKA MIGOWEGO OPARTY NA SZTUCZNEJ SIECI NEURONOWEJ

W niniejszym artykule zaproponowano realizację systemu wspomagajacego rozpoznawanie statycznych znaków języka migowego. Na potrzeby rozwiązania skorzystano z sensora Microsoft Kinect XBOX 360, przygotowano oprogramowanie umożliwiające translację znaków dla osób nie znających tego języka, oparte na sztucznej inteligencji, przetworzono otrzymane informacje oraz utworzono zbiór danych pozwalający na ich poprawną klasyfikację. Istotnym faktem jest również wybranie najbardziej optymalnego rozwiązania, zarówno pod względem możliwości wydajnościowych przeciętnego komputera osobistego jak i efektywności działania systemu.

SŁOWA KLUCZOWE: Microsoft Kinect, sztuczne sieci neuronowe, theano, sieci konwolucyjne, detekcja obrazu

1. WSTĘP

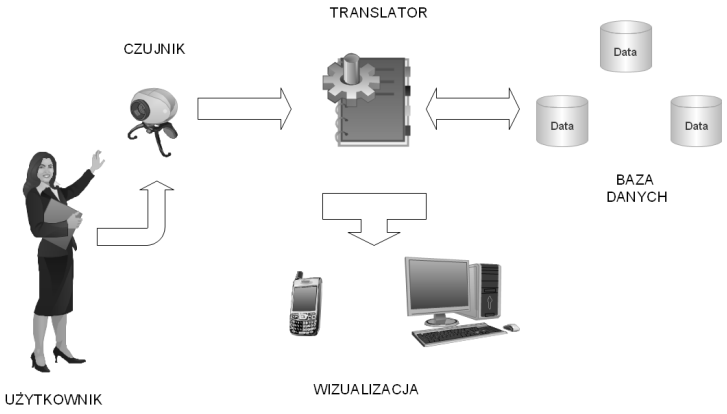
Rozpoznawanie obrazów oraz przetwarzanie danych przy pomocy sztucznej inteligencji, zachęca do poszukiwania nowych, użytecznych rozwiązań. Sieci neuronowe są wzorowane na sposobie pracy struktur mózgowych. Jest to możliwe w związku z rozwojem nauk oraz połączeniem biologii, fizyki, matematyki. Dzięki nieustającym badaniom architektury ludzkiego mózgu, a także pracom nad dogłębnym poznaniem układu nerwowego powstało narzędzie, które w najbliższym czasie będzie w stanie przewyższyć, a może nawet całkowicie zastąpić mniej wydajne odpowiedniki rozwiązywania złożonych zagadnień. Oczywiście nie powinno się z góry uznawać tego za pewnik, gdyż jak wszystkie inne metody ta także posiada pewne wady, m.in. potrzeba naprawdę sporych zasobów materiałów uczących sieć, szczególnie dla nietrywialnych problemów.

* Politechnika Poznańska.

2. ARCHITEKTURA SYSTEMU

Konstrukcję systemu detekcji symboli języka migowego na postać alfanumeryczną postanowiono oprzeć na diagramie przedstawionym na rys. 1. Koncepcyjnie oparto się na technologii w postaci ang. *Human Machine Interface*. Proponowany system składa się z czterech głównych bloków:

- czujnika odpowiedzialnego za komunikację z użytkownikiem (bezpośredni odczyt danych, wsparcie po stronie czujnika w przetwarzaniu sprzętowym obrazu), możliwość odczytu na podstawie głębokości obrazu – CZUJNIK,
- translator, system który w czasie rzeczywistym jest odpowiedzialny za podejmowanie decyzji w procesie komunikacji (sztuczna inteligencja, oparta na sztucznych sieciach neuronowych, architektura dobrana na potrzeby rozwiązania problemu odczytu znaków dla osób głuchoniemych) – TRANSLATOR,
- urządzenia pozwalającego przedstawić sygnały znakowe w postaci alfanumerycznej (smartfon/komputer), aplikacja w formie okienkowej dla systemu operacyjnego Windows/Android – WIZUALIZACJA,
- bazy danych, na podstawie, której dokonano ewaluacji działania system eksperckiego (główny człon aplikacji, elementy bez których nie może istnieć właściwe działanie system – nauka/interpretacja), szybki dostęp do elementów składowych jest niezbędnikiem – BAZA DANYCH.



Rys. 1. Diagram system konwersji symbol na znak alfanumeryczny

3. SIEĆ NEURONOWA

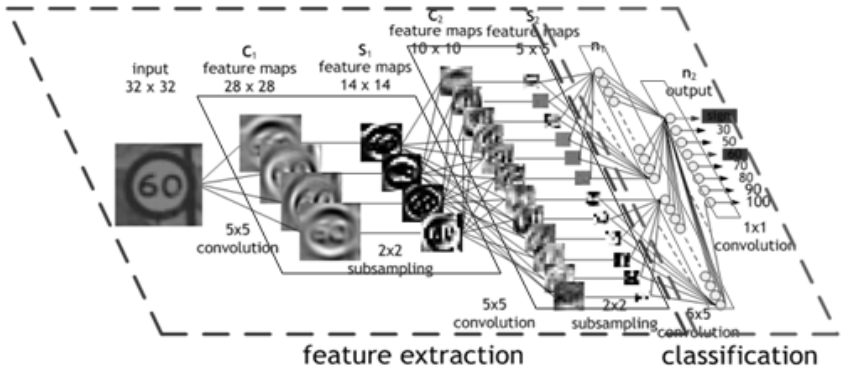
Sztuczne sieci neuronowe to struktury matematyczne naśladujące komórki nerwowe. Przetwarzają dane wejściowe w taki sposób by na wyjściu dokonać ich klasyfikacji. Dzieje się to za pośrednictwem połączeń neuronowych decydujących o przepływie sygnału w sieci. Aby sieć mogła poprawnie klasyfikować

problemy jej postawione, musi wcześniej zostać odpowiednio nauczona. W trakcie nauki wartości parametrów sieci (tj. połączeń) są modyfikowane w taki sposób by na wyjściu sieci otrzymywać jak najbardziej zbliżone wartości do oczekiwanych. Po zakończonym procesie nauki, zakładając że przebiegł on poprawnie oraz że struktura sieci jest wystarczająco rozbudowana by sprostać postawionemu problemowi, zaprojektowany model powinien poprawnie przetwarzać dostarczone mu sygnały wejściowe.

Aktualnie najbardziej efektywną formą połączeń wagowych jaką stosuje się w modelach sieci służących do rozpoznawania obrazów, są połączenia będące jednocześnie wartościami elementów filtru. Idea ta narodziła się, gdy Yann LeCunn wprowadził operację splotu do obszaru sieci neuronowych [1]. Tak powstały sieci konwolucyjne.

3.1. Architektura

Architekturę i budowę sztucznych sieci neuronowych zawarto już w wielu publikacjach, dlatego w niniejszym artykule umieszczono jedynie krótki zarys działania sieci konwolucyjnej. Sieci konwolucyjne to struktury trójwymiarowe, a nie tak jak w przypadku sieci w pełni połączonych – jednowymiarowe. Następne warstwy neuronowe można zdefiniować jako kolejne etapy przetworzonego obrazu, przy czym jeden neuron warstwy bieżącej jest połączony z $n * n$ neuronami warstwy poprzedniej (gdzie n to rozmiar filtru konwolucji). W modelach tych stosuje się określoną wcześniej ilość filtrów (map cech) dla każdej warstwy równą m, tak więc jedna macierz danych wejściowych zostaje przetworzona na m sposobów. Na rys. 2. przedstawiono przykładową architekturę sieci konwolucyjnej z trzema warstwami splotowymi o rozmiarze filtru 5x5. Pierwsza warstwa splotowa składa się z czterech map cech, a druga z dwunastu.



Rys. 2. Przykładowa architektura sieci konwolucyjnej [2]

3.2. Rozwiązania sprzętowe

Do zamodelowania sztucznej sieci neuronowej użyto ogólnodostępnej, open-source'owej biblioteki Theano [3] przeznaczonej dla języka Python w wersji 2.7. Podstawowymi kryteriami wyboru biblioteki oraz języka oprogramowania były:
- ilość dostępnych wzorców przedstawionych w sieci Internet,
- obsługa przez system operacyjny Windows 7,
- zgodność z bibliotekami pozwalającymi na dostęp do zasobów kart graficznych NVIDIA,
- prostota implementacji, ilość gotowych funkcji odpowiadających za modyfikacje spadków gradientowych, warstwy konwolucyjne, funkcje aktywacji,
- biblioteka NumPy pozwalająca na sprawne i szybkie przetwarzanie macierzy.

Zaprojektowana sieć składa się z trzech warstw konwolucyjnych i trzech warstw max–pooling , za nimi znajdują się dwie warstwy w pełni połączone. Nieliniową funkcją aktywacji warstw ukrytych jest Rectified Linear Unit [4], funkcją warstwy wyjściowej jest funkcja softmax. Ze względu na niewielką ilość zbioru uczącego zastosowano technikę dropout [5], pozwalającą na redukcję nadmiernego dopasowania.

Do procesu uczenia wykorzystano spadki gradientowe, zmodyfikowane algorytmem przyspieszającym RMSprop.

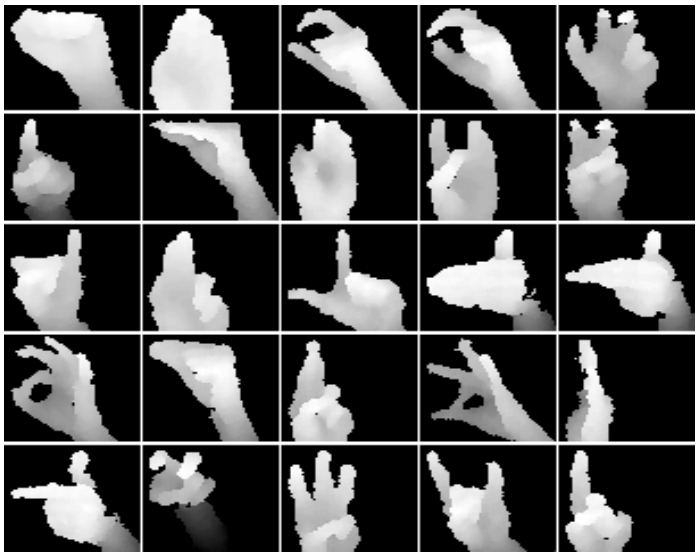
$$E[g^2]_t = 0{,}9E[g^2]_{t-1} + 0{,}1g_t^2 \tag{1}$$

Wartość parametru learning rate i parametru γ wynosi odpowiednio 0,001 oraz 0,9. Wielkości te proponowane są w większości artykułów i opracowań na temat optymalizacji spadku gradientowego, między innymi w wykładzie Geoff'a Hintona z wydziału inżynierii komputerowej uniwersytetu w Toronto [6]. Uczenie odbywa się za pomocą metody mini–batch [7]. Oznacza to że funkcja błędu nie jest obliczana na podstawie wszystkich elementów zbioru treningowego, ale na podstawie mniejszych pakietów tego zbioru.

3.2. Kompozycja i sposób uzyskania danych

Dane wykorzystane w pracy to skany ludzkich dłoni, a dokładniej znaki manualne używane do komunikacji w języku migowym, przedstawione w postaci kwadratowych map bitowych o rozdzielczości 60x60 px. Doboru rozmiaru skanów dokonano empirycznie, poprzez próby pozyskiwania obrazów dłoni o skrajnych rozmiarach, tak żeby 'tło' dłoni stanowiło jak najmniejszy procent całości obrazu. Innym czynnikiem wpływającym na wybór rozdzielczości obrazu była moc obliczeniowa przeciętnego komputera osobistego lub smartfona. Wyważenie odpowiedniej proporcji pomiędzy dostateczną jakością materiału cyfrowego, a jego bezproblemowym przetwarzaniem było tutaj kluczowe.

Do pracy niezbędny był zbiór danych składający się z wielokrotnych przykładów obrazów dokładnie dwudziestu pięciu znaków polskiego alfabetu migowego. Na tę liczbę składały się zarówno znaki niedynamiczne oraz dynamiczne o swoistej niepowtarzalności (tj. takie które nie występowały w postaci niedynamicznej). Zaznaczyć należy, że program definiuje wszystkie znaki jako statyczne. Rozpoznawanie ruchów to złożony problem, który nie należy do trywialnych rozwiązań. Przyjęto jednak, że do zbioru dodane zostaną wszystkie unikalne znaki, tak by możliwa była późniejsza definicja gestów. Przykładowo, znak oznaczający literę ‘A’ jest statyczny, ale jego dynamiczna wersja to litera ‘Ą’. Program klasyfikuje ten znak zawsze jako statyczny, dlatego nigdy nie zwraca litery ‘Ą’.



Rys. 3. Przykładowy zbiór uczący

Zestawy otrzymano poprzez spotkania z ludźmi, o danej charakterystyce dłoni (mniej lub bardziej oryginalnej) i zeskanowaniu tych dłoni za pomocą Kinect’a wraz ze stworzonym oprogramowaniem. Na samym początku pozyskiwania materiału pobierano jeden zestaw (tj. dwadzieścia pięć unikalnych znaków) od jednej osoby. Tak stworzono pierwsze 10 zestawów i otrzymano 250 obrazów. Cel stanowiło około 6250 obrazów. W wyniku analizy czasu potrzebnego do stworzenia danego zestawu (na co składały się nie tylko warunki techniczne, a także skłonność wybranych osób do współpracy, ich liczba, oraz umiejętności manualne) podjęto decyzję o stopniowym zwiększaniu ilości tworzonych obrazów dłoni przypadających na jedną osobę. W połączeniu z dodawaniem materiału uczącego do sieci na bieżąco, dało to bardzo dobre wyniki.

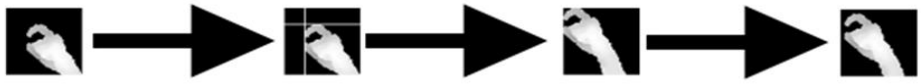
Już przy około trzech tysiącach sztuk skanów sieć uzyskiwała zadowalające wyniki nauki (tj. bardzo mały błąd), co za tym idzie, sam program nie miał problemu z rozróżnieniem znaków pokazywanych przez jego twórców.

Oprócz dużej liczby elementów niezbędnej dla zbioru uczącego konieczne było odpowiednie skonfigurowanie wymiarów tych elementów. Niezwykle ważnym czynnikiem było również ich zróżnicowanie. Przykłady nie do końca wyraźne, lub takie w których dany znak był pokazany pod innym kątem niż zwykle, były tak samo istotne jak te idealne. Wiąże się to oczywiście z cechą adaptacji sieci do złych warunków podczas samego rozpoznawania znaków.

4. SYSTEM WIZYJNY

System wizyjny oparto na sensorze Kinect, którego zadaniem jest dostarczenie informacji do aplikacji w postaci strumieni danych. Wszystkie trzy strumienie, do których zaliczają się strumień wideo, głębokości i audio, można kontrolować i przetwarzać korzystając z Kinect Software Development Kit, który udostępnia producent. Korzystanie ze strumieni dzieli się na trzy podstawowe etapy. Najpierw należy skonfigurować ustawienia danego strumienia, a dalej można rozpocząć odczytywanie. Następnie, gdy nowa ramka z danymi jest odebrana, w kodzie źródłowym wywoływana jest odpowiednia metoda pozwalająca na zarządzanie otrzymanymi informacjami. Ramka, w nomenklaturze Kinect API, to odpowiedni obiekt, który zawiera dane dostarczone z sensora. Ramek używa się, by uniknąć problemów z pamięcią i jej alokacją.

Ramka zawierająca informacje z kamery wideo przetwarzana jest bezpośrednio z danych numerycznych na obraz, po czym następnie zostaje wyświetlona. Dane z kamery wideo nie są wykorzystywane do pracy sieci neuronowej, a służą jedynie prezentacji położenia ciała użytkownika względem sensora. Z kamery głębokości pozyskiwane są informacje potrzebne do funkcjonowania sieci neuronowej. Dostarczona ramka zawiera dwuwymiarową tablicę o rozmiarach 320x240, a każdy element tablicy to jeden piksel obrazu. Dane z NUI dotyczące śledzenia szkieletów, tj. informacje o pozycji użytkownika, również są otrzymywane w postaci ramek. Postanowiono znormalizować otrzymywane dane tak, by do sieci neuronowej dostarczać jak najpoprawniejsze dane. Algorytm znajduje położenie najbardziej skrajnych pikseli, które nie są czarne i na tej podstawie zmienia ustawienia punktu centralnego. Ostatecznie dodawane są marginesy o rozmiarze trzech pikseli by obraz nie znajdował się na samym brzegu obrazka. Na rys. 4 przedstawiono proces normalizacji.



Rys. 4. Proces normalizacji pozycji

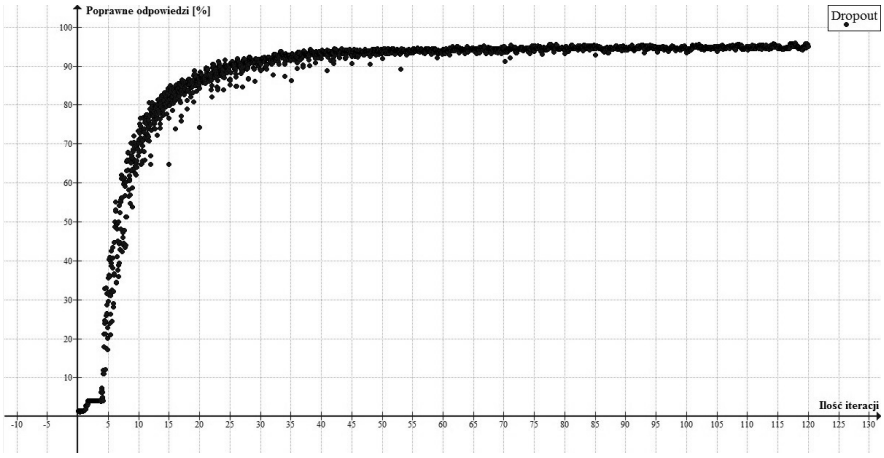
5. WYNIKI

Poniżej na rysunkach 5 – 9 zaprezentowano zależność jakości predykcji dokonywanej przez sieć od rozmiarów zbioru uczącego oraz tego czy zastosowano technikę dropout. Wszystkie otrzymane wyniki zaprezentowano w tabeli 1, dodatkowo umieszczono dwa wykresy ilustrujące jak ten proces przebiegał.

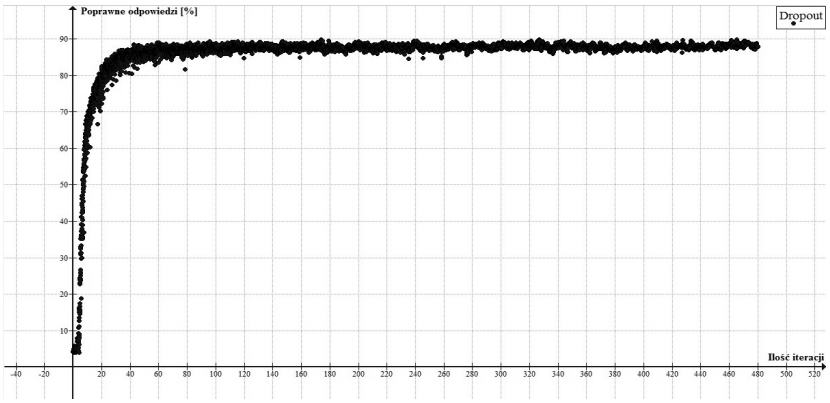
Aby odpowiednio zinterpretować wyniki niezbędne jest prawidłowe opisanie osi. Dla wykresów przedstawiających wyniki nauczania oś pionowa opisana została przez procent prawidłowych odpowiedzi sieci na zadane mapy bitowe z materiału testowego (sieć nie miała z nim wcześniej styczności), natomiast oś pozioma to liczba kolejnych iteracji programu, przy czym każda iteracja składa się z 25 pakietów, a każdy pakiet zawiera 141 map bitowych. Cząstkowe wyniki otrzymywane są po każdym jednorazowym nauczeniu się pakietu. Dla wykresów przedstawiających błąd nauczania oś pionowa prezentuje wartość zwracaną przez funkcję błędu, a oś pionowa – tak jak poprzednio, liczbą kolejnych iteracji.

Wyniki potwierdzają założenia teoretyczne. Stosowanie dropoutu zwiększyło wydajność każdego zestawu treningowego. Na wykresach jakościowych dotyczących pełnego zestawu danych widoczna jest zdecydowana poprawa stabilności między piętnastą, a pięćdziesiątą iteracją gdzie sieć przełamuje barierę 70% poprawnych predykcji. Świadczy to o dużo większej elastyczności i zdolności do adaptacji mimo konfrontacji z zupełnie nieznanymi danymi.

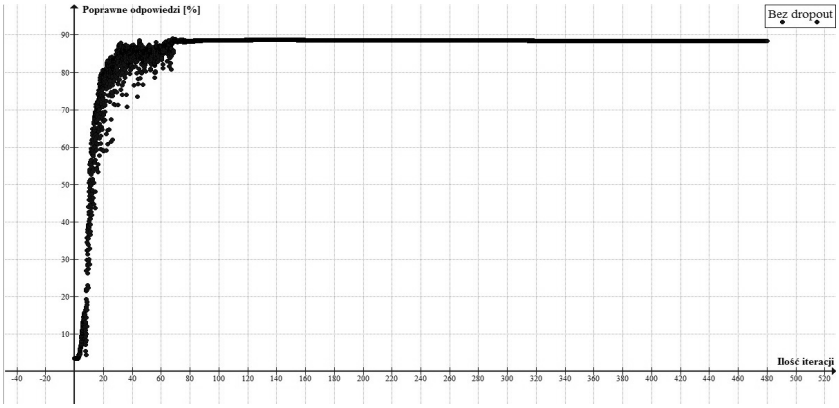
W każdym przykładzie bez dropoutu funkcja kosztu stabilizowała się na pewnej wartości, tak samo jak uzyskiwany wynik. Jest to oczywisty skutek nadmiernego dopasowania. Sieci takie osiągają swoje maksimum którego nie są już w stanie przeskoczyć – najmniejszy zestaw treningowy był przeuczony już po około sześćdziesiatej iteracji i od tego momentu nie wykazał żadnego progresu.



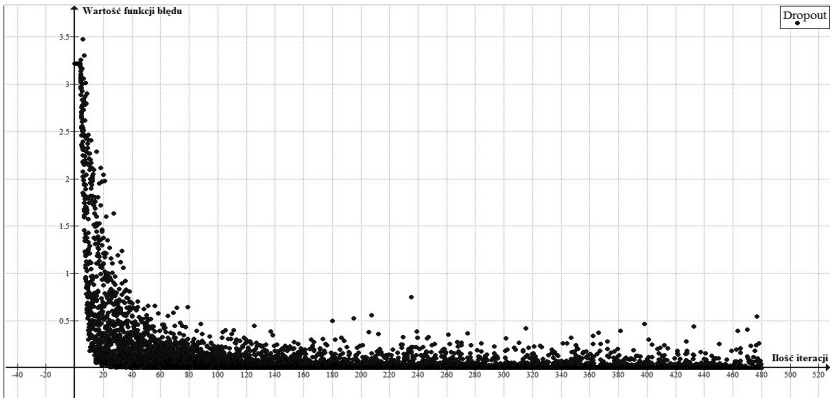
Rys. 5. Poprawne odpowiedzi, pełny zestaw, dropout



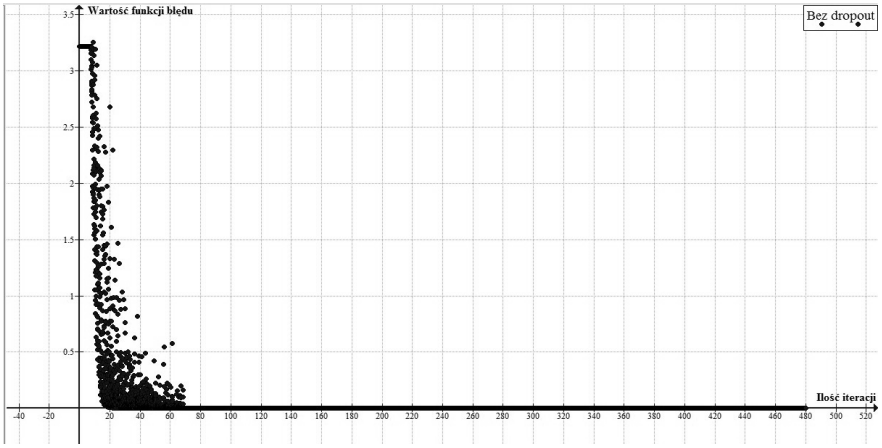
Rys. 6. Poprawne odpowiedzi, 1/4 zestawu, dropout



Rys. 7. Poprawne odpowiedzi, 1/4 zestawu, bez dropoutu



Rys. 8. Wartość funkcji błędu, 1/4 zestawu, dropout



Rys. 9. Wartości funkcji błędu, 1/4 zestawu, bez dropoutu

Tabela 1. Wyniki testów nadmiernego dopasowania

	drop or not to drop			
	Dropout		bez dropout	
Ilość przykładów / learning rate	procent	błąd	procent	błąd
3750 / 0,001	94,77573	0,026602	94,30984	0,001945
1875 / 0,001	90,55425	0,030192	89,25701	0,002881
875 / 0,001	88,38547	0,01	88,3	0

6. WNIOSKI

Zaproponowano rozwiązanie pozwalające na rozpoznawanie poszczególnych znaków alfabetu języka migowego. Dzięki wykorzystaniu splotowej sieci neuronowej stworzono program, który bez widocznego opóźnienia potrafi analizować i rozpoznawać zadane gesty.

Faktem jest, że bardzo ważną rolę pełni materiał treningowy, a szczególnie jego ilość oraz jakość wykonania. Bardzo dobrze zaprezentowały to testy na przeuczenie. Czterokrotne zwiększenie objętości unikalnych elementów uczących zwiększyło procent poprawnych odpowiedzi z 88% do 94%.

Istnieją odmienne możliwości rozwiązania zadanego problemu, opierające się przede wszystkim na zastosowaniu innych narzędzi, zarówno programistycznych jak i sprzętowych. Niewątpliwie nowsza wersja czujnika Kinect w znaczący sposób zwiększyłaby dokładność zebranego materiału (większa rozdzielczość kamer), co więcej umożliwiłaby użycie nowszej wersji pakietu SDK co

jest jednoznaczne z bardziej precyzyjnym oprogramowaniem, np. zmodernizowano proces śledzenia ciała. Alternatywnie istnieje możliwość korzystania z innego oprogramowania zostając przy starszym oprzyrządowaniu, wykorzystującym bibliotekę OpenCV do przetwarzania obrazu wraz z algorytmem śledzenia dłoni.

LITERATURA

[1] Yoshua Bengio, Geoffrey Hinton Yann LeCun, "Deep learning," *Nature*, vol. 52, p. 28, Maj 2005.
[2] (2016, Grudzień) Devblog NVIDIA. [Online]. HYPERLINK "https://devblogs.nvidia.com/parallelforall/deep–learning–nutshell–core–concepts/" https://devblogs.nvidia.com/parallelforall/deep–learning–nutshell–core–concepts/
[3] (2016, Czerwiec) Theano. [Online]. HYPERLINK "http://deeplearning.net/software/theano/index.html" http://deeplearning.net/software/theano/index.html
[4] Benjamin Graham, "Fractional Max–Pooling," University of Warwick, Warwick, WIelka Brytania, Maj 2015. [Online]. HYPERLINK "https://arxiv.org/pdf/1412.6071v4.pdf" https://arxiv.org/pdf/1412.6071v4.pdf
[5] Geoffrey E. Hinton Vinod Nair, "Rectified Linear Units Improve Restricted Boltzmann Machines," University of Toronto, Toronto, 2010.
[6] G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov N. Srivasta, "Dropout – A Simple Way To Prevent Neural Networks from Overfitting," CS Toronto, Toronto, 2014.
[7] Geoffrey Hinton. Overview of mini–batch gradient descent. Presentation.
[8] Sebastian Ruder, "An overview of gradient descent optimization," Insight Centre for Data Analytics, Dublin, 2016.

SIGNS RECOGNITION SYSTEM BASED ON ARTIFICIAL NEURAL NETWORK

In following work there is suggested a solution to recognise certain static characters from sign language. To achieve the objective, there were used tools like Microsoft Kinect and convolutional neural networks. Main problems to overcome were to collect data from Kinect sensor and prepare software based on artificial intelligence, which could process gathered material. For learning purposes around four thousand images were collected. Dataset this large was required for neural networks to work and respond properly. What is also important is to select the most optimal solution for neural networks. The influence of dropout parameter on learning process was studied too.

(Received: 14. 02. 2017, revised: 28. 02. 2017)