JOURNAL of COMPUTER SCIENCES INSTITUTE

JCSI 13 (2019) 302-309
Wysłane: 2019-07-03
Przyjęte: 2019-07-10

Zastosowanie uczenia maszynowego w budowie interfejsu sterowanego głosem na przykładzie odtwarzacza muzyki

Jakub Basiakowski*
Politechnika Lubelska, Instytut Informatyki, Nadbystrzycka 36B, 20-618 Lublin, Polska

Streszczenie. Poniższy artykuł przedstawia wyniki badań wpływu zastosowania uczenia maszynowego w budowie interfejsu sterowanego głosem. Do analizy wykorzystane zostały dwa różne modele: jednokierunkowa sieć neuronowa zawierająca jedną warstwę ukrytą oraz bardziej skomplikowana konwolucyjna sieć neuronowa. Dodatkowo wykonane zostało porównanie modeli użytych w celu realizacji badań pod względem jakości oraz przebiegu treningu.

Słowa kluczowe: uczenie maszynowe; sieć neuronowa; rozpoznawanie głosu

*Autor do korespondencji.

Adres e-mail: jakub.basiakowski@pollub.edu.pl

Applying of machine learning in the construction of a voice-controlled interface on the example of a music player

Jakub Basiakowski*
Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Abstract. The following paper presents the results of research on the impact of machine learning in the construction of a voice-controlled interface. Two different models were used for the analysys: a feedforward neural network containing one hidden layer and a more complicated convolutional neural network. What is more, a comparison of the applied models was presented. This comparison was performed in terms of quality and the course of training.

Keywords: machine learning; neural network; speech recognition

*Corresponding author.

E-mail address: jakub.basiakowski@pollub.edu.pl

1. Wprowadzenie

W dzisiejszych czasach uczenie maszynowe (ang. *machine learning*) znajduje szerokie zastosowanie w tworzonych systemach informatycznych. Różnego rodzaju aplikacje wykorzystują możliwość ciągłego rozwijania się przy użyciu dostarczanych informacji. Dzięki dużemu zainteresowaniu tą dziedziną, powstało wiele algorytmów uczenia maszynowego, które wykorzystywane są w rozwiązywaniu różnorodnych problemów. Jednym z zadań w jakim zastosowanie znajduje uczenie maszynowe jest rozpoznawanie głosu.

Celem pracy jest analiza wpływu zastosowania uczenia maszynowego w budowie interfejsu sterowanego głosem. W tym celu utworzona została aplikacja odtwarzacza muzyki, w której wybrane funkcjonalności możliwe są do wykonania poprzez wprowadzenie odpowiedniego polecenia głosowego. Proces uczenia maszynowego przeprowadzony został z wykorzystaniem dwóch różnych modeli, a następnie opracowana została analiza uzyskanych wyników oraz wykonane zostało porównanie zastosowanych w celu realizacji badań modeli.

W niniejszej pracy postawiona została teza, iż wykorzystanie sieci konwolucyjnej z odpowiednio dobranymi parametrami daje lepsze wyniki klasyfikacji niż prosta sieć jednowarstwowa.

2. Materiały i metody

2.1. Przegląd literatury

Uczenie maszynowe cieszy się dużą popularnością wśród twórców różnego rodzaju systemów. Możliwość ciągłego rozwijania aplikacji na podstawie dostarczanych do niej informacji znajduje zastosowanie w rozwiązywaniu wielu problemów, takich jak na przykład wykrywanie obiektów, rozpoznawanie twarzy czy jak w przypadku tej pracy rozpoznawanie mowy. W związku z dużym zainteresowaniem występującym wokół dziedziny uczenia maszynowego powstało wiele algorytmów, dzięki którym może zostać ono wykorzystane.

W pracy [1] uczenie maszynowe zostało wykorzystane w rozpoznawaniu mowy, a konkretnie do klasyfikacji fonemów. Autorzy przedstawiają rozwiązanie problemu z użyciem histogramów zrekonstruowanej przestrzeni fazowej. W celu realizacji badań wykorzystany został statystyczny algorytm uczenia maszynowego – naiwny klasyfikator Bayesa, który wykorzystuje prawdopodobieństwo do przyporządkowania danych wejściowych. Autorzy pracy przeprowadzili również analizę wpływu normalizowania

danych wejściowych, z której wynika iż dla danych poddanych normalizacji poprawność klasyfikacji była wyższa. Naiwny klasyfikator Bayesa wykorzystany został również w pracy [2]. W tym przypadku zastosowany został jako klasyfikator dwuklasowy w celu weryfikacji czy słowa w rozpoznanym zdaniu zostały wykryte poprawnie.

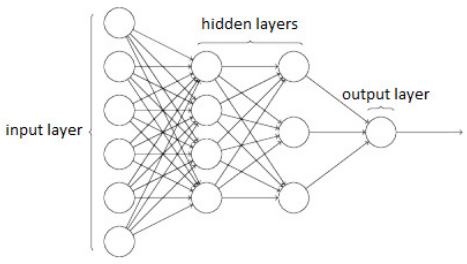
W pracach [3], [4], [5] oraz [6] w rozpoznawaniu mowy wykorzystany został algorytm maszyny wektorów nośnych. Autorzy artykułu [3] wykazali, że mechanizm uczenia maszynowego jakim jest SVM może być z powodzeniem zastosowany w klasyfikowaniu sekwencji mowy o zmiennej długości. Praca [6] przedstawia z kolei pomyślne wykorzystanie tego algorytmu w systemie ciągłego rozpoznawania głosu.

Algorytmy wykorzystane w niniejszej pracy, a mianowicie proste sieci neuronowe oraz sieci konwolucyjne również znajdują zastosowanie w rozpoznawaniu dźwięków mowy. Autorzy publikacji [7], [8] oraz [9] przedstawiają szereg zalet wykorzystania ich w rozwiązywaniu problemów klasyfikacji mowy i opisują zasadę ich działania. Praca [10] przedstawia skuteczne wykorzystanie w pełni połączonej seici jednokierunkowej w celu klasyfikacji wypowiadanych cyfr.

W pracy [11] autorzy zastosowali konwolucyjną sieć neuronową dla rozwiązania problemu rozpoznawania mowy. Klasyfikacji poddano niewielki zbiór słów kluczowych. Przedstawione zostały zalety wykorzystania modelu sieci konwolucyjnej w realizacji tego zadania. Ze względu na podobieństwo problemu zaprezentowanego w publikacji [11] z zadaniem postawionym w niniejszej pracy, zdecydowano na wykorzystanie tego ogólnodostępnego modelu dla klasyfikowania poleceń głosowych w utworzonej w celu realizacji badań aplikacji odtwarzacza muzyki. Na wykorzystanie sieci konwolucyjnej zdecydowali się również autorzy prac [12] i [13]. Publikacja [12] przedstawia wykorzystanie konwolucyjnej sieci neuronowej do klasyfikacji odgłosów występujących w środowisku, takich jak na przykład szczekanie psa czy klakson samochodowy.

2.2. Jednokierunkowa sieć neuronowa

Powstanie sztucznych sieci neuronowych (ang. *artificial neural network*) zainspirowane zostało możliwościami biologicznych neuronów oraz sieci neuronowych występujących w mózgu. Przetwarzają one informacje w sposób imitujący w pewnym stopniu działanie swoich naturalnych odpowiedników [14].



Rys. 1. Schemat sieci neuronowej typu perceptron wielowarstwowy [15]

Jednym z rodzajów sztucznych sieci neuronowych jest sieć jednokierunkowa. W tym podrozdziale przedstawiona zostanie sieć typu perceptron wielowarstwowy (ang. *multilayer perceptron, MLP*). Schemat takiej sieci przedstawiony został na rysunku 1. Składa się ona zazwyczaj z warstwy wejściowej, warstw ukrytych oraz warstwy wyjściowej.

- Warstwa wejściowa (ang. *input layer)* – zawiera dane dostarczone do sieci, a liczba znajdujących się w niej neuronów odpowiada ilości tych danych.
- Warstwy ukryte (ang. *hidden layers*) – zawierają neurony przetwarzające dane. W sieci neuronowej może występować jedna lub więcej warstw ukrytych.
- Warstwa wyjściowa (ang. *output layer*) – zawiera przewidywane odpowiedzi. W przypadku klasyfikacji liczba neuronów w tej warstwie jest równa liczbie klas.

Neurony, z których zbudowana jest sieć mogą posiadać wiele wejść i jedno wyjście. Każde połączenie występujące pomiędzy neuronami posiada przyporządkowaną wagę, przez którą pomnożona zostaje informacja wyjściowa jednego neuronu przekazywana do wejścia neuronu kolejnej warstwy. W docelowym neuronie, sumowane są wszystkie wejściowe dane pomnożone przez odpowiadające im wagi, a następnie do tej sumy dodawana jest pewna wartość ustalająca próg, przy którym neuron powinien się uaktywnić. Wartość ta to tak zwany bias. Otrzymany wynik jest podstawiany do funkcji aktywacji, gdzie zostaje przekształcony we właściwą wartość wyjściową. Równanie 1 przedstawia opisany wyżej schemat działania:

$$y = f\left(\sum_{i=1}^{N} w_i x_i + b\right) \tag{1}$$

gdzie:

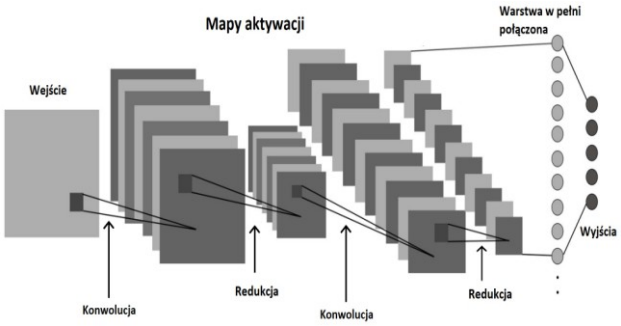
f – funkcja aktywacji,
x – informacja wejściowa,
w – waga informacji,
b – bias.

2.3. Konwolucyjna sieć neuronowa

Zamiast ogólnego mnożenia wektorów wykorzystywanego w zwykłych sieciach neuronowych, sieci konwolucyjne wykorzystują operację splotu (konwolucji) z wykorzystaniem zestawów pewnych filtrów. Operacja konwolucji najczęściej występuje w postaci korelacji krzyżowej [16]. Dzięki jej wykorzystaniu możliwe jest powstanie tak zwanych map aktywacji.

Splotowe sieci neuronowe wykorzystując zdefiniowane filtry, wykrywają pewne wzorce występujące na niewielkiej części danych wejściowych, a następnie są w stanie zauważyć, że te wykryte cechy powtarzają się na przestrzeni całej informacji dostarczonej na wejście. W rezultacie splotu danych wejściowych oraz tychże filtrów powstają wcześniej wspomniane mapy aktywacji, które są w kolejnym kroku zmniejszane (redukowane), a następnie po raz kolejny

poddawane operacji konwolucji. Architektura sieci konwolucyjnych została przedstawiona na rysunku 2.



Rys. 2. Architektura konwolucyjnej sieci neuronowej [17]

Typowa architektura sieci konwolucyjnych pozwala na wyróżnienie trzech głównych warstw:

- Warstwa konwolucyjna (ang. *convolutional layer*) – odpowiada za ekstrakcję cech. W tej warstwie wykonywana jest operacja splotu, w wyniku jej działania powstają mapy aktywacji.
- Warstwa łącząca (ang. *pooling layer*) – redukuje ilość informacji (wymiar map aktywacji). Najczęściej odbywa się to przez obliczenie średniej, lub wybór największej wartości spośród analizowanej niewielkiej części mapy aktywacji.
- Warstwa w pełni połączona (ang. *fully connected layer*) – jest to tradycyjna warstwa występująca również w innych, niekonwolucyjnych typach sieci neuronowych. Dane w niej przetwarzane są w postaci wektorów, dlatego też dostarczane do niej macierze map cech powinny zostać „spłaszczone" do postaci pojedynczego wektora, który zostaje podany na wejście tej warstwy.

3. Aplikacja testowa

Na potrzeby realizacji badania wpływu zastosowania uczenia maszynowego w budowie interfejsu sterowanego głosem utworzona została aplikacja odtwarzacza muzyki. Docelową platformą są urządzenia mobilne z systemem Android. Do prawidłowego działania aplikacji niezbędne są dwa zezwolenia: na nagrywanie dźwięku przez mikrofon urządzenia – w celu wprowadzania poleceń głosowych oraz na odczyt danych z pamięci urządzenia – w tym przypadku konieczny jest dostęp do plików utworów muzycznych, które będą wczytywane, a następnie odtwarzane.

Funkcje aplikacji nie różnią się od możliwości standardowych odtwarzaczy muzyki, jednak niektóre z nich możliwe są do wykonania za pomocą odpowiedniej komendy głosowej. W związku z tym zdefiniowane zostały wymagania funkcjonalne stworzonej aplikacji:

- stworzenie listy odtwarzania,
- usunięcie listy odtwarzania,
- wybór listy odtwarzania za pomocą wypowiedzenia jej nazwy,
- dodawanie utworu do listy odtwarzania,
- usunięcie utworu z listy odtwarzania,
- wyświetlenie listy wszystkich utworów,
- zatrzymanie odtwarzania,
- wznowienie odtwarzania,
- przyspieszenie odtwarzanego utworu o 10 sekund,
- cofnięcie odtwarzanego utworu o 10 sekund,
- przejście do następnego utworu,
- przejście do poprzedniego utworu,
- wyświetlenie listy rozpoznawanych poleceń głosowych,
- wyłączenie aplikacji za pomocą odpowiedniej komendy.

Funkcje aplikacji, które możliwe są do wykonania w sposób nie tylko standardowy (dotknięcie odpowiedniego przycisku na ekranie), ale również za pomocą głosu przedstawione zostały w tabeli 1 wraz z wywołującymi je poleceniami. To właśnie te komendy będą wykorzystane w analizie zastosowania uczenia maszynowego.

Tabela 1. Funkcje aplikacji wywoływane poleceniami głosowymi

Funkcja aplikacji	Polecenie wywołujące funkcję
Zatrzymanie odtwarzania	stop
Wznowienie odtwarzania	go
Przyspieszenie odtwarzanego utworu o 10 sekund	right
Cofnięcie odtwarzanego utworu o 10 sekund	left
Przejście do następnego utworu	forward
Przejście do poprzedniego utworu	backward
Wyłączenie aplikacji	off

4. Plan badań

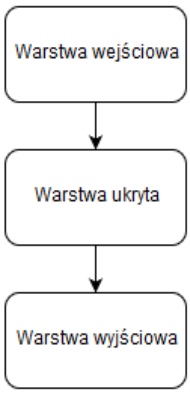
4.1. Wykorzystane modele

Zastosowane w utworzonej aplikacji testowej modele wybrane zostały z ogólnodostępnej biblioteki języka Python – *Tensorflow*. Ta otwartoźródłowa platforma umożliwia nie tylko tworzenie własnych modeli, ale również oferuje szeroki zakres gotowych rozwiązań w wykorzystaniu uczenia maszynowego dla różnych problemów, takich jak na przykład klasyfikowanie obrazów, wykrywanie obiektów czy jak w przypadku tej pracy rozpoznawania głosu [18].

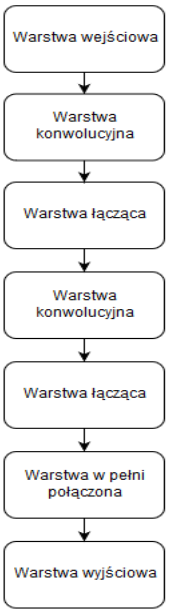
W celu przeprowadzenia badań wykorzystane zostały dwa algorytmy uczenia maszynowego oparte na sztucznych sieciach neuronowych. Pierwszym z nich jest prosta, jednokierunkowa oraz w pełni połączona sieć neuronowa, której poglądowy schemat przedstawiony został na rysunku 3.

Wykorzystana sztuczna sieć neuronowa posiada jedną warstwę ukrytą, w której występuje przetwarzanie danych wejściowych. Do neuronów tej warstwy dodawana jest zmienna wyznaczająca próg aktywacji (bias). Funkcja aktywacji, jaka została zastosowana w tym modelu to funkcja ReLU.

Drugi algorytm, który został wykorzystany w ramach badań to bardziej zaawansowany model – konwolucyjna sieć neuronowa. Poglądowy schemat tejże sieci przedstawia rysunek 4.



Rys. 3. Uproszczony schemat wykorzystanego modelu sieci jednokierunkowej



Rys. 4. Uproszczony schemat wykorzystanego modelu sieci konwolucyjnej

Model sieci konwolucyjnej, który został zastosowany w pracy posiada dwie warstwy konwolucyjne, dwie warstwy łączące, oraz warstwę, która występuje w prostych sieciach neuronowych – w pełni połączoną. W tym przypadku również zastosowana została zmienna biasu, która dodawana jest w każdej warstwie konwolucyjnej sieci oraz w warstwie w pełni połączonej. Funkcją aktywacji neuronów, tak jak w poprzednim modelu jest funkcja ReLU. W przypadku sieci konwolucyjnej należy również wspomnieć o metodzie redukowania informacji (wymiaru map aktywacji utworzonych przez operację splotu) wykonywanej przez warstwy łączące. W przypadku wykorzystanego modelu jest to wybór maksymalnej wartości spośród analizowanego obszaru danych.

4.2. Proces uczenia maszynowego

W celu przeprowadzenia procesu uczenia maszynowego wybranych modeli niezbędne było pozyskanie odpowiednich danych treningowych. W związku z tym wykorzystany został ogólnodostępny zbiór jednosekundowych [19] plików dźwiękowych, z których każdy zawiera jedno słowo (z trzydziestu pięciu występujących w zbiorze) wypowiedziane przez ludzi w różnym wieku oraz różnej płci. Dane te zebrane zostały przez Google aby pomagać w trenowaniu i doskonaleniu systemów wykrywających słowa kluczowe. Jest to ponad dwu-gigabajtowe archiwum danych zawierających ponad 105 tysięcy plików [19].

W niniejszej pracy w procesie uczenia maszynowego wybranych modeli wykorzystane zostały otwartoźródłowe skrypty napisane w języku Python udostępnione przez zespół Tensorflow na platformie GitHub [20]. Trening sieci neuronowych wykonany został przy użyciu pliku *train.py*, którego uruchomienie rozpoczyna cały proces.

Zastosowane sieci przyjmują dane wejściowe w postaci dwuwymiarowej, dlatego też sygnał dźwiękowy przekształcany zostaje do postaci obrazu, a w tym konkretnie przypadku spektrogramu. Odbywa się to poprzez podział całego sygnału na krótkie, trwające kilka milisekund próbki, dla których obliczone zostają amplitudy składowych harmonicznych. Oś pionowa spektrogramu odpowiada częstotliwości, natomiast oś pozioma określa czas. To właśnie spektrogram jest poddawany analizie oraz przetwarzany zostaje w kolejnych warstwach sieci. Przygotowanie danych wejściowych, a mianowicie przekształcenie sygnałów dźwiękowych z opisanego wcześniej zbioru utworzonego przez zespół Google [19] do postaci dwuwymiarowej zrealizowane zostało przez skrypt train.py.

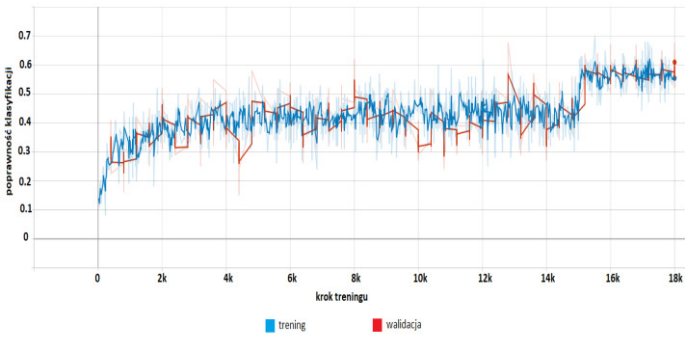
Plik train.py wymagał zastosowania kilku modyfikacji. Niezbędne było określenie modelu, który miał zostać poddany uczeniu maszynowemu. Zmienione zostały również etykiety, które reprezentują rozpoznawane przez model słowa, na te, które zostały wykorzystane w aplikacji testowej. To do nich będą klasyfikowane wprowadzane sygnały dźwiękowe. Kolejnym etapem było określenie liczby kroków oraz szybkości uczenia (ang. *learning rate*), jednak w tym przypadku wybrane zostały wartości domyślne – wykonanych zostało 18 000 kroków dla każdego modelu, z szybkością uczenia wynoszącą 0.001 dla pierwszych 15 000 kroków oraz 0.0001 dla kolejnych 3 000.

Na potrzeby pracy, dla każdego z zastosowanych algorytmów uczenia maszynowego, zapisane zostały po trzy modele, które będą badane – po 6 tysiącach kroków, po 12 tysiącach kroków oraz po zakończeniu całego treningu (po 18 tysiącach kroków). Dodatkowo analizie poddane zostaną same przebiegi uczenia maszynowego tych modeli.

5. Rezultaty badań

5.1. Sieć neuronowa jednokierunkowa

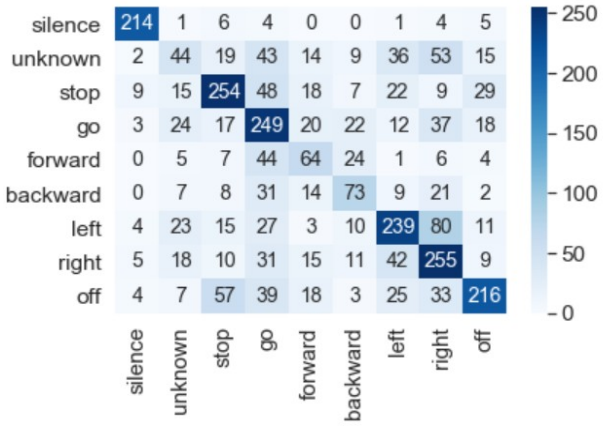
Pierwszym analizowanym modelem jest prosta sieć neuronowa. Przebieg jej uczenia przedstawiony został na rysunku 5. Czas trwania treningu opisywanej sieci nie był długi ze względu na niski stopień jej skomplikowania, trwał on blisko 6 godzin.



Rys.5. Przebieg treningu sieci jednokierunowej

W początkowym etapie treningu (pierwsze 2 tysiące kroków) poprawność klasyfikacji słów wzrastała wraz z kolejnymi krokami. W dalszej części (do 15 tysięcy kroków) zauważyć można, iż sieć przyporządkowuje słowa ze zgodnością wahającą się w granicach od 35% do 50%, bez widocznego progresu. Skok procentowej poprawności klasyfikowania słów wystąpił dopiero w momencie zmniejszenia szybkości uczenia sieci (zmiana wartości długości kroku uczenia z 0.001 do 0.0001) na ostatnie 3 tysiące kroków, jednak wzrost ten nastąpił gwałtownie po czym podobnie jak we wcześniejszej fazie treningu sieć przestała osiągać postępy w poprawnym klasyfikowaniu. Ostatecznie uczony model zakończył trening z prawidłowością rozpoznawania słów wynoszącą około 60%.

W celu dokładniejszej analizy uczenia maszynowego wykorzystanych modeli, po każdych 6 tysiącach kroków wygenerowana została macierz pomyłek (ang. confusion matrix), która jest podstawową metodą oceny poprawności klasyfikacji. Każdy wiersz macierzy reprezentuje instancję przewidywanej klasy, natomiast każda kolumna odpowiada instancji klasy rzeczywistej. Warto również zaznaczyć, iż do testów każdego modelu wykorzystana została walidacja krzyżowa (ang. *cross-validation*), której współczynnik N oznaczający liczebność zbioru testowego w przypadku modelu sieci jednokierunkowej był równy 2531 słów. W celu generowania macierzy wykorzystana została biblioteka Tensorflow, natomiast dla lepszej czytelności zostały one zwizualizowane przy użyciu biblioteki *seaborn* języka Python [21]. Ze względu na brak znaczących postępów w trakcie treningu, dla modelu prostej sieci jednokierunkowej przedstawiona została jedynie macierz wygenerowana po zakończeniu treningu (rysunek 6).



Rys. 6. Macierz błędu dla sieci jednokierunkowej po wykonaniu 18 tysięcy kroków

Macierz błędu wygenerowana po zakończeniu procesu uczenia maszynowego prostej sieci jednokierunkowej wykazała najlepszą poprawność klasyfikowania spośród wszystkich utworzonych macierzy dla tej sieci. Walidacja wykazała niespełna 60% zgodności. Zauważyć można, że na głównej przekątnej znajdują się największe liczby występujące w danych kolumnach. Oznacza to, ze polecenia najczęściej były rozpoznawane prawidłowo, a nie przypisywane do innych klas. Należy jednak zwrócić uwagę, że w macierzy występują wysokie wartości poza główną przekątną, co świadczy o tym, że model w dalszym ciągu często popełnia błędy klasyfikując komendy głosowe. Najlepiej rozpoznawana była klasa reprezentująca ciszę, po czym można wywnioskować, że brak sygnału dźwiękowego jest najłatwiejszy dla rozpoznania przez model. W aplikacji testowej użyty został tylko model, który wykonał wszystkie zaplanowane kroki. Wykorzystanie modelu w aplikacji zostało przetestowane w następujących aspektach:

- prawidłowe rozpoznawanie poleceń z wyciszoną muzyką,
- prawidłowe rozpoznawanie poleceń z muzyką w tle odtwarzaną na niewielkim poziomie głośności (25%),
- prawidłowe rozpoznawanie poleceń z muzyką w tle odtwarzaną na średnim poziomie głośności (50%),
- prawidłowe rozpoznawanie poleceń z muzyką w tle odtwarzaną na wysokim poziomie głośności (100%),
- nie wykonywanie operacji przez aplikację przypisanych do poleceń głosowych podczas odtwarzania muzyki samoczynnie, bez wypowiadania poleceń przez osobę testującą,
- rozpoznawanie poleceń z podłączonym zestawem słuchawkowym.

Tabela 2 prezentuje wyniki przeprowadzonego testu. Wskazują one, które przypadki wystąpiły podczas korzystania z wytrenowanego modelu jednowarstwowej sztucznej sieci jednokierunkowej.

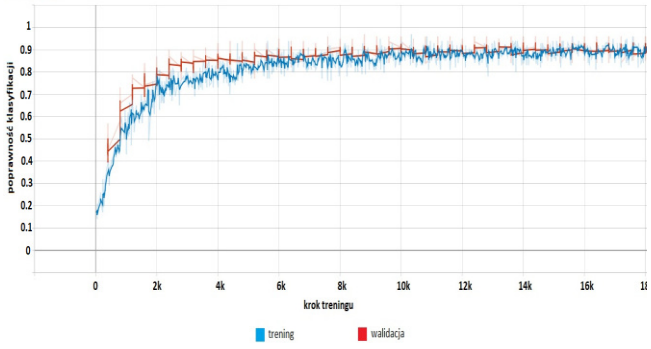
Tabela 2. Wyniki testów dla modelu jednowarstwowej sieci jednokierunkowej

Test	Rezultat
Rozpoznawanie poleceń z wyciszoną muzyką	-
Rozpoznawanie poleceń z muzyką na poziomie głośności 25%	-
Rozpoznawanie poleceń z muzyką na poziomie głośności 50%	-
Rozpoznawanie poleceń z muzyką na poziomie głośności 100%	-
Brak samoczynnego wykonywania operacji	+
Rozpoznawanie poleceń z podłączonym zestawem słuchawkowym	-

Zastosowany model sieci jednokierunkowej nie radzi sobie z poprawną klasyfikacją poleceń. Poziom głośności odtwarzania dźwięku nie ma znaczenia, gdyż żadna operacja nie została wykonana.

5.2. Konwolucyjna sieć nueronowa

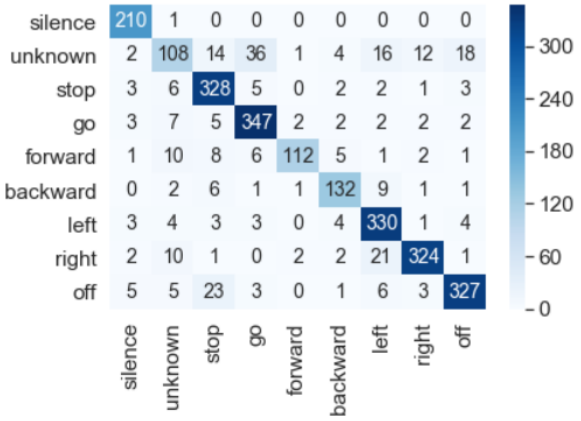
Kolejnym algorytmem, który został wykorzystany do badań jest konwolucyjna sieć neuronowa. Jest ona bardziej skomplikowana od wcześniej opisywanej jednowarstwowej sieci jednokierunkowej, dlatego też trening tej sieci był bardziej czasochłonny, trwał około 30 godzin. Rysunek 7 przedstawia przebieg procesu uczenia maszynowego sieci konwolucyjnej.



Rys. 7. Przebieg treningu sieci konwolucyjnej

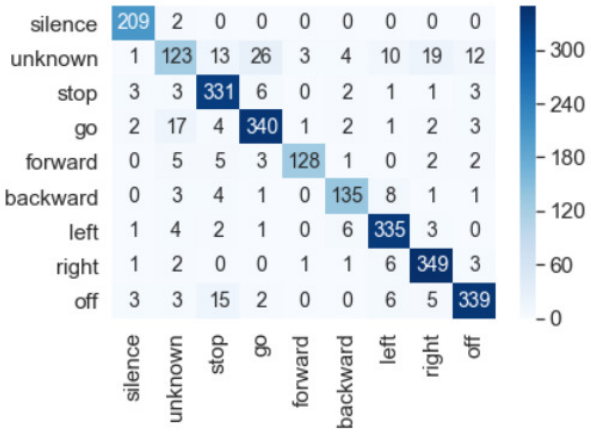
Trening opisywanej sieci przebiegał bardzo efektywnie w początkowej części procesu uczenia maszynowego. Do około 3 tysiąca kroków widoczny jest znaczny wzrost poprawności klasyfikacji wraz z postępem treningu. W kolejnych krokach sieć w dalszym ciągu poprawia wynik zgodności w rozpoznawaniu słów, jednak wzrost ten postępuje już znacznie wolniej. Po przekroczeniu 8 tysięcy kroków treningowych, sieć utrzymuje poprawność klasyfikacji na poziomie w okolicach 85%, a nawet 90%, aż do końca treningu. Zmiana szybkości uczenia modelu w końcowej fazie (po 15 tysiącach kroków treningu) powoduje zmniejszenie wahań poprawności sieci w klasyfikacji poleceń występujących pomiędzy krokami procesu uczenia.

W przypadku modelu konwolucyjnej sieci neuronowej, również zostały wykonane trzy macierze błędu, aby lepiej zobrazować proces uczenia maszynowego. Ponownie w celach testu wykorzystana została walidacja krzyżowa, a współczynnik oznaczający liczebność próbki wynosił jak poprzednio 2531 słów. Pierwsza z macierzy pomyłek przedstawiona została na rysunku 8.



Rys. 8. Macierz błędu dla konwolucyjnej sieci neuronowej po wykonaniu 6 tysięcy kroków treningu

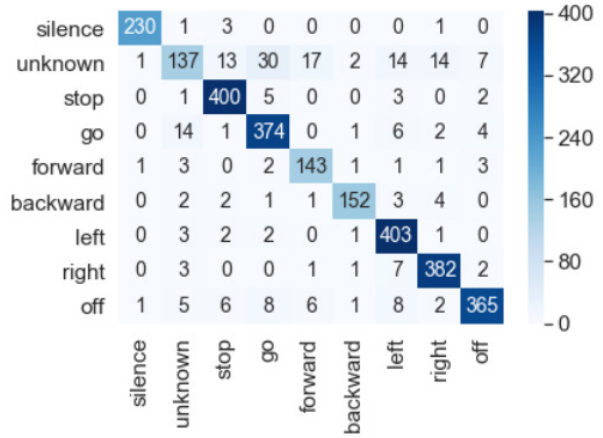
Zdecydowana większość poleceń została poprawnie sklasyfikowana. Na tle całej macierzy najwięcej błędów dostrzec można w klasie unknown oznaczającej nieznane polecenie, do której to najczęściej błędnie przypisywane były komendy, które model powinien znać. Najmniej pomyłek wystąpiło w przypadku, gdy rzeczywistą instancją klasy było słowo forward. Walidacja wykazała, że całkowita poprawność klasyfikacji modelu na tym etapie treningu wynosi już aż 87,6%. Macierz błędu wykonana po 12 tysiącach kroków (rysunek 9) treningu jest bardzo podobna do poprzednio analizowanej macierzy (po 6 tysiącach kroków).



Rys. 9. Macierz błędu dla konwolucyjnej sieci neuronowej po wykonaniu 12 tysięcy kroków treningu

Wszystkie z rozpoznawanych słów są w znaczącym stopniu dobrze klasyfikowane. W dalszym ciągu najwięcej błędów pojawia się w wierszu unknown, gdzie jako nieznane polecenia rozpoznawane są te, które model powinien przydzielać do odpowiadających im klas. Ich liczba jednak nie jest znacząca. Wskaźnik walidacji w tym przypadku wyniósł 90,4%.

W przypadku macierzy wygenerowanej po wykonaniu wszystkich 18 tysięcy kroków (rysunek 10) wskaźnik walidacji jest największy i wynosi 91,9%. Liczby występujące poza główną przekątną są małe, co oznacza, że klasyfikacja odbywa się na zadowalającym poziomie, występuje niewiele błędnie sklasyfikowanych poleceń.



Rys. 10. Macierz błędu dla konwolucyjnej sieci neuronowej po wykonaniu 18 tysięcy kroków treningu

W aplikacji testowej wykorzystane zostały wszystkie trzy modele (zapisane na trzech różnych etapach treningu) konwolucyjnej sieci neuronowej. Testy odbyły się według tych samych aspektów, co w przypadku prostej sieci jednokierunkowej. Wyniki dla modeli tej sieci przedstawia tabela 3.

Sieć konwolucyjna po 6000 kroków treningu bezproblemowo radziła sobie z prawidłowym rozpoznawaniem poleceń w przypadku, gdy odtwarzanie nie było zbyt głośne. W przypadku gdy poziom głośności mediów ustawiony został na 50% maksymalnej wartości, rozpoznawanie poleceń w dalszym ciągu było wykonywane, aczkolwiek konieczne było wypowiadanie komend głośniej niż w poprzednich przypadkach. W momencie, gdy głośność odtwarzania ustawiona została na maksymalną wartość, ponad 50% poleceń było rozpoznawanych nieprawidłowo. Dodatkowo przy wysokim poziomie dźwięku, wynoszącym już około 70% aplikacja wykonywała losowe operacje, pomimo braku wypowiadania rozpoznawanych słów.

Model, który wykonał 12000 kroków, podobnie jak poprzedni nie miał żadnych problemów z rozpoznawaniem poleceń głosowych w przypadku gdy poziom głośności muzyki odtwarzanej w tle nie przekraczał 25%. Głośniejsze wypowiadanie poleceń było niezbędne w celu ich rozpoznania przy maksymalnym poziomie głośności urządzenia. Dodatkowo przy wysokim poziomie głośności aplikacja wykonywała niezamierzone operacje.

Ostatni z testowanych modeli, który wykonał cały trening sprawdził się w aplikacji podobnie jak model po 12 tysiącach kroków treningowych. Rożnica w poprawności klasyfikacji tych modeli była niewielka, stąd też ich działanie w aplikacji testowej jest niemal identyczne. Komendy z łatwością są rozpoznawane prawidłowo do momentu, gdy poziom głośności zbliża się do maksimum.

Tabela 3. Wyniki testów dla modelu konwolucyjnej sieci neuronowej

	Rezultat		
Test	**Sieć konwolucyjna po 6000 krokachów**	**Sieć konwolucyjna po 12000 kroków**	**Sieć konwolucyjna po 18000 kroków**
Rozpoznawanie poleceń z wyciszoną muzyką	+	+	+
Rozpoznawanie poleceń z muzyką na poziomie głośności 25%	+	+	+
Rozpoznawanie poleceń z muzyką na poziomie głośności 50%	+	+	+
Rozpoznawanie poleceń z muzyką na poziomie głośności 100%	-	-/+	-/+
Brak samoczynnego wykonywania operacji	-	-	-
Rozpoznawanie poleceń z podłączonym zestawem słuchawkowym	+	+	+

Wszystkie z testowanych modeli konwolucyjnej sieci neuronowej działają prawidłowo z wykorzystaniem zestawu słuchawkowego, gdyż w tym przypadku mikrofon nie odbiera zakłóceń w postaci odtwarzanej muzyki podczas wprowadzania poleceń głosowych.

6. Dyskusja wyników i wnioski

Trening jednowarstwowej sieci jednokierunkowej wykonany został znacznie szybciej niż sieci konwolucyjnej. Jak jednak wynika z badań, zastosowanie modelu prostej sieci daje znacznie gorsze wyniki w obsługiwaniu aplikacji testowej za pomocą głosu w porównaniu do sieci splotowej. Z rysunków 5 i 7 wynika, że bardziej skomplikowana sieć konwolucyjna już po około 2 tysiącach kroków treningu osiągnęła lepsze rezultaty w klasyfikacji słów niż w stanie była osiągnąć zastosowana zwykła sieć jednowarstwowa. Wykorzystany model konwolucyjnej sieci neuronowej już po wykonaniu 6 tysięcy kroków dał możliwość głosowego sterowania interfejsem. Im więcej kroków treningu zostało wykonanych, tym lepiej spisywał się model sieci splotowej.

Uzyskany wynik jednokierunkowej sieci w pełni połączonej nie jest zadowalający. W przypadku publikacji [10] autorzy wykorzystując podobny algorytm uzyskali znacznie lepszy rezultat (poprawność klasyfikacji około 94%), jednak parametry użytej przez nich sieci różniły się od tych

zastosowanych w niniejszej pracy, co jak można wywnioskować ma znaczący wpływ na efektywność treningu. Sieć konwolucyjna dobrze poradziła sobie z klasyfikacją poleceń, wykorzystany model uzyskał poprawność klasyfikacji na poziomie powyżej 90%. W publikacji [12] gdzie wykorzystany został ten sam algorytm zgodność rozpoznawania odgłosów występujących w ludzkim otoczeniu wyniosła od 60% do 85% w zależności od wykorzystanego zbioru treningowego. Można więc wywnioskować że dane wejściowe dostarczane do sieci mają wpływ na proces treningu.

Analizując uzyskane wyniki można stwierdzić, że postawiona we wprowadzeniu pracy teza jest prawdziwa. Konwolucyjna sieć neuronowa z odpowiednio dobranymi parametrami pozwaliła uzyskać lepsze wyniki klasyfikacji niż jednowarstwowa sieć jednokierunkowa.

Literatura

[1] J. Ye, R. J. Povinelli, M. T. Johnson: „Phenome classification using naive Bayes classifier in reconstructed phase space", IEEE Digital Signal Processing Workshop, 2002

[2] A. Sanchis, A. Juan, E. Vidal: „A Word-Based Naive Bayes Classifier for Confidence Estimation in Speech Recognition", IEEE Transactions on audio, speech and language processing, vol. 20, NO. 2, 2012

[3] N. Smith, M. Gales: „Speech Recognition using SVMs", Cambridge University Engineering Dept, 2002

[4] C. Ittichaichareon, S. Suksri, T. Yingthawornsuk: „Speech Recognition using MFCC", International Conference on Computer Graphics, Simulation and Modeling , 2012

[5] W. M. Campbell, J. P. Campbell, D. A. Reynolds, E. Singer, P. A. Torres-Carrasquillo: „Support vector machines for speaker and language recognition", Computer Speech and Language 20 210–229, 2006

[6] A. Ganapathiraju, J. E. Hamaker, J. Picone: „Applications of Support Vector Machines to speech recognition", IEEE Transactions on signal processing, vol 52, NO. 8, 2004

[7] K. Al Smadi, I. Trrad, T. Al Smadi: „ Artificial Intelligence for Speech Recognition Based on Neural Networks", Journal of Signal and Information Processing, 2015, 6, 66-72, 2015

[8] W. Gevaert, G. Tsenov, V. Mladenov: „Neural networks used for speech recognition",Journal of automatic control, University of Belgrade, vol. 20:1-7 , 2010

[9] M. Tunckanat, R. Kurban S. Sagiroglu: „Voice Recognition Based On Neural Networks", IJCI Proceedings of International Conference on Signal Processing, ISSN 1304-2386, Volume:1, Number:2, 2003

[10] A. Ahad, A. Fayyaz, T. Mehmood: „Speech Recognition using Multilayer Perceptron", Students Conference, ISCON apos:02. IEEE Volume 1, Issue, 16-17, 2002

[11] T. N. Sainath, C. Parada: „Convolutional Neural Networks for Small-footprint Keyword Spotting", Interspeech, 2015

[12] K. J. Piczak: „Environmental Sound Classification With Convolutional Neural Networks", IEEE International Workshop on Machine Learning For Signal Processing, 2015

[13] O. Abdel-Hamid, A. Mohamed, H. Jiang, L. Deng, G. Penn, D. Yu: „Convolutional Neural Networks for Speech Recognition", IEEE/ACM Transactions on audio, speech and language processing, vol. 22, NO. 10, 2014

[14] Tadeusiewicz R.: Sieci neuronowe. Akademicka Oficyna Wydawnicza RM, 1993

[15] Nielsen M.: Neural Networks and Deep Learning. Determination Press, 2015

[16] Goodfellow I., Bengio Y. i Courville A., Deep Learning, 2016

[17] Wprowadzenie do kowolucyjnych sieci neuronowych https://towardsdatascience.com/simple-introduction-to-convolutional-neural-networks-cdf8d3077bac, czerwiec 2019

[18] Strona projektu Tensorflow https://www.tensorflow.org/, czerwiec 2019

[19] Pete Warden: Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition, 2018

[20] Repozytorium biblioteki Tensorflow https://github.com/tensorflow/tensorflow, czerwiec 2019

[21] Dokumentacja biblioteki Seaborn https://seaborn.pydata.org/index.html, czerwiec 2019