02.02.2019

Politechnika Śląska
Wydział Automatyki, Elektroniki i Informatyki

Cyfrowe przetwarzanie sygnałów
Projekt – raport końcowy

Sprzęt

Wymagania

Platforma do uruchomienia oprogramowania stworzonego w projekcie musiała cechować się dobrą wydajnością obliczeniową. Obliczanie współczynników cepstralnych, nawet na komputerze osobistym wysokiej klasy, nie dawało wyników natychmiast i wymagało kilku sekundowego oczekiwania na zakończenie obliczeń.

Kolejnym wymaganiem, jakie postawiono przed dobieraną platformą, była możliwość stworzenia graficznego interfejsu użytkownika. Podjęto taką decyzję, ponieważ stwierdzono, że umożliwi on łatwiejsze korzystanie z urządzenia. Wybrany do realizacji sprzęt powinien albo posiadać wbudowany wyświetlacz, najlepiej dotykowy, na którym wspomniany graficzny interfejs będzie można zrealizować. Inną możliwością spełnienia powyższego wymagania jest opcja polegająca na podłączeniu zewnętrznych urządzeń wejścia-wyjścia, takich jak monitor, klawiatura czy mysz.

Stworzone urządzenie, żeby rozpoznawać dźwięk, musiało najpierw zarejestrować go, a następnie przekazać do programu, który zajmował się jego przetwarzaniem i analizą. Dodatkowo bardzo pomocną opcją przy rozwijaniu oprogramowania, ale i poprawiającą wrażenia użytkownika przy korzystaniu z urządzenia, mogła okazać się opcja odtwarzania dźwięku.

Wybrana platforma sprzętowa

Uwzględniając powyżej wymienione wymagania zdecydowano się wybrać minikomputer Raspberry Pi 3B+ (Rysunek 1). Cechuje go wydajny, cztero-rdzeniowy procesor Broadcom BCM2837B0 64-bitowy ARM-8 Cortex-A53 taktowany sygnałem zegarowym o częstotliwości 1,4 GHz. Posiada sprzętowe wsparcie obliczeń, wykonywanych na liczbach zmiennoprzecinkowych, co pomogło szybko wykonywać algorytm rozpoznawania wzorców.

Rysunek 1 – Raspberry Pi 3B+ – Wybrana platforma sprzętowa do realizacji projektu

Nie posiada wbudowanego ekranu dotykowego, jednak umożliwia podłączanie wielu urządzeń wejścia-wyjścia. Monitor może zostać podłączony do portu HDMI, który znajduje się na płytce. Kolejne urządzenia można podłączyć do portów USB, których płytka posiada aż cztery.

Wybrana platforma sprzętowa nie posiada wbudowanej karty dźwiękowej. Z tego powodu zdecydowano się dokupić zewnętrzną kartę. Musiała ona komunikować się z minikomputerem. Należało zadbać o wybór karty z interfejsem obsługiwanym przez płytkę. Dostępnymi interfejsami były: interfejsy komunikacyjne takie jak SPI czy UART, dostępne na wyprowadzeniach płytki oraz interfejs USB. Zdecydowano się na interfejs USB ze względu na prostotę tego rozwiązania.

Wybrany minikomputer Raspberry Pi 3B+ spełnia jeszcze jedno wymaganie, niewymienione uprzednio. Jest bardzo popularną platformą wśród hobbystów i w Internecie można natknąć się na wiele ciekawych projektów, zrealizowanych z jej pomocą. Autorzy projektu chcieli się z nią zapoznać by zaspokoić ciekawość, dotyczącą ogromnego sukcesu tej platformy.

Urządzenia zewnętrzne

Celem zapewnienia możliwości wprowadzania dźwięku do komputera i wyprowadzania go na zewnątrz zaopatrzono się w dodatkowe urządzenia wejścia-wyjścia (Rysunek 2). Dokupiono zewnętrzną kartę dźwiękową, komunikującą się z komputerem interfejsem USB. Karta ta udostępniała dwa interfejsy mini-jack do przesyłania sygnałów z mikrofonu i do słuchawek, w które to również się zaopatrzono. Wybrano osprzęt audio najniższej klasy z powodu oszczędności, ale i by sprawdzić, jak niższa jakość rejestrowanego sygnału dźwiękowego przełoży się na wyniki rozpoznawania wzorców.

Rysunek 2 – Wybrane zewnętrzne urządzenia do rejestracji i odtwarzania dźwięku

Praca z dźwiękiem

Konfiguracja karty dźwiękowej

Zainstalowano kartę dźwiękową w systemie i skonfigurowano tak, by umożliwić rejestrację i odtwarzanie dźwięku. Rozwiązano pojawiający się problem, związany z wyborem przez system operacyjny złego urządzenia obsługującego odtwarzanie dźwięku. Gdy do płytki podpięty był monitor z głośnikami, zdarzało się, że stawał się on urządzeniem domyślnie odtwarzającym dźwięk. Nie było to zachowaniem pożądanym. Zmieniono to zachowanie, konfigurując system operacyjny tak, by zawsze wybierał jako urządzenie domyślne to urządzenie, które zostanie podłączone do karty dźwiękowej.

Biblioteki

Do wykonywania operacji polegających na odtwarzaniu i rejestrowaniu dźwięków zastosowano biblioteki (Rysunek 3). ALSA jest biblioteką niskiego poziomu i wymaga dużej liczby operacji by umożliwić wykonanie stosunkowo prostych czynności. Z tego powodu wykorzystano bibliotekę ALSA do odtwarzania dźwięku, a nagrywanie zdecydowano się zrealizować w oparciu o inne rozwiązania. Do tego celu świetnie nadał się moduł QtAudio z biblioteki Qt. Qt i tak był wykorzystywany w projekcie do tworzenia graficznego interfejsu użytkownika, zatem nie wymagało to wprowadzania dodatkowych zależności do projektu w celu rejestracji dźwięku.

Rysunek 3 – Zastosowane biblioteki do obsługi dźwięku (od lewej: ALSA, QtAudio)

Rozwijanie oprogramowania

Praca na minikomputerze jest zwykle mniej wygodna niż na komputerze osobistym. Jest to bowiem komputer, którego zasoby sprzętowe są o wiele uboższe, od tych dostępnych w standardowych komputerach osobistych. Dzisiejsze ciężkie zintegrowane środowiska deweloperskie oraz kompilacja przygotowywanego programu, wymagają od programisty wiele cierpliwości. Możliwe jest jednak rozwijanie oprogramowania i jego kompilacja, na szybkim i wygodnym komputerze osobistym. Dopiero program, będący wynikiem kompilacji na komputerze osobistym, jest przesyłany do minikomputera. Ten sposób nazywany jest kompilacją skrośną (ang. cross-compiling).

Podczas realizacji projektu zdecydowano się zastosować tę technikę. Skonfigurowano do tego celu narzędzia zarówno na komputerze osobistym, na którym prowadzone były prace projektowe, jak i w minikomputerze, na którym docelowo projekt miał zostać uruchomiony. Przez pewien czas projekt był z powodzeniem rozwijany w ten sposób, jednak na skutek zaniku napięcia podczas prowadzonych prac, dane konfiguracji zostały utracone. Stwierdzono, że ponowna konfiguracja kompilacji skrośnej wymagać będzie zbyt wiele pracy, w stosunku do korzyści, które przyniesie. Z tego powodu zdecydowano się zmienić sposób pracy.

Kontynuowano tworzenie kodu źródłowego na komputerze osobistym, jednak zdecydowano się przenieść proces kompilacji na minikomputer. Wiązało się to z wydłużeniem czasu kompilacji programu, jednak oszczędzało czas niezbędny do konfiguracji kompilacji skrośnej. Jedynym problemem, jaki należało wówczas rozwiązać, było znalezienie sposobu, na przesyłanie plików źródłowych programu na minikomputer i uruchomienie procesu kompilacji programu. Do tego celu wykorzystano zewnętrzne narzędzia (Rysunek 4).

Rysunek 4 – Oprogramowanie do komunikacji z minikomputerem (od lewej: FileZilla, VNC Viewer, Putty)

FileZilla to oprogramowanie, pozwalające przesyłać pliki protokołem FTP. W projekcie wykorzystano je w celu przesłania źródeł programu z komputera osobistego, na którym zostały stworzone, na minikomputer, na którym były kompilowane. Do uruchomienia procesu kompilacji wykorzystywano programy VNC Viewer oraz Putty. Zapewniały one dostęp do graficznego interfejsu użytkownika zapewnianego przez minikomputer (VNC Viewer) lub do tekstowego (Putty).

Po skompilowaniu programu i uruchomieniu go przeprowadzano testy manualne pozwalające ocenić, czy napisany program działa zgodnie z oczekiwaniami, czy wymaga wprowadzenia poprawek. W przypadku, gdy tester stwierdzał niepoprawne działanie aplikacji, przekazywał informację zwrotną do programisty, który na podstawie zgłoszonych błędów, poprawiał aplikację. Poprawiony kod ponownie przesyłano do minikomputera, gdzie był kompilowany i uruchamiany.

Algorytmy cyfrowego przetwarzania sygnałów

Współczynniki cepstralne

Jedną z zaproponowanych przez nas metod rozpoznawania wzorców mowy jest obliczanie współczynników cepstralnych. Ów algorytm zrealizowaliśmy w oparciu o książkę Tomasza P. Zielińskiego “Cyfrowe Przetwarzanie Sygnałów” [1]. Znajdującą się tam implementację przepisaliśmy jako prototyp z użyciem pakietu MATLAB, by następnie odwzorować funkcjonalność algorytmu wykorzystując bardziej przenaszalny język C++.

W najprostszym ujęciu, obliczanie współczynników cepstralnych to obliczenie zestawu wartości charakteryzujących nagranie. Po uzyskaniu owych wartości przystąpić można z wykorzystaniem algorytmu DTW do porównywania uzyskanych z analizowanego nagrania współczynników ze współczynnikami wzorców znajdujących się w bazie wzorców.

Niniejszy podrozdział poświęcony jest przybliżeniu istoty działania algorytmu oraz nakreśleniu zasady jego działania.

Algorytm obliczania współczynników

Treść mowy ukryta jest w zmiennej charakterystyce częstotliwościowej filtra traktu głosowego, a w jego pobudzeniu występują jedynie cechy osobnicze i intonacyjne. Ruchy szczęki, języka i ust powodują przestrajanie częstotliwości rezonansowych, czyli zmianę położenia i kształtu czterech formant maksimów wyraźnie widocznych w obwiedni widma sygnału mowy. Rozpoznawanie mowy sprowadza się więc do śledzenia zmian charakterystyki częstotliwościowej filtra traktu głosowego, czyli kształtu obwiedni jej widma podczas wypowiadania poszczególnych słów i porównywanie ich z trajektoriami zmian charakterystyk częstotliwościowej filtrów (obwiedni widm) odpowiadającym słowom wzorcowym znajdującym się w bazie danych. Rozpoznawanie może być zależne lub niezależne od mówcy, słów izolowanych lub mowy ciągłej, z zamkniętego lub otwartego słownika [1].

Najczęściej w celu odnajdywania podobieństw między wypowiedzianymi słowami stosuje się algorytm obliczania współczynników cepstralnych. Owe współczynniki otrzymać można w sposób dwojaki. Jednym ze sposobów jest zlogarytmowania modułu widma sygnału i ponowne wyznaczenie prostej lub odwrotnej (nie ma znaczenia na jaką się zdecydujemy) dyskretnej transformaty Fouriera. Natomiast wykorzystywane przez nas podejście korzysta z obliczania p współczynników filtra predykcji a_k, które to z kolei wykorzystywane są do obliczania q współczynników cepstralnych c_k.

$$c_1 = a_1$$

$$c_k = a_k + \sum_{m=1}^{k-1} \frac{m}{k} c_m a_{k-m}, \quad 2 \leq k \leq p$$

$$c_k = \sum_{m=1}^{k-1} \frac{m}{k} c_m a_{k-m}, \quad p+1 \leq k \leq q$$

Po obliczeniu współczynników cepstralnych dodatkowo dokonujemy ich ważenia z wykorzystaniem wag w_k. Otrzymujemy dzięki temu ostateczne wartości współczynników cw_k.

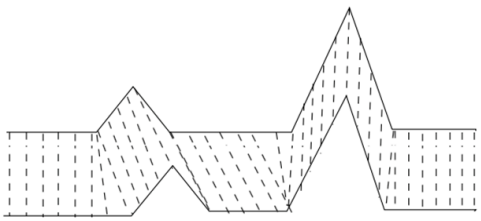
$$cw_k = c_k w_k, \;\; 1 \leq k \leq q$$

$$w_k = 1 + \frac{q}{2}\sin\left(\frac{\pi k}{q}\right), \;\; 1 \leq k \leq q$$

Przyjmuje się, że pierwszych 10-15 współczynników cepstralnych wystarczająco dobrze charakteryzuje sygnał. W naszej implementacji dla każdego analizowanego wzorca i nagrania obliczaliśmy *q*=12 współczynników, dla rzędu filtra predykcji równego *p*=10. Ramka danych obejmująca analizowany fragment sygnału w naszej implementacji wynosi 30ms z przesunięciem równym 10ms. Konfiguracji współczynników nie testowaliśmy dodatkowo, prototyp utworzony w środowisku Matlab dawał zadowalające wyniki. Aczkolwiek niewykluczone, że wybrane przez nas wartości współczynników dalekie są od optymalnych.

Algorytm porównywania DTW

Macierz współczynników cepstralnych obliczona dla poszczególnych nagrań różni się liczbą wierszy, gdyż zależnie od długości nagrania, więcej jest 30 ms “okienek”, w obrębie których analizowane jest nagranie. Niezmienna natomiast jest liczba kolumn w macierzy współczynników, wynosi ona 12 niezależnie od długości ani innych cech nagrania. Wobec powyższych stwierdzeń nie jest możliwe porównywanie współczynników dla dwóch nagrań w sposób “jeden do jeden”. Wymagany jest algorytm, który będzie niewrażliwy (albo ową niewrażliwość będzie maksymalizował) na różnice w rozmiarze macierzy współczynników cepstralnych dla porównywanych nagrań. Postawione przez nas wymagania spełnia nieliniowa transformacja czasowa DTW (ang. Dynamic time Warping).



Rysunek 5 – Ilustracja algorytmu DTW

Jak ukazuje powyższa ilustracja, bardzo podobne sekwencje, jednak nieco przesunięte w czasie, w zasadzie zawierają te same informacje, ale porównywanie ich “jeden do jednego” uniemożliwiłoby wyciągnięcie takiego wniosku. Stąd też konieczność stosowania DTW.

Załóżmy, że dysponujemy macierzą współczynników cepstralnych C_s obliczonych dla analizowanego nagrania oraz macierzą wzorców cepstralnych C_w. Dla każdego wzorca wykonywane są następujące operacje:

1. Tworzona jest macierz d o wymiarach (liczba wzorców x liczba wierszy macierzy do porównania), której elementy mają wartości równe odległości euklidesowej pomiędzy cepstrum słowa o numerze n_s, a ceprstrum wzorca o numerze n_w:

$$d(n_s, n_w) = \sqrt{\sum_{k=1}^{q} (C_s(n_s, k) - C_w(n_w, k))^2}, \;\; n_s = 1 \dots N_s, \;\; n_w = 1 \dots N_w$$

2. Następnie należy obliczyć najmniejszy skumulowany koszt przejścia pomiędzy lewym dolnym, a prawym górnym rogiem tej macierzy, czyli tzw. odległość globalną (skumulowaną). Poprzez akumulację w tym przypadku rozumie się sumowanie wartości z macierzy d przez które "przechodzi się". Macierz d zapewniająca najmniejszy koszt skumulowany zapewnia największe podobieństwo analizowanego nagrania do wzorca w kontekście współczynników cepstralnych.

Warto zaznaczyć, że liczba wszystkich dostępnych "przejść" po macierzy d jest ogólnie bardzo duża, zatem wprowadza się ograniczenie zezwalające jedynie na przejście o jednostkę w górę, prawo lub na ukos (czyli na raz w prawo i do góry). Wobec czego nieznacznie tracimy na dokładności algorytmu, ale znacznie zyskujemy na zmniejszeniu jego złożoności obliczeniowej.

Aby jeszcze nieco przyspieszyć obliczenia wprowadza się pewną modyfikację algorytmu. Mianowicie, ostatnim etapem na drodze do wyznaczenia wzorca najbardziej podobnego do analizowanego nagrania jest utworzenie macierzy g poprzez szereg transformacji macierzy d oraz unormowanie wartości w niej występujących i umieszczenie ich w macierzy a.

$$a(s,w) = \frac{g(N_s, N_w)}{\sqrt{N_s^2 + N_w^2}}$$

Dopiero odnalezienie najmniejszego skumulowanego kosztu przejścia po elementach macierzy a z wykorzystaniem opisanych wcześniej reguł "przechodzenia" zapewni najlepsze dopasowanie analizowanego nagrania do wzorca w bazie danych.

Uwagi implementacyjne

Algorytmy obliczania współczynników cepstralnych oraz DTW zostały w całości zaimplementowane w języku C++. Mnożenie macierzy i wektorów odbywa się przy pomocy biblioteki *boost*. Pamiętaliśmy również o tym, by aplikacja nie wykorzystywała nadmiernie zasobów komputera/urządzenia wbudowanego, zatem umiejętnie przekazujemy pomiędzy funkcjami gigantyczne wręcz macierze współczynników cepstralnych, których rozmiar w pamięci urządzenia przekraczał niejednokrotnie kilkadziesiąt kilobajtów, co dla niejednego urządzenia wbudowanego byłoby wartością "nie do przeskoczenia". Ponadto, w celu przyspieszenia obliczeń wzorce w pamięci programu umieszczane są natychmiastowo po jego uruchomieniu.

Co również bardzo istotne, każdy z wzorców oraz każde z nagrań poddawanych analizie zostaje najpierw przycięte na początku i na końcu w celu usunięcia "ciszy", której obecność jest wysoce niewskazana w kontekście analizy mowy, nie tylko pod kątem obliczania współczynników cepstralnych.

Sieć neuronowa

Narzędzia i technologie

Programowanie sieci neuronowych zostało wykonane w języku Python. Do programowania zostało użyte środowisko IDE PyCharm. Niezbędne były również odpowiednie biblioteki do uczenia maszynowego oraz głębokiego uczenia (Keras oraz TensorFlow).

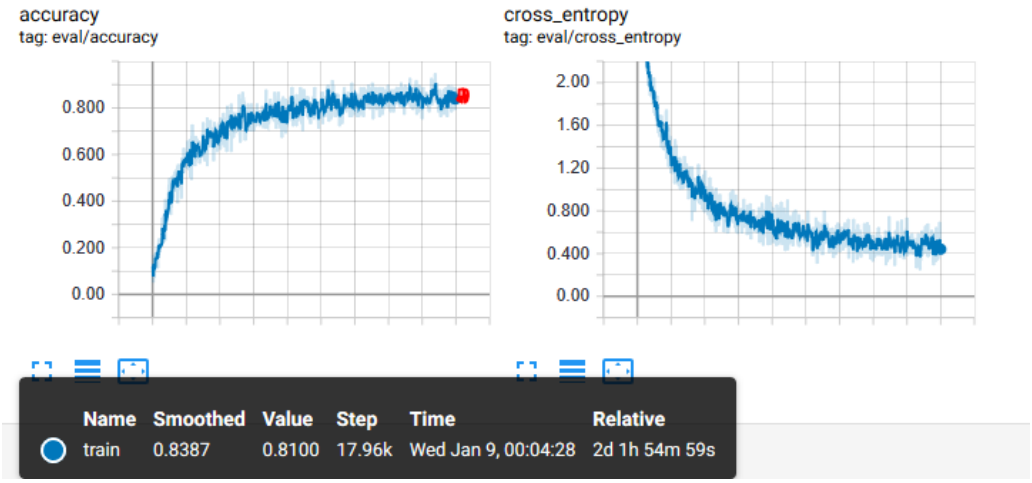
Przygotowanie

Przygotowanie danych do nauki sieci neuronowej polegało na ściągnięciu zestawu uczącego “Speech Commands dataset”, które zawierało ponad 100 tysięcy krótkich nagrań różnych osób wypowiadających słowa takie jak: “yes”, “up”, “down”, “left” itp. Zbiór danych został udostępniony przez Google na otwartej licencji. Ciekawostką jest fakt, iż można pomóc w jego ulepszaniu dodając własne nagrania do zbioru. Całość waży ponad 5GB.

Trenowanie

Trenowanie modelu zajęło ponad 50 godzin na maszynie lokalnej (wydajniejszej obliczeniowo od docelowej), w trakcie których algorytm uczący wykonał 18 tysięcy iteracji.

W trakcie nauki korzystano z narzędzia Tensorboard, który umożliwiał monitorowanie postępów w czasie rzeczywistym. Oprócz numeru aktualnej iteracji dostępny był parametr określający prędkość aktualizacji poszczególnych wag sieci neuronowej, parametr określający procentowo dokładność, z jaką model pomyślnie klasyfikuje próbki, a także wartość entropii krzyżowej przy porównywaniu aktualnego wektora wyników klasyfikacji z poprawnymi etykietami.



Rysunek 6 – Przebieg procesu uczenia

Pomocnym w trenowaniu modelu było automatyczne tworzenie punktów kontrolnych, które zapisywały aktualne wagi sieci neuronowej co każde 100 iteracji, dzięki czemu w przypadku niespodziewanego przerwania trenowania można przywrócić ostatni wynik, zamiast rozpoczynać trening od początku. Co każde 100 iteracji generowana była macierz błędu, zawierająca szczegółowe informacje o ilości poprawnych i błędnych przypadków klasyfikacji poszczególnych próbek.

```
I0730 16:57:38.073667   55030 train.py:243] Confusion Matrix:
 [[258   0   0   0   0   0   0   0   0   0   0   0]
 [  7   6  26  94   7  49   1  15  40   2   0  11]
 [ 10   1 107  80  13  22   0  13  10   1   0   4]
 [  1   3  16 163   6  48   0   5  10   1   0  17]
 [ 15   1  17 114  55  13   0   9  22   5   0   9]
 [  1   1   6  97   3  87   1  12  46   0   0  10]
 [  8   6  86  84  13  24   1   9   9   1   0   6]
 [  9   3  32 112   9  26   1  36  19   0   0   9]
 [  8   2  12  94   9  52   0   6  72   0   0   2]
 [ 16   1  39  74  29  42   0   6  37   9   0   3]
 [ 15   6  17  71  50  37   0   6  32   2   1   9]
 [ 11   1   6 151   5  42   0   8  16   0   0  20]]
```

Rysunek 7 – Macierz błędu

W przypadku klasyfikacji gdzie jest wiele zmiennych, taka macierz jest znacznie bardziej przydatna od pojedynczego wyniku, gdyż daje to informację, jaki rodzaj błędów popełnia sieć. Idealna macierz błędów jest macierzą diagonalną z poprawnymi przypadkami klasyfikacji na przekątnej oraz zerowymi błędami.

Walidacja

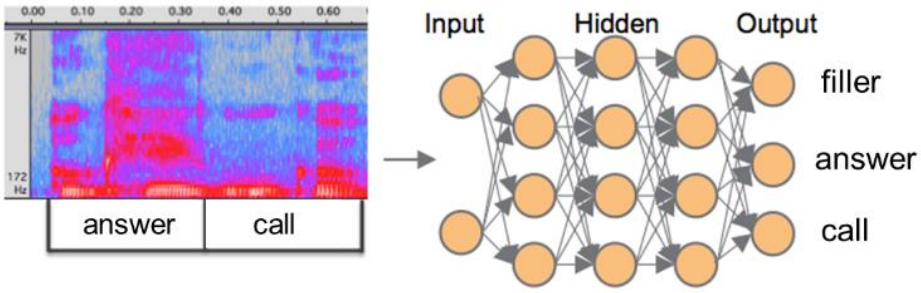
W ramach walidacji modelu zbiór uczący został podzielony na trzy części jeszcze przed procesem trenowania. 80% danych posłużyło do właściwego trenowania sieci, 10% było danymi walidacyjnymi podczas trenowania (były to dane, które służyły do obliczania dokładności modelu) oraz 10% danych testowych, które posłużyły do obliczenia dokładności po zakończeniu trenowania.

Dzielenie danych jest związane ze sprawdzeniem czy model nie stracił zdolności do generalizacji, czyli umiejętności poprawnej klasyfikacji również dla danych, których nigdy wcześniej nie widział. W przypadku, gdy walidacja na danych testowych wypadła znacznie gorzej niż podczas treningu oznacza to, że model dopasował się do danych uczących, nastąpiło przetrenowanie oraz utrata zdolności generalizacji.

Struktura i zasada działania

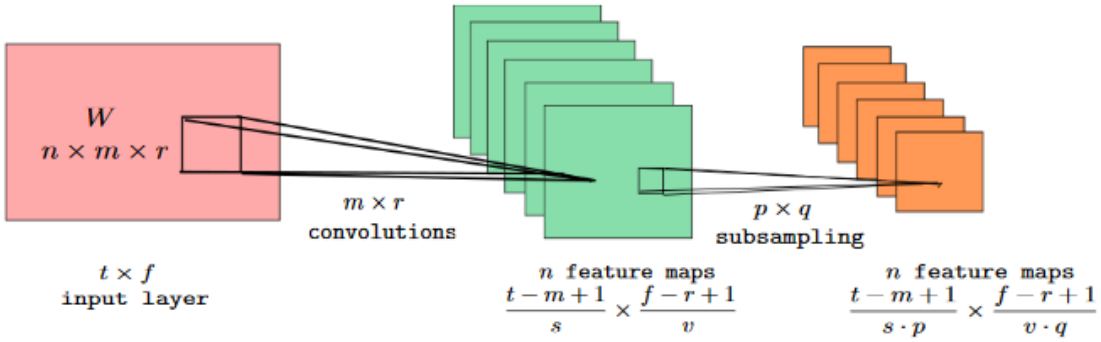
Wykorzystana architektura została oparta o wielowarstwowe konwolucyjne sieci neuronowe, ponieważ w porównaniu do innych (np. rekurencyjnych) są prostsze, szybkie w nauce oraz proste do zrozumienia. Zasada działania jest oparta na podejściu podobnym jak przy rozpoznawaniu obrazów, co z początku wydaje się zaskakujące, zważywszy na postać zapisu pliku audio, jakim jest jednowymiarowy wektor, w którym zapisane są kolejne próbki sygnału.

W pierwszej kolejności definiowane jest okno czasowe, w którym znajduje się dany sygnał, a następnie wygenerowany zostaje obraz, który powstaje poprzez przekształcenie sygnału w spektrogram. Wygenerowane zostają odpowiednie współczynniki cepstralne (zgodnie ze sposobem opisanym w poprzednim rozdziale), a następnie zostaną one podane na wejścia wielowarstwowej konwolucyjnej sieci neuronowej, która dokonuje klasyfikacji.



Rysunek 8 – Spektrogram podawany na wejście sieci neuronowej

Poniżej przedstawiona została wykorzystana struktura. Do warstwy wejściowej podane zostają macierze wraz z odpowiednimi częstotliwościami w funkcji czasu, na każdej z nich wykonane zostaje n konwulsji o rozmiarach m x r. Stworzone zostaje n map cech, dzięki którym zostaną wykryte zależności oraz podobieństwa w danych testowych.



Rysunek 9 – Zastosowana struktura sieci neuronowej

Poniżej można zobaczyć porównanie oraz wartości liczbowe dla poszczególnych warstw sieci neuronowej. Suma wszystkich parametrów sieci neuronowej nie przekroczyła 250 tysięcy, a liczba wykonywanych multiplikacji to niecałe 10 milionów.

type	m	r	n	p	q	Par.	Mul.
conv	20	8	64	1	3	10.2K	4.4M
conv	10	4	64	1	1	164.8K	5.2M
lin	-	-	32	-	-	65.5K	65.5K
dnn	-	-	128	-	-	4.1K	4.1K
softmax	-	-	12	-	-	0.5K	0.5K
Total	-	-	-	-	-	244.2K	9.7M

Tabela 1 – Wartości parametrów sieci neuronowej

Eksport modelu

Efekt finalny, czyli model w postaci sieci po zakończeniu treningu tworzony jest na podstawie pliku w formacie **.ckpt* za pomocą utworzonego w tym celu skryptu. Powstaje tak zwany plik grafu (około 3.5MB) z rozszerzeniem **.pb*, reprezentujący sieć wraz z jej strukturą oraz wagami. Plik ten jest gotowy do przeniesienia na dowolną platformę sprzętową pod warunkiem, że będzie ona w stanie obsłużyć bibliotekę *tensorflow* oraz *Pythona*.

Wydajność i dokładność algorytmów

Sieć neuronowa na zbiorze uczącym osiągnęła wydajność rzędu 85%, natomiast na danych testowych było to zaledwie 50% (podobną dokładność miały współczynniki cepstralne). Słaby wynik mógł być spowodowany niską jakością nagrywania, spowodowaną zastosowaniem budżetowego sprzętu oraz niewłaściwie dobranymi parametrami algorytmu.

Czas wykonania – około 6-7 sekund na urządzeniu wbudowanym (Raspberry PI), około 5-6 sekund na średniej klasy laptopie.

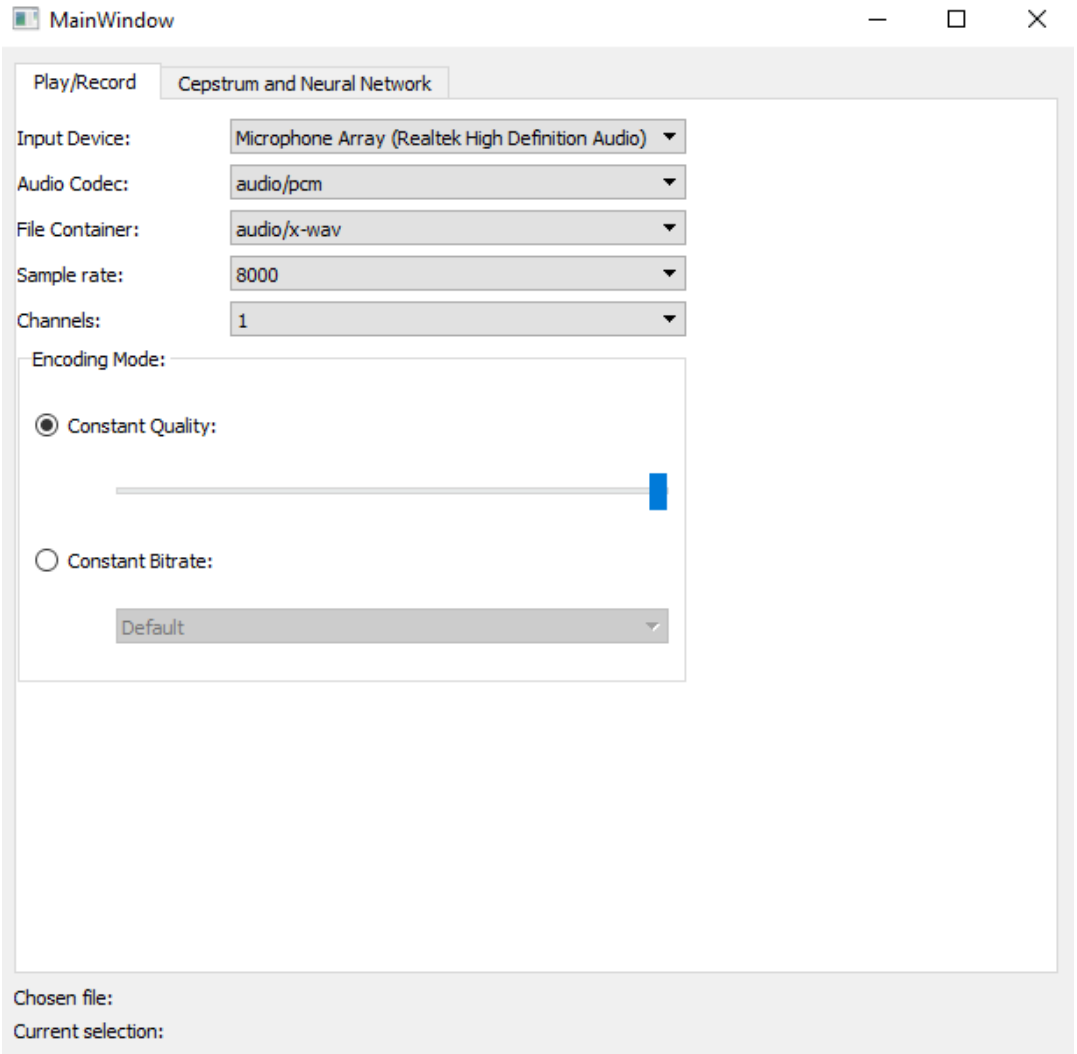
Produkt końcowy

Graficzny interfejs użytkownika

W środowisku *Qt Creator* z wykorzystaniem języka C++ oraz języka znaczników XML opracowaliśmy panel czołowy aplikacji. Składa się on z dwóch zakładek Play/Record oraz Cepstrum and Neural Network. Każda z nich udostępnia zgoła odmienne funkcjonalności, zatem zostanie omówiona osobno.

Zakładka „Play/Record”

Interfejs użytkownika w tej zakładce prezentuje się następująco:

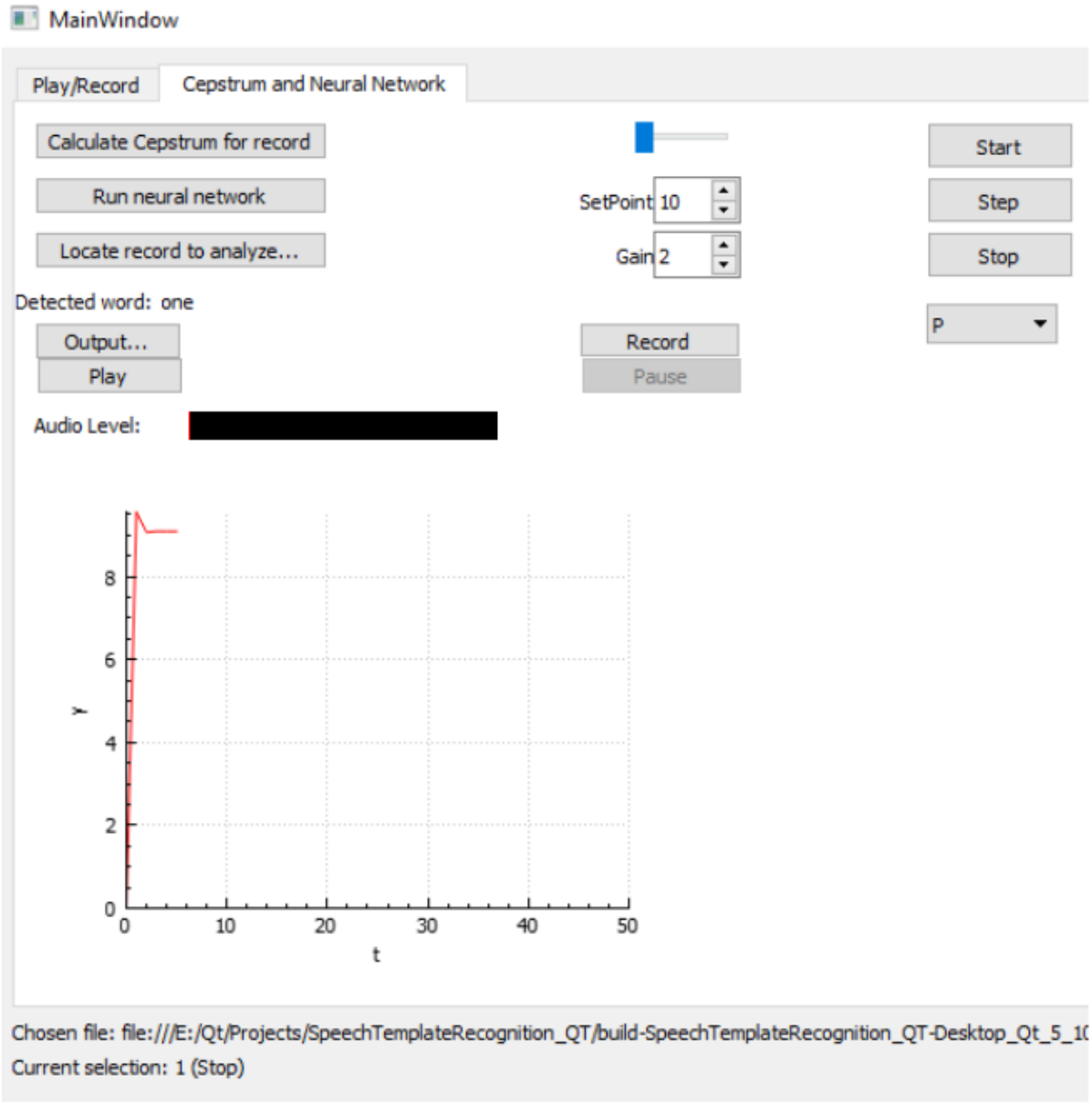


Rysunek 10 – Wygląd zakładki „Play/Record”

Kontrolki zaszyte w opisywanej zakładce pozwalają na modyfikację parametrów akwizycji sygnału. Użytkownik aplikacji ma możliwość wyboru urządzenia wejściowego, kodeka audio, formatu pliku zawierającego nagranie, częstotliwość próbkowania audio oraz liczby rejestrowanych kanałów w nagraniu. Co więcej, aplikacja umożliwia zmianę jakości nagrania (a co za tym idzie zmianę wartości zajmowanego przez plik z nagraniem miejsca). Kontrolka umiejscowiona najniżej umożliwia ustawienie stałej wartości bitrate. Dostęp do wszystkich wymienionych wyżej parametrów akwizycji sygnału uzyskaliśmy dzięki bibliotekom *QtAudio* udostępnianym na licencji freeware przez wspomnianą już platformę *Qt*.

Zakładka „Cepstrum and Neural Network”

Interfejs użytkownika zakładki ukazany jest na ilustracji poniżej:



Rysunek 11 – Wygląd zakładki „Cepstrum and Neural Network”

Ukazana zakładka udostępnia większość zaimplementowanych funkcjonalności. W dolnej części okna znajdują się pola tekstowe informujące użytkownika o ścieżce do pliku dźwiękowego, który zostanie poddany analizie (Chosen file) oraz o ostatnio rozpoznanym wzorcu (Current selection).

W centralnej części okna znajduje się wykres ukazujący przebieg czasowy odpowiedzi symulowanego układu. Powyżej wykresu znajduje się zestaw przycisków i kontrolek umożliwiający nagrywanie oraz odtwarzanie dźwięku. Przycisk Output… umożliwia wybór ścieżki pod którą zostanie zapisane kolejne zarejestrowane przez użytkownika nagranie. Play, Record oraz Pause zgodnie z opisem umożliwiają odpowiednio odtworzenie, wykonanie nowego nagrania oraz zatrzymanie nagrania. Jako dodatkową funkcjonalność ułatwiającą użytkownikowi interakcję z panelem nagrywania jest wskaźnik natężenia dźwięku nagrania usytuowany poniżej omawianych przycisków, na prawo od napisu Audio level.

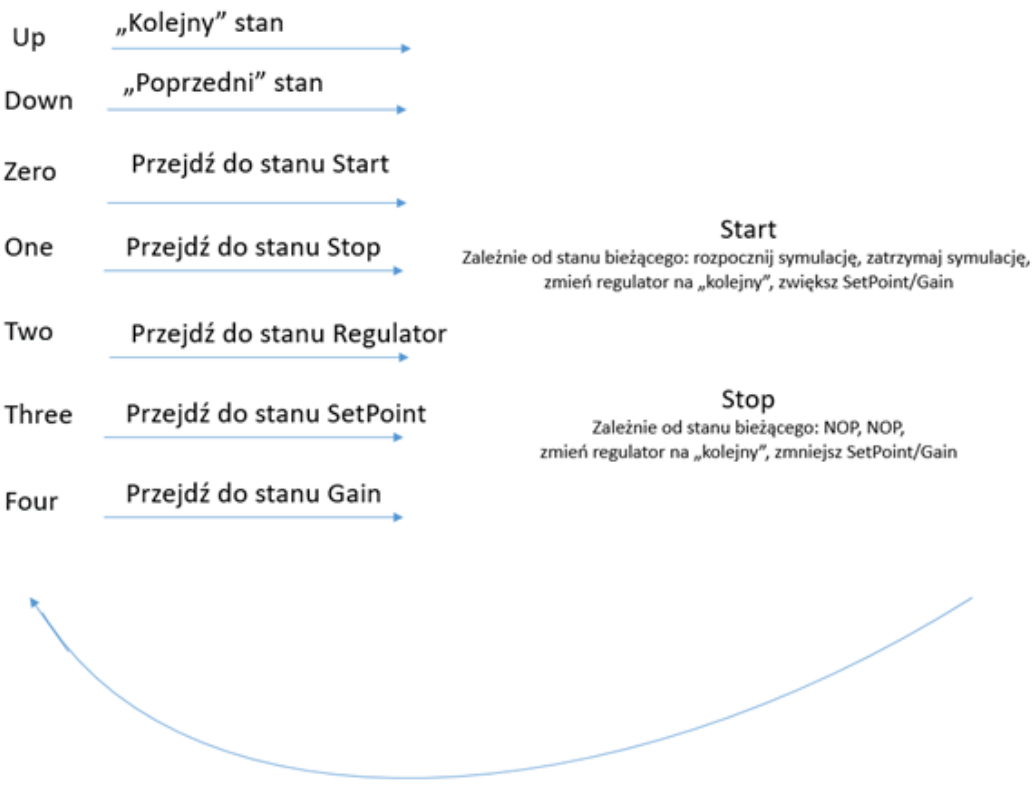
Przyciski Start, Stop oraz Step umożliwiają kontrolowanie przebiegu symulacji. Dzięki nim można kolejno symulację rozpocząć/wznowić, wykonać jeden krok symulacji, czy też całkowicie ją zatrzymać. Poniżej opisanych przycisków znajduje się lista rozwijana udostępniająca wybór regulatora obiektu. Zaimplementowano regulację P oraz PI.

Znajdujący się w górnej środkowej części okna suwak daje dostęp do zmiany prędkości symulacji - im bardziej jest on wysunięty na prawo tym symulacja przebiega wolniej. Poniżej suwaka znajdują się tzw. combo box'y. Służą one zgodnie z opisem do zmiany wzmocnienia (Gain) oraz wartości zadanej (SetPoint) regulatora.

Najważniejszą częścią zakładki Cepstrum and Neural Network z punktu widzenia cyfrowego przetwarzania sygnałów jest sekcja umożliwiająca wskazanie pliku do analizy (Locate record to analyze) oraz próba dopasowania wskazanego nagrania do wzorca przy użyciu współczynników cepstralnych (Calculate Cepstrum for a record), oraz z wykorzystaniem sieci neuronowej (Run neural network).

Nawigacja po aplikacji

Wykorzystanie algorytmów cyfrowego przetwarzania w niniejszym projekcie służy nawigacji po GUI. Diagram zamieszczony poniżej opisuje, jaką akcją skutkuje rozpoznanie wzorca mowy.



Rysunek 12 – Diagram reakcji programu na zarejestrowane polecenia użytkownika

Powyższa grafika ukazuje, że zaimplementowaliśmy reakcję na dziewięć rozpoznawanych wzorców mowy. Słowa "up", "down", "zero" … "four" pełnią funkcję nawigacyjną, natomiast "start" oraz "stop" są swego rodzaju egzekutorami. Zależnie od bieżącego stanu nawigacji wykonują przypisane do stanu zadania "startowe/inkrementacyjne" oraz "zatrzymujące/dekrementacyjne". Sterowanie głosowe jest więc funkcjonalnym ekwiwalentem wszystkich przycisków umożliwiających wpłynięcie na przebieg symulacji.