Politechnika Wrocławska

WYDZIAŁ ELEKTRYCZNY
KIERUNEK: AUTOMATYKA PRZEMYSŁOWA

Szymon Piotr Strong
Nr albumu: 256001

Wydobywanie cech sygnałów akustycznych urządzeń elektrycznych za pomocą funkcji Matlab 'a dla potrzeb głębokiego uczenia

Extraction of acoustic signals features of an electric device by means of Matlab functions for deep learning needs

INŻYNIERSKA PRACA DYPLOMOWA

Studia: Stacjonarne
Opiekun: ***dr inż. Radosław Nalepa***
Katedra: W05/K2 ZSS

ocena	*data, podpis opiekuna*

Wrocław, 2022

SPIS TREŚCI:

1. WSTĘP

Zapoczątkowany w Niemczech Przemysł 4.0 (niem. Industrie 4.0) definiuje się poprzez zmiany w rozwiązaniach przemysłowych, których celem jest reorganizacja i modyfikacja zarządzania procesami przemysłowymi [1]. Źródłem wiedzy na temat jak przeprowadzić takie przemiany jest przetworzenie dużej ilości różnorodnych danych w krótkim czasie (ang. Big Data) [2] pochodzących z tych procesów. Duża ilość danych pochodzi z integracji systemów IT (ang. Information Technology) w procesy przemysłowe Na podstawie kluczowych informacji wydobytych z danych o danym procesie, można podejmować inteligentne decyzje, które prowadzą do stworzenia tzw. inteligentnych fabryk. Sposoby na uzyskanie tych informacji są zawarte w dziedzinie nauki o danych (ang. Data Science). Data Science łączy dziedziny: statystyki, matematyki, programowania, sztucznej inteligencji oraz uczenia maszynowego w celu odkrycia zjawisk występujących w zestawie danych o danym obiekcie. Informatyzacja procesów przemysłowych pozwoliła na jego stworzenie. Przykładem informatyzacji może być Internet Rzeczy (eng. Internet of Things).Jest on systemem przedmiotów fizycznych, które wymieniają się informacjami między sobą bez konieczności ingerencji człowieka w ten proces[3]. Stało się to możliwe poprzez zawarcie w tych przedmiotach różnego rodzaju czujników i oprogramowania, a także komunikacji przeprowadzonej przez internet. Wymiana informacji poprawia działanie tych urządzeń lub pozwala na wyciągnięcie prawidłowych wniosków co do zarządzania nimi. Fundamentem tego rozwiązania jest właśnie Data Science. Produkt bazujący na Data Science ma cykl życiowy składający się z następujących kroków [4]:

1. Zrozumienie rynku - wiedza na temat funkcje jakie powinien mieć produkt, by przyniósł korzyści komercyjne.
2. Zebranie i zrozumienie danych.
3. Modelowanie danych - wydobycie i selekcja cech, trenowanie i późniejsza ocena modelu.
4. Wdrożenie modelu.
5. Akceptacja klienta.

Ta praca inżynierska skupia się na punktach 2 i 3 powyższej listy.

Aby zebrać, zrozumieć i zamodelować dane procesu przemysłowego zdecydowano się stworzyć środowisko do pozyskiwania informacji w postaci cech sygnałów pochodzących od przenośnych narzędzi elektrycznych w różnych stanach pracy. Narzędzia te są często stosowane w przemyśle, więc budowa systemu identyfikacji w jakim stanie pracy jest narzędzie może znaleźć szerokie zastosowanie. Narzędzie elektryczne jest źródłem sygnałów: elektrycznych, termicznych i akustycznych. Zdecydowano się na wykorzystanie sygnałów akustycznych ze względu na bezinwazyjną metodę pomiaru w stosunku do mierzonego obiektu oraz łatwość w zbudowaniu stanowiska pomiarowego (rozdział 2), w przeciwieństwie do wykorzystywania pozostałych typów sygnałów. W tej pracy inżynierskiej założono rozpoznanie stanów pracy narzędzia przy pomocy Data Science. Narzędziem jest wiertarka elektryczna, stanami pracy jest wiercenie lub wyciągnięcie wiertła z drewnianych kłód. System identyfikacji stanu pracy narzędzia może być wykorzystywany także do poprawy prognostyki i usprawnienia diagnostyki w przypadku jego defektu[5]. Taki system więc poprawia efektywność pracy zakładu, w którym zostanie zaimplementowany. Data Science wytycza w ten sposób nowe horyzonty modernizacji przemysłowych.

Wykonanie systemu identyfikacji przeprowadzono poprzez zbudowanie klasyfikatora sygnałów akustycznych w oparciu o głębokie uczenie. Głębokie uczenie definiuje się jako wydobycie wysokopoziomowych informacji o danych na podstawie informacji niskopoziomowych (np.stwierdzenie, że wypowiedziano dane słowo na podstawie sygnału akustycznego wymowy) poprzez podział zestawu danych na wiele poziomów reprezentacji i abstrakcji [6]. Automatyzuje ono budowę adaptacyjnego modelu klasyfikującego, poprzez wynajdowanie uogólnionych wzorców w zestawie danych. Model ten jest wykorzystywany do identyfikacji danych, które są mu nieznane, lecz zawierają znajome mu trendy. Głębokie uczenie wprowadza więc Data Science do aplikacji przemysłowych. Głębokie uczenie o danym obiekcie przeprowadza się na podstawie uczenia sztucznej sieci neuronowej. Prawidłowe jej dobranie i nastawienie pozwala na zbudowanie wspomnianego klasyfikatora (rozdział 6).

Środowisko do pozyskiwania informacji rozumie się nie tylko poprzez zbudowanie wspomnianego klasyfikatora, lecz także stworzenie zrozumiałej ścieżki rozwoju w przypadku dalszego rozwijania takowego projektu poza zakres tej pracy. Rozwój ten powinien się opierać o zaangażowanie większej liczby osób. Na początku pracy z projektem, którego rozwój nie zakończył się, istotną kwestią dla nowych osób jest wprowadzenie do tego projektu. Poprzez zorganizowanie pracy w projekcie w sposób przejrzysty i zrozumiały wprowadzenie to będzie efektywniejsze i szybsze.

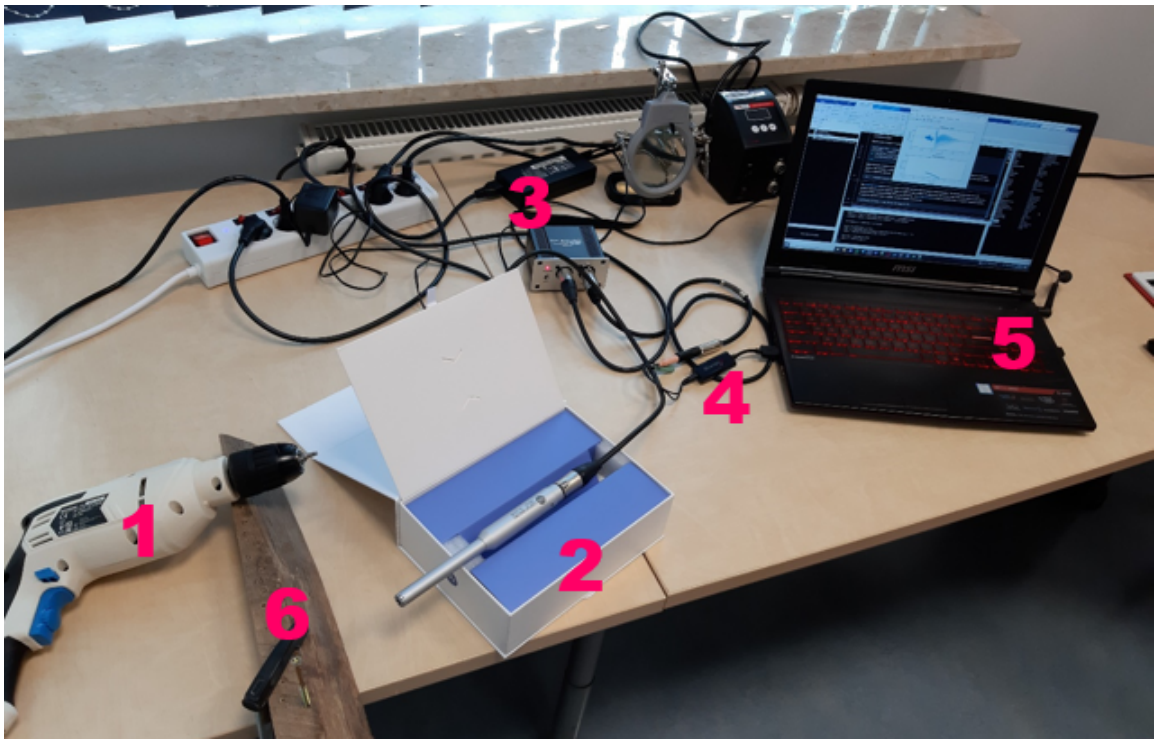
Praca inżynierska składa się z ośmiu rozdziałów wyjaśniających w jaki sposób spełniono założenia projektowe. Rozdział 1 obejmuje wstęp. W rozdziale 2 opisano w jaki sposób wyglądało stanowisko pomiarowe i jak przeprowadzono pomiary akustyczne. Opisano w nim używany sprzęt oraz skrypt pozwalający na archiwizację nagrań. Rozdział 3 obejmuje przyjętą metodykę działania w całej dalszej pracy po stworzeniu odpowiednio dużego zestawu danych pomiarowych. Zamieszczono w nim także opis organizacji plików audio, skryptów Matlaba oraz wyników prac w szeregu katalogów. Rozdział 4 opisuje nadanie etykiet nagraniom, wczytanie plików audio w przestrzeń roboczą Matlaba oraz wykonanie wstępnego przetwarzania danych przed uczeniem sieci neuronowej. W Rozdziale 5 opisano sposób w jaki wydobyto cechy z sygnałów dźwiękowych. Rozdział 6 przedstawia strukturę testowanych sieci oraz drogę, którą przebyto do osiągnięcia najlepiej wytrenowanego modelu. Podsumowanie działań i spis kolejnych możliwych kroków zamieszczono w rozdziale 7. Wykorzystaną literaturę opisano w rozdziale 8.

2. STANOWISKO POMIAROWE I POMIARY

Warunkiem pomiarów wysokiej jakości są odpowiednie: środowisko, w którym wykonywane są pomiary, narzędzia pomiarowe, oraz procedury pomiarowe. Z punktu widzenia pomiarów akustycznych najważniejsza jest cisza, w jakiej wykonuje się pomiar, tak aby uniknąć zakłóceń pochodzących z otoczenia. Środowisko, czyli laboratorium gdzie wykonano pomiary zapewniało takie warunki.

Następnym warunkiem do spełnienia są dobre narzędzia pomiarowe wraz z poprawnie wykorzystywanym obiektem pomiarowym. Stanowisko składało się z następujących elementów:

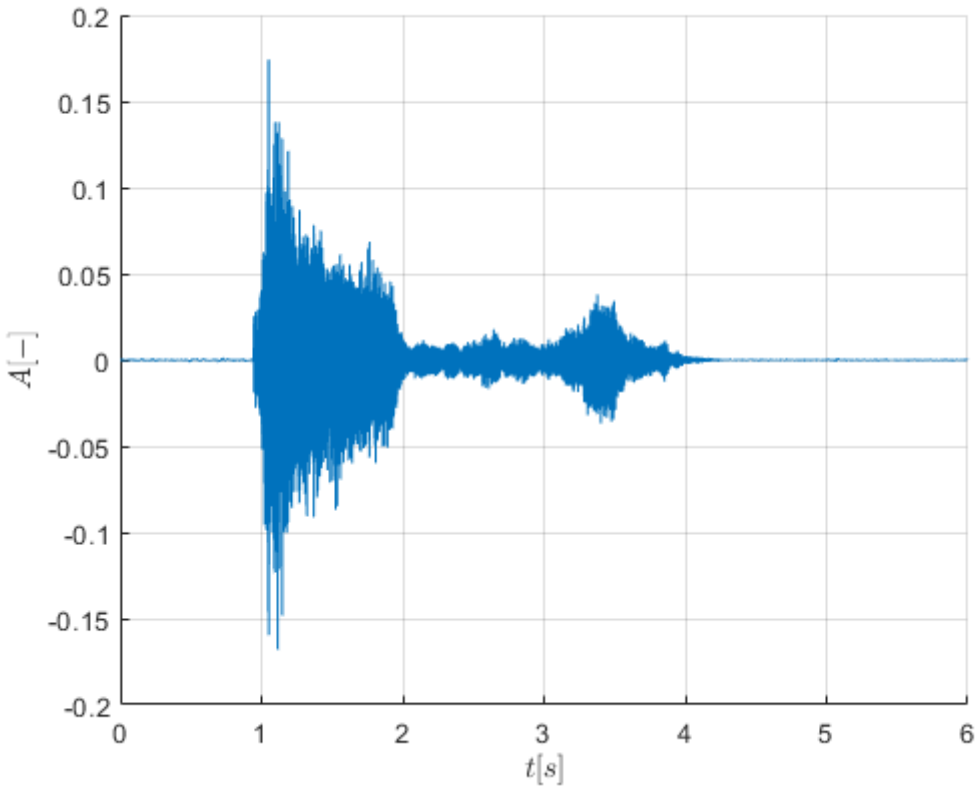
- Wiertarka MacAllister MSHD600 600 W, 0- 3000 rpm, opcją 6-stopniowego ograniczenia spustu. Wiertło miało długość 75 mm i 3 mm średnicy.
- Kłody z drewna świerkowego i dębowego.
- Zaciski mocujących kłody do biurka.
- Mikrofon Sonarworks XREF 20 o zakresie częstotliwości 20 Hz do 20 kHz. Maksymalny poziom dźwięku to 128 dB.
- Zasilacz mikrofonu RH Sound Phantom Power Supply 9 V.
- Karta dźwiękowej Sound Blaster G1 o częstotliwości próbkowania równej 96 kHz
- Komputer z oprogramowaniem Matlab.



Rysunek 2.1: Stanowisko pomiarowe. 1-wiertarka MacAllister MSHD600, 2-mikrofon Sonarworks XREF 20, 3-zasilacz mikrofonu RH Sound Phantom Power Supply 9 V, 4 - karta pomiarowa Sound Blaster G1, 5 - komputer, 6 - kłoda dębowa

Tok pomiaru można streścić w postaci następujących czynności i zjawisk:

- Uruchamiano skrypt pomiarowy, który rejestrował 6-sekundowy odczyt z mikrofonu. Czas ten dobrano empirycznie, ponieważ samo wiercenie nie zajmowało nigdy więcej niż około 4 sekundy. Zapas czasu był przeznaczony na odsunięcie ręki od komputera i złapanie w kłody w celu stabilizacji wiercenia.
- Rozpoczynano proces wiercenia z odpowiednim ograniczeniem prędkości wiertarki znajdującym się na spuście. Starano się przykładać do wiertarki zawsze taką samą siłę. Było to jednak robione ręcznie, co wprowadziło niepewności pomiarowe.
- Zewnętrzna karta dźwiękowa próbkowała sygnał z mikrofonu z częstotliwością 96 kHz.
- Poprzez skrypt generowany był plik nagrania o formacie .wav. W celu sprawdzenia poprawności pomiaru tj. czy całość wiercenia uchwycono w 6-sekundowym przedziale czasu skrypt automatycznie wykreślał przebieg dźwięku w czasie. Przykładowy wykres znajduje się na rysunku 2.2 Po obserwacji wykresu stwierdzono czy pomiar powinien być zapisany.
- Zapis pliku pod nazwą generowaną według klucza pokazanego na rysunku 2.3.



Rysunek 2.2: Przykładowy wykres pomiaru amplitudy dźwięku (A) w czasie.

Pomiar amplitudy dźwięku został znormalizowany przez funkcję Matlaba getaudiodata [7] do typu danych double. Dla tego typu danych zakres mierzonych wartości to [-1, 1]. W [7] nie

zamieszczono dokładnego algorytmu jak normalizowano odczyt mikrofonu do wspomnianego zakresu.

[zestaw danych]_[typ drewna]_[nastawa prędkości wiertarki]_[liczba porządkowa].wav

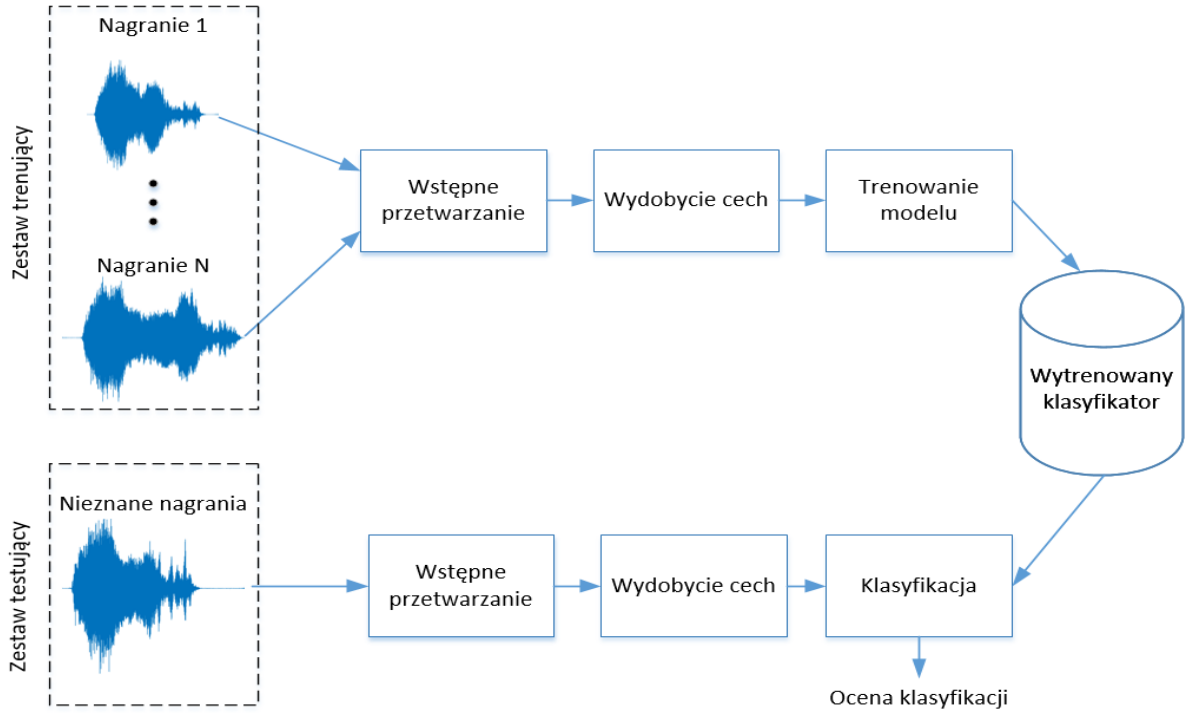
Rysunek 2.3: Klucz nazywania nagrania pomiarowego.

Dla idealnego zbadania obiektu wymaga się aby nie zmieniał on swoich właściwości w trakcie procedury pomiarowej. W rzeczywistości jednakże dla tak intensywnej pracy wiertarki w krótkim odstępie czasu, warunek ten jest trudny do spełnienia. Wiertarka grzeje się, co może wpłynąć na dźwięk jaki się z niej wydobywa. Dodatkowym źródłem błędu jest wiertło, które także zmienia swoje właściwości podczas pomiarów. Wirując w drewnie poprzez tarcie wytwarza ciepło, co wpływa na wydawany dźwięk. Wiertło również ściera się. Po długiej eksploatacji będzie więc ono inaczej pracować dla kolejnych nagrań, a więc inaczej brzmieć. Porównując ze sobą uzyskane sygnały dźwiękowe z tego samego drewna zakłada się, że wiertarka wierciła w tym samym, jednolitym materiale. W praktyce np. świerkowa kłoda posiada słoje, które powodują, że kłoda nie jest tak samo twarda na całej swojej objętości. Błędy pomiarowe generowane są także przez odgłosy otoczenia np. szum wiatraka komputera. Istnieje więc wiele źródeł zakłóceń sygnału. Należy więc stosować procedury im przeciwdziałające. Takie jak zamykanie okna w laboratorium na czas pomiarów, aby nie dostawał się do niego dźwięk z zewnątrz lub przerwa w eksploatacji wiertarki, aby schłodzić wiertło i wiertarkę. Gdy odczyt z mikrofonu osiągał wartości równe ekstremom zakresu [-1, 1] oznaczało to zbyt bliskie położenie mikrofonu w stosunku do wierconej kłody. W takim wypadku należało przesunąć mikrofon na dalszą odległość. Finalnie przyjęto odległość około 80 cm od kłody. Zaznaczono wtedy miejsce położenia mikrofonu tak, aby odległość była stała dla każdego pomiaru. Większa odległość od kłody pozwoliła także chronić mikrofon od tworzonego podczas wiercenia pyłu.

Zebrano 306 nagrań pomiarowych, które były później używane do zbudowania klasyfikatora. Były one podstawą do dalszej pracy.

3. METODYKA DZIAŁANIA W PROJEKCIE

Ogromny wpływ na finalny rezultat w tej pracy miał przyjęty zbiór metod, wykorzystywanych do wypełnienia kolejnych założeń projektowych. Dlatego na początku projektu zgłębiono literaturę, rozwiązania stosowane w przemyśle [5] oraz przykłady skryptów firmy Mathworks [8], [9]. Przyjęta ścieżka działania jak na rysunku 3.1 jest podobna to tej, którą zaproponowano w źródle [8]. Rozpoznawano tam ludzką mowę. Końcowy wynik w tamtym projekcie cechował się wysoką dokładnością walidacyjną (podrozdział 6.2), więc stwierdzono, że do rozpoznawania dźwięku wiertarki także można przyjąć podobną ścieżkę działania.

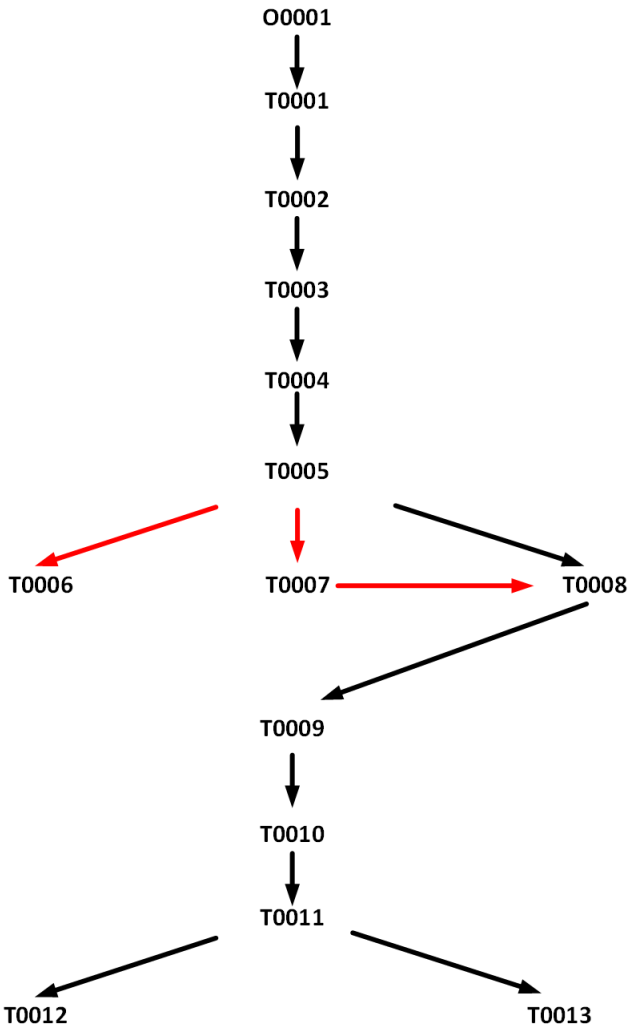


Rysunek 3.1: Schemat działania po zebraniu danych pomiarowych.

Jak widać na rysunku 3.1 całość trzystu sześciu nagrań została rozdzielona na dwa zestawy: trenujący i testujący. Zestaw trenujący jest tam przedstawiony jako nagrania 1 do N, gdzie N to całkowita liczba nagrań w tym zestawie. Nagrania nieznane z punktu widzenia sieci neuronowej pochodzą z zestawu testującego. Nie wykorzystywano ich w procesie uczenia. Oba zestawy zostały poddane transformacjom, które pomagają w lepszej identyfikacji trendów w danych. Są nimi proces wstępnego przetwarzania i wydobywania cech. Na podstawie wydobytego z danych trenujących wektora uczącego następuje trenowanie modelu. Końcowym efektem jest otrzymanie klasyfikatora, który jest w stanie rozpoznać dane z wektora testującego (otrzymanego w ten sam sposób jak wektor uczący) i w procesie klasyfikacji przydzielić mu odpowiednią etykietę (podrozdział 4.1). Ocenę klasyfikacji przeprowadza się na podastawie wielkości z podrozdziału 6.2. Każda część schematu działania jest procesem, który posiada swój szczegółowy opis w jednym z rozdziałów 4, 5 i 6.

3.1. Organizacja plików

W trakcie realizacji projektu wiedzę czerpano z wielu źródeł w celu rozwiązywania napotykanych problemów. Tworzono oddzielne skrypty testujące dane rozwiązanie. W przypadku otrzymania zadowalających wyników kopiowano lub modyfikowano kod w zbiorczym skrypcie. Zestawy danych, na których pracowano były umieszczone w wyznaczonych folderach. Aby pracować na danym zestawie należało zmienić kod określający lokalizację takiego folderu. Efektywna praca z plikami skryptowymi oraz pomiarowymi wymagała więc rozeznania w naniesionych wcześniej zmianach lub przyjętej akurat ścieżki rozwoju. Rozeznanie to wymagało pamiętania przez użytkownika pewnych założeń programu oraz obycia się z tym konkretnym skryptem. Takowe chaotyczne podejście odbierało stworzonemu środowisku możliwość zrozumienia go przez samodzielnie pracujące z nim osoby trzecie. Ten sposób działania jest sprzeczny z założeniem przedstawionym w rozdziale 1, gdzie założona ścieżka rozwoju jest przejrzysta. Wymagana była więc zmiana organizacji całości plików. Przeprowadzono ją poprzez stworzenie hierarchicznego szeregu katalogów (schemat hierarchii przedstawiono na rysunku 3.2), w których znajdują się pliki pomiarowe, skrypty Matlaba lub zrzuty ekranu procesu uczenia sieci neuronowej tak. Wspomniany zbiorczy skrypt został rozbity na pomniejsze charakteryzujące się wykonywaniem określonej funkcji.



Rysunek 3.2: Schemat organizacji katalogów.

Na powyższym rysunku znajduje się organizacja katalogów. Zdecydowano się na typ organizacji hierarchicznej, gdzie skrypty zlokalizowane są w taki sposób, aby bazowały na zawartości wyłącznie poprzednich katalogów. Opis zawartości znajduje się w tabeli 3.1. Pliki .mat w tej tabeli są rezultatami w postaci przestrzeni roboczej skryptów .m w tym samym katalogu. Przykładowo Features.mat jest rezultatem uruchomienia skryptu Features.m.

Tabela 3.1: Opis zawartości katalogów w stworzonym środowisku.

Nazwa katalogu	**Zawartość**
O0001	folder T0001
T0001	folder T0002, wszystkie pliki pomiarowe w formacie .wav
T0002	folder T0003, wyselekcjonowane pliki pomiarowe w formacie .wav
T0003	folder T0004, labels_from_file_names.m, labels_from_file_names_4.m, labels_from_file_names.mat
T0004	folder T0005, Audio_Data_Store.m, Audio_Data_Store.mat
T0005	folder T0006, folder T0007, folder T0008, Features.m, Features.mat
T0006	Classification_Learner_workspace.m, Classification_Learner_workspace.mat
T0007	Feature_selection.m, Feature_selection.mat
T0008	folder T0009, Features_cell.m, Features_cell.mat
T0009	folder T0010, Sorted_features .m, Sorted_features.mat
T0010	folder T0011, Trained_model.m, Trained_model.mat
T0011	folder T0012, folder T0013, Results_workspace.m,
T0012	pliki przestrzeni roboczej w formacie .mat
T0013	zrzuty ekranu z procesu uczenia w formacie .png

4. WSTĘPNE PRZETWARZANIE

W tym rozdziale opisane zostały procesy, które odbywają się bezpośrednio po zgromadzeniu zestawu danych, na których będzie bazować późniejsze wydobywanie cech, uczenie, a także testowanie klasyfikatora. Tymi procesami są wprowadzenie danych w przestrzeń roboczą Matlaba, etykietyzacja nagrań oraz wstępne przetwarzanie danych. Przetwarzanie to jest wsparciem dla wynajdowania trendów, rozpoznawania charakterystycznych zjawisk lub oczyszczania z danych wprowadzających błędy. Ma ono za cel poprawę tempa i jakości uczenia sieci neuronowej.

4.1. Etykietyzacja

Sposobem na identyfikację nagrań pomiarowych jest nadanie im odpowiednich etykiet tzn. przyporządkowania im nazw, które odnoszą się do fizycznych cech pomiaru. Może być to typ drewna, prędkość wiertarki, czy kierunek ruchu wiertarki względem drewna. Etykieta jest podstawową informacją dla sieci neuronowej o przetwarzanym sygnale. W tej pracy testowano modele dla dwóch zbiorów klas. Oba zbiory miały postać kategoryczną, czyli wysokopoziomowej informacji o czynności wykonywanej podczas pomiaru sygnału. Pierwszy zbiór miał postać dwóch etykiet tzw. klas, w których zawarto informacje w jakim kierunku obracało się wiertło (czy było wkręcane bądź wykręcane z kłody). Dla drugiego zbioru do tego dodano jeszcze informację o rodzaju drewna, w którym wiercono, tworząc cztery rozłączne klasy. Etykiety w tym zestawieniu prezentowały się w następujący sposób:

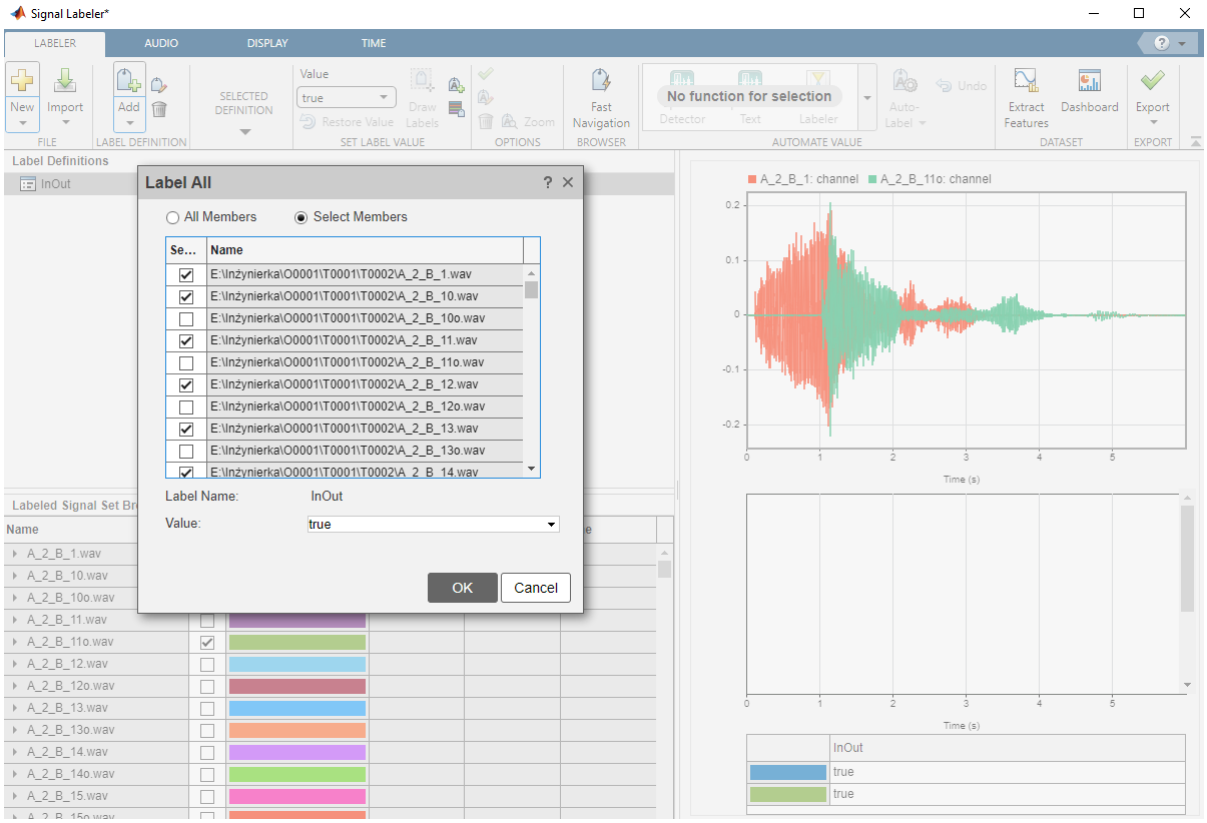
1. wiercenie w świerku,
2. wyciąganie wiertla z świerku,
3. wiercenie w dębie,
4. wyciąganie wiertla z dębu.

Głównie skupiono się na budowaniu klasyfikatora binarnego, o dwóch klasach do rozpoznania, czyli samych czynności wiercenia lub wyciągania wiertla. Dla obu zestawów klas używano tego samego zbioru nagrań. Nadanie etykiet przeprowadzono na dwa sposoby. Poprzez aplikację Matlaba Signal Labeler oraz na podstawie nazwy pliku .wav.

4.1.1. Aplikacja Signal Labeler

Na rysunku 4.1 przedstawiono ręczny sposób nadawania etykiet [10]. Najpierw należy importować pliki .wav do aplikacji. Przyjęta organizacja plików pomaga w tym zadaniu z uwagi na stworzenie już gotowego folderu T0002, gdzie znajdują się wyselekcjonowane pliki. Importowanie więc jest przeprowadzane bardzo prosto. Jedynie należy wybrać folder, z którego chce się zaimportować pliki .wav. Wewnętrzna aplikacja Matlaba Signal Labeler oferuje możliwość stworzenie własnej etykiety. W przypadku dwóch klas stworzono jedną etykietę o charakterze logicznym. Ma ona postać atrybutu. Można także stworzyć etykietę na podstawie cech sygnałów lub danych obszarów czy próbek w sygnale. Możliwym jest także użycie predefiniowanych funkcji, które automatycznie nadają etykiety sygnałom. Jako, że nie posiadano wymaganego doświadczenia z takimi rozwiązaniami zdecydowano się ich nie implementować w procesie etykietyzacji. Praca z tą aplikacją opierała się więc na ręcznym przydzielaniu klasy w polu Label All (rysunek 4.1). Przy dużej liczbie nagrań używanie aplikacji Signal Labeler w taki

sposób jest powolne. Końcowym wynikiem pracy z aplikacją jest eksportowanie macierzy o typie categorical do przestrzeni roboczej.



Rysunek 4.1: Etykietyzacja przy pomocy aplikacji Signal Labeler.

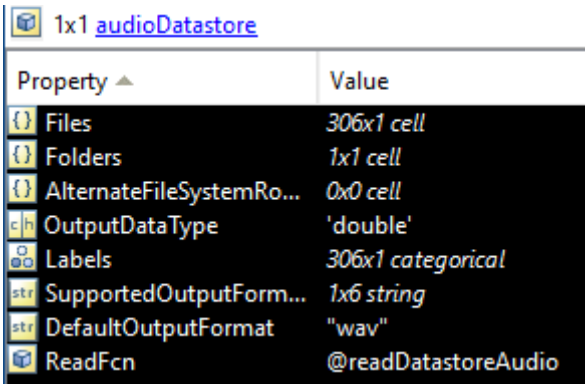
4.1.2. Nazwa pliku .wav

Nagrania zapisano w postaci .wav tak jak pokazano na rysunku 2.3. Nazywano je według przyjętego klucza, który został wykorzystany do procesu automatyzacji nadawania etykiet. Z folderu T0002, gdzie znajdują się wyselekcjonowane pliki pomiarowe, pobrano do przestrzeni roboczej nazwy plików w postaci macierzy categorical. Jest to jednak zabieg, który z każdej nazwy tworzy indywidualną klasę. W celu uzyskania dwóch klas należało dokonać transformacji macierzy na format string i sprawdzić czy dany string w nowo powstałej macierzy zawiera znak "o". Oznacza on, że dane nagrywano proces wyciągania wiertła z drewna. Otrzymuje się w ten sposób binarny wektor, który po transformacji na macierz categorical jest pożądanym wynikiem etykietyzacji.

Użycie takiego sposobu przyśpieszyło proces nadawania etykiet. Każdy inny zestaw danych opisany podobnym kluczem będzie mógł skorzystać z tego skryptu. W przypadku innego wzorca nazewnictwa zmiana szukanych znaków w skrypcie jest czynnością łatwą i intuicyjną. Przykładem jest wygenerowanie wektora etykiet dla czterech klas. W tamtym przypadku sprawdzano znak odpowiadający za typ drewna w danym miejscu stringu. Wyniki dla dwóch i czterech klas zapisano w folderze T0004.

4.2. Wprowadzenie danych do przestrzeni roboczej

Wgrywając pliki do przestrzeni roboczej Matlaba warto zachować informację o ich lokalizacji. Rozumie się przez to, że oprócz surowych danych w postaci amplitudy dźwięku w dziedzinie czasu, podaje się informację o tym z jakiego pliku pochodzą dane, format tego pliku oraz jego lokalizację. Funkcje te oferuje Audio Data Store [8], czyli obiekt w środowisku Matlab, który figuruje w kolejnych skryptach jako baza danych pomiarowych. Wczytywane są z niej nagrania w postaci numerycznych macierzy w celu dalszego przetwarzania. Audio Data Store jest więc źródłem danych w przestrzeni roboczej w każdym kolejnym skrypcie. Audio Data Store, tworzony na podstawie folderu T0002, nie posiada etykiet. Przygotowane miejsce na macierz categorical jest puste. Macierz etykiet jest dopiero później wgrywana do Audio Data Store czyniąc z niej kompletną bazę danych, tak jak przedstawiono to na rysunku 4.2.



Rysunek 4.2: Właściwości Audio Data Store.

Następnym krokiem było rozdzielenie zestawu danych w postaci Audio Data Store na dwie części: zestaw trenujący sieć oraz zestaw testujący nazywany też walidacyjnym. Wszystkich nagrań jest 306. Pomimo wysokiej częstotliwości próbkowania (96 kHz) nie jest to zestaw zawierający dużą ilość danych [11]. Należy więc przyjąć taką proporcję podziału, która pozwala na sterowanie odpowiednio dużego zestawu trenującego, który jest w stanie nauczyć sieć wychwytywania trendów w danych oraz zestawu testującego zdolnego do reprezentacji większości wariacji. Wariację rozumie się poprzez wszystkie możliwe zjawiska, które mogą wystąpić w danych, a co za tym idzie cechach sygnałów. Na podstawie [11] przyjęto podział, gdzie 80% pierwotnego zestawu danych (tj. 245 z 306 nagrań) przypada na zestaw trenujący, a pozostałe 20% na zestaw testujący. Stosowano go też w przykładowej sieci z [8]. Stosunek ten przyniósł zadowalające wyniki (tabela zbiorcza 6.1), więc stwierdzono, że stosowanie innego nie jest konieczne.

Podział na zbiór trenujący i testujący zachował prawie stałą proporcję klas. Nie korzystano z losowego doboru nagrań do zestawów. W każdym zestawie trenującym i testującym zawarto nagrania o wszystkich klasach. Udziały procentowe nagrań o danej klasie w stosunku do wszystkich nagrań w danym zestawie przedstawiono w tabelach 4.1, 4.2, 4.3 i 4.4.

Tabela 4.1: Podział etykiet w zestawie trenującym w wariancie z dwoma klasami.

Nazwa klasy	**Liczba**	**Procentowy udział [%]**
Wyciągnięcie wiertła	112	45,71
Wiercenie	133	54,29

Tabela 4.2: Podział etykiet w zestawie testującym w wariancie z dwoma klasami.

Nazwa klasy	Liczba	Procentowy udział [%]
Wyciągnięcie wiertła	128	45,90
Wiercenie	133	54,10

Tabela 4.3: Podział etykiet w zestawie trenującym w wariancie z czterema klasami.

Nazwa klasy	Liczba	Procentowy udział [%]
Wiercenie w dębie	60	24,49
Wyciągnięcie wiertła z dębu	40	16,33
Wiercenie w świerku	73	29,80
Wyciągnięcie wiertła z świerku	72	29,38

Tabela 4.4: Podział etykiet w zestawie testujący, w wariancie z czterema klasami.

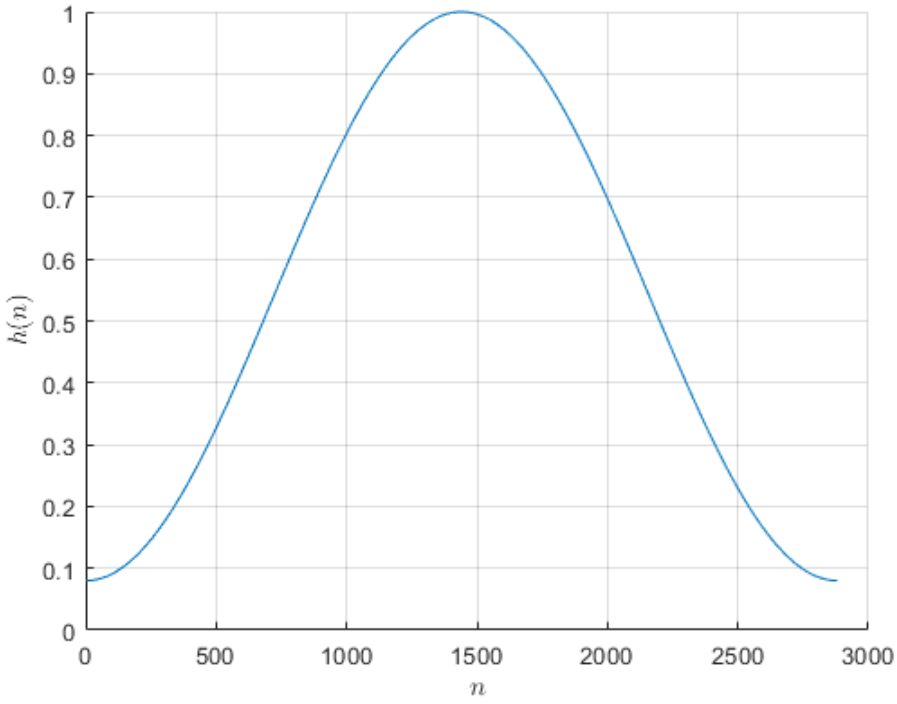
Nazwa klasy	Liczba	Procentowy udział [%]
Wiercenie w dębie	15	24,59
Wyciągnięcie wiertła z dębu	10	16,39
Wiercenie w świerku	18	29,51
Wyciągnięcie wiertła z świerku	18	29,51

4.3. Odrzucenie ciszy na podstawie energii krótkotrwałej

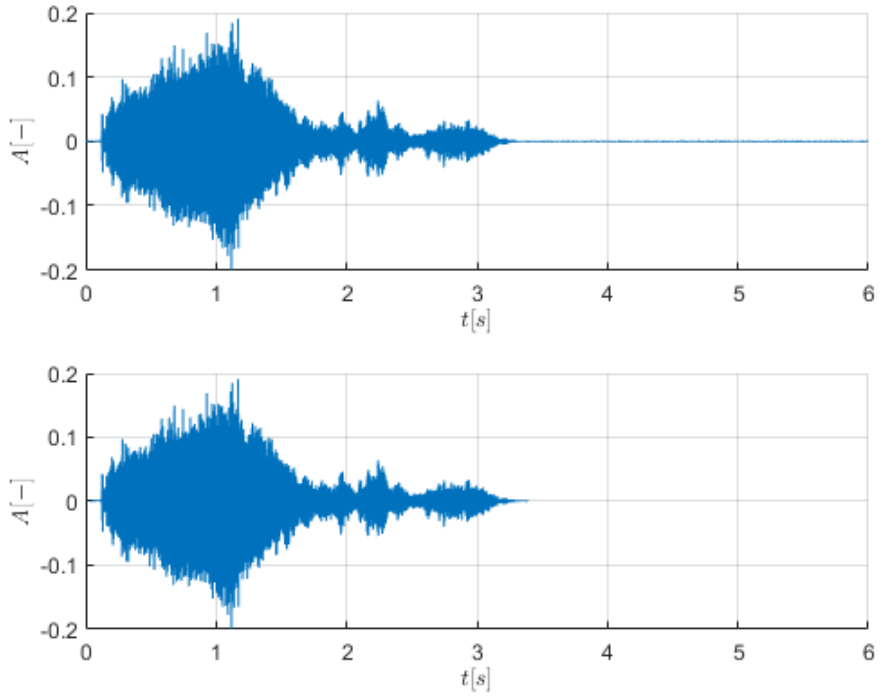
Zdecydowano się na pomiar dźwięku wiertarki w 6-sekundowym przedziale czasowym. Zakres, w którym amplituda dźwięku jest większa od zera wynosi około trzech sekund. Reszta pomiaru to cisza jaką starano się zachować, tak aby nie wprowadzać do danych pomiarowych niepożądanych błędów związanych z odgłosami operatora lub otoczenia. Cisza, która wtedy była potrzebna z punktu widzenia wydobywania cech jest zjawiskiem, które może dodawać zależności związane z różnicami pomiędzy ciszą a dźwiękiem. Z punktu widzenia aplikacji modelu klasyfikującego do zastosowań przemysłowych takie odróżnienie wprowadzałoby skupienie takowego modelu nie na dźwięku wiertarki, ale również jej czasu pracy lub otoczeniu, co byłoby niekorzystne. Koniecznym jest więc pozbycie się części sygnału dźwiękowego, które nie przekraczają danego progu ciszy, definiowanego poprzez progową wartość energii krótkotrwałej. Energię krótkotrwałą obliczano według (ang. Short-Time Energy wzoru 4.1).

$$E = \sum_{n=1}^{S} f(n)^2 \tag{4.1}$$

gdzie E to energia krótkotrwała, S to okno sygnału, n to próbka czasowa, a $f(n)$ to amplituda dźwięku. Sygnał dzielony był na segmenty. Jeden segment miał długość 30 ms. Następnie na tą część sygnału stosowano okno Hamminga (rysunek 4.3). Długość pokrycia następnego segmentu z poprzednim wynosiła 25 ms. Segmentacja oraz stosowanie okien to zabiegi konieczne do skorzystania z wydobywania cech audio [12].



Rysunek 4.3: Okno Hamminga. n - próbka, $h(n)$ - funkcja okna Hamminga.

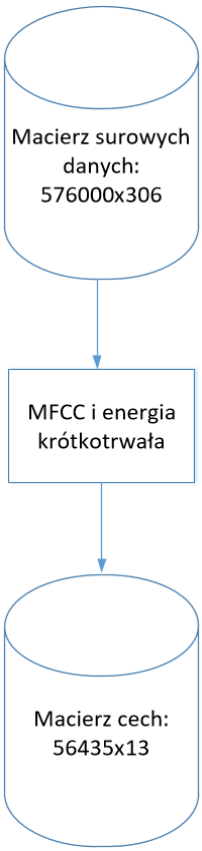


Rysunek 4.4: Usunięcie części sygnału (od ok. 3,5 s do 6 s) nie spełniającej warunku minimalnej energii krótkotrwałej na sygnale w dziedzinie czasu. t -czas, A - amplituda dźwięku (w jednostkach względnych, normalizacja wytłumaczona w rozdziale 2).

Jak widać na rysunku 4.4 amplituda bliska zeru została odrzucona z sygnału. W praktyce energia krótkotrwała była obliczana równolegle cechami sygnałów, które także były wyliczane na podstawie kolejnych segmentów całego zakresu czasu pomiaru. Posiadając rząd wartości cech sygnału oraz energię krótkotrwałą można było zadecydować czy cechy te powinny zostać usunięte. Decyzja ta była dokonywana poprzez porównanie z empirycznie dobraną wartością ($E = 0.05$). Jako że wyniki dokładności walidacji (rozdział 6.5) uznano, że wartość ta została dobrze dobrana. Jest to aspekt projektu do głębszej weryfikacji.

5. WYDOBYCIE CECH SYGNAŁÓW

Cecha sygnału jest wynikiem procesu, który redukuje początkowy zestaw surowych danych do zbiorów, które są łatwiejsze i szybsze do przetworzenia dla sieci neuronowej [13]. Wydobycie tych cech ma jednak sens jedynie w przypadku, gdy uzyskane cechy opisują pierwotny zestaw danych z odpowiednio wysoką dokładnością. W przypadku tego projektu oryginalne dane można porównać poprzez wszystkie próbki w nich zawarte.



Rysunek 5.1: Schemat porównania zestawu danych przed i po wydobyciu cech.

$$N_r = 576000 \cdot 306 = 176256000 \tag{5.1}$$

$$N_f = 56453 \cdot 13 = 733889 \tag{5.2}$$

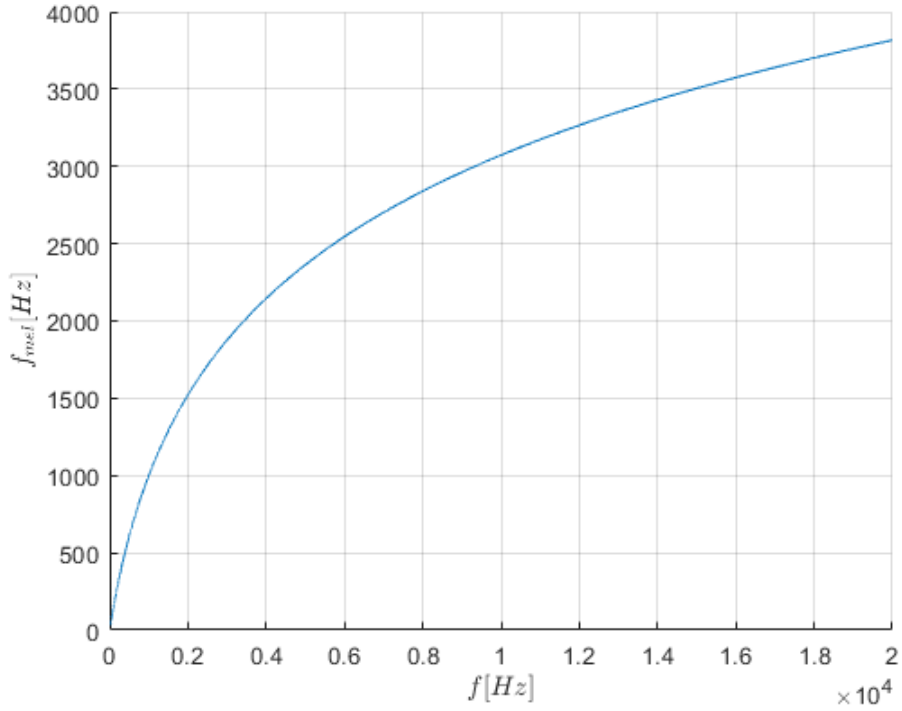
$$\frac{N_r}{N_f} \cdot 100\% \approx 0.42\% \tag{5.3}$$

Wielkości N_r i N_f to kolejno liczba próbek w macierzy przebiegów czasowych i liczba próbek w macierzy cech. Jak widać z wyniku równania 5.3 oraz rysunku 5.1 początkowy zestaw danych zmniejszono ponad dwieście razy. Wydobyte cechy pomogły więc pozbyć się zbędnych danych. Głównie poprzez wydobycie energii krótkotrwałej, która po porównaniu z progiem usuwała cały rząd cech. Następnie po zakończeniu wydobycia ona sama była z macierzy cech usuwana. Matlab posiada 25 dostępnych cech sygnałów do wydobycia. Posiada on więc szeroki wachlarz opcji, w przypadku zmiany podejścia wydobywania cech sygnałów.

MFCC (ang. Mel Frequency Cepstrum Coefficients) to trzynaście współczynników częstotliwościowego cepstrum w skali Mel. Skala Mel jest transformacją logarytmiczną częstotliwości. Transformacja ta ma jednak szczególną funkcję. Przenosi ona liniowy zakres częstotliwości w przestrzeń lepiej rozumianą przez człowieka. Ludzkie ucho potrafi zarejestrować dźwięki z zakresu od 20 Hz do 20 kHz [14], podobnie jak używany mikrofon na stanowisku pomiarowym. Jednakże umiejętność człowieka do rozróżniania częstotliwości na całym zakresie słyszalności jest różna. Przykładowo jest mu łatwo rozróżnić pomiędzy dźwiękami o częstotliwościach 100 Hz i 400 Hz, lecz ta sama różnica częstotliwości dla 12 kHz i 12,3 Hz jest trudniejsza do rozpoznania. Skala Mel pozwala na zniwelowanie tej trudności [15]. Wzór na transformację Mel został dobrany empirycznie. Podawane są różne wzory w literaturze, jednakże w Matlabie stosowane jest równanie 5.4.

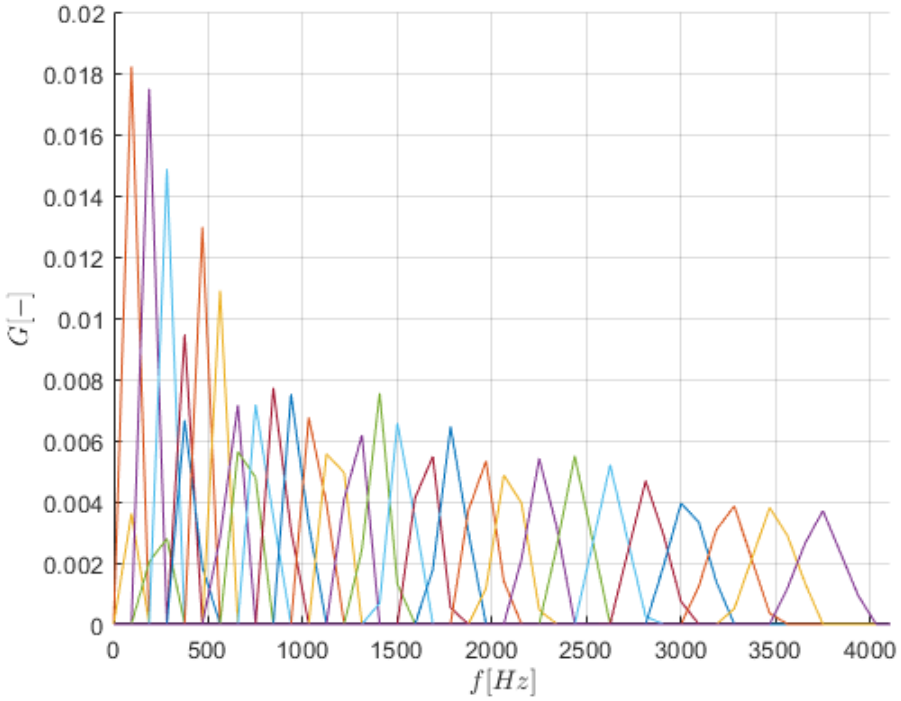
$$f_{Mel} = 2595 \cdot log_{10}(1 + \frac{f}{700}) \tag{5.4}$$

gdzie f_{Mel} to częstotliwość w skali Mel, a f to częstotliwość. Dla zakresu słyszalności funkcja ta przedstawia się tak jak na rysunku 5.2.



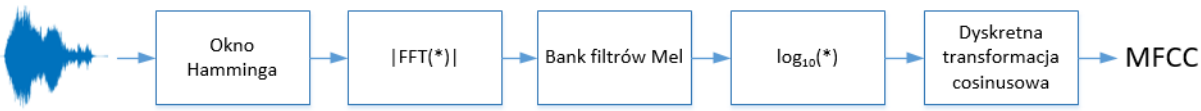
Rysunek 5.2: Skala Mel w zakresie 20 Hz-20 kHz.

W praktyce wykorzystywano bank filtrów Mel, czyli szereg trójkątnych filtrów rozmieszczonych logarytmicznie. Przykładowy bank filtrów można zobaczyć na rysunku 5.3.



Rysunek 5.3: Bank filtrów Mel. G - wzmocnienie, f - częstotliwość.

Posiadając wiedzę o skali Mel można przystąpić do obliczenia MFCC. Podobnie jak dla energii krótkotrwałej sygnał był dzielony na segmenty o długości 30 ms z tą samą długością pokrycia się następnej części (25 ms). Aby z tego segmentu wydobyć MFCC należało zastosować algorytm działania [8] pokazanym na rysunku 5.4.



Rysunek 5.4: Algorytm obliczeniowy MFCC.

Tak samo jak w przypadku energii krótkotrwałej stosowano okno Hamminga (rysunek 4.3). Następnie obliczano FFT czyli szybką transformatę Fouriera. Jest ona wynikiem transformacji sygnału z dziedziny czasu na dziedzinę częstotliwości. Opiera się na dyskretnej transformacji Fouriera (równanie 5.5)

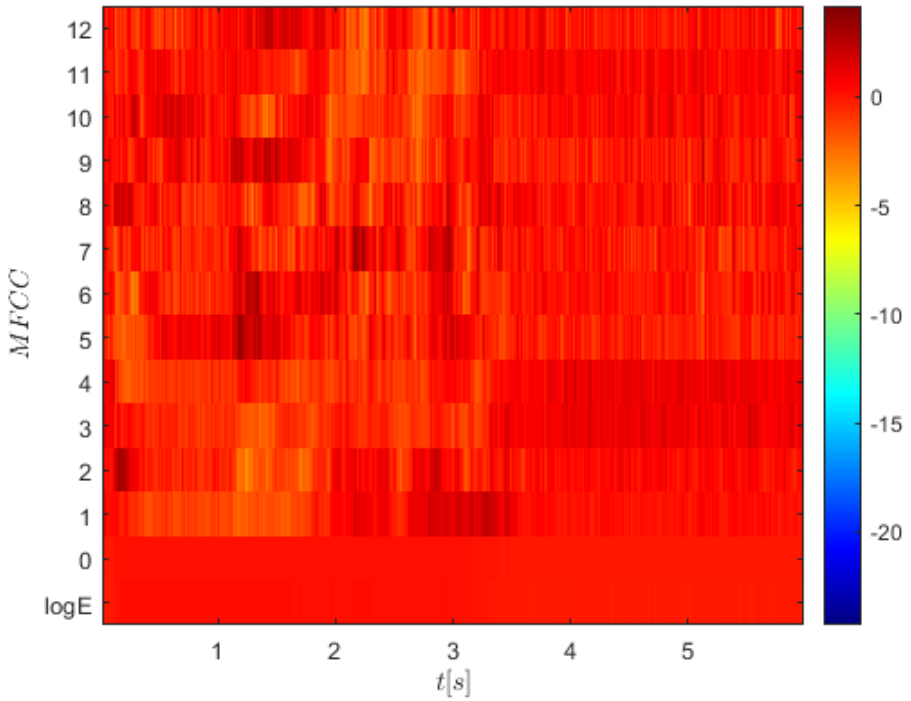
$$F(k) = \sum_{n=1}^{N} f(n) e^{\frac{-j2\pi kn}{N}} \tag{5.5}$$

W równaniu 5.5 $F(k)$ jest dyskretną transformatą Fouriera, N to liczba wszystkich próbek, n to numer próbki sygnału, a k numer składowej częstotliwości. Dyskretna transformacja Fouriera w powyższej formie jest uciążliwa obliczeniowo. Obliczana bezpośrednio potrzebuje ona N^2 kalkulacji [16]. FFT potrzebuje tych kalkulacji $N \cdot log_2(N)$. Przekłada się to na bardzo znaczące skrócenie obliczeń. FFT używa algorytmu dzielącego zbiór N na pary próbek, który stoi za tym przyśpieszeniem.

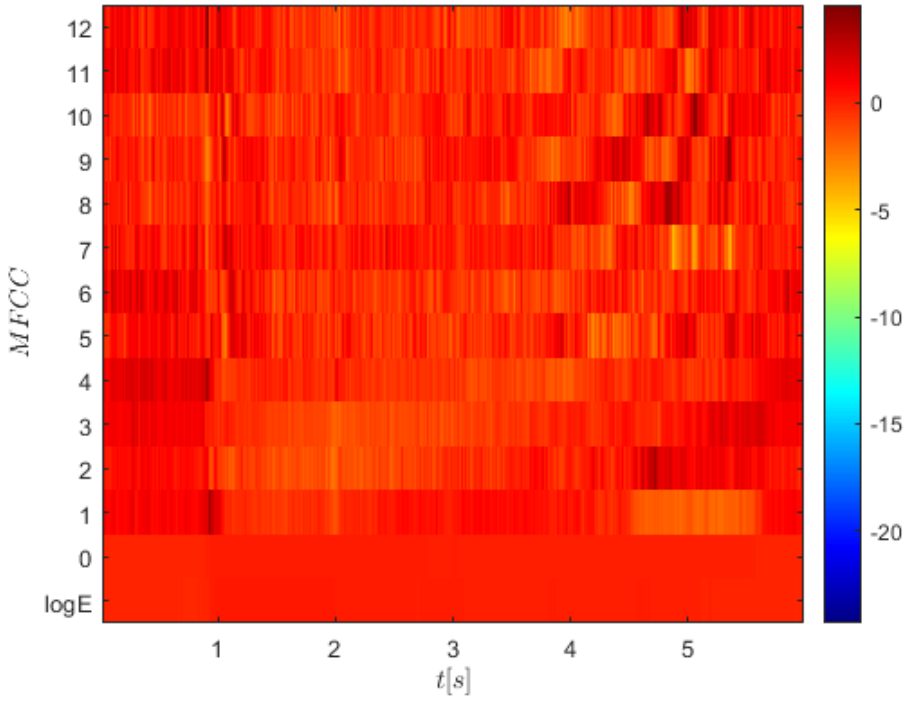
Następnie transformatę przepuszczano przez bank filtrów Mel, aplikując tym samym skalę Mel. Wynik ten następnie logarytmowano. Ostatnim krokiem do uzyskania MFCC było zastosowanie dyskretnej transformacji cosinusowej (DCT) widocznej w równaniu 5.6 [17]

$$B(k) = \sqrt{\frac{2}{N}} \sum_{n=1}^{N} x(n) \frac{1}{\sqrt{1+\delta_{n1}}} cos\left(\frac{\pi}{2N}(2n-1)(k-1)\right) \tag{5.6}$$

gdzie $B(k)$ jest dyskretną transformatą cosinusową , $x(n)$ jest wartością sygnału poddawanego DCT, a δ_{n1} jest deltą Kroneckera. Reszta symboli reprezentuje te same wielkości co w równaniu 5.5. Wynikiem transformacji cosinusowej są MFCC. Wydobyto 13 cech na jeden segment. Przykładowe wyniki wydobytych cech z nagrań znajdują się na rysunkach 5.5 i 5.6



Rysunek 5.5: Przykładowy wykres MFCC dla pomiaru wiercenia w świerku.



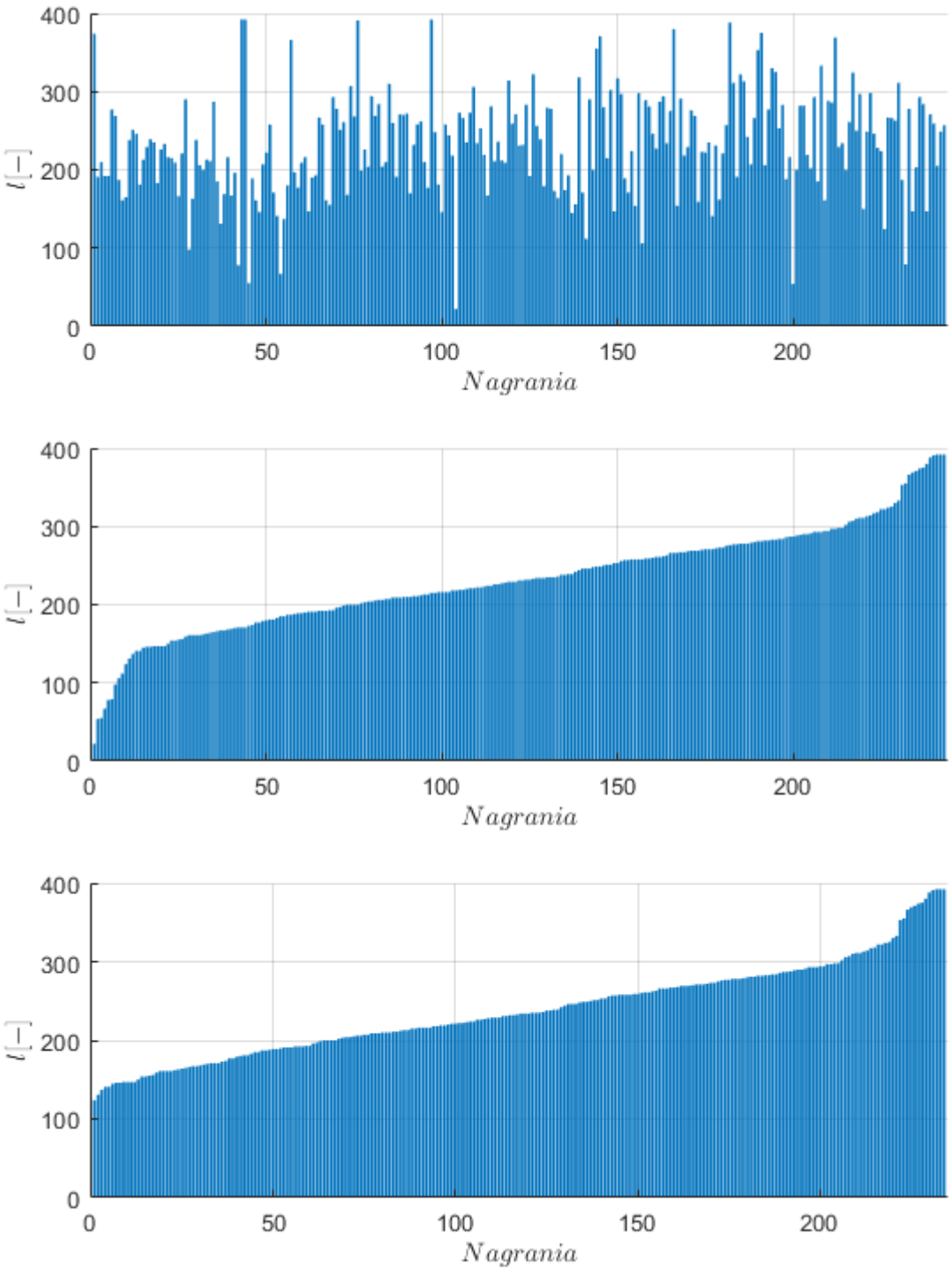
Rysunek 5.6: Przykładowy wykres MFCC dla pomiaru wyciągania wiertła z dębu.

Każde nagranie pomiarowe ma różną ilość wydobytych cech z sygnału. Liczba zestawów trzynastu cech w postaci MFCC jest zwana sekwencją. Zestaw trenujący posiadał 245 nagrań. Sortowano nagrania wg. długości ich sekwencji ze względu na lepszą interpretację danych oraz szybsze uczenie [9]. Podczas uczenia sieci napotykano problem overfittingu, który dokładnie został opisany w podrozdziale 6.4. Jedną z jego przyczyn może być wydobycie cech z nagrań o małej liczbie sekwencji. W takich nagraniach istnieje większa szansa na wydobycie cech z danych wprowadzających błąd w procesie uczenia. Na podstawie średniej długości sekwencji nagrań wyznaczono graniczną długość sekwencji jaką musi posiadać nagranie, aby jego cechy brały udział w procesie uczenia.

$$\bar{l} = \frac{l_1 + l_2 + l_3 + ... + l_{N_t}}{N_t} \tag{5.7}$$

$$l_{gr} = \frac{\bar{l}}{2} \tag{5.8}$$

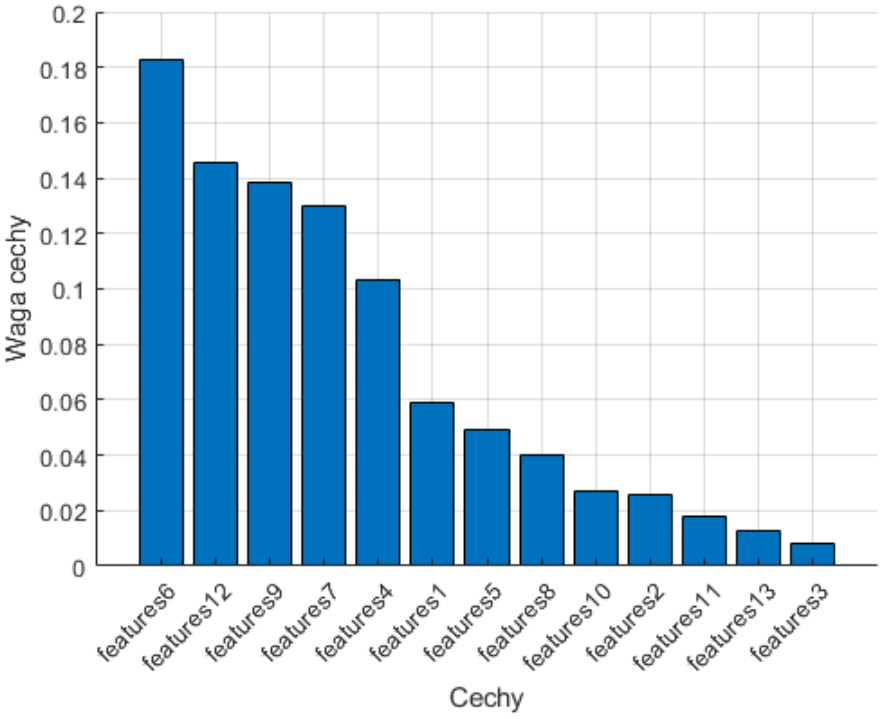
$\bar{l}$ jest średnią długością sekwencji, l długością sekwencji w zestawie trenującym, N_t liczbą wszystkich nagrań w sekwencji, a l_{gr} graniczną długością sekwencji, która jest progiem odrzucenia określonych nagrań. Metoda, gdzie usuwano sekwencje o zbyt małej długości wykorzystała równania 5.7 oraz 5.8. Została ona zastosowana bez podstaw w literaturze. Wymaga się jej dalszej weryfikacji, lecz przyniosła ona pozytywne skutki w postaci zmniejszenia zjawiska overfittingu, które spotęgowane jest zbyt dużą ilością danych wprowadzających zakłócenia (podrozdział 6.4). Dlatego zdecydowano się na jej pozostawienie w skrypcie. Skutki metody usuwania zbyt krótkich sekwencji oraz algorytmu sortującego można zaobserwować na rysunku 5.7.



Rysunek 5.7: Wykresy słupkowe sekwencji. Od góry: przed sortowaniem, po sortowaniu, po odrzuceniu zbyt krótkich sekwencji. l długość sekwencji

5.1. Selekcja, ranking i normalizacja cech

Dalszymi sposobami ulepszenia jakości nauki sieci neuronowej jest selekcja, ranking lub skalowanie cech. W tym przypadku jedne z pośród trzynastu cech w postaci MFCC mogą być bardziej znaczące w perspektywie uczenia sieci neuronowej. Do rankingu cech wykorzystano algorytm MRMR [18](ang. Minimum Redundancy Maximum Relevance). Algorytm ten poszukuje zestawu cech, który jest maksymalnie wzajemnie od siebie różny oraz równocześnie najlepiej reprezentuje odpowiedź, czyli zestaw etykiet testujących. Na podstawie uzyskanego rankingu można także dokonać selekcji cech do uczenia sieci neuronowej. Po zastosowaniu algorytmu otrzymano ranking widoczny na rysunku 5.8.



Rysunek 5.8: Ranking cech na podstawie algorytmu MRMR

Stosowaną metodą skalowania cech był min-max. Jest to normalizacja cech do przedziału [0,1]. Używanie jakiejkolwiek normalizacji jest pożyteczne w przypadku, gdy stosowana sieć neuronowa wymaga określonego zakresu danych wejściowych. Wzór na normalizację min-max podano w równaniu 5.9

$$x_n = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{5.9}$$

gdzie x_n to znormalizowana wartość cechy x to wartość cechy, a x_{min} i x_{max} to kolejno najmniejsza i największa wartość w macierzy cech. Ostatecznie ani ranking cech przy pomocy algorytmu MRMR, ani normalizacja nie były stosowane z uwagi na gorsze wyniki uczenia sieci neuronowej w przypadku ich stosowania. Skrypt implementujący te rozwiązania pozostaje w pobocznej ścieżce w konstruowaniu modelu klasyfikującego. Można to zaobserwować w organizacji plików na rysunku 3.1. Jednakże z uwagi na stworzenie nie tylko dobrze funkcjonującego modelu, ale także środowiska do jego stworzenia w przypadku innego obiektu

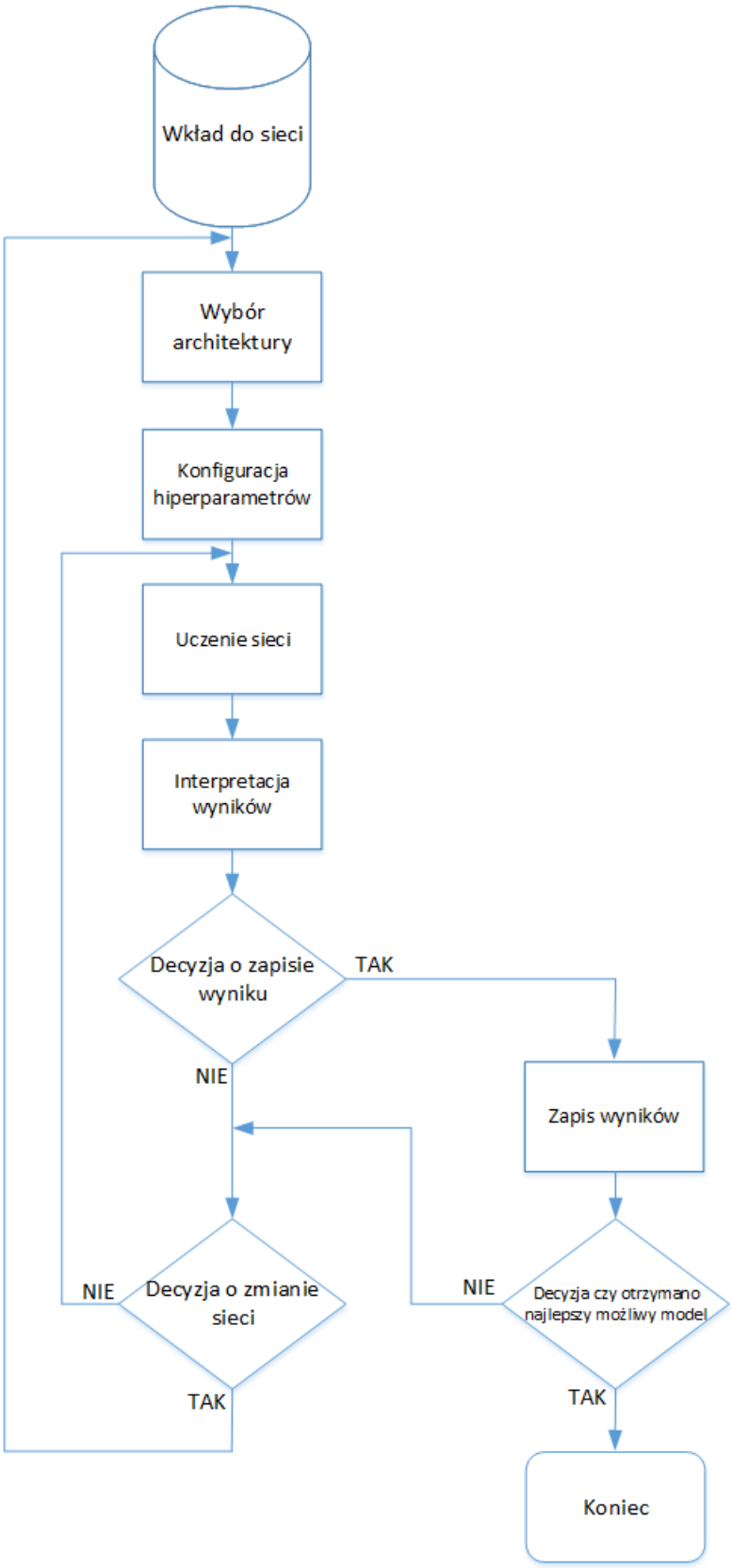
pomiarowego zdecydowano się na pozostawienie skryptu wykonującego powyższe funkcje w projekcie.

6. UCZENIE SZTUCZNEJ SIECI NEURONOWEJ

Rozpoznanie dźwięku i przyporządkowanie mu odpowiedniej etykiety jest zadaniem niealgorytmizowalnym. Rozwiązaniem go jest stworzenie klasyfikatora w oparciu o modelowanie wiedzy. Można osiągnąć taki model poprzez zastosowanie sztucznej sieci neuronowej. Wzorowane są one na podstawie działania biologicznych sieci neuronowych, które zmieniają swoje właściwości w procesie uczenia nowych umiejętności. Sztuczna sieć neuronowa składa się z warstwy wejściowej, wyjściowej i określonej liczby warstw ukrytych. W warstwach tych znajdują się neurony połączone ze sobą w sposób zdefiniowany przez typ warstwy. Do zbudowania klasyfikatora należy zdefiniować architekturę tj. liczbę i typ warstw sieci neuronowej, sposób połączeń oraz jej hiperparametry. Hiperparametry sieci neuronowej są stałymi wartościami lub trybami, nastawianymi przed rozpoczęciem nauki sieci, które definiują sposób uczenia. Należą do nich m.in. liczba neuronów ukrytych, tempo uczenia, rozmiar mini batch. Głębsza analiza stosowanych hiperparametrów oraz architektury sieci neuronowej została umieszczona poniżej w rozdziale.

Budowa modelu predykcyjnego jest procesem, który cechuje się losowością. Oznacza to, że dwie sieci o tej samej architekturze mogą osiągnąć zupełnie różne wyniki. Dzieje się tak z uwagi na fakt, że nie istnieje metoda doboru wag początkowych zapewniająca punkt początkowy problemu treningu sieci. Wagi początkowo dobierane są losowo. Wymagana jest więc w przyjętej metodyce testowanie tej samej sieci aż nie wywnioskuje się, że nie osiągnie ona lepszego wyniku. Schemat działania podczas uczenia sieci pokazano na rysunku 6.1. Mając wkład do sieci, czyli wynik pracy z rozdziałów 2, 3, 4 i 5, można przystąpić do wyboru architektury sieci oraz konfiguracji hiperparametrów. Oba zadania wymagają wkładu ze strony programisty pracującego z projektem. Dla większości zadań wymaga się więc dla nich odpowiedniej wiedzy heurystycznej związanej z sztucznymi sieciami neuronowymi. Większość rozwiązań sprowadza się jednakże do obycia się z tematami związanymi z głębokim uczeniem sieci neuronowych. Dlatego bardzo ważna jest umiejętność interpretacji wyników oraz wyciągniecie poprawnych wniosków co do zmiany sieci. Zapis wyników, a tym samym stworzenie bazy informacji, sukcesywnie nakierowywało na podejmowanie dobrych decyzji, wprowadzania trafnych zmian do sieci.

Wspomniano, że w procesie pokazanym na rysunku 6.1 pierwszym krokiem było zdobycie wkładu do sieci. Nie jest jednak tak, że wszystkie warianty sieci były testowane dla tego samego wkładu. Podczas rozwiązywania problemu overfittingu (podrozdział 6.4) starano się także sprawdzić czy nie można ulepszyć tej części schematu działania. Właśnie dlatego sprawdzano inne sposoby na wydobycie cech. Ta ścieżka nie przyniosła jednakże lepszych efektów niż pierwotna - z cechami MFCC. Wystąpiła jednak jedna pozytywna zmiana wkładu do sieci podczas uczenia sieci. Było nim odrzucenie części nagrań o zbyt niskiej długości sekwencji. Można więc stwierdzić, że przedstawiony schemat działania jest prawidłowy, lecz czasami należy cofnąć się i przeanalizować rozwiązania zastosowane w poprzednich etapach projektu.



Rysunek 6.1: Schemat działania w celu budowy klasyfikatora

6.1. Architektura sieci neuronowej

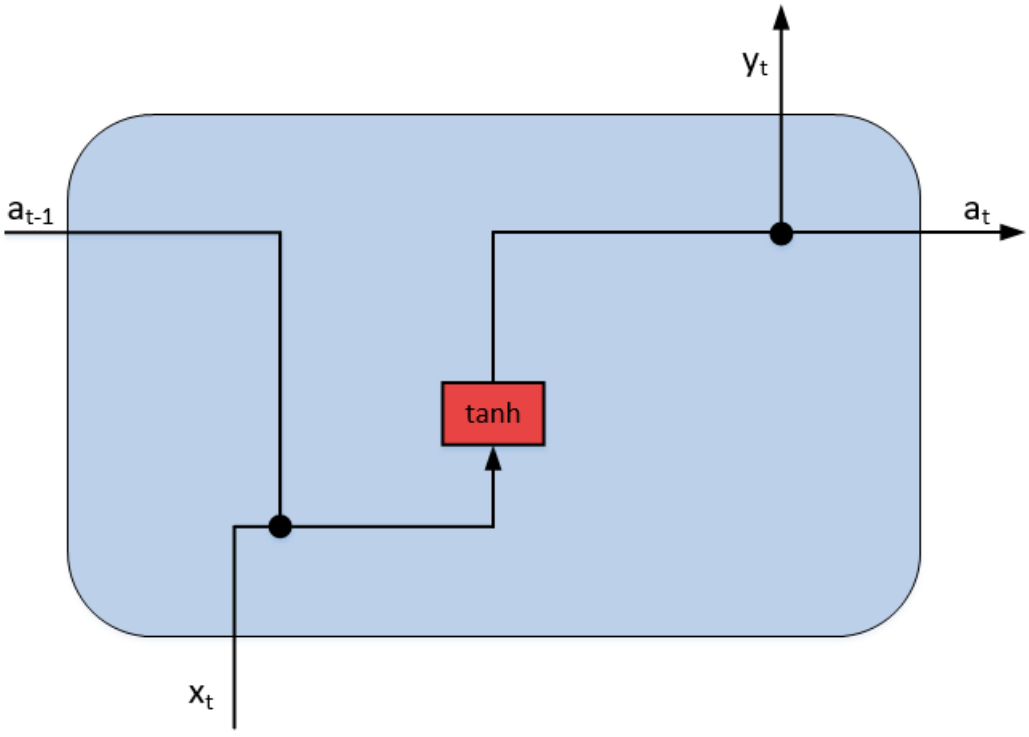
Architektura sieci praktycznie się nie zmieniała na przestrzeni projektu. Decyzję taką podejmowano z uwagi na rosnącą dokładność walidacji dla odpowiedniej zmiany hiperparametrów, a także z powodu gorszych wyników w przypadku jej zmiany. Zastosowano następujące warstwy:

6.1.1. Sekwencyjna warstwa wejściowa

Warstwa wejściowa sieci neuronowej zawsze posiada liczbę neuronów równą liczbie elementów wektora uczącego. W tym przypadku było to trzynaście z uwagi na liczbę kolumn w macierzy cech. Jako, że wydobyte cechy opierały się na sekwencjach, to i taką należało zastosować warstwę wejściową. Warstwa wejściowa przekazuje dalej informacje do warstwy ukrytej.

6.1.2. Warstwa ukryta BiLSTM

Typ sieci neuronowej definiuje się poprzez typ warstwy ukrytej. W tym projekcie stosowano BiLSTM (ang. Bidirectional Long-Short Term Memory). Jest ona typem sieci rekurencyjnej. Operują one na danych czasowych. Różnią się np. od jednokierunkowych sieci tym, że przy pomocy pamięci uwzględniają poprzednie stany w celu wpływu na ówczesne wejście i wyjście. Wynik sieci rekurencyjnej zależy od poprzednich wartości w sekwencji jaka jest przetwarzana.

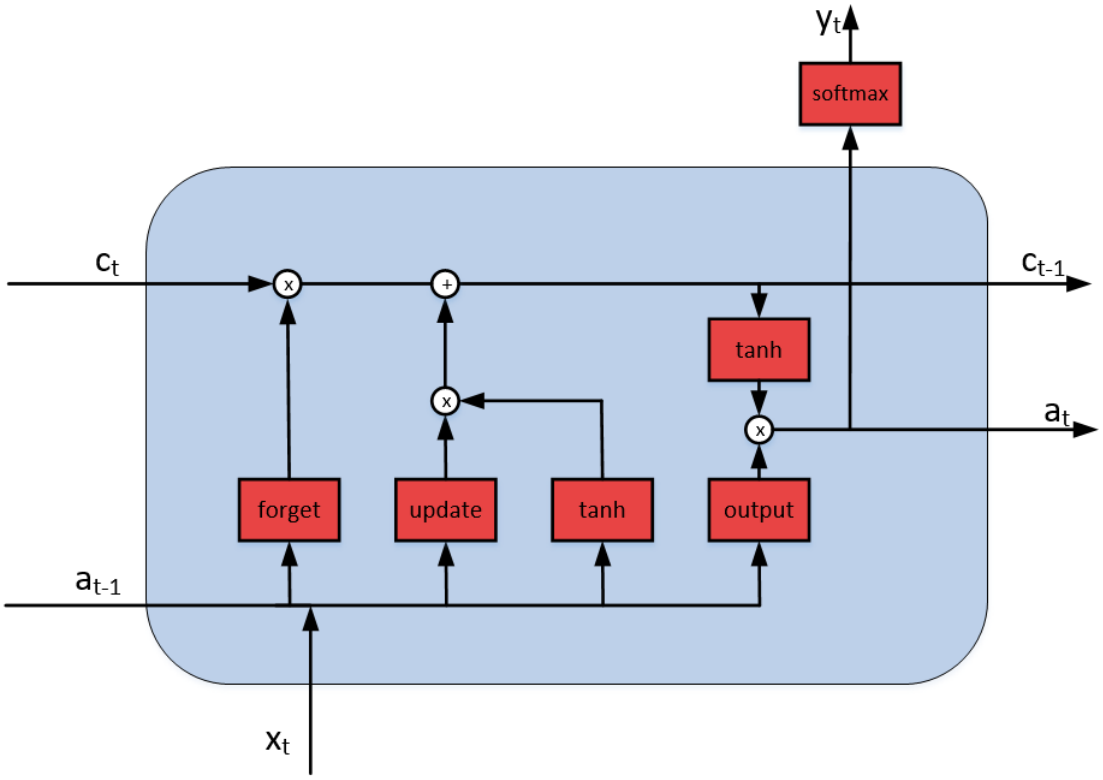


Rysunek 6.2: Schemat komórki pamięci sieci rekurencyjnej. a_{t-1} - wartość ukryta poprzedniego stanu, a_t - wartość ukryta, x_t - wartość wejściowa, y_t - wartość wyjściowa, $tanh$ - funkcja aktywacji w postaci tangensa hiperbolicznego.

Może się jednak okazać, że takowa sieć jest podatna na zjawiska zanikających i eksplodujących gradientów [19]. Gradientem nazywa się wektor, który aktualizuje wagi sieci na podstawie funkcji celu. Funkcja celu jest wyznacznikiem jak dobrze nauczona jest sieć. Celem każdej sieci jest minimalizacja funkcji celu. Zanikający gradient to sytuacja kiedy w złożonych sieciach, o dużej liczbie warstw, zmiana gradientu jest na tyle mała, że nie ma praktycznie żadnego wpływu na wagi w początkowych warstwach. Wynikiem jest zerowy postęp uczenia. Eksplodujący gradient to zjawisko gdy w sieci zmiana gradientu jest zbyt duża przez co zmiany wag w późniejszych warstwach są bardzo duże. Powoduje to niestabilność procesu uczenia.

Ograniczeniem obu zjawisk jest zastosowanie sieci LSTM [20]. Potrafi dobrze korelować próbki daleko oddalone od siebie, nawet na początku i końcu sekwencji. Struktura LSTM posiada bramki posiadające następujące funkcje:

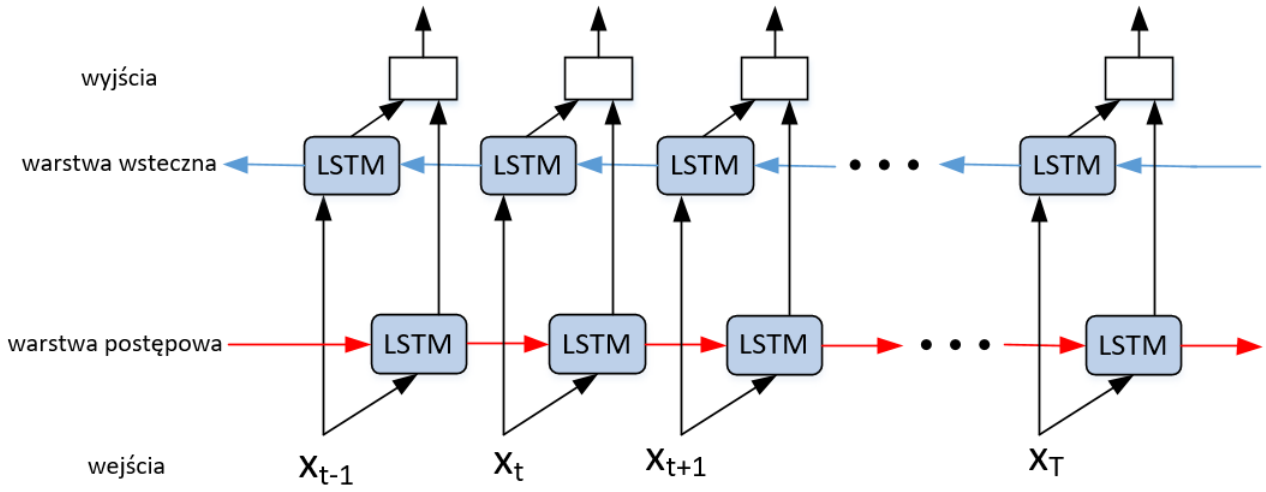
1. bramka forget - decyduje ile informacji komórka pamięci otrzyma z komórki poprzedniego kroku,
2. bramka update - ustala czy komórka pamięci będzie aktualizowana oraz kontroluje ile informacji komórka otrzyma od potencjalnie nowej komórki pamięci,
3. bramka output - kontroluje wartość następnego wyjściowego stanu ukrytego a_t.



Rysunek 6.3: Schemat komórki pamięci sieci LSTM, c_t - teraźniejsza wartość w komórce pamięci, c_{t-1} - teraźniejsza wartość w komórce pamięci z poprzedniej sekwencji, $softmax$ - funkcja aktywacji softmax, reszta wielkości ma takie same znaczenie jak na rysunku 6.2.

Dodatkowym atutem takowej sieci byłaby umiejętność wpływania na wynik poprzez użycie nie tylko danych, które już znalazły się w danej komórce pamięci, lecz takich, które dopiero

w niej się znajdą w przyszłości. Umiejętność tą posiada sieć BiLSTM czyli dwukierunkowa sieć LSTM [20]. Dodaje ona jedną warstwę LSTM, która zmienia kierunek przepływu danych. Zabieg ten jest widoczny na rysunku 6.4



Rysunek 6.4: Schemat sieci BiLSTM, x_t wartość wejściowa o numerze sekwencji t,x_T wartość wejściowa z ostatniej sekwencji

Ilość neuronów w tej warstwie była regularnie zmieniana, w procesie pokazanym na rysunku 6.1.

6.1.3. Warstwa w pełni połączona

Pełni rolę warstwy wyjściowej. Liczba neuronów w tej warstwie jest równa liczbie rozłącznych klas (dwie). Warstwa ta jest jednokierunkowa. Neurony w tej warstwie posiadają połączenie z każdym neuronem warstwy ukrytej. Charakteryzuje się równaniami 6.1 i 6.2.

$$u(x) = \sum_{j=1}^{N_w} W_{ij} x_j + b_i \tag{6.1}$$

$$y(x) = f_{softmax}(u_i(x)) \tag{6.2}$$

gdzie $u(x)$ to wynik sumy wag i biasu, W_{ij} współczynniki wagowe, x_j wartość wektorów wejściowych, b_i - bias, czyli próg aktywacji oraz $f_{softmax}$ funkcja aktywacji softmax.

6.1.4. Warstwa softmax

Warstwa softmax transformuje wektor wejściowy danych w wektor o elementach, których suma równa jest jeden. Funkcja softmax normalizuje dane, które przetwarza. Wzór na funkcję softmax jest widoczny w poniższym równaniu:

$$\sigma(x)_i = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}} \tag{6.3}$$

gdzie $\sigma(x)_i$ to wynik funkcji softmax, x_i - elementy wekotra wejściowego, K - liczba klas modelu,

6.1.5. Warstwa klasyfikacji

Warstwa klasyfikacji, która kalkuluje funkcję celu w sposób cross-entropy. Jest ona typową częścią sieci klasyfikujących zaraz po warstwie softmax.

$$E = \frac{-1}{N} \sum_{n=1}^{N} \sum_{i=1}^{K} w_i t_{ni} ln(y_{ni}) \tag{6.4}$$

Funkcja celu - E, N -liczba próbek, K - liczba klas, w_i - waga klasy i, t_{ni} - wskaźnik, że n-ta próbka należy do i-tej klasy, y_{ni} - wynik funkcji softmax.

6.2. Wielkości opisujące proces uczenia

Dokładność jest wielkością, która mówi o procentowym udziale dobrze nadanych etykiet w stosunku do liczby wszystkich nagrań w danym zestawie danych. Dzieli ona dokładność dla zestawu treningowego oraz dokładność walidacji czyli dla zestawu testującego. Dokładność treningowa jest zaznaczona na rysunkach 6.6 - 6.21 kolorem niebieskim, a walidacyjna czarnym. Głównym wyznacznikiem potrafiącym uogólniać dane jest dokładność walidacyjna, ponieważ sieć nie miała styczności z danymi z zestawu testującego.

$$y_{acc} = \frac{n_p}{N_n} \cdot 100\% \tag{6.5}$$

W równaniu 6.5 y_{acc} jest dokładnością walidacyjną, a n_p i N_n to kolejno liczba prawidłowo sklasyfikowanych nagrań oraz liczba nagrań.
Kolejną wielkością jest funkcja celu. Przy postępach uczenia jest ona funkcją opadającą, która opisuje sumę błędów jaką popełnił klasyfikator w porównaniu z każdą próbką w zestawie trenującym lub testującym. Funkcja celu liczona jest na podstawie równania 6.4. Dla zestawu trenującego zaznaczony jest na rysunkach 6.6 - 6.21 ona na pomarańczowo, a dla zestawu testującego na czarno. W większości przypadków podczas uczenia sieci gdy funkcja celu opada dokładność się zwiększa. Nie ma jednak korelacji matematycznej pomiędzy tymi wielkościami, więc nie można mówić jakieś ogólnej zasadzie zachowania względem siebie [21].

6.3. Hiperparametry sieci

W tym podrozdziale zamieszczono opis hiperparametrów czyli stałymi wartości lub tryby, nastawianymi przed rozpoczęciem nauki sieci, które definiują opcje treningu. To poprzez ich nastawy udało otrzymać modele podane w 6.1. Hiperparametrami sieci były:

- Solver - Technika aktualizacji sieci. Używano techniki typu Adam (skrót od Adaptive Moment Estimation), która działa na podstawie równań 6.6 - 6.8.

$$\theta_{l+1} = \theta_l - \frac{\alpha m_l}{\sqrt{v_l} + \epsilon} \tag{6.6}$$

$$m_l = \beta_1 m_{l-1} + (1 - \beta_1) \nabla E(\theta_l) \tag{6.7}$$

$$v_l = \beta_2 v_{l-1} + (1 - \beta_2) \Big[\nabla E(\theta_l) \Big]^2 \tag{6.8}$$

gdzie l to numer iteracji, θ to gradient, α współczynnik tempa uczenia, m_l i v_l to kolejno wyniki równań 6.7 i 6.8, ϵ to epsilon opisany poniżej, β_1 to współczynnik rozkładu gradientu, β_2 to kwadratowy współczynnik rozkładu gradientu, $\nabla E(\theta_l)$ to gradient funkcji celu.

- Współczynnik rozkładu gradientu (β_1) - domyślnie równy 0,9. Nie zmieniano w trakcie pracy.
- Kwadratowy współczynnik rozkładu gradientu (β_2) - domyślnie równy 0,999. Nie zmieniano w trakcie pracy.
- epsilon (ϵ)
- Próg gradientu - granica, dla której gradient jest ograniczany. Przeciwdziała zjawisku eksplodującemu gradientu (niestabilności sieci) widzanym na rysunku 6.6. Domyślna wartość to nieskończoność. Nie zmieniano w trakcie pracy.
- Rozmiar mini batch - zwykle ustawiony na 32. Liczba próbek, która na raz jest przetwarzana przez sieć. Mniejsze jej wartości mogą uzyskać lepszy wynik uczenia kosztem jego tempa. Przetworzenie przez sieć jednego mini batcha oznacza jedną iterację pracy sieci. Przetworzenie przez sieć całego zestawu danych kończy jedną epokę.
- Maksymalna liczba epok - po przekroczeniu przez proces uczenia tej granicy zakańczany jest on automatycznie. Zwykle ustawione na 20000. W otrzymanych modelach uczenie nie zajmowało więcej niż 11000 epok. Uczenie zatrzymywano ręcznie, więc wartość 20000 zapewniało brak wystąpienia sytuacji, gdzie uczenie kończy się automatycznie przed osiągnięciem potencjalnie najlepszego wyniku.
- Sieć wyjściowa - wynik zwracany na początek sieci. Domyślnie nastawiona jako wynik ostatniej iteracji treningowej. Nie zmieniano w trakcie pracy.
- Tasowanie - randomizacja danych. Nastawione na "nigdy". Nie zmieniano w trakcie pracy.
- Częstotliwość walidacji - częstotliwość, z którą obliczane są dokładność i funkcja celu walidacji. Domyślnie ustawiona na 50 iteracji.
- Cierpliwość walidacji - dopuszczalna liczba wzrostów funkcji celu walidacji przed automatycznym zakończeniem uczenia sieci. Podczas stosowania równa 100.
- Tryb tempa uczenia - określa czy tempo uczenia ma być stałe czy zmienne. Domyślnie nastawione jest stałe tempo uczenia.
- Początkowy współczynnik tempa uczenia (α) - decyduje jak szybko powinna uczyć się sieć.
- Okres obniżania tempa uczenia - co ile epok obniża się tempo uczenia.
- Współczynnik obniżenia tempa uczenia - wymnażany przez współczynnik tempa uczenia, aby je zwolnić.
- Liczba neuronów w warstwie ukrytej - ma duży wpływ na złożoność sieci, jej tempo oraz jakość uczenia. Zadowalające wyniki osiągano w przedziale 40-90 neuronów.

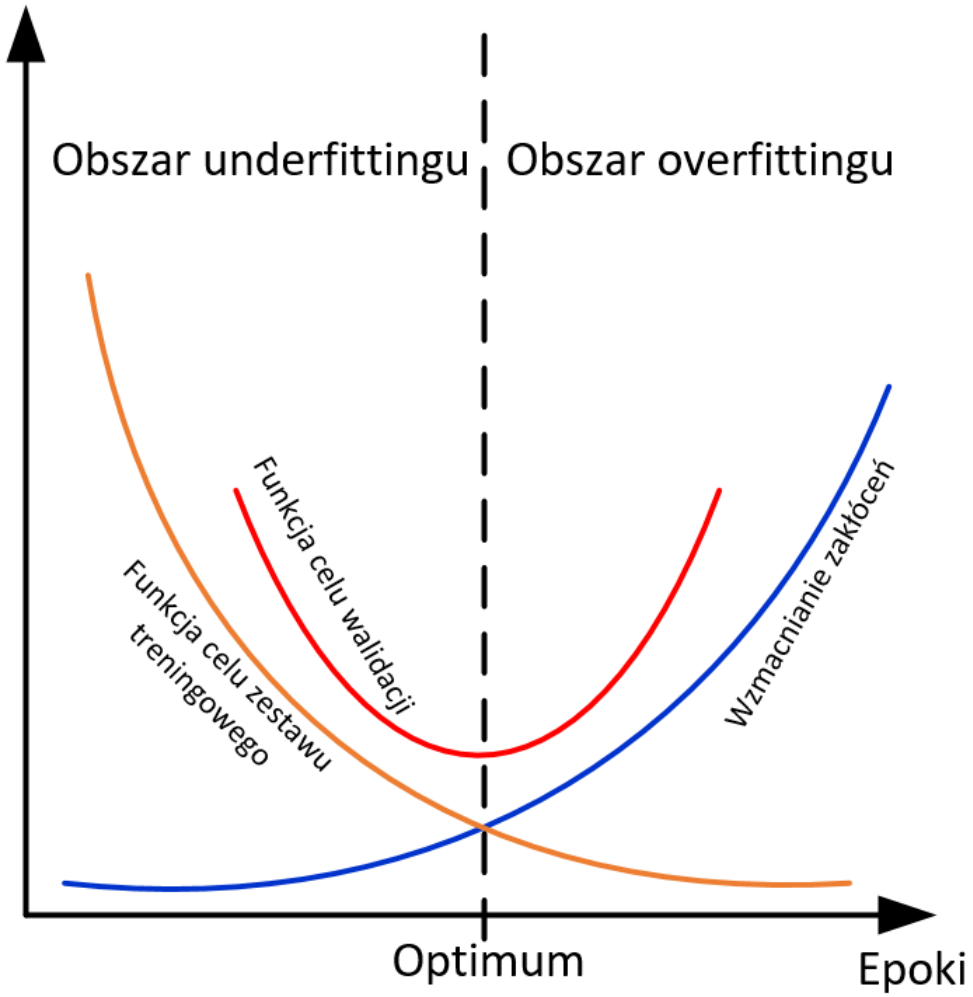
- Regularyzacja L2 (λ) - współczynnik składnika kary w funkcji celu (równianie 6.9), który pozytywnie wpływa na overfitting. Nastawiony domyślnie na 0,0001.

$$E_R(\theta) = E(\theta) + \lambda\Omega(w) \tag{6.9}$$

gdzie $E_R(\theta)$ to funkcja celu po regularyzacji, $E(\theta)$ to funkcja celu, λ to współczynnik regularyzacji, $\Omega(w)$ to funkcja regularyzacji.

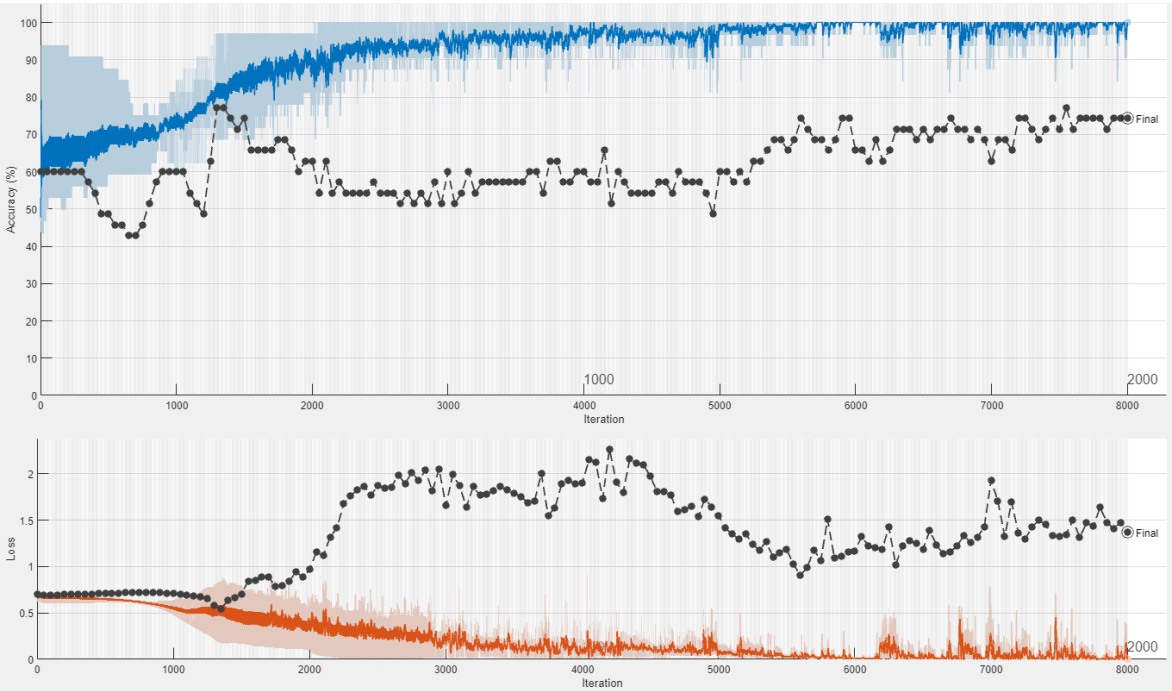
6.4. Problem overfittingu

Overfitting jest najczęściej spotykanym problem podczas budowania modelu klasyfikującego. Można go zdefiniować jako obszar w procesie uczenia kiedy funkcja celu walidacji zaczyna rosnąć, gdy wzmocnienie zakłóceń przez sieć zaczyna być większe od spadku funkcji celu na zestawie trenującym. Sieć w obszarze overfittingu jest przetrenowana i traci zdolność do generalizacji danych. Wynikiem jest także spadek dokładności walidacyjnej i niestabilność uczenia. Dąży się więc do znalezienia minimum globalnego funkcji celu walidacji zwanej optimum. W obszarze przed optimum sieć jest niedouczona (tzw. obszar underfittingu) i może zwiększyć swoją dokładność walidacyjną oraz zmniejszyć funkcję celu walidacji. Uproszczony wykres procesu uczenia został pokazany na rysunku 6.5.



Rysunek 6.5: Uproszczony wykres zjawiska overfittingu.

W praktyce zjawisko overfittingu może wyglądać tak jak na rysunku 6.6. Sieć osiągnęła minimalną funkcję celu w okolicach 1300 iteracji. Następnie funkcja celu wzrastała, a sieć zaczęła być niestabilna. Nie osiągnięto dobrej dokładności walidacji. Główną przyczyną słabego wyniku uczenia jest zbyt złożona budowa sieci. Można wywnioskować, że należy zmniejszyć liczbę neuronów w warstwie ukrytej, a następnie po zinterpretowaniu nowych wyników podjąć dalsze decyzje. Overfitting był już w modelach o lepiej dobranych parametrach minimalizowany. Skutkowało to lepszymi dokładnościami walidacyjnymi.



Rysunek 6.6: Wykresy dokładności i funkcji celu obu zestawów: trenującego i testującego. Oznaczenia: niebieski - dokładność zestawu trenującego, czarny górny - dokładność walidacji, pomarańczowy - funkcja celu zestawu trenującego, czarny dolny- funkcja celu walidacji. Sieć o 300 neuronach w warstwie ukrytej, rozmiarze mini batch 32, po 2000 epokach uczenia, ostatecznej dokładności 74%, początkowym współczynniku uczenia 0,00001 oraz regularyzacji L2 0,0001.

Przyczynami overfittingu są [22]:

- model jest zbyt złożony - posiada zbyt dużą liczbę warstw ukrytych lub neuronów w tych warstwach,
- zestaw danych trenujących zawiera zbyt dużą ilość zakłóceń, danych niepotrzebnych,
- model uczy się zbyt długo,
- zestaw treningowy jest zbyt mały by pokryć wariację wszystkich możliwych zjawisk w danych wejściowych.

Określenie najważniejszej przyczyny overfittingu wymaga zmiany parametrów lub hiperparamterów modelu i sprawdzania wyników. Tak samo jak dla przyjętej metodyki z rysunku 6.1.

6.5. Wyniki

Aby zapewnić dobrą organizację wśród zbiorów wytrenowanych modeli przyporządkowano im nazwy według wzoru widzianego na rysunku 6.7, gdzie H to liczba neuronów w warstwie ukrytej, B to rozmiar mini batch, Ep to epoka, w której zatrzymano uczenie, A to dokładność walidacyjna, I to początkowe tempo uczenia, λ to współczynnik funkcji regularyzacji L2, a f to format pliku (.mat. lub .png). Nazwa pliku zawiera informacje o końcowym wyniku dokładności walidacyjnej oraz o wartości hiperparametrów, które były najczęściej zmieniane podczas ich kolejnych konfiguracji. Reszta hiperparametrów wymieniona w podrozdziale 6.3 nie była zmieniana, więc decydowano się na domyślne ustawienia. Sieci neuronowe testowane w tym rozdziale posiadały zawsze ten sam układ warstw, więc nie zamieszczono informacji o nim w nazwach plików. Składał się on z sekwencyjnej warstwy wejściowej, pojedynczej warsty ukrytej BiLSTM, warstwy w pełni połączonej, warstwy softmax oraz warstwy klasyfikacji. Dokładny ich opis zamieszczono w podrozdziale 6.1.

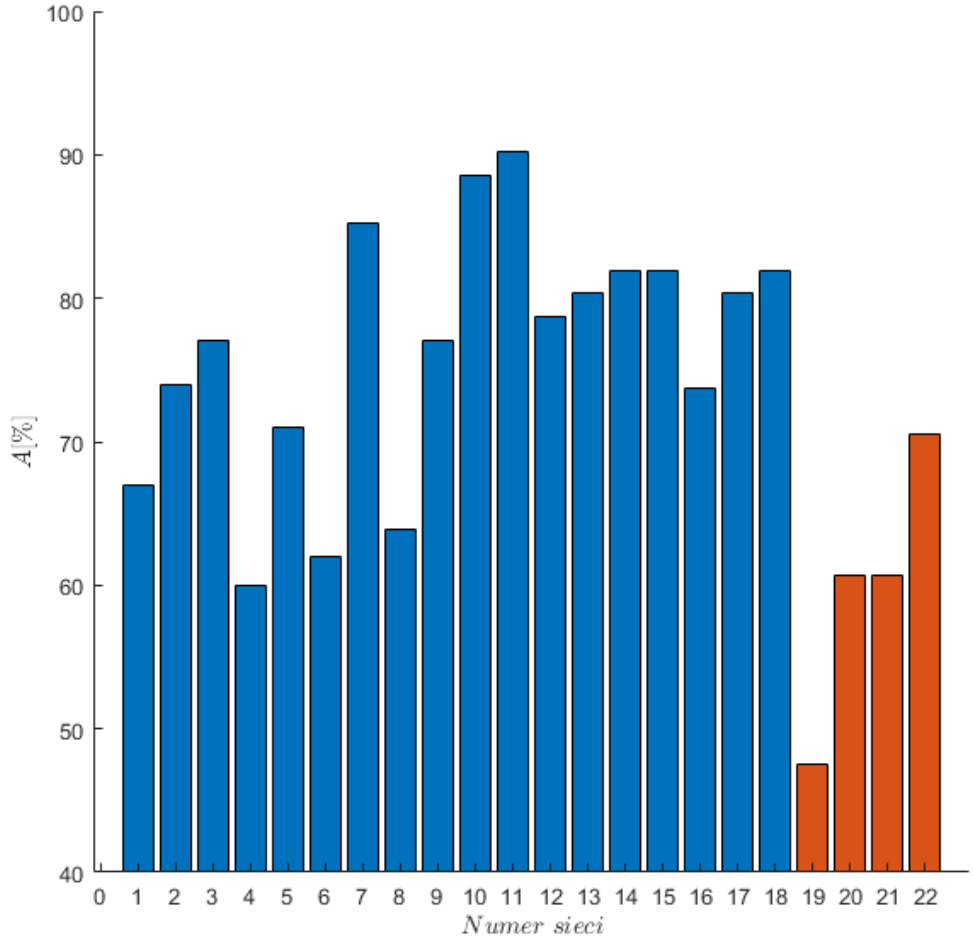
net_h_[H]__b_[B]__e_[Ep]__acc_[A]__il_[I]_L2R_[λ].[f]

Rysunek 6.7: Klucz nazewnictwa wyników

Nagłówki kolumn w tabeli 6.1 zostały nazwane zgodnie z powyższym kluczem. Zawartość jej zebrano na podstawie nazw plików z wynikami.

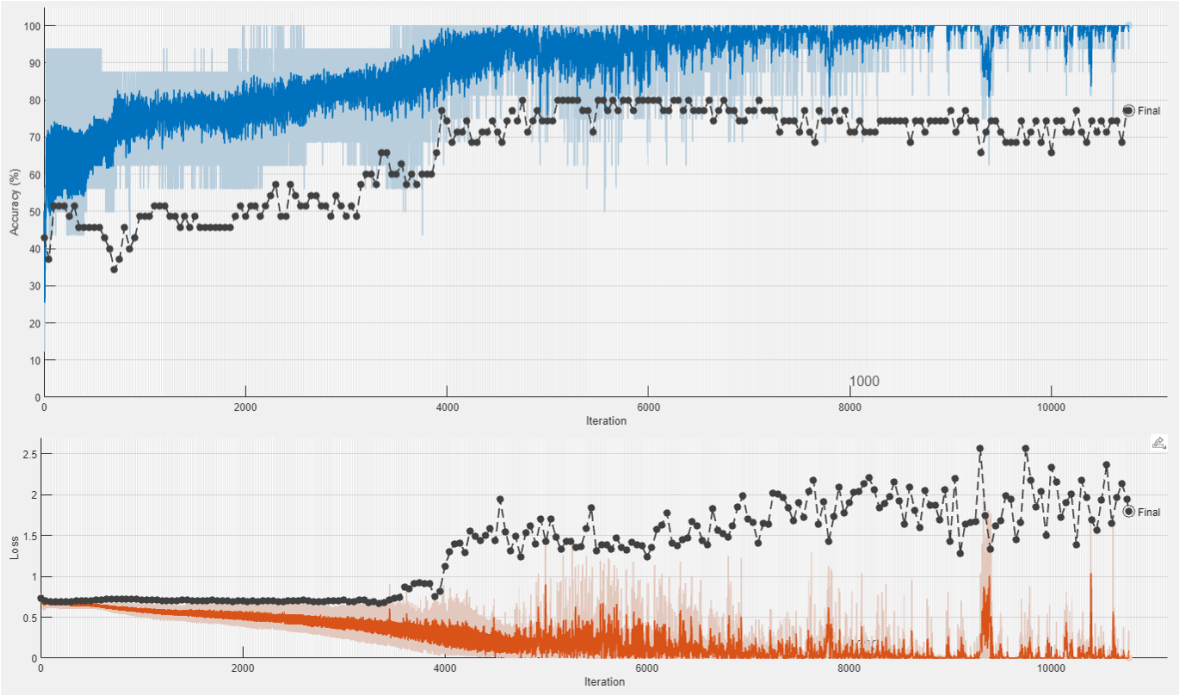
Tabela 6.1: Tabela zbiorcza wyników.

Lp.	**Klasy**	**H**	**B**	**Ep**	**A [%]**	**I** $\cdot 10^{-5}$	$\lambda \cdot 10^{-5}$
1	2	100	32	350	67,00	3	10
2	2	300	32	2000	74,00	1	10
3	2	400	32	1350	77,00	1	10
4	2	40	32	3270	60,00	1	10
5	2	50	32	3270	71,00	1	10
6	2	80	32	1800	62,00	1	10
7	2	55	32	3370	85,25	0,4	72
8	2	57	32	3590	63,93	0,4	72
9	2	57	32	2820	77,05	0,4	72
10	2	57	32	15330	88,52	0,4	72
11	2	57	32	2530	90,16	0,4	72
12	2	60	32	2360	78,69	0,4	70
13	2	60	32	1900	80,32	0,5	70
14	2	60	32	2150	81,96	0,4	70
15	2	60	32	2400	81,96	0,5	70
16	2	70	32	4480	73,77	0,5	10
17	2	70	32	1560	80,32	0,5	10
18	2	75	32	1800	81,96	0,4	10
19	4	50	32	10940	47,54	0,4	90
20	4	50	32	6280	60,66	0,4	78
21	4	50	32	2970	60,66	0,4	73
22	4	50	32	5120	70,49	0,4	73



Rysunek 6.8: Wykres słupkowy dokładności walidacyjnej klasyfikatorów dwóch klas (kolor niebieski) i czterech klas (kolor pomarańczowy) z tabeli 6.1. Numer sieci to liczba porządkowa w tabeli 6.1, a A to dokładność walidacyjna.

Punktem wyjściowym podczas poszukiwań klasyfikatora o dobrej dokładności walidacyjnej była sieć o architekturze pokazanej w podrozdziale 6.1 z domyślnymi nastawami hiperparametrów. Dobrą jakość rozumie się przez zwiększenie dokładności walidacji powyżej 85%. Taka wartość oznacza fakt, że model w dobrym stopniu uogólniał dane. Zrobił on wymierny postęp w stosunku do początkowej wartości dokładności walidacji, która wynosiła ok. 50 %, co wynika z losowej klasyfikacji nagrań. Najpoważniejszym problemem w trakcie procesu treningowego było szybkie przeuczenie sieci w stosunku do osiągnięcia dokładności dla zestawu treningowego powyżej 95%. Widać to dobrze na rysunku 6.6, gdzie występuje zjawisko overfittingu. Podobnie działo się na rysunku 6.9. Walidacyjne funkcje cel w obu tych przypadkach cechowały się niskim spadkiem w stosunku do początkowej wartości tj. zerowej iteracji. Uczenie tych sieci po osiągnięciu minimum stawało się niestabilne, co powoduje brak wzrostu dokładności walidacyjnej. Należy więc zatrzymać proces uczenia w punkcie minimum walidacyjnej funkcji celu.



Rysunek 6.9: Wykresy dokładności i funkcji celu obu zestawów: trenującego i testującego. Oznaczenia: niebieski - dokładność zestawu trenującego, czarny górny - dokładność walidacji, pomarańczowy - funkcja celu zestawu trenującego, czarny dolny- funkcja celu walidacji. Sieć o 400 neuronach w warstwie ukrytej, rozmiarze mini batch 32, po 1350 epokach uczenia, ostatecznej dokładności 77%, początkowym współczynniku uczenia 0,00001 oraz regularyzacji L2 0,0001.

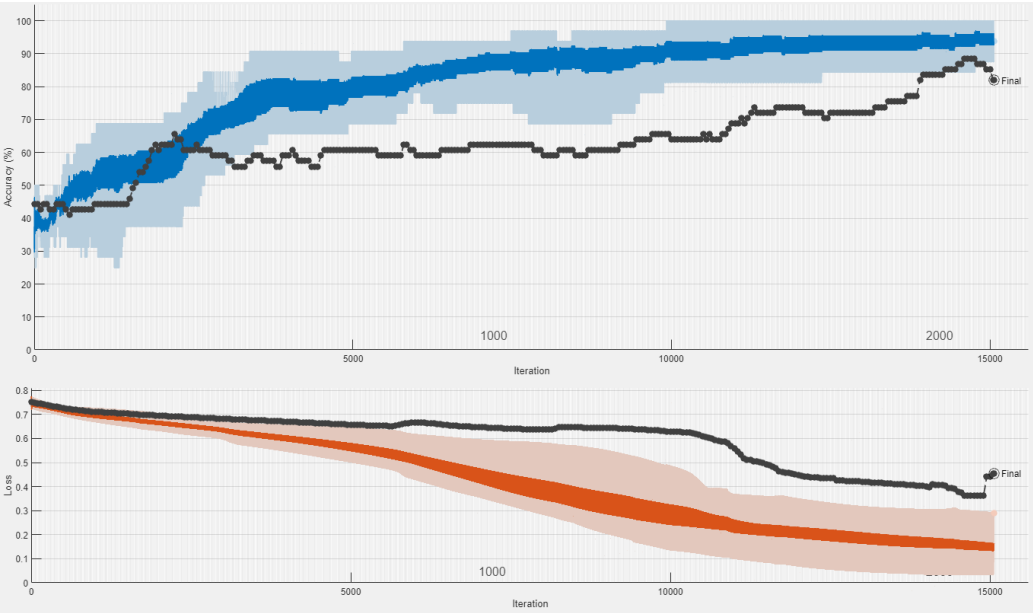
Z uwagi na praktycznie zerowy spadek walidacyjnej funkcji celu z rysunków 6.6 oraz 6.9 w obszarze przed jej dynamicznym wzrostem i późniejszą niestabilnością stwierdzono, że sieć jest zbyt złożona i podatna na szybkie przeuczenie. Należało zatem zmniejszyć liczbę neuronów w warstwie ukrytej. Z początkowych modeli wywnioskowano więc, że należy zakończyć uczenie sieci po przekorczeniu minimum walidacyjnej funkcji celu oraz zmniejszyć liczbę neuronów w warstwie ukrytej.
Istnieją trzy sposoby na zatrzymanie procesu uczenia. Są to:

- zatrzymanie poprzez osiągnięcie wartości epoki maksymalnej,
- zatrzymanie automatyczne polegające na monitorowaniu monotoniczności walidacyjnej funkcji celu,
- zatrzymanie ręczne.

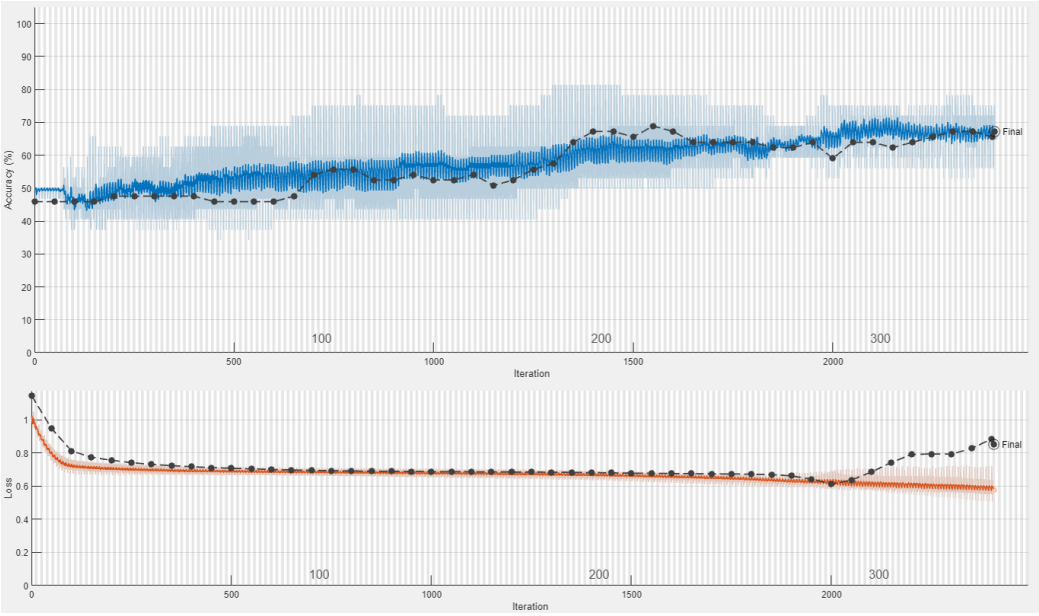
Zatrzymanie poprzez osiągnięcie maksymalnej epoki miało miejsce gdy epoka procesu uczenia zrównała się z hiperparametrem maksymalnej epoki nastawianym przed procesem uczenia. Wartość hiperparametru wpisywana jest ręcznie. Nie istnieje wzór, który mógłby podać kiedy walidacyjna funkcja celu danej sieci osiągnie minimum. Nie można określić jaką wartość epoki maksymalnej należy przyjąć w zależności od pozostałych hiperparametrów. Opieranie się więc na tej metodzie zatrzymywania w poszukiwaniu minimum jest więc nieprawidłowe. Nastawiano więc bardzo dużą wartość epoki maksymalnej w stosunku do przyjętego tempa uczenia. Dla wartości tempa uczenia pokazanego w tabeli 6.1 wartość maksymalnej epoki wynosiła 20000.

W tym przedziale walidacyjne funkcje celu osiągały swoje minima, a proces uczenia zatrzymywany był w inny sposób. Zatrzymanie poprzez osiągnięcie maksymalnej epoki jest więc nadrzędnym warunkiem zatrzymania występującym zawsze, nawet podczas wykorzystywania innych metod zatrzymania.

Automatyczne zatrzymanie sieci oprócz osiągnięcia epoki maksymalnej mogło nastąpić poprzez określenie wymiernej wartości cierpliwości walidacji. Wspomniano już, że gwałtowny wzrost walidacyjnej funkcji celu oznacza przeuczenie sieci. Przykładem są wykresy z rysunków 6.6 i 6.9. Można więc monitorować monotoniczność tej funkcji i na tej podstawie automatycznej zakończyć proces uczenia. W praktyce zliczano przypadki, gdzie wartość walidacyjnej funkcji walidacji była większa od jej poprzedniej wartości. Gdy liczba tych wzrostów zrównała się z wartością cierpliwości walidacji proces uczenia kończył się. Dobrą stroną takiego podejścia jest brak konieczności stałej obserwacji procesu uczenia. Metoda ta jednakże nie osiąga zwykle zamierzonego zatrzymania tuż po osiągnięciu minimum walidacyjnej funkcji celu. Występuje podobny problem jak w przypadku zatrzymania przy pomocy maksymalnej epoki - hiperparametr (w tym przypadku cierpliwość walidacji) nastawiany jest ręcznie. Ponownie nie ma klarownych wytycznych jaką wartość podać. Jak widać na rysunku 6.10 dla ok. 6000 i 8000 iteracji funkcja celu walidacji wzrasta, lecz z osiąga minimum dopiero dla ok. 12500 iteracji. Oznacza to, że w praktyce ta funkcja sporadycznie wzrasta przed osiągnięciem minimum globalnego. Wzrosty te są jednak naliczane. Prowadziło to do zbyt szybkiego, automatycznego zatrzymania procesu uczenia. Dobrym przykładem zbyt szybkiego zatrzymania jest proces uczenia z rysunku 6.11. Można chwilowe wzrosty funkcji celu walidacji wziąć pod uwagę i zwiększyć wartość cierpliwości walidacji. W przypadku gdy funkcja celu walidacji w obszarze niedouczenia maleje na całym zakresie przed osiągnięciem minimum globalnego licznik wzrostów jest zerowy. Po osiągnięciu minimum i rozpoczęciu wzrostu funkcji celu walidacji licznik ten będzie potrzebował większej liczby iteracji do zrównania się z wartością cierpliwości walidacji. Prowadzi to bardzo powolnej reakcji mechanizmu zatrzymania, gdy nauczanie sieci wejdzie w obszar overfittingu. W efekcie proces uczenia wygląda podobnie jak na rysunkach 6.6 i 6.9. Automatyczne zatrzymanie sieci poprzez monitorowanie monotoniczności funkcji celu walidacji nie zakańczało regularnie procesów uczenia w punkcie minimum lub punkcie mu bliskiemu. Podjęto, więc decyzję o jej niestosowaniu.

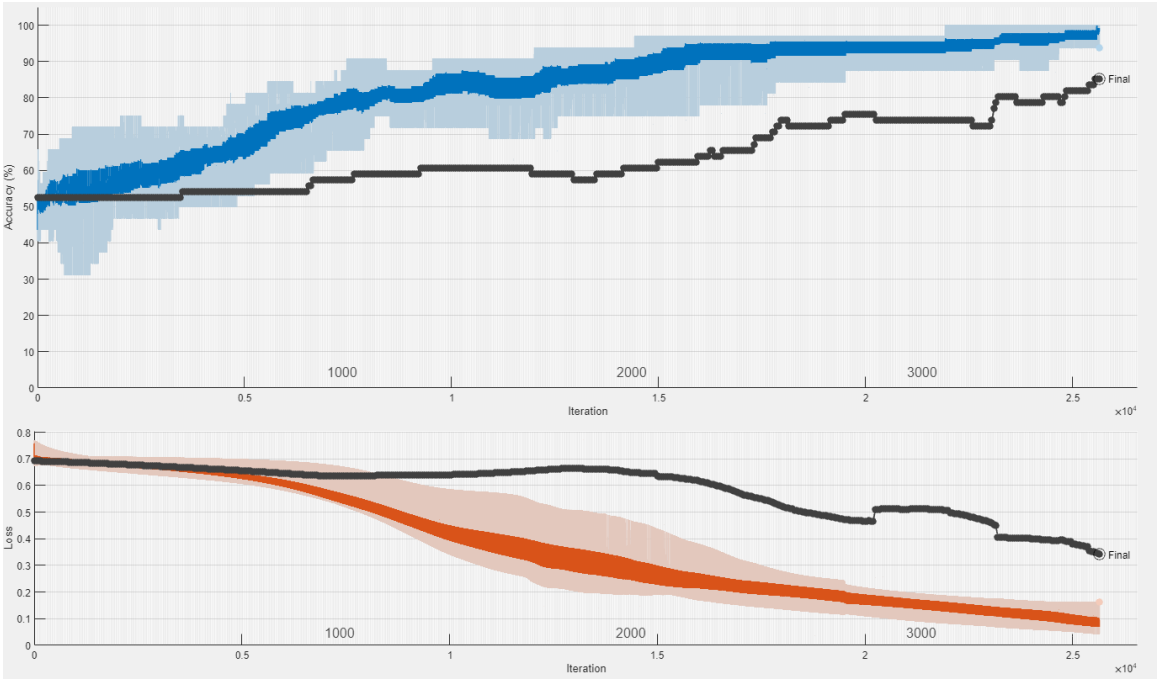


Rysunek 6.10: Wykresy dokładności i funkcji celu obu zestawów: trenującego i testującego. Oznaczenia: niebieski - dokładność zestawu trenującego, czarny górny - dokładność walidacji, pomarańczowy - funkcja celu zestawu trenującego, czarny dolny- funkcja celu walidacji. Sieć o 60 neuronach w warstwie ukrytej, rozmiarze mini batch 32, po 2400 epokach uczenia, ostatecznej dokładności 81,96%, początkowym współczynniku uczenia 0,000005 oraz regularyzacji L2 0,0007.



Rysunek 6.11: Wykresy dokładności i funkcji celu obu zestawów: trenującego i testującego. Oznaczenia: niebieski - dokładność zestawu trenującego, czarny górny - dokładność walidacji, pomarańczowy - funkcja celu zestawu trenującego, czarny dolny- funkcja celu walidacji. Sieć o 100 neuronach w warstwie ukrytej, rozmiarze mini batch 32, po 350 epokach uczenia, ostatecznej dokładności 67%, początkowym współczynniku uczenia 0,00003 oraz regularyzacji L2 0,0001.

Metoda zatrzymywania ręcznego wymagała stałej obserwacji procesu uczenia. Kończono proces według własnej oceny czy model znalazł swoje optimum tak jak pokazano to na rysunku 6.5. Metoda ta wymaga więc rozwinięcia intuicji kiedy funkcja celu walidacji jest w jakim obszarze uczenia. Wyrobienie intuicji polegało na zebranym doświadczeniu podczas obserwacji kolejnych procesów uczenia modeli. Opierano się na różnicy pomiędzy początkową i aktualną wartością funkcji celu walidacji. Dodatkowo monitorowano funkcję celu zestawu trenującego. Na rysunkach 6.12, 6.13 i 6.14 pokazano przypadki ręcznego zatrzymywania.

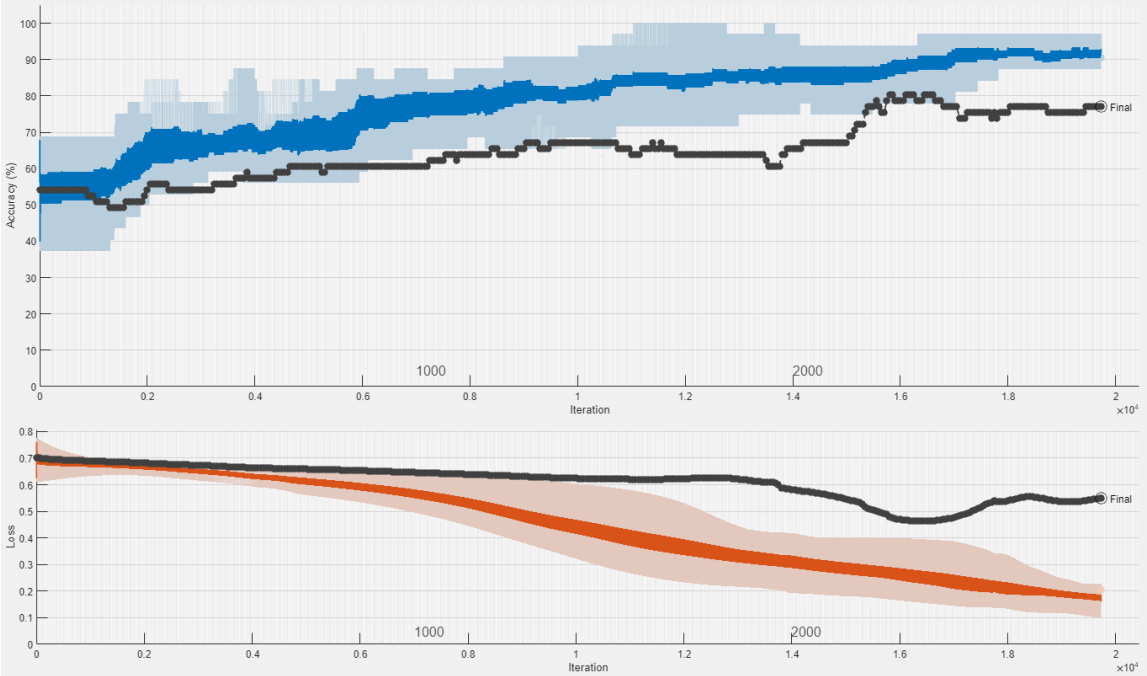


Rysunek 6.12: Wykresy dokładności i funkcji celu obu zestawów: trenującego i testującego. Oznaczenia: niebieski - dokładność zestawu trenującego, czarny górny - dokładność walidacji, pomarańczowy - funkcja celu zestawu trenującego, czarny dolny- funkcja celu walidacji. Sieć o 55 neuronach w warstwie ukrytej, rozmiarze mini batch 32, po 3370 epokach uczenia, ostatecznej dokładności 85,25%, początkowym współczynniku uczenia 0,000004 oraz regularyzacji L2 0,00072.

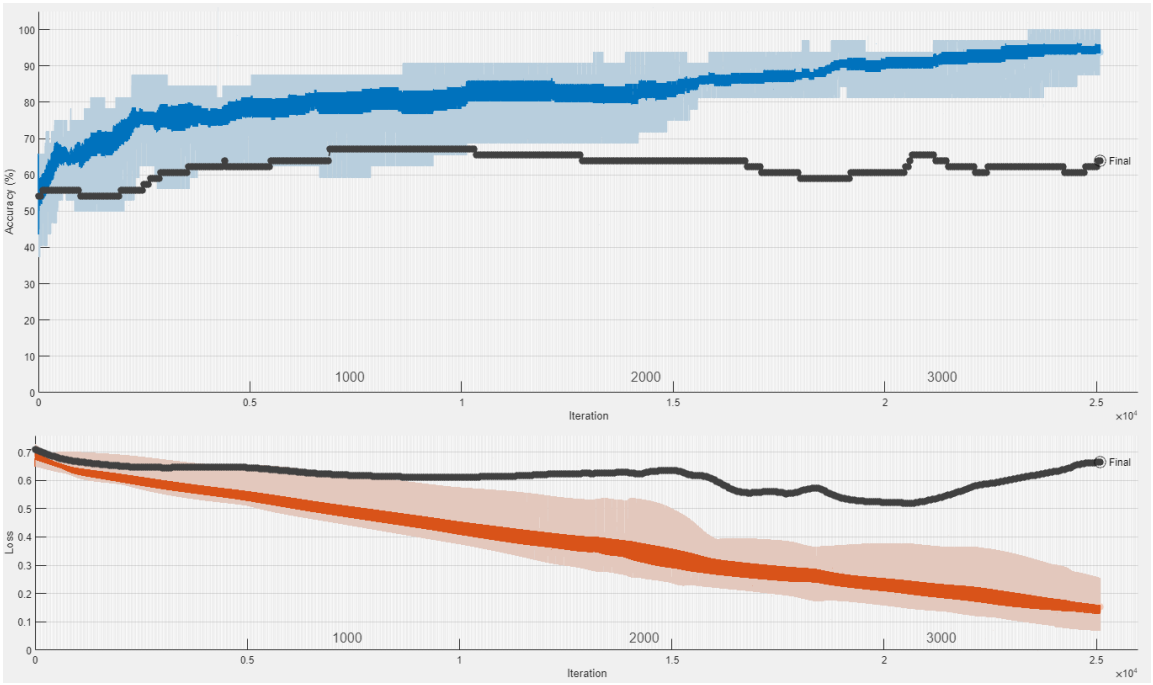
Funkcja celu zestawu treningowego dostarcza informacji, czy funkcja celu walidacji osiągnęła swoje minimum globalne. Sieć aktualizuje się na podstawie tej funkcji. Jeżeli jest ona bliska zeru, to zmiany w sieci nie zachodzą lub są bardzo małe. Przy wartości funkcji zestawu testującego bliskiej zeru oraz rosnącej funkcji celu walidacji należy zakończyć uczenie sieci ponieważ funkcja celu walidacji osiągnęła już swoje minimum. W takiej sytuacji dalsze uczenie oznacza przeuczanie sieci, co prowadzi do zmniejszenia dokładności walidacyjnej, co można zaobserwować na rysunku 6.10, gdzie po osiągnięciu minimum spadła ona z ok. 88 % do 81,96%. Na rysunku 6.12 chciano uniknąć podobnej sytuacji, więc zatrzymano uczenie. Nastąpiło ono jednak zbyt szybko, ponieważ, funkcja celu walidacji wciąż malała. Wypracowano więc następujące reguły co do ręcznego zatrzymywania:

- jeżeli funkcja celu walidacji maleje nie należy zatrzymywać uczenia,
- jeżeli funkcja celu walidacji rośnie, a funkcja celu zestawu trenującego jest bliska zeru należy zatrzymać sieć,

Tymi regułami kierowano się przy zatrzymywaniu procesów uczenia na rysunkach 6.13 i 6.14.



Rysunek 6.13: Wykresy dokładności i funkcji celu obu zestawów: trenującego i testującego. Oznaczenia: niebieski - dokładność zestawu trenującego, czarny górny - dokładność walidacji, pomarańczowy - funkcja celu zestawu trenującego, czarny dolny- funkcja celu walidacji. Sieć o 57 neuronach w warstwie ukrytej, rozmiarze mini batch 32, po 2820 epokach uczenia, ostatecznej dokładności 77,05%, początkowym współczynniku uczenia 0,000004 oraz regularyzacji L2 0,00072.



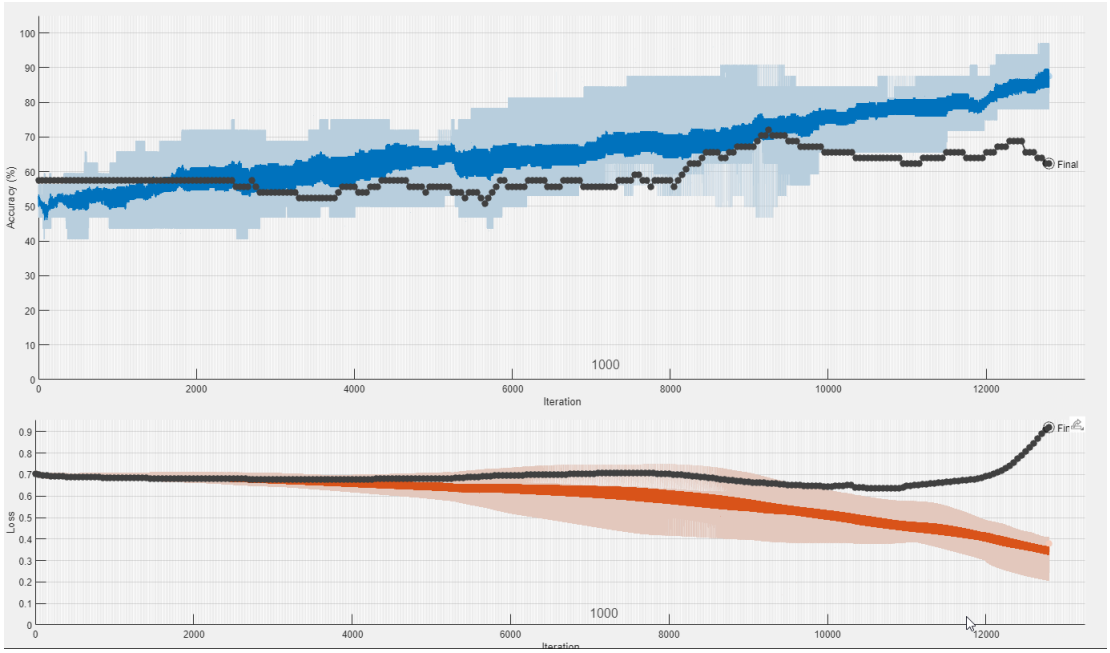
Rysunek 6.14: Wykresy dokładności i funkcji celu obu zestawów: trenującego i testującego. Oznaczenia: niebieski - dokładność zestawu trenującego, czarny górny - dokładność walidacji, pomarańczowy - funkcja celu zestawu trenującego, czarny dolny- funkcja celu walidacji. Sieć o 57 neuronach w warstwie ukrytej, rozmiarze mini batch 32, po 2820 epokach uczenia, ostatecznej dokładności 77,05%, początkowym współczynniku uczenia 0,000004 oraz regularyzacji L2 0,00072.

Na rysunku 6.13 według obranych reguł nieprawidłowo zatrzymano proces uczenia. Funkcja celu zestawu trenującego miała wartość w momencie zatrzymania ok. 0,2, co oznacza, że sieć mogła się zaktualizować w korzystny sposób, tj. lepiej uogólniać dane. Oznaczałoby to spadek funkcji celu walidacji i możliwy wzrost dokładności walidacyjnej. Nie występował także stały wzrost funkcji celu walidacji. Proces uczenia zatrzymano więc zbyt szybko.
Odmienną sytuację można było zaobserwować na rysunku 6.14. Funkcja celu zestawu treningowego miała mniejszą wartość niż na rysunku 6.13 w momencie zatrzymania, a funkcja celu walidacji cechowała się wzrostem od ok. 400 epok. W momencie zatrzymania wartość funkcji celu walidacji była bardzo zbliżona do początkowej przez ten wzrost, co oznacza dalekie przekroczenie punktu optimum. Nie zatrzymano procesu uczenia wcześniej z powodu słabego postępu dokładności walidacji w stosunku do wartości początkowej. Biorąc pod uwagę te czynniki można stwierdzić, że podjęto właściwą decyzję co do zatrzymania procesu uczenia w tamtym momencie. Z procesu uczenia z rysunku 6.14 wywnioskowano, że sieć ta wymaga ponownej próby uczenia i ewentualnej zmiany hiperparametrów.

Zatrzymywanie ręczne w porównaniu do pozostałych metod jest najlepszym podejściem, ponieważ po wytrenowaniu odpowiedniej ilości modeli rozwija się umiejętność kończenia uczenia w momencie bliskim minimum globalnego funkcji celu walidacji.

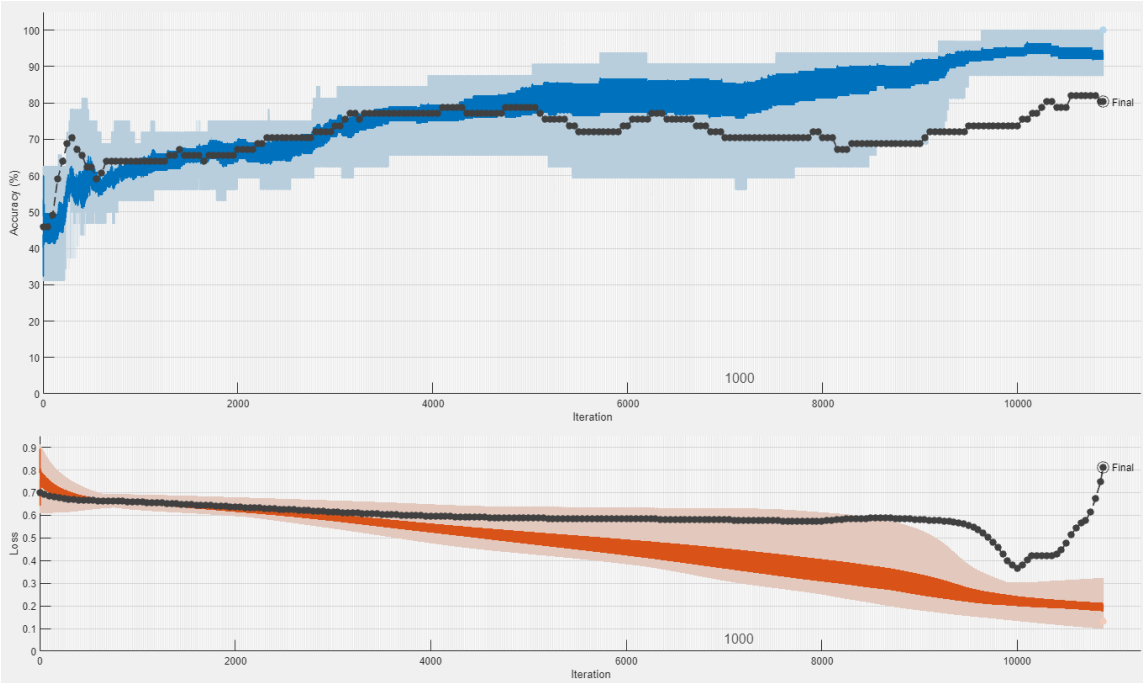
Analizując rysunki 6.6 i 6.9, gdzie znajdowało się kolejno 300 i 400 neuronów w warstwie ukrytej można wywnioskować, że należy zmienić hiperprametry tych sieci. Trzonem uzyskanych postępu w obszarze dokładności walidacyjnej z rysunku 6.8 począwszy od 47,54% a skończywszy na 90,16 % była regularna zmiana konfiguracji hiperparametrów sieci. W procesie ich konfiguracji (rysunek 6.1) głównie skupiano się na tych, które mogłyby pomóc w zminima-

lizowaniu zjawiska overfittingu. Jedną z przyczyn jest zbyt duża złożoność sieci (podrozdział 6.4), więc nastąpił duży nacisk na dobranie odpowiedniej liczby neuronów w warstwie ukrytej. Ze wspomnianych 300 neuronów zmniejszono tą liczbę na 80, Wynik tej operacji można zaobserwować na rysunku 6.15.



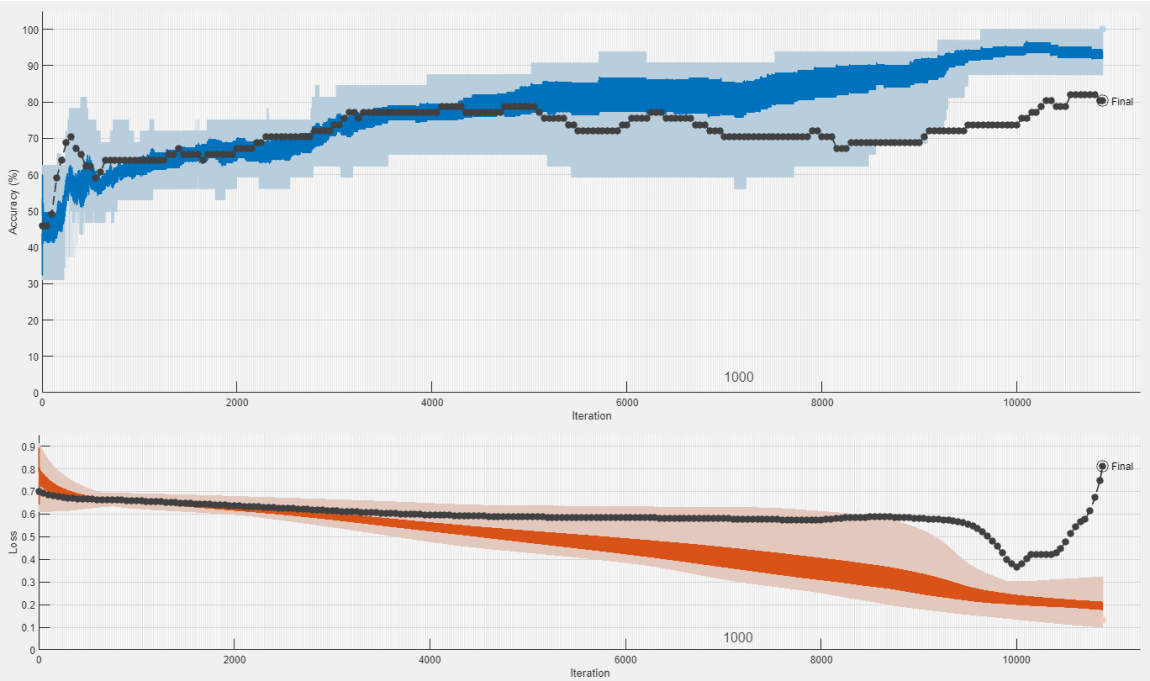
Rysunek 6.15: Wykresy dokładności i funkcji celu obu zestawów: trenującego i testującego. Oznaczenia: niebieski - dokładność zestawu trenującego, czarny górny - dokładność walidacji, pomarańczowy - funkcja celu zestawu trenującego, czarny dolny- funkcja celu walidacji. Sieć o 80 neuronach w warstwie ukrytej, rozmiarze mini batch 32, po 1800 epokach uczenia, ostatecznej dokładności 62%, początkowym współczynniku uczenia 0,00001 oraz regularyzacji L2 0,0001.

Pomimo gorszego wyniku dokładności walidacyjne w stosunku sieci z rysunku 6.15 do sieci z rysunków 6.6 i 6.9 (kolejno 74% i 77%) można było spadek walidacyjnej funkcji celu na szerokim zakresie tj. od ok. 8000 do 11500 iteracji. Tak długi zakres spadku tej funkcji nie występował na rysunkach 6.6 i 6.9, a więc zjawisko overfittingu występowało w mniejszym stopniu. Wywnioskowano, że należy jeszcze bardziej zmniejszyć liczbę neuronów w warstwie ukrytej.

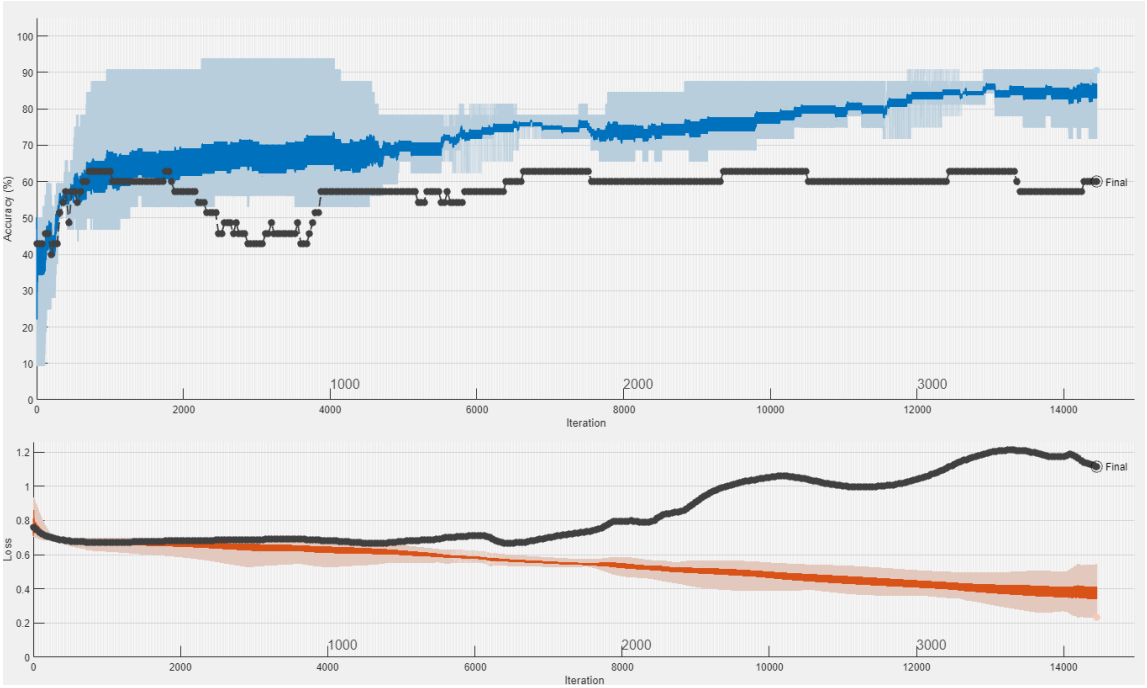


Rysunek 6.16: Wykresy dokładności i funkcji celu obu zestawów: trenującego i testującego. Oznaczenia: niebieski - dokładność zestawu trenującego, czarny górny - dokładność walidacji, pomarańczowy - funkcja celu zestawu trenującego, czarny dolny- funkcja celu walidacji. Sieć o 75 neuronach w warstwie ukrytej, rozmiarze mini batch 32, po 1560 epokach uczenia, ostatecznej dokładności 81,96%, początkowym współczynniku uczenia 0,000004 oraz regularyzacji L2 0,0001.

Na powyższym rysunku funkcja celu walidacji wyraźnie opada i posiada swoje minimum wynoszące około 0,5. W stosunku do początkowej wartości funckji wynoszącej ponad 0,7 można wnioskować, że można osiągnąć niższą wartość minimum. Znów zmniejszono liczbę neuronów w warstwie ukrytej. Otrzymano w ten sposób wykres procesu uczenia z rysunku 6.17. Pomimo gorszej końcowej dokładności walidacyjnej w porównaniu do poprzedniego modelu (80,32 % w stosunku do 81,96 %)można stwierdzić, że minimum funkcji celu walidacji osiągnęło minimum poniżej wartości 0,4. Jest to postęp w minimalizacji overfittingu. Funkcja celu zestawu treningowego nie była bliska zeru w momencie osiągnięcia minimum przez drugą funkcję. Można więc stwierdzić, że dalej należy zmniejszyć liczbę neuronów w warstwie ukrytej.



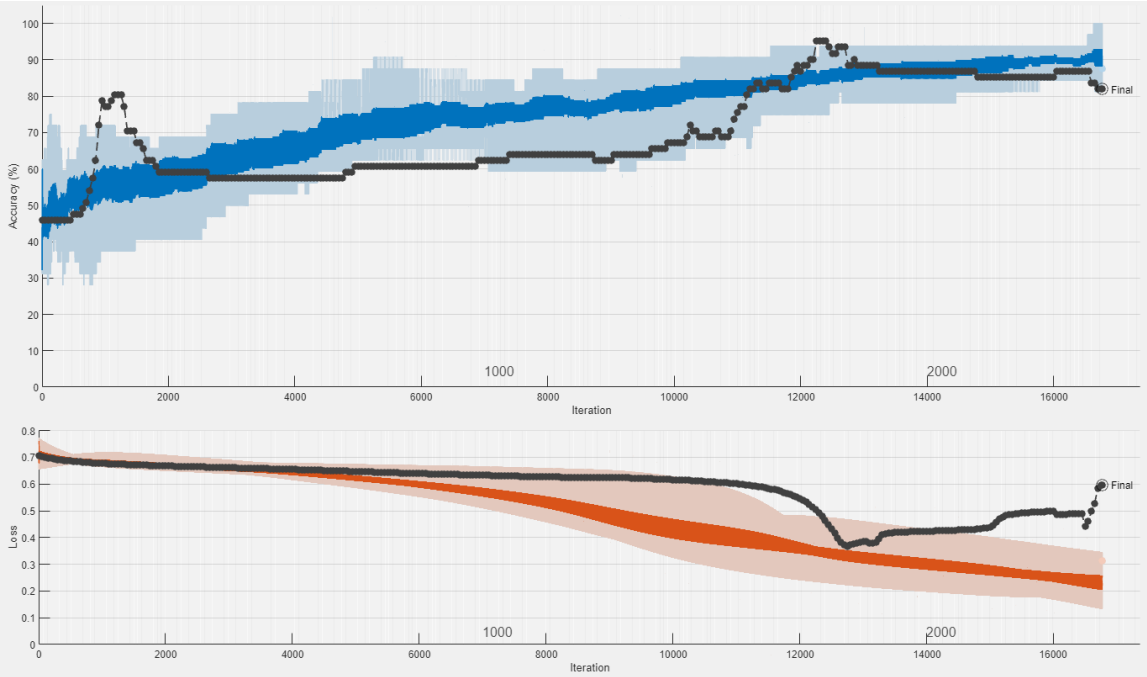
Rysunek 6.17: Wykresy dokładności i funkcji celu obu zestawów: trenującego i testującego. Oznaczenia: niebieski - dokładność zestawu trenującego, czarny górny - dokładność walidacji, pomarańczowy - funkcja celu zestawu trenującego, czarny dolny- funkcja celu walidacji. Sieć o 70 neuronach w warstwie ukrytej, rozmiarze mini batch 32, po 1560 epokach uczenia, ostatecznej dokładności 80,32%, początkowym współczynniku uczenia 0,000005 oraz regularyzacji L2 0,0001.



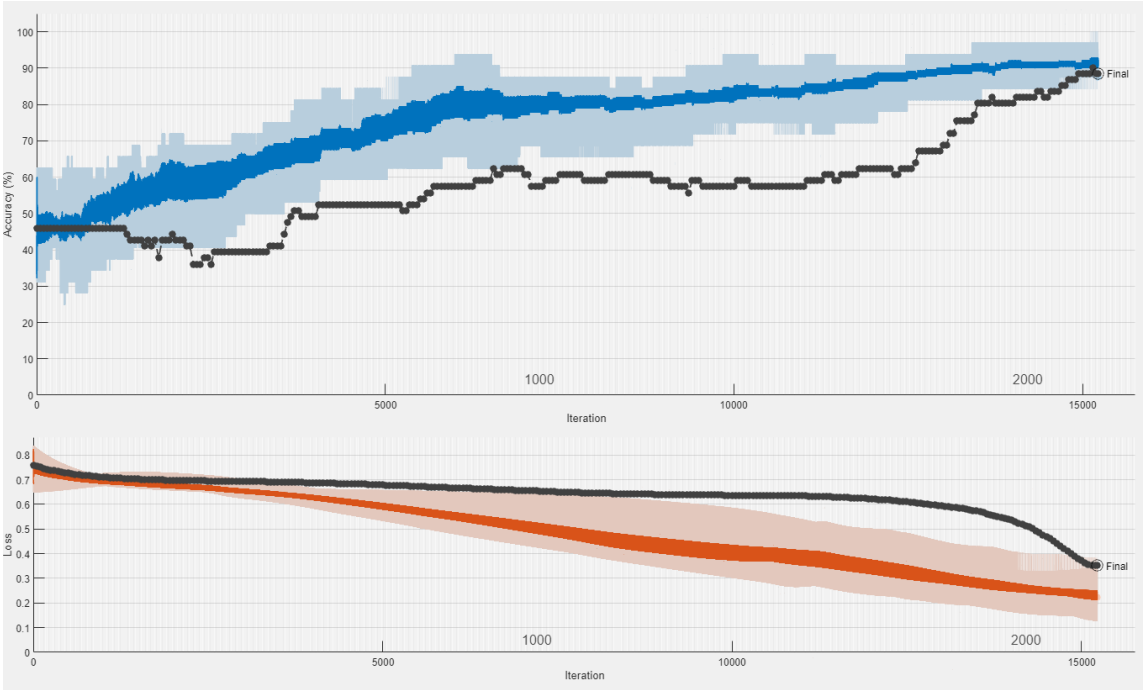
Rysunek 6.18: Wykresy dokładności i funkcji celu obu zestawów: trenującego i testującego. Oznaczenia: niebieski - dokładność zestawu trenującego, czarny górny - dokładność walidacji, pomarańczowy - funkcja celu zestawu trenującego, czarny dolny- funkcja celu walidacji. Sieć o 40 neuronach w warstwie ukrytej, rozmiarze mini batch 32, po 3270 epokach uczenia, ostatecznej dokładności 60%, początkowym współczynniku uczenia 0,000001 oraz regularyzacji L2 0,0001.

Jednakże podczas jedynie zmiany zmiany liczby neuronów w warstwie ukrytej nie uzyskano lepszych wyników jak na rysunkach 6.16 i 6.17 ani pod względem dokładności walidacji, ani pod względem minimalizacji overfittingu. Przykładem jest proces uczenia z rysunku 6.18. Analizując wyniki z rysunków 6.16, 6.17 i 6.18 można stwierdzić, że klasyfikator o najlepszej możliwej dokładności walidacyjnej posiada liczbę neuronów w warstwie ukrytej zawierającej się w przedziale od 40 do 80. Podczas trenowania modeli o liczbie z tego zakresu i współczynniku regularyzacji L2 równemu 0,0001 nie uzyskano lepszej dokładności niż jak z rysunku 6.16 stwierdzono, że należy zmiennić inny hiperparametr. Regularyzacja L2 minimalizuje zjawisko overfittingu, a jako, że dla rysunków 6.16, 6.17 i 6.18 hiperparametr sterujący jej wpływem na proces uczenia pozostawiano w domyślnej wartości zdecydowano się sprawdzić jakie wyniki przyniesie jego zmiana.

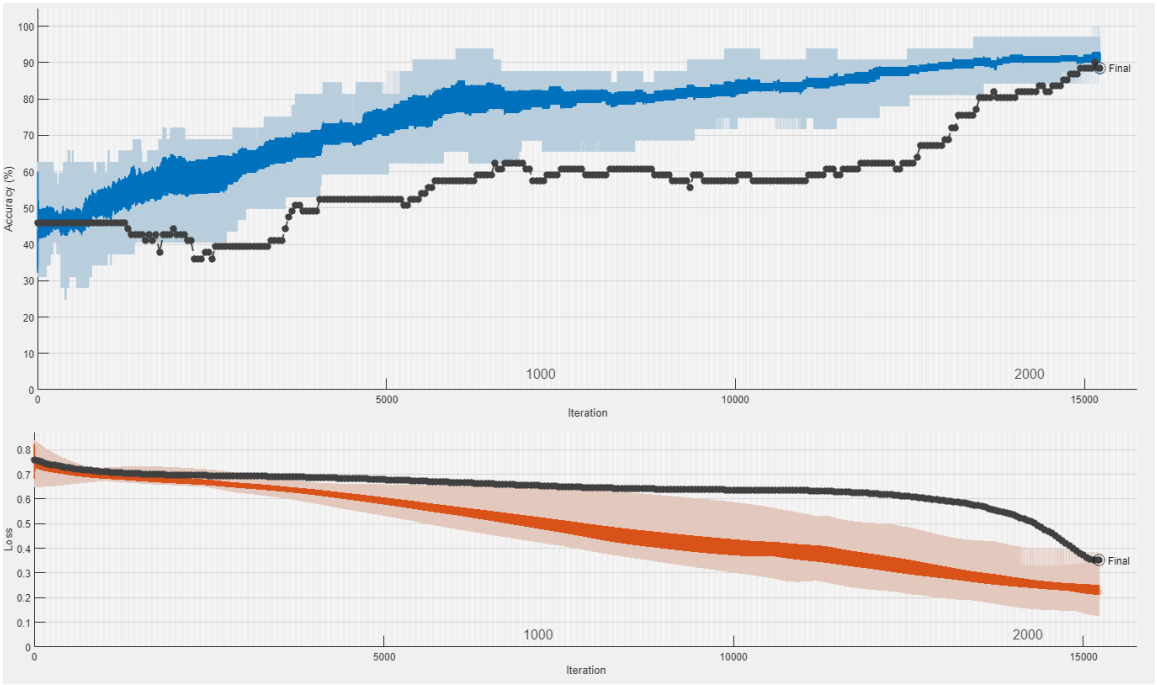
Zgodnie z teorią regularyzacja L2 prowadzi do zmniejszenia wartości wag, które przyczyniły się do zwiększenia funkcji celu. Wagi o wartości bliskiej zru nie mają praktycznie wpływi na ostateczny wynik klasyfikacji. Złożoność sieci, zależna od m.in. liczby neuronów w warstwie ukrytej może być w ten sposób zmniejszona. Można stwierdzić, że regularyzacja L2 prowadzi do uproszczenia sieci w trakcie uczenia. To zjawisko przeciwdziała overfittingowi. Zwiększenie współczynnika regularyzacji λ (równanie 6.9) prowadzi do przyśpieszenia minimalizacji wag wprowadzających błąD, a więc szybszemu uproszczeniu się modelu. Należy więc zwiększyć ten współczynnik z domyślnej wartości 0,0001. Przy zbyt dużym współczynniku λ jednakże może dojść do sytuacji, gdzie wagi zostaną nastawione na wartości bliskie zeru zbyt szybko, tracąc tym samym zdolność do dalszej nauki. Wyniki z zwiększonym współczynnikiem regularyzacji można zaobserwować na rysunkach 6.12, 6.19, 6.20 i 6.21.



Rysunek 6.19: Wykresy dokładności i funkcji celu obu zestawów: trenującego i testującego. Oznaczenia: niebieski - dokładność zestawu trenującego, czarny górny - dokładność walidacji, pomarańczowy - funkcja celu zestawu trenującego, czarny dolny- funkcja celu walidacji. Sieć o 60 neuronach w warstwie ukrytej, rozmiarze mini batch 32, po 2400 epokach uczenia, ostatecznej dokładności 81,96%, początkowym współczynniku uczenia 0,000004 oraz regularyzacji L2 0,0007.



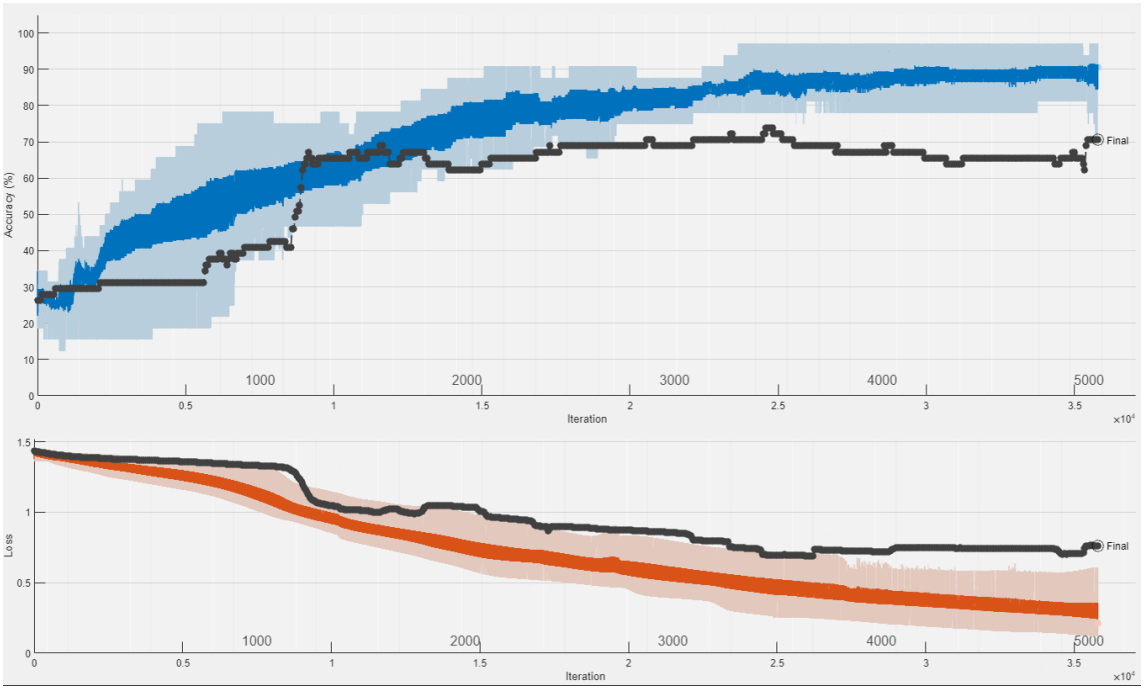
Rysunek 6.20: Wykresy dokładności i funkcji celu obu zestawów: trenującego i testującego. Oznaczenia: niebieski - dokładność zestawu trenującego, czarny górny - dokładność walidacji, pomarańczowy - funkcja celu zestawu trenującego, czarny dolny- funkcja celu walidacji. Sieć o 57 neuronach w warstwie ukrytej, rozmiarze mini batch 32, po 15330 epokach uczenia, ostatecznej dokładności 88,52%, początkowym współczynniku uczenia 0,000004 oraz regularyzacji L2 0,00072.



Rysunek 6.21: Wykresy dokładności i funkcji celu obu zestawów: trenującego i testującego. Oznaczenia: niebieski - dokładność zestawu trenującego, czarny górny - dokładność walidacji, pomarańczowy - funkcja celu zestawu trenującego, czarny dolny- funkcja celu walidacji.. Sieć o 57 neuronach w warstwie ukrytej, rozmiarze mini batch 32, po 15330 epokach uczenia, ostatecznej dokładności 90,16%, początkowym współczynniku uczenia 0,000004 oraz regularyzacji L2 0,00072.

Jak widać na rysunkach 6.12, 6.20 i 6.21 osiągnięto dokładności walidacyjne powyżej 85%, co oznacza, że zwiększenie współczynnika regularyzacji L2 oraz obniżenie liczby neuronów w warstwie ukrytej skutkowało poprawą jakości otrzymanych klasyfikatorów w stosunku do modelu uzyskanego w procesie uczenia z rysunku 6.16. Klasyfikator o 90,16 % dokładności walidacyjnej jest modelem, który najlepiej uogólnił dane zestawu treningowego. **Można więc stwierdzić, że jest to najlepszy model zbudowany w tym projekcie.** Jednakże na rysunku 6.21 można zaobserwować, że proces uczenia został zatrzymany zbyt szybko. Jest to dowodem na to, że dla kolejnych iteracji uczenia sieci o tych samych hiperparametrach można uzyskać jeszcze lepszy model. Wynik ten jednak jest zadowalający.

Sieć o podobnych hiperparameterach zastosowano także do rozpoznawania także w jakim drewnie wiercono, bądź wyciągano wiertło (cztery klasy z podrozdziału 4.1). Najlepszą dokładność walidacyjną uzyskano na podstawie procesu uczenia z poniższego rysunku.



Rysunek 6.22: Wykresy dokładności i funkcji celu obu zestawów: trenującego i testującego. Oznaczenia: niebieski - dokładność zestawu trenującego, czarny górny - dokładność walidacji, pomarańczowy - funkcja celu zestawu trenującego, czarny dolny- funkcja celu walidacji. Sieć o 50 neuronach w warstwie ukrytej, rozmiarze mini batch 32, po 5120 epokach uczenia, ostatecznej dokładności 70,49%, początkowym współczynniku uczenia 0,000004 oraz regularyzacji L2 0,0007. Proces uczenia dla klasyfikacji czterech klas.

Klasyfikacja 4 klas jest jednak zadaniem trudniejszym, wymagającym tak jak w przypadku dwóch klas, dobrej nastawy hiperparametrów lub nawet zmiany architektury sieci i innych czynności w procesie wstępnego przetwarzania i wydobywania cech. Jednakże tak jak w przypadku klasyfikatora dwóch klas model o 70, 49% dokładności walidacyjnej jest dobrym punktem wyjściowym.

7. PODSUMOWANIE

W toku przeprowadzonych badań zbudowano modele klasyfikujące, które w zadowalającym stopniu rozpoznają sygnały dźwiękowe z nagrań i przydzielają im prawidłowe etykiety. Są nimi modele, które osiągnęły dokładność walidacyjną powyżej 85%, a więc o numerach 7, 10, 11 w tabeli zbiorczej 6.1. Mają one bardzo podobne nastawy hiperparametrów. Sieć o dokładności walidacyjnej 90,16% posiada następujące hiperparametry:

- Solver - Adam
- Współczynnik rozkładu gradientu (β_1) - 0,9
- Kwadratowy współczynnik rozkładu gradientu (β_2) - 0,999
- epsilon (ϵ) - 10^{-8}
- Próg gradientu - nieskończoność
- Rozmiar mini batch - 32
- Maksymalna liczba epok - 20000
- Sieć wyjściowa - wynik ostatniej iteracji treningowej.
- Tasowanie - nigdy
- Częstotliwość walidacji - 50
- Tryb tempa uczenia - stałe
- Początkowy współczynnik tempa uczenia (α) - 0,000004
- Liczba neuronów w warstwie ukrytej - 57
- Regularyzacja L2 (λ) - 0,00072

Wnioski wyciągnięte na podstawie pracy w projekcie wyglądają następująco:

- zestaw trenujący w postaci 245 nagrań wystarczył do nauczenia sieci oraz uogólnienia zestawu danych. Przykładem jest postęp jak można zaobserwować porównując procesy uczenia z rysunków 6.6 - 6.21 klasyfikatorów dwóch klas.

  Jednakże do zbudowania klasyfikatora czterech klas taki zestaw danych może już nie wystarczyć. Przykładowo porównując tabele 4.1 i 4.3 można zauważyć, że ten sam zestaw danych oferował (przed procesem wstępnego przetwarzania) 112 nagrań dla klasy wyciągania wiertła w klasyfikatorze 2 klas, a jedynie 40 dla klasy wyciągania z dębu w klasyfikatorze czterech klas. Jako, że przyjęte klasy są rozłączne dla klasyfikatora czterech klas ilość nagrań do nauki rozpoznawania danej klasy mogła spaść prawie trzykrotnie w porównaniu do klasyfikatora dwóch klas. Proces uczenia na podstawie tak zmniejszonego zestawu nagrań mógł oznaczać nie pokrycie wszystkich możliwych zjawisk występujących w danych. Efektem była gorsza jakość uogólniania danych. Można ją zaobserowować na rysunku 6.22, gdzie osiągnięto 70,49% dokładności walidacyjnej.Rozwiązaniem tego problemu jest rozszerzenie danych pomiarowych [5].Polega ono na stworzeniu nowych nagrań na podstawie transformacji rzeczywistych pomiarów. Transformacje polegają m.in.na zmniejszeniu amplitudy sygnału, przesunięcie go w czasie. Jednakże przy

obecnej formie zestawu danych można stwierdzić, że zbudowanie klasyfikatora czterech klas jest dużym wyzwaniem.

- Procedura pomiarowa może być znacznie przyśpieszona poprzez automatyzację zmiany nazwy plik .wav po każdym uruchomieniu skryptu pomiarowego. Wyeliminowałoby to potrzebę wprowadzania każdej nazwy ręcznie.

- Wybrana metoda poolingu (podrozdział 6.3) dodawała zera do wkładu do sieci. Wprowadza to zakłócenia, które mogły spowodować nasilenie się zjawiska overfittingu. Wskazane jest więc przetestowanie metod poolingu, które nie dodają zer do wkładu, który jest pokazany na rysunku 5.7.

- Najlepsze uzyskane klasyfikatory (numery 7, 10, 11 w tabeli zbiorczej 6.1) mogą zostać dalej poprawione po zastosowaniu się do algorytmu działania z rysunku 6.1. Metodyka uczenia sieci w nim przedstawiona jest słuszna z uwagi na otrzymanie wyników powyżej 85%.

- Najdłuższy proces uczenia sieci (numer 19 w tabeli zbiorczej 6.1) wynosił 59 minut. Zdecydowana większość sieci uczyła się poniżej 30 minut. W celu skrócenia czasu przetwarzania można zamiast karty graficznej komputera zastosować wyspecjalizowany układ elektroniczny. Może nim być układ typu programowalnej macierzy bramek (ang. Field Programmable Gate Array). Układy te bardzo dobrze nadają się do zadań z zakresu głębokiego uczenia z uwagi na ich możliwości równoległego przetwarzania danych. Równolegle wymnażanie przyśpieszyłoby proces uczenia z uwagi na równolegle położone neurony warstwy ukrytej. Przetworzenie całości danych przez sieć w jednej iteracji znacznie skróciłoby czas uczenia. Dedykowany układ elektroniczny ograniczyłby także zużycie mocy potrzebnej do obliczeń [23].

- W podrozdziale 4.1 stwierdzono, że etykietyzacja nagrań przy pomocy ich nazw plików .wav jest zabiegiem dużo szybszym od ręcznego wprowadzania każdej etykiety w aplikacji Signal Labeler. Aplikacja ta oferuje jednakże bardziej zaawansowane możliwości oznaczeń nagrań, np. czy w przebiegu czasowym występuje dane zjawisko w określonym momencie pomiaru. Takowe oznaczenia wymagają głębszego zrozumienia analizowanych pomiarów. W celu zgromadzenia dokładniejszej wysokopoziomowej informacji o sygnale, należy lepiej zaznajomić się z oferowanymi funkcjami aplikacji. Wspomoże to proces uczenia.

- Pomimo otrzymania zadowalających wyników tj. klasyfikatorów wysokiej dokładności walidacyjnej (numery 7, 10, 11 w tabeli zbiorczej 6.1) przy pomocy MFCC należy dalej testować inne zestawy dostępnych typów cech. Matlab oferuje 24 inne typy cech do wydobycia. Testowanie różnych typów cech lub ich kombinacji może przynieść pozytywne skutki, które pozwolą na możliwe zbudowanie jeszcze lepszych klasyfikatorów niż pokazano w tabeli 6.1.

8. LITERATURA

[1] S. El Hamdi, A. Abouabdellah i M. Oudani, "Industry 4.0: Fundamentals and Main Challenges," w *2019 International Colloquium on Logistics and Supply Chain Management (LOGISTIQUA)*, 2019, s. 1. DOI: 10.1109/LOGISTIQUA.2019.8907280.

[2] "Co to jest Big Data." (2022), adr.: https://www.oracle.com/pl/big-data/what-is-big-data/.

[3] "What is IoT." (2022), adr.: https://www.oracle.com/in/internet-of-things/what-is-iot/#industrial-iot.

[4] M. Tabladillo. "The Team Data Science Process lifecycle." (2022), adr.: https://learn.microsoft.com/en-us/azure/architecture/data-science-process/lifecycle.

[5] T. Tran, N. T. Pham i J. Lundgren, "A deep learning approach for detecting drill bit failures from a small sound dataset," *Scientific Reports*, 2022. adr.: https://doi.org/10.1038/s41598-022-13237-7.

[6] W. Zhang, G. Yang, Y. Lin, C. Ji i M. M. Gupta, "On Definition of Deep Learning," w *2018 World Automation Congress (WAC)*, 2018, s. 2. DOI: 10.23919/WAC.2018.8430387.

[7] "getaudiodata." (2022), adr.: https://www.mathworks.com/help/matlab/ref/audiorecorder.getaudiodata.html.

[8] "Speaker Identification Using Pitch and MFCC." (2022), adr.: https://www.mathworks.com/help/audio/ug/speaker-identification-using-pitch-and-mfcc.html.

[9] "Sequence Classification Using Deep Learning." (2022), adr.: https://www.mathworks.com/help/deeplearning/ug/classify-sequence-data-using-lstm-networks.html.

[10] "Using Signal Labeler App." (2022), adr.: https://www.mathworks.com/help/signal/ug/using-signal-labeler.html.

[11] A. A. Tokuç. "Splitting a Dataset into Train and Test Sets." (2022), adr.: https://www.baeldung.com/cs/train-test-datasets-ratio.

[12] "audioFeatureExtractor." (2022), adr.: https://www.mathworks.com/help/audio/ref/audiofeatureextractor.html.

[13] S. Chatterjee. "What is Feature Extraction? Feature Extraction in Image Processing." (2022), adr.: https://www.mygreatlearning.com/blog/feature-extraction-in-image-processing/.

[14] F. D. i. i. r. Purves D Augustine GJ, "The Audible Spectrum," w *Neuroscience, 2nd edition*. 2001.

[15] mlearnere. "Learning from Audio: The Mel Scale, Mel Spectrograms, and Mel Frequency Cepstral Coefficients." (2021), adr.: https://towardsdatascience.com/learning-from-audio-the-mel-scale-mel-spectrograms-and-mel-frequency-cepstral-coefficients-f5752b6324a8.

[16] C. Malkin. "Fast Fourier Transform." (2019), adr.: https://towardsdatascience.com/fast-fourier-transform-937926e591cb.

[17] "dct." (2022), adr.: https://www.mathworks.com/help/signal/ref/dct.html?searchHighlight=dct&s_tid=srchtitle_dct_1.

[18] "fscmrmr." (2022), adr.: https://uk.mathworks.com/help/stats/fscmrmr.html.

[19] D. Ibanez. "Encoder-Decoder Models for Natural Language Processing." (2022), adr.: https://www.baeldung.com/cs/nlp-encoder-decoder-models#1-what-are-vanishingexploding-gradients.

[20] E. Zvornicanin. "Differences Between Bidirectional and Unidirectional LSTM." (2022), adr.: https://www.baeldung.com/cs/bidirectional-vs-unidirectional-lstm.

[21] "Accuracy and Loss." (2019), adr.: https://machine-learning.paperspace.com/wiki/accuracy-and-loss.

[22] "What Is Overfitting?" (2022), adr.: https://aws.amazon.com/what-is/overfitting/.

[23] J.-C. See, J.-J. Chang, H.-F. Ng, K.-M. Mok i W.-K. Lee, "Design and Implementation of Deep Learning Core for FPGA Platform," w *2021 International Conference on Computer and Information Sciences (ICCOINS)*, 2021, s. 207–212. DOI: 10.1109/ICCOINS49721.2021.9497159.