AGH
Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie
Wydział Inżynierii Mechanicznej i Robotyki

Praca inżynierska

Jakub Gucik

kierunek studiów: Inżynieria Akustyczna

specjalność: Inżynieria Dźwięku w Mediach i Kulturze

Automatyczna ocena poprawności wymowy w języku angielskim

Opiekun: dr inż. Marcin Witkowski

Kraków, 2024

Chciałbym serdecznie podziękować
Panu dr inż. Marcinowi Witkowskiemu
za pomoc merytoryczną oraz redakcyjną,
opiekę promotorską, a także za poświęcony czas.

Dziękujemy polskiej infrastrukturze obliczeń wielkiej skali
PLGrid (Centra HPC: ACK Cyfronet AGH)
za udostępnienie sprzętu komputerowego i wsparcie
w ramach grantu obliczeniowego nr. PLG/2023/016645.

Spis treści

1 Wprowadzenie

Ostatnia dekada to okres gwałtownych zmian na rynku technologicznym i informatycznym powodowanych gwałtownym rozwojem branży zajmującej się sztuczną inteligencją oraz uczeniem maszynowym. W wyniku modernizacji i nowych rozwiązań wprowadzonych przez te dyscypliny nauki, wiele z zagadnień i innowacji, które dotychczas były ograniczone poprzez brak narzędzi i barierę obliczeniową, jest ponownie rozpatrywanych i ulepszanych do obecnych standardów osiągając lepsze i bardziej obiecujące wyniki od swoich poprzedników.

Szansę rozwoju i nowoczesnej implementacji otrzymał również algorytm Goodness of Pronunciation (GoP), którego początki sięgają lat 90 XX wieku. Jednym z pierwszych tekstów naukowych traktujących o wykorzystaniu algorytmu zaliczającego się w poczet programów Computer-Assisted Language Learning (CALL), służących do oceny wymowy i interaktywnej nauki, sformułowany został w 1998 roku na uniwersytecie w Cambridge [23], a z biegiem lat został usprawniony oraz wykorzystany w próbach wprowadzenia nowych rozwiązań dotyczących oceny wymowy na poziomie fonetycznym.

Najważniejszym założeniem przedstawionej problematyki jest dostarczenie skutecznego narzędzia do automatycznej oceny wymowy umożliwiającego samodzielną naukę języka angielskiego, a w perspektywie pojawiających się adaptacji, również innych języków. Co więcej takie rozwiązanie może pomóc nie tylko w podstawowej nauce języka, ale również w detekcji wad wymowy i pracy nad jej poprawą. W przyszłości algorytm ten mógłby pomóc i posłużyć nawet do wspomagania wykrywania chorób i schorzeń wpływających na trakt głosowy.

System powinien być trenowany i przystosowywany do użycia ogólnego, czyli niezależnego od płci, wieku czy stanu zdrowotnego danego użytkownika, gdyż celem funkcjonującego algorytmu i modeli jest określanie poprawnej wymowy niezależnie od cech mówcy. Stwarza to problem klasyfikacji wyjątków czy generalizacji zagadnienia, jednak tworzone modele w oparciu o dostarczone dane mogą być odpowiednio przystosowane i przetworzone, tak by w ramach określonej grupy prezentować dobre wyniki. Jest to problem dotyczący wszystkich rozwiązań bazujących na uczeniu maszynowym i nieustannie prowadzący do nowych poprawek rozwiązań dotyczących objętości i specyfikacji danych wejściowych. Samo zagadnienie jest interdyscyplinarne, gdyż wykorzystuje wiedzę z różnych dziedzin takich jak matematyka, statystka, akustyka i techniki programistyczne. Ostatecznym celem prowadzonych badań jest stworzenie obiecującego i skutecznego narzędzia do automatycznej nauki języka angielskiego czy walidacji poprawności wymowy.

Pomimo pojawiających się zmian i alternatyw jakie zaproponowano w ciągu niemal 30 lat, u podstaw zagadnienia niezmienne pozostaje wykorzystanie uniwersalności matematyki oraz statystyki w wykorzystaniu narzędzi takich jak:

- ekstrakcja cech mówcy i wykorzystanie parametrów mel-cepstralnych (ang. Mel-Frequency Cepstral Coefficients, MFCC),
- wykorzystanie techniki Forced Allignment (FA),
- wykorzystanie Ukrytych Stanów Markov'a (ang. Hidden Markov Model, HMM).

W ramach wykorzystywanych narzędzi w obecnie proponowanych rozwiązaniach, istnieje różnica w implementacji względem pierwotnych prac naukowych, a także w doborze pozostałych elementów wykorzystywanych w kolejnych próbach udoskonalenia algorytmu. Najważniejszą znaczącą różnicą pomiędzy oryginalnymi wersjami GoP, a najnowszymi proponowanymi rozwiązaniami jest zastosowanie sieci neuronowych. W pierwotnych publikacjach w modelowaniu wykorzystywano modelowanie miksturami Gaussa (ang. Gaussian Mixture Model, GMM), które skutecznie rozwijały przedstawione zagadnienie, jednak okazały się niewystarczające, by uzyskać zadowalające wyniki przy systemach wyposażonych w ten model probabilistyczny. W związku z tym wraz z nasilającym się rozwojem sieci neuronowych podjęto udane próby wprowadzenia modeli opartych na uczeniu głębokim do algorytmu GoP.

Obecnie, jednym z częściej cytowanych podejść w implementacjach [7] [16] [17] [24] jest metoda przedstawiona w publikacji wydanej w 2015r., w czasopiśmie "Speech Communication"praca pod tytułem: "Improved mispronunciation detection with deep neural network trained acoustic models and transfer learning based logistic regression classifiers"[11]. Jest to realizacja bazująca na sieciach neuronowych i na niej opiera się metodyka wykorzystana w wykonaniu tego projektu oraz w użytym zestawie narzędzi Kaldi [4], który to udostępnia wymagane narzędzia i modele umożliwiające stworzenie i przetestowanie działającego skrypu opartego o GoP.

W związku z powyższym w tej pracy inżynierskiej przeanalizowano i wykorzystano podejście zaproponowane w pracy naukowej przywołanej w poprzednim akapicie [11]. W implementacji działającego algorytmu GoP wykorzystano narzędzia Kaldi [4], a w celu zapewnienia sieci neuronowej wykorzystywanej w GoP, wytrenowano odpowiednie modele z użyciem architektur przedstawionych w rozdziale . Finalnie przeanalizowano wyniki i działanie wszystkich elementów stworzonego systemu, w celu ewaluacji skuteczności zastosowanej metody.

1.1 Motywacja

Motywacja do opracowania i przeanalizowania podanego tematu zrodziła się wraz z dostrzeżonymi możliwościami wykorzystania algorytmu GoP, a także ze względu na małe nasycenie rynku podobnymi rozwiązaniami i technologiami. Głównym celem miary GoP jest pomoc w nauce języka angielskiego - jednak patrząc szerzej na wspomniany temat, można zauważyć, iż w przyszłości można wykorzystać to zagadnienie jako detektor błędów wypowiedzi, a co za tym idzie jako narzędzie diagnostyczne. Wprowadzając również rozszerzenia i dostosowując elementy składowe, a także programy działające w obrębie wspomaganego uczenia języków, może posłużyć w pomocy w zadaniach akustycznych ludziom z szerokiego spektrum chorób i schorzeń, nie tylko kwalifikujących się w poczet wad akustycznych. GoP znajduje również zastosowanie jako narzędzie w fazie ewaluacji systemów typu Automatic Speech Recognition (ASR). Algorytm jest także zagadnieniem rozwijającym się i osiągającym zróżnicowane wyniki w zależności od zastosowanych rozwiązań - obecnie dzięki publikacji z 2015 roku [11] rozpoczęto intensywniejsze prace i rozwój nad wprowadzaniem sieci neuronowych do algorytmu, jak również eksperymentuje się z dodaniem mniejszych usprawnień.

Dodatkowym powodem do zbadania tego zagadnienia jest zainteresowanie technologią mowy oraz szeroko rozumianym uczeniem maszynowym. Wykonane projekty oraz zadania w ramach nauki na uczelni oraz wykraczające poza standardowy kurs, pozwoliły zapoznać się i opanować w szerszym zakresie możliwości oraz dostępne narzędzia rynku sztucznej inteligencji, zachęcając do zgłębiania tej tematyki.

W związku z powyższym, kolejnymi priorytetami w poszukiwaniu tematu i pomysłu na wykonanie pracy inżynierskiej były:

- zagadnienie mogące pomóc w diagnozie/leczeniu schorzeń i defektów akustycznych,
- tematyka nowa, rozwijająca się, nie rozpowszechniona szeroko na rynku,
- temat w zakresie powyższych zainteresowań.

1.2 Cel pracy

Celem pracy nad tematem inżynierskim była realizacja programowa przedstawionego problemu za pomocą dostępnych narzędzi, a także przeprowadzenie studium literaturowego dotyczącego badanego algorytmu. W zakres działań wchodził także przegląd dostępnych środków i technik mogących umożliwić realizację oraz przyczynić się do rozwoju przedstawionego zagadnienia. Poniżej przedstawiono dokładnie wypunktowane cele pracy inżynierskiej ustalone przed przystąpieniem do zadania:

- Analiza dostępnych metod oraz baz danych umożliwiających opracowanie prototypu metody oceny GoP,
- Implementacja i uruchomienie prototypu wybranej metody GoP,
- Ocena skuteczności zaimplementowanej metody.

2 Studium literaturowe i wykorzystane dane

2.1 Definicja i opis algorytmu GoP

Algorytm GoP opiera się na prawdopodobieństwach a posteriori wykorzystywanych do oceny wymowy na poziomie fonemów mówcy. W pierwszym, wyżej wspomnianym oryginalnym modelu wykorzystującym Gaussian Mixture Model-Hidden Markov Model (GMM-HMM) zaproponowanym w 1998 roku [23] wyrażenie GoP zostało zdefiniowane jako znormalizowana logarytmiczna miara prawdopodbieństwa a posteriori w odniesieniu do fonemów i czasu ich trwania. Przed zdefiniowaniem algorytmu umożliwiającego ocenę GoP, należy wytłumaczyć czym są Ukryte Stany Markowa.

Ukryte Modele Markowa są narzędziem statystycznym użytecznym w modelowaniu i analizie sekwencji zdarzeń. Służą one także do przewidywania zjawisk o takim charakterze i ich klasyfikacji. W wypadku sygnału akustycznego, ciągiem zdarzeń będą parametryzowane, krótkie sekwencje fonemów opisywane wektorami przy dodatkowym założeniu, iż te najmniejsze elementy są stacjonarne, czyli nie zmieniają się w określonym przedziale czasu. HMM umożliwia modelowanie jednowymiarowych szeregów czasowych pojedynczego procesu.

Sformułowana zależność umożliwiająca określenie poprawności wymowy została opisana jako [23]

$$GOP(p) = \frac{1}{t_e - t_s + 1} \log P(p|o), \tag{1}$$

gdzie o to obserwacje wejściowe, p to fonem, P to prawdopodobieństwo, t_s to początek trwania fonemu (pierwszy indeks ramki), a t_e to koniec trwania fonemu (ostatni indeks ramki).

Przyjmując odpowiednio, iż $p(q_i) \approx p(q_j)$ dla dowolnego q_i, q_j, będącymi parą fonemów z całkowitego zestawu fonemów Q, odpowiednio prawdziwa jest zależność

$$\log P(p|o) = \frac{P(o|p) \cdot P(p)}{\sum_{q \epsilon Q} P(o|p) \cdot P(q)} \approx \frac{P(o|p)}{\Sigma_{q \epsilon Q} P(o|q)} \tag{2}$$

w której wykorzystane zostało twierdzenie Bayesa, wiążące prawdopodobieństwa dwóch zdarzeń warunkujących się nawzajem.

Licznik w równaniach (1) i (2) jest odpowiednio wyliczany poprzez proces Forced Allignment, czyli przyporządkowania transkrypcji ortograficznych do odpowiednich fragmentów lub całości nagrań audio w celu automatycznej segmentacji fonetycznej. Mianownik w równaniu (2) jest natomiast kalkulowany na bazie algorytmu dekodującego Viterbiego[1]. W przedstawionym przypadku stosowany jest w odniesieniu do Ukrytych Modeli Markowa do dekodowania sekwencji stanów ukrytych o największym prawdopodobieństwie zwrócenia sekwencji obserwacji z użyciem niewymuszonej pętli fonetycznej. Konkretnie, działanie tej procedury jest oparte na kryterium najwyższej wiarygodności w celu wyszukania optymalnej ścieżki dojścia dekodera do aktualnego stanu. Droga ta składa się ze ścieżki o najmniejszej metryce dojścia do któregoś ze stanów poprzednich oraz przejścia do aktualnego stanu. W wypadku algorytmu GoP będą to stany odpowiadające za konkretne fonemy, a samą operację można wyrazić za pomocą iteracji i wspomnianą wcześniej pętlą.

Zastosowane w pierwszym rozwiązaniu pomysły i modele GMM, były wprowadzeniem do wykorzystania w późniejszym okresie sieci neuronowych i przeformułowaniu oryginalnej definicji pod nowe metody dostępne na dzisiejszym rynku. Formuła matematyczna Goodness of Pronunciation-Neural Network (GoP-NN) oparta o głębokie uczenie jest odmienna od przedstawionej w równaniach (1) i (2). Sama definicja GoP-NN pierwszy raz pojawiła się siedemnaście lat później po oryginalnej tezie wykorzystującej GMM-HMM [23] w czasopiśmie "Speech Communucation"[11].

Przed wyjaśnieniem, czym tak naprawdę jest GoP-NN i jak różni się w swojej definicji od poprzedników należy wyjaśnić dwa znaczące pojęcia używane w opisie podejścia bazującego na sieciach neuronowych: trifony oraz senony.

Trifony to najprościej mówiąc zestawianie trzech fonemów następujących po sobie bezpośrednio, z czego każdy wyraz o długości n fonemów zawiera $n\text{-}2$ trifonów (pod warunkiem, iż $n \geqslant 2$.)

W wypadku pojęcia senonów, by odpowiednio zrozumieć i przedstawić ich istotę należy zgłębić konstrukcje głębokich sieci neuronowych oraz algorytmu HMM. W architekturze Deep Neural Network (DNN) połączonej z krokami ekstrakcji cech i danych, jednym z zadań obsługiwanych przez algorytm jest transformacja danych wejściowych w postaci akustycznej na zestaw fonemów. W nauce jaką jest fonetyka oraz lingwistyka, fonem zdefiniowany jest jako jakikolwiek dźwięk bądź gest kategoryzowany jako zdarzenie fizyczne bez względu na miejsce jakie zajmuje w fonologii danego języka.

[1]Głównym zastosowaniem algorytmu jest dekodowanie kodów splotowych i implementacja w technologiach telekomunikacyjnych m.in.. w odbiornikach nieliniowych

Bazując na tej definicji i konstrukcji używanych narzędzi senonem zostanie określone każde zdarzenie akustyczne, włącznie z ciszą zarejestrowane na poziomie fonetycznym, będące odzwierciedlane przez odpowiedni Stan Markowa. W związku z tym w przeciwieństwie do fonemów w zakres senonów wliczymy także ciszę w trakcie przerw w wymowie lub następującą przed oraz po wypowiedzi, a także sygnały akustyczne nie będące mową np. gwizdy. Pełne omówienie tego zagadnienia można znaleźć w oryginalnej pracy z 1992 roku [13].

GoP-NN w definicji jest opisany za pomocą logarytmu fonetycznego prawdopodobieństwa a posteriori (ang. Log Phone Posterior, LPP) wykorzystującego stosunek pomiędzy rozważanym fonemem, a najwyżej punktującym w zestawieniu. Formuła LPP opisana jest jako

$$LPP(p) = \log P(p|o; t_s, t_e) \tag{3}$$

i wykorzystywana jest w definicji GoP-NN, która przedstawia się następująco

$$GOP(p) = \log \frac{LPP(p)}{\max_{q \epsilon Q} LPP(q)}. \tag{4}$$

Formułę LPP przedstawioną w równaniu (3) przybliża się do postaci upraszczającej obliczenia opisanej jako

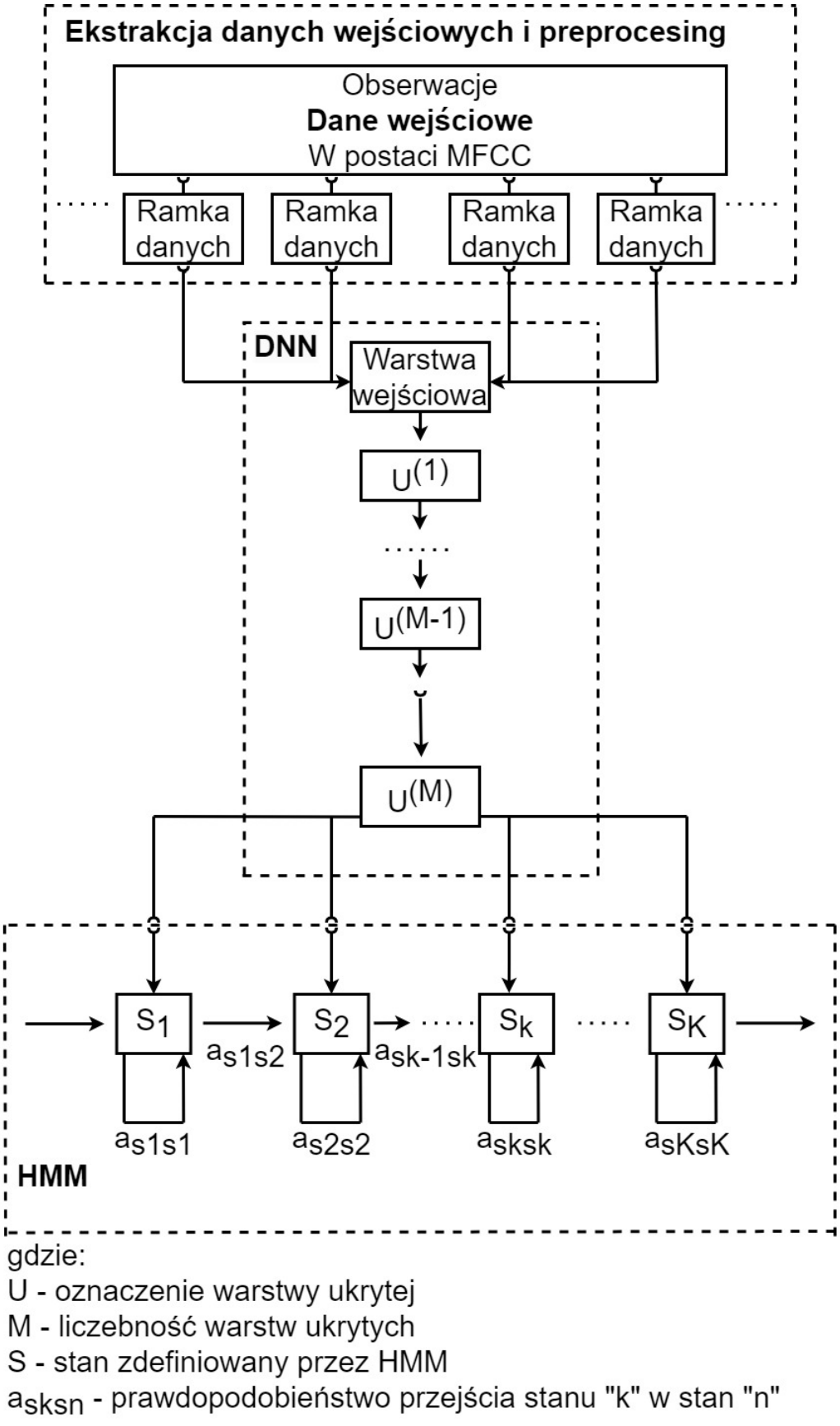
$$LPP(p) \approx \frac{1}{t_e - t_s + 1} \sum_{t=t_S}^{t_e} \log(P(p|o_t)). \tag{5}$$

Wyrażenie $p(p|o_t)$ przedstawiane jest jako

$$P(p|o_t) = \sum_{s \epsilon p} P(s|o_t) \tag{6}$$

gdzie s to oznaczenie senonu, a $s|s \epsilon p$ to stany należące do tych trifonów, których obecnym fonemem jest p.

Interpretując powyższe formuły matematyczne i przekładając je na wykorzystane narzędzia, przedstawiono na Rysunku 1 graficznie proces kolejnych kroków i elementów działającego systemu.



Rysunek 1: Uproszczony schemat teoretyczny działającego algorytmu GoP

2.2 Implementacja rozwiązań i algorytmu

Większość zadań w przedstawionej pracy została wykonana z użyciem skryptów i narzędzi dostępnych w zestawie Kaldi [4], a podstawowymi językami użytymi w kodzie są skrypty typu bash, Perl, Python oraz C++. W trakcie pracy wykorzystano również dostępną wiedzę, przeprowadzono odpowiednie modyfikacje i implementacje programistyczne zagadnień, a także dodano potrzebne elementy dotyczące treningu sieci i ewaluacji GoP, a także analizy i interpretacji wyników.

W związku z wymogiem zapewnienia odpowiedniej sieci neuronowej, będącej bazowym elementem wykorzystywanym w GoP, zdecydowano się użyć modeli automatycznego rozpoznawania mowy ASR. Powodem takiej decyzji jest fakt, iż wiele z obecnie dostępnych sieci neuronowych użytych w systemach ASR, trenowanych jest do przetwarzania danych i produkowania sekwencji senonów. Umożliwiają one otrzymanie niezbędnych w GoP, prawdopodobieństw a posteriori dla fonemów, poprzez sumowanie prawdopodobieństw a posteriori senonów koresponduj̨ących z docelowym fonemem. Dokładne studium tego zagadnienia przedstawiono w pracy "An improved DNN-based approach to mispronunciation detection and diagnosis of L2 learners' speech"[12].

Poniżej przedstawiono szczegółowy opis kroków i rozwiązań, które zostały wykorzystane w metodzie z wyszczególnieniem na dwie główne fazy: wytrenowanie modelu ASR oraz ewaluacja modelu z algorytmem na metryce GoP.

2.2.1 Wykorzystane bazy danych i narzędzia

W realizacji pracy inżynierskiej przeanalizowano dostępne bazy danych w poszukiwaniu zestawu spełniającego wymagania przedstawione poniżej:

- Baza danych na zasadach Open-Source, bądź innych umożliwiających darmowy i publiczny dostęp,
- Korpus posiadający i udostępniający transkrypcje plików audio,
- Pliki audio w bazie posiadające ocenę fonetyczną umożliwiającą ewaluację wyników miary GoP,
- Baza danych zawierająca co najmniej 200h danych audio do treningu ASR,
- Dane w części do ewaluacji GoP składające się z nagrań w nie natywnym angielskim.

Ze względu na przyjęte założenia w projekcie inżynierskim wykorzystano dwie z wszystkich przeanalizowanych baz danych, zawierających nagrania audio w formie wypowiedzi zróżnicowanych mówców oraz zestaw narzędzi pomocniczych. W celu zapoznania z elementami bazy oraz nomenklaturą przed przystąpieniem do omówienia procesu, przedstawiono wykorzystane narzędzia oraz zestawy danych w kolejnych sekcjach.

Speechocean762 [15]

Pierwszą bazą danych wykorzystywaną bezpośrednio w algorytmie GoP jest korpus mowy Speechocean762 [15]. Inicjatywą dla zbioru było stworzenie zestawu danych akustycznych zawierających mowę dla specyficznych zadań oceny wypowiedzi mówcy. Co więcej baza ta jest całkowicie darmowa dla użytku zarówno komercyjnego oraz niekomercyjnego. Pliki audio nagrane są w nie natywnym angielskim i mają służyć jako zbiór sentencji do oceny wymowy w zróżnicowanych aspektach sygnału mowy. Sam korpus składa się z 5000 nagrań w języku angielskim, a wszyscy z mówców biorących udział w nagraniach są nie natywnymi mówcami języka bazowego zbioru, a ich ojczystym językiem jest Mandaryński. Dodatkowo w bazie, aż połowa nagranych głosów należy do dzieci, a w samej bazie znaleźć można dokładny opis płci oraz wieku. Dodatkowym atutem tej bazy jest pełna lista ręcznych adnotacji obejmujących szeroki zakres szczegółów dotyczących bazy na poziomie zdań, słów czy nawet fonemów. Sama baza została skonstruowana przy współpracy z ekspertami, którzy dokonali obiektywnej oceny (indywidualnie pod tymi samymi warunkami w tej samej metryce) danych wypowiedzi na kilku poziomach. W poniższym przeglądzie poziomów oceny, opisane zostało jedynie pierwsze kryterium, które jest przedmiotem tego projektu. Kolejnyne poziomy oceny to:

- Poziom fonetyczny

  Zadaniem oceny w aspekcie fonetycznym była ocena każdego fonemu wypowiedzi w zakresie punktowym od 0 do 2, gdzie:

  - 2 - wymowa jest poprawna,
  - 1 - wymowa jest poprawna, ale występował ciężki akcent na analizowanym fonemie,
  - 0 - wymowa jest zła lub nietrafna.

- Poziom słów
  - Dokładność
  - Akcent
- Poziom zdań
  - Dokładność
  - Kompletność
  - Prozodia

W strukturze i formie etykietowania bazy stykamy się z przejrzystym systemem zawierającym pliki typu JavaScript Object Notation (JSON) z wymienionymi wyżej ocenami oraz uporządkowaniami, które uwzględniają wiek, płeć i kolejne nagrania mówców. Sama baza rozdziela elementy w odrębnych przestrzeniach dla każdego z mówców, a same nagrania audio operują na formacie Waveform Audio Format (WAV).

Librispeech [18]

Drugą z wykorzystywanych baz danych jest korpus LibriSpeech [18] - zestaw danych bazujący na audiobookach. Zawartość przedstawionego korpusu to około 1000 godzin mowy angielskiej w oryginalnej częstotliwości 16kHz. Zestaw danych został pierwotnie stworzony przez Vasilla Panayotova z asystą Daniela Poveya [18], a dane pozyskane zostały z audiobooków zamieszczonych na stronie projektu LibriVox [5] oraz projektu Gutenberg [6], które wspierają inicjatywę udostępniania tej kategorii zasobów w warunkach domeny publicznej. W przygotowaniu zestaw ten został również odpowiednio sklasyfikowany i uszeregowany, a dane do treningu modeli za pomocą tej bazy zostały rozdzielone na 3 zestawy: 100hr, 360hr oraz 500hr. Istnieje również zestaw łączony 960h. Drugim kryterium podziału nagrań jest skomplikowanie i trudność wypowiedzi pod kątem jakości. W tej kategorii dane zostały podzielone na zbiory zawierające czystą mowę ('clean') oraz takie z dodatkowymi efektami czy zakłóceniami ('other'). Ideą tego podziału jest możliwość przetestowania wytrenowanych modeli w warunkach czystej mowy, a także "bardziej realistycznych", wymagających i uwzględniających elementy wpływające na zrozumiałość mowy. Sam korpus zapewnia również korespondujące teksty wyekstrahowane z odpowiadających książek wymienionych wyżej projektów, gdzie sumarycznie zawierają około 977 tysięcy unikalnych słów, a także niezbędne modele językowe n-gram (ang. Language Models, LM) wykorzystywane do ewaluacji wyników modeli ASR.

Modele językowe n-gram to modele probabilistyczne opierające się na statystykach i służą do przewidywania kolejnego elementu ciągu, gdzie n oznacza ustaloną długość sekwencji. Tworzenie skutecznego systemu bazuje na analizie dużej ilości tekstu zamienianego na prawdopodobieństwa w procesie normalizacji. Taki model pozwala na predykcję kolejnego elementu na podstawie sekwencji n.

W ramach wykorzystywanych bezpośrednio w projekcie pomniejszych baz danych z tego korpusu możemy wyszczególnić zestawy:

- "clean"960h - trening większości modeli ASR,
- "other"960h - trening jednych z modeli na powyższej bazie w celu porównania skuteczności i wpływu danych treningowych na wyniki,
- "other"500h - dodatkowa baza z pozostałymi 500h nagrań wliczającymi się w powyższe bazy (trening na mniejszym zbiorze),
- "clean"360h - zbiór danych wykorzystywany do treningu modeli trifonicznych,
- "clean"100h - zbiór danych wykorzystywany do treningu modelu monofonicznego.

W projekcie inżynierskim z korpusu Librispeech [18] używane są również modele językowe:

- Language Model 4 (LM4) - model językowy 4-gram,
- Language Model 3 (LM3) - model językowy 3-gram,
- Pruned Language Model 3 (PLM3) - lekko zredukowany model językowy 3-gram,
- Small Language Model 3 (SLM3) - mocno zredukowany model językowy 3-gram.

W odniesieniu do struktury bazy nagrań podobnie jak w wypadku pierwszego wykorzystywanego zestawu, każdy z mówców posiada wydzieloną przestrzeń indeksowaną numerycznie oraz rozdzielone podprzestrzenie korespondujące z czytanymi rozdziałami. W bazie znajdziemy również zapisane meta-dane odnoszące się do płci mówców oraz całkowitego materiału audio w bazie, a także informacje o wykorzystanych rozdziałach, książkach z przedstawionych projektów oraz o właściwym czasie trwania elementów składowych.

Wykorzystane narzędzia

W ramach wykonywania projektu inżynierskiego skorzystano z różnego typu oprogramowania oraz środków. Najważniejszymi elementami całego projektu był zestaw narzędzi Kaldi [4] oraz infrastruktura PLGrid [3]. Dzięki tym dwóm składowym praca mogła zostać zrealizowana - zestaw narzędzi zapewnił niezbędny inwentarz do wykorzystania, a program PLGrid [3] pozwolił na przeprowadzenie obliczeń na zaawansowanych klastrach.

W ramach pakietu narzędzi Kaldi [4] wykorzystano narzędzia z dwóch projektów:

- Librispeech [18]
- Speechocean762 [15]

Poszczególne elementy składowe pozwoliły na przetestowanie i wdrożenie działającej wersji algorytmu. Za pomocą bazy Librispeech [18] wykonano trening modelu ASR, a Speechocean762 [15] wykorzystany został w przypadku pracy z algorytmem GoP. Zestaw Kaldi [4] zawierał wiele narzędzi i plików pomocniczych, które zostały wykorzystane w działaniu z dwoma wymienionymi wcześniej zasobami.

Infrastruktura PLGrid [3] umożliwiła dostęp do klastrów Cyfronetu, Akademii Górniczo-Hutniczej (ang. AGH University of Science and Technology, AGH) w Krakowie, gdzie możliwe było wykorzystanie wydajnych i zaawansowanych kart graficznych (ang. Graphics Processing Unit, GPU) - A100, do przeprowadzenia długich i rozległych obliczeń trwających w niektórych przypadkach do kilku dni. Oprócz serwera Athena, udostępniono również klaster Ares, który z kolei pozwalał na przeprowadzenie działań na wielu procesorach (ang. Central Processing Unit, CPU) i zapewniał przestrzeń do przechowywania danych nawet do 8TB. Dostęp do infrastruktury oraz wsparcie w trakcie prac zostały przyznane w ramach grantu obliczeniowego nr. PLG/2023/016645.

W pracy wykorzystywano również podstawowe programy klientów protokołu Secure Shell (SSH), takich jak Putty czy WinSCP. W ramach edycji i obróbki kodu wykorzystywano zintegrowane śrdowisko programistyczne (ang. Integrated Development Environment, IDE) PyCharm (Python) oraz Visual Studio Code (Perl, bash, C++), a także inne mniejsze edytory tekstów i grafiki w celu obróbki danych. Analizę statystyczną i wizualizację danych wykonano w programie MatLab.

2.2.2 Trening sieci neuronowej - ASR

Na sam trening sieci neuronowej składa się wiele kroków obejmujących przygotowanie danych, przetwarzanie plików audio, trenowanie modeli pośrednich, przygotowanie narzędzi użytecznych w procesie czy sam właściwy trening modelu ASR. Ten podrozdział przedstawia opis teoretyczny wszystkich etapów składających się na tworzenie systemu ASR w tej pracy inżynierskiej oraz omówienie elementów składowych wykorzystywanych w tym procesie.

Sam trening sieci neuronowej odpowiedzialnej za rozpoznawanie mowy składa się z wielu etapów - jest to proces długi, nierzadko trwający kilkanaście godzin. W przypadku przyjętego na potrzeby tej pracy schematu działań, na tworzenie modelu ASR składa się ponad 20 kroków obejmujących obróbkę danych audio, pre-processing oraz właściwy trening. Poniżej zebrano te kroki i opisano kolejne etapy w celu uzyskania modelu ASR.

Przygotowanie danych

W pierwszej kolejności pobierane są dane treningowe przedstawione w Podrozdziale *2.2.1 Wykorzystane bazy danych i narzędzia* jako zestawy o określonej liczbie godzin i objętości. Razem z nimi dostarczane są modele językowe wymagane do poprawnego działania i ewaluacji wytrenowanych modeli. Dodatkowo na oficjalnej stronie projektu możemy zapoznać się i nabyć wymagane narzędzia oraz tzw. "chain"model trenowany również na wykorzystywanych w tym projekcie bazach - nie jest on jednak użyteczny pod kątem algorytmu GoP i nie będzie tutaj omawiany. Pierwszorzędnie trenowany jest model na danych o długości 100h, który jest najmniejszym dostępnym zestawem - w późniejszych krokach i iteracjach dodatkowo konfigurowane są większe zestawy w tym wymienione wykorzystywane bazy 360h, 500h i 960h. W ramach przygotowania danych, bazy są odpowiednio etykietowane i segregowane zgodnie z wymogami używanych skryptów oraz lokalizacji - unifikowany jest również system nazewnictwa opierający się na małych literach i na znakach underscore.

Tworzenie słowników

Kolejnym ważnym krokiem w procesie przygotowania danych jest stworzenie słowników powstających na bazie skryptu auto-generującego ciąg fonemów dla określonych słów dostępnych w bazie, a które są mapowaniem ortograficznym i fonetycznym angielskiego słownictwa. Należy nadmienić, iż w obecnej implementacji wykorzystywany jest słownik open-source, języka angielskiego CMUdict, którego w ramach przygotowania leksykonów, używa się jako uzupełniane mapowania słownictwa znajdującego się jednocześnie w danych i słowniku. Zasoby CMUdict są wykorzystywane w celu zaoszczędzenia czasu i zasobów.

Konwrsja formatów danych

Następnie powracając do pozyskanych wcześniej modeli językowych, konwertowane są one z powszechnie przyjętego systemu zapisu ARPA do formatu przetwarzanego i akceptowalnego w zestawie Kaldi [4] - ConstArpaLm. Jest to kompaktowy format przechowywania modeli typu ARPA odznaczający się możliwością lepszej kompresji, a w związku z tym lepszym wykorzystaniem pamięci. Dzięki temu uzyskano możliwość zapisu większych modeli w pamięci oraz przechowywania zestawów danych tabelarycznych.

Ekstrakcja danych i cech

Kontynuując przygotowanie danych tworzone są odpowiednie współczynniki i cechy, które wykorzystywane są w procesie treningu. W wypadku zestawu Librispeech [18] i przyjętych założeń, jest to MFCC oraz średnie i odchylenia standardowe obliczone przy użyciu Cepstral Mean and Variance Normalization (CMVN) [21] na mówcę. Przy bardzo dużym rozmiarze zestawu danych równie pokaźną objętość zajmują wyekstrahowane cechy cepstralne. W związku z tym przy przetwarzaniu MFCC, jednocześnie wykonywany jest podział na foldery, jednostki oraz tworzone są linki symboliczne pozwalające połączyć informacje oraz swobodnie się między nimi poruszać.

Przetwarzanie statystyk CMVN

W przypadku CMVN wykonywane są one dla każdego z mówców, a nie dla każdej z wypowiedzi. Sam proces normalizacji i statystyk CMVN opisany jest w pracy z 1998 roku [21]. Celem takiego zabiegu jest uzyskanie informacji korelujących z konkretnym mówcą i dostosowanie danych do wykorzystywanego w późniejszych krokach Feature space Maximum Likelihood Linear Regression (fMLLR).

Jest to transformacja cech nakierowana na adaptację w kierunku mówcy. O samej transformacji oraz jej innej wersji można przeczytać w publikacji z 2008 roku [20].

Trening modelu monofonicznego

W przygotowaniu do treningu właściwych końcowych modeli, przeprowadzany jest trening modelu monofonicznego. Jest to model akustyczny, który nie uwzględnia żadnych informacji kontekstowych odnoszących się do poprzedzającego lub następującego fonemu innego niż bieżący. Jest to model bazowy użyty dalej do budowy kolejnych modeli trifonicznych, opisanych w kolejnych krokach. W samym treningu wykorzystywane są cechy MFCC oraz ich pierwsze (delta) oraz drugie pochodne (delta-delta), a więcej na ich temat można przeczytać w publikacji [14]. Podczas treningu używana jest normalizacja CMVN, a do treningu wykorzystywany jest przetworzony i przygotowany wcześniej zestaw danych "clean"o długości 100h, który zawiera 29 tys. wypowiedzi.

Trening kolejnych modeli

W kolejnych krokach powtarzana jest procedura przedstawiona w poprzedzających sekcjach, dostosowana do kolejnych bardziej złożonych modeli. Poniżej przedstawiono następujące po sobie treningi systemów, gdzie w nawiasach kwadratowych wskazano nazewnictwo modeli.

1. Model delta + delta-delta trifoniczny [tri1b] - model trenowany jest na podzbiorze 5 tys. nagrań, a jako podstawę wykorzystuje model monofoniczny. Wykorzystywane są już w nim informacje kontekstowe tj. dane o poprzedzającym oraz następującym fonemie, a dodatkowo wykorzystuje pierwszą oraz drugą pochodną MFCC (delta oraz delta-delta). Wykorzystywane są one w przystępny i prosty sposób, by uwzględnić oraz opisać dynamikę sygnału przy małych nakładach obliczeniowych.

2. Model Linear Discriminant Analysis+Maximum Likelihood Linear Transform (LDA+MLLT) [tri2b] - model trenowany jest na podzbiorze 10 tys. nagrań. Jest to kolejna ewolucja poprzednich modeli wykorzystująca Liniową Analizę Dyskryminacyjną (ang. Linear Discriminant Analysis, LDA) oraz Maximum Likelihood Linear Transform (MLLT). LDA ma na celu zbudowanie stanów HMM na bazie wektorów dostępnych cech, ale z wykorzystaniem zredukowanej przestrzeni dla wszystkich danych.

   Kolejno MLLT wykorzystuje zredukowaną przestrzeń cech stworzoną przez LDA i dostarcza unikatowe transformacje dla każdego z mówców. Wynikiem takiego procesu jest zbliżenie się do normalizacji danych dla kolejnych mówców w celu minimalizacji różnic pomiędzy kolejnymi jednostkami. Więcej na temat MLLT można znaleźć w publikacji [9].

3. Model Linear Discriminant Analysis+Maximum Likelihood Linear Transform +Speaker Adaptive Training (LDA+MLLT+SAT) [tri3b] - model trenowany również na podzbiorze 10 tys. nagrań. Wykorzystuje wdrożone w poprzednim kroku zmiany modelowe i rozszerza działanie modelu o Speaker Adaptive Training (SAT). Element ten jest również wykorzystywany do normalizacji na poziomie mówcy oraz ewentualnego pojawiającego się w nagraniach hałasu. Wykonywane jest to poprzez wytrenowanie równoległe mniejszej sieci neuronowej opartej na i-vectorach (tzw. sieć iVecNN), która z użyciem tych wektorów mówców jako danych wejściowych, uzyskuje dane wyjściowe w postaci przesunięć liniowych cech. Same i-vectory to wektory danych reprezentujących fragmenty wypowiedzi za pomocą danych niskiej rozdzielczości uzyskiwanych na drodze analizy czynnikowej, a więcej na ich temat dowiedzieć się można z publikacji [22]. Następnie wykorzystywane są one na wejściu nowej, właściwej sieci neuronowej, co w wyniku takiej operacji tworzy bardziej znormalizowane cechy pod kątem danych mówców.

4. Finalny model tri3b - model z punktu 3 zostaje wytrenowany na całym zestawie 100h danych.

Modelowanie prawdopodobieństw

Kolejnym krokiem w toku postępowania jest wymodelowanie prawdopodobieństw wypowiedzi oraz ciszy dla obecnych danych treningowych oraz przetworzenie folderów ze zbiorem modeli językowych zgodnie z zasadami formatów ARPA oraz ConstArpaLm. Z teorią oraz z samym procesem zapoznać się można w oryginalnej pracy naukowej twórców [10].

Forced Allignment

Następnie powracając do procesu Forced Allignment z pomocą fMLLR wykonywane jest uszeregowanie danych audio i transkrypcji z już znormalizowanymi danymi mówców. Po treningu z SAT, model akustyczny nie jest już trenowany na oryginalnych cechach, ale na znormalizowanych danych pod kątem mówców. W dosłownym tego znaczeniu usuwamy dane odnoszące się do tożsamości mówcy poprzez jej estymację z użyciem odwróconej macierzy fMLLR, usunięciu jej z modelu poprzez wymnożenie z wektorem cech. Powstają w ten sposób modele akustyczne, które w przybliżeniu są niezależne od mówców i mogą być użyte w procesie Forced Allignement.

Trening modelu tri6b

Kondensując kolejne kroki powtarzano niektóre z poprzednich etapów w celu poszerzenia bazy nagrań w formie dostosowanej do przetworzonych już danych. Celem było uzyskanie 960h pełnej bazy Librispeech [18] poprzez złożenie pomniejszych baz o długości 100h, 360h oraz 500h. Pomiędzy iteracjami i kolejnymi zespoleniami wyliczano cechy, przekształcano je oraz wykorzystywano Forced Allignment w celu treningu kolejnych modeli SAT na większych bazach, będącymi podstawą pod kolejne bardziej zaawansowane modele. Finalnie wytrenowano na mieszanej bazie 960h - system tri6b (LDA+MLLT+SAT), będący podstawą właściwego treningu ASR.

Obróka danych wysokiej jakości

Ostatnim z kroków przed właściwymi działaniami z systemem ASR jest obróbka danych wysokiej jakości i rozdzielczości oraz powiązanych z nimi i-vectorami. Od tego momentu bezwzględnie wszystkie operacje są wykonywane na GPU, przez wzgląd na wielkość i długość wykonywanych zadań. W celu uzyskania odpowiednich Wymuszonych Uszeregowań oraz treningu na danych nie idealnych, wykonywana jest perturbacja danych zwana najczęściej z języka angielskiego jako "data augmentation". Wykonywana jest ona w zakresie szybkości wypowiedzi oraz jej głośności w celu rozszerzenia bazy oraz otrzymania niskiej rozdzielczości danych. Podobnie jak dla trenowanych wcześniej modeli również i dla modyfikowanych danych wyliczane są cechy MFCC, CMVN oraz przeprowadzane jest Forced Allignment z fMLLR. Należy podkreślić, iż w wypadku danych niskiej rozdzielczości pozyskiwane MFCC składa się z 40 cech cepstralnych, gdy dla podstawowego procesu liczba ta wynosiła 13.

W przypadku ekstrakcji i-vectorów dodatkowo ekstrahowane są modele Universal Background Models (UBM) [19] oraz wykorzystywany jest algorytm Principal Component Analysis (PCA) w celu redukcji wymiarowości cech oraz danych, gdyż wybierany do treningu i ekstrakcji zestaw danych jest pomniejszony, ponieważ całość Librispeech [18] była by za duża. W związku z tym finalnie na etapie używane jest około 20%, czyli niecałe 200h nagrań. Zawężenie zbioru tych danych jest również związane z czułością i-vectorów na ilość danych. Sama ekstrakcja na perturbowanych danych polega na kombinowaniu krótkich segmentów audio, zgodnie z założeniami projektu, gdyż wymagane są one do analizy fonetycznej. Algorytm paruje wypowiedzi i traktuje je z podobieństwem jako jednego tożsamego mówcę, dzięki czemu na określonym zestawie danych daje to wysoką różnorodność mówców wygenerowaną sztucznie przez perturbowane dane. Celem takiej operacji jest osiągnięcie lepszej generalizacji problemu normalizacji przedstawionego w poprzednich punktach - daje nam to lepszą skuteczność w dekodowaniu wyników na wypowiedź. Ekstrakcja wektorów następuje również dla właściwych zestawów treningowych oraz testowych (walidacyjnych). W efekcie wszystkich przygotowań można przystąpić do właściwego treningu ASR.

2.2.3 Metryka GoP - ocena i przebieg

Po wytrenowaniu odpowiedniego modelu ASR następuje czas na walidację i testy modelu na algorytmie GoP. W porównaniu do obszernego systemu treningu ASR i jego wymagań, podstawowy algorytm GoP wykona swoje zadania na jednostce CPU oraz zamyka się w mniejszej ilości kroków. Jednak wiele z elementów przygotowanych i stworzonych w ramach zadań na modelu ASR zostaje wykorzystanych w fazie z algorytmem GoP. Poniżej przedstawiono kolejne zadania wykonywane w zakresie oceny wymowy:

Wymagania GoP

Do wykorzystania algorytmu GoP w zestawie narzędzi Kaldi [4] wymagany jest:

- model nnet3 opisany w Podrozdziale *3.1.3 Trening modelu* lub inny zgodny z przeprowadzanymi działaniami,
- ekstraktor i-vectorów - wykorzystywany z fazy ASR,
- modele językowe - wykorzystywane z fazy ASR.

W związku z tym w początkowej fazie odpowiednie modele i wymagane dane są łączone i przenoszone do odpowiednich lokalizacji, a podstawowy zestaw danych Speechocean762 [15] jest pobierany i zachowywany.

Przygotowanie danych i propagacja poprzez sieć neuronową

Podobnie jak w przypadku modelu ASR dane są przetwarzane i sortowane. Wyliczane są również cechy MFCC oraz statystyki CMVN, a także przeprowadzane są działania na ekstrakcji i-vectorów z bazy danych. Kolejno dane treningowe przepuszczane są przez model ASR i propagowane przez kolejne warstwy wytrenowanej sieci neuronowej. Na wyjściu otrzymujemy odpowiednie prawdopodobieństwa modelujące stany HMM.

Tworzenie słowników i podział danych

Po propagacji i otrzymaniu danych z warstwy wyjściowej, z pomocą tego samego skryptu, co w przypadku modelu ASR, tworzone są słowniki leksykalno-fonetyczne na bazie drugiego zestawu danych przyporządkowanego algorytmowi GoP i umożliwiający ocenę wyników działania.

Kolejnym krokiem w przetwarzaniu danych wyjściowych jest podzielenie danych z uwzględnieniem mówców oraz przygotowanie transkrypcji na poziomie fonetycznym. W przestawionych rozwiązaniach częściowo wykorzystywane jest kodowanie numeryczne przyporządkowujące określone liczby do symboli. Wykonywane jest to z pomocą skryptów Perl konwertujących symbole na dane typu integer oraz odwrotnie.

Forced Allignment

Z uzyskanymi wartościami z przeprowadzonych działań, kolejno tworzone są techniką Forced Allignment, uszeregowania treningowe z pomocą logarytmicznego prawdopodobieństwa, a dokładnie za pomocą funkcji "log-likelihood", która funkcjonuje jako miara dopasowania danych do modelu. Wyjaśnia jak dobrze parametr θ (estymator) opisuje zebrane dane.

Czyszczenie danych

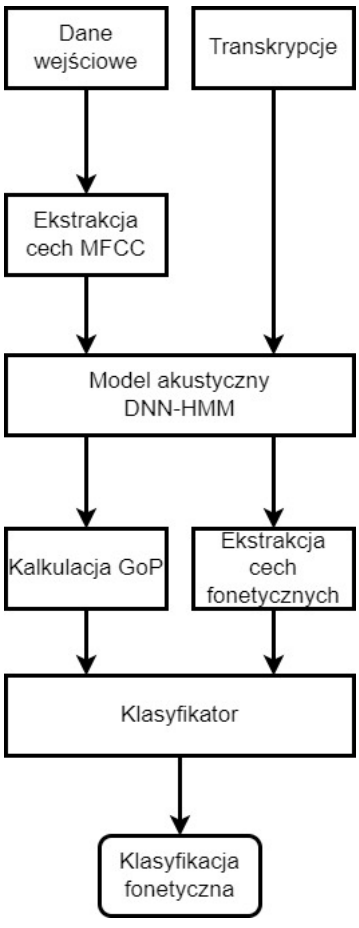
Następnie utworzone słowniki fonetyczne są "oczyszczane"tj. opisy fonetyczne są przetwarzane tak, by usnąć informacje o akcencie oraz markerach Part-Of-Speech (POS), oznaczających słowa jako elementy językowe np. słowo, liczba.

W ostatnim kroku przed oceną GoP, poszczególne uszeregowania na poziomie modelu zostają przekonwertowane do sekwencji fonemów zgodnie z wcześniej wspomnianą nomenklaturą numeryczną.

Ocena GoP

Ostatecznie można przejść do właściwej oceny w algorytmie GoP - w przedstawionym w zestawie Kaldi [4] rozwiązaniu wyliczana jest metryka oraz powiązane z nią cechy w odpowiednim formacie, przedstawiającym indeksowanie fonemów (odnoszące się do plików numerycznych wyjaśniających fonem) oraz ich wartość metryki. Jednakże, by móc ocenić właściwie wypowiedzi w ostatnim kroku następuje wizualizacja oraz trening prostego modelu regresji wielomianowej, który ma za zadanie przekonwertowanie miary GoP opartej na prawdopodobieństwach do prostej i przystępnej oceny w skali od 0 do 2. W ten sposób wyniki interpretowane są na dwa sposoby - jako problem regresji liniowej oraz jako problem klasyfikacyjny, a podejście to opisane jest w Podrozdziale *3.1.4 Ewaluacja modeli i wyników*.

Podsumowując podane w tym rozdziale kroki i działania, przedstawiono graficznie na Rysunku 2 podstawowe kroki w algorytmie GoP.



Rysunek 2: Schemat podstawowych kroków i działań w algorytmie GoP

3 Metoda

Przeprowadzone działania w celu uzyskania finalnego rezultatu skupiały się na implementacji algorytmu GoP oraz treningu i modyfikacji modelu ASR. Wykonano je z pomocą wymienionych wcześniej narzędzi opartych na teorii zawartej w rozdziale *2. Studium literaturowe i wykorzystane dane*, a poniżej przedstawiono zastosowane rozwiązania oraz kolejne etapy wykonanej pracy.

3.1 Zastosowane rozwiązania

W analizie podjętych kroków w trakcie wykonywania zadań należy podkreślić, iż do poprawnego działania skryptów oraz narzędzi wymagany jest system Linux oraz zapewnienie odpowiedniego systemu bibliotek. Warto wspomnieć, iż istnieje możliwość instalacji Kaldiego [4] na systemach Windows jak podano w oficjalnej dokumentacji. Opcja ta jednak nie jest rekomendowana przez wzgląd na pojawiające się problemy oraz słabszą wydajność na tym systemie. Dodatkowo należy podkreślić, iż wykonanie treningu ASR na standardowym sprzęcie domowym może być problematyczne ze względu na poziom skomplikowania obliczeń. W tym wypadku, by odwzorować działania i rezultaty należy skorzystać z jak najbardziej wydajnych kart GPU lub dedykowanej infrastruktury.

3.1.1 Kalkulacja miary GoP

Realizacja lokalna na dostępnym sprzęcie wykonana na samym początku projektu w ramach testów oraz oceny wykonalności projektu nie różni się znacząco od implementacji na klastrach PLGrid [3], dlatego w opisie działań nie będą one rozróżniane.

Jednym z najważniejszych aspektów podczas wykonywania projektu jest kontrola wersji oraz lokalizacji w dostępnych plikach systemowych, dlatego pierwszorzędnie przygotowano odpowiednie zestawy danych i katalogów pod używane narzędzia z tj. przestrzeń roboczą, przestrzeń przechowania danych oraz przestrzeń do analizy przebiegu eksperymentu oraz wyników. W trakcie kolejnych działań tworzono lub sprawdzano automatyczne linki symboliczne pomiędzy systemami plików. Są one nie tylko przydatne w szybkim i sprawnym poruszaniu się po znaczących rozmiarach katalogów, ale również kluczowe w odwołaniach skryptowych używanych narzędzi.

We współpracy z narzędziami oferowanymi przez zestaw Kaldi [4] istotne jest przeanalizowanie toku pracy od głównych plików wykonywalnych, aż do pomniejszych odwołań, gdyż często należy dostosować je do swoich potrzeb oraz zasobów sprzętowych.

Modyfikacji podlegały również same skrypty i kod, gdyż należało je odpowiednio przystosować do wykorzystywanej infrastruktury i założonych celów projektowych. Kontrolowano system plików i odwołań, parametry obciążające zasoby systemowe oraz poszczególne wybrane etapy procesów. Dbano również o proces automatyzacji kolejnych zadań oraz stworzono proste skrypty ewaluujące oraz analizujące i wizualizujące dane. Wiele wykonywanych czynności opierało się również o wykorzystanie szerokiej gamy komend systemowych.

Warto wspomnieć, iż wykorzystywana infrastruktura wykorzystywała kolejkę Simple Linux Utility for Resource Management (SLURM), czyli program open-source do planowania zadań dla jąder Linux i Unix. Jest to ważny aspekt, gdyż było to kolejne narzędzie, którego nauka obsługi stanowiła również część wykonanych prac. W trakcie pracy posługiwano się dwoma trybami zlecanych zadań:

1. interaktywny - tryb pracy zdalnej w czasie rzeczywistym używany do wykonywania zadań bieżących nie wymagających oczekiwania, ale korzystających z GPU,

2. sbatch - tryb pracy zadań zlecanych oczekujących w kolejce, wykonujących obliczenia w długim okresie pracy na zdeklarowanych zasobach obliczeniowych. W związku z powyższym należało również odpowiednio planować działania i rozsądnie testować wprowadzane zmiany, by nie marnować dostępnych zasobów obliczeniowych, a przede wszystkim ograniczonego czasu.

Istotnym z punktu widzenia działania algorytmu GoP elementem jest plik wykonywalny. Działanie kodu w formie teoretycznej zostało omówione w Podrozdziale *"2.2.3 Metryka GoP - ocena i przebieg"*. W przypadku implementacji i wykorzystaniu kodu, a także jego modyfikacji należy pamiętać jednak o odpowiednim dostosowaniu zmiennych, odpowiedzialnych za przechowywanie danych czy linki Uniform Resource Locator (URL) do przesyłu danych, a także parametry odpowiedzialne za wykorzystanie zasobów - przykładowo odpowiedzialne za równoległe wykonywanych zadań. Jest to ważna kwestia z punktu widzenia bezproblemowego przeprowadzenia działań z dobrym oszacowaniem dobranych parametrów i czasu.

Należy również zadbać o zlokalizowanie:

- bazy Librispeech,
- modelu ASR,
- ekstraktora i-vectorów,
- modelu językowego.

Pozostałe dostępne opcje dotyczą dostosowania parametrów poszczególnych etapów np. ekstrakcji MFCC, a w Podrozdziale *"3.2 Problemy i rozwiązania"* zostaną przedstawione najważniejsze z funkcji oraz problemy, które pojawiły się w trakcie opracowania projektu wraz z zastosowanymi rozwiązaniami, a które mogły by pojawić się podczas replikacji przeprowadzonego projektu.

3.1.2 Trening modelu

Trening modelu ASR jest bardziej rozległym procesem pod kątem wykorzystywanych zasobów i wymaganej przestrzeni. W związku z tym niezwykle ważna jest orientacja w systemie plików i lokalizacji. Podobnie jak w przypadku samego algorytmu GoP, należy także dostosować lokalizację do używanej infrastruktury oraz zapewnić odpowiednie linki typu URL do zasobów internetowych. Istotnym elementem jest również kontrola poszczególnych wykonywanych kroków pod kątem parametrów oraz lokalizacji przechowywanych danych, gdyż w przeciwieństwie do sekcji GoP znajduje się tu o wiele więcej indywidualnych parametrów używanych w procesie. W związku z tym dość łatwo przekroczyć wymagania pamięci dotyczące procesu.

Sam wykorzystywany model to nnet3, a w ramach dostępnych i możliwych do wykorzystania architektur dostępne są systemy operujące na klasycznej sieci neuronowej, ale zaliczające się do specyficznej odmiany nazwanej Time Delay Neural Network (TDNN). Modele tego typu wykorzystują wielowarstwowe sztuczne sieci neuronowe opatrzone odpowiednimi danymi i funkcjami służącymi w celu klasyfikacji danych jednowymiarowych lub dwuwymiarowych, których wyraźna segmentacja i podział nie są możliwe lub wymagane. W języku angielskim, takie dane i cel określany są jako "shift-invariance classification". w związku z tym klasyfikacja sygnału mowy czy rozpoznawanie sygnałów akustycznych jest ułatwione, ponieważ nie są wymagane dokładne znaczniki czasowe rozpoczęcia oraz zakończenia występowania danych sygnałów dźwiękowych.

W przeciwieństwie również do standardowych jednostek sieci, każda kolejna warstwa oprócz otrzymania danych z poprzedniej funkcji aktywacji, otrzymuje informacje i dane z serii kontekstowej - w przypadku danych akustycznych będą to kolejne wartości cech wejściowych w czasie. Podsumowując można więc określić sieć tą jako typ konwolucyjnego systemu ze splotem w dziedzinie czasu, dostosowanej do potrzeb baz danych i wymaganych wyjściowych parametrów, które odpowiednio współpracują z algorytmem i skryptami dotyczącymi GoP. Początek sieci opatrzony jest dwoma wejściami, odpowiednio dla składowych i-vectorów (wymiar 1D - 100) oraz dla MFCC (wymiar 1D - 40). W samym zadaniu rozpoznawania mowy i późniejszej ocenie, klasyfikacji danych i próbek wykorzystano metodę LDA. Główną różnicą w wykorzystanych dwóch modelach jest wewnętrzna struktura przetwarzania danych i wykorzystywane warstwy. Opis i różnice zostały przedstawione poniżej:

- System 1B - wariant prosty

  Wariant 1B składa się 5 warstw ukrytych bazujących na funkcji aktywacji Rectified Linear Unit (ReLU) oraz normalizacji wsadowej (ang. batch normalization), odpowiedzialnej za standaryzowanie, przeskalowanie i przesuwanie danych podczas uczenia. Kolejne warstwy sieci są odpowiednio dostosowane do wejść/wyjść warstw poprzedzających oraz następujących.

- System 1C - sieć typu "bottleneck"

  Wariant 1C jest znacznie bardziej rozbudowanym systemem w porównaniu do systemu 1B. Nazwa tej sieci nawiązuje do butelki przez wzgląd, iż ostatnia wyjściowa warstwa posiada znacznie mniejszy wymiar od warstw poprzedzających, a sam przebieg danych przez warstwy ma na celu kompresję i zmniejszenie wymiarowości danych. Podobnie jak pierwszy wariant bazuje na tej samej warstwie wejściowej, czyli ReLU wraz z normalizacją wsadową. Dalej jednak architektura została rozbudowana o 15 warstw ukrytych typu Time Delay Neural Network-Factorized (TDNN-F). Są to warstwy bazujące na faktoryzacji, a wprowadzenie ich zaproponowano w pracy naukowej z 2018 roku [8]. Dodatkowo zastosowano również w poszczególnych warstwach tzw. "time stride", czyli przesunięcie ramek sygnału w dziedzinie czasu.

W treningu modeli ASR wykorzystywana jest funkcja kosztu Binary Cross-Entropy/Log-Loss bazująca na problemie klasyfikacji binarnej. Jest to funkcja mierząca różnicę pomiędzy przewidzianymi rezultatami, a realnymi klasami obiektów i opisuje ją zależność

$$Log\ Loss = \frac{1}{N}\sum_{i=1}^{N} -(y_i \cdot \log(p_i) + (1 - y_i) \cdot \log(1 - p_i)) \tag{7}$$

gdzie N to liczba wszystkich obserwacji, y_i to kolejne rezultaty rzeczywiste, a p_i to kolejne przewidziane rezultaty.

W ramach modyfikacji powyższych modeli oraz procesu treningu modelu ASR wprowadzano szereg zmian mających na celu poprawić wyniki algorytmu GoP, z czego w dużej mierze skupiono się na modyfikacji architektury sieci neuronowej. Różnice w konfiguracjach przedstawiono powyżej, a samą strukturę można modyfikować bezpośrednio w skrypcie zmieniając rozmiary, kolejność, typy oraz wejścia/wyjścia poszczególnych warstw.

W standardowym treningu kontrolowane są również epoki, czyli liczba iteracji przetworzenia całej dostarczonej bazy danych. W przeprowadzonym treningu liczba epok lub wielkości zestawu wykorzystanych danych różniła się w zależności od skomplikowania procesu przekładającego się na długość wykonywanych obliczeń i wymaganych zasobów sprzętowych.

W celu utrzymania optymalnego wykorzystania zasobów i dokładności obliczeń modyfikowano również wielkość podzbiorów treningowych. Im większy zestaw tym większe wymagania obliczeniowe, jednak istnieje większa szansa na lepsze wytrenowanie. Sugerowany rozmiar dla kart GPU o pamięci 80GB RAM (A100) to 512, natomiast w przypadku treningu na multi CPU to 128. W ramach testów sprawdzono wpływ parametru na sieć w zakresie wartości od 256 do 1024, a wartość ta była również dyktowana przez pozostałe parametry.

Kolejną podstawową opcją w ramach treningu sieci neuronowej jest parametr Learning Rate (LR), odpowiadający za tempo i rozmiar kroku w poszukiwaniu minimum funkcji kosztu. Co więcej zestaw narzędzi Kaldi [4] umożliwia użycie postępującego LR wraz z kolejnymi iteracjami treningu. W takiej sytuacji przebieg LR rozpoczyna się od Inital Effective Learning Rate (IELR), co iterację przybliżając się z odpowiednio dobranym krokiem do Final Effective Learning Rate (FELR). Dzięki przedstawionym opcjom, można zoptymalizować proces uczenia, a w trakcie projektu wykorzystano różne konfiguracje wspomnianych ustawień.

W ramach treningu sieci neuronowej można również zdefiniować oraz wykorzystać rozbudowanie architektury sieci poprzez dodawanie warstw z kolejnymi iteracjami treningu. W ten sposób nawet podstawowe proste modele 1B mogą być rozbudowane za pomocą podstawowych warstw do bardziej skomplikowanych i wymagających modeli wraz z progresem treningu. W ramach projektu przetestowano warianty od statycznych modeli bez rozbudowy architektury do wariantów z 2 warstwami na kolejne iteracje.

Kolejną ze zmian, możliwą do wprowadzenia to operowanie na normalizacjach średnich, wariancji lub wariantu łączonego. W ramach przetestowanych modeli uwzględniono opcje bez normalizacji i z normalizacją CMVN w celu zbadania wpływu na wyniki i trening sieci.

W trakcie procesu nauki można wpłynąć także na wymieszanie próbek przy każdej iteracji. Jest to ustawienie opcjonalne, które wprowadza dodatkową losowość do treningu, gdyż próbki są mieszane zawsze na początku treningu bez względu na ingerencję zewnętrzną. Celem takiego zabiegu jest zmienność położenia pojedynczej próbki w różnych Mini-Batchach na kolejne iteracje, gdyż 2 próbki w tym samym zestawie mogą wpłynąć wzajemnie na swoje gradienty. Jest to jednak opcja wymagająca, gdyż przy wysokiej zmienności wzrasta nam zapotrzebowanie na większą pamięć oraz zasoby dysku.

W ramach kontroli wielkości przetwarzanych danych należy zwrócić uwagę na liczbę próbek przyporządkowanych na jedną iterację treningu dla danego zadania, a także na liczbę próbek na pojedynczy proces, które to są wykorzystywane na wyliczanie prawdopodobieństw a priori. Głównym celem manipulowania tymi ustawieniami jest dostosowanie wymagań zasobów obliczeniowych i długość obliczeń - niemniej zbadano również ich istotność na przebieg treningu.

3.1.3 Ewaluacja modeli i wyników

W przetwarzaniu danych oraz ewaluacji modeli i wyników wykorzystano poniższe metody oraz zasoby:

- Dekodowanie - w celu otrzymania wyników i miar wykonuje się dekodowanie danych z wykorzystaniem modeli językowych. W przypadku klasycznego dekodera wykorzystywany jest tzw. 'beam search"oparty na algorytmie Viterbiego opisanego w Podrozdziale *2.1 Definicja i opis algorytmu GoP*. "Beam search"reguluje długość sekwencji analizowanej w algorytmie Viterbiego i bazuje na wyborach bardziej prawdopodobnej hipotezy i najbardziej możliwej sekwencji.

- Porównanie wyników Word Error Rate (WER) - kalkulacja miary (wyjaśnionej w rozdziale *4 Wyniki*) odbywała się z użyciem danych wynikowych dekodowanych, a w ramach implementacji wykonywano porównanie różnych transkrypcji w parach w postaci reprezentacji liczbami całkowitymi lub tekstowej i przeliczano na całkowity ostateczny WER dla modelu.

- Konwertowanie danych - w przypadku narzędzi Kaldi [4] operacje przeprowadzane są na wielu rodzajach danych w tym specyficznych dla wykorzystywanego zestawu. W związku z tym do ewaluacji modelu i jego propagacji przydatne również są metody pozwalające na kopie i konwersję danych. Każdy rodzaj danych posiada swoją odrębną charakterystykę, w związku z tym istnieje szeroka i różnorodna gama dostępnych skryptów. W ramach pomocy i możliwości wglądu w wykorzystane metody, przedstawiono ich listę w *Załączniku 1.*

- Ewaluacja wyników algorytmu GoP - to integralna część algorytmu i rozważana jest w kategoriach problemu klasyfikacyjnego oraz jako zagadnienie regresji liniowej. W ramach oceny klasyfikacyjnej wyniki porównywane są z wynikami eksperckimi i odpowiednio klasyfikowane jako poprawne lub niepoprawne klasyfikacje. Kolejno wyliczane są miary Precision, Recall, F1-Score oraz Accuracy. W ramach regresji liniowej wyliczane są natomiast błędy średniokwadratowe dla obu zbiorów - testowego oraz treningowego.

- Analiza i Wizualizacja danych - w ramach opracowania wyników końcowych zmodyfikowano lub utworzono od podstaw wymagane skrypty kompilujące i porządkujące dane, a także wizualizujące je w odpowiedni sposób z wykorzystaniem oprogramowania MatLab oraz arkuszy kalkulacyjnych. We wszystkich możliwych wypadkach zadbano o automatyzację procesu oraz rozszerzono i utworzono wywodzące się oryginalnych skryptów wersje kodu dostosowane do testowanych modeli oraz równoległości wykonywanych zadań.

W samej metodyce ewaluacji wyniki treningowe ASR porównywano do modeli referencyjnych. Są to wyniki uzyskane za pomocą modeli z domyślnymi parametrami zdefiniowanymi w źródłowych skryptach Kaldi [4]. W Tabeli 1 przedstawiono różnice w użytych parametrach w modelach referencyjnych oraz zakres parametrów użytych w projekcie.

Tabela 1: Porównanie zakresów parametrów modeli trenowanych z modelami referencyjnymi (liczba epok < 1 wskazuje na użycie mniejszych zestawów danych niż domyślne)

Parametr \Model	Modele Referencyjne	Modele Trenowane
Architektura	1B / 1C	1B / 1C
Epoki	4	0,2 - 3
Mini Batch	256	256 - 1024
IELR	0,0017	0,0017 - 0,045
FELR	0,00017	0,00001 - 0,00017
Normalizacja	Brak	Brak / CMVN
Rozbudowanie sieci	2	Brak / 2
Losowość danych	5000	150 - 5000
Próbki na iterację	400000	400000
Próbki a priori	20000	1250 - 20000

3.2 Problemy i rozwiązania

W trakcie wykonywania pracy inżynierskiej znaczącą składową w przypadku tak skomplikowanych i rozbudowanych narzędzi jest rozwiązywanie problemów i pojawiających się błędów. Dużą część czasu przeznaczono na te czynności, dlatego stworzono dodatkową sekcję mającą na celu udokumentowanie napotkanych problemów i użytych rozwiązań - również w celu pomocy i zaadresowania ich do osób replikujących tą pracę lub pracujących z zestawem narzędzi Kaldi [4].

Niekompatybilne modele

W trakcie opracowywania projektu pierwotnie zdecydowano się przetestować również pre-trenowane modele z zestawu Kaldi [4]. W ramach dostępnych modeli można wykorzystać tzw. 'chain' model, który również jest systemem wywodzącym się od nnet3, jednak różniącego się w założeniach, celach i architekturze.

Należy podkreślić, iż nie jest on kompatybilny z przeprowadzanymi obliczeniami GoP, pomimo faktu, iż są to również modele Deep Neural Network-Hidden Markov Model (DNN-HMM). Co więcej należy również przyłożyć szczególną uwagę do samego procesu treningu, gdyż domyślnym modelem w repozytorium Kaldi [4] jest właśnie nnet3-chain. W związku z tym należy odpowiednio przygotować wstępne skrypty oraz ustawienia pod trening na nnet3-cleaned.

Instalacja Kaldi i kompilacja pod GPU

W trakcie instalacji i kompilacji Kaldi'ego należy zdefiniować kompilację na CPU lub GPU. W wypadku operowania jedynie na algorytmie GoP wystarczy wersja na CPU. W przypadku trenowania modelu ASR wymagana jest wersja z GPU. Należy zadbać również o odpowiednie wersje Compute Unified Device Architecture (CUDA) oraz narzędzi kompilacyjnych Nvidia CUDA Compiler (NVCC). Wagę należy przyłożyć również do zawartości zestawu Kaldi [4] przy instalacji, gdyż niezwykle ważne są linki łączące poszczególne sekcje i elementy. W przypadku złej instalacji lub kopiowania plików z serwisu GitHub mogą pojawić się błędy z kompilacją lub przetworzeniem danych, a także wykrywaniem dostępnych kart graficznych. W przypadku problemów pierwszym sugerowanym krokiem jest reinstalacja lub w przypadku ryzyka usunięcia ważnych danych, prześledzenie ścieżki pracy i połączeń poszczególnych skryptów.

Kontrola zasobów

W wypadku wielu skryptów na poziomie zestawu Kaldi domyślne skrypty zakładają dużą ilość zlecanych zadań oraz równoległości przeprowadzanych działań w celu przyśpieszenia wykonania całego procesu. Zaleca się więc prześledzenie skryptów i zaniżenie parametrów odpowiedzialnych za te ustawienia lub dostosowanie wymagań sprzętowych. Jest to ważna konfiguracja patrząc z perspektywy klastrów zdalnych - duże zasoby mogą spowodować przedłużenie oczekiwania na zasoby oraz spotykanie się z częstymi błędami Out-of-memory (OOM).

W związku z kontrolą zasobów należy uważać również na parametry treningowe, często indywidualne dla niektórych z procesów. Wiele z nich potrafi zaabsorbować pokaźne zasoby pamięci, które z łatwością mogą przysporzyć problemów z obsługą sprzętu. W tym aspekcie zalecane jest przeprowadzanie kolejnych kroków w mniejszych zadaniach kontrolując przetwarzane dane i parametry.

Systemy operacyjne

W przypadku lokalnej realizacji zagadnienia i instalacji narzędzi Kaldi pojawia się wymóg korzystania ze środowiska Linux w celu pełnego wykorzystania potencjału zestawu, przez wzgląd, iż systemy Windows nie wspierają zbyt dobrze prezentowanych rozwiązań. Alternatywą dla urządzenia z wymaganym systemem, może być maszyna wirtualna bądź technologia Windows Subsystem for Linux 2 (WSL2). Jednak w obu wymienionych przypadkach występują problemy z płynnością lub procesowaniem danych audio. W takim wypadku polecanym rozwiązaniem jest skorzystanie z infrastruktury udostępniającej zewnętrzne serwery ze środowiskiem Linux lub Ubuntu, a w przypadku braku takiej możliwości skorzystanie z drugiego systemu zainstalowanego z wykorzystaniem interfejsu Unified Extensible Firmware Interface (UEFI).

Ekstrakcja/kopiowanie danych

W niektórych sytuacjach może nastąpić również problem z ekstrakcją/kopiowaniem i odczytywaniem cech oraz danych za pomocą wymienionych wcześniej narzędzi lub w trakcie wykonywania skryptów. Powodem takiego błędu może być przede wszystkim niekompatybilny model, który zwraca nieodpowiedni format wyników - w takim wypadku należy zweryfikować wykorzystywany model. Drugim powodem takiego błędu są niepożądane zmiany w skryptach ekstrakcji danych lub ich obróbki. W niektórych przypadkach kod może wygenerować błędy (np. za małej pamięci) i kontynuować działanie. W takim wypadku może dojść do wybrakowania danych lub ich uszkodzenia. w sytuacji sugerującej takie problemy należy przeanalizować komunikaty konsolowe i powtórzyć części procesu, które mogą być przyczyną takiego zachowania, a także porównać modyfikacje skryptów względem kodu źródłowego w repozytorium Kaldi [4].

Dekodowanie - przykład nakładania się nazw oraz wykorzystania procesów GPU

W wielu wypadkach należy również uważać na łudząco podobne skrypty w różnych lokalizacjach Kaldi'ego. Przykładowo ***decode.sh*** pojawia się co najmniej trzykrotnie w różnych miejscach repozytorium. Należy jednak uważać na wykorzystywane wersje, gdyż kod kryjący się we wnętrzu pliku spełnia zupełnie inne zadania pomimo wspólnej nazwy. Może to doprowadzić do wielu pomyłek, błędów czy nawet uszkodzenia danych. Często rozwiązaniem jest porównanie wersji z repozytorium lub odświeżenie lokalnego, wykorzystywanej wersji narzędzi.

Dodatkowo należy zwracać uwagę na wymagania skryptów dotyczące zasobów. Wiele z nich można wydzielić do przeprowadzenia działań na CPU w celu zaoszczędzenia czasu, a nawet pominięcia błędów powodowanych brakiem wyłączności pamięci GPU dla danego procesu.

Dublowanie danych

W większości przypadków skrypty i zestaw narzędzi Kaldi [4] jasno komunikuje wymagania dotyczące poprawy błędów. W niektórych wypadkach jednak pojawiające się dublujące lub po prostu nieukończone procesy danych mogą blokować działanie skryptów. Stąd niezmiernie ważnym elementem eksperymentów jest manualna kontrola nad przetwarzaniem danych i czyszczenie ręczne niepotrzebnych folderów i plików.

4 Wyniki

W ramach analizy wyników oraz wyciągnięcia wniosków dotyczących wykonanej pracy zweryfikowano otrzymane obliczenia i modele. Ze względu na przeprowadzone rozległe działania wybrano z przetestowanych blisko 20 konfiguracji parametrów i architektur sieci neuronowych cztery najskuteczniejsze modele przedstawione w Podrozdziale *4.1 Najlepsze modele*. Mianem najlepszych modeli określono systemy, które uzyskały najwyższe i najlepsze wyniki skuteczności na algorytmie GoP, gdyż był to główny cel przedstawionej pracy. Przedstawiono je i przeanalizowano w Podrozdziale *4.2 Wyniki algorytmu GoP*, gdzie w pierwszej części porównano wyniki algorytmu GoP w kontekście problemu regresji liniowej, a w drugiej dotyczącej miar klasyfikacji.

Przeanalizowano również wyniki wybranych modeli pod kątem skuteczności rozpoznawania mowy. Działania te polegały na zestawieniu i przeanalizowaniu wyników wybranych modeli oraz porównania ich z rezultatami referencyjnych modeli, zapewnionych przez zespół pracujący nad narzędziami Kaldi [4], uzyskanych na parametrach przyjętych jako domyślne początkowe parametry treningów i walidacji. Przedstawione wyniki zweryfikowano za pomocą metryki Word Error Rate (WER) i dotyczą modeli ASR na bazie Librispeech [18].

Miara WER dla systemów rozpoznawania mowy zdefiniowana jest jako

$$WER = \frac{S + I + D}{A} \tag{8}$$

gdzie S to liczba podmienionych słów (źle odczytanych), I to liczba wstawionych dodatkowo słów, D to liczba usuniętych słów, a A to wszystkie słowa pojawiające się w wypowiedzi.

Wyniki związane z rozpoznawaniem mowy weryfikowane za pomocą miary przedstawionej w równaniu (8) znajdują się w Podrozdziale *4.3 Wyniki modeli - WER*. W podanej sekcji, by ocenić również sam proces treningu oraz uzyskane wyniki, zestawiono w formie graficznej od Rysunku 3 do 8, dane dotyczące przebiegu uczenia modeli ASR. Obserwacje i wnioski dotyczące przebiegu treningu mogą uprościć i pomóc w dalszym procesie tworzenia lepszych modeli oraz pomóc porównać proces nauki dobranych systemów. Oprócz indywidualnej oceny treningu i wyników modeli ASR, wykonano również ocenę wpływu skuteczności modeli na wyniki algorytmu GoP, by zidentyfikować odpowiednie zależności pomiędzy dwoma głównymi składowymi pracy inżynierskiej. Przeanalizowano w jaki sposób odpowiednie parametry wpływają na cały proces oraz jak efektywniej i lepiej można wykorzystać dostępne narzędzia.

Finalnie porównano najlepsze wyniki w celu uzyskania optymalnego i odpowiedniego pod przedstawione zagadnienie sposobu i podejścia do uzyskania dobrych wyników, a także obrania odpowiedniego kierunku w celu ulepszenia przedstawionego algorytmu.

4.1 Najlepsze modele

Podczas wyboru najlepszych wyników kierowano się rezultatami algorytmu GoP, nie uwzględniając wyników modeli ASR, gdyż nie jest to cel tej pracy. Uwzględniono natomiast wpływ treningu modeli i poszczególnych kroków nauki na przedstawione wyniki GoP. W Tabeli 2 przedstawiono zestawienie parametrów i architektur najlepszych modeli uzyskanych w drodze treningu, uzyskujących najwyższe wartości miar klasyfikacyjnych algorytmu GoP.

Tabela 2: Porównanie modeli uzyskujących najlepsze wyniki na algorytmie GoP

Parametr \Model	1C (FNorm)	1C (F)	1C (S)	1B (Basic)
Architektura	1C	1C	1C	1B
Epoki	0,5	0,5	0,5	3
Mini Batch	1024	1024	512	256
IELR	0,002	0,002	0,0017	0,0017
FELR	0,00005	0,00005	0,00017	0,00017
Normalizacja	CMVN	Brak	Brak	Brak
Rozbudowanie sieci	2	2	2	2
Losowość danych	5000	5000	365	1500
Próbki na iterację	400000	400000	400000	400000
Próbki a priori	20000	20000	3000	12000

Wybrane modele różnią się znacząco konfiguracjami, co zostanie omówione w poniższych podrozdziałach pod kątem zarówno modelu ASR, jak i algorytmu GoP. Uwzględnione zostaną również pozostałe testowane modele nie wyszczególnione w tym zestawieniu w celu uzyskania dodatkowych wniosków i obserwacji, które mogą pomóc w zaobserwowaniu zależności treningowych i algorytmicznych. W przypadku wybranych modeli użyta zostanie nomenklatura podana w Tabeli 2.

4.2 Wyniki algorytmu GoP

W analizie wyników GoP wykorzystano 5 miar przedstawionych poniżej wraz z wyjaśnieniami:

- Accuracy - miara dokładności klasyfikacji zdefiniowana jako

$$Accuraccy = \frac{Poprawne\ klasyfikacje}{Wszystkie\ klasyfikacje}. \tag{9}$$

- Precision - miara precyzji mówiąca o ilości dobrych przyporządkowań danej klasy w odniesieniu do wszystkich pozytywnych klasyfikacji, wyrażona jako

$$Precision = \frac{Poprawne\ klasyfikacje\ analizowanej\ grupy}{Wszystkie\ poprawne\ klasyfikacje}. \tag{10}$$

- Recall - miara mówiąca, jak dużo z elementów klasy zostało odpowiednio przyporządkowane. Parametr Recall opisany jest jako

$$Recall = \frac{Poprawne\ klasyfikacje\ analizowanej\ grupy}{Wszystkie\ klasyfikacje\ analizowanej\ grupy}. \tag{11}$$

- F1-Score - miara zestawiająca parametry "precision"oraz "recall"w sposób narzucający kary za niskie wartości któregokolwiek z parametrów, wyrażona jako

$$F1 - Score = 2 \cdot \frac{Recall \cdot Precison}{Recall + Precision}. \tag{12}$$

- Błąd średniokwadratowy (ang. Mean Square Error, MSE) - obrazujący średnią błędów będących różnicami pomiędzy wartościami oczekiwanymi/przewidywanymi Y_i, a wartościami rzeczywistymi/obserwacjami Y_j w sekwencji n obserwacji. Miara MSE sformułowana jest jako

$$MSE = \frac{1}{n} \cdot \sum_{i=1}^{n} (Y_i - Y_j)^2. \tag{13}$$

Po zdefiniowaniu odpowiednich miar można przedstawić wyniki GoP wybranych modeli. Pierwsza część wyników opiera się na analizie problemu w kategorii regresji liniowej i ocenie MSE obliczonego zgodnie z równaniem (13), a także na interpretacji parametru Accuracy przedstawionego w równaniu (9), odnoszącego się do całego zbioru danych.

Tabela 3: Wyniki najlepszych modeli dla algorytmu GoP na danych testowych

Model	Accuracy	MSE
1B (Basic)	0,43	0,65
1C (S)	0,50	0,57
1C (F)	0,53	0,54
1C (FNorm)	**0,55**	**0,53**

Wyniki dla danych testowych przedstawione w Tabeli 3 pokazują, iż parametr Accuracy uzyskuje maksymalnie wartość 0,55, a MSE zredukowane zostaje do wartości 0,53. W porównaniu do najsłabszych z przedstawionych wyników osiągniętych w ramach modelu 1B (Basic), można zauważyć, iż wprowadzone zmiany poprawiły skuteczność algorytmu. Odwołując się do konfiguracji modeli przedstawionych w Tabeli 2, widać, że najistotniejsza okazała się zmiana architektury modelu z 1B na 1C, wprowadzająca poprawę Accuracy o 0,07 oraz zmniejszenie błędu średniokwadratowego o 0,08. Kolejne zmiany parametrów i wprowadzenie normalizacji CMVN, wpływały coraz słabiej na uzyskiwane rezultaty. W związku z tym, najpewniejszym krokiem w celu dalszej znaczącej poprawy skuteczności GoP, jest wprowadzenie i testowanie dalszych zmian w architekturze sieci neuronowej. W ramach tego procesu można rozbudować i przekształcić używaną w tej pracy inżynierskiej sieć nnet3 lub dostosować i wykorzystać nowsze rozwiązania w dziedzinie modeli ASR. Drugie z przedstawionych rozwiązań wiąże się z wymogiem wprowadzenia zmian do sieci neuronowej, które pozwolą dostosować ją do algorytmu GoP. Porównując jednak wyniki modelu nnet3 do obecnie wykorzystywanych modeli ASR [2] jest to pożądany krok mogący znacząco poprawić wyniki GoP.

Tabela 4: Wyniki najlepszych modeli dla algorytmu GoP na danych testowych - parametry klasyfikacyjne

Parametr	Precision			Recall			F1-Score		
Model \Klasa	0	1	2	0	1	2	0	1	2
1B (Basic)	0,29	0,05	0,99	0,42	0,74	0,41	0,34	0,09	0,58
1C (S)	0,27	0,06	0,99	0,39	0,72	0,50	0,32	0,11	0,66
1C (F)	0,26	0,06	0,98	0,37	0,69	0,53	0,31	0,11	0,69
1C (FNorm)	**0,28**	**0,06**	**0,98**	**0,41**	**0,67**	**0,54**	**0,33**	**0,11**	**0,70**

W celu zmierzenia skuteczności klasyfikacji wykorzystano miary opisane w Podrozdziale *3.1.4 Ewaluacja modeli i wyników* wykorzystując zbiór Speechocean762 [15]. W Tabeli 4 przedstawione są wyniki miar klasyfikacyjnych na danych testowych. W zestawieniu widać, iż rezultaty osiągają szeroki zakres wartości od 0,05, aż do 0,99.

Przyglądając się parametrowi Precision widać, iż znacząco niższe wyniki osiągnęły klasy 0 i 1 w porównaniu do klasy 2. W związku z tym można zauważyć, że ze wszystkich poprawnych klasyfikacji, najłatwiejsze do przyporządkowania dla modeli były bezbłędne wypowiedzi. Problemem jest więc rozpoznawanie błędów, a w szczególności stanów pośrednich, czyli wypowiedzi z błędami akcentowymi, co sugeruje najsłabszy wynik klasy 1. W tym wypadku należy jednak zauważyć, iż nie koniecznie musi być to wina modeli, szczególnie zwracając uwagę na zbieżność wyników pomiędzy kolejnymi modelami.

Winowajcą takich rezultatów mogą być progi klasyfikacyjne, które za bardzo wyczulone na dobrą wymowę, źle sprawują się w warunkach klasyfikacji błędów. Ręczna modyfikacja progów lub wprowadzenie dodatkowych zmian w przeliczaniu wartości GoP w celu otrzymania przyporządkowania do grup od 0 do 2, mogła by usprawnić i poprawić wyniki algorytmu.

W przypadku danych związanych z Recall, który przedstawia procent poprawnie sklasyfikowanych obiektów w obrębie danej klasy, mamy do czynienia z wynikami bardziej zróżnicowanymi. Rezultaty w Tabeli 4 wskazują, iż różnice pomiędzy modelami przedstawionymi w Podrozdziale *4.1 Najlepsze modele* wpłynęły na uzyskane wyniki. Zwiększenie losowości danych w treningu poprawiło wydajność klasyfikacji w grupie poprawnej wymowy (klasa 2), kosztem delikatnej regresji w pozostałych. Zastosowanie normalizacji CMVN wyrównało niemalże wyniki klasy 0 dla modelu 1C (FNorm) do wyników modelu 1B, kosztem ponownego obniżenia efektywności dla klasy pośredniej (klasa 1).

W związku z tym wyłaniają się co najmniej trzy przedstawione elementy z którymi dalsze eksperymenty mogły by pomóc w uzyskaniu lepszych rezultatów, z zastrzeżeniem, iż w przeciwieństwie do miar Accuracy i MSE wprowadzone zmiany muszą być dostosowane do pożądanego efektu i celu działań. Konkretnie, w zależności od celu projektu np. detekcji błędów - musimy wybrać i przetestować zmiany, które poprawią skuteczność w najistotniejszych dla założeń projektu grupach, w tym przykładowym zastosowaniu - w klasach 0 i 1.

Ostatnim z analizowanych parametrów jest F1-Score uwzględniający oba powyższe parametry, dodatkowo penalizując jakąkolwiek niską wartość składowych elementów - Precision lub Recall. Widać to wyraźnie w przypadku klasy 1 - wysoka wartość Recall, przy niskiej Precision, powoduje niskie wartości F1-Score. Odwołując się więc do definicji tych parametrów przedstawionych na początku tego podrozdziału, można zinterpretować F1-Score jako miarę zbalansowania wyników klasyfikacji pomiędzy grupą, a całym zbiorem. Ujawnia to problem niezbalansowania liczebności klas w użytym zbiorze Speechocean762 [15]. Na podstawie wyników klasy 1 można zauważyć, iż musiała być to klasa mniej liczna, gdyż mała ilość poprawnych klasyfikacji w małej grupie może dać wysoki wynik Recall oscylujący w okolicy 0,70. Jednocześnie ta sama mała liczba poprawnych klasyfikacji w przełożeniu na znacznie większy zbiór danych, skutkować będzie niskimi wartościami Precision. Potwierdzić można to analizując etykiety przyporządkowane w bazie Speechocean762 [15] jako rzeczywiste klasy. Na ich podstawie można zaobserwować, iż klasa 2 jest liczniejsza od pozostałych.

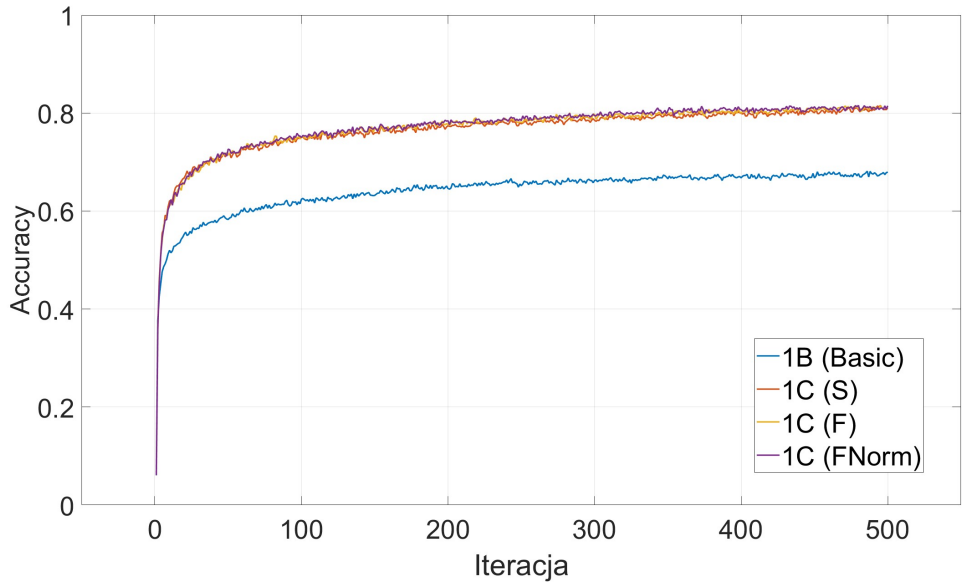
W ramach powyższych obserwacji możemy uznać wyniki za zadowalające, uwzględniając cel pracy. Wyciągnięte wnioski i obserwacje pozostawiają szeroki zakres propozycji działań i eksperymentów mogących poprawić skuteczność obecnego algorytmu GoP.

4.3 Wyniki treningu ASR

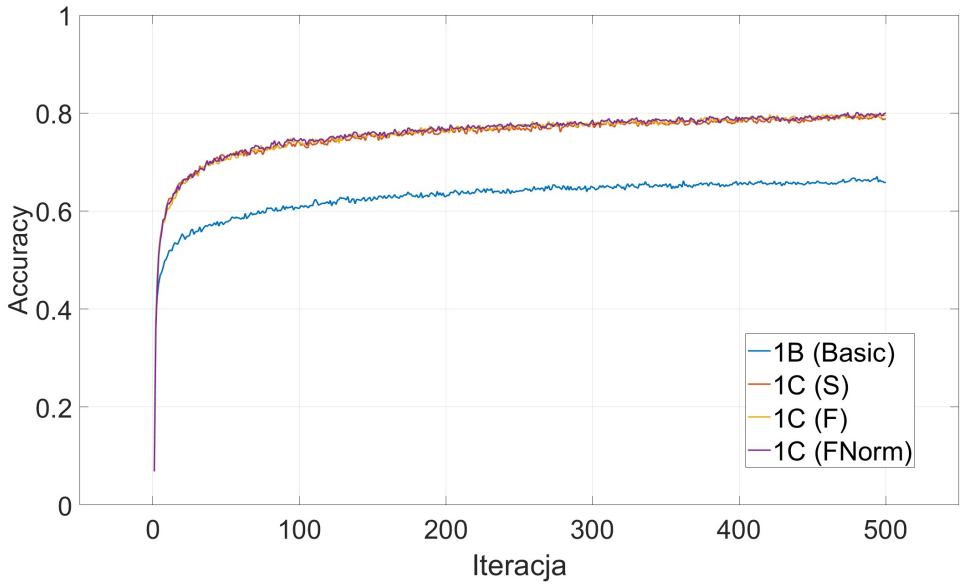
W celu analizy procesu treningu wykorzystano miarę Accuracy dostosowaną do modeli ASR opisaną jako

$$Accuraccy\ (ASR) = \frac{Poprawnie\ rozpoznane\ elementy\ mowy}{Wszystkie\ elementy\ mowy}. \tag{14}$$

Jako referencję wykorzystano transkrypcje z książek opracowywanych w korpusie Librispeech[18], a więc z projektu LirbiVox[5] oraz Project Gutenberg[6] - jako poprawne odwzorowanie mowy. W związku z tym parametr ten wskazuje jaki procent przełożonych przez ASR słów zgadza się z referencyjną transkrypcją. Na Rysunkach od 3 do 8 przedstawiono na wykresie zmiany Accuracy (ASR) oraz funkcji kosztu przedstawionej w Podrozdziale *3.1.3 Trening modelu*, wraz z treningiem, czterech wybranych modeli w zakresie 500 iteracji.

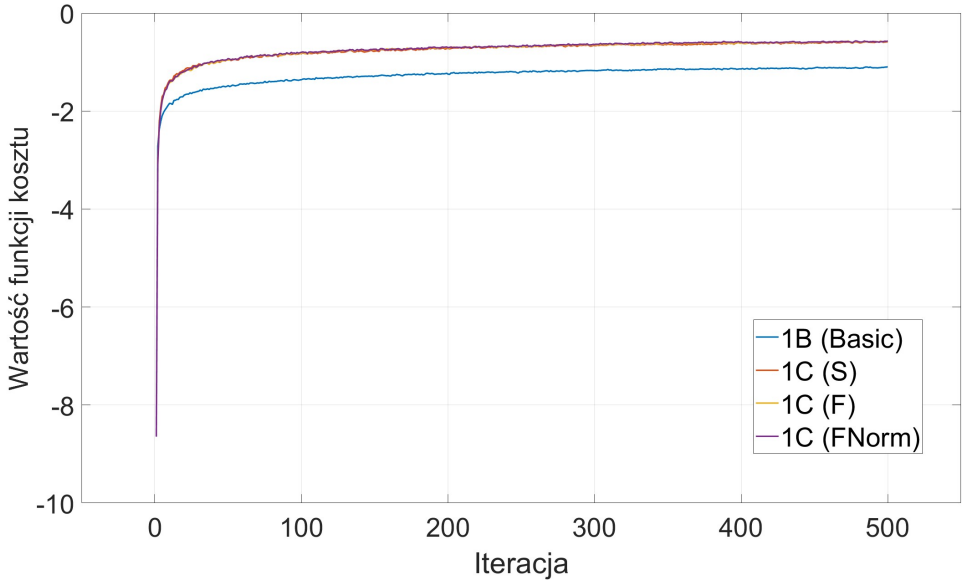


Rysunek 3: Porównanie przebiegu treningu wybranych modeli - Accuracy (ASR) na danych treningowych

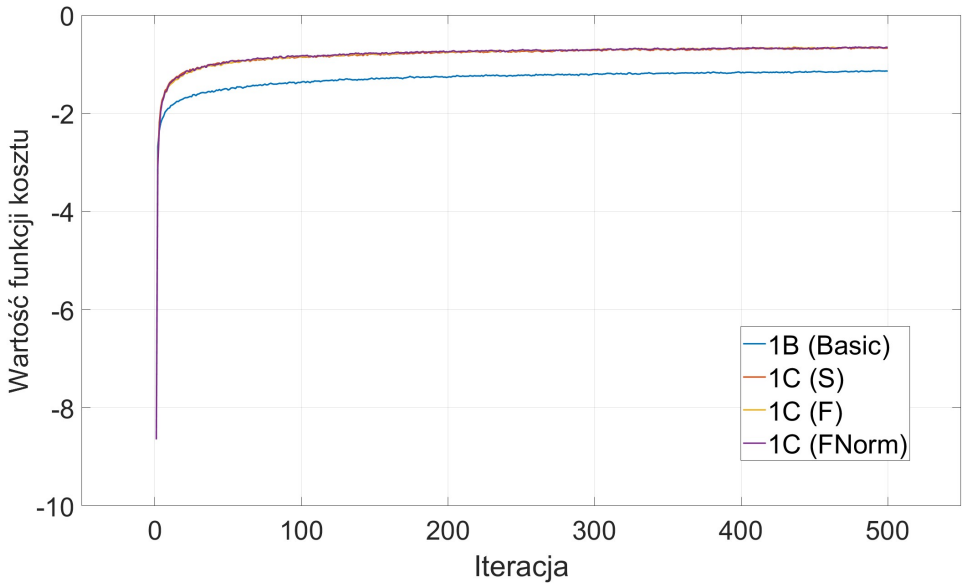


Rysunek 4: Porównanie przebiegu treningu wybranych modeli - Accuracy (ASR) na danych testowych

Przedstawione dane na Rysunku 3 oraz 4 pokazują, iż modele 1C okazały się skuteczniejsze i w przedstawionej liczbie iteracji osiągnęły blisko 80% Accuracy (ASR), gdzie model 1B (Basic) osiągnął niecałe 65%. W przypadku danych testowych sytuacja jest podobna ze spodziewaną lekką regresją wartości Accuracy (ASR), przez wzgląd na pojawiające się nowe i trudniejsze dane w zbiorze testowym. Warto jednak podkreślić, iż dotrenowanie modeli 1B jest nieskuteczne, natomiast modele 1C mogły by zostać dotrenowane lub przetrenowane na szerszej bazie danych w celu osiągnięcia jeszcze lepszych wyników, co zostało uzasadnione w Podrozdziale *4.5 Wyniki modeli - WER*.

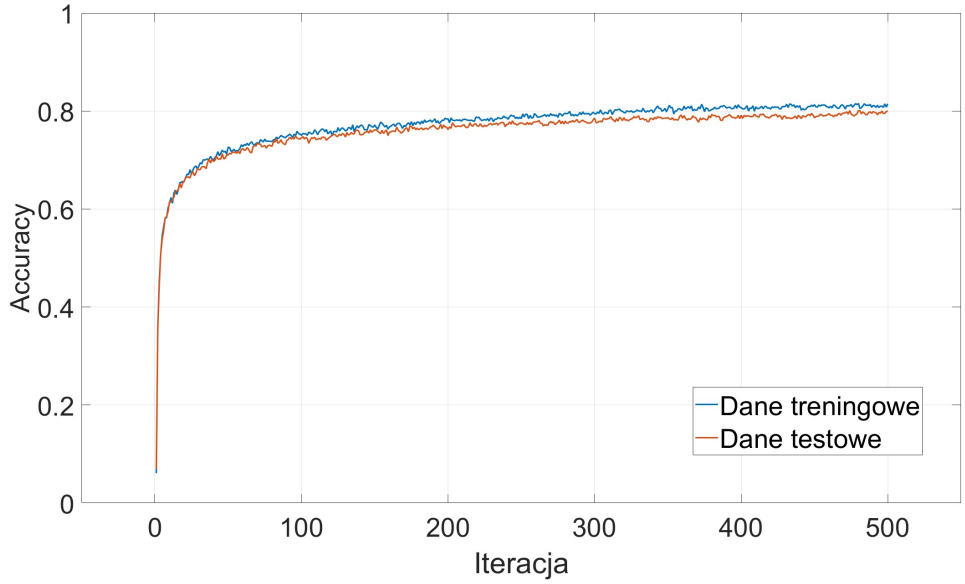


Rysunek 5: Porównanie przebiegu treningu wybranych modeli - funkcja kosztu na danych treningowych

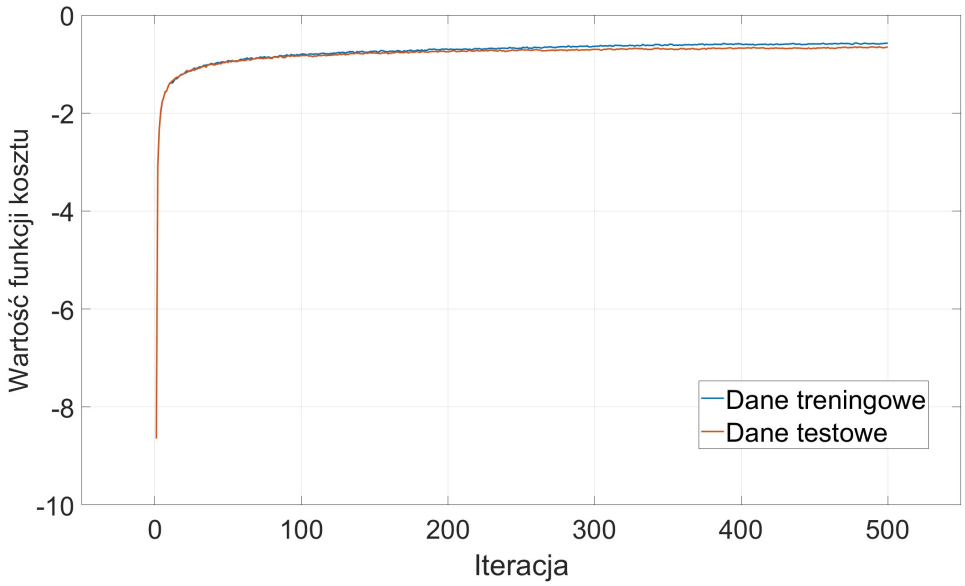


Rysunek 6: Porównanie przebiegu treningu wybranych modeli - funkcja kosztu na danych testowych

W przypadku funkcji kosztu przedstawionej na Rysunku 5 oraz 6, można zaobserwować, że faza największych zmian przypada na pierwsze 100 iteracji treningu. Kolejne iteracje to stabilizacja i bardzo znikome zmiany, sugerujące skrócenie lub zmodyfikowanie procesu treningu. Należy jednak uwzględnić, iż we wszystkich modelach użyto progresywnego LR, który mógł wpłynąć znacząco na wyrównanie i zmniejszenie gradientu wartości w końcowych fazach treningu.



Rysunek 7: Porównanie przebiegu treningu modelu 1C (FNorm) - Accuracy (ASR)



Rysunek 8: Porównanie przebiegu treningu modelu 1C (FNorm) - funkcja kosztu

Dla porównania przebiegu treningu i walidacji zestawiono na Rysunku 7 oraz 8, Accuracy (ASR) oraz funkcję kosztu dla najlepszego modelu 1C (FNorm). Wyniki treningowe są odpowiednie nieco wyższe, co jest spodziewane przy walidacji na nowym, bardziej wymagającym zbiorze danych. Widać również ten efekt na przebiegu funkcji kosztu. Co ważne możemy zaobserwować, iż wyniki na obu zestawach są bardzo zbliżone do siebie. Daje to pewność, iż modele nie zostały przetrenowane, gdyż w przeciwnym wypadku, jeżeli krzywe postępu treningu znacząco odbiegałyby od siebie, należałoby zbadać taką ewentualność.

4.4 Wyniki modeli referencyjnych ASR

W początkowej fazie przeanalizowano architekturę oraz wyniki systemów referencyjnych.

Tabela 5: Wyniki WER [%] dla modeli referencyjnych - dane treningowe

Dane	dev				dev_other			
Model \Model językowy	LM4	LM3	PLM3	SLM3	LM4	LM3	PLM3	SLM3
1C (Referencyjny)	4,20	4,37	5,31	5,86	12,55	13,00	14,98	15,88
1B (Referencyjny)	4,52	4,80	6,02	6,80	12,54	13,16	15,51	17,12

Tabela 6: Wyniki WER [%] dla modeli referencyjnych - dane testowe

Dane	test				test_other			
Model \Model językowy	LM4	LM3	PLM3	SLM3	LM4	LM3	PLM3	SLM3
1C (Referencyjny)	4,91	4,99	5,93	6,49	12,94	13,38	15,11	16,28
1B (Referencyjny)	5,00	5,22	6,40	7,14	12,56	13,04	15,58	16,88

W Tabeli 5 i 6 przestawiono wyniki modeli referencyjnych ASR w postaci WER dla dwóch wykorzystanych systemów. W podanych danych uzyskanych z pomocą modeli językowych przedstawionych w Podrozdziale *2.2.1 Wykorzystane bazy danych i narzędzia* widać, iż model referencyjny 1C sprawuje się lepiej niż model referencyjny 1B, jednak różnica wyników WER nie jest znaczna, gdyż maksymalna różnica to około 2 punkty. W obecnych czasach nie są to wyniki zaliczające się do czołówki rynku, a wiele nowszych modeli ASR osiąga na tych zbiorach niższy WER na poziomie 3% [1]. Lepsze rezultaty również w rozpoznawaniu mowy w czasie rzeczywistym osiągają takie systemy jak Whisper czy wav2wec 2.0.[2]. Należy pamiętać, iż Kaldi to zestaw narzędzi z przeszło 10 letnią historią, a wykorzystanie go w tej pracy podyktowane jest przystępniejszą implementacją i zasobami dotyczącymi algorytmu GoP.

4.5 Wyniki modeli - WER

Wyniki treningu ASR zostały opracowane w porównaniu z modelami referencyjnymi oraz wynikami wybranych modeli.

Tabela 7: Wyniki WER [%] dla najlepszych modeli - dane treningowe

Dane	dev				dev_other			
Model \Model językowy	LM4	LM3	PLM3	SLM3	LM4	LM3	PLM3	SLM3
1B (Basic)	4,53	4,70	5,93	6,59	12,51	13,15	15,68	17,04
1C (S)	4,38	4,50	5,56	6,13	12,03	12,54	14,50	15,86
1C (F)	4,36	4,51	5,46	6,06	12,02	12,44	14,45	15,55
1C (FNorm)	**4,36**	**4,53**	**5,57**	**6,17**	**12,11**	**12,52**	**14,64**	**15,75**

Tabela 8: Wyniki WER [%] dla najlepszych modeli - dane testowe

Dane	test				test_other			
Model \Model językowy	LM4	LM3	PLM3	SLM3	LM4	LM3	PLM3	SLM3
1B (Basic)	5,04	5,16	6,27	7,02	12,51	13,14	15,44	16,94
1C (S)	4,84	4,95	5,89	6,51	12,55	13,02	15,09	16,24
1C (F)	4,81	4,98	5,89	6,40	12,57	12,85	14,95	15,99
1C (FNorm)	**4,89**	**4,97**	**5,95**	**6,62**	**12,40**	**12,84**	**14,97**	**16,07**

Na podstawie wyników przedstawionych w Tabeli 7 oraz 8, wytrenowanych modeli najlepiej radzących sobie w algorytmie GoP, widać, iż największe znaczenie dla wyniku modeli ASR ocenianego za pomocą miary WER miała wprowadzona poprawka i zmiana architektury 1B na 1C ostatecznego modelu 1C (FNorm). Widać, iż w ewaluacji za pomocą wszystkich modeli językowych, wyniki WER są lepsze dla modeli 1C z wyjątkiem niewielkiej różnicy przy LM4 dla modeli 1C (F) oraz 1C (S).

Znaczenie również miało, choć w mniejszej mierze zastosowanie większej losowości danych w modelu 1C (F) względem 1C (S) oraz zmiany próbek na prawdopodobieństwo a priori. W tym porównaniu jednak, największe znaczenie miało zoptymalizowanie ustawień tak, by można było trenować model na większym Mini Batchu oraz zwiększenie zakresu LR, który mógł przyczynić się do bardziej optymalnego procesu uczenia.

Warto zauważyć, iż wprowadzenie do ostatniego modelu normalizacji CMVN nie wpłynęło pozytywnie na wyniki ASR w większości przypadków, ale mimo tego zaobserwowano pozytywne zmiany w poprzednio analizowanej mierze GoP w Podrozdziale *4.2 Wyniki algorytmu GoP*. Można więc zauważyć, iż wyniki modelu nie są do końca skorelowane z wynikami algorytmu GoP, więc nie wszystkie rozwiązania skuteczne w treningu modelu ASR, będą przekładały się na pozytywny wpływ na wyniki GoP.

Finalnie warto również zauważyć, iż wybrane wytrenowane modele osiągnęły wyniki zbliżone lub nieco lepsze od rezultatów modeli referencyjnych przedstawionych w Tabeli 5 oraz 6.

Tabela 9: Porównanie wyników WER [%] najlepszego modelu i wyjściowego modelu trifonicznego - dane treningowe

Dane	dev				dev_other			
Model \Model językowy	LM4	LM3	PLM3	SLM3	LM4	LM3	PLM3	SLM3
1C (FNorm)	4,36	4,53	5,57	6,17	12,11	12,52	14,64	15,57
tri6b_cleaned	6,93	7,39	9,19	10,39	19,93	21,16	24,13	26,05

Tabela 10: Porównanie wyników WER [%] najlepszego modelu i wyjściowego modelu trifonicznego - dane testowe

Dane	test				test_other			
Model \Model językowy	LM4	LM3	PLM3	SLM3	LM4	LM3	PLM3	SLM3
1C (FNorm)	4,89	4,97	5,95	6,62	12,40	12,84	14,97	16,07
tri6b_cleaned	7,61	7,96	9,68	10,89	21,36	22,46	25,52	27,44

W Tabeli 9 oraz 10, przestawiono również wyniki najlepszego z modeli 1C (FNorm) w porównaniu do wyników modelu trifonicznego będącego podstawą systemu ASR. Celem przedstawienia tego porównania jest obserwacja postępu treningowego w ostatniej fazie, który jest znaczący, co można uznać za sukces całego procesu. W niektórych przypadkach, takich jak bardziej skomplikowane i wymagające bazy typu 'other' wykorzystywane z mniejszymi modelami językowymi PLM3 i SLM3, różnica i redukcja WER wynosi, aż około 10 punktów procentowych.

Tabela 11: Porównanie wyników WER [%] najlepszych modeli ASR - dane treningowe

Dane	dev				dev_other			
Model \Model językowy	LM4	LM3	PLM3	SLM3	LM4	LM3	PLM3	SLM3
1C (FNorm)	4,36	4,53	5,57	6,17	12,11	12,52	14,64	15,57
1C (Dotrenowany)	4,32	4,54	5,57	6,13	12,23	12,66	14,73	15,86
1B (Rozszerzony LR)	4,46	4,61	5,81	6,58	11,92	12,44	14,82	16,47
1B (Basic)	4,53	4,70	5,93	6,59	12,51	13,15	15,68	17,04

Tabela 12: Porównanie wyników WER [%] najlepszych modeli ASR - dane testowe

Dane	test				test_other			
Model \Model językowy	LM4	LM3	PLM3	SLM3	LM4	LM3	PLM3	SLM3
1C (FNorm)	4,89	4,97	5,95	6,62	12,40	12,84	14,97	16,07
1C (Dotrenowany)	4,96	5,10	5,97	6,54	12,57	12,99	15,05	16,24
1B (Rozszerzony LR)	5,03	5,21	6,35	7,17	12,36	12,79	15,31	16,78
1B (Basic)	5,04	5,16	6,27	7,02	12,51	13,14	15,44	16,24

Należy zwrócić również uwagę na wyniki pozostałych modeli ASR, które nie uzyskały znaczących wyników w mierze GoP, ale osiągnęły dobrą skuteczność w rozpoznawaniu mowy. Rezultaty tych modeli przedstawiono w Tabelach 11 i 12, wraz z dwoma modelami 1C (FNorm) i 1B (Basic) uwzględnionymi w poprzednich wynikach.

Model 1B (Rozszerzony LR) to model podstawowy 1B ze rozszerzonym LR. Wykorzystano w nim szerszy zakres wartości IELR oraz FELR, mający na celu znaczące przyśpieszenie wstępnego procesu, a zwiększenie dokładności ostatnich kroków treningu. Wyniki okazały się bardzo dobre, gdyż model uzyskał rezultaty przebijające wybrane modele w niektórych z baz walidacyjnych, a nawet uzyskując wyniki lepsze od modeli referencyjnych przedstawionych w Tabeli 5 oraz 6. Sugeruje to, iż przy większych zasobach czasowych, możliwe było by zoptymalizowanie i usprawnienie procesu pod kątem LR i epok oraz pozostałych parametrów sieci neuronowej.

W przypadku modeli 1C należy zauważyć również, iż dotrenowanie podstawowego modelu przyniosło pozytywne skutki dla rezultatów modelu 1C (Dotrenowany). W związku z tym można przyjąć, iż dalszy trening modeli 1C lub rozszerzenie bazy treningowej dla tych systemów - przez wzgląd na ich skomplikowanie skutkujące zapotrzebowaniem na większą ilość zasobów oraz czasu w przeprowadzanych działaniach, mógłby przynieść znacznie lepsze wyniki, zarówno dla modeli ASR, jak również w przypadku algorytmu GoP.

Należy zwrócić również uwagę na wyniki pozostałych nie przytoczonych tu modeli 1B, które przedstawiają sobą odwrotne wnioski, niż przedstawione powyżej dla modeli 1C. Dotrenowany na bazie modelu 1B (Basic) kontrolowany w punktach 6,9 oraz 12 epok model, osiągał co raz to gorsze rezultaty WER. Jest to bezpośrednia oznaka przetrenowania i wyznacznik optymalnej liczby epok treningu dla modelu 1B w takiej konfiguracji.

4.6 Konkluzja

Podsumowując wynik samego algorytmu GoP, można go uznać za umiarkowanie zadowalający z zastrzeżeniem dostrzeżonych potencjalnie problemów czy interpretacji danych, które mogły wpłynąć na jego skuteczność, takich jak liczność klas czy ustalenie progów. Poprawa tych elementów mogła by poskutkować znaczącą poprawą wyników. Innym rozwiązaniem do opracowania lepiej działającego algorytmu, może być zawężenie i rozwiązanie problemu klasyfikacji binarnej błędów wymowy.

W przypadku wyników ASR można uznać je za bardzo pozytywne, tym bardziej, iż wprowadzone zmiany poprawiły wyniki i wpłynęły na miarę GoP, co otwiera i ukazuje możliwe drogi rozwoju zarówno treningu ASR, jak i poprawy rezultatów algorytmu GoP.

5 Podsumowanie

W ramach wykonanej pracy osiągnięto wszystkie z założonych celów. Poniżej przedstawiono wszystkie z nich z uzasadnieniem sukcesów i przeprowadzonych działań.

Cel 1 - Analiza dostępnych metod oraz baz danych umożliwiających opracowanie prototypu metody oceny GoP

Pierwszy cel osiągnięto poprzez wykonanie dogłębnej analizy dostępnych metod oraz baz danych umożliwiających opracowanie prototypu GoP. W ramach poszukiwań odpowiednich narzędzi zgłębiono wiele dostępnych korpusów oraz publikacji dotyczących miary GoP. W trakcie studiowania prac naukowych i dostępnej teorii, zdecydowano się użyć w implementacji algorytmu GoP, rozwiązania przedstawionego w pracy naukowej "Improved mispronunciation detection with deep neural network trained acoustic models and transfer learning based logistic regression classifiers"z 2015 roku [11], która doprowadziła do rozwoju nad wprowadzaniem sieci neuronowych do algorytmu GoP. W poszukiwaniach odpowiedniej bazy danych do wykonania zadania, zdecydowano się wykorzystać korpus Librispeech [18], który posłużył do treningu wymaganego modelu ASR oraz zestaw danych Speechocean762 [15], który został wykorzystany bezpośrednio w opracowaniu prototypu GoP. Obie z baz danych zostały wybrane przez wzgląd, na darmowy dostęp do ich zasobów, zawarte transkrypcje audio, wielkość oraz uwzględnienie oceny fonetycznej.

Cel 2 - Implementacja i uruchomienie prototypu wybranej metody GoP

W realizacji drugiego z postawionych celów wykorzystano zestaw narzędzi Kaldi [4]. Narzędzia te udostępniały szereg przydatnych zasobów i rozwiązań umożliwiających przeprowadzenie działań mających na celu implementację i uruchomienie algorytmu GoP. W związku z przyjętym rozwiązaniem [11] oraz założeniami pracy, w celu uzyskania działającego prototypu GoP należało wytrenować również model ASR. W osiągnięciu założonego celu wykorzystano zasoby, sprzęt komputerowy oraz wsparcie zapewnione przez infrastrukturę PLGrid [3], a sam trening modeli ASR wykonano, wykorzystując architekturę bazowego modelu nnet3 znajdującego się w zestawie Kaldi [4].

Wdrożono również implementację algorytmu GoP oraz skutecznie uruchomiono prototyp wybranej metody działającej jako klasyfikator wypowiedzi i wykorzystującej bazę danych Speechocean762 [15]. W pracy nad algorytmem GoP również wykorzystano narzędzia Kaldi [4] oraz infrastrukturę PLGrid [3], która posłużyła również jako magazyn danych.

W wykonanych działaniach nie posłużono się żadnym z gotowych rozwiązań oraz wstępnie lub w pełni trenowanymi modelami i przeprowadzono implementację miary GoP oraz trening modelu ASR od podstaw.

Cel 3 - Ocena skuteczności zaimplementowanej metody

Cel trzeci, czyli ocena skuteczności zaimplementowanej metody składającej się na algorytm GoP oraz modele ASR, również został osiągnięty. W procesie analizy wyników GoP wybrano najlepsze modele ASR osiągające najwyższe rezultaty miary GoP. Wyniki interpretowano w zakresie problematyki regresji liniowej (MSE) oraz miar klasyfikacji. Wyniki algorytmu GoP mogłyby być lepsze, lecz wiązałoby się to z koniecznością pozyskania lepszej bazy o bardziej zróżnicowanym rozkładzie klas. W przypadku modeli ASR ocenianych w zakresie rozpoznawania mowy wyniki są zadowalające, a wprowadzone modyfikacje poprawiły skuteczność modelu względem modelu bazowego, dostępnego w kodzie źródłowym w repozytorium zestawu narzędzi Kaldi [4], na stronie internetowej serwisu GitHub. Wprowadzone zmiany wpłynęły również pozytywnie na uzyskiwane wyniki miary GoP.

Potencjalny rozwój projektu

Przeprowadzone działania oraz wnioski, które wyciągnięto z przeprowadzonych eksperymentów sugerują, iż możliwa jest poprawa skuteczności przedstawionych rozwiązań oraz ich optymalizacja. Patrząc przyszłościowo można rozważać również wykorzystanie badanego algorytmu w innych zastosowaniach, nie tylko w ocenie poprawności wymowy, ale również w detektorach błędów czy innych bardziej specyficznych klasyfikatorach operujących na sygnałach mowy. Miara GoP zaimplementowana w odpowiednich projektach mogłaby przyczynić się również do pomocy w rozwiązywaniu problemów wad wymowy czy schorzeń powiązanych z traktem głosowym. Co więcej równolegle z poprawą wyników algorytmu, możliwa byłaby implementacja aplikacji opartej częściowo na tej mierze, działająca jako wirtualny nauczyciel języka angielskiego.

Literatura

[1] Automatic speech recognition on librispeech. https://paperswithcode.com/sota/automatic-speech-recognition-on-librispeech-8. [Dostęp: 21 grudnia 2023].

[2] Benchmarking top open source speech recognition models: Whisper, facebook wav2vec2, and kaldi. https://deepgram.com/learn/benchmarking-top-open-source-speech-models. [Dostęp: 12 grudnia 2023].

[3] Infrastruktura plgrid. https://www.plgrid.pl. [Dostęp: 2 listopada 2023].

[4] Kaldi toolkit. https://kaldi-asr.org. [Dostęp: 19 października 2023].

[5] Librivox. https://librivox.org. [Dostęp: 21 października 2023].

[6] Project gutenberg. https://www.gutenberg.org. [Dostęp: 21 października 2023].

[7] Nancy F. Chen and Haizhou Li. Computer-assisted pronunciation training: From pronunciation scoring towards spoken language learning. In *2016 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*, pages 1–7, 2016.

[8] Yiming Wang Ke Li Hainan Xu Masha Yarmohamadi Sanjeev Khudanpur Daniel Povey, Gaofeng Cheng. Semi-orthogonal low-rank matrix factorization for deep neural networks. In *Proc. Interspeech 2018*, 2018.

[9] M.J.F. Gales. Maximum likelihood linear transformations for hmm-based speech recognition. *Computer Speech Language*, 12(2):75–98, 1998.

[10] Minhua Wu Daniel Povey Sanjeev Khudanpur Guoguo Chen, Hainan Xu. Pronunciation and silence probability modeling for asr. In *Proc. Interspeech 2015*, 2015.

[11] Qian Y. Soong F. K. Wang Y. Hu, W. Improved mispronunciation detection with deep neural network trained acoustic models and transfer learning based logistic regression classifiers. *Speech Communication, 67 (January)*, pages 154–166, 2015.

[12] Wenping Hu, Yao Qian, and Frank K. Soong. An improved DNN-based approach to mispronunciation detection and diagnosis of L2 learners' speech. In *Proc. Speech and Language Technology in Education (SLaTE 2015)*, pages 71–76, 2015.

[13] M.Y. Hwang and X. Huang. Subphonetic modeling with markov states-senone. In *[Proceedings] ICASSP-92: 1992 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 33–36 vol.1, 1992.

[14] Aleksandra Król-Nowak i Katarzyna Kotarba. In *Podstawy uczenia maszynowego*, page 34, 2022.

[15] Yongqing Wang Zhiyong Yan Qiong Song Yukai Huang Ke Li Daniel Povey Yujun Wang Junbo Zhang, Zhiwen Zhang. speechocean762: An Open-Source Non-native English Speech Corpus For Pronunciation Assessment. In *Proc. Interspeech 2021*, 2021.

[16] Wai-Kim Leung, Xunying Liu, and Helen Meng. Cnn-rnn-ctc based end-to-end mispronunciation detection and diagnosis. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8132–8136, 2019.

[17] Kun Li, Xiaojun Qian, and Helen Meng. Mispronunciation detection and diagnosis in l2 english speech using multidistribution deep neural networks. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(1):193–207, 2017.

[18] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: An asr corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5206–5210, 2015.

[19] Douglas Reynolds. *Universal Background Models*, pages 1349–1352. Springer US, Boston, MA, 2009.

[20] Balakrishnan Varadarajan, Daniel Povey, and Stephen M. Chu. Quick fmllr for speaker adaptation in speech recognition. In *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 4297–4300, 2008.

[21] Laurila K. Viikki O. Cepstral domain segmental feature vector normalization for noise robust speech recognition. In *„Speech Communication" 1998, nr 25, 133–147 Elsevier*, 1998.

[22] Chen Chen Zhaoxin Zhang Yi Xin Wei Wang, Wenjie Song. I-vector features and deep neural network modeling for language recognition. In *2018 International Conference on Identification, Information and Knowledge in the Internet of Things, IIKI 2018*, 2018.

[23] S.M. Witt and S.J. Young. Phone-level pronunciation scoring and assessment for interactive language learning. 1998.

[24] Songjun Cao Binghuai Lin Long Ma Xiaoshuo Xu, Yueteng Kang. Explore wav2vec 2.0 for mispronunciation detection. *Conference of the International Speech Communication Association (INTERSPEECH)*, 2021.

Załącznik 1 - nazwy funkcji konwertujących wraz z opisami

Poniżej przedstawiono nazewnictwo funkcji konwertujących wraz z opisem zastosowania.

- ***lattice-copy.cc*** - kopia lub konwersja danych typu "lattice"będących reprezentacją sekwencji słownych i fonetycznych na formaty tekstowe
- ***copy-feats.cc*** - kopiowanie cech w formacie danych "feat"
- ***nnet3-am-copy.cc*** - konwersja modeli akustycznych pomiędzy formatami "raw"oraz "mdl"
- ***nnet3-compute.cc*** - propagacja danych przez wybrany model akustyczny
- ***nnet3-copy.cc*** - kopiowanie modeli akustycznych w formacie "raw"

Załącznik 2 - wykaz skrótów i nazw wykorzystanych w pracy inżynierskiej - część 1

Tabela 13: Wykaz skrótów i nazw wykorzystanych w pracy inżynierskiej - część 1

Skrót	Nazwa w języku angielskim
AGH	AGH University of Science and Technology
ASR	Automatic Speech Recognition
CALL	Computer-Assisted Language Learning
CMVN	Cepstral Mean and Variance Normalization
CPU	Central Processing Unit
CUDA	Compute Unified Device Architecture
DNN	Deep Neural Network
DNN-HMM	Deep Neural Network-Hidden Markov Model
FA	Forced Allignment
FELR	Final Effective Learning Rate
fMLLR	Feature space Maximum Likelihood Linear Regression
GMM	Gaussian Mixture Model
GMM-HMM	Gaussian Mixture Model-Hidden Markov Model
GoP	Goodness of Pronunciation
GoP-NN	Goodness of Pronunciation-Neural Network
GPU	Graphics Processing Unit
HMM	Hidden Markov Model
IDE	Integrated Development Environment
IELR	Inital Effective Learning Rate
JSON	JavaScript Object Notation
LDA	Linear Discriminant Analysis
LM4	Language Model 4
LM3	Language Model 3
LPP	Log Phone Posterior
LPR	Log Posterior Ratio
MLLT	Maximum Likelihood Linear Transform

Załącznik 3 - wykaz skrótów i nazw wykorzystanych w pracy inżynierskiej - część 2

Tabela 14: Wykaz skrótów i nazw wykorzystanych w pracy inżynierskiej - część 2

MSE	Mean Square Error
MFCC	Mel-Frequency Cepstral Coefficients
NVCC	Nvidia CUDA Compiler
OOM	Out-of-memory
PCA	Principal Component Analysis
PLM3	Pruned Language Model 3
POS	Part-Of-Speech
ReLU	Rectified Linear Unit
SAT	Speaker Adaptive Training
SLM3	Small Language Model 3
SLURM	Simple Linux Utility for Resource Management
SSH	Secure Shell
TDNN	Time Delay Neural Network
TDNN-F	Time Delay Neural Network-Factorized
UBM	Universal Background Models
UEFI	Unified Extensible Firmware Interface
URL	Uniform Resource Locator
WAV	Waveform Audio Format
WER	Word Error Rate
WSJ	Wall Street Journal
WSL2	Windows Subsystem for Linux