Kauno technologijos universitetas

Elektros ir elektronikos fakultetas

Garsyno LIEPA atpažinimo su Kaldi paketu naudojant giliuosius neuroninius tinklus sistemos sukūrimas ir tyrimas

Baigiamasis magistro projektas

Domantas Anglickis

Projekto autorius

Doc. dr. Kastytis Ratkevičius

Vadovas

Kaunas, 2022

Kauno technologijos universitetas

Elektros ir elektronikos fakultetas

Garsyno LIEPA atpažinimo su Kaldi paketu naudojant giliuosius neuroninius tinklus sistemos sukūrimas ir tyrimas

Baigiamasis magistro projektas

Valdymo technologijos (6211EX014)

Domantas Anglickis

Projekto autorius

Doc. dr. Kastytis Ratkevičius

Vadovas

Lekt. dr. Kęstas Rimkus

Recenzentas

Kaunas, 2022

Kauno technologijos universitetas

Elektros ir elektronikos fakultetas

Domantas Anglickis

Garsyno LIEPA atpažinimo su Kaldi paketu naudojant giliuosius neuroninius tinklus sistemos sukūrimas ir tyrimas

Akademinio sąžiningumo deklaracija

Patvirtinu, kad:

1. baigiamąjį projektą parengiau savarankiškai ir sąžiningai, nepažeisdama(s) kitų asmenų autoriaus ar kitų teisių, laikydamasi(s) Lietuvos Respublikos autorių teisių ir gretutinių teisių įstatymo nuostatų, Kauno technologijos universiteto (toliau – Universitetas) intelektinės nuosavybės valdymo ir perdavimo nuostatų bei Universiteto akademinės etikos kodekse nustatytų etikos reikalavimų;

2. baigiamajame projekte visi pateikti duomenys ir tyrimų rezultatai yra teisingi ir gauti teisėtai, nei viena šio projekto dalis nėra plagijuota nuo jokių spausdintinių ar elektroninių šaltinių, visos baigiamojo projekto tekste pateiktos citatos ir nuorodos yra nurodytos literatūros sąraše;

3. įstatymų nenumatytų piniginių sumų už baigiamąjį projektą ar jo dalis niekam nesu mokėjęs (-usi);

4. suprantu, kad išaiškėjus nesąžiningumo ar kitų asmenų teisių pažeidimo faktui, man bus taikomos akademinės nuobaudos pagal Universitete galiojančią tvarką ir būsiu pašalinta(s) iš Universiteto, o baigiamasis projektas gali būti pateiktas Akademinės etikos ir procedūrų kontrolieriaus tarnybai nagrinėjant galimą akademinės etikos pažeidimą.

Domantas Anglickis

Patvirtinta elektroniniu būdu

Anglickis, Domantas. Garsyno LIEPA atpažinimo su Kaldi paketu naudojant giliuosius neuroninius tinklus sistemos sukūrimas ir tyrimas. Magistro baigiamasis projektas / vadovas doc. dr. Kastytis Ratkevičius; Kauno technologijos universitetas, Elektros ir elektronikos fakultetas.

Studijų kryptis ir sritis (studijų krypčių grupė): elektronikos inžinerija, inžinerijos mokslai.

Reikšminiai žodžiai: Kaldi, Liepa, ASR, DNN.

Kaunas, 2022. 48 p.

Santrauka

Šiame magistro baigiamajame darbe, su Kaldi šnekos atpažinimo programiniu paketu, naudojant giliuosius neuroninius tinklus, sukuriama ir tiriama lietuviško garsyno Liepa atpažinimo sistema. Apžvelgiamas automatinių šnekos atpažinimo sistemų veikimas, giliųjų neuroninių tinklų taikymas automatinėse šnekos atpažinimo sistemose, programinio paketo Kaldi funkcionalumas, lietuviškas garsynas Liepa ir susiję moksliniai tyrimai. Pateikiama hibridinės automatinės šnekos atpažinimo sistemos, sudarytos iš paslėptų Markovo modelių, Gauso mišinių modelių ir giliųjų neuroninių tinklų, struktūra ir metodinis aprašas. Atliekamas modelio, su giliaisiais neuroniniais tinklais, tikslumo priklausomybės patikrinimas, nuo paslėptų Markovo modelių ir Gauso mišinių modelių parametrų. Išbandoma 18 skirtingų neuroninio tinklo architektūrų, sudarytų iš laiko vėlinimo neuroninių tinklų, ilgos trumpalaikės atminties neuroninių tinklų ir dvikryptės ilgos trumpalaikės atminties neuroninių tinklų kombinacijų. Atliekamas pasirinktos neuroninio tinklo architektūros mokymo parametrų optimizavimas ir kryžminė patikra. Pateikiami gauti rezultatai ir išvados.

Anglickis, Domantas. Creation and Investigation of LIEPA Speech Corpus Recognition System Using Kaldi Package and Deep Neural Networks. Master's Final Degree Project / supervisor doc. dr. Kastytis Ratkevičius; Faculty of Electrical and Electronics Engineering, Kaunas University of Technology.

Study field and area (study field group): electronics engineering, engineering science.

Keywords: Kaldi, Liepa, ASR, DNN.

Kaunas, 2022. 48 pages.

Summary

In this master's thesis, automatic speech recognition system for Lithuanian speech corpus Liepa is created and investigated, using Kaldi speech recognition toolkit and deep neural networks. The operation of automatic speech recognition systems, application of deep neural networks in automatic speech recognition systems, functionality of the software package Kaldi, Lithuanian speech corpus Liepa and related research works are reviewed. The structure and methodological description of a hybrid automatic speech recognition system consisting of hidden Markov models, Gaussian mixture models, and deep neural networks are presented. The dependence of the accuracy of the model with deep neural networks on the parameters of the hidden Markov models and Gaussian mixture models is checked. 18 different neural network architectures consisting of combinations of time delay neural networks, long short-term memory neural networks, and bidirectional long short-term neural networks are tested. Optimization and cross-validation of training parameters of the selected neural network architecture is performed. The obtained results and conclusions are presented.

Turinys

Lentelių sąrašas

Paveikslų sąrašas

Santrumpų sąrašas

Santrumpos:

ASR – automatinis šnekos atpažinimas (ang. Automatic Speech Recognition);

MFCC – Melų dažnių skalės kepstriniai koeficientai (ang. Mel Frequency Cepstral Coefficients);

HMM – paslėptas Markovo modelis (ang. Hidden Markov model);

GMM – Gauso mišinių modeliai (ang. Gaussian Mixture models);

DNN – gilusis neuroninis tinklas (ang. Deep Neural Network);

CNN – konvoliucinis neuroninis tinklas (ang. Convolutional Neural Network);

RNN – rekurentinis neuroninis tinklas (ang. Recurrent Neural Network);

TDNN – laiko vėlinimo neuroninis tinklas (ang. Time-delay Neural Network);

LSTM – ilga trumpalaikė atmintis (ang. Long Short-term Memory);

BLSTM – dvikryptė ilga trumpalaikė atmintis (ang. Bidirectional Long Short-term Memory);

ETE – atpažinimas nuo signalo vieneto pradžios iki galo (ang. end-to-end recognition);

LDA – tiesinė diskriminantinė analizė (ang. Linear Discriminant Analysis);

MLLT – didžiausios tikimybės linijinė transformacija (ang. Maximum Likelihood Linear transform);

SAT – adaptyvus diktoriaus mokymas (ang. Speaker Adaptive Training);

CMVN – kepstrinė vidurkio ir dispersijos normalizacija (ang. Cepstral Mean and Variance Normalization);

SRILM – Stanfordo tyrimų instituto kalbos modelis (ang. Stanford Research Institute language model);

UBM – universalaus fono modelis (ang. Universal Background Model);

ICS - vidinis kovariacijos poslinkis (ang. Internal Covariate Shift);

WER – žodžių klaidų dažnis (ang. Word Error Rate);

SER – sakinių klaidų dažnis (ang. Sentence Error Rate);

Įvadas

Šnekos atpažinimas, taip pat žinomas kaip automatinis šnekos atpažinimas (toliau ASR), arba kompiuterinis šnekos atpažinimas yra procesas ir su juo susijusios technologijos, paverčiančios kalbos signalą į atitinkamą žodžių seką ar kitus kalbinius objektus naudojant įvairiuose įrenginiuose įdiegtus algoritmus [1]. Istoriškai ASR programos apėmė rinkimą balsu, skambučių nukreipimą, interaktyvų balso atsakymą, duomenų įvedimą ir diktavimą, balso komandų valdymą, žaidimus, struktūrizuotų dokumentų kūrimą (pvz., medicininius ir teisinius įrašus), prietaiso valdymą balsu, kompiuterinį kalbos mokymąsi, turinio pagrindu atliekama sakytinę garso paiešką ir robotiką.

Sparčiai tobulėjant kompiuterinėms technologijoms, su garso atpažinimu susijusios technologijos taip pat labai sparčiai tobulėja ir ankščiau iššūkių kėlusios programos tampa realybe. Visi išmanieji telefonai ir kompiuteriai, kurie turi mikrofoną ir prieiga prie, pvz., „Google“ ar „Bing“ paieškos sistemų, gali atlikti paiešką balsu. Taip pat jau tapo įprasta, kad atskiri gamintojai savo įrenginiuose siūlo balsu valdomus asmeninius asistentus. ASR plačiai taikomas ir namų pramogų sistemose, mašininiame vertime, namų automatikos sistemose, transporto priemonėse, kurios naudoja balsu valdomas navigacijos ir pramogų sistemas bei įvairiose į kalbą orientuotose informacijos apdorojimo programose, kurios naudoja ASR išvesties apdorojimą [2].

Remiantis pasaulio kalbų statistika [3], pasaulyje yra 7151 žinoma kalba, kuria šneka bent vienas žmogus, kuriam tai yra gimtoji kalba. Didelė šnekamų kalbų įvairovė nereiškia, kad jos yra vienodai pasiskirsčiusios tarp pasaulio gyventojų – tik 23 kalbomis šneka daugiau nei pusė pasaulio gyventojų [3]. Populiariausioms šnekamosioms kalboms yra sukurta daug ASR, natūralios kalbos apdorojimo ir kompiuterinės kalbotyros išteklių. Tačiau nepopuliarioms kalboms dažnai trūksta tokių išteklių. Kadangi lietuvių kalba yra nepopuliari kalba, tyrimai susiję su kalbos atpažinimu ir jo vystymu yra svarbūs ir reikalingi.

Šio magistrinio darbo tikslas yra atlikti giliaisiais neuroniniais tinklais paremtų metodų efektyvumo tyrimą, atpažįstant lietuviško garsyno LIEPA turinį, naudojant Kaldi programinį paketą. Pagrindiniai darbo uždaviniai:

- atlikti literatūros analizę;
- pateikti sukurtos ASR sistemos metodinį aprašą;
- atlikti modelio su giliaisiais neuroniniais tinklais tikslumo priklausomybės nuo klasikinių modelių parametrų tyrimą;
- parinkti optimalią neuroninio tinklo architektūrą ir mokymo parametrus;
- atlikti kryžminį patikrinimą;
- pateikti rezultatus ir išvadas.

Šis darbas padalintas į 3 skyrius – pirmajame pateikiama literatūros apžvalga, antrajame pateikiamas metodo aprašas ir trečiajame pateikiami eksperimentų rezultatai ir išvados.

1. Literatūros apžvalga

Šiame skyriuje atliekama literatūros analizė, apžvelgiant automatinio šnekos atpažinimo pagrindus, giliųjų neuroninių tinklų taikymą ASR sistemose, programinio paketo Kaldi funkcionalumą, lietuviško garsyno Liepa savybes ir susijusius mokslinius tyrimus.

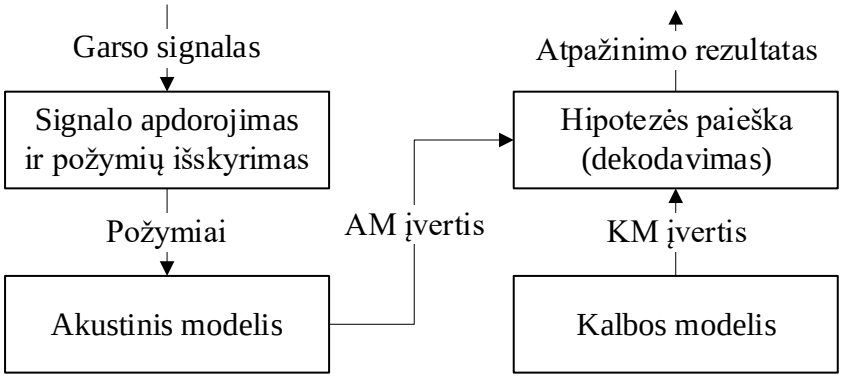
1.1. Automatinis šnekos atpažinimas

Automatinis šnekos atpažinimas (ASR) – tai aparatūrinės ir programinės įrangos naudojimas žmogaus šnekai identifikuoti ir paversti į rašytinį formatą. ASR taikymo galimybės yra labai plačios. Daugelis pramonės šakų ASR technologiją pritaikė klientų aptarnavimo kokybės gerinimui, pvz.:

- balsu valdomi virtualūs asistentai („Google Assistant“, „Apple“ „Siri“, „Amazon“ „Alexa“, „Microsoft“ „Cortana“);
- transkripcijos ir diktavimo sistemos naudojamos fiksuojant, pvz., įmonės susitikimus, klientų skambučius, paciento medicininės pastabas;
- kalbos mokyme, vertime;
- automobilio informacinėse ir pramoginėse sistemose, leidžiančiuose valdyti įvairias sistemos funkcijas balsu;
- saugumo didinimui reikalaujant balso atpažinimo patikros norint pasiekti tam tikras sritis;
- prieinamumo didinimui (žmonėms su negalia).

Dirbant gyvoje aplinkoje, ASR sistemos sąlygos paprastai nėra idealios, o tai turi įtakos sistemos tikslumui. Dažniausi pasitaikantys trikdžiai yra:

- foninis triukšmas;
- skirtingos diktorių savybės (lytis, dialektas, akcentas, balso aukštis, garsumas, greitis);
- prasta įrašinėjimo įranga;
- homofonai (vienodai tariami, bet skirtingai rašomi žodžiai);
- žodžių ribų trūkumas.



1 pav. ASR sistemos architektūra [1].

Klasikinės ASR sistemos [1] architektūra yra pavaizduota 1 pav. ASR sistema turi 4 pagrindinius komponentus: šnekos signalo apdorojimo ir požymių išskyrimo, akustinio modelio, kalbos modelio ir hipotezės paieškos (dekodavimo) komponentus. Šnekos signalo apdorojimo ir požymių išskyrimo komponentas priima garso signalo įėjimą, pašalina iš jo triukšmus ir kanalo iškraipymus, konvertuoja signalą iš laiko srities į dažnių sritį ir išskiria ryškiausių požymių vektorius, tinkančius sekančio akustinio modelio komponento įėjimui.

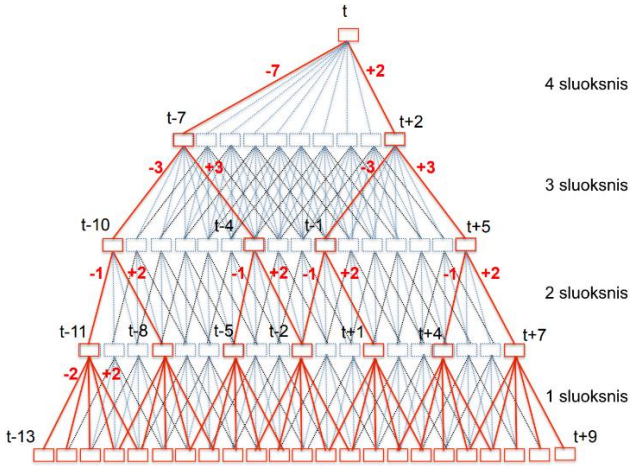
Akustinis modelis, kurio įėjimas yra požymių vektorius, sujungia akustinę ir fonetinę informaciją, pagal kurią generuoja kintamo ilgio požymių sekų, akustinio modelio tikimybinius įverčius. Kalbos modelis įvertina hipotetinės žodžių sekos tikimybę, pagal išmoktas koreliacijas tarp žodžių, iš mokymo žodyno. Dekodavimo komponentas sujungia akustinio ir kalbos modelių įverčius, atsižvelgiant į požymių vektorių seką ir hipotetinę žodžių seką, ir išveda žodžių seką su aukščiausiu tikimybiniu įverčiu, kaip atpažinimo rezultatą.

Požymių išskyrimas yra pagrindinė ASR sistemos dalis, laikoma sistemos širdimi [4]. Jos tikslas – iš įėjimo kalbos (signalo) išskirti tas savybes, kurios padeda sistemai identifikuoti šneką. Požymių išskyrimas suspaudžia įėjimo signalo dydį (į vektorių), nepakenkdamas kalbos signalo stipriui. Yra daug požymių išskyrimo būdų, tačiau ASR srityje dažniausiai naudojami Melų dažnių skalės kepstriniai koeficientai (toliau MFCC) [5]. MFCC požymių išskyrimo technika iš esmės apima lango funkcijos pritaikymą, diskretinės Furjė transformacijos taikymą, signalo stiprio logaritmų skaičiavimą ir dažnių dedamosios keitimą į melų skalės gaubtinę, po to, taikant atvirkštinę diskretinę kosinusinę transformaciją, apskaičiuojami Melų skalės kepstrai [6].

Akustinis modelis nustato ryšį tarp akustinės ir kalbinės informacijos [7]. Dažniausiai akustiniai modeliai yra paremti paslėptais Markovo modeliais (toliau HMM), Gauso mišinių modeliais (toliau GMM) ir giliaisiais neuroniniais tinklais (toliau DNN). HMM yra tikimybinis modelis, naudojamas paaiškinti arba išvesti bet kokio atsitiktinio proceso tikimybinę charakteristiką [8]. HMM modeliai leidžia rasti šnekos ištarimo sudarymo iš tam tikrų fonemų (garsinis kalbos elementas, turintis skiriamąją funkciją) tikimybę. GMM yra parametrinė tikimybės tankio funkcija, pavaizduota kaip svertinė Gauso komponentų tankių suma [9].

1.2. Gilieji neuroniniai tinklai

Gilaus mokymosi eroje neuroniniai tinklai žymiai pagerino ASR sistemų rezultatus [10]. Taikomi įvairūs metodai, pvz., konvoliuciniai neuroniniai tinklai (toliau CNN), rekurentiniai neuroniniai tinklai (toliau RNN) ir jų modifikacijos. Šiame darbe sukurtoje ASR sistemoje naudojami laiko vėlinimo neuroniniai tinklai (toliau TDNN), ilgos trumpalaikės atminties neuroniniai tinklai (toliau LSTM) ir dvikrypčiai ilgos trumpalaikės atminties neuroniniai tinklai (toliau BLSTM).

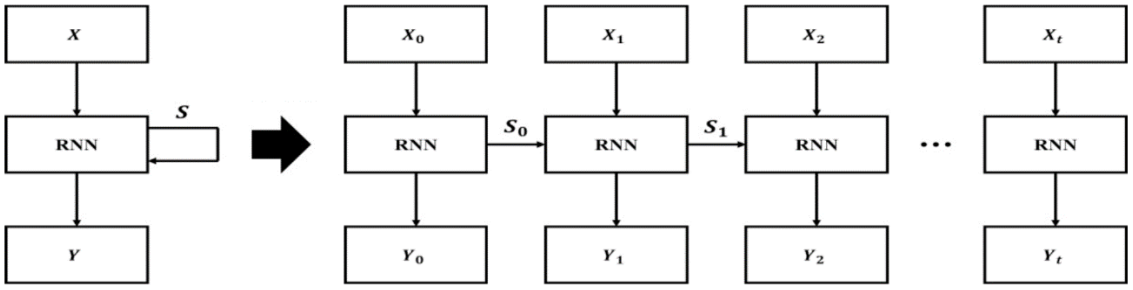


2 pav. TDNN su daline atranka (raudona) ir be papildomos atrankos (mėlyna + raudona) [13].

TDNN tinklą pirmasis pasiūlė Waibel [11,12], o vėliau išpopuliarino Povey ir kt. [13], panaudoję jį kaip akustinio modelio dalį. Tinklo įėjimas yra akustinių požymių kadrai, o išėjimas yra fonemų tikimybių skirstinys.

Viename TDNN sluoksnyje kiekvienas įėjimo kadras yra stulpelio vektorius, vaizduojantis vieną signalo laiko žingsnį, o eilutės atspindi požymių vertes. Tinklas naudoja mažesnę svorių matricą (branduolį arba filtrą), kuri slysta per šį signalą ir konvertuoja jį į įsėjimą naudojant konvoliucijos operaciją. Neuronai, naudojami daugelyje neuroninių tinklų, apskaičiuoja svertinę savo įėjimų sumą ir perduoda šią sumą per netiesinę funkciją, dažniausiai slenkstinę arba sigmoidinę funkciją. TDNN modelyje šie neuronai modifikuojami įvedant vėlavimus [11]. Tokiu būdu TDNN modelis turi galimybę susieti ir palyginti dabartinį įėjimą su praeities įvykių istorija. Siekiant sumažinti skaičiavimų kiekį, Povey ir kt. [13] pasiūlė dalinės atrankos metodą (žr. 2 pav.), kuriame kiekviename tinklo sluoksnyje paslėpti aktyvavimai apskaičiuojami tik keliais laiko momentais. Tinkamai parinkus laiko etapus, kuriais apskaičiuojami aktyvavimai, galima sumažinti skaičiavimų kiekį, tuo pačiu užtikrinant, kad informacija iš visų įėjimo konteksto laiko etapų būtų apdorojama ir nebūtų prarasta.

LSTM yra patobulinta tradicinio rekurentinio neuroninio tinklo versija. RNN tinklas yra matomas 3 pav.

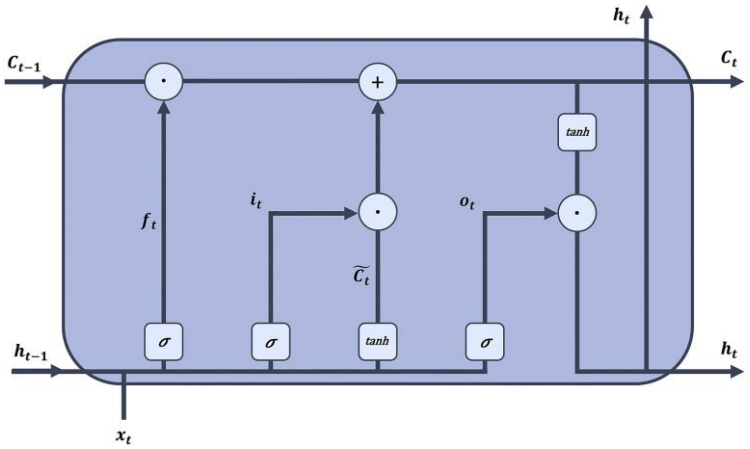


3 pav. Pilnai rekurentinis neuroninis tinklas [14].

RNN tinkle, įėjimo vertė (*x*) perduodama per modelį, kuris tuo laiko momentu turi paslėptas arba išmoktas būsenas *S*. Modelis sukuria išėjimą *Y*, kuris atitinka norimą rezultatą. Naudojant tokį būdą, galima konvertuoti, pavyzdžiui, įėjimo anglų kalbą į išėjimo vokiečių kalbą. Todėl RNN yra plačiai naudojamas kaip seka-į-seką (ang. sequence-to-sequence) modelis. Tačiau tą patį galime padaryti ir su klasikiniais neuroniniais tinklais. RNN nauda yra ta, kad jie perduoda išėjimą sau, kad jį būtų galima panaudoti kito perdavimo metu. Tai suteikia neoriniam tinklui sąryšį su ankstesniais įėjimais (tai kartais yra labai svarbu semantiškai painiose užduotyse, pavyzdžiui, vertime ar kalbos atpažinime). Todėl klasikiniai RNN yra ne kas kita, kaip visiškai apjungti tinklai, kurie neuroninius išėjimus perduoda atgal neuronams.

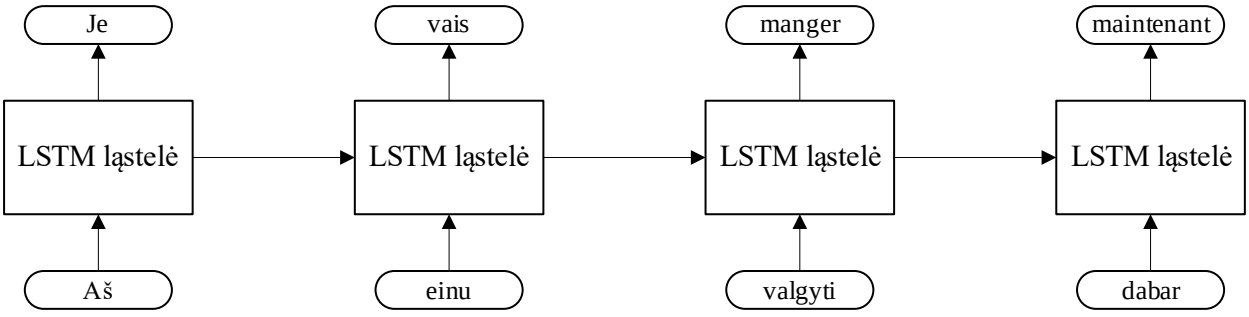
Istoriškai, RNN tinklų atsiradimas labai padėjo naujausių technologijų kūrimui [15]. Tačiau kyla kelios problemos, kai norima apmokyti klasikinius rekurentinius neuroninius tinklus. Jeigu įprasto neuroninio tinklo mokymui taikomas atgalinis paskirstymas (ang. backpropagation), klaidos yra apskaičiuojamos atvirkščiai, tam, kad gradientų atnaujinimai taptų žinomi ir juos galėtų naudoti optimizatorius. Rekurentinis atgalinis paskirstymas yra labai sudėtingas, todėl buvo sugalvota [16], kad reikia išskleisti, t.y. padaryti daug tinklo kopijų (su lygiai ta pačia inicializacija) ir jas tobulinti. Toks metodas leidžia lengviau apskaičiuoti gradientus ir juos sujungti, tai leido apmokyti rekurentinius tinklus. Gradientų sujungimas reiškia, kad reikia taikyti daugybos operacijas. Klasikiniai RNN dažniausiai apjungiami naudojant sigmoidines arba hiperbolines tangentines (tanh) aktyvinimo funkcijas. Kadangi šių funkcijų išvestinės beveik visada yra mažesnės už 1, atsiranda didelė išnykstančių gradientų problema, todėl klasikiniai RNN negalėjo būti naudojami, kai duomenų sekos tapo ilgos, jie tiesiog užstringa arba labai lėtai mokosi.

LSTM tinklai efektyviai atskiria išėjimus ir atmintį. Vadinamosiose atminties ląstelėse yra generuojamos prognozės ir atnaujinama atmintis. Vizualiai tai yra pavaizduota 4 pav.



4 pav. LSTM ląstelės struktūra [17].

Visas LSTM tinklo funkcionalumas yra vykdomas atminties ląstelėje. Kintamieji h_{t-1} ir h_t nurodo atminties ląstelės išėjimus atitinkamai t-1 ir t laiko momentais, t.y. ankstesnės ląstelės išėjimą į dabartinę ląstelę ir dabartinės ląstelės įsėjimą į kitą ląstelę. Kintamieji c_{t-1} ir c_t žymi atmintį žinomais laiko žingsniais. Atmintis yra atskirta nuo išėjimo kintamųjų ir yra atskiras objektas. Modelyje yra trys „vartai“, kuriuos vaizduoja trys elementų junginių blokai ląstelėje. Kairėje pusėje yra „užmiršimo vartai“, kurie pagal paskutinį išėjimą ir dabartinį įėjimą, naudojant sigmoidinę aktyvacijos funkciją, apskaičiuoja tai, kas gali būti užmiršta, t.y. ištrinta iš atminties, kuri yra susijusi su dabartiniu ir praeitu įėjimu. Pašalinimas atliekamas padauginus rezultatą iš atminties. Viduryje yra „įėjimo vartai“, kurie praėjusiam išėjimui ir dabartiniam įėjimui pritaiko abi sigmoidinę ir tanh aktyvacijos funkcijas. Sigmoidinė aktyvacija leidžia žinoti, ką reikia atskirti nuo įėjimų, o tanh aktyvacija normalizuoja reikšmes į diapazoną [-1, +1], taip stabilizuojant mokymo procesą. Rezultatai pirmiausia yra sudauginami (kad būtų užtikrinta normalizacija) ir po to yra pridedami prie atminties. Dešinėje pusėje yra „išėjimo vartai“, kuriose per tanh aktyvacijos funkciją yra normalizuojama atminties vertė, tuomet praėjusiam išėjimui ir dabartiniam įėjimui yra pritaikoma sigmoidinė aktyvacijos funkcija. Gauti rezultatai yra sudauginami, t.y. sužinome, ką reikia nuspėti dabartinei įėjimo vertei. Ši vertė tampa išėjimu, kuri kartu su atmintimi yra perduodama į kitą ląstelę.

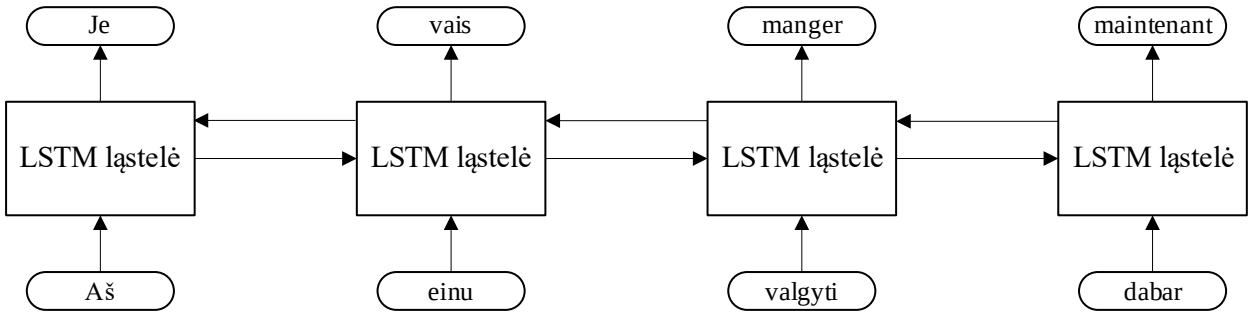


5 pav. Išskleistas LSTM modelis.

LSTM pagrindinis privalumas lyginant su klasikiniu RNN yra tai, kad atmintis yra atskirta nuo išėjimų. Visi elementai, kurie sukelia gradientų nykimą yra pačioje ląstelėje. Tarp ląstelinėje komunikacijoje – vieninteliai elementai kurie yra naudojami gradiento skaičiavimui yra daugyba ir sudėtis, t.y. tiesinės operacijos. Tai leidžia LSTM užtikrinti, kad gradientai tarp ląstelių visada bus lygūs 1, o tai išsprendžia nykstančių gradientų problemą.

LSTM vienakryptiškumas dalyje taikymo sričių gali apriboti sistemos rezultatus. Tarkime, norime apdoroti žodžių seką „Aš einu valgyti dabar“ naudojant LSTM su tikslu išversti žodžių junginį į prancūzų kalbą. Tokių duomenų apdirbimas paremtas žetonų principu, t.y. kiekvienas žetonas yra perleidžiamas per LSTM ląstelę, kuri apdoroja įėjimo žetoną ir sau perduodą paslėptą būseną. Atlikus išskleidimą, toks procesas yra matomas 5 pav.

Tai parodo, kad LSTM yra vienakrypčiai, t.y. sekos yra apdorojamos viena kryptimi, šiuo atveju iš kairės į dešinę. Toks būdas yra logiškas, kadangi daugelyje kalbų tekstas yra skaitomas ir rašomas iš kairės į dešinę. Vertimo tikslais tai nesukelia didelių problemų, kadangi nėra žinoma, kas bus pasakyta ateityje, todėl nereikia žinoti kas nutiks po dabartinio įėjimo žodžio. Tačiau vienakryptiškumas gali apriboti kuriamo modelio rezultatus, situacijose, kuriose reikia suprasti, pvz., tekstą, o ne atlikti seka-į-seką veiksmus. Norint suprasti tekstą reikia atlikti dvikrypčius veiksmus, todėl šiai užduočiai vienakrypčio LSTM neužtenka. Tam reikia dvikrypčio LSTM, kuriame sekos yra apdorojamos tiek iš kairės į dešinę, tiek ir iš dešinės į kairę, t.y. frazė „Aš einu valgyti dabar“ yra apdorojama kaip „Aš → einu → valgyti → dabar“ ir kaip „Aš ← einu ← valgyti ← dabar“. Tai suteikia kontekstą užduotims, kurioms reikia abiejų krypčių. Dvikrypčio LSTM arba BLSTM tinklo struktūra, ankščiau minėtam pavyzdžiui yra matoma 6 pav.



6 pav. Išskleistas BLSTM modelis.

Teoriškai BLSTM veikia dvikrypčiu būdu, tačiau praktiškai jie nėra dvikrypčiai, t.y jie yra dvi vienakryptės LSTM ląstelės, kurių išėjimai yra apjungiami. Išėjimus sujungti galima keliais metodais, pvz., vykdant:

- vektorių sumavimą, kai išėjimas yra lygus LSTM→ + LSTM←;
- vektorių vidurkio skaičiavimą, kai išėjimas yra lygus 0,5*(LSTM→ + LSTM←);
- vektorių daugybą, kai išėjimas yra lygus LSTM→ * LSTM←;
- vektorių sujungimą, kai išėjimo vektorius yra dvigubai didesnis nei įvesties vektorius, nes jie yra sujungiami pagal vektorių sujungimo taisykles.

Šiame darbe naudojamas vektorių sujungimas.

1.3. Programinis paketas Kaldi

Programinis paketas Kaldi yra nemokamas atviro kodo įrankis, skirtas šnekos atpažinimo tyrimams ir yra parašytas C++ programavimo kalba [18]. Šį programinį paketą galima paleisti „Microsoft Windows“ operacinėje sistemoje, tačiau rekomenduojama naudoti „Unix“ tipo operacines sistemas. Kaldi pakete kiekvienas ASR sistemos komponentas traktuojamas kaip nepriklausomas modulis. Išrinkti moduliai veikia kartu kaip viena grandininė sistema, todėl yra atskiras modulis kiekvienai pagrindinei ASR sistemos užduočiai, pvz., požymių išskyrimui, akustiniam ir fonetiniam modeliavimui bei kalbos modeliavimui.

Šiuolaikinės ASR sistemos skirstomos į šnekos atpažinimo nuo signalo vieneto pradžios iki galo sistemas (toliau ETE) ir grandininės sistemas [19]. Kaldi tipas yra grandininis ir skiriasi nuo ETE sistemos kiekvieno modulio lygyje. ETE sistemos įėjimas gali būti ir neapdorotas garso signalas, ir rankiniu būdu atrinkti požymiai, o grandininėse sistemose naudojami specifiniai moduliai, skirti šnekos požymių išskyrimui. ETE sistemos akustinis modelis gali pateikti simbolius kaip išėjimą, o grandininėje architektūroje kaip išėjimas pateikiamas fonetinių vienetų tikimybių skirstinys. Kitas skirtumas nuo ETE sistemų yra tas, kad HMM-DNN hibridinė architektūra negali būti mokoma nuo nulio, be iš anksto egzistuojančių sulygiuotų kadro lygio fonemų.

ETE architektūrose yra specialūs mechanizmai ar tinklo dalys, skirtos kalbos modelio kūrimui. Taip pat naudojami mechanizmai leidžiantys įterpti ir išorinius kalbos modelius į sistemą. Grandininėse sistemose kalbos modelis visada yra atskiras komponentas, reikalingas dekodavimo metu, įvertinantis tikimybę, kad žodžių sekos sudarys teisingą frazę.

Kaldi dekodavimo grafas, vadinamas HCLG grafu, yra svertinis baigtinių būsenų keitiklis, sudarytas iš keturių kitų grafų [20]. H žymi HMM struktūrą, kurios įėjimas yra akustinės būsenos, išėjimas – nuo konteksto priklausančios fonemos. Svoriai šiame grafe yra HMM perėjimo tikimybės. C reiškia nuo konteksto priklausomą perženklinimą, kai įėjimas, išreikštas nuo konteksto priklausomomis fonemomis, yra susiejamas su nuo konteksto nepriklausomomis fonemomis. L yra leksika, susiejanti fonemas su žodžiais. Svoriai L žymi tarimo tikimybes. Galiausiai G yra kalbos modelis, kuris naudoja žodžius kaip įėjimą, modeliuodamas jų priklausomybės tikimybes. Šie elementai sukuriami atskirai ir sujungiami kartu, prieš atliekant dekodavimą. HCLG gaunamas sujungiant atskirus grafus iš dešinės į kairę, naudojant formulę: $H \cdot (C \cdot (L \cdot G))$.

1.4. Lietuviškas garsynas Liepa

Kuriant ASR sistemas, viena iš problemų yra ta, kad tos pačios balso komandos – kiekvieną sykį yra sakomos skirtingai ir skirtingoje aplinkoje, kadangi vartotojai, kurie naudos balso atpažinimo sistemą yra labai skirtingi – skiriasi jų amžius, balso tembras, fizinė ir emocinė būklė, akustinė aplinka ir kt. Šnekos atpažinimo sistema turi susidoroti su skirtumais tarp balso komandų ir vienas iš šios problemos sprendimų galėtų būti platesnio profilio ir su daugiau duomenų garsyno sukūrimas. Populiarioms kalbos tai nėra didelė problema, kadangi jos turi labai daug kalbinės medžiagos, tačiau tokioms mažai paplitusioms kalboms kaip lietuvių, garsynų sudarymas leidžia gauti žymiai geresnius lietuvių šnekos atpažinimo rezultatus.

Siekiant pagerinti kalbos atpažinimo kokybę Lietuvoje, garsyną Liepa sukūrė Sigita Laurinčiukaitė ir kt. [21]. Garsyno autoriai nusprendė padalinti jį į dvi dalis – vieną, skirtą lietuvių šnekos atpažinimui ir kitą, skirtą teksto į šneką sintezei. Kuriant garsyną, autoriai siekė užtikrinti gerą kalbos specifikos perteikimą, naudojant skirtingus fonemų rinkinius. Pirma dalis apima 100 valandų garso įrašų, kuriuos ištaria 376 diktoriai, antra dalis apima 13 valandų garso įrašų, kuriuos ištaria 4 diktoriai. Pirmoje dalyje turinį sudaro žodžiai, frazės, sakiniai, tekstai, o antroje dalyje yra tik sakiniai. Abiejose dalyse iš viso naudojamos 92 fonemos, kalbos tipas yra tęstinis, garso įrašų dažnis yra 22 kHz ir kvantavimas yra 16 bitų.

Šiame darbe naudojama 1 garsyno dalis, kuri susideda iš trijų smulkesnių dedamųjų:

- žodžių ir trumpų žodžių junginių (toliau L_Zod);
- žodžių sekų (toliau L_Sek);
- sakinių (toliau L_Sak).

1.5. Susiję moksliniai tyrimai

Gailius Raškinis ir kt. [22] yra vieni iš pirmųjų Lietuvos mokslininkų, kurie automatiniam kalbos atpažinimui naudojo Kaldi programinį paketą. Autoriai straipsnyje apžvelgia atliktus tyrimus, susijusius su žodžių susiejimu su daliniais žodžiais Lietuviškose ASR sistemose. Taip pat palyginami įvairūs fonemomis ir grafemomis pagrįsti atvaizdavimai įvairiuose akustiniuose modeliuose.

Laurynas Dovydaitis [23] daktaro disertacijoje tyrė Lietuviško diktoriaus identifikavimo problemą, naudojant rekuriantinius neuroninius tinklus. Autorius naudojo garsyną Liepa ir geriausią atpažinimo rezultatą pasiekė naudojant LSTM tinklo architektūrą, nuo 3 % iki 6 % tiksliau, lyginant su HMM tiksliausiu modeliu [23].

Gintarė Žekienė [24] daktaro disertacijoje kūrė hibridinę lietuvių kalbos komandų atpažinimo sistemą, kuri apjungtų du ar daugiau kalbos atpažintuvų, remiantis prielaida, kad suklydus vienam atpažintuvui, kitas/kiti priims teisingą sprendimą. Autorės sukurtos hibridinės sistemos visais atvejais leido tiksliau atpažinti eksperimentams naudotus garsynus (vienas iš jų buvo Liepa).

Darius Kairys [25] magistro darbe sukūrė garsyno Liepa žodžių ir frazių sekų atpažinimo sistemą, naudojant TensorFlow programinį paketą, kuriame realizuotas *DeepSpeech2* algoritmas. Žodžių tyrimo dalyje geriausias pasiektas neteisingų žodžių įvertis yra 9,32 %, sekų tyrimo dalyje 28,19 %.

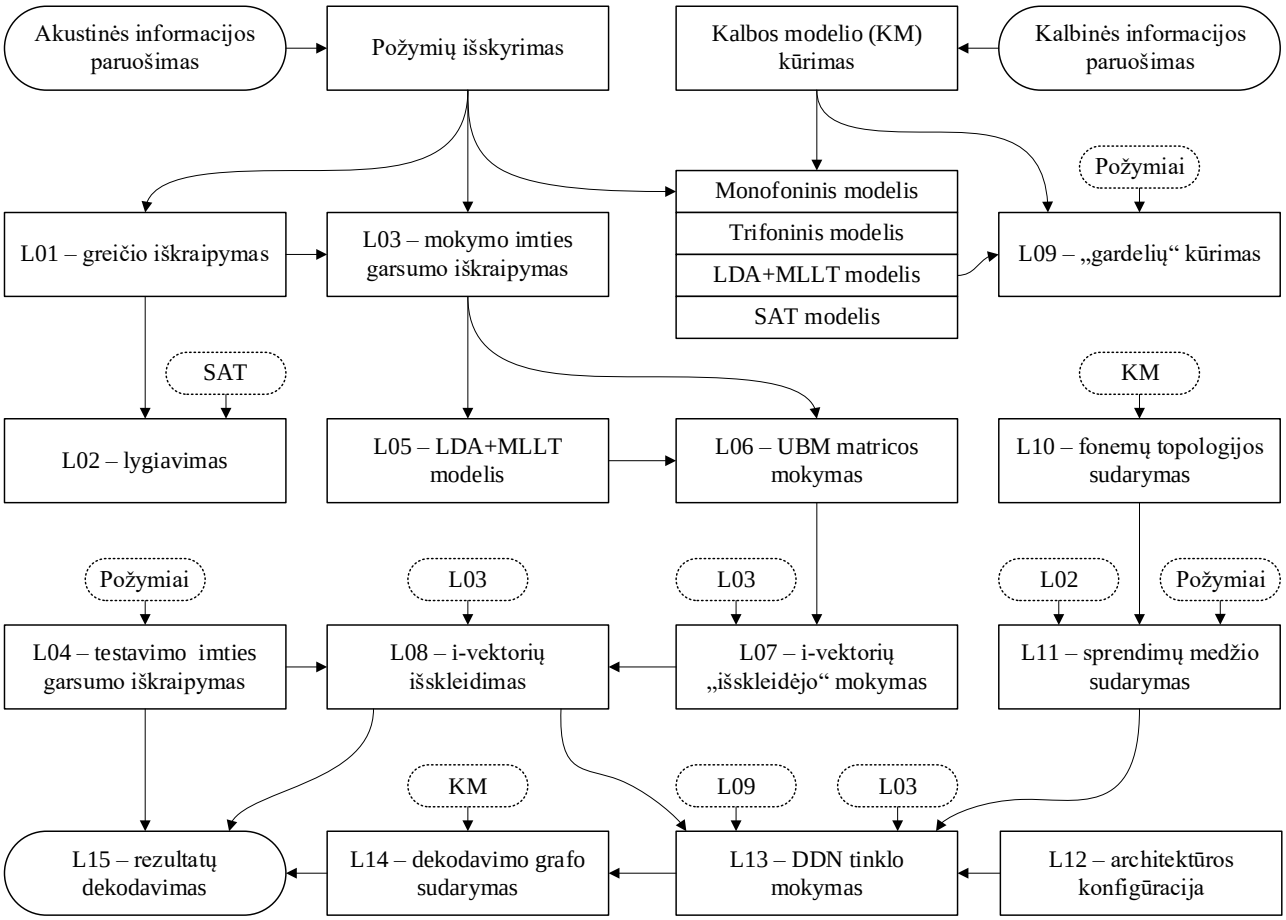
Giedrius Sinkevičius [26] magistro darbe skaičių garsyno tyrimams naudojo Kaldi paketą. Autorius pirmiausia atliko bandymus su 30 diktorių balso įrašais, kuriuose yra tariami lietuviški skaičiai nuo 0 iki 9. Bandymų rezultatai buvo lyginami su *HTK* programiniu paketu. Autorius tolimesniuose tyrimuose praplėtė garsyną iki 100 diktorių įrašų, su 5 dB foniniu garsu ir atliko bandymus su skirtingais HMM-GMM metodais.

Gytis Baltrušaitis [27] magistro darbe su Kaldi paketu sukūrė garsyno LIEPA atpažinimo sistemą. Žodžių atpažinimo dalyje geriausias rezultatas gautas naudojant poerdvės Gauso mišinių modelį [28], kurio neteisingų žodžių ir sakinių įverčiai atitinkamai yra 2,78 % ir 3,78 %. Sekų atpažinimo dalyje geriausias rezultatas gautas naudojant TDNN modelį (naudojama senesnė Kaldi paketo nnet2 modifikacija), kurio neteisingų žodžių ir sakinų įverčiai atitinkamai yra 7,52 % ir 68,22 %. Sakinių atpažinimo dalyje taip pat geriausias rezultatas pasiektas su TDNN modeliu, kurio neteisingų žodžių ir sakinų įverčiai atitinkamai yra 10,24 % ir 53,62 %.

Gediminas Norkus [29] magistro darbe tyrė lietuviškų vardų garsyno atpažinimo tikslumą, palygindamas Kaldi ir TensorFlow programinių paketų rezultatus. Anot autoriaus, Kaldi paketas žymiai geriau atpažįsta vardų garsyną: „TDNN *p-norm* metodu tikslumas siekia 98,65 %, palyginus su TensorFlow paketu – 87,24 % tikslumas“ [29].

2. Metodo aprašas

Sukurta ASR sistema yra hibridinė sistema, susidedanti iš HMM-GMM ir HMM-DNN akustinių modelių. HMM-GMM modelis naudojamas kaip HMM-DNN modelio mokymo pagrindas. Sudarytas modelis yra grandininis procesas, kurio apibendrinta struktūra yra pavaizduota 7 pav.



7 pav. Sukurtos ASR sistemos struktūra.

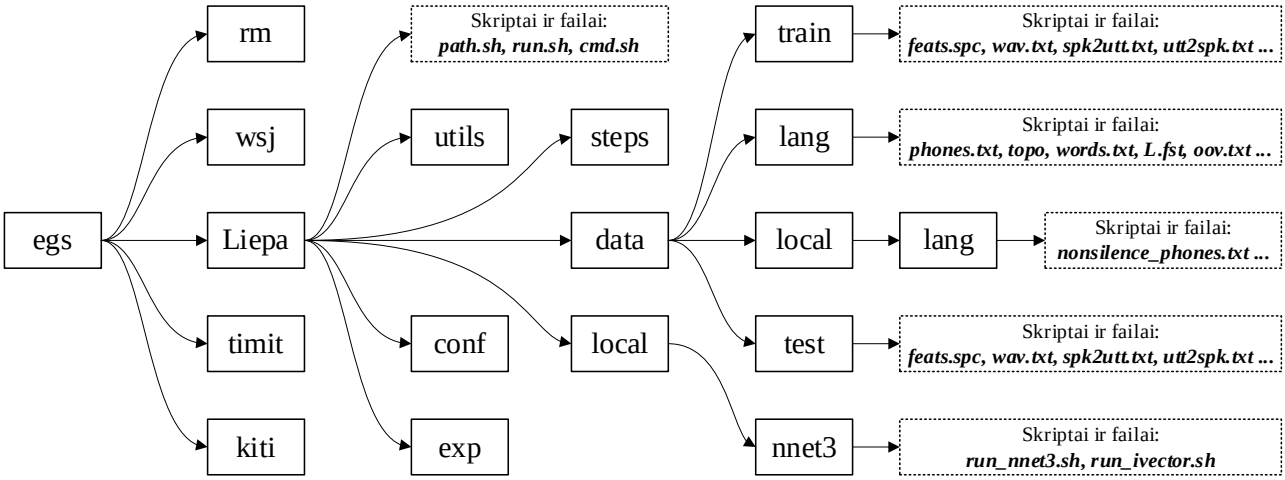
Pirmas sistemos žingsnis yra akustinės ir kalbinės informacijos paruošimas. Akustinės informacijos požymių išskyrimui naudojami Melų dažnių skalės kepstriniai koeficientai (MFCC). Sudarius kalbos modelį yra kuriamas pirmas paprastas HMM-GMM monofoninis modelis, kuriame HMM būsenos modeliuoja nuo konteksto nepriklausomas fonemas. Šiam modeliui taikomas priverstinis mokymo imties duomenų lygiavimas, siekiant apytiksliai įvertinti fonemų ribas, kurios yra reikalingos sudėtingesnių modelių mokymui. Po lygiavimo sudaromas trifoninis modelis, kurio požymiams taikoma tiesinė diskriminantinė analizė (toliau LDA) ir didžiausios tikimybės linijinė transformacija (toliau MLLT). Paskutinis HMM-GMM modelis yra gaunamas pritaikius diktorių adaptyvųjį mokymą (SAT), LDA+MLLT modeliui. DNN modelio kūrimas dėl patogumo ir sistemiškumo yra padalintas į 15 atskirų lygmenų *L01-15*. Pirmi 8 lygmenys yra skirti duomenų iškraipymui ir i-vektorių išskleidimui. Likę žingsniai apima „gardelių“ kūrimą, specialios fonemų topologijos sudarymą, sprendimų medžio sudarymą, architektūros konfigūraciją, DNN tinklo mokymą, dekodavimo grafo sudarymą ir rezultatų dekodavimą. Sekančiuose poskyriuose pateikiama 7 pav. pavaizduotos sistemos Kaldi programinio paketo aplankų struktūra, aprašomi MFCC požymių ir kalbos modelio sudarymo žingsniai, pateikiami HMM-GMM metodų ir DNN sistemos lygmenų *L01-15* aprašymai.

2.1. ASR sistemos aplankų struktūra

Kaldi paketo aukščiausio lygio aplankai yra tokie:

- *egs* – pavyzdžių ir projektų aplankas;
- *src* – pirminio kodo aplankas;
- *tools* – įrankiai, būtini Kaldi paketo funkcionavimui;
- *misc* – pagalbinis aplankas įvairiai informacijai;
- *windows* – „Windows“ operaciniai sistemai skirtų failų aplankas.

Šiame darbe naudojamas *egs* aplankas ir *src* aplankas, kuriame yra patalpintas modelio iškviečiamų skriptų veikimui būtinas pirminis kodas. Sukurto modelio pagrindinis projektas yra laikomas *egs* aplanke, kurio struktūra yra parodyta 8 pav.



8 pav. Pavyzdžių ir projektų aplanko *egs* struktūra [30].

Aplanke *egs* taip pat yra pateikiamos pavyzdinės ASR sistemos, sukurtos anglų ir kitų kalbų populiariausiems garsynams, pvz., „*Wall Street Journal*“ (*wsj*), „*TIMIT*“ ir „*Resource Management*“ (*rm*) garsynams [30]. Visų trijų Liepa garsyno dalių modeliai yra laikomi aplanke *kaldi/egs/Liepa*. Šie modeliai yra vienodi, tik skiriasi jų akustinė ir kalbinė informacija.

Kiekvieno Modelio viršutiniame aplanke yra pagrindinis programos paleidimo skriptas *run.sh*, komandų skriptas *cmd.sh*, skirtas kelių kompiuterių naudojimo valdymui, nuorodų į failų vietas diske skriptas *path.sh*, ir reikalingi standartiniai poaplankiai:

- *utils* – patalpinami įvairūs Kaldi įrankiai, kurie naudojami modelyje;
- *steps* – patalpinami naudojami Kaldi bibliotekų skriptai;
- *conf* – patalpinami atskirų skriptų konfigūracijos failai;
- *local* – patalpinami DNN modelio vykdymo skriptai;
- *data* – patalpinama akustinė ir kalbinė informacija;
- *exp* – patalpinama visų mokymo ir lygiavimo skriptų išėjimo informacija.

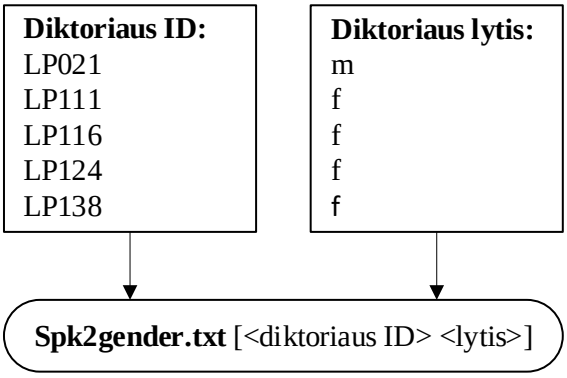
Sudarius būtinų Kaldi paketo aplankų struktūrą, visi tolimesniuose poskyriuose aprašomi sukurtos ASR sistemos failai, yra laikomi *data* ir *exp* aplankuose.

2.2. MFCC požymių išskyrimas ir kalbos modelio kūrimas

Pagrindinių MFCC požymių išskyrimui reikalinga tinkamu formatu paruošta akustinė informacija. Nepriklausomai nuo garsyno formos ir dydžio, Kaldi pakete yra nurodytas standartinis formatas, kaip reikia paruošti šią informaciją, kuri yra laikoma tekstinio formato failuose:

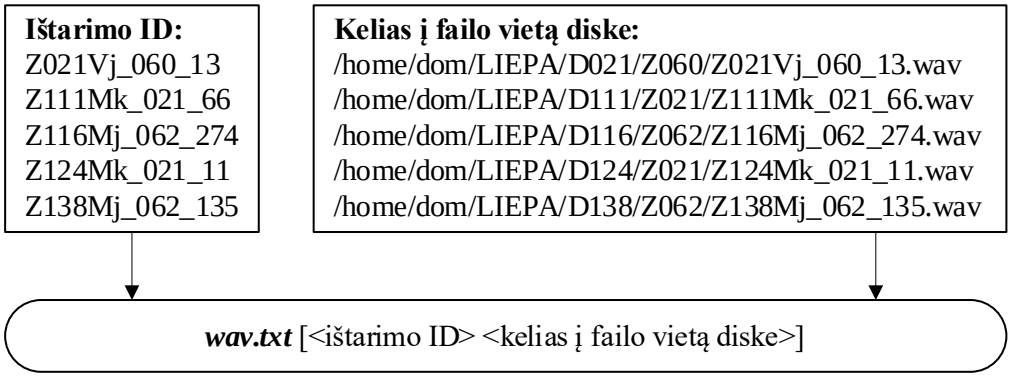
- *spk2gender.txt* [<diktoriaus ID> <lytis>];
- *wav.txt* [<ištarimo ID> <kelias į failo vietą diske>];
- *text.txt* [<ištarimo ID> <teksto transkripcija>];
- *utt2spk.txt* [<ištarimo ID> <diktoriaus ID>];
- *corpus.txt* [<teksto transkripcija>];
- *spk2utt.txt* [<diktoriaus ID> <ištarimo ID 1> <ištarimo ID 2> ... <ištarimo ID n>].

Pirmi 5 ankščiau nurodyti failai yra sudaromi rankiniu būdu, paskutinis failas yra sugeneruojamas naudojant Kaldi modulį *utt2spk_to_spk2utt.pl*. Failas *corpus.txt* yra bendras mokymo ir dekodavimo imčiai, o likę failai yra kuriami atskirai skirtingoms duomenų imtims. Failų paruošimas atliekamas 3 kartus, kiekvienai garsyno daliai atskirai.



9 pav. Failo *spk2gender.txt* struktūra.

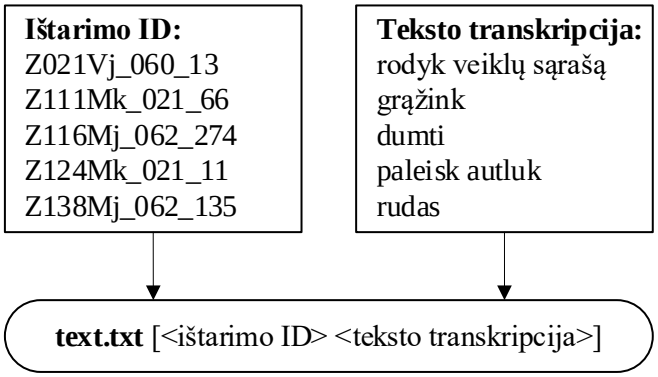
Faile *spk2gender.txt* nurodoma diktoriaus lytis. Visi diktoriai turi unikalų identifikacijos numerį kuriam yra priskiriamas lyties trumpinys (f – moteris, m – vyras). Failo ištrauka pateikta 9 pav.



10 pav. Failo *wav.txt* struktūra.

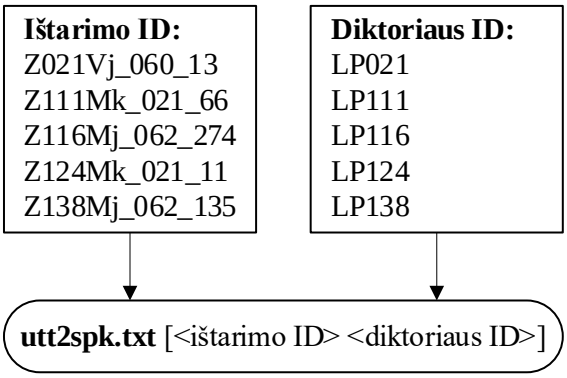
Failas *wav.txt* sujungia kiekvieną ištarimą (sakinys ar žodis ištartas per tam tikrą įrašinėjimo sesiją) su jį atitinkančiu garso failu, esančiu kompiuterio diske. Pasirinkta tokia pavadinimų schema, kad kalbos ID sutaptų su garso failo pavadinimu diske. Failo ištrauka matoma 10 pav.

Faile *text.txt* susiejamas kiekvienas diktoriaus ištarimas su jį atitinkančia teksto transkripcija.



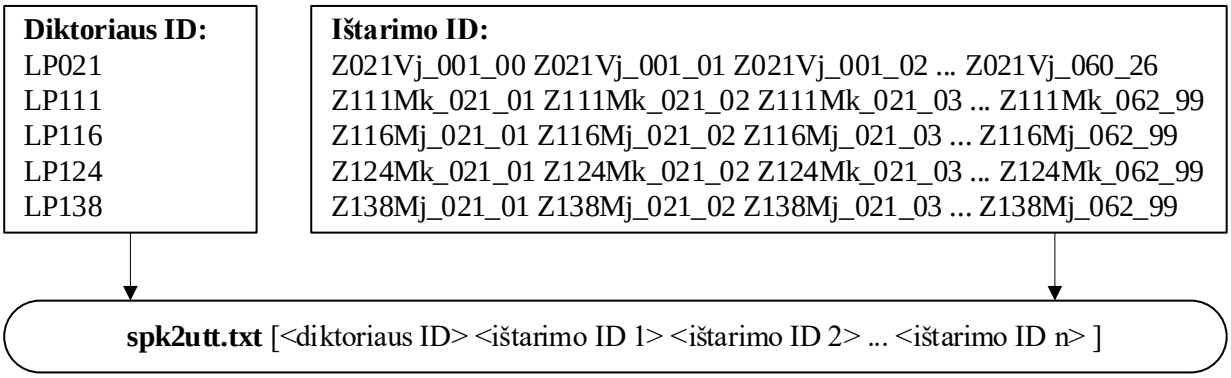
11 pav. Failo *text.txt* struktūra.

Priklausomai nuo tiriamos garsyno dalies, teksto transkripcija gali būti vienas žodis arba žodžių junginys (L_Zod dalis), žodžių seka (L_Sek dalis) arba rišlus sakinys (L_Sak dalis). Failo *text.txt* ištrauka iš L_Zod garsyno dalie pateikta 11 pav.



12 pav. Failo *utt2spk.txt* struktūra.

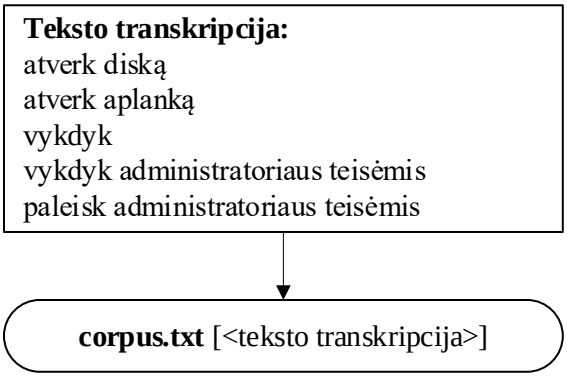
Failas *utt2spk.txt* nurodo ASR sistemai, kuris ištarimas priklauso atitinkamam diktoriui. Visi diktoriai atlieka daugiau nei po vieną ištarimą. Failo *utt2spk.txt* ištrauka pateikta 12 pav.



13 pav. Failo *spk2utt.txt* struktūra.

Faile *spk2utt.txt* kiekvienam diktoriui yra priskiriami visi jo ištarimai. Visi diktoriai atliko skirtingą skaičių ištarimų, todėl kintamojo *n* ribos yra neapibrėžtos ir yra skirtingos kiekvienam diktoriui. Aprašyto failo ištrauka pateikta 13 pav.

Paskutinis paruošiamas akustinės informacijos failas yra *corpus.txt*, kuriame yra surašoma kiekviena teksto transkripcija atsirandanti visuose ASR sistemos diktorių ištarimuose. Tokio failo ištrauka parodyta 14 pav.



14 pav. Failo *corpus.txt* struktūra.

Atskirų garsyno dalių akustinės informacijos kiekis ir pasiskirstymas tarp mokymo ir dekodavimo imčių yra nevienodas. Ši informacija pateikta 1 lentelėje. Daugiausia unikalių ištarimų atliekama L_Zod dalyje, o daugiausia diktorių yra L_Sak dalyje. Visais atvejais mokymo duomenų yra daugiau nei dekodavimo duomenų ir jų santykis apytiksliai yra 1/5.

1 lentelė. Diktorių ir ištarimų skaičiaus pasiskirstymas mokymo ir dekodavimo imtyse.

	Liepa žodžiai L_Zod		**Liepa sekos L_Sek**		**Liepa sakiniai L_Sak**	
	Mokymas	**Dekodavimas**	**Mokymas**	**Dekodavimas**	**Mokymas**	**Dekodavimas**
Diktoriai	145	37	136	44	237	55
Ištarimų kiekis	22246	5448	2700	1130	17518	4642

Apdorojus garsyno akustinę informaciją, sekančiame ASR sistemos žingsnyje yra išskiriami MFCC požymiai. HMM-GMM modeliui išskiriami žemos rezoliucijos požymiai (13 MFCC vektorių) iš 25 ms ilgio kadrų, su 10 ms poslinkiu tarp jų. Tai atliekama su Kaldi moduliu *make_mfcc.sh*, išskiriant požymius tiek mokymo tiek ir dekodavimo akustinei informacijai.

Stacionarus triukšmas (balso trakto, garso takelio, kambario ir kt. impulsinis atsakas) sukelia sudėtingus, tačiau sistemingus MFCC koeficientų pasiskirstymo pokyčius. Norint sumažinti šį neigiamą efektą ir turėti nuo triukšmo nepriklausomus požymius, taikoma kepstrinė vidurkio ir dispersijos normalizacija (toliau CMVN). CMVN tiesiškai transformuoja kepstrinius MFCC požymių koeficientus taip, kad jie turėtų vienodą segmentinę statistiką (nulinis vidurkis, vienetinė dispersija), nepriklausomai nuo triukšmo [31]. CMVN normalizacija nėra atliekama kiekvienam ištarimui ir yra taikoma diktoriams, todėl šis algoritmas nereikalauja daug resursų. Kaldi pakete požymių normalizacija vykdoma su Kaldi moduliu *compute_cmvn_stats.sh*, kuris taikomas tiek mokymo tiek ir testavimo imties akustinės informacijos požymiams.

Klaidos atsiradusios duomenų paruošimo ir požymių išskyrimo metu, gali kelti problemas vėlesniuose programos vykdymo žingsniuose, todėl naudojamas specialus Kaldi modulis *fix_data_dir.sh*, kuris ištrina akustinės informacijos elementus kuriems trūksta vienos iš reikalingų dalių, pavyzdžiui MFCC požymių.

Apdorojus akustinę informaciją ir išskyrus jos požymius, prieš kuriant ASR sistemos kalbos modelį, reikia apdoroti kalbinę informaciją, todėl ranka papildomai paruošiami dar 3 tekstiniai failai:

- *nonsilence_phones.txt [<fonema>]*
- *silence_phones.txt [<fonema>]*
- *lexicon.txt [<žodis> <fonema 1> <fonema 2> ... <fonema n>]*

n	Ie	k'	n'	d'	R	Aa	o	dZ'	J	L	ee	uO	eE	h'	x'
U	a	e	S'	r	k	uu	w	ea	z	Ee	O	f'	L'	dZ	
l'	d	t	s'	m	E	Ea	j'	g	Z'	f	uU	iE	R'	ts	
i	t'	u	ii	N	A	j	m'	aa	b	N'	b'	z'	Uu	tS	
s	r'	I	S	tS'	M	p	v	W	Ii	Z	oO	x	dz	dz'	
v'	iI	p'	uo	eA	oo	aA	g'	l	Oo	Uo	ie	ts'	h	M'	

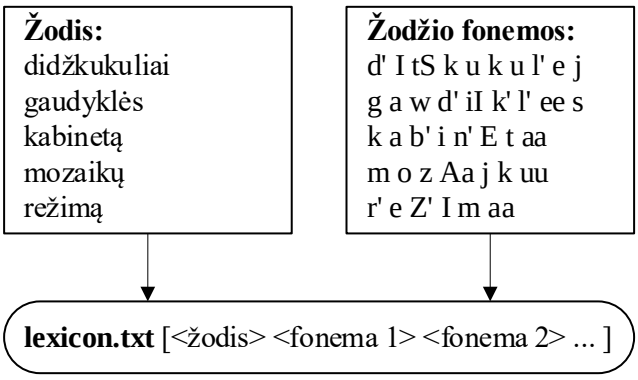
15 pav. Garsyno Liepa fonemos.

Garsyne Liepa naudojama 91 fonema. Faile *nonsilence_phones.txt* surašomos visos 15 pav. matomos fonemos. Lietuvių kalboje yra tik 67 fonemos, todėl garsyne yra 24 papildomos unikalios fonemos.

Faile *silence_phones.txt* surašomos 6 „tylos“ fonemos, kurios atitinka įvairius natūralios kalbos elementus:

- pauzę;
- įkvėpimą;
- iškvėpimą;
- čepsėjimą;
- nurijimą;
- tylą.

Faile *lexicon.txt* surašomas kiekvienas Liepa garsyno žodis ar sakinys, išskaidytas į fonemas. L_Zod garsyno dalies šio failo ištrauka matoma 16 pav.



16 pav. Failo *lexicon.txt* struktūra.

Šie žodžiai sudaro sistemos žodyną, kuris apima 13076 žodžius. Šis failas yra vienodas visoms garsyno dalims (L_Sak, L_Sek, L_Zod). Sugeneravus pirminius tris failus (*nonsilence_phones.txt, silence_phones.txt, lexicon.txt*), su Kaldi moduliu *prepare_lang.sh* pridedamos nuo žodžio vietos priklausančios fonemos ir kiti išvestiniai failai, kurie yra reikalingi kalbos modelio kūrimui. Kalbos modeliai apima įvairių statistinių ir tikimybinių metodų naudojimą tam tikros žodžių sekos būvimo sakinyje tikimybei nustatyti.

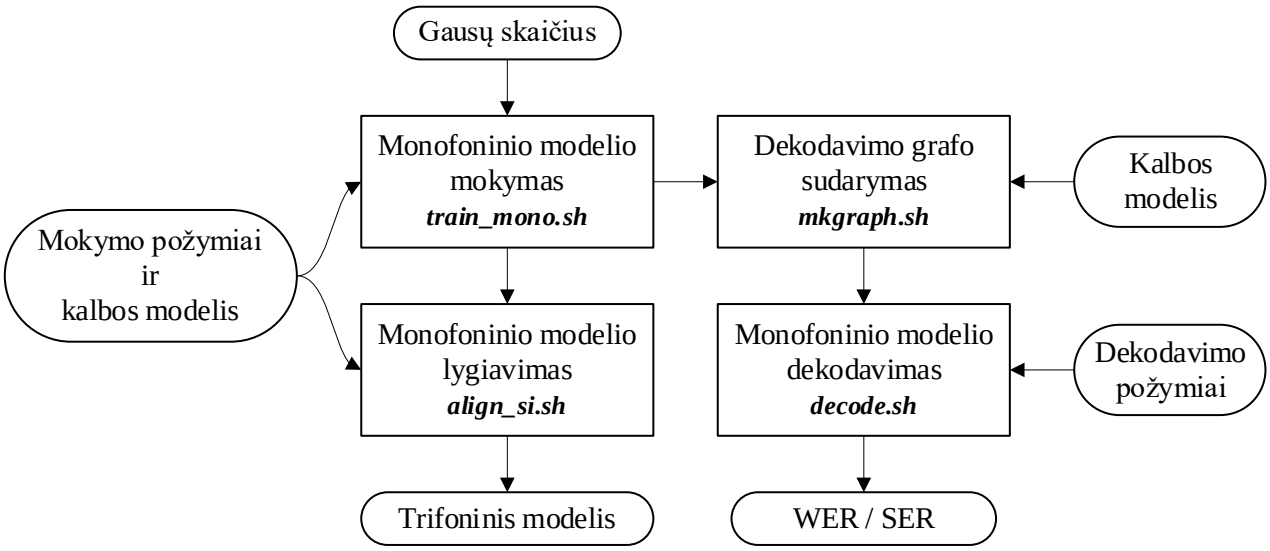
Praktikoje dažniausiai naudojami dviejų tipų kalbos modeliai:

- statistiniai modeliai;
- neuroniniai modeliai.

Statistiniai kalbos modeliai naudoja tradicinius statistinius metodus, pvz., n-gramų ar HMM modelius, kad išmoktų žodžių pasiskirstymo tikimybes, o neuroniniai modeliai kalbos modeliavimui naudoja įvairius neuroninius tinklus. Sukurtoje ASR sistemoje naudojamas n-gramų statistinis modelis. Kalbos modelio sudarymui naudojamas Stanfordo tyrimų instituto kalbos modelio (toliau SRILM) kūrimo įrankis. SRILM yra C++ bibliotekų, vykdomųjų programų ir pagalbinių skriptų rinkinys, skirtas kurti ir eksperimentuoti su statistiniais kalbos modeliais, skirtais kalbos atpažinimui ir kitoms programoms [32]. SRILM įrankis sukuria kalbos modelį kaip „*ARPA*“ tipo duomenų failą, kuriame tikimybės yra išreikštos logaritmu su pagrindu 10. Ši tikimybių vaizdavimo forma paprastai yra kompaktiškesnė, nes tikimybės gali būti labai mažos.

2.3. HMM-GMM modeliai

Išskyrus MFCC požymius ir sudarius kalbos modelį, ASR sistemos akustinio modelio parametrų įvertinimo optimizavimas atliekamas kartojant eilę HMM-GMM modelių mokymo ir lygiavimo žingsnių. Sulygiavus garso įrašus ir tekstą su paskutiniu akustiniu modeliu, gautą modelį galima naudoti kaip kitų mokymo algoritmų įėjimą, todėl po kiekvieno HMM-GMM modelio atliekamas lygiavimas, kuris sulygiuoja garso įrašus ir tekstą iš naujo. Pagrindinis lygiavimo algoritmas visada naudojamas tas pats, t.y. skirtingiems Kaldi įrankiams tinka skirtingų akustinių modelių įėjimo duomenys. Pvz., nuo diktoriaus nepriklausantis lygiavimas lygiavimo metu praleis diktoriaus specifinę informaciją. Po SAT metodo, DNN modelio mokymui vietoje pradinių požymių naudojami pagal diktorius normalizuoti požymiai, todėl norint atlikti lygiavimą, iš požymių yra pašalinama diktorių tapatybė.

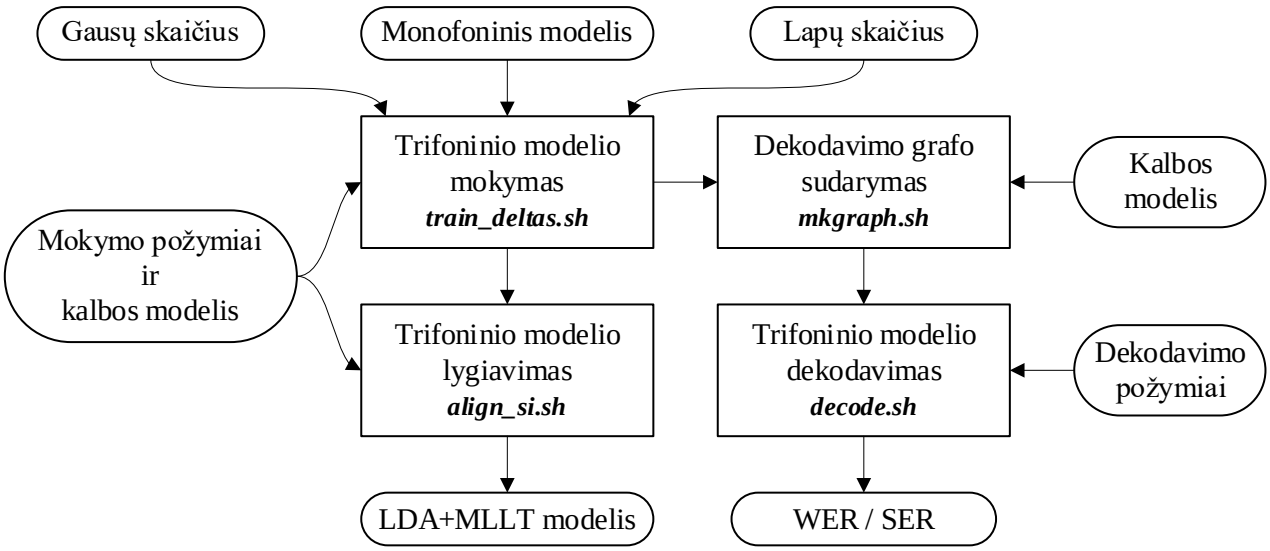


17 pav. Monofoninio modelio mokymo struktūra.

Pirmas HMM-GMM akustinis modelis yra monofoninis, kuriame nėra jokios kontekstinės informacijos apie ankstesnę ar sekančią fonemą. Jis naudojamas kaip trifoninių modelių, kuriuose naudojama kontekstinė informacija, mokymo pagrindas. Monofoninio modelio realizacijos, Kaldi pakete, struktūra parodyta 17 pav.

Modelio mokymas vykdomas su Kaldi moduliu *train_mono.sh*, kuriam reikalinga mokymo akustinė informacija su išskleistais požymiais, kalbos modelis ir Gauso mišinių skaičius, kurį galima keisti, siekiant optimizuoti modelio rezultatus. Pagal apmokytą modelį ir kalbos modelį, su Kaldi moduliu *mkgraph.sh*, kuriamas HCLG dekodavimo grafas, skirtas Kaldi dekodavimo moduliui *decode.sh*, kuris pagal dekodavimo imties akustinės informacijos požymius, skaičiuoja žodžių klaidų dažnį (toliau WER) ir sakinių klaidų dažnį (toliau SER). Norint naudoti monofoninio modelio išėjimą sekančiam HMM-GMM modeliui, atliekamas lygiavimas, su Kaldi moduliu *align_si.sh*, kuris pagal pradinę akustinę informaciją ir kalbos modelį sulygiuoja garso įrašus ir tekstą iš naujo.

Monofoniniai akustiniai modeliai atspindi vienos fonemos akustinius parametrus, tačiau yra žinoma, kad fonemos labai skirsis priklausomai nuo konkretaus konteksto. Trifoniniai modeliai vaizduoja fonemas dviejų kitų (kairiosios ir dešiniosios) fonemų kontekste. Pasirinktas *delta + delta-delta* mokymo algoritmas, kuris papildo MFCC požymius papildomais dinaminiais koeficientais (*delta* ir dviguba *delta*). Šie koeficientai yra požymių pirmos ir antros eilės išvestinių skaitiniai įverčiai. Koeficientas *delta* apskaičiuojamas pagal pirminius MFCC požymius, o koeficientas *delta-delta* apskaičiuojamas pagal naujai gautą *deltą* koeficientą. Trifoninio modelio mokymo struktūra, Kaldi pakete, parodyta 18 pav.



18 pav. Trifoninio modelio mokymo struktūra.

Trifoninis modelis realizuojamas su Kaldi įrankiu *train_deltas.sh*, kurio vykdymui reikia mokymo imties akustinės informacijos požymių, kalbos modelio ir monofoninio modelio sulygiuoto išėjimo. Taip pat įvedami optimizavimo parametrai, t.y. Gauso mišinių skaičius ir HMM klasifikatorių medžių lapų skaičius. Dekodavimo grafo sudarymo, trifoninio modelio dekodavimo ir lygiavimo moduliai yra identiški monofoninėje sistemoje naudotiems Kaldi moduliams.

LDA-MLLT modelis gaunamas trifoniniui modeliui pritaikius linijinę diskriminantinę analizę (LDA) ir didžiausios tikimybės linijinę transformaciją (MLLT). LDA sumažina požymių vektorių išsibarstymą. MLLT transformacija perima LDA požymius ir sukuria unikalią kiekvieno diktoriaus transformaciją, kas leidžia sumažinti skirtumus tarp diktorių, todėl MLLT yra žingsnis link diktorių normalizavimo. LDA-MLLT akustinio modelio realizacija Kaldi pakete yra identiška trifoniniui metodui, tik skiriasi naudojamas modelio mokymo modulis (*train_lda_mllt.sh*), kuriam papildomai nurodomas LDA metodo kadrų dalinimo kontekstas į kairę ir dešinę puses (po 3 fonemas į abi puses).

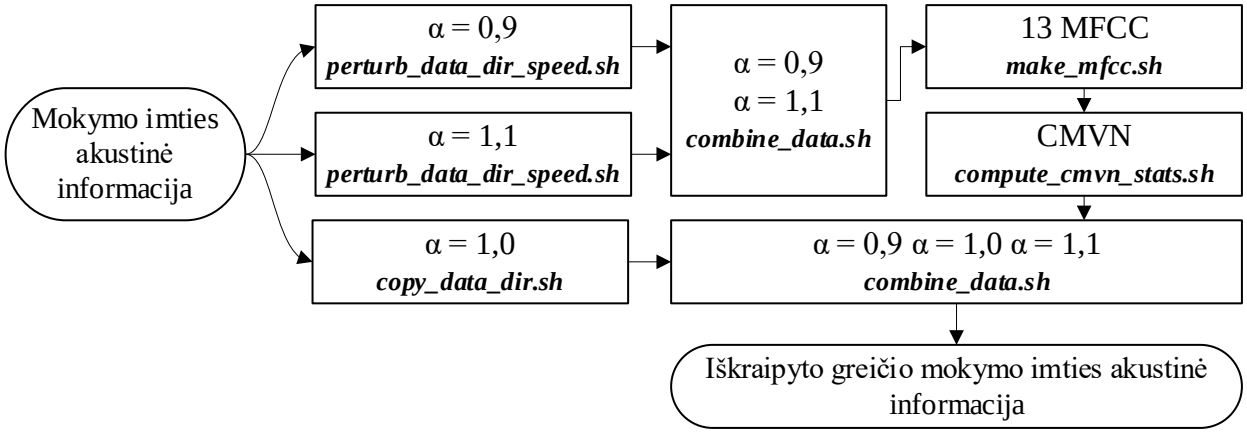
Paskutiniame HMM-GMM modelyje, sulygiuotam LDA+MLLT modelio išėjimui pritaikomas diktorių adaptyvusis mokymas SAT. Kaip ir LDA+MLLT modelis, SAT taip pat atlieka diktorių ir triukšmo normalizavimą, prisitaikydamas prie kiekvieno konkretaus diktoriaus su tam tikra duomenų transformacija. Dėl to gaunami požymiai, nepriklausantys nuo įrašymo aplinkos ir diktorių, kurie bus naudojami DNN modelio mokymui. SAT modelio realizacijos struktūra, Kaldi pakete, yra identiška trifoninio modelio realizacijai, tik skiriasi naudojami Kaldi moduliai mokymui (*train_sat.sh*), dekodavimui (*decode_fmllr.sh*) ir lygiavimui (*align_fmllr.sh*).

2.4. DNN modelis

Sukurtos ASR sistemos DNN modelis yra 15 lygmenų sistema. Dėl klaidų ieškojimo ir lengvesnio parametrų derinimo proceso, sistemos vykdymas gali būti sustabdytas bet kuriame lygmenyje, pvz., pakeitus architektūra lygmenyje *L12*, nereikia iš naujo mokyti HMM-GMM modelių ir vykdyti pirmų 11 DNN modelio lygmenų. Lygmenyse *L01-04* atliekamas akustinės informacijos iškraipymas, *L05-L08* išskleidžiami i-vektoriai, *L09-11* kuriamas sprendimų medis, *L-12* sudaroma neuroninio tinklo architektūra, *L13* vykdomas modelio mokymas, *L14* – kuriamas HCLG dekodavimo grafas ir paskutiniame *L-15* lygmenyje atliekamas dekodavimas po kurio gaunami WER/SER klaidų įverčiai. Toliau pateikiami visų DNN modelio lygmenų paaiškinimai.

2.4.1. Duomenų paruošimas neuroninio tinklo mokymui

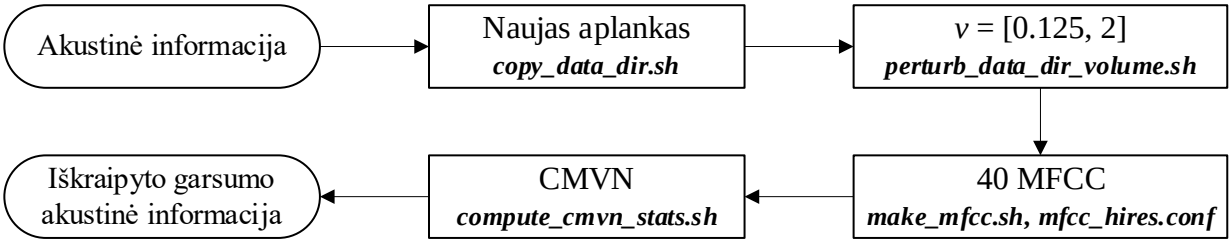
Akustinės informacijos greičio keitimo lygmuo *L01* naudojamas siekiant padidinti mokymo duomenų kiekį, išvengti per didelio modelio prisitaikymo prie specifinių duomenų ir pagerinti jo universalumą [33]. Lygmenyje *L01* keičiamas akustinės informacijos greitis, sukuriant 3 skirtingas originalios akustinės informacijos versijas, su skirtingais greičio koeficientais α (α = 0,9; α = 1,0; α = 1,1).



19 pav. Akustinės informacijos greičio keitimo lygmens *L01* struktūrinė schema.

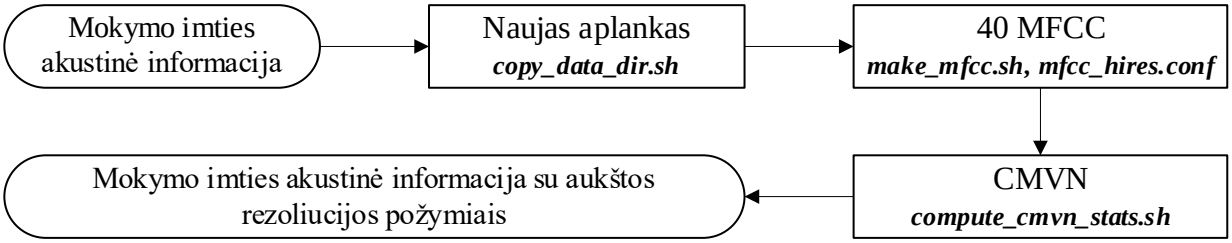
Naudojant įėjimo duomenis su Kaldi paketo moduliu *perturb_data_dir_speech.sh* sukuriama dviejų skirtingų greičių (α = 0,9 ir α = 1,1) akustinė informacija, kuri sujungiama su Kaldi moduliu *combine_data.sh* (žr. 19 pav.). Iškraipytiems duomenims išskiriami žemos rezoliucijos 13 dimensijų MFCC požymiai, su Kaldi moduliu *make_mfcc.sh*. Požymiams pritaikoma CMVN normalizacija su Kaldi moduliu *compute_cmvn_stats.sh* ir gautas rezultatas sujungiamas su originalaus greičio (α = 1,0) mokymo imties akustine informacija. Lygmenyje *L02*, iškraipyto greičio akustinei mokymo informacijai, atliekamas lygiavimas su SAT metodo išėjimu, naudojant Kaldi įrankį *align_fmllr.sh*. Gautas rezultatas bus naudojamas lygmenyje *L11*, kuriame bus sudaromas sprendimų medis.

Lygmenyje *L03* atliekamas mokymo imties akustinės informacijos garsumo atsitiktinis iškraipymas, kuris kaip ir greičio iškraipymas, didina akustinio modelio universalumą. Garsumas keičiamas su atsitiktiniu koeficientu *v*, kurio ribos yra nuo 0,125 iki 2.



20 pav. Akustinės informacijos garsumo iškraipymo lygmens *L03* struktūrinė schema.

Akustinė informacija, su Kaldi moduliu *copy_data_dir.sh*, nukopijuojama į naują aplanką, kuriame, su moduliu *perturb_data_dir_volume.sh,* keičiamas kiekvieno elemento garsumas pagal koeficiento *v* atsitiktinę vertę (žr. 20 pav.). Skirtingai nuo L01 lygmens, čia išskiriami aukštos rezoliucijos 40 dimensijų MFCC požymiai, kuriems pritaikius CMVN normalizaciją, gaunami iškraipyto garsumo išėjimo duomenys. Požymių dimensijos keičiamos atskirame konfigūracijos faile *mfcc_hires.conf*. Jeigu požymių iškyrimo moduliui nenurodomas papildomas konfigūracijos failas, pagal nutylėjimą išskleidžiami žemos rezoliucijos požymiai. Lygmenyje *L03* procesas kartojamas du kartus – su pakeisto ir originalaus greičio mokymo imties akustine informacija.

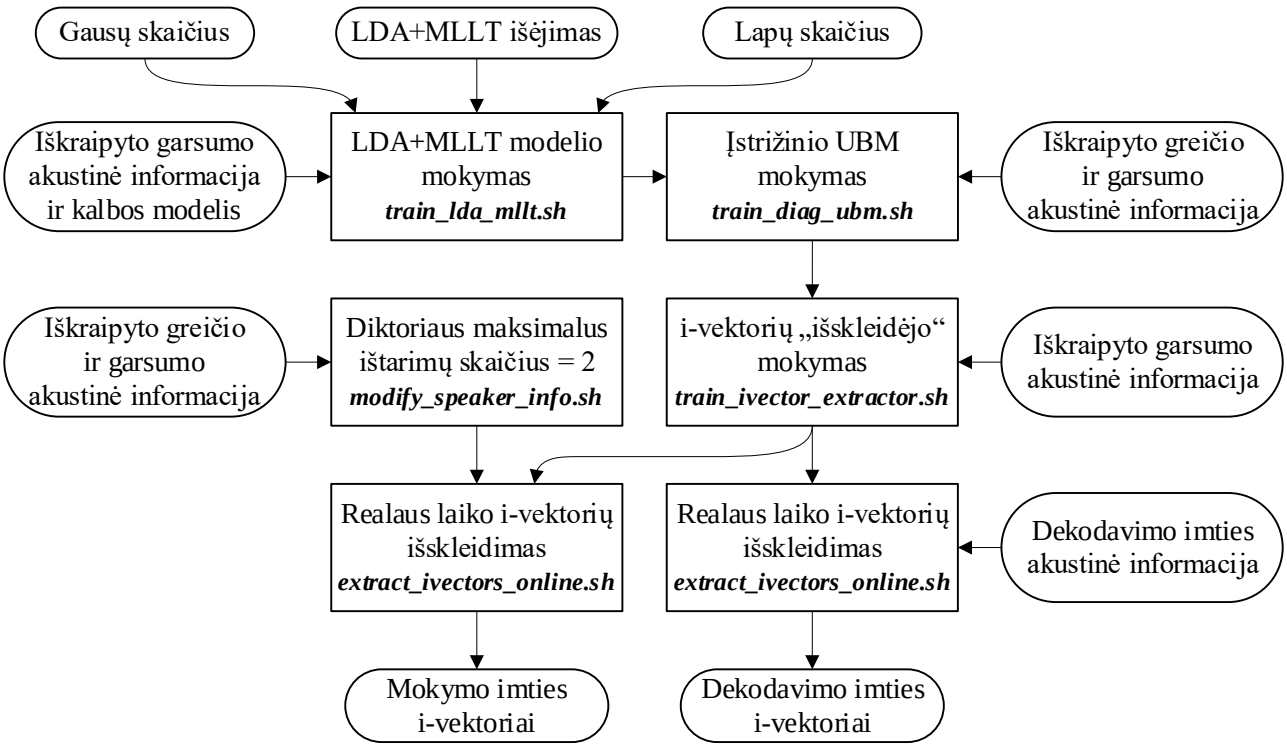


21 pav. Dekodavimo imties paruošimo lygmens *L04* struktūrinė schema.

Lygmenyje *L04*, dekodavimo akustinei informacijai išskiriami aukštos rezoliucijos, 40 dimensijų MFCC požymiai. Pirmiausia, su Kaldi moduliu *copy_data_dir.sh*, nukopijuojama dekodavimo imties originalaus greičio akustinė informacija į naują aplanką ir naudojant tuos pačius modulius, kaip ir *L03* lygmenyje, išskiriami aukštos rezoliucijos MFCC požymiai (žr. 21 pav.).

Modernios diktoriaus atpažinimo sistemos yra paremtos i-vektoriais [34], todėl lygmenyse *L05-08* vykdomas jų išskleidimas. I-vektorius yra mažos dimensijos, Baum-Welch algoritmo (naudojamo nežinomiems HMM parametrams rasti) atvaizdas, gautas iš universalaus fono modelio (UBM) [35], vienoje erdvėje apimančioje visas diktoriaus ir įrašinėjimo sesijos kintamumo charakteristikas [34]. UBM yra didelės apimties GMM modelis, kuris atvaizduoja nuo diktorių nepriklausomų požymių pasiskirstymą [36]. Kaldi pakete i-vektoriai apskaičiuojami iš kintamo dydžio šnekos „gabalo“ (1500 ms, 1000 ms arba 1600 ms), naudojant i-vektorių „išskleidėją“. Šis procesas atliekamas lygmenyse *L05-8*, kurių tarpusavio ryšiai ir struktūra parodyta 22 pav. Lygmenyje *L05* sukuriamas atskiras mažas akustinis modelis, kadangi įstrižinio UBM mokymui (lygmuo *L06*) reikalinga LDA+MLLT transformacija. LDA+MLLT mokymas vykdomas su Kaldi moduliu *train_lda_mllt.sh*, kuriam reikalinga iškraipyto garsumo, bet originalaus greičio akustinė informacija, kalbos modelis, trečio HMM-GMM modelio sulygiuoto išėjimo informacija ir optimizavimo parametrai.

Lygmenyje *L06* su Kaldi moduliu *train_diag_ubm.sh* mokomas įstrižinis UBM, kurio mokymui reikia iškraipyto greičio ir garsumo akustinės informacijos ir *L05* lygmens išėjimo. Pagal sudarytą įstrižinį UBM modelį ir iškraipyto garsumo akustinę informaciją, lygmenyje *L07*, mokomas i-vektorių „išskleidėjas“, naudojant Kaldi modulį *train_ivector_extractor.sh*.



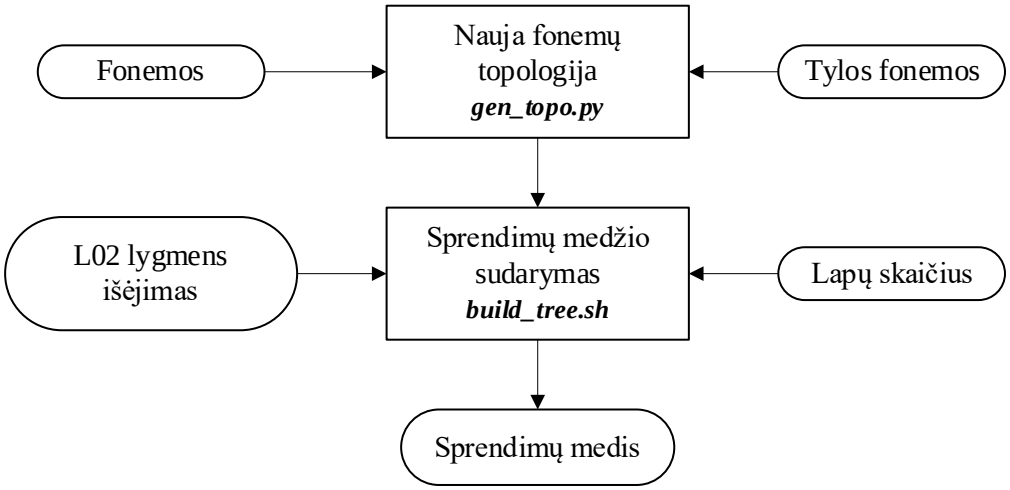
22 pav. Vektorių išskleidimo lygmenų *L05-8* struktūrinė schema.

Didesnis diktorių skaičius yra naudingas bendrų požymių išskyrimui, taip pat leidžia geriau atliktį atskirų ištarimų dekodavimą, todėl lygmenyje *L08* pirmiausia atliekamas diktorių padalijimas į jų pačių kopijas, kurios gali turėti ne daugiau 2 ištarimų. Diktorių kopijos kuriamos tik iškraipytai mokymo akustinei informacijai, naudojant Kaldi modulį *modify_speaker_info*.sh, ir nėra kuriamos dekodavimo imčiai. Lygmenyje *L08*, naudojant Kaldi modulį *extract_ivectors_online.sh*, gaunami 2 išėjimai - mokymo ir dekodavimo imčių i-vektoriai.

Lygmenyje *L09* generuojamos „gardelės“ su juose esančių alternatyvių tarimų lygiavimais. Šnekos atpažinimo „gardelė“ yra pažymėtas, pasvertas, nukreiptas aciklinis grafas, kuriame kiekvienas pilnas kelias reiškia alternatyvią transkripcijos hipotezę, įvertintą pagal atpažinimo balą tam tikram ištarimui [37]. „Gardelės“ yra naudingos, kai norima daugiau nei vienos alternatyvos iš atpažinimo žingsnio. Pvz., kad iš „gardelės“ atrinktumėme geriausią hipotezę, šnekos dekodavimas dalinamas į du žingsnius: pirmąjį, kuris sukuria „gardeles“ naudojant gana paprastus akustinius ir kalbos modelius, ir antrąjį pakartotinio įvertinimo žingsnį, kuriame naudojami sudėtingesni modeliai, kurių naudojimas pirmajame žingsnyje yra neefektyvus [38]. Lygmenyje *L09* su Kaldi moduliu *align_fmllr_lats.sh*, pagal kalbos modelį, apskaičiuojamos „gardelės“ HMM-GMM SAT metodo nesulygiuotam išėjimui, kurios bus naudojamas neuroninio tinklo mokyme.

Lygmenyse *L10-11* yra sudaromas sprendimų medis. Pirmiausia, siekiant kontroliuoti tyliųjų ir netyliųjų HMM modelių būsenų skaičių, lygmenyje *L10* sudaroma speciali Kaldi paketo fonemų topologija, kuri turi tik vieną HMM būseną per fonemą ir naudojama modeliams su neoriniais tinklais.

Lygmenyje *L11,* remiantis nauja topologija, kuriamas sprendimų medis. Abiejų lygmenų struktūra parodyta 23 pav.

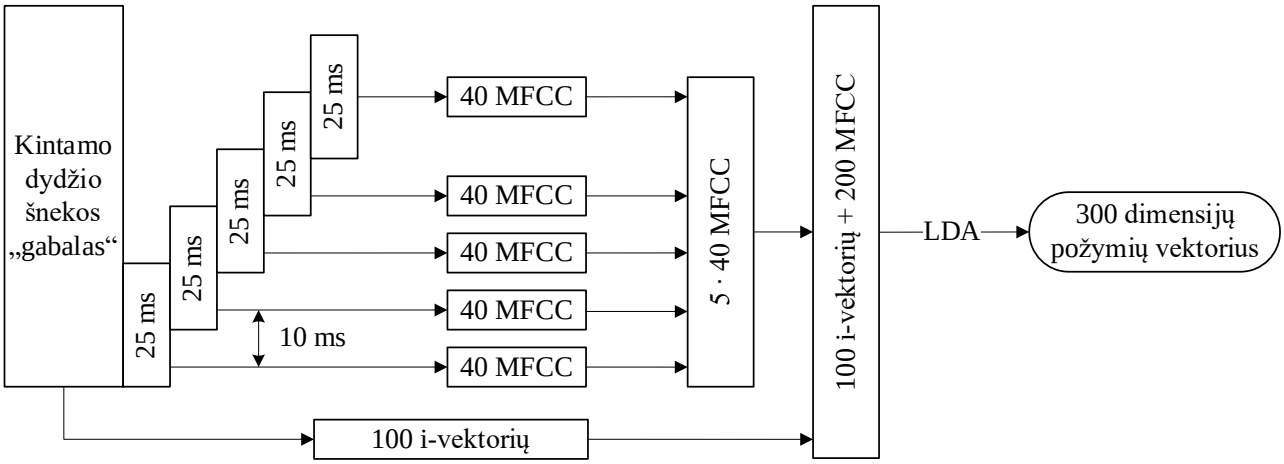


23 pav. Lygmenų *L10-11* struktūra.

Modifikuota topologija kuriama su Kaldi moduliu *gen_topo.py,* kuris pagal fonemas ir tylos fonemas sudaro naują monofoninę topologiją. Sprendimų medis kuriamas pagal *L02* lygmens išėjimą, su Kaldi moduliu *build_tree.sh,* kuris yra identiškas HMM-GMM sistemos modelių mokymo įrankiams, pvz., SAT metodo, tik šis įrankis papildomai atlieka lygiavimą su *L10* lygmens monofonine topologija ir sustoja po medžio kūrimo ir modelio inicijavimo etapo, pakartotinai nevertinant Gausų ir iš naujo netreniruojant perėjimų. Kaip ir kitiems HMM-GMM modeliams, nurodomas HMM klasifikatorių medžių lapų skaičius. Sudarytas sprendimų medis bus naudojamas neuroninio tinklo mokyme.

2.4.2. Neuroninio tinklo architektūra

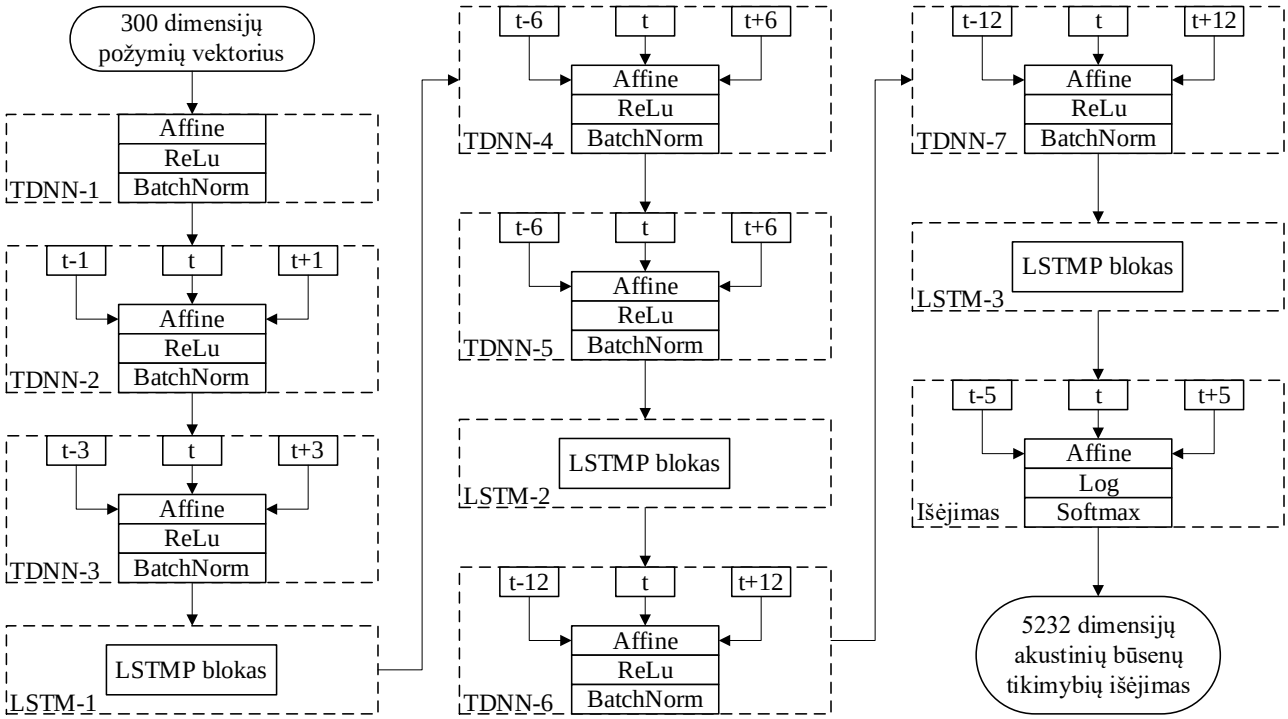
Lygmenyje *L12* sudaroma neuroninio tinklo architektūra. Šiame darbe išbandomos įvairios TDNN LSTM, BLSTM ir jų kombinacijų architektūros, kurių įėjimas yra vienodas ir pavaizduotas 24 pav.



24 pav. Tinklo architektūros įėjimo požymiai.

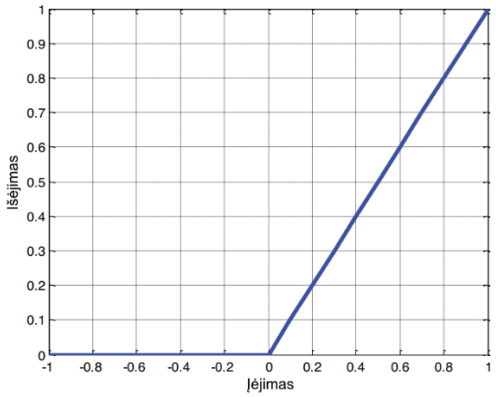
Modelio įėjimas susideda iš dviejų tipų požymių: 40 dimensijų didelės raiškos MFCC požymių, išskirtų iš 25 ms ilgio ir 10 ms poslinkio kadrų ir 100 dimensijų i-vektorių, apskaičiuotų iš kintamo dydžio šnekos įrašo „gabalo“.

Penki iš eilės einantys MFCC vektoriai ir i-vektoriai sujungiami ir sudaro 300 dimensijų esamo kadro požymių vektorių. Norint išsaugoti požymių vektoriaus matmenis, jam taikoma LDA transformacija, todėl modelio įėjimas yra 300 dimensijų požymių vektorius.



25 pav. Hibridinė TDNN-LSTM neuroninio tinklo architektūra.

Visos ištirtos architektūros susideda iš kelių blokų, pvz., 25 pav. pavaizduotoje TDNN-LSTM architektūroje yra 7 TDNN ir 3 LSTM blokai. Šie dviejų tipų blokai yra pagrindas, kiekvienai šiame darbe išbandytai architektūrai. Kaldi pakete BLSTM sluoksnis susideda iš dviejų LSTM sluoksnių, po vieną abiem kryptimis (pirmyn ir atgal), todėl toliau aprašomi tik TDNN ir LSTM blokai.



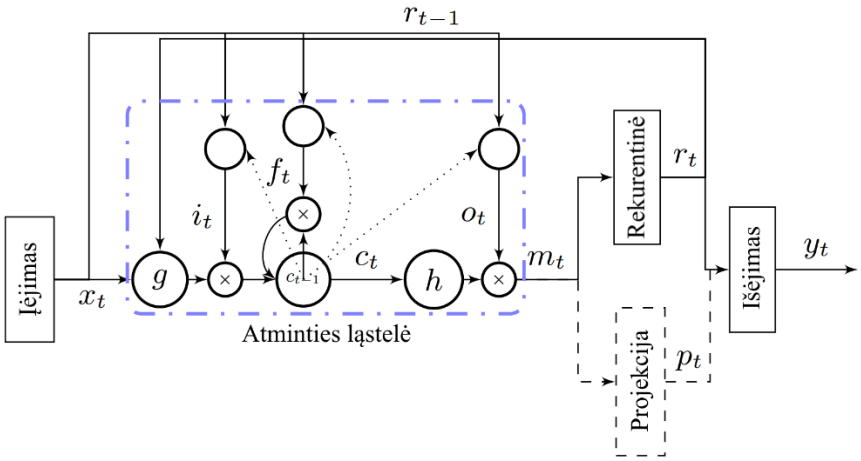
26 pav. *ReLU* aktyvacijos funkcijos grafikas [39].

Kiekvienas TDNN blokas susideda iš tiesinės transformacijos (*affine*) sluoksnio, aktyvacijos (*ReLU*) sluoksnio ir paketinės normalizacijos (*BatchNorm*) sluoksnio. Pirmas TDNN vidinis sluoksnis atlieka linijinę transformaciją iš 300 dimensijų įėjimo į 1024 dimensijų išėjimą. Antrame vidiniame sluoksnyje yra pritaikoma ištaisyta linijinė aktyvacijos funkcija (*ReLU*), kuri yra tiesinė funkcija ir tiesiogiai į išėjimą perduoda funkcijos įėjimą, jeigu jis yra didesnis už 0 [39]. *ReLU* funkcijos grafikas pateiktas 26 pav.

Paskutiniame vidiniame TDNN sluoksnyje, įėjimams taikoma paketinė normalizacija, kuri leidžia sumažinti vidinį kovariacijos poslinkį (toliau ICS). ICS yra apibrėžiamas kaip neuroninio tinklo aktyvacijų pasiskirstymo pokytis dėl tinklo parametrų pasikeitimo mokymo metu [40].

Kaip nurodo TDNN modelio pavadinimas, kiekvienas laiko vėlinimo neuroninio tinklo blokas (išskyrus pirmąjį) atlieka laiko konvoliuciją, t.y. vykdo ankščiau aprašytas matematines operacijas ne tik esamam įėjimo vektoriui, bet ir praeities ir ateities įėjimo vektoriams. Kontekstas tarp atskirų TDNN blokų skiriasi taip:

- TDNN blokas 2 apdoroja įėjimo vektorius laiku t-1, t ir t+1,
- TDNN blokas 3 apdoroja įėjimo vektorius laiku t-3, t ir t+3,
- TDNN blokai 4 ir 5 apdoroja įėjimo vektorius laiku t-6, t ir t+6,
- TDNN blokai 6 ir 7 apdoroja įėjimo vektorius laiku t-12, t ir t+12.



27 pav. LSTM architektūra su papildoma ne rekurentine išėjimo projekcija [41].

Pirmas 25 pav. architektūros LSTM-1 blokas yra vienakryptis rekurentinis sluoksnis, kurio įėjimas yra 1280 dimensijų vektorius (3 TDNN bloko išėjimas), išėjimas – 1536 dimensijų vektorius. Klasikinis LSTM tinklas susieja įėjimo seką $x = (x_1, ..., x_T)$ su išėjimo seka $y = (y_1, ..., y_T)$, apskaičiuodamas tinklo vienetų aktyvavimus laike nuo t = 1 iki T, pagal šias lygtis [41]:

$$i_t = \sigma(W_{ix}x_t + W_{im}m_{t-1} + W_{ic}c_{t-1} + b_i) \quad (1)$$

$$f_t = \sigma(W_{fx}x_t + W_{mf}m_{t-1} + W_{cf}c_{t-1} + b_f) \quad (2)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g(W_{cx}x_t + W_{cm}m_{t-1} + b_c) \quad (3)$$

$$o_t = \sigma(W_{ox}x_t + W_{om}m_{t-1} + W_{oc}c_t + b_o) \quad (4)$$

$$m_t = o_t \odot h(c_t) \quad (5)$$

$$y_t = W_{ym}m_t + b_y \quad (6)$$

Čia W žymi svorių matricas (pvz., nuo įėjimo „vartų“ iki įėjimo svorių matrica yra W_{ix}); b žymi biaso vektorius (pvz., b_i yra įėjimo „vartų“ biaso vektorius); σ yra logistinė sigmoidinė funkcija; i, f, o ir c atitinkamai yra įėjimo „vartų“, užmiršimo „vartų“, išėjimo „vartų“ ir ląstelių aktyvacijos vektoriai, kurie visi yra tokio paties dydžio kaip ir ląstelės išėjimo aktyvacijos vektorius m; simbolis ⊙ reiškia per kiekvieną elementą atliekamą sandaugą; g ir h yra ląstelės įėjimo ir išėjimo aktyvacijos funkcijos, kurios dažniausiai yra hiperbolinės tangentinės funkcijos (*tanh*). Kaldi pakete yra naudojama patobulinta LSTM architektūra, parodyta 27 pav. (rodomas vienas atminties blokas dėl aiškumo) [41].

Patobulinta architektūra leidžia sumažinti mokymo skaičiavimų sudėtingumą. Tai pasiekiama prijungus LSTM ląstelės išėjimą prie papildomos rekurentinės projekcijos, kuri per grįžtamąjį ryšį yra prijungta prie ląstelės įėjimo ir visų modelio „vartų“. Taip pat šalia rekurentinės projekcijos pridedamas ne rekurentinis sluoksnis, kuris yra tiesiogiai prijungtas prie išėjimo sluoksnio. Tai leidžia padidinti projekcinių sluoksnių išėjimo dydį, nepadidinant rekurentinių sujungimų parametrų skaičiaus. Atlikus šiuos pakeitimus LSTMP (p raidė reiškia papildomas projekcijas) architektūros lygtys yra tokios [41]:

$$i_t = \sigma(W_{ix}x_t + W_{ir}r_{t-1} + W_{ic}c_{t-1} + b_i) \quad (7)$$

$$f_t = \sigma(W_{fx}x_t + W_{rf}r_{t-1} + W_{cf}c_{t-1} + b_f) \quad (8)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g(W_{cx}x_t + W_{cr}r_{t-1} + b_c) \quad (9)$$

$$o_t = \sigma(W_{ox}x_t + W_{or}r_{t-1} + W_{oc}c_t + b_o) \quad (10)$$

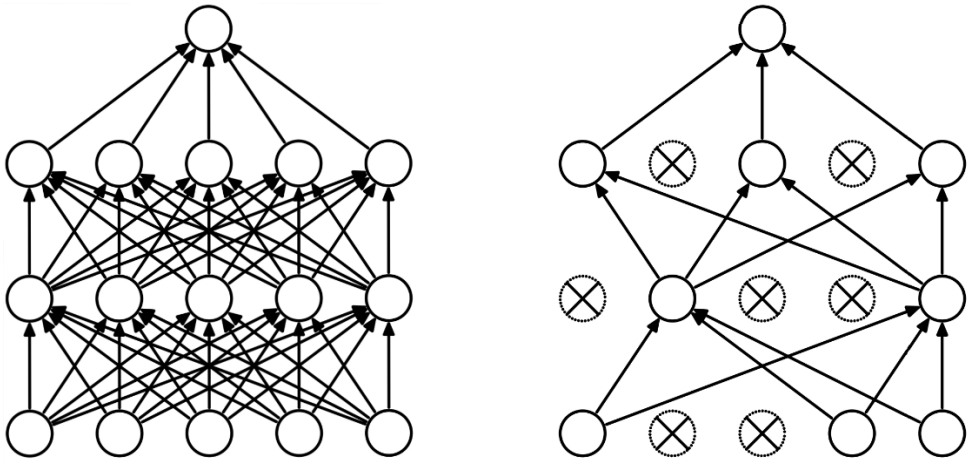
$$m_t = o_t \odot h(c_t) \quad (11)$$

$$r_t = W_{rm}m_t \quad (12)$$

$$p_t = W_{pm}m_t \quad (13)$$

$$y_t = W_{yr}r_t + W_{yp}p_t + b_y \quad (14)$$

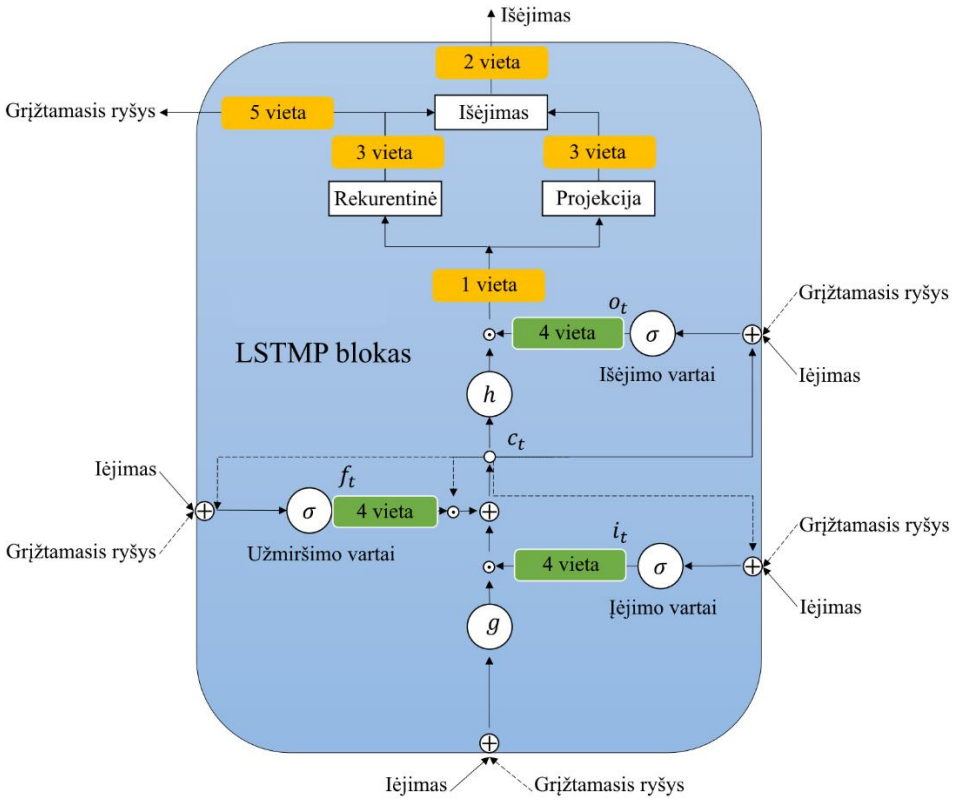
Čia *r* ir *p* žymi rekurentinės ir ne rekurentinės projekcijų aktyvacijas. Siekiant dar labiau sumažinti tinklo sudėtingumą, naudojamas „išmetimas“ kuris apsaugo nuo per didelio prisitaikymo ir suteikia galimybę efektyviai naudoti eksponentiškai daug skirtingų neuroninių tinklų architektūrų [42].



28 pav. Neuroninis tinklas su „išmetimu“ [42].

Terminas „išmetimas“ reiškia tinklo elementų (paslėptų ir nepaslėptų) laikiną pašalinimą iš tinklo kartu su visais jo įeinančiais ir išeinančiais ryšiais, kaip parodyta 28 pav. Pasirinkimas, kuriuos elementus išmesti, yra atsitiktinis. Paprasčiausiu atveju kiekvienas elementas išlaikomas su fiksuota tikimybe *p*, nepriklausoma nuo kitų elementų, kur *p* galima pasirinkti naudojant eksperimentus arba tiesiog nustatyti 0,5, kas yra artima optimaliam pasirinkimui daugeliui tinklų ir užduočių [42]. Kaldi kūrėjų atliktas tyrimas [43] rodo, kad toks „išmetimo“ parametrų pasirinkimas yra neefektyvus, o tikimybę *p* reikia pritaikyti kiekvienam modeliui atskirai. Taip pat rekomenduojama naudoti laike besikeičiantį „išmetimo“ tvarkaraštį, kurio tikimybė *p* pradžioje ir pabaigoje yra 0, ir pasiekia savo piką arčiau mokymo pradžios.

Šiame modelyje epochų skaičius yra iš anksto apibrėžtas, todėl galima „išmetimo“ tvarkaraštį susieti su epochomis. „Išmetimo“ tvarkaraštis Kaldi pakete yra išreikštas dalimis kaip linijinė funkcija intervale [0, 1], kurioje f (0) yra „išmetimo“ tikimybė mokymo pradžioje ir f (1) yra „išmetimo“ tikimybė kuomet modelis jau matė visus mokymo duomenis. Pvz., Kaldi pakete „išmetimo“ tvarkaraštis „*0@0,0.3@0.5,0@1*“ reiškia funkciją f (x), kuri yra tiesiškai interpoliuota tarp taškų f (0) = 0, f (0,5) = 0,3 ir f (1) = 0. Tai sutrumpintai užrašoma „*0, 0.3@0.5, 0*“, kadangi visada pirmas taškas yra x = 0 ir paskutinis taškas yra x = 1. Kaldi kūrėjai rekomenduoja „išmetimo“ grafiką formoje „*0, 0@0.2,p@0.5, 0*“ [43], kur p yra konstanta, derinama kiekvienam modeliui atskirai.



29 pav. LSTMP bloko galimos „išmetimo“ pozicijos [43].

„Išmetimą“ galima taikyti įvairiose vietose (žr. 29 pav.): prieš projekcijas (1 vieta); už bloko išėjimo sluoksnio (2 vieta); už projekcijų (3 vieta); už įėjimo, užmiršimo ir išėjimo „vartų“ (4 vieta); už rekurentinės projekcijos grįžtamojo ryšio. Šiame darbe pasirinkta 4 vieta, todėl 7, 8 ir 10 lygtys yra pakeičiamos tokiomis lygtimis [43]:

$$i_t = \sigma(W_{ix}x_t + W_{ir}r_{t-1} + W_{ic}c_{t-1} + b_i) \odot m_{drop}^{i_t} \tag{15}$$

$$f_t = \sigma(W_{fx}x_t + W_{rf}r_{t-1} + W_{cf}c_{t-1} + b_f) \odot m_{drop}^{f_t} \tag{16}$$

$$o_t = \sigma(W_{ox}x_t + W_{or}r_{t-1} + W_{oc}c_t + b_o) \odot m_{drop}^{o_t} \tag{17}$$

Čia $m_{drop}^{i_t}$, $m_{drop}^{f_t}$ ir $m_{drop}^{o_t}$ yra atitinkamų „vartų“ elementų „išmetimo“ matricos.

Šiame darbe naudojama Kaldi paketo „greita“ LSTMP modifikacija, kuri pagrindinį architektūros netiesiškumą realizuoja kaip vieną komponentą, t.y. sujungia 9, 11, 15, 16, ir 17 formules į bendrą matematinę operaciją, tai gerokai sumažina mokymo laiką. „Greita“ LSTMP modifikacija naudojama visuose trijuose LSTM sluoksniuose (žr. 25 pav.).

Paskutinis 25 pav. parodytos architektūros sluoksnis apdoroja LSTM-3 bloko išėjimo vektorius laiko kontekste t-5, t ir t+5, pritaikydamas *affine* transformaciją, ir *log-softmax* operaciją, kuri konvertuoja išėjimą į tikimybių skirstinį. Galutinis modelio išėjimas yra 5232 dimensijų akustinių būsenų tikimybių skirstinys.

2.4.3. Neuroninio tinklo mokymas ir dekodavimas

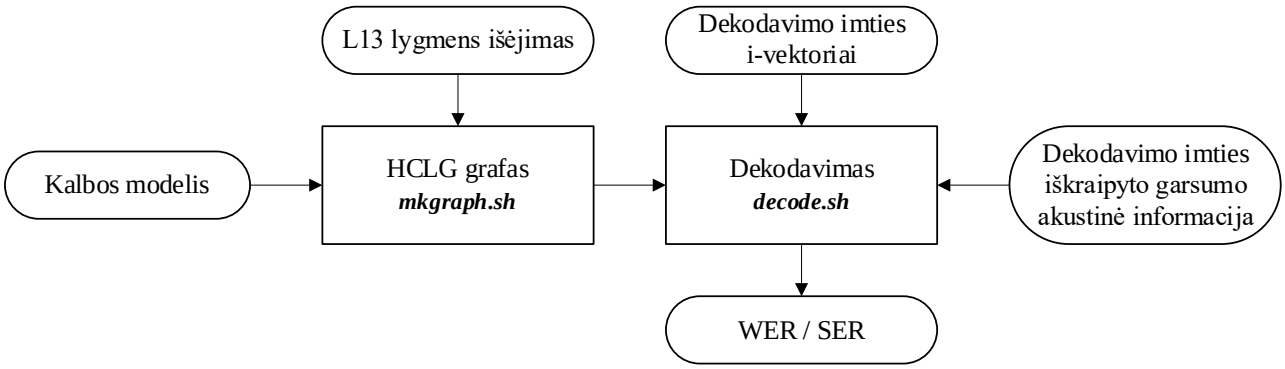
Sudarius tinklo architektūrą, lygmenyje *L13* vykdomas neuroninio tinklo mokymas, naudojant Kaldi modulį *train.py*. Numatytosios parametrų reikšmės ir paaiškinimai pateikti 2 lentelėje.

2 lentelė. Kaldi mokymo modulio *train.py* parametrai.

Parametras	**Paaiškinimas**
leaky-hmm-coefficient = 0.1	Parametras leidžia pereiti tikimybėms iš kiekvienos HMM būsenos į kiekvieną kitą būseną, kad būtų užtikrintas laipsniškas konteksto pamiršimas [44]. Tai prilygsta HMM sustabdymui ir paleidimui iš naujo su tam tikra tikimybe kiekviename kadre.
l2-regularize = $5 \cdot 10^{-4}$	Parametras, kuris supaprastiną modelį ir padeda išvengti per didelio prisitaikymo. Pasirinktas L_2 sureguliavimas, nes L_1 sureguliavimas turi tendenciją sumažinti svorius iki 0, t.y. atsiranda tikimybė, kad neteksime dalies įėjimų [45]. L_2 sureguliavimas gali sumažinti svorius iki labai žemų verčių, bet niekada iki 0.
dropout-schedule = '0,0@0.20,0.1@0.50,0'	Nurodomas LSTMP „išmetimo“ tvarkaraštis.
frames-per-iter = 1200000	Nurodomas apdorojamų kadrų skaičius vienoje iteracijoje.
max-param-change = 2.0	Maksimalus parametrų pokytis riboja, kiek leidžiama keistis parametrams viename mini pakete, t.y. matrica, vaizduojanti bet kurio tam tikro sluoksnio parametrų pasikeitimą, bet kuriame mini pakete, negali viršyti šios vertės. Jei viršijama parametro vertė, mini paketui naudojamas mokymosi greitis yra padauginamas iš konstantos, mažesnės už vienetą.
num-epochs = 4	Parametras nurodo, kiek kartų modelis bus apmokomas su visais mokymo imties duomenimis.
num-jobs-initial = 1	Nurodomas pradinis lygiagrečių darbų skaičius, šiuo atveju nurodomas 1, kadangi mokymui naudojama 1 vaizdo plokštė.
num-jobs-final = 1	Nurodomas galutinis lygiagrečių darbų skaičius.
initial-effective-lrate = 0.001	Pradinio mokymosi greičio parametras valdo greitį, kuriuo modelis mokosi, t.y. valdo paskirstytos klaidos kiekį, su kuriuo modelio svoriai kiekvieną kartą yra atnaujinami.
final-effective-lrate = 0.0001	Galutinis mokymosi greitis naudojamas siekiant atlikti didelius svorio pokyčius mokymosi proceso pradžioje ir nedidelius pakeitimus mokymosi proceso pabaigoje. Mokymosi greičio vertė nuo pradinės iki galutinės vertės kinta pagal tiesinę funkciją. Galutinė mokymosi greičio vertė dažniausiai yra 10-20 kartų mažesnė už pradinę.
momentum = 0.0	Inertiškumo parametras leidžia svorių atnaujinimo metu įtraukti eksponentinius svertinius ankstesnių svorio atnaujinimų vidurkius. Dėl šio pokyčio stochastinio gradiento nusileidime daugelis ankstesnių atnaujinimų viena kryptimi tęsis ta kryptimi ir ateityje. Dažniausiai naudojamos parametro vertės yra 0.5, 0.9 ir 0.99 [46].
num-chunk-per-minibatch = 64	Kaldi paketas sujungia atskirus mokymo pavyzdžius į mini paketus, kuriuose yra daug skirtingų pavyzdžių (kiekvienas originalus pavyzdys gauna skirtingą "n" indeksą). Kai „gabalų“ dydžiai yra kintami, svarbu užtikrinti, kad tik „panašūs“ pavyzdžiai būtų sujungti į mini paketus, tai leidžia išvengti perkompiliavimo kiekvienam mini paketui. Kaldi sujungia tik tos pačios struktūros dalis, t.y. tokio paties dydžio kairiojo ir dešiniojo konteksto. Šis parametras nurodo mini paketo dimensijas.

Parametras	Paaiškinimas
chunk-width = 140, 100, 160	„Gabalo“ plotis yra kiekvieno duomenų „gabalo“, kurį vertiname mokydami arba dekoduodami, išėjimo kadrų skaičius. Šiame darbe naudojamas kintamo dydžio „gabalo“ plotis kuris leidžia naudoti gana didelius „gabalus“ ir išvengti duomenų praradimo dėl failų, kurie nėra tikslūs „gabalo“ dydžio kartotiniai.
chunk-left-context = 20	Nurodomas kontekstinės informacijos kiekis, kuris yra pridedamas į „gabalą“ kairėje jo pusėje.
chunk-right-context = 20	Nurodomas kontekstinės informacijos kiekis, kuris yra pridedamas į „gabalą“ dešinėje jo pusėje.
use-gpu = true	Nurodoma, kad mokymui naudojama vaizdo plokštė.
online-ivector-dir	Nurodoma mokymo imčiai išskleistų i-vektorių vieta diske (L08 lygmuo)
feat-dir	Nurodoma požymių vieta diske (*L03* lygmens išėjimas).
tree-dir	Nurodoma sprendimų medžio vieta diske (L11 lygmens išėjimas).
lat-dir	Nurodoma „gardelių“ vieta diske (*L09* lygmens išėjimas).
dir	Nurodoma L12 lygmens išėjimo vieta diske.

Apmokius neuroninį tinklą, vykdomi paskutiniai *L14* ir *L15* lygmenys, kuriu struktūra ir tarpusavio ryšiai pavaizduoti 30 pav.



30 pav. Lygmenų *L14-15* struktūra.

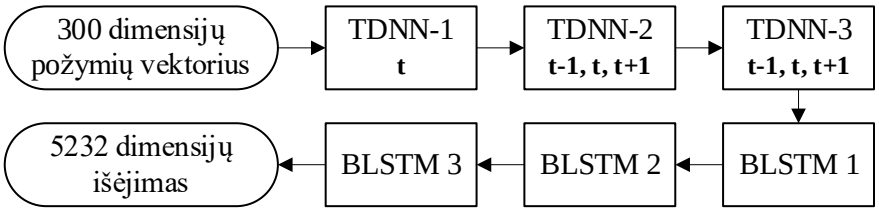
Lygmenyje *L14*, pagal apmokytą neuroninį tinklą ir kalbos modelį, sukuriamas išplėstas HCLG dekodavimo grafas, kurio kūrimui naudojamas Kaldi paketo modulis *mkgraph.sh*. Paskutiniame modelio lygmenyje *L15*, naudojant dekodavimo imties iškraipyto garsumo akustinę informaciją (*L04* lygmens išėjimas), dekodavimo grafą ir dekodavimo imties i-vektorius (*L08* lygmens 2 išėjimas), atliekamas dekodavimas, naudojant Kaldi paketo modulį *decode.sh*. Galutinis modelio rezultatas yra neteisingų žodžių kiekio įvertis WER išreikštas procentais ir neteisingų sakinių kiekio įvertis SER išreikštas procentais.

3. Eksperimentai ir rezultatai

Šiame darbe atlikti 4 pagrindiniai eksperimentai ir tyrimai: nustatyta HMM-GMM modelių optimizuotų parametrų įtaka DNN modelio rezultatams, atliktas įvairių DNN architektūrų tyrimas, atliktas geriausios architektūros parametrų optimizavimas ir atliktas kryžminis patikrinimas.

3.1. HMM-GMM parametrų įtakos DNN modeliui tyrimas

Bandymo esmė yra nustatyti, ar parametrai, kurie leidžia gauti geresnius HMM-GMM modelių rezultatus, turės įtakos DNN modeliui. Šie parametrai yra HMM klasifikatorių medžių lapų skaičius (toliau – lapų skaičius), bei Gauso mišinių skaičius (toliau – Gausų skaičius). Šio eksperimento metu atliekami du bandymai – vienas su numatytaisiais Kaldi paketo parametrais, antras su Gyčio Baltrušaičio [27] parinktais optimaliais HMM-GMM modelių parametrais.



31 pav. TDNN-BLSTM supaprastintos architektūros struktūra.

DNN modelyje naudojami numatytieji Kaldi paketo parametrai. Pirmiems bandymams pasirinkta TDNN-BLSTM architektūra, kurios supaprastinta struktūra parodyta 31 pav. Architektūra sudaro 3 TDNN blokai ir 3 LSTM blokai. 3-5 lentelėse pateikiami visų trijų garsyno dalių eksperimento rezultatai.

3 lentelė. L_Sak garsyno dalies atpažinimo tikslumo priklausomybė nuo parametrų.

Metodas	**Parametrų tipas**	**Lapų / Gausų skaičius**	**Klaidos įvertis, %**	
			WER	**SER**
Monofoninis	Numatytieji	1000	48,53	94,68
	Optimizuoti	75000	32,68	86,04
Trifoninis	Numatytieji	2000/11000	29,65	84,53
	Optimizuoti	8000/70000	26,63	79,45
LDA+MLTT	Numatytieji	2000/11000	27,73	81,88
	Optimizuoti	12000/72000	25,50	77,29
SAT	Numatytieji	2000/11000	19,91	74,81
	Optimizuoti	13000/65000	16,17	66,73
TDNN-BLSTM	Numatytieji	-	12,92	59,84
	Optimizuoti	-	12,84	58,96

3 lentelėje pateikti rezultatai parodo, kad neoptimizuotas DNN modelis lenkia visus HMM-GMM modelius. TDNN-BLSTM architektūros WER klaidos įvertis buvo 6,9 % mažesnis už SAT metodo, su numatytaisiais parametrais. Vertinant pagal SER įverti, gauti rezultatai yra proporcingi rezultatams gautiems pagal žodžių atpažinimo klaidos WER įvertį. Optimizuoti parametrai pagerino HMM-GMM modelių rezultatus, tačiau DNN modelio WER klaida sumažėjo tik per 0,08 %.

L_Sak tyrimo dalyje toks įverčio pokytis yra per mažas, kad tolimesniuose tyrimuose, būtu naudojami HMM-GMM modelių parametrai, kurie yra žymiai didesni už numatytuosius parametrus.

4 lentelė. L_Sek garsyno dalies atpažinimo tikslumo priklausomybė nuo parametrų.

Metodas	**Parametrų tipas**	**Lapų / Gausų skaičius**	**Klaidos įvertis, %**	
			WER	**SER**
Monofoninis	Numatytieji	1000	26,53	94,67
	Optimizuoti	6000	18,76	88,26
Trifoninis	Numatytieji	2000/11000	12,56	77,19
	Optimizuoti	3500/19250	12,09	72,81
LDA+MLTT	Numatytieji	2000/11000	11,83	70,67
	Optimizuoti	3000/16500	11,79	70,21
SAT	Numatytieji	2000/11000	10,95	72,15
	Optimizuoti	3500/19250	9,33	67,06
TDNN-BLSTM	Numatytieji	-	5,61	41,24
	Optimizuoti	-	5,52	41,03

4 lentelės rezultatuose matome tą patį dėsningumą – DNN modelis, L_Sek garsyno dalį, iš visų metodų atpažįsta geriausiai. Lyginant su SAT metodu, neteisingų žodžių klaidos įvertį WER, pavyko sumažinti per 5,34 %. Skirtingai nuo L_Sak garsyno dalies, L_Sek dalyje DNN modelis leidžia gerokai sumažinti neteisingų sakinių klaidos įvertį SER per 30,91 %, lyginant su SAT metodu. Optimizuoti parametrai, kaip ir L_Sak dalyje, didelės įtakos neturėjo DNN modeliui ir paklaidą sumažino per 0,09 %.

5 lentelė. L_Zod garsyno dalies atpažinimo tikslumo priklausomybė nuo parametrų.

Metodas	**Parametrų tipas**	**Lapų / Gausų skaičius**	**Klaidos įvertis, %**	
			WER	**SER**
Monofoninis	Numatytieji	1000	15,92	19,45
	Optimizuoti	13000	9,80	12,44
Trifoninis	Numatytieji	2000/11000	14,38	14,46
	Optimizuoti	600/10200	9,62	11,67
LDA+MLTT	Numatytieji	2000/11000	13,20	13,74
	Optimizuoti	750/24000	8,96	10,45
SAT	Numatytieji	2000/11000	4,99	6,47
	Optimizuoti	2000/11000	4,98	6,38
TDNN-BLSTM	Numatytieji	-	7,15	8,26
	Optimizuoti	-	7,14	8,21

Atlikus pirmuosius bandymus su L_Zod garsyno dalimi, pastebima, kad DNN modelis žodžius atpažįsta su didele 7,15 % WER klaida, lyginant su paprastesniu SAT metodu, kurio žodžių atpažinimo klaida yra 4,99 %. SAT metodo numatytieji parametrai leidžia gauti geriausią rezultatą, todėl šio modelio optimizuotų parametrų L_Zod garsyno dalyje nėra. DNN modeliui optimizuoti parametrai, kaip ir kituose garsyno dalyse, realios įtakos neturi.

Bendrai apžvelgiant visų trijų garsyno dalių pirmuosius bandymų rezultatus, galima teigti, kad geresni rezultatai gauti apmokant HMM-GMM modelius neturi įtakos DNN modelio rezultatams. Tolesniems tyrimams, visoms garsyno dalims, naudojami tie parametrai, kurie yra mažesni, siekiant sumažinti nereikalingų skaičiavimų kiekį.

3.2. DNN modelio architektūrų tyrimas

Šiame eksperimente išbandomos skirtingos architektūros, sudarytos iš TDNN, LSTM ir BLSTM blokų kombinacijų. Šiame eksperimente keičiama tik tinklo architektūra, visi kiti modelio parametrai yra konstantos. Visos architektūros išbandomos su kiekviena atskira garsyno dalimi. Išbandyta 18 skirtingų tinklo architektūrų. Kiekvienai architektūrai suteiktas unikalus vardas, sudarytas iš 3 simbolių, pvz., T-B-08 reiškia architektūrą, sudarytą iš TDNN ir BLSTM blokų, kurios eksperimentų vykdymo eilės numeris yra 8. Kiekvienos architektūros TDNN, BLSTM ir LSTM sluoksnių skaičius ir eiliškumas pateikti 6 lentelėje. Taip pat nurodomas atskirų TDNN sluoksnių laiko kontekstas. 1 priede pateiktas pavyzdinis architektūros T-L-14 sudarymo programinis kodas.

6 lentelė. Architektūrų struktūra.

A – architektūra; L – LSTM sluoksnių skaičius; B – BLSTM sluoksnių skaičius; T – TDNN sluoksnių skaičius					
A	**T**	**B**	**L**	**Sluoksnių eiliškumas**	**TDNN laiko kontekstas**
B-B-01	-	3	-	B1-B2-B3	-
B-B-02	-	5	-	B1-B2-B3-B4-B5	-
B-B-03	-	8	-	B1-B2-B3-B4-B5-B6-B7-B8	-
L-L-04	-	-	3	L1-L2-L3	-
L-L-05	-	-	5	L1-L2-L3-L4-L5	-
L-L-06	-	-	8	L1-L2-L3-L4-L5-L6-L7-L8	-
T-B-07	3	3	-	T1-T2-T3-B1-B2-B3	T2(-1,0,1); T3(-1,0,1)
T-B-08	5	3	-	T1-T2-T3-B1-T4-B2-T5-B3	T2(-1,0,1); T3(-3,0,3); T4(-6,0,6); T5(-12,0,12)
T-B-09	7	3	-	T1-T2-T3-B1-T4-T5-B2-T6-T7-B3	T2(-1,0,1,2); T3(-3,0,3); T4|5|6|7(-3,0,3)
T-L-10	3	-	3	T1-T2-T3-L1-L2-L3	T2(-1,0,1); T3(-1,0,1)
T-L-11	5	-	3	T1-T2-T3-L1-T4-L2-T5-L3	T2(-1,0,1); T3(-3,0,3); T4(-6,0,6); T5(-12,0,12)
T-L-12	7	-	3	T1-T2-T3-L1-T4-T5-L2-T6-T7-L3	T2(-1,0,1,2); T3(-3,0,3); T4|5|6|7(-3,0,3)
T-B-13	7	3	-	T1-T2-T3-B1-T4-T5-B2-T6-T7-B3	T2(-1,0,1); T3(-3,0,3); T4|5(-6,0,6); T6|7(-12,0,12)
T-L-14	7	-	3	T1-T2-T3-L1-T4-T5-L2-T6-T7-L3	T2(-1,0,1); T3(-3,0,3); T4|5(-6,0,6); T6|7(-12,0,12)
T-L-15	9	-	3	T1-T2-T3-L1-T4-T5-T6-L2-T7-T8-T9-L3	T2(-1,0,1); T3(-3,0,3); T4|5|6(-6,0,6); T7|8|9(-12,0,12)
T-L-16	9	-	4	T1-T2-T3-L1-T4-T5-L2-T6-T7-L3-T8-T9-L4	T2(-1,0,1); T3(-3,0,3); T4|5(-6,0,6); T6|7|8|9(-12,0,12)
T-L-17	7	-	3	T1-T2-T3-T4-T5-T6-T7-L1-L2-L3	T2(-1,0,1); T3(-1,0,1); T4|5|6|7(-3,0,3)
T-T-18	6	-	-	T1-T2-T3-T4-T5-T6	T2(-1,0,1); T3(-1,0,1,2); T4|5(-3,0,3); T6(-6,-3,0)

1-6 architektūros yra homogeniškos, sudarytos iš 3-8 iš eilės einančių BLSTM arba LSTM sluoksnių. Kaldi pakete, BLSTM blokas yra sudarytas iš dviejų „greitų“ LSTMP blokų, po vieną abiem kryptimis (pirmyn ir atgal), todėl architektūra, pvz., su 3 BLSTM blokais turi 6 LSTM blokus. 18 architektūra taip pat yra homogeniška ir yra sudaryta iš 6 iš eilės einančių TDNN blokų. Visos kitos architektūros yra hibridinės TDNN, BLSTM ir LSTM sluoksnių architektūros.

Visų 18 ištirtų architektūrų rezultatai pateikti 7 lentelėje. Lyginant homogenines architektūras, geriausi rezultatai gaunami naudojant architektūras su BLSTM sluoksniais. L_Sak dalyje, geriausias rezultatas (WER = 12,61 %, SER = 60,04 %) gautas su B-B-1 architektūra, nes papildomi 2 BLSTM sluoksniai architektūroje B-B-02, WER rezultatą pagerino tik 0,04 %. Toks pat dėsningumas pastebimas ir L_Sek garsyno dalyje. Garsyno dalyje L_Zod, geriausias rezultatas gautas su B-B-03 architektūra (WER = 7,08 %, SER = 8,54 %). Pastebima, kad sudėtingesnės homogeninės architektūros, šneką atpažįsta prasčiau, už paprastesnes architektūras, sudarytas iš mažiau sluoksnių.

7 lentelė. HMM-DNN modelio tinklo architektūrų rezultatai.

Architektūra	**WER, %**			**SER, %**		
	L_Sak	**L_Sek**	**L_Zod**	**L_Sak**	**L_Sek**	**L_Zod**
B-B-01	12,61	5,70	8,63	60,04	43,54	9,88
B-B-02	12,57	5,62	7,38	59,74	44,51	8,87
B-B-03	13,14	5,42	7,08	60,43	44,87	8,54
L-L-04	14,76	6,01	10,24	66,48	47,35	11,40
L-L-05	14,37	6,09	8,20	65,40	48,67	9,91
L-L-06	14,90	6,58	9,71	66,98	51,50	11,36
T-B-07	12,92	5,61	7,15	59,84	41,24	8,26
T-B-08	12,17	5,14	7,47	56,87	38,67	8,74
T-B-09	12,93	5,18	7,21	59,20	41,68	8,32
T-L-10	14,11	5,65	12,80	62,54	43,19	13,84
T-L-11	12,54	5,21	8,88	58,60	40,35	10,22
T-L-12	12,65	5,33	8,29	59,37	41,24	9,77
T-B-13	11,69	4,91	6,98	55,73	38,14	8,19
T-L-14	***11,08***	***4,99***	***6,85***	***54,65***	***37,52***	***8,08***
T-L-15	11,19	4,85	7,27	54,79	36,28	8,35
T-L-16	11,09	4,80	6,96	54,65	36,64	8,15
T-L-17	15,22	5,58	8,66	64,63	43,36	10,21
T-T-18	13,28	5,87	14,25	60,45	43,10	13,82

Lyginant hibridines ir homogenines sistemas, geriausi rezultatai gaunami naudojant hibridines TDNN ir LSTM architektūras. TDNN sluoksnių platesnis laiko kontekstas leidžia pagerinti visų garsyno dalių rezultatus. Tolimesniuose eksperimentuose, visose garsyno dalyse, pasirinkta naudoti T-L-14 architektūrą. L_Sak ir L_Zod dalyse šios architektūros klaidos įverčiai yra geriausi iš visų 18 bandymų. L_Sek tyrimo dalyje, architektūrų T-B-13, T-L-15 ir T-L-16 rezultatai minimaliai yra geresni, tačiau šios architektūros yra sudėtingesnės ir minimalus klaidos sumažinimas neatperka atsirandančių papildomų skaičiavimų.

Lyginant blogiausios ir geriausios architektūros rezultatus, L_Sak dalyje WER įvertis pagerėjo per 4,14 %, SER įvertis pagerėjo per 9,98 %. L_Sek dalyje WER įvertis pagerėjo per 1,59 %, SER įvertis pagerėjo per 13,98 %. L_Zod dalyje WER įvertis pagerėjo per 7,4 %, SER įvertis pagerėjo per 5,76 %.

3.3. DNN modelio parametrų optimizavimas

Šiame poskyryje pateikiami kiekvienos garsyno dalies, DNN modelio parametrų optimizavimo rezultatai. Parametrai keičiami tokia eilės tvarka:

1. epochų skaičius (2.4.3 sk., parametras *num-epochs*);
2. mokymo greitis (2.4.3 sk., parametrai *initial-effective-lrate* ir *final-effective-lrate*);
3. L_2 sureguliavimas (2.4.3 sk., parametras *l2-regularize*);
4. inertiškumas (2.4.3 sk., parametras *momentum*);
5. išmetimo „tvarkaraštis“ (2.4.3 sk., parametras *dropout-schedule*).

Kiekviename žingsnyje rastas optimalus parametras, toliau naudojamas kaip konstanta, ieškant sekančio parametro optimalios vertės.

Atskirų garsyno dalių atpažinimo rezultatų priklausomybė nuo mokymo epochų skaičiaus, pateikta 8 lentelėje. Visoms dalims, geriausias rezultatas gautas atliekant mokymą 4 kartus. Tai yra numatytoji Kaldi paketo parametro vertė.

8 lentelė. Atskirų garsyno dalių tikslumo priklausomybė nuo epochų skaičiaus.

Garsyno dalis	**Klaidos įvertis**	**Epochų skaičius**			
		3	**4**	**5**	**6**
L_Sak	**WER, %**	11,60	***11,08***	11,27	11,40
	SER, %	55,84	***53,01***	54,23	54,98
L_Sek	**WER, %**	4,98	***4,89***	5,02	5,10
	SER, %	37,08	***36,19***	37,43	38,32
L_Zod	**WER, %**	7,45	***6,85***	8,56	8,94
	SER, %	8,24	***8,08***	9,32	9,71

Atskirų garsyno dalių atpažinimo rezultatų priklausomybė nuo pradinio ir galutinio mokymosi greičio, pateikta 9 lentelėje. Visoms dalims, geriausias rezultatas gautas kuomet pradinis ir galutinis mokymosi greitis yra atitinkamai 0,05 ir 0,15, tačiau toliau pasirinkta naudoti mažesnį mokymosi greitį (0,1 ir 0,01), nes parametro padidinimas WER įvertį pagerino tik per 0,01 - 0,02 %, SER įvertį – per 0,05 - 0,15 %.

9 lentelė. Tikslumo priklausomybė nuo pradinio ir galutinio mokymosi greičio.

Mok. gr.: prad. / gal.	**WER, %**					**SER, %**				
	0,00001	**0,0001**	**0,001**	**0,01**	**0,05**	**0,00001**	**0,0001**	**0,001**	**0,01**	**0,05**
L_Sak										
0,0001	13,50	-	-	-	-	62,34	-	-	-	-
0,001	11,96	11,08	-	-	-	57,93	54,65	-	-	-
0,01	10,38	10,06	9,57	-	-	54,18	53,48	50,58	-	-
0,1	9,78	9,68	9,55	***8,61***	-	52,37	51,59	50,51	***48,13***	-
0,15	-	-	-	-	8,59	-	-	-	-	47,99

Mok. gr.: prad. / gal.	WER, %					SER, %				
	0,00001	**0,0001**	**0,001**	**0,01**	**0,05**	**0,00001**	**0,0001**	**0,001**	**0,01**	**0,05**
L_Sek										
0,0001	5,64	-	-	-	-	42,04	-	-	-	-
0,001	5,08	4,89	-	-	-	39,91	36,19	-	-	-
0,01	4,69	4,71	4,65	-	-	37,70	37,35	35,01	-	-
0,1	4,82	4,59	4,42	***4,23***	-	37,43	36,11	34,78	***34,6***	-
0,15	-	-	-	-	4,22	-	-	-	-	34,45
L_Zod										
0,0001	12,39	-	-	-	-	13,16	-	-	-	-
0,001	8,34	6,85	-	-	-	9,49	8,08	-	-	-
0,01	5,30	6,83	6,26	-	-	6,42	7,91	7,42	-	-
0,1	4,73	3,81	3,69	***3,41***	-	6,00	5,16	4,98	***4,46***	-
0,15	-	-	-	-	3,40	-	-	-	-	4,41

Atskirų garsyno dalių atpažinimo rezultatų priklausomybė nuo L_2 sureguliavimo, pateikta 10 lentelėje. Visoms garsyno dalims, geriausias rezultatas gautas su mažu sureguliavimu ($5 \cdot 10^{-7}$).

10 lentelė. Atskirų garsyno dalių tikslumo priklausomybė nuo L_2 sureguliavimo.

Garsyno dalis	Klaidos įvertis	L_2 sureguliavimas					
		$5 \cdot 10^{-4}$	$5 \cdot 10^{-5}$	$5 \cdot 10^{-6}$	$1 \cdot 10^{-6}$	$5 \cdot 10^{-7}$	$1 \cdot 10^{-7}$
L_Sak	**WER, %**	8,61	8,06	7,66	7,38	***7,30***	7,52
	SER, %	48,13	45,41	43,86	42,24	***41,47***	42,52
L_Sek	**WER, %**	4,41	4,23	4,24	3,77	***3,61***	3,94
	SER, %	34,34	34,60	35,66	31,77	***31,15***	31,95
L_Zod	**WER, %**	5,39	3,41	4,68	4,63	***3,35***	3,87
	SER, %	6,50	4,46	5,75	5,51	***4,61***	4,68

Atskirų garsyno dalių atpažinimo rezultatų priklausomybė nuo inertiškumo pateikta, 11 lentelėje. Visoms garsyno dalims, geriausias rezultatas gautas su 0,5 inertiškumu. Tiek didinant, tiek mažinant inertiškumą, atpažinimo klaidų įverčiai didėja. Su dideliu 0,9 inertiškumu gauta 100 % klaida.

11 lentelė. Atskirų garsyno dalių tikslumo priklausomybė nuo inertiškumo.

Garsyno dalis	Klaidos įvertis	Inertiškumas						
		0,00	**0,40**	**0,45**	**0,50**	**0,55**	**0,60**	**0,90**
L_Sak	**WER, %**	7,30	7,32	7,18	***7,14***	7,34	7,59	100
	SER, %	41,47	42,52	42,37	***42,55***	43,13	44,23	100
L_Sek	**WER, %**	3,61	3,91	4,24	***3,14***	5,89	5,65	100
	SER, %	31,15	35,75	39,03	***32,04***	47,08	46,19	100
L_Zod	**WER, %**	3,35	4,61	4,04	***3,28***	5,34	20,75	100
	SER, %	4,61	5,45	5,19	***4,15***	6,33	20,21	100

Atskirų garsyno dalių, atpažinimo rezultatų priklausomybė nuo „išmetimo“ tvarkaraščio, pateikta 12 lentelėje. Geriausias rezultatas gautas su tikimybe $p = 0,1$. Tai yra numatytoji parametro vertė.

12 lentelė. Atskirų garsyno dalių tikslumo nuo „išmetimo“ tvarkaraščio tikimybės *p*.

Garsyno dalis	**Klaidos įvertis**	**„Išmetimo“ tvarkaraščio '0,0@0.20,*p*@0.50,0' tikimybė *p***				
		0,0	**0,1**	**0,2**	**0,3**	**0,4**
L_Sak	**WER, %**	8,82	***7,14***	9,55	13,38	10,21
	SER, %	48,56	***42,55***	52,01	62,14	54,07
L_Sek	**WER, %**	5,53	***3,14***	6,00	8,41	6,42
	SER, %	43,10	***32,04***	49,29	60,00	50,44
L_Zod	**WER, %**	3,63	***3,28***	3,84	3,56	5,78
	SER, %	4,79	***4,15***	4,75	4,74	6,74

Atlikus pasirinktų parametrų optimizavimą, matome, kad nepriklausomai nuo tiriamos garsyno dalies, geriausi rezultatai gaunami naudojant tuos pačius parametrus, visoms garsyno dalims. Naudojant optimalius parametrus, L_Sak dalyje neteisingų žodžių įvertis WER sumažėjo per 3,94 %, nuo 11,08 % iki 7,14 %. Neteisingų sakinių įvertis SER sumažėjo per 10,46 %, nuo 53,01 % iki 42,55 %. L_Sek dalyje neteisingų žodžių įvertis WER sumažėjo per 1,75 %, nuo 4,89 % iki 3,14 %. Neteisingų sakinių įvertis SER sumažėjo per 4,15 %, nuo 36,19 % iki 32,04 %. L_Zod dalyje neteisingų žodžių įvertis WER sumažėjo per 3,57 %, nuo 6,85 % iki 3,28 %. Neteisingų sakinių įvertis SER sumažėjo per 3,93 %, nuo 8,08 % iki 4,15 %.

3.4. Kryžminė patikra

Siekiant patikrinti sukurtos ASR sistemos efektyvumą, atliekamas 5-kartus kryžminis patikrinimas. Garsynas yra padalijamas į 5 panašaus dydžio dalis, iš kurių viena yra dekodavimo imtis, o likusios keturios sudaro mokymo imtį. Eksperimentas atliekamas 5 kartus, kiekvieną syki naudojant kitą dalį kaip dekodavimo imtį. Visi ankščiau atlikti eksperimentai buvo vykdomi naudojant 1 dalį, kaip dekodavimo imtį. Kryžminio patikrinimo rezultatai pateikti 13 lentelėje.

13 lentelė. Kryžminio patikrinimo rezultatai.

	WER, %			**SER, %**		
	L_Sak	**L_Sek**	**L_Zod**	**L_Sak**	**L_Sek**	**L_Zod**
1 dalis	7,14	3,14	3,28	42,55	32,04	4,15
2 dalis	8,07	7,35	2,72	48,41	48,39	3,81
3 dalis	7,77	6,11	2,57	45,68	42,62	3,35
4 dalis	6,33	6,79	3,64	41,99	43,26	5,06
5 dalis	7,06	6,81	3,40	43,22	44,51	4,83
Vidurkis	***7,27***	***6,04***	***3,12***	***44,37***	***42,16***	***4,24***

Atlikus kryžminį patikrinimą, matome, kad L_Sak dalies atpažinimo tikslumas svyruoja nuo 6,33 % iki 8,07 %, L_Sek dalies atpažinimo tikslumas svyruoja nuo 3,14 % iki 7,35 % ir L_Zod dalies atpažinimo tikslumas svyruoja nuo 2,57 % iki 3,64 %.

Išvados

1. Eksperimentais nustatyta, kad naudojant sukurtą DNN modelį, lyginant su klasikiniais HMM-GMM modeliais, galima sumažinti šnekos atpažinimo klaidos įverčius. Sakinių tyrimo dalyje, HMM-GMM modelių mažiausi klaidingų žodžių ir sakinių įverčiai atitinkamai yra 16,17 % ir 66,73 %, tuo tarpu DNN modelio mažiausi įverčiai yra 7,14 % ir 41,99 %. Sekų tyrimo dalyje, HMM-GMM modelių mažiausi klaidingų žodžių ir sakinių įverčiai atitinkamai yra 9,33 % ir 67,06 %, tuo tarpu DNN modelio mažiausi įverčiai yra 3,14 % ir 32,04 %. Žodžių tyrimo dalyje, HMM-GMM modelių mažiausi klaidingų žodžių ir sakinių įverčiai atitinkamai yra 4,98 % ir 6,38 %, tuo tarpu DNN modelio mažiausi įverčiai yra 3,28 % ir 4,26 %.
2. Eksperimentais nustatyta, kad parinkti optimalūs HMM-GMM modelių parametrai, visų garsyno dalių tyrimuose, neturi įtakos DNN modelio šnekos atpažinimo tikslumui.
3. Atlikus bandymus su 18 skirtingų DNN modelio architektūrų, nustatyta, kad naudojant tuos pačius požymių rinkinius ir parametrus, tik keičiant neuroninio tinklo architektūros sluoksnių tipą ir išdėstymą, galima sumažinti šnekos atpažinimo klaidos įverčius. Sakinių tyrimo dalyje, priklausomai nuo architektūros, neteisingų žodžių ir sakinių klaidos įverčiai atitinkamai svyruoja nuo 11,08 % iki 15,22 % ir nuo 54,65 % iki 66,98 %. Sekų tyrimo dalyje, priklausomai nuo architektūros, neteisingų žodžių ir sakinių klaidos įverčiai atitinkamai svyruoja nuo 4,80 % iki 6,58 % ir nuo 36,28 % iki 51,50 %. Žodžių tyrimo dalyje, priklausomai nuo architektūros, neteisingų žodžių ir sakinių klaidos įverčiai atitinkamai svyruoja nuo 6,85 % iki 14,25 % ir nuo 8,08 % iki 13,84 %.
4. Eksperimentais nustatyta, kad keičiant neuroninio tinklo mokymo parametrus ir nekeičiant architektūros, galima sumažinti šnekos atpažinimo klaidos įverčius. Sakinių tyrimo dalyje, naudojant numatytuosius parametrus, klaidingų žodžių ir sakinių įverčiai atitinkamai yra 11,08 % ir 54,65 %, tuo tarpu naudojant optimizuotus parametrus, modelio įverčiai yra 7,14 % ir 42,55 %. Sekų tyrimo dalyje, naudojant numatytuosius parametrus, klaidingų žodžių ir sakinių įverčiai atitinkamai yra 4,99 % ir 37,52 %, tuo tarpu naudojant optimizuotus parametrus, modelio įverčiai yra 3,14 % ir 32,04 %. Žodžių tyrimo dalyje, naudojant numatytuosius parametrus, klaidingų žodžių ir sakinių įverčiai atitinkamai yra 6,85 % ir 8,08 %, tuo tarpu naudojant optimizuotus parametrus, modelio įverčiai yra 3,28 % ir 4,15 %.
5. Atlikus kryžminį patikrinimą, gauta, kad sukurtas modelis nėra per daug prisitaikęs prie mokymo imties. Sakinių tyrimo dalyje, neteisingų žodžių ir sakinių klaidos įverčiai, atitinkamai svyruoja nuo 6,33 % iki 8,07 % (vidurkis 7,27 %) ir nuo 41,99 % iki 48,41 % (vidurkis 44,37 %). Sekų tyrimo dalyje, neteisingų žodžių ir sakinių klaidos įverčiai, atitinkamai svyruoja nuo 3,14 % iki 7,35 % (vidurkis 6,04 %) ir nuo 32,04 % iki 48,39 % (vidurkis 42,16 %). Žodžių tyrimo dalyje, neteisingų žodžių ir sakinių klaidos įverčiai, atitinkamai svyruoja nuo 2,57 % iki 3,64 % (vidurkis 3,12 %) ir nuo 3,35 % iki 5,06 % (vidurkis 4,24 %).
6. Gautus rezultatus galima palyginti su kitų Lietuvos autorių darbais, atliktais su Liepa garsynu:
    - sekų ir žodžių tyrimo dalių, kryžminio patikrinimo atpažinimo rezultatai yra žymiai geresni nei Dariaus Kairio [25] magistro darbe, naudojant TensorFlow programinį paketą, gauti rezultatai (žodžių tyrimo dalyje WER = 9,32 %, sekų tyrimo dalyje – WER = 28,19 %);
    - visų garsyno dalių tyrimų, neteisingų žodžių ir sakinių geriausi įverčiai gali būti palyginti su Gyčio Baltrušaičio [27] magistro darbe, naudojant Kaldi paketą, gautais rezultatais: sakinių tyrimo dalyje WER = 10,24 % ir SER = 53,62 %, sekų tyrimo dalyje WER = 7,52 % ir SER = 68,22 %, žodžių tyrimo dalyje WER = 2,78 % ir SER = 3,78 %.

Literatūros sąrašas

1. YU, Dong ir Li DENG. Automatic Speech Recognition – A Deep Learning Approach: Signals and Communication Technology [interaktyvus]. Springer, 2014 [žiūrėta 2022-05-19]. ISBN 9781447157793. Prieiga per: http://books.google.com/books?vid=ISBN9781447157793
2. HE, Xiaodong ir Li DENG. Speech-centric information processing: An optimization-oriented approach. Proceedings of the IEEE [interaktyvus]. 2013, t. 101, nr. 5, p. 1116-1135 [žiūrėta 2022-05-19]. Prieiga per: doi: 10.1109/JPROC.2012.2236631
3. Ethnologue: Languages of the World [interaktyvus]. 2022 [žiūrėta 2022-05-19]. Prieiga per: https://www.ethnologue.com
4. NARANG, Shreya ir Divya GUPTA. Speech Feature Extraction Techniques: A Review. International Journal of Computer Science and Mobile Computing [interaktyvus]. 2015, t. 4, nr. 3, p. 107-114 [žiūrėta 2022-05-19]. Prieiga per: https://www.ijcsmc.com/docs/papers/March 2015/V4I3201545.pdf
5. CUTAJAR, Michelle, Edward GATT, Ivan GRECH, Owen CASHA ir Joseph MICALLEF. Comparative study of automatic speech recognition techniques. IET Signal Processing 7 [interaktyvus]. 2013, nr. 1, p. 25-46 [žiūrėta 2022-05-19]. Prieiga per: doi: 10.1049/iet-spr .2012.0151
6. RAO, K. Sreenivasa ir Manjunath K. E. Speech Recognition Using Articulatory and Excitation Source Features: SpringerBriefs in Speech Technology [interaktyvus]. Springer, 2017 [žiūrėta 2022-05-19]. ISBN 9783319492209. Prieiga per: http://books.google.com/books?vid=ISBN 9783319492209
7. BHATT, Shobha, Anurag JAIN ir Amita DEV. Acoustic modeling in speech recognition: a systematic review. International Journal of Advanced Computer Science and Applications. 2020, t. 11, nr. 4 [žiūrėta 2022-05-19]. Prieiga per: doi: 10.14569/IJACSA.2020.0110455
8. FINE, Shai, Yoram SINGER ir Naftali TISHBY. The hierarchical hidden Markov model: Analysis and applications. Machine learning [interaktyvus]. 1998, t. 32, nr. 1, p. 41-62 [žiūrėta 2022-05-19]. Prieiga per: doi: 10.1023/A:1007469218079
9. REYNOLDS, Douglas A. Gaussian mixture models. Encyclopedia of biometrics [interaktyvus]. 2009, t. 741, p. 659-663 [žiūrėta 2022-05-19]. Prieiga per: doi: 10.1007/978-1-4899-7488-4_196
10. PAPASTRATIS, Ilias. Speech Recognition: a review of the different deep learning approaches. https://theaisummer.com/ [interaktyvus]. 2021 [žiūrėta 2022-05-19]. Prieiga per: https://theai summer.com/speech-recognition/
11. WAIBEL, Alex, Toshiyuki HANAZAWA, Geoffrey HINTON, Kiyohiro SHIKANO ir Kevin J. LANG. Phoneme recognition using time-delay neural networks. IEEE transactions on acoustics, speech, and signal processing [interaktyvus]. 1989, t. 37, nr. 3, p. 328-339 [žiūrėta 2022-05-19]. Prieiga per: doi: 10.1109/29.21701
12. WAIBEL, Alex. Modular construction of time-delay neural networks for speech recognition. Neural computation [interaktyvus]. 1989, t. 1, nr. 1, p. 39-46 [žiūrėta 2022-05-19]. Prieiga per: doi: 10.1162/neco.1989.1.1.39
13. PEDDINTI, Vijayaditya, Daniel POVEY ir Sanjeev KHUNDANPUR. A time delay neural network architecture for efficient modeling of long temporal contexts. Sixteenth annual conference of the international speech communication association [interaktyvus]. 2015 [žiūrėta 2022-05-19]. Prieiga per: doi: 10.21437/Interspeech.2015-647

14. DUA, Mohit, Rohit YADAV, Divya MAMGAI ir Sonali BRODIYA. An improved RNN-LSTM based novel approach for sheet music generation. Procedia Computer Science [interaktyvus]. 2020, t. 171, p. 465-474 [žiūrėta 2022-05-19]. Prieiga per: doi: 10.1016/j.procs.2020.04.049
15. HIBAT-ALLAH, Mohamed, Martin GANAHL, Lauren E. HAYWARD, Roger G. MELKO, ir Juan CARRASQUILLA. Recurrent neural network wave functions. Physical Review Research [interaktyvus]. 2020, t. 2, nr. 2 [žiūrėta 2022-05-19]. Prieiga per: doi: 10.48550/arXiv.2002.02973
16. RUMELHART, David E., Geoffrey E. HINTON ir Ronald J. WILLIAMS. Learning representations by back-propagating errors. Nature [interaktyvus]. 1986, t. 323, nr. 9, p. 533-536 [žiūrėta 2022-05-19]. Prieiga per: doi: 10.1038/323533a0
17. JORGES, Christoph, Cordula BERKENBRINK, Britta STUMPE. Prediction and reconstruction of ocean wave heights based on bathymetric data using LSTM neural networks. Ocean Engineering [interaktyvus]. 2021, t. 232, [žiūrėta 2022-05-19]. ISSN 0029-8018. Prieiga per: doi: 10.1016/j.oceaneng.2021.109046
18. POVEY, Daniel ir kt. The Kaldi speech recognition toolkit. In IEEE 2011 workshop on automatic speech recognition and understanding [interaktyvus]. 2011 [žiūrėta 2022-05-19]. Prieiga per: https://www.danielpovey.com/files/2011_asru_kaldi.pdf
19. WANG, Dong, Xiaodong WANG ir Shaohe LV. An overview of end-to-end automatic speech recognition. Symmetry [interaktyvus]. 2019, t. 11, nr. 8:1018 [žiūrėta 2022-05-19]. Prieiga per: doi: 10.3390/sym11081018
20. GEORGESCU, Alexandru-Lucian, Horia CUCU ir Corneliu BURILEANU. Kaldi-based DNN architectures for speech recognition in Romanian. 2019 International Conference on Speech Technology and Human-Computer Dialogue (SpeD) [interaktyvus]. 2019, p. 1-6 [žiūrėta 2022-05-19]. Prieiga per: doi: 10.1109/SPED.2019.8906555
21. LAURINČIUKAITĖ, Sigita, Laimutis TELKSNYS, Pijus KASPARAITIS, Regina KLIUKIENĖ ir Vilma PAUKŠTYTĖ. Lithuanian Speech Corpus Liepa for Development of Human-Computer Interfaces Working in Voice Recognition and Synthesis Mode Informatica [interaktyvus]. 2018, t. 29, p. 487-498 [žiūrėta 2022-05-19]. Prieiga per: doi: 10.15388/Informatica.2018.177
22. RAŠKINIS, Gailius, Gintarė PAŠKAUSKAITĖ, Aušra SAUDARGIENĖ, Airenas VAIČIŪNAS ir Asta KAZLAUSKIENĖ. Comparison of Phonemic and Graphemic Word to Sub-Word Unit Mappings for Lithuanian Phone-Level Speech Transcription. Informatica [interaktyvus]. 2019, t. 30, nr. 3, p. 573-593 [žiūrėta 2022-05-19]. Prieiga per: doi: 10.15388/Informatica.2019.219
23. DOVYDAITIS, Laurynas. Lietuviškai kalbančio diktoriaus identifikavimo naudojant rekurentinius neuroninius tinklus tikslumo tyrimas: daktaro disertacija [interaktyvus]. Vilniaus universitetas, 2018 [žiūrėta 2022-05-19]. Prieiga per: https://www.mii.lt/files/doc/lt/doktorantura/apgintos_disertacijos/kf_dis_2018_dovydaitis.pdf
24. ŽEKIENĖ, Gintarė. Hybrid recognition technology for Lithuanian voice commands: daktaro disertacija [interaktyvus]. Kauno technologijos universitetas, 2021 [žiūrėta 2022-05-19]. Prieiga per: https://vb.ktu.edu/permalink/f/1slhar4/ELABAETD92291818
25. KAIRYS, Darius. Garsyno LIEPA žodžių ir frazių sekų atpažinimo su TensorFlow paketu sistemos sukūrimas ir tyrimas: magistro darbas [interaktyvus]. Kauno technologijos universitetas, 2021 [žiūrėta 2022-05-19]. Prieiga per: https://vb.ktu.edu/permalink/f/1slhar4/ELABAETD95813554

26. SINKEVIČIUS, Giedrius. Skaičių garsyno atpažinimo su Kaldi paketu sistemos sukūrimas ir tyrimas: magistro darbas [interaktyvus]. Kauno technologijos universitetas, 2019 [žiūrėta 2022-05-19]. Prieiga per: https://vb.ktu.edu/permalink/f/1slhar4/ELABAETD37738633
27. BALTRUŠAITIS, Gytis. Garsyno LIEPA atpažinimo su Kaldi paketu sistemos sukūrimas ir tyrimas: magistro darbas [interaktyvus]. Kauno technologijos universitetas, 2020 [žiūrėta 2022-05-19]. Prieiga per: https://epubl.ktu.edu/object/elaba:61700662/
28. POVEY, Daniel ir kt. The subspace Gaussian mixture model – A structured model for speech recognition. Computer Speech & Language [interaktyvus]. 2011, t. 25, nr. 2, p. 404-439 [žiūrėta 2022-05-19]. Prieiga per: doi: 10.1016/j.csl.2010.06.003
29. NORKUS, Gediminas. Vardų garsyno atpažinimo su Kaldi ir TensorFlow paketais sistemos sukūrimas ir tyrimas: magistro darbas [interaktyvus]. Kauno technologijos universitetas, 2020 [žiūrėta 2022-05-19]. Prieiga per: https://vb.ktu.edu/permalink/f/1slhar4/ELABAETD61911745
30. CHODROFF, Eleanor. Corpus phonetics tutorial [interaktyvus]. 2015 [žiūrėta 2022-05-19]. Prieiga per: doi: 10.48550/arXiv.1811.05553
31. STRAND, Ole Morten ir Andreas EGEBERG. Cepstral mean and variance normalization in the model domain. COST278 and ISCA Tutorial and Research Workshop (ITRW) on Robustness Issues in Conversational Interaction [interaktyvus]. 2004 [žiūrėta 2022-05-19]. Prieiga per: https://www.isca-speech.org/archive_open/archive_papers/robust2004/rob4_38.pdf
32. STOLCKE, Andreas. SRILM-an extensible language modeling toolkit. Seventh international conference on spoken language processing [interaktyvus]. 2002 [žiūrėta 2022-05-19]. Prieiga per: https://www.isca-speech.org/archive_v0/archive_papers/icslp_2002/i02_0901.pdf
33. KO, Tom, Vijayaditya PEDDINTI, Daniel POVEY ir Sanjeev KHUDANPUR. Audio augmentation for speech recognition. Sixteenth annual conference of the international speech communication association [interaktyvus]. 2015 [žiūrėta 2022-05-19]. Prieiga per: doi: 10.21437/Interspeech.2015-711
34. DEHAK, Najim, Patrick J. KENNY, Réda DEHAK, Pierre DUMOUCHEL ir Pierre OUELLET. Front-End Factor Analysis for Speaker Verification. IEEE Transactions on Audio, Speech, and Language Processing [interaktyvus]. 2011, t.19, nr. 4, p. 788-798 [žiūrėta 2022-05-19]. Prieiga per: doi: 10.1109/TASL.2010.2064307
35. KHOSRAVANI, Abbas, Cornelius GLACKIN, Nazim DUGAN, Gérard CHOLLET, ir Nigel CANNINGS. The Intelligent Voice 2016 Speaker Recognition System [interaktyvus]. 2016 [žiūrėta 2022-05-19]. Prieiga per: doi: 10.48550/arXiv.1611.00514
36. REYNOLDS, Douglas A., Thomas F. QUATIERI ir Robert B. DUNN. Speaker verification using adapted Gaussian mixture models. Digital signal processing [interaktyvus]. 2000, t. 10, nr. 1-3, p. 19-41 [žiūrėta 2022-05-19]. Prieiga per: doi: 10.1006/dspr.1999.0361
37. LJOLIEL, Andrej, Fernando PEREIRA ir Michael RILEY. Efficient general lattice generation and rescoring. Sixth European Conference on Speech Communication and Technology [interaktyvus]. 1999 [žiūrėta 2022-05-19]. Prieiga per: https://www.researchgate.net/publication/221481222_Efficient_general_lattice_generation_and_rescoring
38. SAK, Hasim, Murat SARACLAR ir Tunga GUNGOR. On-the-fly lattice rescoring for real-time automatic speech recognition. INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association [interaktyvus]. 2010, p. 2450-2453 [žiūrėta 2022-05-19]. Prieiga per: doi: 10.21437/Interspeech.2010-532

39. JIN, Xiaojie, Chunyan XU, Jiashi FENG, Yunchao WEI, Junjun XIONG ir Shuicheng YAN. Deep Learning with S-Shaped Rectified Linear Activation Units. Proceedings of the AAAI Conference on Artificial Intelligence, [interaktyvus]. 2016, t. 30, nr. 1 [žiūrėta 2022-05-19]. Prieiga per: doi: 10.48550/arXiv.1512.07030
40. IOFFE, Sergey ir Christian SZEGEDY. Batch normalization: Accelerating deep network training by reducing internal covariate shift. International conference on machine learning [interaktyvus]. 2015, p. 448-456 [žiūrėta 2022-05-19]. Prieiga per: doi: 10.48550/arXiv.1502.03167
41. Sak, Hasim, Andrew SENIOR ir Françoise BEAUFAYS. Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition [interaktyvus]. 2014 [žiūrėta 2022-05-19]. Prieiga per: doi: 10.48550/arXiv.1402.1128
42. Srivastava, NITISH, Geoffrey HINTON, Alex KRIZHEVSKY, Ilya SUTSKEVER ir Ruslan SALAKHUTDINOV. Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research [interaktyvus]. 2014, t. 15, nr. 1, p. 1929-1958 [žiūrėta 2022-05-19]. Prieiga per: https://www.cs.toronto.edu/~rsalakhu/papers/srivastava14a.pdf
43. CHENG, Gaofeng, Vijayaditya PEDDINTI, Daniel POVEY, Vimal MANOHAR, Sanjeev KHUNDANPUR ir Yonghong YAN. An Exploration of Dropout with LSTMs. Interspeech [interaktyvus]. 2017, p. 1586-1590 [žiūrėta 2022-05-19]. Prieiga per: doi: 10.21437/Interspeech .2017-129
44. POVEY, Daniel, Vijayaditya PEDDINTI, Daniel GALVEZ, Pegah GHAHREMANI, Vimal MANOHAR, Xingyu NA, Yiming WANG ir Sanjeev KHUNDAPUR. Purely sequence-trained neural networks for ASR based on lattice-free MMI. Interspeech [interaktyvus]. 2016, p. 2751-2755 [žiūrėta 2022-05-19]. Prieiga per: doi: 10.21437/Interspeech.2016-595
45. KHAN, Najeeb, Jawad SHAH ir Ian STAVNESS. Bridgeout: stochastic bridge regularization for deep neural networks. IEEE Access [interaktyvus]. 2018, t. 6, p. 42961-42970 [žiūrėta 2022-05-19]. Prieiga per: doi: 10.48550/arXiv.1804.08042
46. GOODFELLOW, Ian, Yoshua BENGIO ir Aaron COURVILLE. Deep Learning: Adaptive Computation and Machine Learning series [interaktyvus]. MIT Press, 2016 [žiūrėta 2022-05-19]. ISBN 9780262035613. Prieiga per: http://books.google.com/books?vid=ISBN9780262035613

Priedai

1 priedas. Neuroninio tinklo architektūros sudarymo programinis kodas

```
# architektūros T-L-14 sudarymo lygmuo L12
# patikrinama ar nereikia programos stabdyti po L11 lygmens
if [ $stage -le 12 ]; then

  # suskaičiuojamos išėjimo dimensijos pagal L11 lygmenyje sudarytą sprendimų medį
  num_targets=$(tree-info $treedir/tree |grep num-pdfs|awk '{print $2}')

  # nurodomi LSTM tinklo papildomi pasirinktini nustatymai
  lstm_opts="decay-time=40"

  # sukuriamas konfigūracijos aplankas
  mkdir -p $dir/configs

  # nurodoma nuo kurios vietos reikia nukopijuoti konfigūracija į atskirą failą
  cat <<EOF > $dir/configs/network.xconfig

  # nurodomos įėjimo požymių dimensijos
  input dim=100 name=ivector
  input dim=40 name=input

  # su LDA transformacija sudaromas 300 dimensijų įėjimo vektorius
  fixed-affine-layer name=lda input=Append(-2,-1,0,1,2,ReplaceIndex(ivector, t, 0)) affine-transform-file=$dir/configs/lda.mat

  # konfigūruojami TDNN-1,2,3 sluoksniai, nurodant dimensijas ir laiko kontekstą
  relu-renorm-layer name=tdnn1 dim=1024
  relu-renorm-layer name=tdnn2 input=Append(-1,0,1) dim=1024
  relu-renorm-layer name=tdnn3 input=Append(-3,0,3) dim=1024

  # konfigūruojamas LSTM-1 sluoksnis, nurodant jo dimensijas ir parametrus
  fast-lstmp-layer name=lstm1 cell-dim=1024 recurrent-projection-dim=256 non-recurrent-projection-dim=256 delay=-3 $lstm_opts

  # konfigūruojami TDNN-4,5 sluoksniai, nurodant dimensijas ir laiko kontekstą
  relu-renorm-layer name=tdnn4 input=Append(-6,0,6) dim=1024
  relu-renorm-layer name=tdnn5 input=Append(-6,0,6) dim=1024

  # konfigūruojamas LSTM-2 sluoksnis, nurodant jo dimensijas ir parametrus
  fast-lstmp-layer name=lstm2 cell-dim=1024 recurrent-projection-dim=256 non-recurrent-projection-dim=256 delay=-3 $lstm_opts

  # konfigūruojami TDNN-6,7 sluoksniai, nurodant dimensijas ir laiko kontekstą
  relu-renorm-layer name=tdnn6 input=Append(-12,0,12) dim=1024
  relu-renorm-layer name=tdnn7 input=Append(-12,0,12) dim=1024

  # konfigūruojamas LSTM-3 sluoksnis, nurodant jo dimensijas ir parametrus
  fast-lstmp-layer name=lstm3 cell-dim=1024 recurrent-projection-dim=256 non-recurrent-projection-dim=256 delay=-3 $lstm_opts

  # nurodomas tarpinio išėjimo outl laiko kontekstas
  linear-component name=outl input=Append(-5,0,5)

  # sudaromas išėjimo sluoksnis, nurodomi jo parametrai
  output-layer name=output input=outl output-delay=0 include-log-softmax=true dim=$num_targets max-change=1.5

  # nurodoma konfigūracijos pabaigos vieta
  EOF

  # sudaroma galutinė konfigūracija su Kaldi moduliu xconfig_to_configs.py
  steps/nnet3/xconfig_to_configs.py --xconfig-file $dir/configs/network.xconfig --config-dir $dir/configs/

# konfigūracijos sudarymo lygmens L12 pabaiga
fi
```