LATVIJAS UNIVERSITĀTE
DATORIKAS FAKULTĀTE

Runātāju segmentēšana skaņas ierakstā, izmantojot neironu tīklus

BAKALAURA DARBS

Autors: **Dāvis Mednis**
Studenta apliecības nr.: dm14018
Darba vadītājs: M. dat. Artūrs Znotiņš

RĪGA 2018

ANOTĀCIJA

Runātāju segmentēšana skaņas ierakstā ir audio analīzes problēma, kas paredz ierakstā dzirdamu cilvēka runas fragmentu identificēšanu un grupēšanu gadījumos, kad vairākus fragmentus izteicis viens un tas pats runātājs. Runātāju segmentēšana ir aktuāla problēma audio ierakstu transkripcijas procesā, kur nepieciešams atbildēt uz jautājumu "kas runāja kad?".

Darbā izpētīts mākslīgo neironu tīklu un dziļās mašīnmācīšanās metožu potenciāls un iespējas runātāju segmentēšanas problēmas risināšanā. Tiek apskatīti gatavi runātāju segmentēšanas risinājumi un to darbības pamatprincipi. Praktiskajā daļā tika izveidots uz neironu tīkliem bāzētas runātāju segmentēšanas sistēmas prototips un datu kopa sistēmas apmācībai. Tika salīdzināti apskatīto runātāju segmentēšanas sistēmu rezultāti reālas darbības scenārijā un salīdzināti ar izstrādātā prototipa sniegumu.

No iegūtajiem rezultātiem tika secināts, ka spējīga runas segmentēšanas risinājuma izstrādāšanai nepieciešama kvalitatīva apmācības datu kopa. Tika secināts, ka šobrīd neeksistē kvalitatīvs un viegli lietojams uz neironu tīkliem bāzēts runātāju segmentēšanas risinājums, kas ir brīvi pieejams.

Atslēgvārdi: runātāju segmentēšana, audio analīze, dziļā mašīnmācīšanās, neironu tīkli

ABSTRACT

SPEAKER DIARIZATION IN AN AUDIO RECORDING USING NEURAL NETWORKS

Speaker diarization is an audio analysis problem that involves identifying and grouping audible human speech fragments if multiple utterances were made by the same speaker. Speaker diarization is a relevant issue in the process of transcribing audio recordings where it is necessary to answer the question "Who spoke when?".

The paper studies the potential of artificial neural networks, deep engineering techniques and the possibilities of solving the speaker segmentation problem using neural networks. Ready-made speech segmentation solutions and basic principles of operation are reviewed and their result baselines are obtained. The author develops a speaker segmentation system prototype based on artificial neural networks. A training dataset is created using freely available datasets. The author compares the results of the developed prototype with results of the existing solutions

The author concludes that a well made training dataset is required to train a performant neural network based solution. It was determined that currently there is no state-of-the-art solution for speaker diarization based on neural networks.

Keywords: speaker diarization, audio analysis, deep learning, artificial neural networks

SATURS

APZĪMĒJUMU SARAKSTS

Apzīmējums	Nozīme
MFCC	*Mel-frequency cepstral coefficients* - Mel-frekvenču cepstrālie koeficienti
VAD	*Voice activity detecion* – cilvēka runas noteikšana
RNT	*RNN – Recurrent neural network* - Rekurents neironu tīkls
LSTM	*Long Short-Term Memory* - garas īstermiņa atmiņas neironu tīkli
GRU	*Gated recurrent unit* – normētā rekurentā šūna
LIUM	*Laboratoire d'Informatique de l'Université du Mans* – runātāju diarizācijas rīks
TIMIT	runas skaņas ierakstu datu kopa angļu valodā

IEVADS

Ņemot vērā pieaugošo datu apjomu, ko ikdienā saražo mūsdienu sabiedrība, rodas nepieciešamība pēc datu apstrādes metodēm, kuru darbībai nav nepieciešama nepārtraukta cilvēka iesaiste. Ievērojama daļa datu, kas tiek saražoti ikdienā ir skaņas ieraksti, kuros dzirdama cilvēka runa. Cilvēka runas audio ierakstus var apstrādāt ar tādām metodēm kā transkripcija vai runas atpazīšana, kur viena no apstrādes apakšproblēmām ir runātāju segmentēšana.

Darba mērķis ir izpētīt esošus risinājumus runātāju segmentēšanai. Autors apskatīs gan uz neironu tīkliem bāzētus, gan citādi strādājošus risinājumus. Pašrocīgi tiks izstrādāts uz neironu tīkliem bāzēts prototips runātāju segmentēšanas risinājumam un salīdzinās izpētīto risinājumu veiktspēju, segmentējot reālā dzīves situācijā sastopamu skaņas ierakstu.

Lai izdotos sasniegt darba mērķus, tiek izvirzīti sekojoši darba uzdevumi:

- apzināt esošās runāju segmentēšanas metodes;
- noteikt runātāju segmentēšanas precizitātes bāzlīniju, izmantojot eksistējošos risinājumus;
- izveidot neironu tīkla apmācības datu kopu no brīvi pieejamām runas un trokšņu audio ierakstu datu kopām;
- pašrocīgi izstrādāt un apmācīt uz neironu tīkliem bāzētu runātāju segmentēšanas modeli;
- salīdzināt izstrādātā prototipa sniegtos rezultātus ar iepriekš iegūto bāzlīniju

Darba saturs ir sadalīts trīs nodaļās – runātāju segmentēšanas un esošo risinājumu apskats, neironu tīklu teorētiskās puses apskats un praktiskā darba apskata nodaļa. Pirmajā nodaļā tiek vispārīgi izklāstīta runātāju segmentēšanas problēma un tās apakšproblēmas, kā arī detalizēti apskatīti esošie segmentēšanas risinājumi. Otrajā nodaļa tiek apskatīta neironu tīklu un to uzbūve, un elementi, kas vēlāk tiek lietoti runātāju segmentēšanas prototipa izstrādē. Trešajā nodaļā autors apraksta veiktā praktiskā darba un pētījumu gaitu un rezultātus, tajā skaitā datu kopas izveidi, neironu tīkla prototipa arhitektūras izvēli, datu priekšapstrādi, neironu tīkla apmācības procesu un rezultātu novērtējumu.

1. RUNĀTĀJU SEGMENTĒŠANA

Runātāju segmentēšana skaņas ierakstā ir process, kura rezultātā tiek iegūta informācija par fragmentiem skaņas ierakstā, kur ir dzirdama cilvēka runa, kā arī brīžiem, kad notiek runātāja maiņa. Runātāju segmentēšana parasti tiek veikta līdztekus ar iegūto runas fragmentu klāsterēšanu, lai noteiktu, kurus runas fragmentus teicis viens un tas pats runātājs. Runātāju segmetēšanas procesu mēdz saukt arī par runātāju diarizāciju (*speaker diarization*).

Šī darba ietvaros tiek apskatīta cilvēka runas noteikšana un runātāja maiņas brīža noteikšana skaņas ierakstos, ar nosacījumu, ka noteikšanas sistēma ar konkrēto audio ierakstu vai runātājiem sastopas pirmo reizi. Tas nozīmē, ka sistēmai jāspēj ģeneralizēt un funkcionēt neatkarīgi no runātāju runas tembra.

1.1. Risināmās problēmas

Runas segmentēšanu var izdalīt trijās atsevišķi risināmās apakšproblēmās:

- cilvēka runas noteikšana;
- runātāja maiņas noteikšana;
- iegūto runātāju runas fragmentu klāsterēšana.

Cilvēka runas noteikšana (*VAD – voice activity detection*) paredz skaņas ieraksta analīzi, kuras rezultātā tiek iegūta informācija par laika momentiem ierakstā, kad ir dzirdama cilvēka runa. Segmentēšanas sistēmai jāspēj atšķirt ierakstā dzirdamus trokšņus, klusuma fragmentus un citas skaņas no cilvēka balss. Cilvēka runas noteikšana nav triviāls uzdevums, jo noteikšanas sistēmai jāspēj vispārināt, lai atpazītu dažādus cilvēka balss tembrus un atmestu fona trokšņus. Liela daļa runas noteikšanas algoritmi krasi zaudē savu precizitāti gadījumos, kad fona trokšņu līmenis ir augsts [1].

Runātāja maiņas noteikšana paredz iegūto runas fragmentu analīzi, lai iegūtu laika momentus ierakstā, kad beidz runāt viens runātājs un sāk runāt nākamais. Šīs apakšproblēmas lielākais izaicinājums ir skaņas fragmentu apstrāde, kuros dzirdami vairāk kā viens runātājs. Šādos gadījumos nevar precīzi noteikt vienu momentu laikā, kur notiek runātāja maiņa.

Pēdējais solis segmentēšanas procesā paredz iegūto fragmentu sadalīšanu runātāju grupās, kur katrā grupā jābūt viena runātāja izteiktiem runas fragmentiem. Tāpat kā runātāja maiņas noteikšanas problēmu, arī šo problēmu ir grūti risināt gadījumos, ja pārklājas divu vai vairāku runātāju balsis.

1.2. Eksistējoši risinājumi

Viens no populārākajiem risinājumiem runātāju segmentēšnai un diarizācijai ir LIUM (*Laboratoire d'Informatique de l'Université du Mans*) runātāju diarizācijas rīku kopums, kas radīts radio un TV pārraižu ierakstu apstrādei. LIUM rīku kopuma darbības pamatā ir skaņas ierakstu aprakstošo rakstura iezīmju statistiska analīze.

Kā skaņu aprakstošo rakstura iezīmju LIUM rīkā lieto mel-frekvenču cepstrālos koeficientus (*MFCC - Mel-Frequency Cepstral Coefficient*). MFCC ir labs veids, ka modelēt cilvēka dzirdes subjektīvo izpratni par skaņas signāla toņa augstumu un frekvenci. MFCC aprēķinu pamatā ir mel-frekvenču skala, kas apraksta skaņas signāla augstumu par pamatu ņemot cilvēka dzirdes un uztveres logaritmiskās īpašības. Ņemot vērā, ka cilvēka uztvere ir subjektīva, nav precīzas formulas, kā pārveidot frekvenci mel-spektrā, taču par vispārpieņemtu ir atzīta formula [2]:

$$Mel(f) = 2595log_{10}\left(1 + \frac{f}{700}\right),$$

kur $Mel(f)$ ir logaritmiska frekvenču skalas f reprezentācija.

MFCC tiek aprēķināti katram signāla 20 - 40 ms garam fragmentam, tos veidojot tā, ka tie savā starpā daļēji pārklājas. Katram fragmentam noskaidro mel-frekvenču spektru, veicot ātro Furjē transformāciju un pārveidojot iegūto lineāro frekvenču spektru mel-skalā. Pēc tam spektram piemēro 12 filtrus, kuru rezultāti sastāda MFCC. Matemātiski MFCC aprēķinu apraksta formula [3]:

$$C_n = \sqrt{\frac{2}{k}}\sum_{k=1}^{K}(logS_k)\cos\left[\frac{n(k-0.5)\pi}{k}\right], \qquad n = 1, 2, \dots N$$

kur $S_k(k = 1, 2, \dots K)$ ir piemēroto filtru rezultāti, bet N ir signāla vērtību skaits katrā skaņas fragmentā. LIUM rīkā tiek lietoti 13 koeficienti [4], kur kā 13. koeficienta vērtība tiek lietota enerģija, kuru iegūst kā logaritmu no signāla enerģijas vērtības katrai signāla vērtībai jeb

$$\{s_n, n = 1, \dots, N\}, E = log\sum_{k=1}^{N} s_n^2 \quad [3].$$

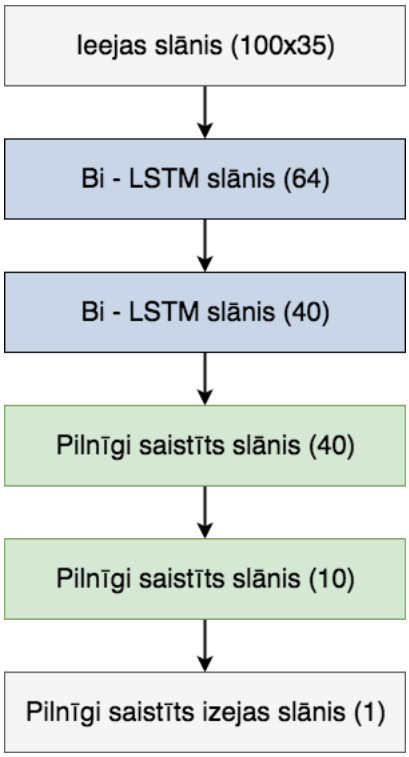
Skaņas ieraksta segmentācijai, runas noteikšanai un klāsterēšanai LIUM rīku kopa izmanto Viterbi algoritmu un slēptos Markova modeļus (*Hidden Markov models*), kā arī Gausa maisījuma modeļus (*Gaussian Mixture models*) [4].

1.3. Pētījumi ar neironu tīkliem

Līdz ar neironu tīklu lietojuma popularizāciju, pēdējos gados ir veikti pētījumi, kas piedāvā risināt runātāju segmentācijas problēmas, izmantojot neironu tīklus. Viens no piedāvātajiem risinājumiem runātāja maiņas noteikšanai ir Jina Rikvinga pētījumā minētais

risinājums, kas piedāvā problēmas risināšanai lietot divvirzienu LSTM neironu tīklu. Pētījuma autori piedāvā problēmas risināšanai pieiet ar virkņu marķēšanas metodi (*sequence labeling*).

Līdzīgi kā LIUM rīku kopā, arī šim neironu tīklu risinājumam skaņas ieraksts pirms došanas apstrādei neironu tīklam, tiek saskaldīts 32 ms garos fragmentos, kur katrs nākamais ar iepriekšējo pārklājas par 50%. Katram fragmentam tiek aprēķināti 11 MFCC koeficienti, to pirmais un otrais atvasinājums, kā arī fragmenta signāla enerģijas pirmais un otrais atvasinājums, kas kopā sastāda 35 rakstura iezīmes katram skaņas fragmentam.



1.1. att. ***Rikvinga pētījumā piedāvā arhitektūra***

Rikvinga pētījumā piedāvātā neironu tīkla arhitektūra (skat. 1.1. att.) sastāv no diviem savienotiem divvirzienu LSTM slāņiem, kas ļauj ieejas datus apstrādāt abos virzienos. Pēc LSTM slāņiem neironu tīklā iekļauti trīs pilnīgi saistītie slāņi, kas veido vairākslāņu perceptronu. Pēdējais pilnīgi saistītais slānis izmanto sigmoīda aktivācijas funkciju un atgriež varbūtību intervālā no 0 līdz 1, kur 1 nozīmē, ka konkrētajā fragmentā ir sastopama runātāja maiņa, bet 0, ka runātājs nemainās.

Apmācības procesā neironu tīklā tiek ievadītas 3,2 sekunžu garas fragmentu virknes, kur katra virkne ar katru nākošo pārklājās par 75%. Lai risinātu situācijas, kad runātāja maiņa nenotiek viena fragmenta ietvaros, treniņa datos pozitīvi marķēto vērtību apgabali tiek paplašināti aptuveni 500 ms uz katru pusi no maiņas momenta. Lielāks pozitīvo vērtību skaits datu kopā samazina arī klašu nelīdzsvarotību, kura pastāv, jo runātāja maiņas momenti skaņas

ierakstā sastopami krietni retāk nekā momenti, kur maiņa nenotiek. Krasi nevienlīdzīgs klašu paraugu skaits treniņa datos pasliktina apmācītā modeļa sasniedzamo rezultātu precizitāti.

1.4. Pielietojums praksē

Runātāju segmentācija ir aktuāla problēma, kad nepieciešams apstrādāt lielu daudzumu cilvēka runas skaņas ierakstu bez cilvēka iesaistes. Klasisks piemērs, kur runātāju segmentācija ir noderīga, ir sapulču vai citu notikumu transkripcija [5], kad to veic kāds cilvēks. Segmentētā skaņas ieraksta ir daudz vieglāk orientēties gan transkripcijas laikā, gan vēlāk, kad, atsaucoties uz transkripciju, kāds vēlas atrast attiecīgo vietu skaņas ierakstā.

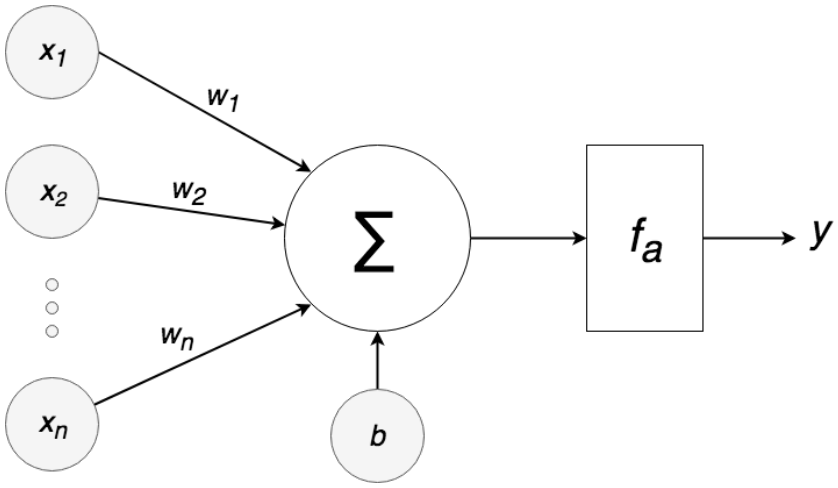
2. NEIRONU TĪKLI

Mākslīgais neironu tīkls (turpmāk – neironu tīkls) ir datu apstrādes sistēma, kas tuvināti modelē zīdītāju smadzenēs sastopamo bioloģisko neironu struktūru mazākos mērogos. Apjomīgā mākslīgajā neironu tīklā var būt vairāki simti tūkstoši datu apstrādes vienību jeb mākslīgo neironu, savukārt zīdītāja smadzenēs ir vairāki miljardi neironu [6].

Kā minēts Aganoviča-Kustrina publikācijā [7], neironu tīkli savas zināšanas gūst saskatot likumsakarības tiem padotajos ievades datos un savu funkcionalitāti iegūst apmācības, nevis programmēšanas ceļā. Neironu tīkls var saturēt simtiem vai tūkstošiem mākslīgo neironu, kas savā starpā savienoti ar koeficientiem jeb svariem un izkārtoti slāņos.

2.1. Mākslīgais neirons

Mākslīgais neirons ir neironu tīklu datu apstrādes pamatvienība, kas radīts kā abstrakts modelis bioloģiskam neironam. Katram neironam ir fiksēts skaits ieejas kanālu, kas ar neironu savienoti ar svērtām saitēm (skat. 2.1. att.).



2.1. att. ***Mākslīgais neirons***

Standarta gadījumā neironam ir $n + 1$ ieejas kanāli, kur n ir ieejas kanāli no iepriekšējiem slāņiem vai ieejas, bet 1 ir noslieces (*bias*) kanāls (b). Noslieces kanāls satur vērtību, kas nav atkarīga no ieejas datiem. Parasti noslieces kanāls sākotnēji satur vērtību 1, kura tīkla apmācības laika tiek koriģēta, līdzīgi kā tīkla svari (w). Tipiski neirona izejas vērtība tiek noteikta visu ieejas kanālu un to svaru reizinājumam, pielietojot neirona aktivācijas funkciju (f_a). Minētajā veidā iegūto neirona izejas vērtību (y) matemātiski apraksta vienādojums [8]:

$$y = f_a\left(\sum_{i=1}^{n} w_i x_i + b\right)$$

Neironu tīkla uzvedību nosaka ieejas kanālu svaru vērtības, neironu aktivācijas funkcijas un neironu izkārtojuma un savienojumu struktūra tīklā. Tīkla apmācības procesā tiek veikta ieejas saišu svaru korekcija, lai pie katriem ieejas datiem tuvinātu tīkla izejas vērtību vēlamajai.

Mākslīgais neirons funkcionē diskrētos laika soļos, ieejas vērtības tiek nolasītas un izejas vērtība tiek aprēķināta katram laika momentam atsevišķi [9]. Mākslīgos neironus mēdz dēvēt arī par perceptroniem. Vienslāņa perceptrons spēj apmācības rezultātā veikt tikai lineāru klasifikāciju, neatkarīgi no tā, vai neirona aktivācijas funkcija ir lineāra vai nelineāra. Šis pamatojams ar to, ka perceptrona izejā vienmēr būs tikai viena vērtība [8], [10].

2.2. Aktivācijas funkcijas

Mākslīgo neironu aktivācijas funkcijas nosaka galīgo neirona izejas vērtību , summējot tā ieejas vērtības. Aktivācijas funkcijas nodrošina neironu izejas vērtību normalizāciju un nenonākšanu ekstrēmos. Aktivācijas funkcijas atvieglo neironu tīkla spēju mācīties un pārlieku nepielāgoties pie ieejas datiem.

2.3. Neironu tīklu arhitektūra

Viens no aspektiem, kas nosaka neironu tīklu funkcionalitāti, ir tā neironu un to savienojumu izkārtojuma arhitektūra jeb topoloģija. Tipiski gan bioloģiskie, gan mākslīgie neironu tīkli ir izkārtoti slāņos, kur katrā no tiem ir galīgs skaits neironu. Mākslīgie neironi iedalās trīs tipos, atkarībā no tā, kā tie izvietoti neironu tīklu arhitektūrā:

- ieejas neironos – neironi, kas saņem ieejas vērtību no tīkla ārpuses;
- izejas neironos – neironi, kas veido tīkla izejas vērtību;
- slēptajos neironos – neironi, kas savienoti tikai ar citiem tīkla iekšienē esošiem neironiem.

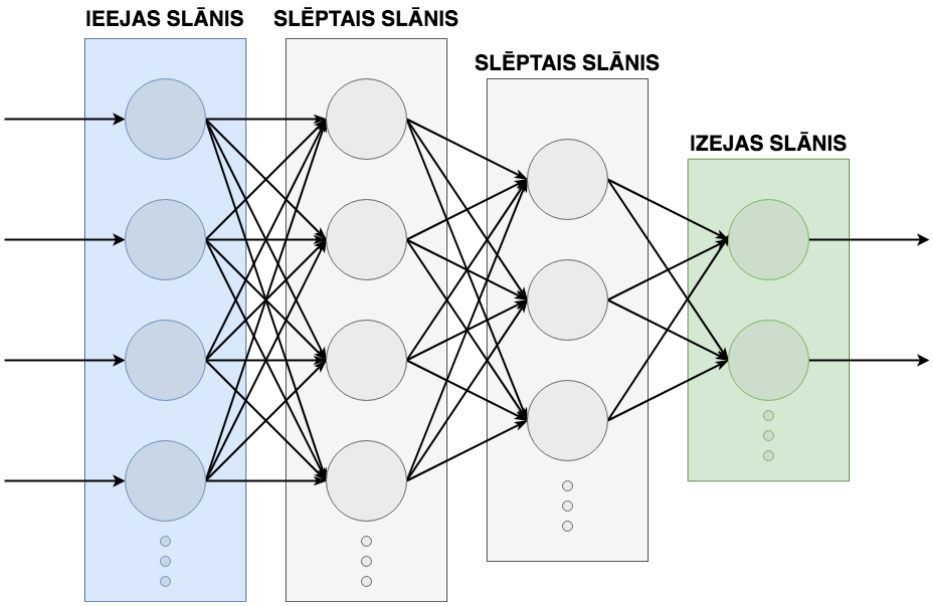
Līdzīgi kā neironi, arī tīkla slāņi iedalās līdzīgos tipos:

- ieejas slānis – slānis, kas sastāv no ieejas neironiem;
- izejas slānis – slānis, kas sastāv tikai no izejas neironiem;
- slēptie slāņi – slāņi, kas sastāv tikai no slēptajiem neironiem [11].

Neironu tīkli pēc uzbūves tiek dalīti divās pamatklasēs – vienvirziena neironu tīklos (*feedforward neural network*) un rekurentajos neironu tīkos (*recurrent neural network*). To galvenā atšķirība ir tā, ka vienvirziena neironu tīklos dati plūst tikai vienā virzienā – no ieejas uz izeju, bet rekurentajos neironu tīklos dati var plūst arī pretējā virzienā [12].

2.3.1. Vairākslāņu perceptrons

Vairākslāņu perceptroni ir viena no primitīvākajām vienvirziena neironu tīklu topoloģijām. Vairākslāņu perceptrons sastāv no vismaz trim slāņiem – ieejas slāņa, vismaz viena slēptā slāņa un izejas slāņa (skat. 2.2. att.). Katrs slānis ar sev līdztekus esošajiem ir pilnībā saistīts.



2.2. att. ***Vairākslāņu perceptrons***

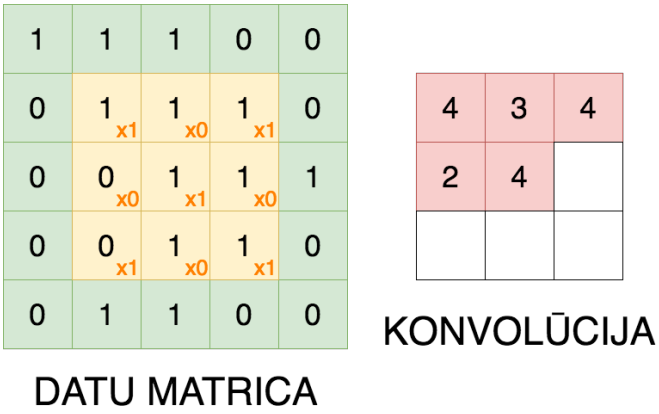
Vairākslāņu perceptrona galvenā priekšrocība, kas to atšķir no prasta perceptrona, ir spēja veikt nelineāru klasifikāciju. Tas ir iespējams, pateicoties faktam, ka vairākslāņu perceptrona slēptajos slāņos neironu aktivācijas funkcijas arī ir nelineāras.

2.3.2. Konvolūciju neironu tīkli

Konvolūciju neironu tīkli ir vienvirziena topoloģijas paveids, kas no parasta vairākslāņu perceptrona atšķiras ar to, ka tā uzbūvē ir viens vai vairāku konvolūciju slāņi. Standarta gadījumā konvolūciju slāņa ieejas datu izmērs ir $m \times m \times r$, kur m ir datu matricas platums un garums, bet r datu kanālu skaits. Katram konvolūciju slānim ir k filtri, kas ir izmērā $n \times n \times q$, kur n ir mazāks nekā datu matricas izmērs (m) un q ir vienāds ar r vai mazāks un var mainīties atkarībā no filtra [13].

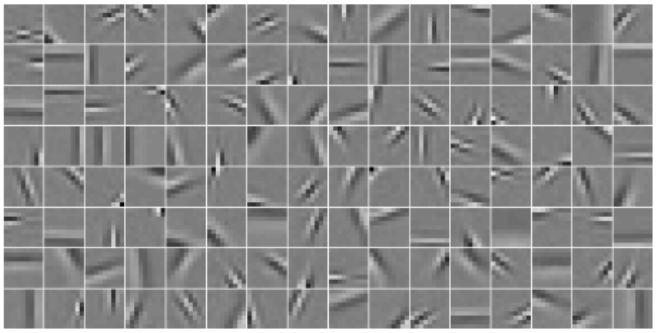
Konvolūciju slāņa izejas dati ir matrica, kas rodas veicot filtra konvolūciju uz visām ieejas datu matricas iespējamajām pozīcijām (skat. 2.3. att.). Konvolūcijas paredzētas konkrētu

ievades datu pazīmju konstatēšanai. Katrs konvolūciju filtrs ir atšķirīgs un paredzēts citu pazīmju noteikšanai. Filtra ietekmi nosaka tā šūnu vērtības un izkārtojums.



2.3. att. ***Konvolūciju filtra pielietošana***

Konvolūciju slāņus tipiski izmanto, piemēram, attēlu apstrādes, runas un skaņu atpazīšanas, kā arī dabiskās valodas apstrādes problēmu risināšanā. Konvolūcijas samazina neironu tīklam apstrādājamo datu apjomu, izceļot ieejas datos sastopamas raksturīgās iezīmes (skat. 2.4. att.), kuras var ietekmēt neironu tīkla izejas vērtības [14].



2.4. att. ***Filtru noteiktās pazīmes*** *[15]*

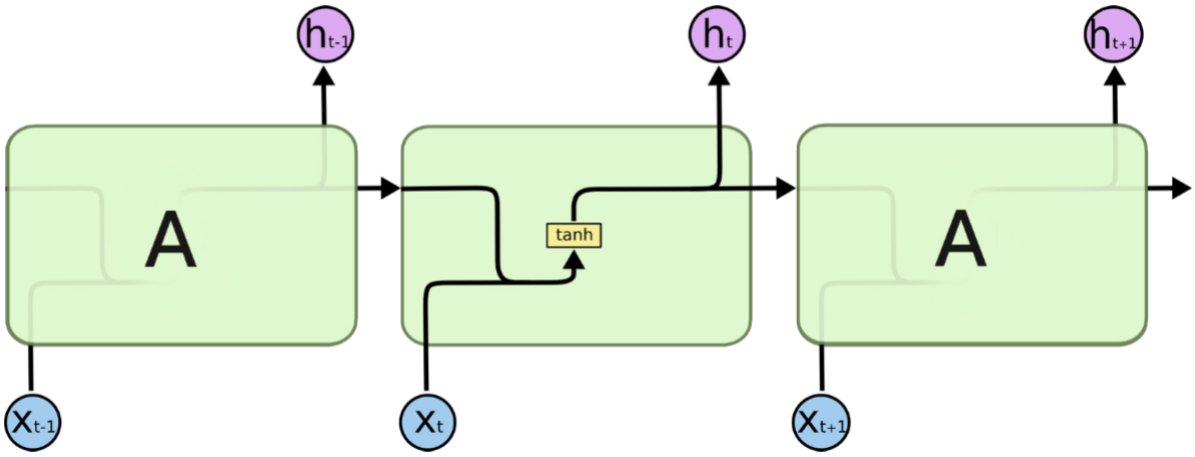
Tipiski kopā ar konvolūciju slāņiem tiek lietoti arī dimensionalitāti samazinošie slāņi (*pooling layers*). Dimensionalitāti samazinošais slānis, sadala datu matricu vienādos reģionos, kur katra reģiona vērtībām pielieto kādu transformācijas funkciju. Visbiežāk sastopamā transformācijas funkcija šādos slāņos ir, maksimuma funkcija, kas izejā padod tikai lielāko no reģiona vērtībām. Vēsturiski tika lietota arī aritmētiskā vidējā transformācijas funkcija, taču pētījumu [16] rezultātā secināts, ka maksimuma funkcija sniedz labākus rezultātus. Ieejas datu dimensionalitātes samazināšana ne tikai vienkāršo apstrādājamos datus, samazinot veicamo aprēķinu daudzumu, bet arī aizsargā neironu tīklu pret pārmācīšanos (*overfitting*) [17].

Veiktā literatūras apskata laikā [18] tika noskaidrots, ka dimensionalitāti samazinošos slāņus var aizstāt ar konvolūciju slāņiem, kas piemēro filtrus ar lielāku izkliedējumu, tādējādi samazinot tīkla sarežģītību.

2.3.3. Rekurentie neironu tīkli

Rekurentie neironu tīkli (RNT), pretēji vienvirziena neironu tīkliem, spēj informāciju nodot ne tikai no ieejas uz izeju, bet arī nākamajam laika solim. Iespēju datus padot nākamajiem laika soļiem var dēvēt arī par neironu tīkla atmiņu. Līdztekus katra RNT neirona ieejas datiem laika solī t, katrs neirons ieejā saņem arī savu iepriekš aprēķināto izejas vērtību laika solī $t - 1$. Šādā veidā RNT ir spējīgi ietekmēt izejas vērtības atkarībā no iepriekšējo laika soļu izejas vērtībām.

Standarta rekurentajiem neironu tīkliem (skat. 2.5. att.) ir raksturīgas tādas nepilnības kā "eksplodējošie" gradienti un zūdošie gradienti. Neironu tīklu kontekstā gradienti apraksta neirona izejas vērtības izmaiņu atkarībā no ieejas vērtības izmaiņas. "Eksplodējošo" gradientu gadījumā, apmācības algoritms piešķir pārāk lielus neirona ieejas kanālu svarus, kas noved pie sliktiem tīkla darbības rādītājiem. Šo problēmu var viegli risināt, neņemot vērā pārāk lielus gradientus. Zūdošo gradientu problēma izpaužas kā pārāk mazi gradienti, kas liedz tīklam mācīties vai padara apmācības procesu ļoti lēnu. Zūdošo gradientu problēmas risināšanai tika izgudroti LSTM neironu tīkli [19], kas sīkāk aprakstīti 2.2.4 nodaļā. Šī problēmas apmācībās procesā liedz RNT saglabāt atmiņā izejas vērtībās no senākiem laika soļiem, tās ar laiku vairs netiek ņemtas vērā [20].

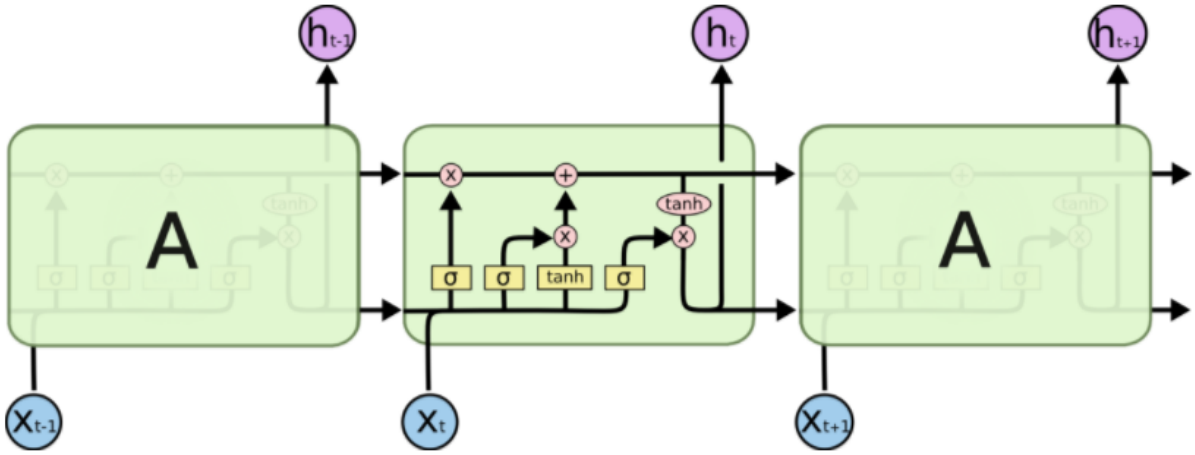


2.5. att. ***RNN topoloģija*** *[21]*

RNT no citām neironu tīklu topoloģijām izceļas ar labākiem rezultātiem problēmu risināšanā, kur tīkla ievades dati ir virkne vai saraksts, kas var būt, piemēram, teksts, audio celiņš vai video ieraksts [22].

2.3.4. LSTM neironu tīkli

LSTM jeb garas īstermiņa neironu tīkli (Long Short-Term Memory) ir rekurento neironu tīklu paveids, kas tika radīts, lai risinātu standarta RNT raksturīgos trūkumus, kas liedz tiem veiksmīgi saglabāt atmiņā vērtības no senākiem laika soļiem.



2.6. att. ***LSTM šūnas uzbūve*** *[21]*

Standarta LSTM šūna (skat. 2.6. att.) sastāv no četriem neironu tīkla slāņiem, kas atbild šūnas stāvokli un izejas vērtību:

- pirmais slānis, kas ar sigmoīda aktivācijas palīdzību nosaka, kādu informāciju dzēst no šūnas atmiņas;
- otrais slānis, kas nosaka kādas vērtības no ievades datiem tiks saglabātas šūnas atmiņā;
- trešais slānis, kas veic šūnas atmiņas atjaunināšanu ar iepriekš noteiktajām vērtībām;
- ceturtais slānis, kas nosaka kāda būs šūnas izejas vērtība, atkarībā no šūnas atmiņas stāvokļa [23].

LSTM tīklu trūkums ir fakts, katrā šūnā ir liels skaits apmācāmu parametru, līdz ar to LSTM apmācība ir lēns process, kam nepieciešamas lielas datu kopas, lai tīkls spētu vispārināt un pārlieku nepielāgotos. Lai uzlabotu LSTM tīklu veiktspēju, strādājot ar mazām datu kopām, 2014. gadā [24] tika izgudrotas GRU (*Gated Recurrent Unit*) šūnas, kuru struktūra ir vienkāršāka par LSTM šūnām un tās satur mazāk apmācāmu parametru.

2.4. Neironu tīklu apmācība

Lai neironu tīkls varētu sākt pildīt no tā sagaidīto uzdevumu, tam ir jāiemāca nepieciešamā funkcionalitāte. Neironu tīkla apmācīšana notiek tīklā ievadot lielu daudzumu

informācijas, kas saistīta ar risināmo problēmu. Neironu tīkls, datiem plūstot cauri tā slāņiem, transformē tam padotos ieejas datus izejas datos. Iegūtie izejas dati tiek salīdzināti ar apmācības kopas ieejas datu marķējumiem jeb vēlamajām vērtībām. Balstoties uz iegūto izejas datu nobīdes no vēlamajiem datiem, ar atgriezeniskās saites palīdzību tiek koriģēti neironu tīkla svari. Šāds process tiek atkārtots iterācijās līdz brīdim, kad tiek sasniegti iespējami labi neironu tīkla atgrieztie rezultāti.

Ja tīkla apmācība tiek turpināta pārāk ilgi, tīkls var sākt pārlieku pielāgoties tā ieejas datiem, tādējādi zaudējot spēju vispārināt un reaģēt uz nelielām variācijām ieejas datos. Lai samazinātu pārliekas pielāgošanās risku, apmācības datu kopu parasti sadala divās daļās – apmācības datos un validācijas datos. Apmācības dati tiek lietoti pašā apmācības procesā tīkla svaru koriģēšanai, bet validācijas datus lieto, lai pārbaudītu tīkla veiktspēju un identificētu brīdi, kad tīkls sāk pārlieku pielāgoties apmācības datiem.

2.4.1. Pārraudzītā apmācība

Pārraudzītā mašīnmācīšanās paredz nepārtrauktu atgriezeniskās saites plūsmu neironu tīkla ietvaros. Pārraudzītajā mācīšanās procesā treniņa datu kopai jāsatur ievades datiem atbilstošie sagaidāmie izvades dati jeb marķējumi. Tīkla sniegtā rezultāta uzlabošanai tiek lietota optimizācijas funkcija, kas koriģē tīkla svarus, lai samazinātu nobīdi, kas ir starp tīkla izejas datiem un vēlamajiem izejas datiem apmācības kopā. Pārraudzītās mācīšanās vājā puse ir marķētas datu kopas nepieciešamība, kuras izveide var prasīt daudz, potenciāli dārga, manuāla darba, turpretim stiprā puse ir salīdzinoši mazāks nepieciešamais datu apjoms.

2.4.2. Nepārraudzītā apmācība

Nepārraudzītā mašīnmācīšanās paredz neironu tīkla patstāvīgu nemarķētas datu kopas apstrādi, kā vienīgo atgriezenisko saiti izmantojot novērtējuma funkciju, kas ļauj novērtēt iegūto rezultātu pret sagaidīto. Bez atgriezeniskās saites, apmācības ceļā, tīkls patstāvīgi saskata treniņa datos likumsakarības, kas ļauj saņemtos datus sagrupēt loģiskās grupās vai izcelt speciālos gadījumus, kad kāda vērtību grupa īpaši atšķiras no citiem datiem. Tādejādi var veikt datu analīzi milzīgām datu kopām bez cilvēka iesaistes. Nepārraudzītās apmācības vājā puse ir nepieciešamais lielais datu apjoms, lai būtu iespējams noteikt likumsakarības.

2.4.3. Stimulētā apmācībā

Stimulētajam mācīšanās procesam nav nepieciešama treniņa datu kopa, tā vietā eksistē aģents, kas ģenerē neironu tīklā ievadāmos datus atkarībā no aģentam dotās ievades – veiktajām darbībām. Pēc katras apmācības iterācijas aģents novērtē iterācijas veiksmīgumu un kā

atgriezenisko saiti tīklam sniedz vērtējumu. Apmācības mērķis ir maksimizēt sniegto vērtējumu variējot ar aģentam padoto ievadi.

2.5. Neironu tīklu pielietojumi

Mākslīgos neironu tīklus pielieto problēmām, kuru risināšanā nepieciešama abstrakta pieeja, kas apgrūtina šīs problēmas risināt lietojot tradicionālos algoritmus. Neironu tīklu pielietošanas priekšrocība ir tā darbības principa līdzība cilvēka smadzeņu darbības principam, kas ļauj ievades datos saskatīt likumsakarības, atpazīt pazīmes un vispārināt. Tādejādi neironu tīkla izstrādes procesā nav jāveic konkrētam uzdevumam specifisku detaļu apstrāde, jo tīkls visu svarīgo informāciju mērķa sasniegšanai spēj apstrādāt patstāvīgi [25].

Datu klasifikācija ir viena no visbiežāk, ar mākslīgajiem neironu tīkliem risinātajām problēmām. Datu klasifikācijas problēmas ietver pazīmju un virkņu atpazīšanu datos, tai skaitā, audio ierakstos, attēlos un tekstā. Pazīmju un virkņu atpazīšana ļauj izveidot neironu tīklu bāzētas audio, attēlu un teksta atpazīšanas un klasificēšanas sistēmas, kuras savā starpā kombinējot atsevišķos var sasniegt cilvēka līmenim līdzīgus rezultātus.

3. PRAKTISKAIS DARBS

Darba praktiskajā daļā autors ir patstāvīgi izstrādājis uz neironu tīkliem bāzētu risinājumu runātāju segmentēšanai.

Risinājuma izstrādes gaitā, balstoties uz līdzīgiem pētījumiem, tika eksperimentāli piemeklēta neironu tīkla arhitektūra, izveidota apmācības datu kopa no brīvi pieejamiem resursiem, tika veikta datu kopas priekšapstrāde un pielāgošana neironu tīkla vajadzībām, un tika apmācīts izstrādātais neironu tīkls. Autora izstrādā risinājuma pirmkods ir pieejams platformā *Github*: *github.com/dmednis/speaker-segmenter.*

Lai spētu novērtēt izstrādā risinājuma veiktspēju, tika noskaidrota segmentēšanas precizitātes bāzlīnija, izmantojot esošos brīvi pieejamos risinājumus. Tika veikta iegūto rezultātu salīdzināšana.

3.1. Datu kopa

Ņemot vērā, ka Rikvinga pētījumā lietotās apmācības datu kopas ETAPE iegūšana ir maksas pakalpojums, autors pieņēma lēmumu izstrādāt savu apmācības datu kopu no brīvi pieejamiem resursiem. Lai atkārtotu Rikvinga pētījumā iegūtos rezultātus un apmācītu neironu tīkla modeli runātāja maiņas brīža noteikšanai, nepieciešami marķēti audio ieraksti, kur dzirdami pēc iespējas vairāki runātāji. Runas noteikšanas modeļa apmācīšanai, savukārt, nepieciešami marķēti skaņas ieraksti, kas kategorizēti divās klasēs – cilvēka runas fragmentos un fragmentos, kur nav dzirdama cilvēka runa.

Izvērtējot tīmeklī brīvi pieejamās datu kopas, tika izvēlētas četras, kas atbilst uzstādītajiem kritērijiem:

- *TIMIT* runas korpuss – angļu valodā ierakstīts runas korpuss, kas sastāv no 630 runātāju ierunātiem skaņas fragmentiem. Kopējais skaņas ierakstu apjoms ir aptuveni piecas stundas [26].
- *LJ Speech* datu kopa – angļu valodā ierakstīts viena runātāja runāts teksts. Sastāv no 13100 skaņas fragmentiem, kas kopā sastāda 24 stundas [27].
- *LibriSpeech* runas korpuss – angļu valodā apkopots runas korpuss, kas sastāv no audio grāmatu fragmentiem, kurus runājuši dažādi runātāji. Kopā korpusā sastopami 2484 runātāju ierunāti runas fragmenti un kopējais skaņas ierakstu ilgums ir aptuveni 1000 stundas, taču šī praktiskā darba ietvaros tika lietota tikai daļa no visas datu kopas, kas sastādīja apmēram 20 stundas skaņas ierakstu. [28].
- *UrbanSound8K* datu kopa – skaņas ierakstu kopa, kur apkopoti 10 dažādu pilsētvidē sastopamu skaņu audio ieraksti. Datu kopā ir tādas skaņu klases kā ielu

mūzika, auto braukšanas troksnis un citas. Kopējais skaņas ierakstu garums ir 27 stundas [29]. Šī skaņu kopa tika izmantota, lai apmācītu runas noteikšanas modeli atpazīt ierakstus, kur nav sastopama cilvēka runa, bet ir citi trokšņi.

Papildu tīmeklī atrastajām datu kopām, autors ierakstīja apmēram stundu garu ierakstu ar LU Datorikas fakultātes 303. telpas fona troksni, kur nav dzirdama cilvēka runa. Šis ieraksts, līdzīgi kā pilsētvides skaņas, tika lietots runas noteikšanas modeļa apmācībā.

3.1.1. Datu priekšapstrāde

Ņemot vērā, ka katra no tīmeklī iegūtajām datu kopām bija strukturēta citādi, pirms tālākas datu apstrādes autors veica datu struktūras normalizāciju. Iegūtie runas dati tika sagrupēti pa runātājiem, tai skaitā apvienojot katra runātāja ierakstītos skaņas klipus vienā. Grupēšana pa runātājiem nepieciešama, lai varētu uzģenerēt apmācības datu kopu priekš runātāja maiņas brīža noteikšanas. Visi trokšņu un klusuma dati tika apvienoti vienuviet, to grupēšana nav nepieciešama, jo darba ietvaros risināto problēmu risināšanai nav būtiska trokšņu piederība konkrētai klasei.

Autors veica datu augmentāciju ar mērķi palielināt izveidojamo runātāju datu kopu un vairotu datu dažādību. Tika izvēlēti četri reālās situācijās sastopami fona trokšņu skaņas ieraksti un katrs no tiem apvienoti ar katru no *TIMIT* datu kopas runas fragmentiem, tādējādi palielinot no *TIMIT* datu kopas iegūto skaņas ierakstu apjomu piecas reizes. Ņemot vērā, ka visas lietotās brīvi pieejamās datu kopas satur tīrus skaņu ierakstus – bez fona trokšņiem, pievienojot runas fragmentiem fona trokšņus mākslīgi, tiek uzlabota iespēja neironu tīklu apmācīt vispārināt un izšķirt runas fragmentus arī gadījumos, ja ieraksts nav tīrs. Skaņas ierakstu apvienošanai tika izmantota atvērtā koda bibliotēka *audio_degrader* [30].

Lai izveidoto treniņa datu kopu varētu veiksmīgi lietot neironu tīkla apmācībai, bija nepieciešams veikt skaņas ierakstu rakstura iezīmju aprēķināšanu. Autors sākotnēji izvēlējās balstīties uz Rikvinga pētījumu, kur rakstura iezīmju aprēķināšana tika veikta 32 ms gariem fragmentiem, kuri pārklājas ar katru nākamo par 16 ms jeb 50% [31]. Taču, veicot pirmos apmācības eksperimentus, tika secināts, ka labākus rezultātus iespējams sasniegt, izvēloties vērtību aprēķināšanu veikt garākiem fragmentiem.

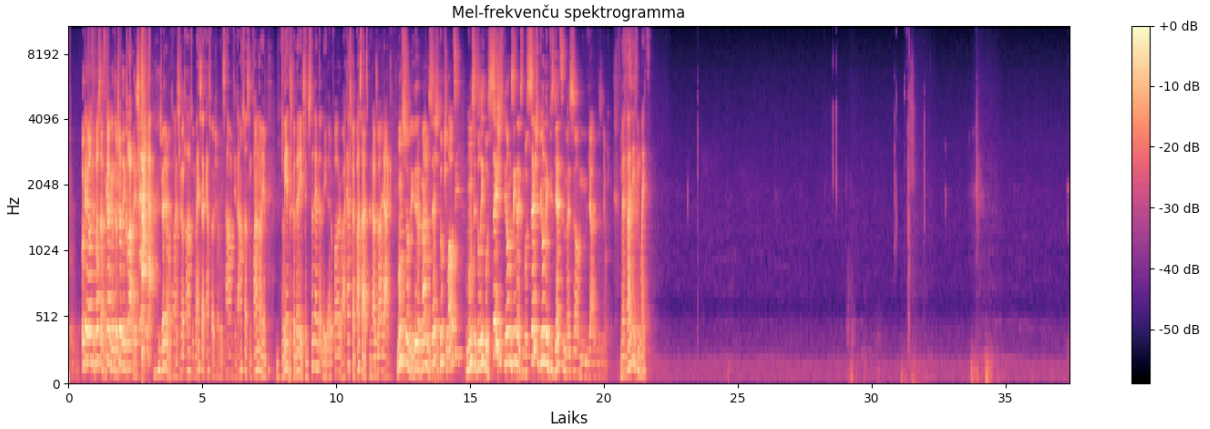
Eksperimentālā kārtībā autors nonāca pie risinājuma, kur rakstura iezīmes tiek aprēķinātas 100 ms gariem fragmentiem, kas ar katru nākamo pārklājas par 75 ms jeb 75%. Tāpat, atšķirībā no Rikvinga pieejas, eksperimentālā kārtā tika izvēlēts kā rakstura iezīmes lietot mel-frekvenču spektrogrammu 64 joslu platumā un tās pirmo atvasinājumu, kas kopā sastāda 128 vērtību vektoru vienam skaņas fragmentam. Mel-frekvenču spektrogrammas pirmais atvasinājums apraksta mel-frekvences izmaiņu laika vienībā [32].

Datu priekšapstrāde tika veikta *Python 3* programmēšanas vidē [33]. Audio failu manipulācijai un rakstura iezīmju aprēķināšanai tika izmantota *librosa* bibliotēka [34]. Datu manipulācijām atmiņā tika izmantota bibliotēka *numpy* [35]. Lai audio failu apstrāde nebūtu jāveic pie katras apmācāmā tīkla testēšanas iterācijas, apstrādātie rakstura iezīmju dati tika saglabāti ierīces atmiņā, izmantojot *pickle* [36] datu struktūru serializācijas formātu.

3.2. Risinājuma piemeklēšana

Balstoties uz Jina Rikvinga pētījumu un izstrādāto risinājumu runātāja maiņas noteikšanai, izmantojot LSTM neironu tīklus [31], autors nolēma izmantot Rikvinga piedāvāto tīkla arhitektūru un skaņas rakstura iezīmju noteikšanas veidu. Autors savu izstrādāto risinājumu mēģinās atkārtot pētījumā sasniegtos rezultātus runātāja maiņas noteikšanā un apmācīt modeli runas noteikšanas problēmas risināšanai.

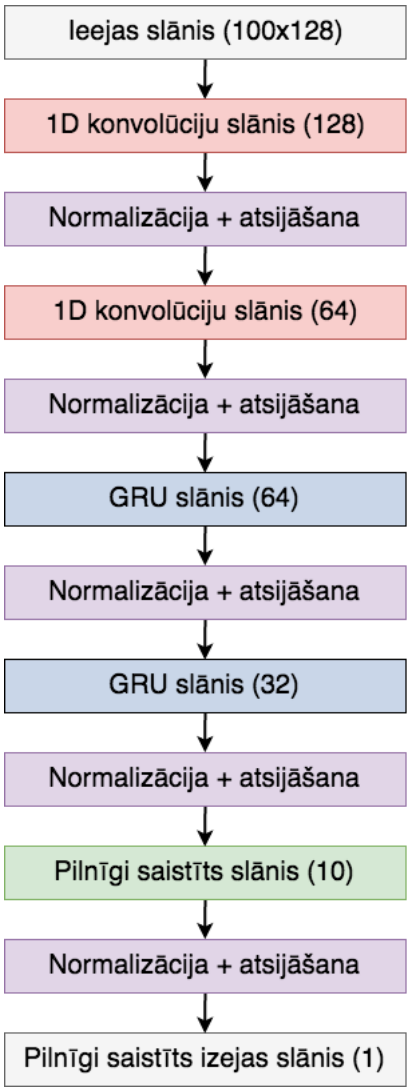
Veicot pirmos eksperimentus, izmantojot Rikvinga piedāvāto arhitektūru, tika iegūti neapmierinoši zemi runas noteikšanas rezultāti jeb aptuveni 67%, veicot novērtēšanu pret marķētu testa datu kopu. Cenšoties rast risinājumu zemajai precizitātei, tika noskaidrots, ka MFCC aprēķinu procesā tiek zaudēta informācija par skaņas ierakstu, kura būtu noderīga tīkla apmācības procesā. MFCC kā skaņas datu rakstura iezīmes tika izmantotas skaņas atpazīšanas sistēmās, kas bāzētas uz slēpto Markova modeļu un Gausa maisījuma modeļu bāzes, jo minētajos gadījumos, lai sasniegtu labu rezultātu ir nepieciešami ieejas dati, kas ir nekorelēti savā starpā. Neironu tīklu risinājumos datu savstarpēja korelācija rezultātu neietekmē, tāpēc balstoties uz Muhammeda Huziafaha pētījumu [37], autors pieņēma lēmumu par skaņas rakstura iezīmju reprezentāciju izmantot mel-frekvenču skalas spektrogrammu (skat. 3.1. att.).



3.1. att. ***Skaņas fragmenta mel-frekvenču spektrogramma***

Līdztekus mel-spektrogrammu ieviešanai apmācības procesā, eksperimentu gaitā tika uzlabota arī Rikvinga piedāvātā neironu tīkla arhitektūra (skat. 3.2. att.). Balstoties uz

Huziafaha [37] pētījumu par konvolūciju neironu tīklu lietošanu skaņas ierakstu atpazīšanai, autors pieņēma lēmumu papildināt Rikvinga piedāvāto arhitektūru ar diviem konvolūciju slāņiem uzreiz pēc tīkla ieejas slāņa. Ar mērķi paātrināt tīkla apmācības procesu un samazināt pārliekas pielāgotības risku, LSTM slāņi tika aizstāti ar GRU slāņiem, kam pie mazām datu kopām vajadzētu uzlabot iegūtos rezultātus [24], kā arī starp visiem slāņiem tika ievietoti datu atsijāšanas slāņi un aktivāciju normalizācija slāņi.

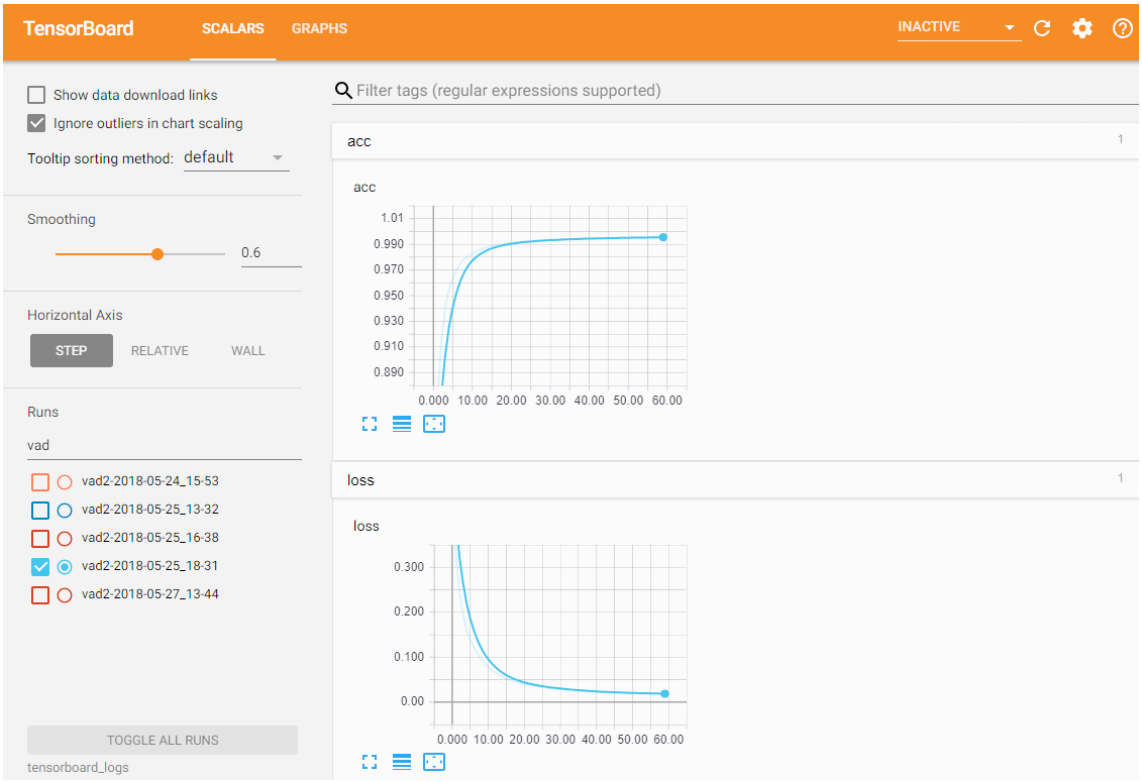


3.2. att. ***Autora piedāvātā neironu tīkla arhitektūra***

Ņemot vērā sarežģījumus un nepieciešamos uzlabojumus, ar kuriem autoram nācās saskarties, izstrādājot runas noteikšanas un runātāja maiņas noteikšanas risinājumus, tika nolemts neveikt runas fragmentu grupēšanu jeb klāsterēšanu šī praktiskā darba ietvaros. Kā viens no galvenajiem iemesliem klāsterēšanas neveikšanai ir nesekmīgie mēģinājumi un negatīvie rezultāti runātāju segmentu iegūšanā.

3.3. Apmācības process

Autors neironu tīkla apmācību, tāpat kā datu priekšapstrādi, veica programmēšanas vidē *Python 3*, izmantojot bibliotēku *Tensorflow* [38] un tai paredzēto *keras* [39] augsta līmeņa programmātisko saskarni. Datu priekšapstrāde un pirmie tīkla apmācības eksperimenti tika veikti uz *Apple MacBook Pro* datora, taču, eksperimentu gaitā palielinot apmācāmā tīkla sarežģītību un dziļumu, katra apmācības iterācija kļuva pārāk ilga, lai būtu iespējams atvēlētajā laikā sasniegt uzstādītos mērķus. Sadarbībā ar LU Datorikas fakultātes projektu studiju "DF LAB", eksperimentu un apmācības platforma tika pārcelta uz jaudīgāku datoru, kas aprīkots ar *Nvidia GeForce GTX 1080 Ti* grafisko karti. Izmantojot grafiskās kartes lielo paralēlās skaitļošanas jaudu, vienas apmācības iterācijas ilgumu izdevās samazināt aptuveni 20 reizes.



3.3. att. ***Tensorboard rīka saskarne***

Lai sekotu līdzi modeļa apmācības progresam, tika lietots *Tensorboard* vizualizācijas rīks (skat. 3.3. att.). Progresa izvērtēšanai tika lietotas šādas mērījumu vērtības:

- apmācības precizitāte – tīkla izejas datu atbilstība treniņa datu kopas marķējumam;
- apmācības kļūda – tīkla kļūdas funkcijas rezultāts apmācot uz treniņa datu kopu;

- validācijas precizitāte - tīkla izejas datu atbilstība validācijas datu kopas marķējumam;
- validācijas kļūda – tīkla kļūdas funkcijas rezultāts pārbaudot pret validācijas datu kopu.

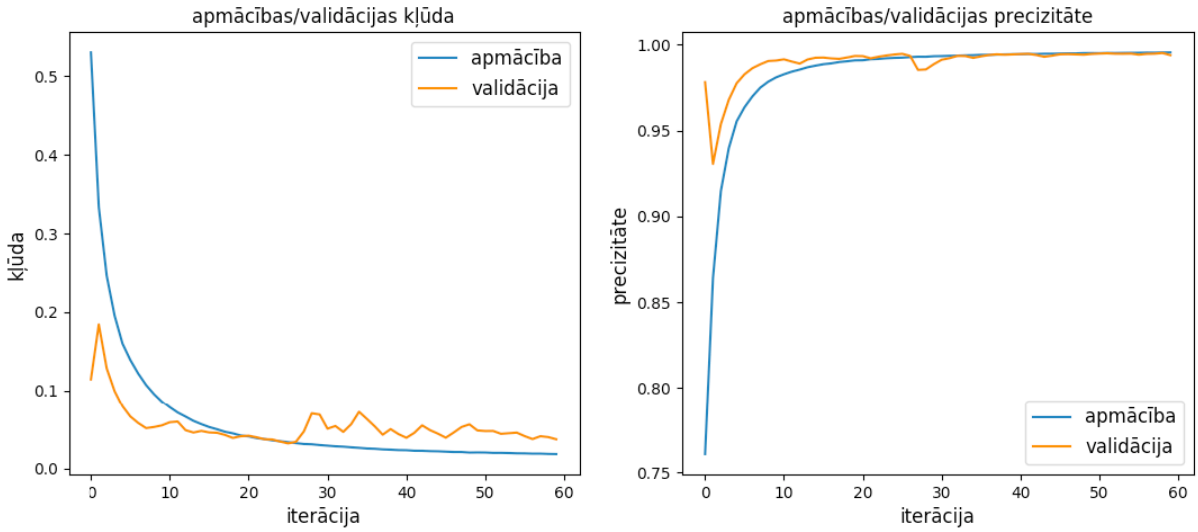
Iepriekš minētās mērījumu vērtības var izmantot, lai noteiktu apmācāma tīkla veiktspēju un to vai tas apmācības procesā nav pārlieku pielāgojies apmācības datiem. Pazīme, kas liecina, ka tīkls sāk pārlieku pielāgoties, ir validācijas kļūdas palielināšanās, kad apmācības kļūda turpina samazināties. Šajā brīdī apmācības procesu vajadzētu apturēt, jo tīkla labākā veiktspēja pie attiecīgajiem apmācības parametriem un apstākļiem ir sasniegta [40].

3.3.1. Runas noteikšanas modeļa apmācība

Lai veiktu runas noteikšanas modeļa apmācību, visa pieejamā runas apmācības datu kopa tika apvienota, jo runas noteikšanai nav svarīga balss piederība. Visi runas dati tika marķēti ar vērtību 1. Līdzīgi arī trokšņu un klusuma apmācības datu kopas tika apvienotas un marķētas ar vērtību 0. Runas noteikšanu šis modelis risinās kā bināras klasifikācijas problēmu katram ieejas datu fragmentam. Pirms apmācības sākuma tika atšķelta daļa no abām apmācības datu kopām, kas tika izraudzīta par validācijas kopu. Validācijas kopu tīklā apmācības procesā izmanto kā etalonu tīkla precizitātes noteikšanai starp apmācības iterācijām. Validācijas kopā nevajadzētu būt datiem, kas sastopami apmācības kopā, lai tīkls pārlieku nepielāgotos pie šiem datiem un spētu vispārināt.

Apmācības procesā neironu tīklam apmācības dati tika padoti kopās pa 100 fragmentiem, kur katrs fragments ir 100 ms garš un ar katru nākamo pārklājas par 75%. Katram fragmentam ir tam atbilstošs runas vai ne-runas marķējums.

Apmācības process norisinājās 60 iterācijas un pēc katras iterācijas tika veikta tīkla svaru saglabāšana. Apmācības ilgums bija aptuveni 120 minūtes.



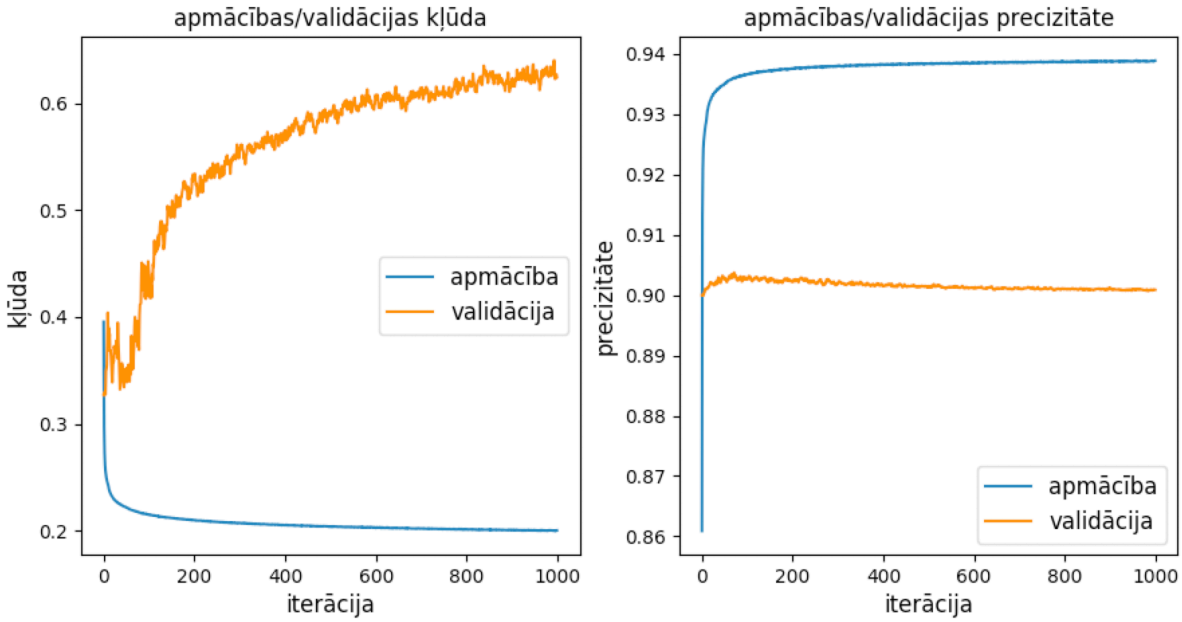
3.4. att. ***Runas noteikšanas tīkla apmācības rezultāti pa iterācijām***

Pēc apmācības beigām tika izvērtēti rezultāti un secināts, ka tīkls vislabāko precizitāti sasniedza aptuveni 25. apmācības iterācija, kad tīkla validācijas kļūda bijusi viszemākā, bet pēc tam sākusi palielināties, kas liecina, ka tīkls sācis pārlieku pielāgoties ievades datiem (skat. 3.4. att.).

3.3.2. Runātāja maiņas noteikšanas modeļa apmācība

Runātāja maiņas noteikšanas modeļa apmācībā tika izmantoti tikai no *TIMIT* un *LibriSpeech* datu kopām iegūtie skaņas fragmenti. Ņemot vērā, ka datu kopas skaņu fragmenti bija grupēti pēc runātāja piederības, autoram bija viegli ģenerēt datu kopas marķējumu, kas norādītu uz runātāja maiņas vietu. Tāpat kā runas noteikšanas modelis, arī šis modelis runātāja maiņas vietu nosaka, bināri klasificējot tam ieejā padotos runas fragmentus, ar izejas vērtību 1 atzīmējot runātāja maiņas punktus. Autors ņēma vērā Rikvinga pētījumā ieteikto risinājumu klāšu nelīdzsvarotības risināšanai [31], kas paredz paplašināt runātāja maiņas marķējuma punktus 500 ms uz katru pusi no runātāja maiņas punkta. Neskatoties uz vērā ņemtajiem ieteikumiem, pēc datu kopas marķēšanas pozitīvi marķētie kadri sastādīja tikai aptuveni 1,5% no kopējā skaita. Lai palielinātu pozitīvi marķēto kadru skaitu, autors izlēma runātāju fragmentus saskaldīt sīkākos 12,5 sekunžu fragmentos un sajaukt savā starpā tā, ka pēc kārtas nebūtu divu fragmentu no viena runātāja. Šāda apstrāde palielināja pozitīvi marķēto kadru īpatsvaru līdz 10% no kopējā skaita.

Apmācības process tika veikts ar tādiem pašiem parametriem, ka runas noteikšanas modeļa apmācības gaitā. Pēc 60 apmācības iterācijām tīkla precizitāte nostabilizējās uz 90% robežas. Veicot apmācītā modeļa testēšanu un reālu skaņas ierakstu un ņemot vērā, ka pozitīvo marķējumu īpatsvars ir 10%, tika secināts, ka modelis visus kadrus marķē negatīvi.



3.5. att. ***Runātāju maiņas brīžu noteikšanas modeļa apmācības rezultāti pa iterācijām***

Eksperimentālā kārtā tika izmēģinātas dažādas iespējamās tīkla arhitektūras variācijas un apmācības parametri, taču autoram neizdevās iegūt pozitīvu apmācības rezultātu. Eksperimentālos nolūkos tika veikta tīkla apmācība 1000 iterāciju garumā (skat. 3.5. att.), kur tika novērots, ka tīkla precizitāte arī pēc lielāka iterāciju skaita neuzlabojas.

3.4. Rezultātu pēcapstrāde

Iegūto rezultātu kopa pirms datu analizēšanas tika normalizēta, jo tīkla ieejā tika padoti ir fragmenti, kas savā starpā pārklājas. Kadriem, kuriem tīkls aprēķinājis vairākas izejas vērtības, tika aprēķināta aritmētiski vidējā vērtība. Par pozitīvi marķētiem tika pieņemti tie kadri, kam tīkls izejas vērtību piešķīris vismaz 0,5. Pārējie kadri tika atzīti par negatīvi marķētiem.

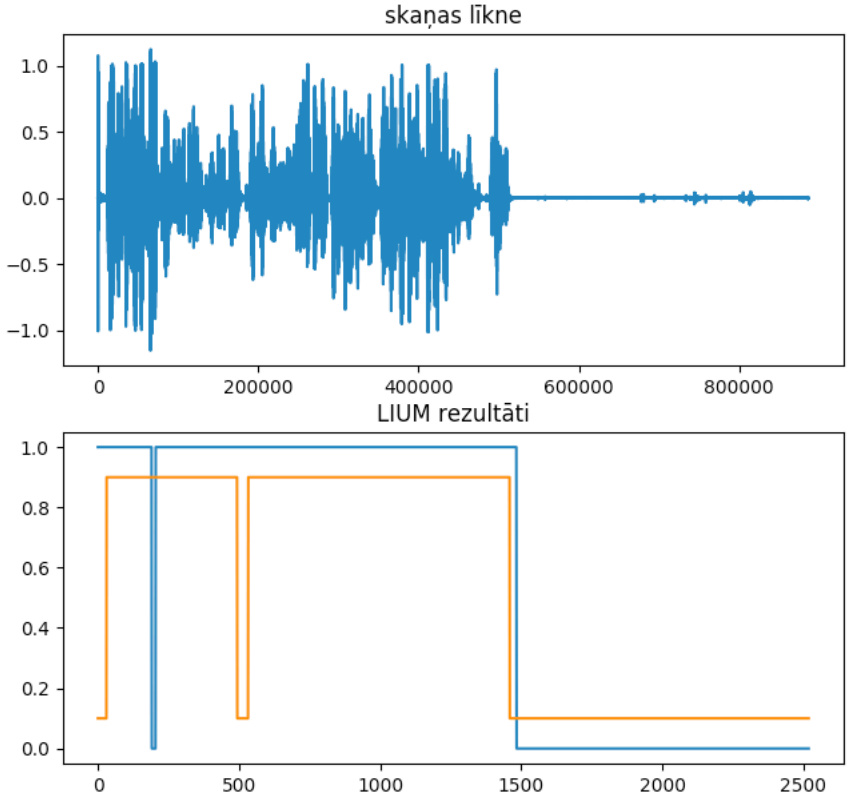
Veicot runas noteikšanas modeļa testēšanu uz reāliem skaņas ierakstiem, tika novērots, ka tīkla atgrieztās pozitīvās vērtības ir ļoti fragmentētas, taču ar tendenci būt blīvākām periodos, kur dzirdama runa. Izejas datiem tika veikta pēcapstrāde ar mērķi padarīt rezultātus homogēnākus. Tika apgrieztas visu klusuma fragmentu vērtības, kas ir īsākas par 50 kadriem

un visas runas fragmentu vērtības, kas īsākas par 20 kadriem. Šāda kombinācija tika atrasta eksperimentālā kārtā un tā deva vislabākos rezultātus.

3.5. Esošo risinājumu novērtēšana

Lai būtu iespējams novērtēt praktiskās daļas ietvaros izstrādāto modeļu precizitāti, tika izveidoti divi reālām dzīves situācijām pietuvināti skaņas ieraksti, kur vienā no tiem dzirdama cilvēka balss, kas mijas ar klusuma fragmentiem, bet otrā dzirdama vairāku cilvēku saruna. Autors veica minēto testa skaņas ierakstu marķēšanu - attiecīgi vienā pozitīvi marķējot klusuma fragmentus, bet otrā runātāja maiņas brīžus.

Lai iegūtu runas noteikšanas un runātāja maiņas brīža noteikšanas precizitātes bāzlīniju, tika veikta izveidoto testa skaņas ierakstu analīze izmantojot LIUM rīku kopu. Lai lietotu LIUM rīku kopu, bija nepieciešams uz testa ierīces uzstādīt *Java* programmēšanas vidi, uz kā balstīts LIUM rīks. Veicamo darbību komandas un parametrus LIUM rīks pieņem caur komandrindas saskarni.



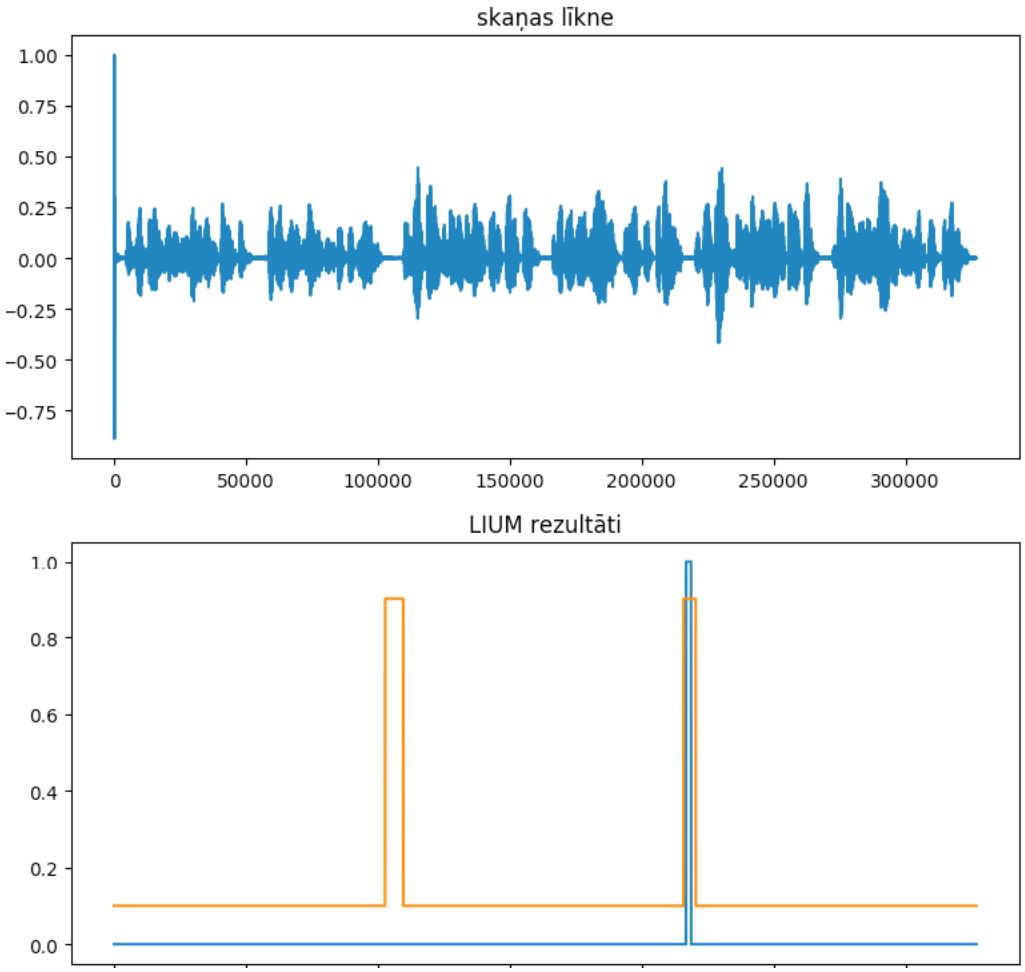
3.6. att. ***LIUM rīku kopas runas noteikšanas rezultāti***

Vispirms, izmantojot LIUM rīku, tika iegūti runas noteikšanas rezultāti, analizējot izveidoto testa ierakstu. Iegūtie rezultāti tika attēloti grafiski (skat. 3.6. att.), lai varētu izdarīt vizuālu veiktspējas novērtējumu. LIUM sniegtie rezultāti (zilā līkne) gandrīz precīzi sakrīt ar testa ieraksta marķējumu (oranžā līkne).

Iegūtie rezultāti tika kvantificēti, izmantojot atvērtā koda bibliotēku *pyannote.metrics* [41], kas salīdzinot iegūtos rezultātus ar datu marķējuma bāzlīniju aprēķināja četrus mērījumus:

- bezkļūdības novērtējums (*accuracy*) – korekti marķēto kadru attiecība pret kopējo kadru skaitu
- kļūda (*error rate*) – nekorekti marķēto kadru attiecība pret kopējo kadru skaitu
- precizitāte (*precision*) – pareizi marķēto kadru attiecība pret visu marķēto kadru skaitu
- atdeve (*recall*) – pareizi marķēto pozitīvo runas kadru attiecība pret pozitīvo visu runas kadru skaitu

Pēc tam tika analizēts otrs testa ieraksts un ar LIUM rīka palīdzību noteikti runātāja maiņas punkti. Iegūtie rezultāti nav tik precīzi, kā runas noteikšanas gadījumā, jo noteikts tikai viens no diviem maiņas punktiem (skat. 3.7. att.). Ar zilo līkni apzīmēti LIUM sniegtie rezultāti, bet ar oranžo atzīmēts testa ieraksta marķējums.



3.7. att. ***LIUM rīku kopas runātāja maiņas noteikšanas rezultāti***

Iegūtie rezultāti arī tika kvantificēti izmantojot *pyannote.metrics* bibliotēkas piedāvātos rīkus. Segmentācijas precizitāte tika vērtēta pēc diviem mērījumiem:

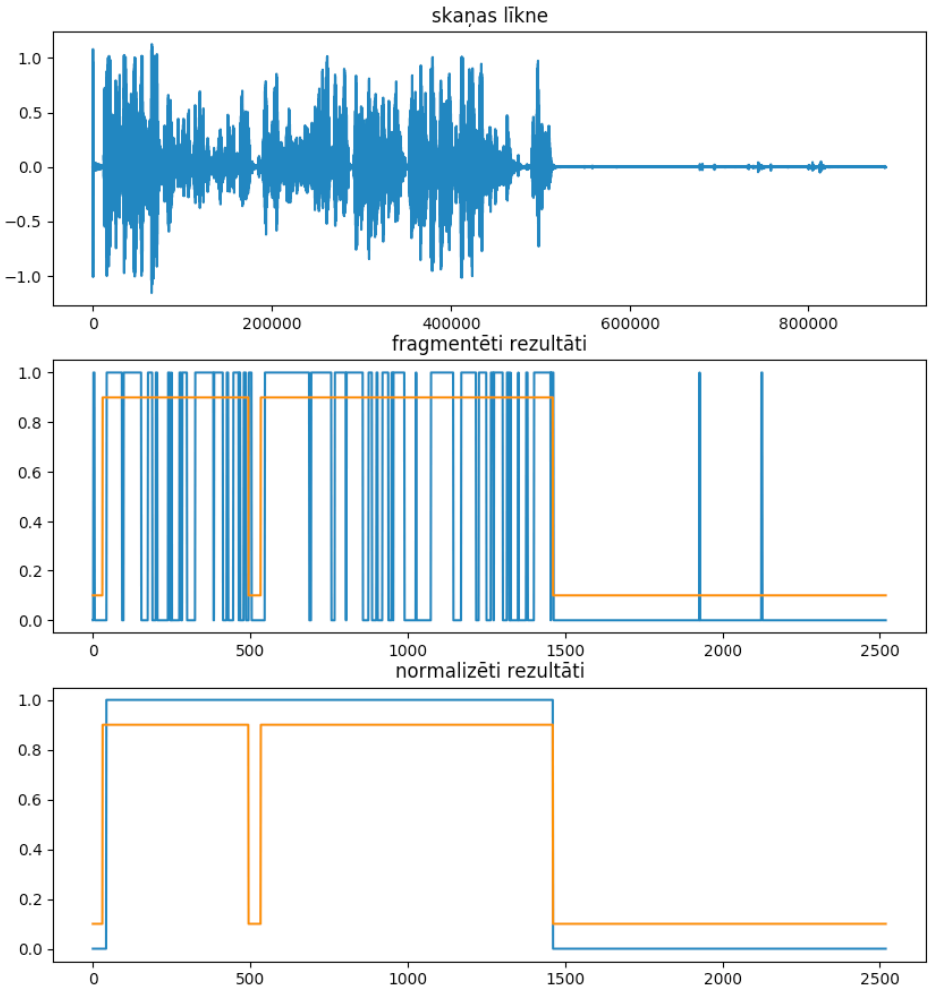
- pārklājums (*coverage*) – attiecība starp testa ierakstā marķētā segmenta garumu un garāko to pārklājošā prognozētā segmenta pārklājošās daļas garumu
- tīrība (*purity*) – procentuāli garākā marķēto segmentu daļa, kas pārklāta ar prognozētu segmentu

Minētie mērījumi tiek noteikti katram segmentam, taču gala mērījumi tiek noteikti, aprēķinot vidējās svērtās vērtības, ņemot vērā katra segmenta garumu.

Kvantificētās LIUM rīka rezultātu vērtības skatīt 3.6. nodaļā, kur tās salīdzinātas ar autora izstrādā risinājuma rādītājiem.

3.6. Izstrādātā risinājuma rezultāti un salīdzinājums

Autora izstrādātais risinājums pēc tā apmācības tika pārbaudīts un novērtēts, izmantojot tās pašas metodes, kas tika lietotas esošo risinājumu novērtēšanai. Autora apmācītais, uz neironu tīkliem bāzētais, runas noteikšanas modelis sasniedza līdzvērtīgus rezultātus kā LIUM rīks.



3.8. att. ***Autora piedāvātā runas noteikšanas risinājuma rezultāti***

Grafiskajā rezultātu attēlojumā (skat. 3.8. att.) redzama analizētā audio līkne (augšā), neapstrādāti modeļa sniegtie analīzes rezultāti (vidū ar zilu līkni) un normalizētie rezultāti (apakšā ar zilu līkni). Ar oranžo līkni apzīmēts testa skaņas ieraksta etalona marķējums.

3.1. tabula ***Runas noteikšanas salīdzinājums***

	Bezkļūdības novērtējums	**Kļūda**	**Precizitāte**	**Atdeve**
LIUM	93%	7,4%	93,8%	98,9%
Autora risinājums	96,1%	3,9%	97,2%	98,8%

Tabulā 3.1. redzams apskatīto runas noteikšanas risinājumu sniegto rezultātu salīdzinājums. No rezultātiem var secināt, ka autora piedāvātā risinājuma kļūda ir gandrīz par 3% mazāka nekā LIUM risinājuma kļūda.

Vienīgais risinājums, ar kuru autoram izdevās iegūt derīgus runātāja maiņas brīža noteikšanas rezultātus, bija LIUM rīku kopa. Autoram neizdevās atkārtot Rikvinga pētījumā [31] sasniegtos rezultātus, izmantojot pētījumā aprakstīto arhitektūru un parametrus, taču rezultātu salīdzināšanas nolūkos tiks izmantoti Rikvinga publikācijā norādītie segmentēšanas rezultāti. Autors eksperimentālā kārtā centās pielāgot runas noteikšanas tīklu arī segmentēšanas problēmas risināšanai, taču praktiskajam darbam atvēlētajā laikā tas neizdevās.

3.2. tabula ***Runātāju segmentēšanas rezultātu salīdzinājums***

	Pārklājums	**Tīrība**
LIUM	66,3%	100%
Rikvinga risinājums[31]	84,4%	94,7%
Autora risinājums	-	-

Tabulā 3.2. redzami novērtēto runātāju segmentēšanas risinājumu sniegtie rezultāti. Redzams, ka Rikvinga piedāvātais neironu tīklu risinājums sasniedzis krietni augstāku pārklājumu nekā LIUM rīks, toties LIUM rīks sasniedzis augstāku tīrību. Tiesa gan augstais tīrības rezultāts varētu būt nejaušība , kas raksturīga tikai šim konkrētajam skaņas ierakstam, jo nav prognozēta neviena runātāju maiņas vieta, kam nebūtu jābūt konkrētajā vietā.

REZULTĀTI

Darba izstrādes gaitā veiktā literatūras apskata ietvaros tika iepazītas un izpētītas izplatītākās runātāju segmentēšanas metodes. Runātāju segmentēšanas problēma tika izdalīta apakšproblēmās – runas noteikšanā, runātāja maiņas brīža noteikšanā un runātāju grupēšanā. Tika apskatīti eksistējošie segmentēšanas risinājumi, kas tika aprakstīti darba pirmajā daļā.

Darba otrajā daļā tika veikts mākslīgo neironu tīklu teorijas apskats, kur uzsvars tika likts uz tēmām, kas varētu būt noderīgas runātāju segmentēšanas problēmas risināšanai, izmantojot neironu tīklus.

Darba trešās – praktiskās daļas ietvaros tika izveidots rekurenta konvolūciju neironu tīkla prototips, kuru lietot runātāju segmentēšanai. Tika izveidota apmācības datu kopa no citām brīvi pieejamām datu kopām. Izveidotā apmācības datu kopa satur pa runātājiem grupētus runas skaņas ierakstus, kā arī fona trokšņu un klusuma skaņas ierakstus, kurus lietot modeļu apmācībai.

Tika apmācīti divi uz izveidotā prototipa bāzēti modeļi – runas noteikšanas modelis un runātāja maiņas noteikšanas modelis. Runas noteikšanas modelis uzrādīja labus rezultātus, taču runātāja maiņas brīža noteikšanas modelis nespēja iemācīties no tā sagaidāmo funkcionalitāti.

Tika iegūtas pirmajā daļā apskatīto esošo segmentēšanas risinājumu rezultātu bāzlīnijas, kas tika salīdzinātas ar izveidotā neironu tīkla prototipa sniegtajiem rezultātiem.

Lielākā daļa izvirzīto darba uzdevumu tika izpildīti pilnībā, taču par daļēji izpildītu var uzskatīt neironu tīklu bāzētā runātāju segmentēšanas prototipa izveidi, jo netika sasniegti pozitīvi rezultāti runātāja maiņas brīža noteikšanā.

SECINĀJUMI

Izstrādājot šo darbu, darba autors nonāca pie secinājuma, ka jebkura neironu tīklu risinājuma panākumu atslēga ir kvalitatīva, marķēta datu kopa, kas pietiekamā apjomā spēj reprezentēt ar neironu tīklu risināmo problēmu. Arī šī darba ietvaros autors saskārās ar situāciju, kurā runātāju segmentēšanas problēmas risināšanai ir nepieciešams pašrocīgi izveidot apmācības datu kopu. Darba autors secināja, ka datu kopas izveide ir sarežģīts uzdevums. Pēc vairākiem mēģinājumiem darba autoram neizdevās apmācīt modeli, kas spēj noteikt runātāja brīžus skaņas ierakstā. Autors pieņem, ka neveiksme saistāma ar kraso klašu nevienlīdzību izveidotajā datu kopā.

Neskatoties uz to, ka datu kopas izveide ir sarežģīts uzdevums, paša neironu tīkla apmācīšana, vismaz triviālā līmenī ir viegli paveicama. Pateicoties tīmeklī brīvi pieejamajiem informācijas resursiem un atvērtā koda neironu tīklu apmācības bibliotēkām, gandrīz jebkurš, kam ir elementāras programmēšanas prasmes, var apmācīt triviālus neironu tīklu modeļus, piemēram, ar roku rakstītu ciparu atpazīšanai.

Pēc esošo runātāju segmentēšanas risinājumu izpētes autors secina, ka pieejamie rīki nav nedz labi dokumentēti, nedz viegli lietojami. Lai novērtētu LIUM rīka veiktspēju autoram nācās pavadīt vairākas dienas eksperimentējot ar ieejas parametriem, kuru lietošana nekur nav dokumentēta. Autors secina arī to, ka šobrīd neeksistē dominējošs un sevi pierādījis, uz neironu tīkliem bāzēts, runātāju segmentēšanas risinājums, kas būtu ērts un pieejams lietošanai arī cilvēkiem bez specifiskām priekšzināšanām. Autors ir pārliecināts, ka brīdī, kad šāds risinājums būs radīts, tas būs uz neironu tīkliem bāzēts un pārspēs visus līdz šim pieejamos risinājumus.

PATEICĪBAS

Darba autors vēlas izteikt pateicību darba vadītājam Artūram Znotiņam par sniegto atbalstu un padomiem darba izstrādes gaitā.

Darba autors izsaka īpašu pateicību LU Datorikas fakultātes projektu studijai "DF LAB" par iespēju izmantot projektu studijas telpas un inventāru, tai skaitā studijas jaudīgo darbstaciju, kas tika izmantota darbā aprakstīto neironu tīklu apmācībā.

Īpaša pateicība arī cīņu biedriem Austrim Cīrulniekam, Elvijam Ernestam Viļķelam un Dagnijai Strīķei par morālo atbalstu grūtos bakalaura darba izstrādes brīžos.

IZMANTOTĀ LITERATŪRA

[1] J. Ramirez, J. M., and J. C., "Voice Activity Detection. Fundamentals and Speech Recognition System Robustness," *Robust Speech Recognit. Underst.*, no. June, pp. 1–23, 2007.

[2] D. O'Shaughnessy, *Speech Processing: A Dynamic and Optimization-Oriented Approach*. 2003.

[3] M. Xu, L. Duan, J. Cai, L. Chia, C. Xu, and Q. Tian, "HMM-based audio keyword generation," *5th Pacific Rim Conf. Multimed.*, pp. 566–574, 2004.

[4] S. Meigner and T. Merlin, "LIUM SPKDIARIZATION: AN OPEN SOURCE TOOLKIT FOR DIARIZATION," *Proc. C. SPUD Work.*, 2010.

[5] X. Anguera, C. Wooters, and J. M. Pardo, "Robust Speaker Diarization for meetings," *Int. Work. Mach. Learn. Multimodal Interact.*, no. October, pp. 346–358, 2006.

[6] "A Basic Introduction To Neural Networks." [Tiešsaite]. Pieejams: http://pages.cs.wisc.edu/~bolo/shipyard/neural/local.html. [Skatīts: 16.05.2018].

[7] S. Agatonovic-Kustrin and R. Beresford, "Basic concepts of artificial neural network (ANN) modeling and its application in pharmaceutical research," *J. Pharm. Biomed. Anal.*, vol. 22, no. 5, pp. 717–727, 2000.

[8] J. Zuters, "Neironu tīklu uzbūve un darbība." [Tiešsaite]. Pieejams: http://home.lu.lv/~janiszu/courses/eanns/annsconstruction.pdf. [Skatīts: 16.05.2018].

[9] A. Schmidt, "A Modular Neural Network Architecture with Additional Generalization Abilities for High Dimensional Input Vectors Acknowledgement," *Architecture*, no. September, 1996.

[10] D. Shiffman, *The Nature of Code*. 2012.

[11] E. Fiesler, "Neural network topologies," *Handb. Neural Comput. E. Fiesler R ...*, pp. 1–17, 1996.

[12] A. Krenker, J. Bešter, and A. Kos, "Introduction to the Artificial Neural Networks," *Eur. J. Gastroenterol. Hepatol.*, vol. 19, no. 12, pp. 1046–1054, 2011.

[13] "Feature extraction using convolution." [Tiešsaite]. Pieejams: http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution. [Skatīts: 16.05.2018].

[14] D. Stutz, "Understanding Convolutional Neural Networks," *Nips 2016*, no. 3, pp. 1–23, 2014.

[15] K. Gregor, K. Kavukcuoglu, A. Szlam, R. Fergus, and Y. LeCun, "Sparse Coding for Feature Learning." [Tiešsaite]. Pieejams:

https://cs.nyu.edu/~yann/research/sparse/index.html. [Skatīts: 16.05.2018].

[16] D. C. Cireşan, U. Meier, J. Masci, L. M. Gambardella, and J. Schmidhuber, "Flexible, high performance convolutional neural networks for image classification," *IJCAI Int. Jt. Conf. Artif. Intell.*, pp. 1237–1242, 2011.

[17] D. Scherer, A. Müller, and S. Behnke, "Evaluation of pooling operations in convolutional architectures for object recognition," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 6354 LNCS, no. PART 3, pp. 92–101, 2010.

[18] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for Simplicity: The All Convolutional Net," pp. 1–14, 2014.

[19] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training Recurrent Neural Networks," no. 2, 2012.

[20] B. Dolhansky, "Artificial Neural Networks: Linear Regression (Part 1)," 2013. [Tiešsaite]. Pieejams: http://briandolhansky.com/blog/artificial-neural-networks-linear-regression-part-1. [Skatīts: 16.05.2018].

[21] D. Britz, "Recurrent Neural Networks Tutorial, Part 1 – Introduction to RNNs," 2015. [Tiešsaite]. Pieejams: http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/. [Skatīts: 16.05.2018].

[22] Y. Bengio, P. Simard, and P. Frasconi, "Learning Long-Term Dependencies with Graident Descent is Difficult," *Saudi Med J*, vol. 33, pp. 3–8, 2012.

[23] S. Hochreiter and J. Urgen Schmidhuber, "Long Short-Term Memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[24] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling," pp. 1–9, 2014.

[25] "Training and Evaluating the CNN MNIST Classifier." [Tiešsaite]. Pieejams: https://www.tensorflow.org/tutorials/layers#training_and_evaluating_the_cnn_mnist_classifier. [Skatīts: 16.05.2018].

[26] J. Garofolo *et al.*, "TIMIT Acoustic-Phonetic Continuous Speech Corpus," *Linguist. Data Consort.*, no. October 2015, p. 1, 1993.

[27] K. Ito, "The LJ Speech Dataset," 2017. [Tiešsaite]. Pieejams: https://keithito.com/LJ-Speech-Dataset/.

[28] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: An ASR corpus based on public domain audio books," *ICASSP, IEEE Int. Conf. Acoust. Speech Signal Process. - Proc.*, vol. 2015–Augus, pp. 5206–5210, 2015.

[29] J. Salamon, C. Jacoby, and J. P. Bello, "A Dataset and Taxonomy for Urban Sound

Research," *Proc. ACM Int. Conf. Multimed. - MM '14*, no. 3, pp. 1041–1044, 2014.

[30] E. Molina, "audio_degrader," 2016. [Tiešsaite]. Pieejams: https://github.com/EliosMolina/audio_degrader. [Skatīts: 16.05.2018].

[31] R. Yin, C. Barras, and F.- Orsay, "Speaker Change Detection in Broadcast TV using Bidirectional Long Short-Term Memory Networks," pp. 3827–3831, 2017.

[32] J. Gibson, M. Van Segbroeck, and S. Narayanan, "Comparing Time-Frequency Representations for Directional Derivative Features - SAIL Publications - Aigaion 2.0."

[33] "Python." [Tiešsaite]. Pieejams: https://www.python.org/. [Skatīts: 16.05.2018].

[34] B. Mcfee *et al.*, "librosa: Audio and Music Signal Analysis in Python," *PROC. 14th PYTHON Sci. CONF*, no. Scipy, pp. 1–7, 2015.

[35] O. Travis E, "A guide to NumPy," *Trelgol Publ.*, 2006.

[36] "Python object serialization." [Tiešsaite]. Pieejams: https://docs.python.org/3/library/pickle.html. [Skatīts: 16.05.2018].

[37] M. Huzaifah, "Comparison of Time-Frequency Representations for Environmental Sound Classification using Convolutional Neural Networks," pp. 1–5, 2017.

[38] "Tensorflow." [Tiešsaite]. Pieejams: https://www.tensorflow.org/. [Skatīts: 16.05.2018].

[39] F. Chollet, "Keras," 2015. .

[40] J. Brownlee, "How to Diagnose Overfitting and Underfitting of LSTM Models," 2017. .

[41] B. Herv'e, "pyannote.metrics: a toolkit for reproducible evaluation, diagnostic, and error analysis of speaker diarization systems," *Interspeech 2017, 18th Annual Conference of the International Speech Communication Association*, 2017. [Tiešsaite]. Pieejams: https://pyannote.github.io/pyannote-metrics/. [Skatīts: 16.05.2018].

PIELIKUMI

1. pielikums **Izstrādātā neironu tīkla prototipa definīcija**

```
def model(input_shape, optimizer):
    print('Building CONV LSTM RNN model ...')

    recurrent_layer = CuDNNGRU if gpu else GRU

    model = Sequential()

    model.add(ZeroPadding1D(1,
                            input_shape=input_shape))

    model.add(Conv1D(128, 3))

    model.add(LeakyReLU())

    model.add(BatchNormalization())

    model.add(Dropout(0.4))

    model.add(ZeroPadding1D(1))

    model.add(Conv1D(64, 3))

    model.add(LeakyReLU())

    model.add(BatchNormalization())

    model.add(Dropout(0.4))

    model.add(recurrent_layer(64, return_sequences=True))

    model.add(LeakyReLU())

    model.add(Dropout(0.4))

    model.add(BatchNormalization())

    model.add(recurrent_layer(32, return_sequences=True))

    model.add(LeakyReLU())

    model.add(Dropout(0.4))

    model.add(TimeDistributed(Dense(10)))

    model.add(LeakyReLU())

    model.add(BatchNormalization())

    model.add(Dropout(0.4))

    model.add(TimeDistributed(Dense(1, activation="sigmoid")))

    print("Compiling ...")
    model.compile(loss='binary_crossentropy', optimizer=optimizer,
metrics=['accuracy'])
    model.summary()

    return model
```

Bakalaura darbs “Runātāju segmentēšana skaņas ierakstā, izmantojot neironu tīklus” izstrādāts Latvijas Universitātes Datorikas fakultātē.

Ar savu parakstu apliecinu, ka pētījums veikts patstāvīgi, izmantoti tikai tajā norādītie informācijas avoti un iesniegtā darba elektroniskā kopija atbilst izdrukai.

Autors: _____ Dāvis Mednis

Rekomendēju/nerekomendēju darbu aizstāvēšanai
Vadītājs: M.dat. Artūrs Znotiņš _____ 28.05.2018.

Recenzents: docents, Dr. dat. Kārlis Freivalds

Darbs iesniegts Datorikas fakultātē 28.05.2018.
Dekāna pilnvarotā persona:
vecākā metodiķe Ārija Sproģe _____

Darbs aizstāvēts bakalaura gala pārbaudījuma komisijas sēdē
__.06.2018. prot. nr. _____
Komisijas sekretārs(-e): _____ _____