

Danielius VISOCKAS

GARSO ATKŪRIMO TRIUKŠMINGOJE APLINKOJE TYRIMAS
INVESTIGATION OF SOUND RECONSTRUCTION IN NOISY
ENVIRONMENT

Magistro baigiamasis darbas

Informacinių elektroninių sistemų studijų programa, valstybinis kodas 6211BX015

Dirbtinio intelekto sistemų specializacija

Informatikos inžinerijos studijų kryptis

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS
ELEKTRONIKOS FAKULTETAS
ELEKTRONINIŲ SISTEMŲ KATEDRA

TVIRTINU
Katedros vedėjas

(parašas)

Prof. dr. D. Navakauskas

2021 m. _____ mėn. ____ d.

Danielius VISOCKAS

GARSO ATKŪRIMO TRIUKŠMINGOJE APLINKOJE TYRIMAS
INVESTIGATION OF SOUND RECONSTRUCTION IN NOISY
ENVIRONMENT

Magistro baigiamasis darbas

Informacinių elektroninių sistemų studijų programa, valstybinis kodas 6211BX015

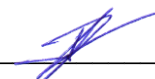
Dirbtinio intelekto sistemų specializacija

Informatikos inžinerijos studijų kryptis

Vadovas

doc. dr. Raimond Laptik

(pedag. vardas, moksl. laipsnis, vardas, pavardė)


(parašas) (data)

Lietuvių kalbos konsultantas

lekt. dr. Anželika Gaidienė

(pedag. vardas, moksl. laipsnis, vardas, pavardė)

(parašas) (data)

Vilniaus Gedimino technikos universitetas Elektronikos fakultetas Elektroninių sistemų katedra	ISBN Egz. sk. Data-.....-.....
Antrosios pakopos studijų Informacinių elektroninių sistemų programos magistro baigiamasis darbas	
Pavadinimas	Garso atkūrimo triukšmingoje aplinkoje tyrimas
Autorius	Danielius Visockas
Vadovas	Raimond Laptik
Kalba: lietuvių	
Anotacija Garso atkūrimo triukšmingoje aplinkoje tyrimas. Magistro baigiamasis darbas informatikos inžinerijos laipsniui. Vilniaus Gedimino technikos universitetas. Vilnius, 2021, 62 p., 20 iliustr., 9 lent., 60 bibl., 2 priedai. Naujausi garso atkūrimo triukšmingoje aplinkoje algoritmai veikia vis tiksliau, tačiau dauguma jų labai skirtingi. Baigiamojo darbo tikslas – ištirti dažniausiai pasikartojančių modelių elementų įtaką garso atkūrimo triukšmingoje aplinkoje problemai spręsti ir įgyvendinti algoritmą leidžiantį šią problemą spręsti. Tikslui pasiekti, individualiai keisti literatūroje minimi parametrai ir vertinama jų pakitimo įtaka, remiantis objektyviais kriterijais. Atliekami eksperimentai ir pasiūlomas naujas, algoritmas, gebantis spręsti garso atkūrimo triukšmingoje aplinkoje problemą.	
Prasminiai žodžiai: garso atkūrimas, triukšmo šalinimas, kalbos gryninimas	

Vilnius Gediminas Technical University
Faculty of Electronics
Department of Electronic Systems

ISBN	ISSN
Copies No.	
Date-.....-.....	

Master Degree Studies **Information Electronics Systems** study programme Master Graduation Thesis

Title	Investigation of Sound Reconstruction in Noisy Environment
-------	--

Author **Danielius Visockas**

Academic supervisor	Raimond Laptik
---------------------	-----------------------

Thesis language: Lithuanian

Annotation

Sound reconstruction algorithms are getting faster and more accurate, however they differ a lot. The main goal of this thesis - to research the most common elements of currently used models. It is done by isolating the parameters of the model and changing them individually, while evaluating how the parameter change impacts objective speech quality metrics. Experiments are conducted and new sound reconstruction model is proposed.

Master thesis consists of 62 pages, 20 figures, 9 tables, 60 references, 2 annexes

Keywords: sound reconstruction, speech enhancement, noise removal

(Baigiamojo darbo sąžiningumo deklaracijos forma)

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

Danielius Visockas, 20134386

(Studento vardas ir pavardė, studento pažymėjimo Nr.)

Elektronikos fakultetas

(Fakultetas)

Informacinės elektroninės sistemos, DISfm-19

(Studijų programa, akademinė grupė)

BAIGIAMOJO DARBO (PROJEKTO)

SĄŽININGUMO DEKLARACIJA

2021 m. gegužės 10 d.

Patvirtinu, kad mano baigiamasis darbas tema „Garso atkūrimo triukšmingoje aplinkoje tyrimas“ patvirtintas 2019 m. lapkričio 08 d. dekanų potvarkiu Nr. 233el, yra savarankiškai parašytas. Šiame darbe pateikta medžiaga nėra plagijuota. Tiesiogiai ar netiesiogiai panaudotos kitų šaltinių citatos pažymėtos literatūros nuorodose.

Mano darbo vadovas docentas daktaras Raimond Laptik.

Kitų asmenų indėlio į parengtą baigiamąjį darbą nėra. Jokių įstatymų nenumatytų piniginių sumų už šį darbą niekam nesu mokėjęs (-usi).



(Parašas)

Danielius Visockas
(Vardas ir pavardė)

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS
ELEKTRONIKOS FAKULTETAS
ELEKTRONINIŲ SISTEMŲ KATEDRA

Technologijos mokslų sritis

Elektros ir elektronikos inžinerijos mokslo kryptis

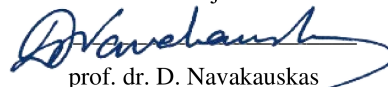
Informatikos inžinerijos studijų kryptis

Informacinių elektroninių sistemų studijų programa, valst. kodas 6211BX015

Dirbtinio intelekto sistemų specializacija

TVIRTINU

Katedros vedėjas



prof. dr. D. Navakauskas

2021 m. 04 mėn. 30 d.

MAGISTRO BAIGIAMOJO DARBO

UŽDUOTIS

Studentui Danielius VISOCKAS

Baigiamojo darbo tema: Garso atkūrimo triukšmingoje aplinkoje tyrimas

Investigation of Sound Reconstruction in Noisy Environment

Patvirtinta 2019 m. 11 mėn. 08 d. dekanų potvarkiu Nr. 233el

Baigiamojo darbo užbaigimo terminas 2021 m. 05 mėn. 20 d.

Darbo tikslas – Ištirti garso atkūrimo algoritmus.

Darbo uždaviniai

1. Atlikti garso atkūrimo algoritmų analitinę apžvalgą.
2. Įgyvendinti garso atkūrimo algoritmą.
3. Ištirti garso atkūrimo algoritmą.

Aiškinamojo rašto turinys

Įvadas

Darbo aktualumas ir tikslas

Darbo uždaviniai

Naudoti tyrimo ir analizės metodai

Darbo naujumas ir praktinė nauda

Darbo struktūra

1. Analitinė literatūros apžvalga
2. Garso atkūrimo algoritmo įgyvendinimas
3. Garso atkūrimo algoritmo tyrimas
4. Garso atkūrimo algoritmo tyrimo rezultatai Apibendrinimas. Išvados

Literatūra

Priedai

A priedas. CD su magistro baigiamojo darbo aiškinamuoju raštu

B priedas. Pranešimo 24-oje Lietuvos jaunųjų mokslininkų konferencijoje medžiaga

Vadovas

(parašas)

doc.dr. Raimond Laptik

(pedag. vardas, moksl. laipsnis, vardas, pavardė)

Užduotį gavau

(parašas)

202_ m. _____ mėn. ____ d.

TURINYS

Įvadas	7
Darbo aktualumas ir tikslas	7
Darbo uždaviniai	7
Taikyti tyrimo ir analizės metodai	7
Darbo naujumas ir praktinė nauda	7
Darbo struktūra	8
1. Analitinė literatūros apžvalga	9
1.1. Duomenys ir jų atvaizdavimas	9
1.1.1. Dažnių ašis	9
1.1.2. Melo skalė	11
1.1.3. Poskyrio apibendrinimas	12
1.2. Statistiniai algoritmai	12
1.2.1. Slenkančio vidurkio filtrai	12
1.2.2. Spektro atimties metodas	13
1.2.3. Statistiškai optimalūs metodai	14
1.3. Neuroniniai tinklai	15
1.3.1. Daugiasluoksniai perceptronai	16
1.3.2. Rekurentiniai neuroniniai tinklai	16
1.3.3. Sąsukų neuroniniai tinklai	18
1.3.4. Poskyrio apibendrinimas	21
1.4. Trumpas skyriaus apibendrinimas	21
2. Garso atkūrimo algoritmo įgyvendinimas	22
2.1. Algoritmo pasirinkimas	22
2.2. Duomenys ir jų paruošimas	22
2.3. Modelio architektūra	23
2.3.1. Įvesties duomenys	23
2.3.2. Enkoderio filtrai	24
2.3.3. Normalizacija	25
2.3.4. Aktyvavimo funkcija	26
2.3.5. Trumpa enkoderio apžvalga	27
2.3.6. Glaudinto signalo filtravimas dėmesio moduliais	28
2.3.7. Dekoderio filtrai	29
2.4. Modelio optimizavimas	31
2.4.1. Klaidos funkcija	32
2.4.2. Mokymo algoritmas	33
2.5. Skyriaus apibendrinimas	34
3. Garso atkūrimo algoritmo tyrimas ir rezultatai	35
3.1. Duomenų rinkinys	35
3.2. Vertinimo kriterijai	35
3.3. Tirti parametrai	36
3.3.1. Modelio sluoksnių kiekis	36
3.3.2. Enkoderio filtrai	38
3.3.3. Savidėmesingumo moduliai	39
3.3.4. Dekoderio parametrai	40
3.3.5. Normalizacijos	41
3.3.6. Aktyvavimo funkcijos	42
3.3.7. Klaidos funkcijos	43
3.3.8. Mokymo algoritmai	44
3.3.9. Optimalūs algoritmo parametrai	45

3.4. Rezultatų palyginimas	46
3.5. Trumpas skyriaus apibendrinimas.....	46
Apibendrinimas	47
Literatūra	49
Priedai	54
A priedas. Pasiūlyto algoritmo programos išeities kodas.....	54
B priedas. Pranešimo 24-oje Lietuvos jaunųjų mokslininkų konferencijoje medžiaga	57

SANTRUMPOS

ASR (angl. *Automatic speech recognition*) – kalbos atpažinimas

CNN (angl. *Convolutional neural network*) – sąsūkos neuroninis tinklas

FFT (angl. *Fast Fourier transform*) – greitoji Furjė transformacija

GPI – grafinis procesorinis įrenginys

GRU (angl. *Gated recurrent unit*) – užtvertas rekurentinis vienetas

LSTM (angl. *Long-short term memory*) – ilgo ir trumpo laikotarpio atminties ląstelė

MMSE (angl. *Minimum mean squared error*) – mažiausia vidutinė kvadratinė klaida

MLP (angl. *Multi-layer perceptron*) – daugiasluoksnis perceptronas

MSE (angl. *Mean squared error*) – vidutinė kvadratinė klaida

RNN (angl. *Recurrent neural network*) – rekurentinis neuroninis tinklas

SA (angl. *Self-attention*) – dėmesingumo modulis

SNR (angl. *Signal to noise ratio*) – signalo ir triukšmo santykis

STFT (angl. *Short time fourier transform*) – trumpalaikė Furjė transformacija

VAD (angl. *Voice activity detection*) – balso aptikimas

ĮVADAS

Darbo aktualumas ir tikslas

Balsu ir kitomis garso formomis grįsti sprendimai plačiai taikomi šiais laikais. Kalbėjimas telefonu, klausos aparatai, balso atpažinimas – tik dalis naudojamų garso technologijų. Tačiau daugelis jų pradeda prastai veikti atsiradus triukšmams. Garso atkūrimo metodų tikslas – pašalinti triukšmus, kad minėtos technologijos galėtų apdoroti garso signalą, tarsi jokie triukšmai neegzistuos.

Triukšmų šalinimo uždaviniams spręsti reikia, kad garso atkūrimo algoritmai veiktų greitai ir prisitaikytų prie įvairių garso bei šalinamo triukšmo ypatybių. Šio darbo tikslas – ištirti garso atkūrimo triukšmingoje aplinkoje algoritmų veikimo principus, įgyvendinti ir eksperimentiškai ištirti galimus algoritmų įgyvendinimo bei optimizavimo būdus.

Darbo uždaviniai

Baigiamojo darbo tikslui pasiekti keliama trys uždaviniai:

1. Atlikti garso atkūrimo triukšmingoje aplinkoje algoritmų analizę.
2. Ištirtų algoritmų pagrindu pasiūlyti ir įgyvendinti naują algoritmą garso atkūrimo triukšmingoje aplinkoje problemai spręsti.
3. Atlikti įgyvendinto algoritmo patikrą ir palyginti su kitais tiriamais algoritmais. Pateikti tyrimo rezultatus.

Taikyti tyrimo ir analizės metodai

Tyrimas grindžiamas esamų algoritmų analize ir eksperimentais. Eksperimentai remiasi analitine literatūros apžvalga ir ją studijuojant parametrų bei sprendimų paieškos erdve. Naudojamas viešai prieinamas duomenų rinkinys, skirtas modeliuoti triukšmingą ir švarų balsą. Eksperimentiniams tyrimams atlikti naudojamas kompiuteris su dvylikos branduolių „AMD Ryzen Threadripper 1920x“ 4,0 GHz procesoriumi, 64 GB RAM operatyviosios atminties ir „ASUS GeForce GT1080Ti“ grafiniu procesoriniu įrenginiu, turinčiu 11 GB operatyviosios atminties.

Tirto algoritmo programinis kodas įgyvendintas *Python* programavimo kalba, *PyTorch* ir *fast.ai* mašininio mokymo bibliotekomis.

Darbo naujumas ir praktinė nauda

Garso atkūrimo triukšmingoje aplinkoje algoritmai – plačiai ir sparčiai tiriama sritis. Praėjusiame dešimtmetyje išpopuliarėjus giliuoju mokymu grįstiems mašininio mokymo metodams pasiūlyta daug įvairių algoritmų, gebančių atkurti garsą triukšmingoje aplinkoje. Dauguma siūlomų

algoritmų skiriasi. Nėra žinoma, kurios naudojamų algoritmų dalys naudingos, o kurios tobulintinos.

Ištyrus algoritmų veikimo sudedamąsias dalis, galima gauti aiškią praktinę naudą. Algoritmus optimizuoti mažos spartos įrenginiams (klausos aparatams), pritaikyti mažą duomenų kiekį turinčioms kalboms. Taip pat įsigilinus į esmines algoritmo veikimo vietas galima pagerinti kalbos atpažinimo sistemų tikslumą ar įdiegti realiu laiku veikiančias, triukšmingo garso atkūrimo sistemas skambučių centruose, kur būna triukšminga, kai dirba daug žmonių.

Darbo struktūra

Darbo pradžioje atliekama išsami taikomų algoritmų ir jų dalių analizė, pateikiamas jų palyginimas. Apžvelgiami klasikiniai, statistiniais metodais grįsti ir šiuolaikiški, dirbtinių neuronų tinklais grįsti algoritmai. Atliekant eksperimentinius tyrimus, palyginamos algoritmų dalys ir parenkamos tolimesnei analizei. Analizė atliekama apžvelgiant duomenų rinkinį bei jo apdorojimo metodus, tinkamus garso atkūrimo problemai spręsti. Atlikus apžvalgą, kuriamas ir su kitais lyginamas algoritmas garso atkūrimo triukšmingoje aplinkoje problemai spręsti.

Algoritmams palyginti pasirenkami analitinėje apžvalgoje minimi sprendimai ir aptariami jų lyginimo būdai. Kadangi šio darbo tikslas – ištirti garso atkūrimo algoritmus ir jų dedamąsias dalis – pasirenkami algoritmai, kurių programinis kodas yra viešai prieinamas. Viešas algoritmų prieinamumas leidžia tyrinėti algoritmų veikimą naudojant identiškus duomenis.

Darbo pabaigoje pateikiami ir apibendrinami pasiekti rezultatai, pasiūlomos tolimesnių tyrimų kryptys.

1. ANALITINĖ LITERATŪROS APŽVALGA

Šiame skyriuje aptariami ankstesniuose tyrimuose ir sistemose taikyti garso bei kitų signalų atkūrimo metodai. Poskyrių eiliškumas – funkcinis. Visų pirma apžvelgiami duomenys ir taikomos transformacijos, vėliau apžvelgiami įvairūs duomenų modeliavimo metodai garso atkūrimo problemai spręsti. Kiekvienas poskyris apžvelgiamas chronologiškai – aptariami istoriškai taikyti metodai ir pagaliau prieinama prie šiuolaikinių metodų.

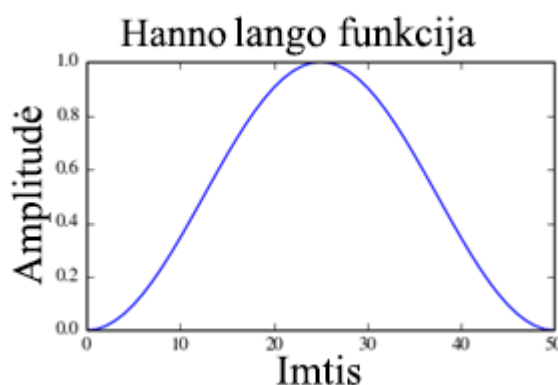
1.1. Duomenys ir jų atvaizdavimas

1.1.1. Dažnių ašis

Dauguma ankstesnių garso atkūrimo algoritmų modeliuojant naudoja dažnių ašį (spektrogramas), gaunamą taikant STFT algoritmą [1]. STFT algoritmo esmė – išskaidyti laiko ašyje esantį signalą į daugelį periodinių tam tikro dažnio signalų.

$$STFT\{x[n]\}(m, \omega) = X(m, \omega) = \sum_{n=-\infty}^{\infty} x[n]w[n-m]e^{-j\omega n} \quad (1.1)$$

Be STFT funkcijos apskaičiavimo, signalas turi būti suskaidomas į segmentus, vadinamus langais. Lango esmė – skaičiuoti STFT transformaciją tik mažai daliai viso signalo, taip padidinant galutinio spektro rezoliuciją. Lango ilgis – algoritmo įgyvendintojo pasirenkamas parametras, dažniausiai svyruojantis nuo 16 iki 64 ms. Kadangi garso signalas dažniausiai būna diskretaus laiko – laikas kvantavimo metu tampa tiesiog skaičių masyvu, tikrasis lango dydis priklauso nuo diskretizavimo dažnio. Diskretizuojant signalą 16 kHz dažniu – lango dydžio apatinė lango ilgio riba bus 256 imtys, aukštesnioji – 1024. Taip pat signalo skaidymas į segmentus sukelia artefaktų – transformacijos sukeltą triukšmą, kurio nėra pradiniam signalui. Tam, kad būtų išvengta šių artefaktų – kiekvienas pasirinktas segmentas transformuojamas taip, kad signalo pradžia ir pabaiga artėtų prie nulio.

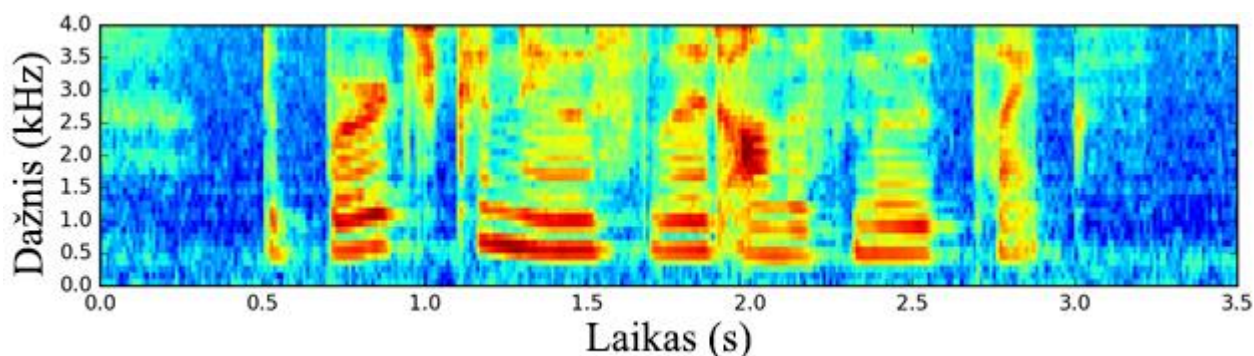


1.1 pav. Hanno lango funkcija

Paskirsčius signalą į segmentus ir juos transformavus lango funkcija, ja slenkama tolyn per pusę lango ilgio imčių. Atlikus šiuos veiksmus gaunami du signalai – dažnių, esančių signale amplitudės ir tų dažnių fazės. Tam, kad abu šie signalai būtų sujungti – jie pakeliami kvadratu. Taip gaunama spektrograma.

$$\text{spektrograma}\{x(t)\}(\tau, \omega) = |X(\tau, \omega)|^2 \quad (1.2)$$

Spektrogramomis grįstų modelių tikslas – nuspėti matricą, kurią sudauginus su triukšmingo garso spektrograma būtų gaunamas švarus signalas [2], [3]. Taip pat problemą galima formuluoti priešingai – tokiu atveju algoritmo tikslas tampa nuspėti triukšmo signalą. Jį atėmus iš triukšmingo garso signalo gaunamas švarus signalas [4], [5].



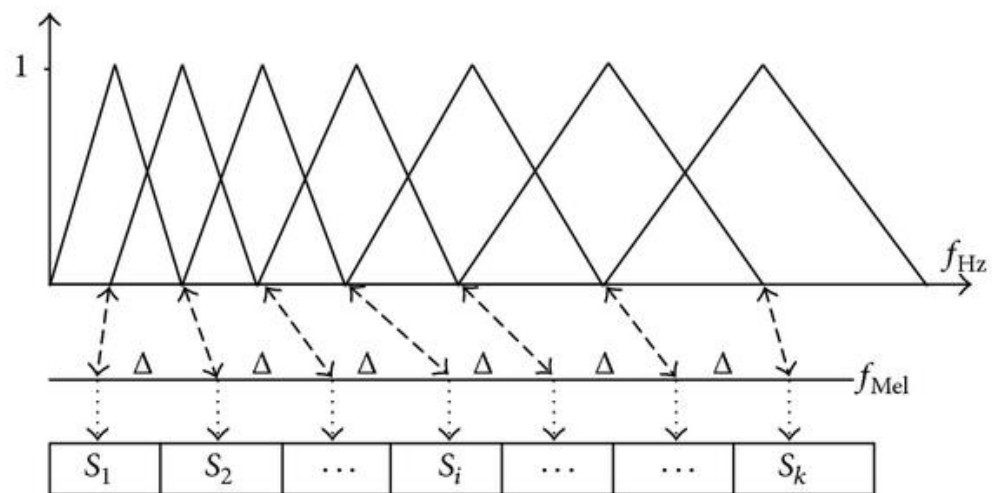
1.2 pav. Spektrograma

Nors dažnių sritimi grįsti modeliai sudaro didelę dalį tyrimų, jie turi keletą trūkumų. Pirmiausia, STFT nebūtinai tinka garso signalams ir jų atkūrimui. Antra, kadangi naudojantis dažnių ašimi dažnių ir fazės informaciją dažnai sujungiama į bendrą atvaizdavimą (paprastai taip daroma tam, siekiant išvengti skaičiavimų kompleksinių skaičių erdvėje), tinkamas fazės atkūrimas tampa labai sunkia problema. Sunku atkurti fazę net ir žinant visą dažnių informaciją [6]–[8]. Galiausiai, kad būtų galima tiksliai atkurti švarų garso signalą, reikalinga aukštos rezoliucijos dažnių informacija. Kad būtų galima tai atlikti taikant STFT, reikia ilgo laiko intervalo, o tai stipriai apriboja algoritmo veikimo realiuoju laiku ar trumpo vėlinimo galimybes. Šis dažnių srities taikymas apriboja aukštos kokybės garso atkūrimą telekomunikacijose ar klausos aparatuose [9], [10].

Tačiau spektrogramomis plačiai naudojamosi įvertinant algoritmų veikimą vizualiai. Jose lengviau matyti, kurie dažnių ruožai yra filtruojami, kurie ne, ir lengviau pastebimi algoritmo sukurti artefaktai

1.1.2. Melo skalė

Nagrinėjant spektrogramų pritaikymą garso atkūrimo, atpažinimo ir kitoms problemoms spręsti dažnai naudojamosi Melo skalė. Tai žmogaus klausos aparato įkvėpta skalė, kurios esmė – padidinti žemų dažnių rezoliuciją sumažinant aukštesnių dažnių rezoliuciją. Melo skalė pritaikoma STFT naudojantis trikampaiais filtrais, jie paskirstomi netiesiškai. Šis netiesiškumas modeliuoja žmogaus dažnių supratimą [11]. Galiausiai gautas spektras suspaudžiamas taikant logaritminę funkciją, kuri modeliuoja mūsų netiesinį garsumo supratimą [12]. Šie mūsų klausos sistemos įkvėpti metodai veikia geriau negu grynas STFT signalas ir yra mažiau jautrūs triukšmui [13].



1.3 pav. Melo skalės trikampaiai filtrai

Šis filtravimo tipas žemesniems dažniams suteikia aukštesnę rezoliuciją negu aukštesniems. Tai puikiai tinka žmogaus kalbai ar kitiems žemų dažnių signalams. Aukštuose dažniuose nesant didelio informacijos kiekio informacijos glaudinimas juose tiesiog sumažina reikiamą skaičiavimų kiekį. Tačiau jeigu norimas modeliuoti garsas egzistuoja ir aukštesniuose dažniuose, Melo skalės pritaikymas gali sunaikinti nenorimą prarasti informaciją.

Nors Melo skalė labai dažnai naudojamosi, jos dažnių ruožų pakeitimas gali nežymiai pagerinti algoritmą. Keletas skirtingų dažnių ruožų, bet tokius pat filtrus naudojančių spektro glaudinimo metodų – Bark skalė [14] ir Opus skalė [15]. Tai tik keli mėginimai pakeisti skalę, tačiau neaišku, kaip nustatyti, kokie tikslūs dažnių ruožai galėtų padėti efektyviau spręsti problemas.

1.1.3. Poskyrio apibendrinimas

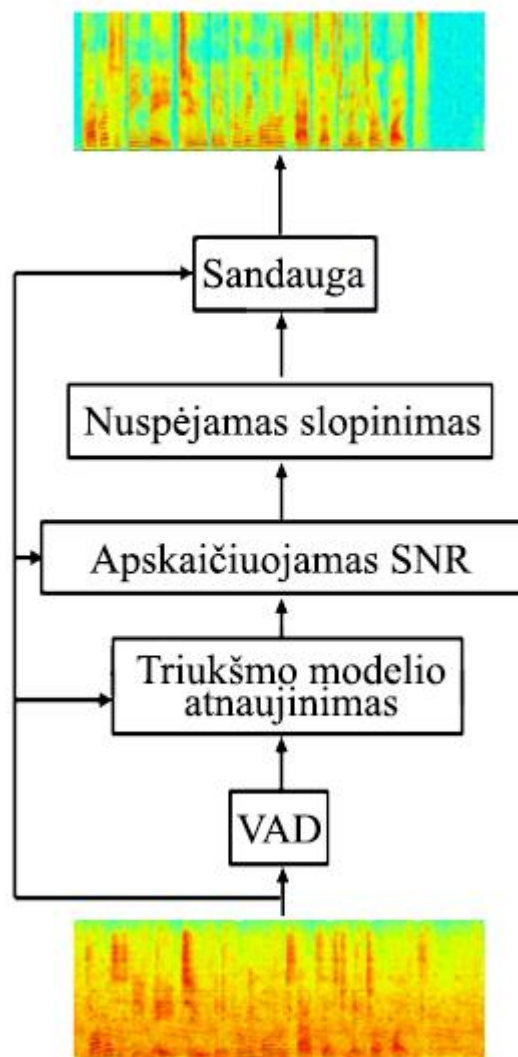
Apžvelgti įprastos ir Melo spektrogramos, jų ypatybės ir pagrindiniai parametrai. Taip pat aprašyti būdai, kaip jas taikant sprendžiamos signalų atkūrimo triukšmingoje aplinkoje problemos.

1.2. Statistiniai algoritmai

Statistiniai modeliai nereikalauja išankstinių duomenų apie modeliuojamus garsus bei triukšmus. Reikalavimas būtinai pateikti duomenis gali pasirodyti geras sprendimas, tačiau jo kaina – modelis visuomet bus lėtesnis, nes kaskart gavus naują duomenų imtį – reikės skaičiuoti ir nuspėti, kuris triukšmas ar garsas neturėtų būti slopinamas. Taip pat statistiniai metodai sunkiai gali atskirti kelis panašius garsus. Pavyzdžiui, šnekant dviem asmenims šie metodai sunkiai nustato, kuris asmuo turėtų būti prislopintas. Todėl šis algoritmų tipas, priklausomai nuo pasirinktų parametrų, prislopina arba palieka abiejų kalbėtojų garsą.

1.2.1. Slenkančio vidurkio filtrai

1.4 pav. atskleidžiami pagrindiniai statistiniais metodais grįstos triukšmo slopinimo sistemos elementai. Balso (ar kito norimo garso) signalo aptikimo detektorius (VAD) apskaičiuoja tikimybę, kad garsas egzistuoja kiekvienoje dažnių juostoje. Triukšmo modelio nuspėjimo vienetas nuolat atnauja informaciją apie triukšmo spektrą, naudodamas rekursinį slenkančio lango filtrą:



1.4 pav. Statistiniais metodais grįstos balso atkūrimo sistemos veikimo schema

1.2.2. Spektro atimties metodas

Spektro atimties metodai – statistinių metodų dalis pradėta tyrinėti aštuntajame dvidešimto amžiaus dešimtmetyje. Jų tikslas – nuspėti matricą, kurią sudauginus su triukšminga garso spektrograma, gautas signalas būtų kuo panašesnis į švarų [16].

$$\lambda_k(m) = 1 - \alpha_k(m)\lambda_k(m-1) + \alpha_k(m)|X_k(m)|^2, \quad (1.3)$$

čia $\lambda_k(m)$ – triukšmo pasiskirstymas laiko lange m , dažnyje k . $X_k(m)$ – trumpalaikė, triukšmingo signalo, Furjė transformacija, $\alpha_k(m)$ – triukšmo buvimo dabartiniame lange tikimybė, apskaičiuojama taip:

$$\alpha_k(m) = \alpha_0(1 - p_k(m))(1 - p_{total}(m)), \quad (1.4)$$

čia $p_k(m)$ – tikimybė kad dažnio ruože esantis signalas yra kalba (šią tikimybę grąžina VAD modulis), $p_{total}(m)$ – bendra tikimybė, kad lange m egzistuoja kalbos signalas.

Tada triukšmo nuokrypis (1.4) taikomas apskaičiuoti dviem reikšmėms – *posterior SNR* ($\gamma_k(m)$) ir *prior SNR* ($\xi_k(m)$):

$$\gamma_k(m) \triangleq \frac{|X_k(m)|^2}{\lambda_k(m)}, \quad (1.5)$$

$$\xi_k(m) \triangleq \frac{|\hat{S}_k(m)|^2}{\lambda_k(m)} \approx \beta \xi_k(m-1) + (1-\beta) \max\{0, \gamma_k(m) - 1\}, \quad (1.6)$$

čia S_k yra švarios kalbos triukšmas, β – rekursyviai apskaičiuojamas SNR nuspėjimo parametras. Gaunama triukšmo filtravimo formulė:

$$g_k(m) = g(\xi_k(m), \gamma_k(m)). \quad (1.7)$$

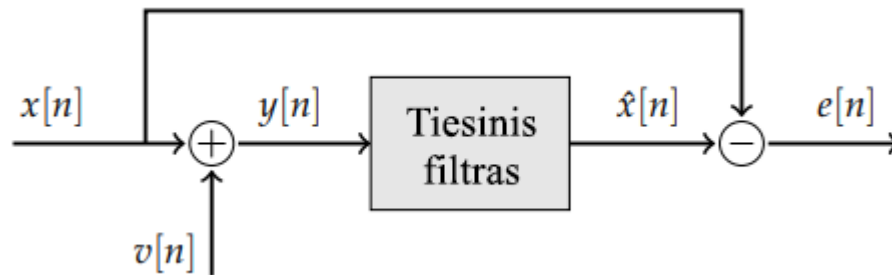
Literatūroje minimos skirtingos siūlomos triukšmo filtravimo funkcijos $g(\cdot)$, jos visos paremtos skirtingomis prielaidomis apie triukšmo signalų spektrus ar efektyvumą jas skaičiuojant [16], [17].

Nors garso spektro atimtimi grįsti garso atkūrimo triukšmingoje aplinkoje metodai efektyviai geba pašalinti signale esantį triukšmą, jie turi keletą trūkumų. Visų pirma – triukšmo lygio β nuspėjimas: tai sunki problema esant nestacionariam, laike besikeičiančiam triukšmui. Dėl sunkumų nuspėjant triukšmo lygį – spektro lygis algoritmui gali pasirodyti neigiamas. Tačiau spektre negali egzistuoti neigiamos vertės. Šiai problemai spręsti buvo pasiūlyta daug sprendimų, tačiau paprasčiausias ir dažniausias – visas neigiamas vertes prilyginti nuliui [18]. Kiti galimi variantai – neigiamoms vertėms priskirti gretimų dažnių ruožų vertes, jeigu jos yra teigiamos. Tačiau iki šiol nėra pasiūlyta, kaip šią problemą išspręsti iki galo. Daugumą spektro atimtimi grįstų garso atkūrimo metodų apsunkina pridėtinis, originaliame signale net neegzistuojantis triukšmas, dažnai vadinamas muzikiniu.

1.2.3. Statistiškai optimalūs metodai

Nors spektro atimtimi grįsti modeliai yra efektyvūs, jie grįsti įvairiomis euristicomis, kurios nėra matematiškai optimalios. Jeigu garso atkūrimo triukšmingoje aplinkoje problema performuluojama kaip statistinio nuspėjimo (angl. *statistical estimation*) problema, turinti aiškiai apibrėžtą objektyvų tikslą, galima įgyvendinti spektro atimties metodo trūkumų neturinčius algoritmus [19]. Didžioji dalis šių filtrų minimizuoja MMSE skirtumą [20] tarp nuspėto ir realaus švaraus signalo. Šis algoritmų grupė dar žinoma kaip Wienerio [21] filtrai.

Paprasti Wienerio filtrai – tiesiniai filtrai, bandantys triukšmingą signalą $y[n]$ išskaidyti į švarų signalą $x[n]$ ir triukšmo signalą $v[n]$. Daroma prielaida, kad šie du signalai nekoreliuoja vienas su kitu ir yra stacionarūs stochastiniai procesai [19].



1.5 pav. Wienerio filtras

Šis filtras dažniausiai įgyvendinamas kaip baigtinio impulso atsako (FIR) filtras arba begalinio impulso atsako (IIR) filtras. Wienerio filtro nuspėjamas švarus signalas $\hat{x}[n]$ gaunamas:

$$\hat{x}[n] = h_0^T y(n), \quad (1.8)$$

čia h_0 – optimalūs Wienerio filtro koeficientai:

$$h_0 = [h_1, h_2, \dots, h_L]^T. \quad (1.9)$$

O $y(n)$ – paskutinės L triukšmingo signalo vertės:

$$y(n) = [y[n], y[n-1], \dots, y[n-L+1]]^T. \quad (1.10)$$

Optimalus (Wienerio) filtras optimizuojamas taikant:

$$h_0 = \arg \min J_x(h), \quad (1.11)$$

čia $J_x(h)$ – kvadratinė klaida:

$$J_x(h) = e^2[n] = (x[n] - \hat{x}[n])^2. \quad (1.12)$$

1.3. Neuroniniai tinklai

Neuroninių tinklų esmė – suspausti didelį duomenų kiekį į iš anksto aprašytą duomenų struktūrą. Šio algoritmo tipo kūrimas buvo įkvėptas biologinių procesų ir struktūrų – mokymosi ir smegenų. Gali pasirodyti, kad mokymasis negali būti blogas sprendimas. Tačiau praėjo pusė amžiaus nuo neuroninių tinklų atsiradimo, kol jie tapo dažniausiais algoritmais garso atkūrimo problemoms spręsti. Paminėtina šio reiškinio priežastis – mokymas reikalauja didelio duomenų kiekio ir daug resursų. Nors žemutinė kiekio riba, reikalinga gerai veikiančiam algoritmui apmokyti, nėra tiksliai žinoma, apytikslis duomenų kiekis yra apie 100 tūkstančių pavyzdžių. Iki atsirandant

internetui ir neturint šių dienų duomenų saugojimo technologijų, šiuo algoritimų tipu dažniausiai naudodavosi tik didelius biudžetus turinčios kompanijos.

Tačiau šiais laikais apytikslė duomenų riba, reikalinga išmokyti neuroninį tinklą spręsti garso atkūrimo ar kitą problemą telpa daugelyje neprofesionalių įrenginių. Tai leidžia vis plačiau naudotis neuroniniais tinklais sprendžiant garso atkūrimo problemas.

1.3.1. Daugiasluoksniai perceptronai

Daugiasluoksniai perceptronai – populiariausia neuroninių tinklų architektūra. Tai mašininio mokymosi modelis, kuris įgyvendina parametrizuotą funkciją:

$$o = f(x, \theta), \quad (1.13)$$

čia x – įėjimo signalas, θ – modelio parametrai, o – modelio įėjimo duomenų transformacija, arba paprasčiau – jo išeiga. Pats paprasčiausias – vieno sluoksnio – perceptronas gali būti aprašytas taip:

$$o = f^{\{1\}}(x, \theta) = \phi(Wx + b), \quad \theta = \{W, b\} \quad (1.14)$$

W ir b – neuroninio tinklo parametrai ir $\phi(\cdot)$ – netiesinė funkcija, dažniausiai vadinama aktyvacijos funkcija. $\{1\}$ žymimas modelio sluoksnis. Tačiau šis modelis yra labai ribotas ir nesugeba aproksimuoti sudėtingų funkcijų (Marvin Minsky, Papert ir Block 1970). Šiai problemai spręsti tokie vieno sluoksnio modeliai tiesiog sudedami vienas po kito. Taip gaunamas daugiasluoksnio perceptrono modelis:

$$o = f^{\{L\}}(\dots f^{\{2\}}(f^{\{1\}}(x, \theta_1), \theta_2), \theta_L). \quad (1.15)$$

Daugiasluoksnių perceptronų modeliai, remiantis universalia aproksimavimo teorema, geba modeliuoti praktiškai bet kokią funkciją [23]. Pasak šios teoremos, $L=2$ užtenka aproksimuoti bet kokią funkciją, tačiau tokiu atveju tinklo plotis kiekviename sluoksnyje gali būti per platus praktiniams skaičiavimams. Tačiau sluoksnių skaičiui L didėjant, neuroninio tinklo struktūra įgalina modelį aproksimuoti vis sunkesnes ir sunkesnes funkcijas, negu tinklas, turintis mažesnę sluoksnių skaičių [24]. Šis faktas, kad didinant sluoksnių skaičių modeliai geba modeliuoti vis kompleksiškesnes funkcijas, lėmė vadinamojo giliojo mokymo pradžią. Žodis gilusis vartojamas sluoksnių kiekiui nusakyti. Nors gilesnius tinklus daug sunkiau mokyti, tyrėjai yra sėkmingai apmokę tinklus, turinčius daugiau negu 1000 sluoksnių [25].

1.3.2. Rekurentiniai neuroniniai tinklai

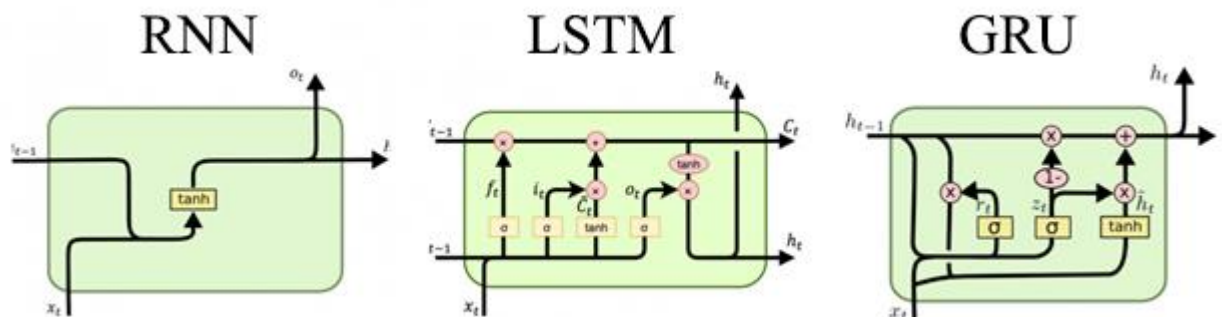
Rekurentiniai neuroniniai tinklai (RNN) – neuroninių tinklų pogrupis sukurtas sekoms modeliuoti. Nors universalus aproksimavimo teorema teigia, kad daugiasluoksniai perceptronai geba modeliuoti bet kokią funkciją, praktiškai tai ne visuomet įgyvendinama [26].

Paprastas vieno sluoksnio rekurentinis neuroninis tinklas apibrėžiamas taip:

$$h^{\{n\}} = \phi(Wx^{\{n\}} + Vh^{\{n-1\}} + b), \quad (1.16)$$

čia ϕ , W ir b – tokie patys parametrai kaip ir daugiasluoksniame perceptrone. Esminis skirtumas – tinklo išeiga yra skaičiuojama tam tikru metu n . Taip pat skirtingai negu MLP turi V parametą, kuris yra svorių matrica, nusakanti prieš vieną laiko momentą atgal buvusios būsenos svarbą. 1.16 formulė atskleidžia esminį skirtumą tarp MLP ir RNN architektūrų – RNN tinklas turi išeigą, priklausančią nuo laiko. MLP šios priklausomybės nuo laiko neturi. Tai esminis skirtumas įgalinantis rekurentines architektūras efektyviau išmokyti funkcijas priklausančias nuo pozicijos (šiuo atveju laiko). Kaip ir MLP, RNN modeliai gali būti kombinuojami į gilesnius modelius taip padidinant jų efektyvumą. Norint išmokyti tinklą modeliuoti ilgesnes sekas kiekvienam praėjusio laiko momentui tiesiog reikia sukurti papildomą V svorį. Tačiau padidėjus modelio kompleksiskumui, jie tampa sunkiai apmokomi. Taip nutinka dėl to, kad didėjant sluoksnių ir praeities laiko verčių kiekiui – tinklo gradientai tiesiog išnyksta.

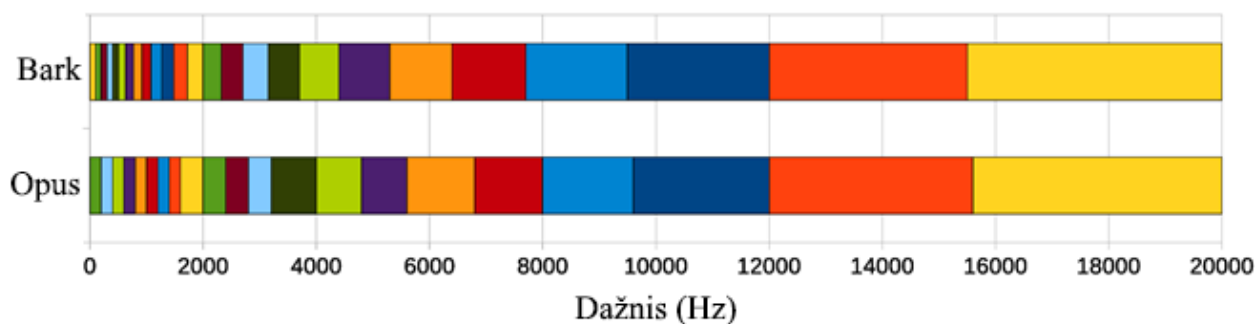
Šiai problemai spręsti buvo pasiūlyta keletas RNN patobulinimų. Populiariausi patobulinimų variantai – LSTM [27] ir GRU [28].



1.6 pav. Rekurentinių tinklų architektūros

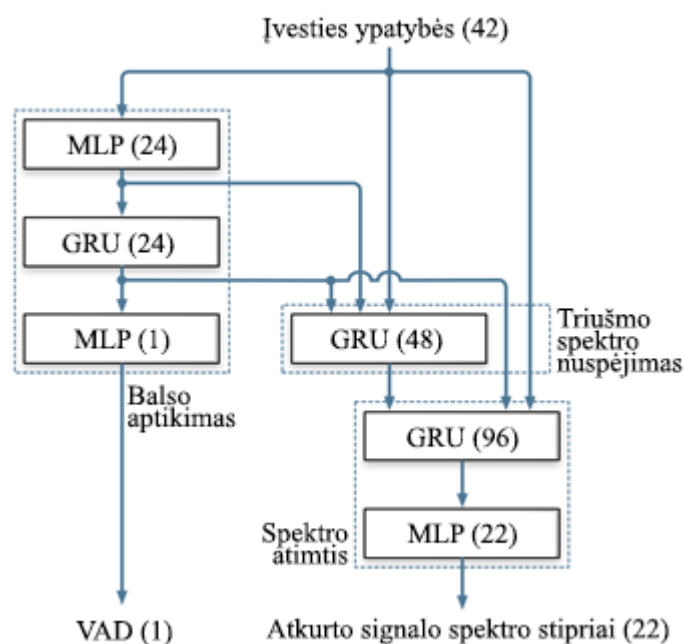
Šių architektūrų tikslas – įgalinti rekurentinius neuroninius tinklus išmokyti kompleksiškesnes signalų transformacijas, o tai padaryti ypač sunku dėl anksčiau minėtos nykstančių gradientų problemos. Tai atliekama dalinant keletą parametrų tarp skirtingo laiko imčių ir tarsi įgalinant atmintį. Tačiau gebėjimas atsiminti ilgesnes sekas vis tiek yra problema, nes ilgesnei atminčiai reikia didesnio kiekio parametrų matricių.

Vienas iš būdų, kaip rekurentiniais tinklais naudojamosi garso atkūrimo problemoms spręsti, – *RNN* neuroninio tinklo architektūra. Šiam modeliui apskaičiuojamas signalo keprastas (angl. *cepstrum*) paremtas *Opus* skale, bei pirmoji ir antroji, pirmų šešių keprasto verčių, išvestinės [15].



1.7 pav. Opus skalė

Tuomet šis ypatybių rinkinys yra modeliuojamas trimis GRU ir trimis MLP sluoksniais ir gaunamos dvi vertės – VAD ir reikiamas dažnių juostos garsumas. VAD tikslas – aptikti kalbą, o jeigu kalbos neaptinkama, tinklas tiesiog nutildo visą garsą.



1.7 pav. RNNoise tinklo architektūra

Jeigu kalbos signalas vis dėlto būna aptinkamas, bemokant tinklą yra skaičiuojama MSE klaida tarp nuspėtų ir realių švaraus garso spektro juostų. Kadangi šis tinklas yra tik trijų sluoksnių gylio, jis turi nedaug parametrų ir telpa į mažą talpą turinčius įrenginius. Tačiau dėl mažo parametrų kiekio jis negeba prisitaikyti prie skirtingų triukšmo tipų.

1.3.3. Sąsūkų neuroniniai tinklai

Sąsūkomis grįsti neuroniniai tinklai (CNN) yra panašūs į rekurentinius, nes turi bendras parametrų matricas. Rekurentiniai tinklai turi skirtingas matricas kiekvienam skirtingam laiko momentui, tačiau jeigu norimas modeliuoti signalas yra stipriai koreliuojantis su savimi, didelė dalis parametrų saugos tą pačią informaciją. Sąsūkų neuroniniai tinklai, panašiai kaip rekurentiniai

tinklai irgi turi keletą skirtingų filtrų, tačiau pamatinis skirtumas – skirtingose signalo vietose pritaikomi tie patys filtrai (priešingai negu skirtingoms vietoms taikomi skirtingi filtrai rekurentiniuose tinkluose). Ši savybė gaunama taikant sąsūkos operaciją tarp įvesties signalo ir filtro (dar vadinamo branduoliu), todėl šie tinklai ir vadinami sąsūkų neuroniniais tinklais. Esminė sąsūkos operacija apibrėžiama taip:

$$o_{t,a} = \sum_{m=1}^M \sum_{n=1}^N x_{t+m,a+n} w_{m,n}, \quad (1.17)$$

čia x – pamatinis modeliuojamas signalas, parametrizuotas laiko t ir amplitudės a , w – sąsūkos branduolys.

Taip pat kaip MLP ir RNN tinkluose, sąsūkos sluoksniai irgi turi daugelį sluoksnių ir netiesinių aktyvacijų tarp jų. Tačiau, skirtingai nuo jų, skaičiuojant kiekvieno sluoksnio išeią, parametrinė matrica dauginama ne su visu signalu, o tik su jo dalimi. Tai – pamatinis CNN ir kitų architektūrų skirtumas.

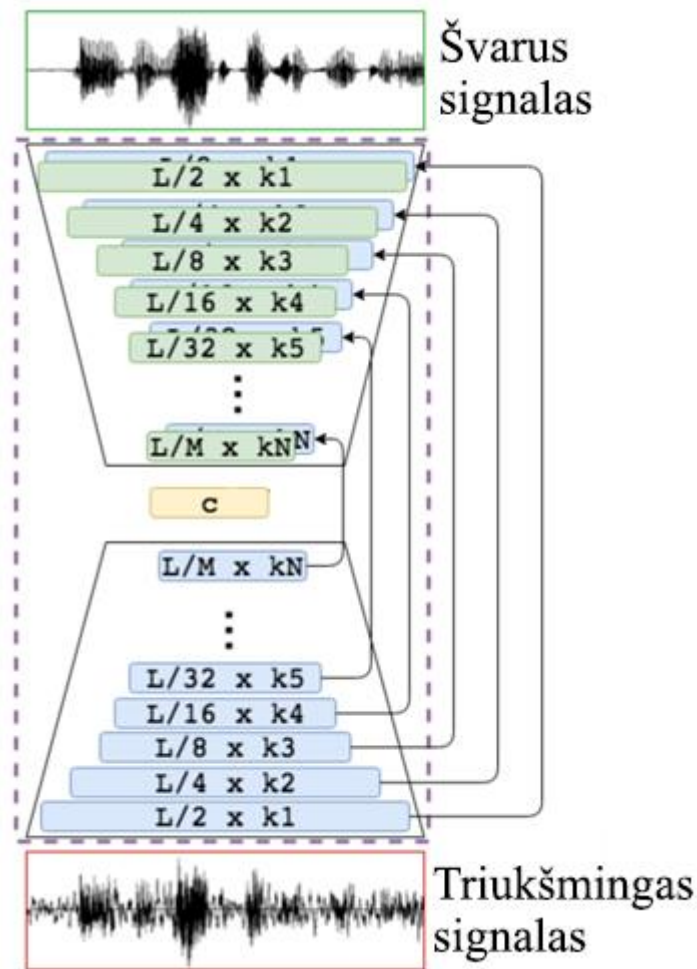
Kiekvienas sluoksnis turi skirtingus sąsūkos branduolius. Kiekvienas jų sukuria atskirą naują ypatybių signalą. Tam, kad šis signalas užimtų mažiau atminties, 1.17 aprašytoje formulėje t ir a parametrai didinami didesniu negu 1 poslinkiu. Kitas būdas sumažinti informacijos kiekį – telkimas (angl. *pooling*), tačiau garso atkūrimo uždaviniams spręsti jis netaikomas, nes jis glaudina signalą ir taip sunaikina jame esančią informaciją.

Kadangi modelio parametrų kiekis priklauso nuo jo filtrų dydžio M ir N , o ne nuo įėjimo signalo dydžio RNN ir MLP modeliuose, sąsūkomis grįsti tinklai reikalauja mažesnio kiekio parametrų panašiam efektyvumui pasiekti. Tai ypač pastebima sprendžiant kompiuterinės regos bei garso apdorojimo problemas. Kalbos signaluose tyla neperteikia naudingos informacijos, o žmogaus ištartos balsės jos turi daugiau. Tokiais atvejais balsės randantis sąsūkos operatorius gali daug efektyviau jas rasti, nešvaistydamas parametrų nenaudingoms signalo vietoms filtruoti.

Vieni iš sėkmingiausių būdų, kaip spręsti garso atkūrimo triukšmingoje aplinkoje problemą – SEGAN [29] ir *Conv-TasNet* [5] sąsūkomis grįsti neuroniniai tinklai.

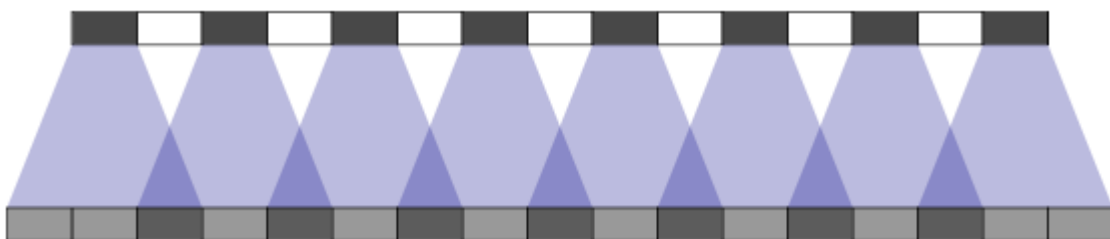
SEGAN tinklas iš tikrųjų sudarytas iš dviejų neuroninių tinklų – generatoriaus ir diskriminatoriaus. Generatoriaus tikslas – panaikinti iš triukšmingo signalo triukšmą. Diskriminatoriaus tikslas – nustatyti, kaip gerai triukšmas buvo panaikintas. Diskriminatoriaus tinklas iš tiesų veikia kaip klaidos funkcija, kurią generatoriaus tinklas turėtų minimizuoti. Jis sudarytas iš vienuolikos sąsūkos sluoksnių, turinčių po 31 vertės ilgio filtrus ir vieną MLP sluoksnį gale tinklo, kurio tikslas nuspręsti, ar signalas triukšmingas ar ne. Generatoriaus architektūra – autoenkoderis – sudaryta iš enkoderio, kurio tikslas – suglaudinti signalą, ir dekoderio, kurio tikslas

iš suglaudinto signalo atkurti švarų. Tai įgyvendinama naudojantis 11 sąsūkos sluoksnių, kurių kiekvienas kurių turi 16 skirtingų 31 verčių ilgio sąsūkos branduolių, kurių kiekvieno poslinkis – 2.



1.8 pav. SEGAN architektūra

Suglaudintas signalas dekoderyje atkuriamas naudojantis transponuotosiomis sąsūkomis (angl. *transposed convolutions*). Transponuotųjų sąsūkų filtrų bei poslinkio parametrai tokio pat dydžio, tačiau yra atskiri parametrai. Transponuotosios sąsūkos geba atkurti didesnį verčių kiekį turinčius signalus, tačiau neatsargiai parinkus jų parametrus atsiranda nepageidautinų artefaktų – garso signaluose tai gali būti zirzimas ar anksčiau minėtas muzikinis triukšmas [30]. Tai atsitinka dėl to, kad tam tikrose signalo dalyse susideda kelių transponuotų sąsūkų filtrų išeigos.



1.9 pav. Transponuotosios sąsūkos vizualizacija

Paminėtina, kad kiekviename dekoderio sluoksnyje, įėjimo signalas turi dvigubai didesnį kiekį kanalų, nes prie originalaus dekoduojamo signalo prijungiami atitinkamo enkoderio sluoksnio išseigos signalai. Tai vadinamosios praleidimo jungtys (angl. *skip-connections*).

Conv-TasNet architektūra labai panaši į SEGAN, tačiau skiriasi keletas pasirinkimų. Skaičiavimų kiekiui sumažinti kiekviename sluoksnyje sąsūkos poslinkis yra 50 % sąsūkos ilgio. Taip pat sąsūkos branduolių kiekis kiekviename sluoksnyje padvigubinamas, tačiau tam, kad būtų išlaikytas tas pats kiekis parametrų, sąsūkų vertės sumažinamos iki 16. Tačiau kadangi autoriai naudojo dvigubai mažesnio diskretizacijos dažnio signalus (8 kHz), tai yra tokio pat dydžio sąsūką kaip ir SEGAN. Jos autoriai naudojo 16 kHz diskretizacijos dažnio signalus. Taip pat kiekvieno sąsūkos sluoksnio ypatybės normalizuojamos naudojant globalią sluoksnių normalizaciją [31]. Tiek SEGAN, tiek *Conv-TasNet* architektūros aktyvacijos funkcijoms naudoja parametrinę rektifikatorių (PReLU). Nors abi architektūros panašios, jų veikimas nebuvo palygintas.

1.3.4. Poskyrio apibendrinimas

Apžvelgti daugiasluoksnių perceptronų, rekurentiniai ir sąsūkomis grįsti neuroninių tinklų modeliai. Taip pat aprašyti būdai, kaip juos taikant sprendžiamos signalų atkūrimo triukšmingoje aplinkoje problemos.

1.4. Trumpas skyriaus apibendrinimas

Nors statistiniai metodai tiriami nuo XX a. aštunto dešimtmečio ir šie tyrimai davė gerų rezultatų, jų kūrimas – labai sunki problema, reikalaujanti daug eksperimentavimo ir sunkiai prisitaikanti prie skirtingų duomenų.

Neuroniniais tinklais grįsti metodai davė labai gerų rezultatų, tačiau labai dideli tinklai apsunkina algoritmų veikimą realiuoju laiku. Tačiau apžvelgti pagrindinės architektūros ir jų pasirinkimų elementai reikalingi norint toliau tobulinti algoritmus, skirtus garso atkūrimui triukšmingoje aplinkoje.

2. GARSO ATKŪRIMO ALGORITMO ĮGYVENDINIMAS

Šiame skyriuje apžvelgiami svarbiausi sprendimai, kuriuos reikia priimti kuriant garso atkūrimo algoritmą. Pirmas ir svarbiausias sprendimas yra algoritmo tipo pasirinkimas. Pasirinkus kuriamo algoritmo tipą, antras svarbiausias sprendimas – kokius ir kaip transformuotus duomenis algoritmas turės apdoroti. Žinant duomenų tipą galima priimti sprendimą, kaip modeliuoti garso signalą. Pagaliau nusprendus, kaip modeliuojamas signalas, galima priimti paskutinį sprendimą – kaip pasirinktas algoritmas bus vertinamas ir optimizuojamas.

2.1. Algoritmo pasirinkimas

Svarbiausias pasirinkimas nagrinėjant algoritmus – pasirinkti jo tipą. Kadangi šiame darbe nagrinėjama garso atkūrimo problema, algoritmas turėtų prisitaikyti prie įvairių norimo atkurti garso ypatybių. Dėl šios norimos ypatybės atmetami statistiniai algoritmai.

Renkantis neuroninio tinklo architektūrą MLP tinklai atmetami dėl pernelyg didelio parametrų reikalavimo sekų problemoms spręsti. Rekurentiniai tinklai, nors ir puikiai tiktų nagrinėjamai problemai modeliuoti, yra itin sunkiai apmokomi ir net optimalus mokymas reikalauja labai didelių skaičiavimo resursų ir laiko [32]. Paskutinis likęs neatmestas algoritmų variantas – sąsukų neuroniniai tinklai, esantys efektyvūs ir gebantys prisitaikyti prie skirtingų signalų.

2.2. Duomenys ir jų paruošimas

Duomenys – viena iš svarbiausių neuroninių tinklų sudedamųjų. Turint didelius duomenų kiekius tinklų mokymas gali užtrukti labai ilgai, tačiau rezultatas dažnai bus geresnis negu naudojantis mažu duomenų kiekiu [33]. Tačiau atliekant eksperimentus – labai svarbu, kad bandant skirtingus modulius ir modelio dalis būtų galima jų veikimą palyginti. Taip pat svarbu, kad eksperimentai netruktų ilgai ir būtų galima ištirti kuo daugiau skirtingų modeliavimo variantų.

Optimalaus duomenų rinkinio parinkimo problemai spręsti tyrime naudojamosi viešai prieinamu duomenų rinkiniu [10]. Duomenų rinkinyje aiškiai atskirti duomenys, naudotini modeliui mokyti ir testuoti. Jie rinkinyje suskirstyti į triukšmingo garso ir švaraus garso poras. Švarus garsas gaunamas iš 84 kalbėtojų įrašų (41 vyras ir 41 moteris) anglų kalba. Antrasis, triukšmingas garsas, gaunamas prie švarių garsų pridėdant triukšmą. Naudojama 10 skirtingų triukšmo tipų – 2 dirbtinai sugeneruoti ir 8 realaus triukšmo įrašai, paimti iš *Demand* duomenų bazės. Dirbtinis triukšmas generuojamas baltą triukšmą filtruojant filtru, kurio dažnių atsakas toks pats kaip duomenų rinkinyje esančių vyrų. Šis triukšmo tipas imituoja neraiškią šneką. Antrasis dirbtinis triukšmas sukuriamas atsitiktiniu būdu parenkant šešis atskirus kalbėtojus ir sudedant jų garso signalus. Taip sukuriamas triukšmas, kuris imituoja žmonių minią. Natūralūs triukšmai, naudojami kuriant rinkinį:

- Virtuvėje girdimas triukšmas;
- Susitikimų kambario triukšmas;
- Trys viešos vietos (kavinė, restoranas ir transporto stotelė);
- Mašinos triukšmas;
- Metro stotelės triukšmas (aplink kalbantys žmonės, pravažiuojantys traukiniai);
- Triukšmas sankryžoje (vaikščiojantys, kalbantys žmonės, pravažiuojančios mašinos).

Algoritmo mokymo duomenims sukurti naudotos SNR: 15 dB, 10 dB, 5 dB ir 0 dB. Triukšmo ir švaraus signalo sudėčiai taikytas ITU-T P.56 metodas [34].

Visi duomenys rinkinyje kvantuoti 48 kHz dažniu, tačiau tyrimui šis dažnis sumažinamas iki 16 kHz. Duomenys kompiuteryje saugomi WAV formatu ir prieš modeliavimą kiekvienas failas paverčiamas į duomenų masyvą, kurio vertės normalizuojamos tarp -1 ir 1.

2.3. Modelio architektūra

Algoritmui įgyvendinti ir tolesniems tyrimams taikomas sąsūkomis grįstas neuroninio tinklo, autoenkoderio modelis.

Svarbiausi klausimai, į kuriuos reikia atsakyti kuriant tokio tipo modelį, yra:

- Kokie ir kaip transformuoti duomenys bus modeliuojami?
- Kaip bus filtruojami duomenys?
- Kiek sluoksnių modelis turėtų turėti?
- Kaip bus normalizuojami filtruoti duomenys?
- Kokias aktyvavimo funkcijas pasirinkti?
- Kaip filtruoti enkoderio suglaudintą signalą?
- Kaip iš suglaudinto signalo dekoduoti garso signalą?
- Ar derėtų sujungti enkoderio ir dekoderio duomenis?

Šiame skyriuje minėti klausimai nagrinėjami ir pateikiami pagrindiniai atsakymų variantai.

2.3.1. Įvesties duomenys

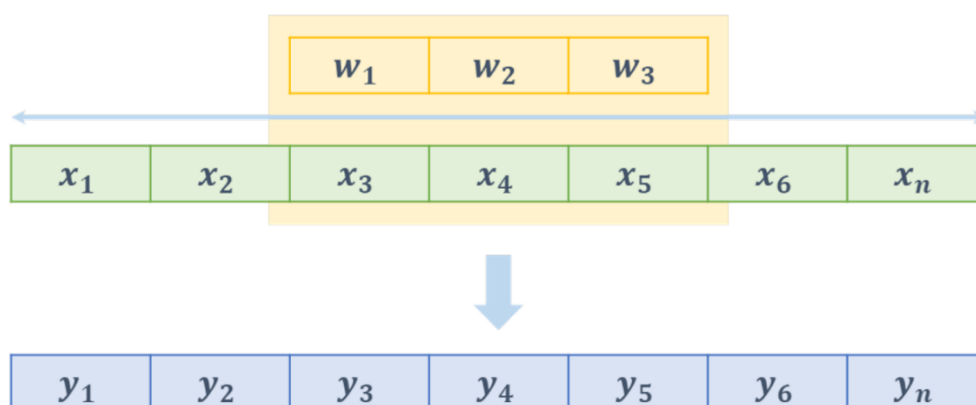
Nors naudojimas spektrogramomis buvo sėkmingas (minėta analitinėje literatūros apžvalgoje), jų kūrimas reikalauja daug kruopščių sprendimų. Kompiuterinės regos srityje žmonių lygis pasiektas tada, kai buvo nustota remtis tyrėjų sugalvotomis naudingos ypatybėmis ir modeliai patys galėjo išmokti jiems naudingas ypatybes iš pamatinių duomenų [35]. Šis pasirinkimas lemia, kad modeliui būtina turėti didesnę kiekį parametrų ir ilgiau mokytis. Tačiau tai padaro patį modelį lankstesnį – jis turi galimybę prisitaikyti prie skirtingų duomenų rinkinių dažninių charakteristikų [36]. Modeliui paduodami 2.2 poskyryje aprašyto duomenų rinkinio – dviejų sekundžių 16 kHz garso signalai – partija (angl. *batch*). Partijos kiekis P svyruoja priklausomai nuo modelio dydžio,

tačiau yra didinamas tol, kol telpa grafiniame procesoriuje. Kadangi modeliuojami signalai yra vieno kanalo, įėjties vektorius yra trijų dimensijų – $P, 1, L$. Čia P – partijos dydis, $L = 32,000$ – diskretizuoto signalo verčių kiekis.

2.3.2. Enkoderio filtrai

Sąsukomis grįstų enkoderių tikslas – atrasti duomenyse dažniausiai pasikartojančias ir naudingas ypatybes. Šie filtrai neuroniniais tinklais grįstuose modeliuose beveik visuomet būna bent kelių sluoksnių gylis – filtruoti duomenys vėl filtruojami ir tai kartojama, kol pasiekiamas galutinis iš anksto nustatytas sluoksnių skaičius (gylis). Tai vienas iš pamatinių neuroninio tinklo parametrų. Jeigu tinklas turės daug sluoksnių, jis dėl nykstančių gradientų gali negebėti išmokyti reikiamų ypatybių. Sąsukos filtrų dydis ir neuroninio tinklo sluoksnių gylis nusako svarbiausią parametą – suvokimo lauką (angl. *Receptive field*) [37]. Pavieniuose filtruose, skirtuose garso atkūrimo problemai spręsti, jis dažniausiai būna tarp 2 ms ir 6 ms. 16 kHz diskretizavimo dažnį turinčiame signale filtro imtis turėtų būti tarp 32 ir 96 [5]. Palyginus su spektrogramomis grįstų sprendimų suvokimo lauku, tai gana mažo dydžio filtrai.

2.1 paveikslėlyje pavaizduotas tik vieno filtro pavyzdys. Jeigu jį pritaikysime pamatiniam garso signalui, šis filtras galės išmokyti tik vieną signalo ypatybę. Nors didinant tinklo gylį filtrų bus daugiau, jie tiesiog modeliuos šios ypatybės pasikartojimą signale. Tam, kad modelis išmokyti platesnį spektrą ypatybių, pavyzdžiui, ne tik balses, bet ir priebalses arba kitus garso vienetų, pirmajame sluoksnyje filtrų kiekis būna didesnis, palyginus su prieš tai buvusių kanalų kiekiu [38], [39]. Taip sukuriamas signalas, sudarytas iš daugelio ypatybių ir jų pasikartojimų signale – ypatybių žemėlapis (angl. *feature map*). Ši daugelio filtrų pritaikymo ankstesniam signalui procedūra kartojama ir taip gaunamos vis abstraktesnės ypatybės.



2.1 pav. Vienos dimensijos sąsukos operacija

Kitas svarbus sąsūkos parametras – kiek verčių bus slenkamas filtras. Šis parametras literatūroje kinta nuo 1 [29] iki 50 % filtro ilgio [5] – 32 verčių filtro poslinkis tampa 16. Didesnis poslinkis reiškia mažesnę jutiminio lauko rezoliuciją, tačiau didesnį signalo glaudinimą. Kuo didesnis poslinkis tarp filtrų pritaikymų, tuo mažesnis skaičiavimų kiekis reikalingas signalui apdoroti. Taikant tokį filtrą daugelio branduolių kompiuteriuose galima šiuos skaičiavimus nesunkiai paskirstyti. Atliekant tyrimą dėl 2.3.7 aptartų priežasčių naudojamas pusės filtro verčių poslinkis.

Paskutinis sąsūkos operacijos parametras, kurį reikia pasirinkti, – užpildymas (angl. *padding*). Jis reikalingas tam, kad sudauginus signalo dalį su sąsūkos filtru gaunamas ypatybių žemėlapis turėtų norimą verčių kiekį. Taip pat jeigu sąsūkos filtras turi daug verčių, užpildymas išsprendžia problemą, kad dešiniausios filtro vertės nebus sudauginamos su pirmosiomis signalo vertėmis, nes tada signalo jutimiam laukui iki galo nepersiklojant su signalu modelis gali netinkamai sumodeliuoti signalą. Dažniausiai užpildymui naudojama nulio vertė, taip dauginant ją su sąsūkos filtru nebus perduota signale neegzistuojanti informacija.

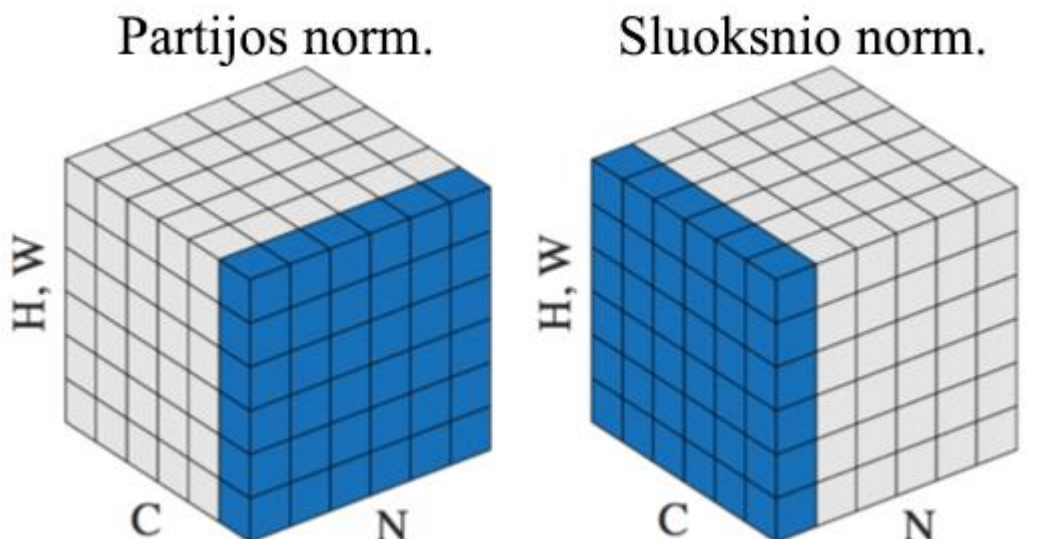
2.3.3. Normalizacija

Normalizacijos procedūros buvo sukurtos tam, kad paspartintų ir stabilizuotų neuroniniais tinklais grįstų modelių mokymąsi [40]. Kiekvienas parametrizuotas neuroninio tinklo sluoksnis turi tam tikrą statistinį pasiskirstymą. Šis pasiskirstymas atsiranda atsitiktinai inicializuojant sluoksnio parametrus ir mokymo metu kintant sluoksnio parametrus. Normalizacijos tikslas – šiuos parametrus transformuoti taip, kad mokymo algoritams būtų lengviau optimizuoti sluoksnių filtrų parametrus.

Viena populiariausių normalizacijos funkcijų – partijos normalizacija (angl. *Batch normalization*), plačiai taikoma kompiuterinės regos problemoms spręsti.

$$y = \frac{x - E[x]}{\sqrt{Var[x] + \varepsilon}} \gamma + \beta \quad (2.1)$$

Šios normalizacijos tikslas – normalizuoti gautą ypatybių vektorių taip, kad visų vieno kanalo ypatybių vidurkis būtų kuo arčiau nulio ir standartinis nuokrypis būtų lygus vienetui. Normalizacija atliekama naudojant skalės parametą γ ir nuokrypio parametą β . Konstanta ε – labai mažos vertės skaičius, naudojamas siekiant išvengti dalybos iš nulio. Normalizuojamas signalas x – atskiros ypatybės vektorius (žr. 2.2. pav.). Šie parametrai mokomi kartu su visais kitais tinklo parametrais.



2.2 pav. Normalizacijos metodų palyginimas (mėlynai nuspalvintas normalizuojamas vektorius)

Tačiau empiriškai nustatyta, kad nors šis metodas irgi paspartina neuroninių tinklų mokymą sprendžiant garso atkūrimo problemas – sluoksnių normalizacija [41] šioms problemoms – efektyvesnė. 2.1 formulė nekeičiama ir veikia taip pat, kaip aprašyta partijos normalizacijoje, tik jai perduodamas normalizuotinas vektorius yra ne atskiro kanalo ypatybės, tačiau atskiro garso signalo visi ypatybių kanalai [5]. Tyrimo metu nagrinėjamos abi – partijos ir sluoksnio normalizacijos ir jų įtaka algoritmo veikimui.

2.3.4. Aktyvavimo funkcija

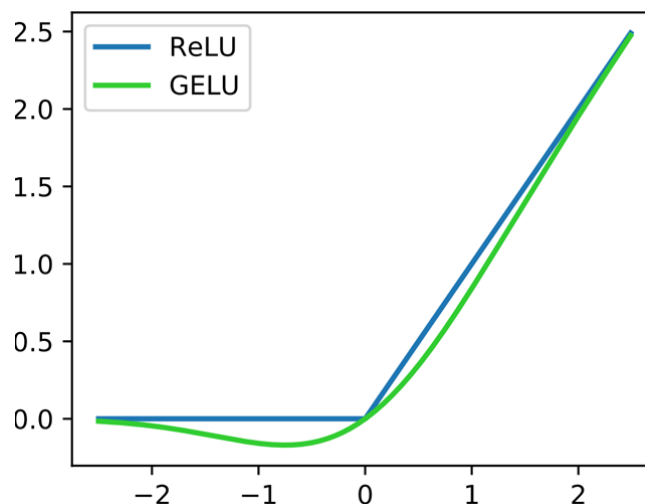
Aktyvavimo funkcijos (taip pat vadinamos netiesiškumu) tikslas – transformuoti įėjimo signalą netiesiškai ir taip įgalinti neuroninį tinklą išmokti kompleksiškesnes įvesties ir išvesties priklausomybes. Nenaudojant aktyvavimo funkcijų dingsta prasmė turėti daugelį neuroninio tinklo sluoksnių, nes daugelio matricių daugybą galima suredukuoti iki vienos matricos.

Populiariausia aktyvavimo funkcija kompiuterinės regos srityje – ReLU (2.2) [42]. Esminis jos privalumas – ją naudojant neuroniniai tinklai greičiau ir stabiliau mokosi. Tačiau garso atkūrimo triukšmingoje aplinkoje problemoms spręsti ReLU pasirenkama retai. Minėtai problemai spręsti dažniausiai naudojama šiek tiek modifikuota ReLU funkcija – PReLU [35] (2.3). Čia α – neuroninio tinklo parametras, kintantis mokymo metu. Literatūroje siūloma ir daug kitų funkcijų, netaikytų garso atkūrimo problemai spręsti. Pora jų – ELU (angl. *exponential linear unit*) ir GELU (angl. *gaussian error linear unit*) [43]. Atliekant tyrimą nagrinėjamos kompiuterinės regos srities literatūroje aprašytos ReLU funkcijos, garso atkūrimo problemai spręsti naudojamos PReLU ir tolesnio tyrinėjimo reikalaujančiomis GELU (2.4) aktyvavimo funkcijomis.

$$ReLU(x) = \max(0, x) \quad (2.2)$$

$$PReLU(x) = \max(\alpha x, x) \quad (2.3)$$

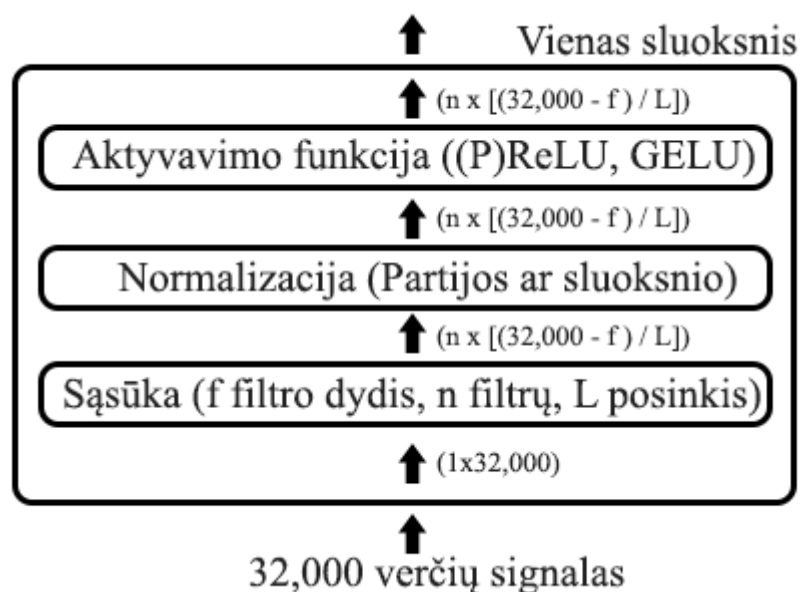
$$GELU(x) = 0.5x(1 + \tanh\left(\sqrt{\frac{2}{\pi}}(x + 0.044715x^3)\right)) \quad (2.4)$$



2.3 pav. ReLU ir GELU aktyvavimo funkcijos

2.3.5. Trumpa enkoderio apžvalga

Apžvelgus įvesties duomenis, sąsūkos filtrus, normalizacijos metodus bei aktyvacijos funkcijas, galima apžvelgti enkoderio taikymą bei pagrindinius tyrinėtinius parametrus. Pamatinis funkcinis blokas – sluoksnis, iš kurio bus sudaromas enkoderis, pavaizduotas 2.4 paveikslėlyje.



2.4 pav. Enkoderio sluoksnis

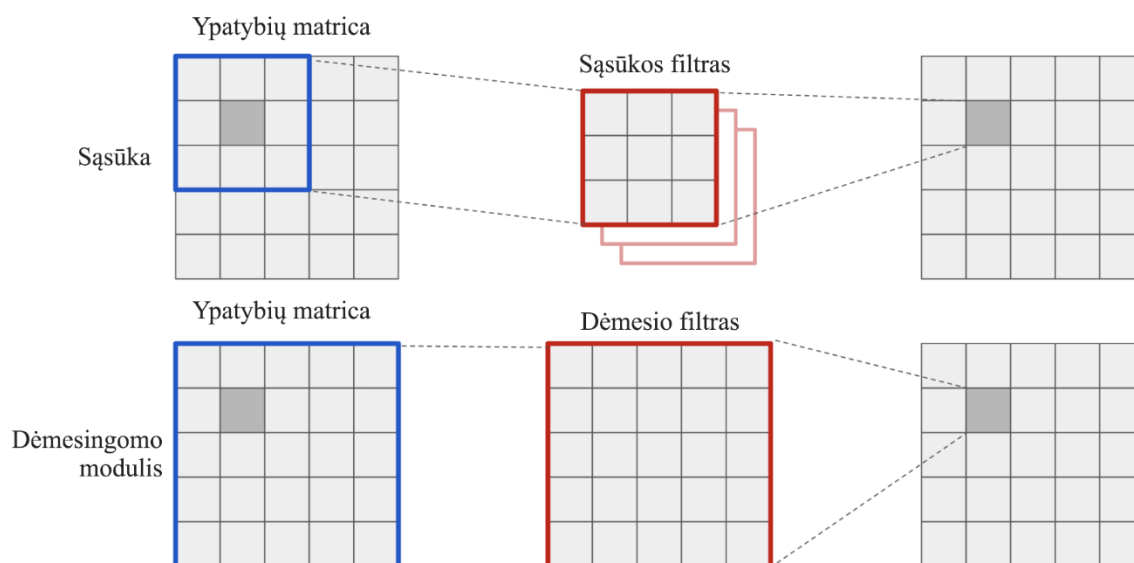
Pagrindiniai tiriami enkoderio parametrai:

- Kanalų gausėjimas kiekviename sluoksnyje (filtrų kiekis);
- Filtrų poslinkiai;
- Aktyvavimo funkcijos;
- Normalizacijos tipas;
- Optimalus enkoderio sluoksnių kiekis.

2.3.6. Glaudinto signalo filtravimas dėmesio moduliais

Pritaikius enkoderio filtrus pamatiniam garso signalui gaunamas abstraktus ypatybių žemėlapis – suglaudintas signalas. Jo galima toliau nefiltruoti ir tiesiog perduoti šį signalą dekoderiui, kad atkurtų norimą švarų signalą. Tačiau suglaudintas signalas įgalina kurti visą signalą aprėpiančius filtrus. Tai leidžia signalo pradžiai, tiesiogiai daryti įtaką jo pabaigai ir atvirkščiai [44].

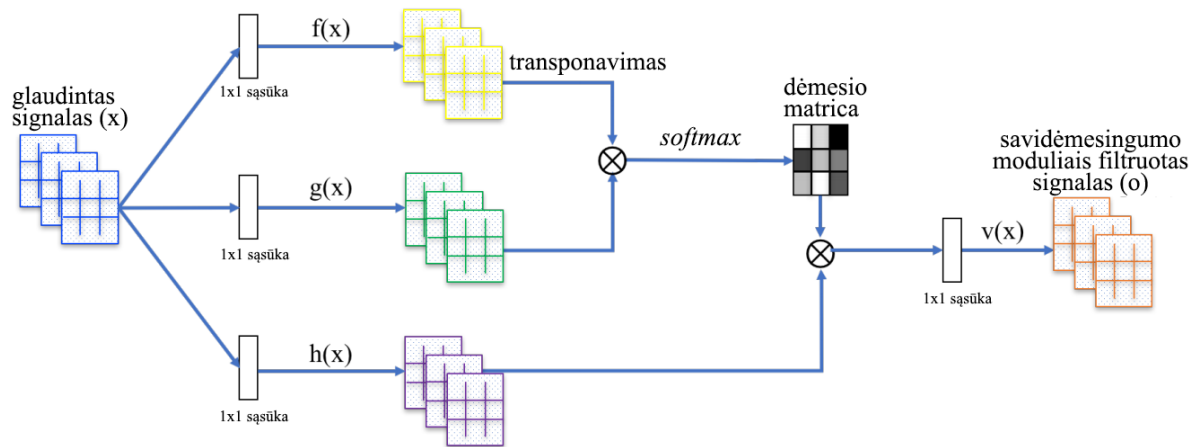
Vienas tokių filtravimo metodų – savidėmesingumo (angl. *self-attention*) moduliai [45]. Savidėmesingumo moduliai – transformerio tipo architektūrų pagrindas, tačiau jie gali būti taikomi ir sąsūkomis grįstuose tinkluose. Jų tikslas – atrasti sąsajas tarp skirtingų signalo vietų [46]. Sprendžiant tyrinėjamą problemą šie moduliai gali padėti atrasti triukšmus ir juos pašalinti. Dėmesingumo moduliai garso atkūrimo problemai spręsti anksčiau nenagrinėti, todėl tyrimo metu naudojamosi mašininio teksto vertimo literatūroje aprašytais metodais [47].



2.5 pav. Dėmesingumo modulio palyginimas su sąsūkos operatoriumi

Filtruojant glaudintą signalą sukuriama trys sąsūkos filtrai (f , g ir h), jų sąveikos su glaudintu signalu pavaizduotos 2.6 pav. Pritaikius filtrus signalui gaunamas naujas ypatybių žemėlapis. Tačiau esant kompleksiniam signalui vieno minėtų filtrų bloko neužtenka: dėmesingumo matricos išmoka tik matytų signalų vidurkį. Tam, kad savidėmesingumo filtrai gebėtų filtruoti kompleksinius signalus, vietoje vieno bloko – naudojamosi keliais (dažniausiai 8) [45]. Tačiau

esant daugiau negu vienam savidėmesingumo filtrui padidėja signalo kanalų kiekis. Tam, kad jis vėl būtų sulyginamas su glaudinto signalo kanalų kiekiu, taikoma vienos vertės ilgio sąsūka. Sąsūkos filtrų kiekis lygus enkoderio suglaudinto signalo kanalų kiekiui.



2.6 pav. Savidėmesingumo filtro veikimo diagrama

2.3.7. Dekoderio filtrai

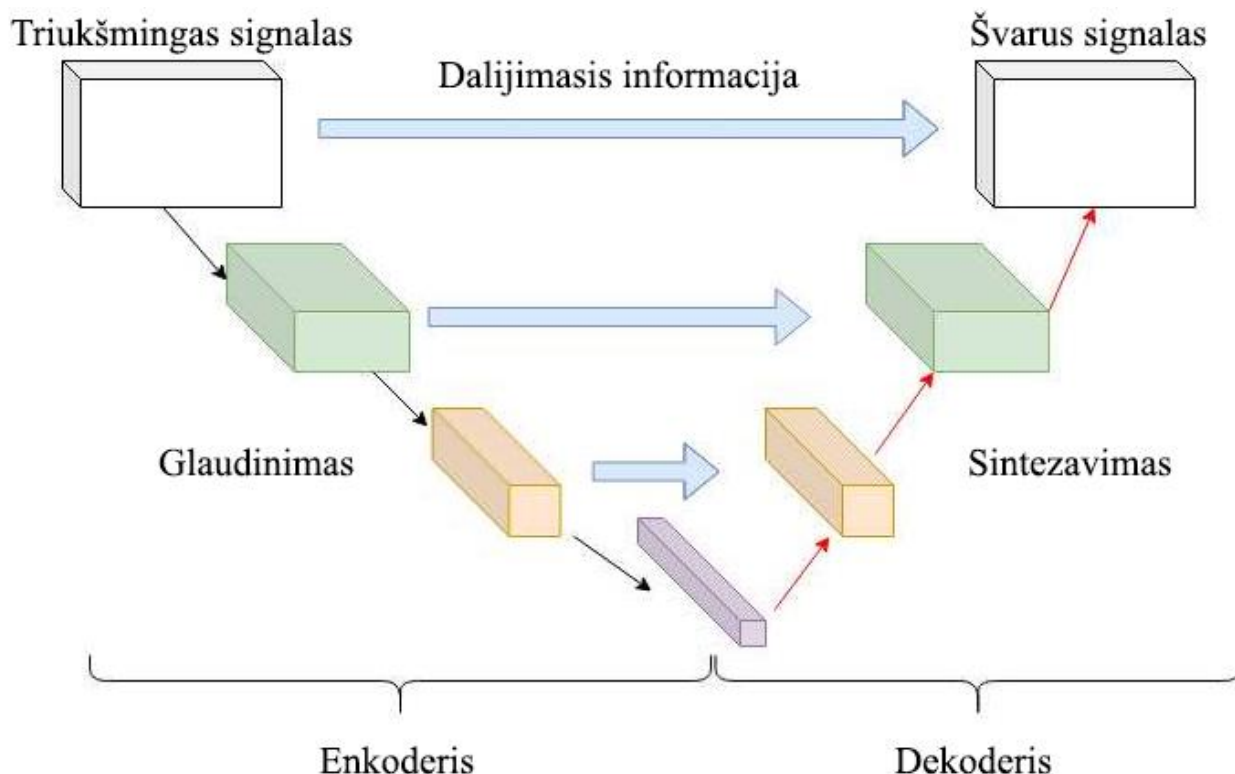
Kai enkoderio ir dėmesingumo moduliai suglaudina ir išfiltruoja signalą, reikia jį atkurti. Enkoderis suglaudina originalų signalą, o dekoderio filtrų tikslas – jį išplėsti. Ši atkūrimo problema dar vadinama sintezavimu.

Kaip jau minėta literatūros apžvalgoje, dažniausiai signalui atkurti naudojamosi transponuotųjų sąsūkų sluoksniais. Kadangi transponuotoji sąsūka – atvirkštinė sąsūkai operacija, esant identiškiems sluoksnių kiekiams, filtrų bei poslinkių parametrų, atkurtas signalas visuomet bus atkurtas tokių pat dimensijų kaip ir pirminis, triukšmingas, signalas. Tačiau tam, kad signalo transformavimas transponuotosiomis sąsūkomis nesukurtų artefaktų, filtro verčių kiekis ir poslinkis būtinai turi būti dalūs [48]. Vienas optimaliausių variantų, siūlomų literatūroje, – taikyti pusės filtro verčių ilgio poslinkį, prieš filtravimą signalo pradžią bei pabaigą užpildant (angl. *padding*) pusės sąsūkos filtro verčių ilgio nulio signalu. Pritaikius filtrą pritaikymo atitinkamos vertės iš signalo pašalinamos [30].

Tam, kad išvengtume artefaktų – atliekant tyrimą taikomas metodas, aprašytas kompiuterinės regos srities literatūroje. Metodo esmė – išskaidyti transponuotąją sąsūką į dvi dalis: signalo verčių išplėtimą (angl. *upsampling*) ir kanalų atkūrimą (filtravimą). Verčių išplėtimas dažniausiai įgyvendinamas naudojant artimiausių kaimynų metodą (angl. *nearest neighbor*). Šios transformacijos tikslas – atkurti enkoderio sąsūkos filtrų suspaustą signalą. Atkūrus reikiamą signalo dydį jis filtruojamas sąsūkų filtrais. Taip daroma tam, kad būtų atkurtas reikiamas signalo

kanalų (gylio) dydis. Sąsūkų filtrų kiekis lygus atitinkamo enkoderio sluoksnio įėjimo signalo kanalų kiekiui [49].

Kadangi dekoderis dažniausiai turi tiek pat sluoksnių, kiek ir enkoderis, norint filtruoti kompleksiškesnius signalus reikia gilesnio tinklo. Gilesniame tinkle gradientai gali dingti ir tada nepavyks apmokyti tinklo. Šiai problemai spręsti dažniausiai pasitelkiamos praleidimo jungtys (angl. *skip-connections*). Jų tikslas – perduoti informaciją iš enkoderio dekoderiui. Taip dekoderio filtrai geba suvokti pamatinį ir filtruotą signalą vienu metu. Taip pat šis informacijos perdavimas padeda nenunykti gradientams [50].



2.7 pav. Informacijos dalijimasis tarp enkoderio ir dekoderio naudojant praleidimo jungtis

Norint dalytis informaciją pasirenkama viena iš dviejų operacijų. Signalai sudedami, sudedant atitinkamus jų elementus (angl. *element-wise addition*) arba tiesiog pridedant vieno signalo vertes prie kito. Taip padidinamas kanalų skaičius (angl. *concatenation*) [5], [29].

Sudėjus atitinkamus elementus, modelio tikslas yra kiekviename sluoksnyje išmokyti atvirkštinį triukšmo signalą – kaukę. Sintezuojant ją sudėjus su pamatiniu signalu, triukšmas bus pašalintas sujungiant duomenis.

Jeigu pasirenkamas kanalų sudėjimas, modelio tikslas yra taip išmokyti sąryšį tarp sudėtų ypatybių, kad sintezavimo metu transformuojant glaudintą signalą sąsūkos filtrais šie filtrai, stebėdami abu signalus, nuspręstų, koks turi būti švarus signalas. Šio metodo trūkumas – kiekvienas

dekoderio sluoksnio filtras turi turėti dvigubai daugiau parametrų sąsūkos filtruose, negu anksčiau minėtas elementų sudėjimo metodas.

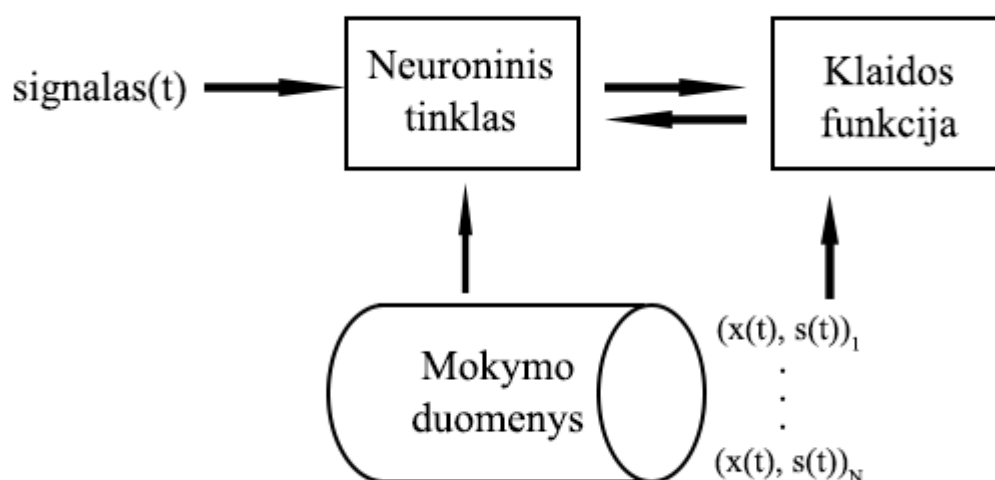
Atliekant tyrimą analizuojami abu metodai ir nesidalijimo informacija metodas (žr. 2.1 programos kodą).

2.1 dalijimosi informacija kodas

```
def resolve_skip_op(self, enoder_skip, decoder_out):
    if self.skip_op == 'add':
        return enoder_skip + decoder_out
    elif self.skip_op == 'cat':
        return torch.cat((enoder_skip, decoder_out), dim=1)
    elif self.skip_op == 'none':
        return decoder_out
```

2.4. Modelio optimizavimas

Modeliuojant neuroninius tinklus neužtenka tiesiog parinkti filtrų tipus bei jų parametrus. Šis modelių tipas visuomet optimizuojamas keičiant filtrų vertes taip, kad jos kuo efektyviau modeliuotų mokymo duomenų rinkinį. Sąsūkos dirbtiniams neuroniniams tinklams optimizuoti taikomi įvairūs giliojo mokymo algoritmai. Dauguma jų remiasi atgalinio klaidos sklidimo algoritmu. Šis algoritmas remiasi klaidos funkcija, kuri apskaičiuoja klaidą tarp modelio nuspėto ir tikrojo signalo [51]. Šiame poskyryje rašoma apie tyrime taikomas klaidos funkcijas ir mokymo algoritmus, atnaujinančius neuroninio tinklo parametrus.



2.8 pav. Neuroninio tinklo mokymo diagrama

2.4.1. Klaidos funkcija

Klaidos funkcijos esmė – objektyviai įvertinti neuroninio tinklo spėjimą. Neuroninio tinklo parametrai atliekant tyrimą inicijuojami *Kaiming* metodu [35]. Pradinių parametro verčių tikslas nėra tinkamai modeliuoti signalus, bet įgalinti 2.4.2 aprašomus mokymo algoritmus efektyviau keisti neuroninio tinklo parametrų vertes taip, kad parametrai tinkamai modeliuotų duomenų rinkinį.

Viena populiariausių klaidos funkcijų – MSE (2.5). Ji pasitelkiama daugeliui problemų spręsti ir yra labai paprasta:

$$MSE(x, \hat{x}) = \frac{1}{N} \sum_{n=0}^N (x - \hat{x})^2, \quad (2.5)$$

čia N – lyginamų verčių skaičius, x – tikroji vertė, $\hat{x}$ – nuspėjama vertė. Šio metodo trūkumas – kalbant apie garso atkūrimą triukšmingoje aplinkoje, jis naudojimas išmoko modelį visų galimų signalų vidurkio, nepasirinkdamas nė vieno konkretaus [29].

MSE vidurkio išmokymo problema dažnai sprendžiama pasitelkiant GAN architektūros tinklus, tačiau tyrime jie netaikomi dėl nestabilumo [44].

Garso atkūrimo triukšmingoje aplinkoje problemai spręsti buvo pasiūlyta specializuota klaidos funkcija – *weightedSDR* [52] (2.9). Jos tikslas – modeliuoti garso atkūrimo tikslumą triukšmo sąlygomis:

$$\alpha = \frac{\|y\|^2}{\|y\|^2 + \|z\|^2} \quad (2.6)$$

$$\hat{z} = x - \hat{y} \quad (2.7)$$

$$E_{SDR}(y, \hat{y}) = -\frac{\langle y, \hat{y} \rangle}{\|y\| \|\hat{y}\|} \quad (2.8)$$

$$E_{wSDR} = \alpha E_{SDR}(y, \hat{y}) + (1 - \alpha) E_{SDR}(z, \hat{z}), \quad (2.9)$$

čia x – triukšmingas įėjimo signalas, y – švarus signalas, $\hat{y}$ – modelio nuspėjamas švarus signalas, z – triukšmas. Tik paskelbę apie šį metodą, autoriai pastebėjo, kad mokymo metu jų tinklas buvo mokytas naudojant ir duomenis, skirtus tik testavimui. Mano žiniomis, šis metodas daugiau jokiuose straipsniuose nebuvo minėtas ir yra tyrinėtinas.

2.4.2. Mokymo algoritmas

Žinant tinklo spėjimo klaidą, ja remiantis galima pakeisti neuroninio tinklo sąsūkų ir kitus svorius taip, kad ši klaida būtų minimizuojama. Paprasčiausias yra gradiento nuolydžio algoritmas, kuris apskaičiuoja viso duomenų rinkinio klaidą ir pagal ją pakeičia modelio parametrus. Tačiau didėjant mokymo duomenų rinkiniams pradėti naudoti stochastinio gradiento nuolydžio ir kiti algoritmai. Taikant šiuos algoritmus, tinklo spėjimų klaida skaičiuojama ne visam duomenų rinkiniui, o duomenų partijai (angl. *batch*). Partijos dydis parenkamas toks, kad tilptų procesorinio įrenginio atmintyje [53]. Stochastinis gradiento nuolydžio algoritmas aprašomas taip:

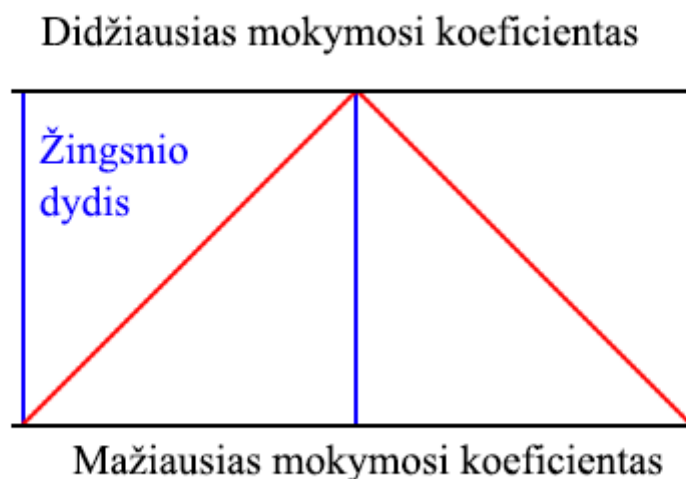
$$\theta^t = \theta^{t-1} - \alpha_t \frac{\partial L}{\partial \theta}, \quad (2.10)$$

čia α – mokymosi koeficientas, L – klaidos funkcijos vertė, θ – modelio parametras.

Garso atkūrimo triukšmingoje aplinkoje problemoms spręsti dažniausiai taikomi patobulinti stochastinio nuolydžio algoritmai – *RMSprop* [54] ir *AdamW* [55].

RMSprop – alternatyvus momentui, stochastinis algoritmas. Jame kiekvienas parametras turi savo adaptyvų mokymosi koeficientą (angl. *learning rate*), siekiant paspartinti tinklo mokymą. Algoritmo esmė – mokymosi koeficientas dalijamas iš parametro vertės, remiantis paskutinių šio parametro gradiento amplitude.

AdamW – adaptyvus gradiento nuolydžiu grįstas algoritmas, skirtas stochastinių tikslo funkcijų optimizacijai. Kaip ir *RMSprop*, algoritmas kiekvienam modelio parametrai sukuria atskirą mokymosi koeficientą ir taip įgalina greitesnį tinklo mokymąsi. Kartu su adaptyviu kiekvieno parametro mokymosi koeficientu, *AdamW* algoritmas kiekvienos partijos mokymosi metu, sukelia parametru nykimą (angl. *weight decay*). Nykimo dydis – tyrimo metu nustatomas parametras, leidžiantis tinklui nepersimokyti (angl. *overfit*).



2.9 pav. Mokymosi koeficiento kitimas, taikant vieno ciklo metodą. Raudonai pažymėtas mokymosi koeficientas

Kadangi abu aprašyti algoritmai remiasi stochastiniu nuolydžiu, 2.9 formulėje aprašytas α , mokymosi koeficientas tyrimo metu turi būti parenkamas tyrėjo. Jeigu jis parenkamas per mažas, tinklas gali nepasiekti mažiausios klaidos vertės. Jeigu per didelis – tinklas gali taip pat niekada nekonverguoti. Šiai problemai spręsti prieš vykdant naują eksperimentą, vieno ciklo metodu parenkamas mokymosi koeficientas [56]. Taip pat vieno ciklo metodas taikomas ne tik mokymosi koeficientui parinkti, tačiau ir jam keisti mokymo metu. Tai įgyvendinama iš pradžių didinant ir pasiekus tam tikrą iteracijų kiekį, mažinant mokymosi koeficientą.

2.5. Skyriaus apibendrinimas

Skyriuje aprašyti ir pagrįsti tiriami autoenkoderio sąsūkų neuroninio tinklo parametrai. Visų pirma aprašytas duomenų rinkinys, naudotinas tyrime, jo savybės bei taikytos transformacijos. Paskui aprašomi tyrinėjami enkoderio filtrai, naudojami signalui glaudenti, bei kiti jų parametrai. Enkoderio filtravimo metu modelis turėtų išmokti atpažinti triukšmo ar švaraus signalo fragmentus ir juos suglaudenti. Taip pat pristatytas galimas suglaudento signalo filtravimo metodas savidėmesingumo moduliais. Jie, tikėtina, įgalina modelį filtruoti visą signalą, o ne tik jo fragmentus. Suglaudintą ir filtruotą signalą reikia atkurti. Tam galima pasitelkti dažnai naudojamus transponuotųjų sąsūkų filtrus. Kaip alternatyva tyrime naudojamosi paprastais sąsūkų filtrais, prieš tai išplėtus signalą.

Sumodeliavus signalą svarbu, kad jis būtų kuo švaresnis. Siekiant šio tikslo analizuojamos galimos klaidos funkcijos – MSE, *WeightedSDR*. Remiantis šių klaidos funkcijų vertėmis, galima keisti modelį, jas minimizuojant. Aprašomi šiai optimizacijai atlikti galimi algoritmai – *RMSprop* ir *AdamW*. Kartu su šiais algoritmais tyrimuose taikomas vieno ciklo metodas.

3. GARSO ATKŪRIMO ALGORITMO TYRIMAS IR REZULTATAI

3.1. Duomenų rinkinys

Tyrimui naudojamas 2.2 aprašytas duomenų rinkinys. Rinkinys turi aiškiai atskirtas duomenų dalis, kurios gali būti naudojamos algoritmo mokymui ir su kuriomis derėtų testuoti algoritmą. Testavimo rinkinio triukšmai ir kalbėtojai į mokymo algoritmą nėra įtraukti. Taip testuojamas algoritmo gebėjimas išmokyti abstrakčias garso ypatybes, o ne tik atkartoti mokymo duomenis.

Mokymo metu mokymo duomenų rinkinys paskirstomas į dvi dalis – mokymo ir patikrinimo. Mokymo dalis naudojama sąsūkos neuroniniam tinklui mokyti, patikros dalis naudojama mokymo metu stebėti ar modelis nepersimoko. Patikrinimo duomenų rinkinys sudaromas prieš kiekvieną eksperimentą: atsitiktine tvarka iš mokymo rinkinio išrenkama 10 % garso failų.

3.2. Vertinimo kriterijai

Neuroninių tinklų parametrų optimizacijai taikomos klaidos funkcijos, tačiau jų tikslas – nuspėti, ar gerai tinklas atkūrė signalą. Nors klaidos funkcijos ir turi sąryšį su garso kokybe, jos negeba iki galo jos modeliuoti. Dažnai tyrimuose taikomas MOS metodas. Apklausiami grupė žmonių, jiems pateikiamas atkurtas ir tikrasis švarūs garsai ir prašoma įvertinti atkurto garso kokybę pagal penkiabalę skalę. Nors šis metodas yra vienas tiksliausių, jis yra ir pats lėčiausias. Norint įgyvendinti daugelį eksperimentų, nuolat klausinėti žmonių neišeina.

Triukšmingoje aplinkoje atkurto garso kokybei matuoti buvo sukurti algoritmai, gebantys aproksimuoti žmonių garso kokybės suvokimą. Tačiau šie algoritmai netinka stochastinei parametrų optimizacijai, nes yra lėti ir nediferencijuojami [57]. Atliekant tyrimą remtasi PESQ (angl. *Perceptual Evaluation of Speech Quality*) [58] ir STOI (angl. *short-time objective intelligibility*) [59] objektyviais vertinimo kriterijais. Jie skaičiuojami naudojant testavimo duomenų rinkinį po algoritmo mokymo naudojant klaidos funkciją ir mokymo duomenų rinkinį.

PESQ – ITU (angl. *International Telecommunication Union*) organizacijos rekomenduojamas algoritmas garso iškraipymams įvertinti. Tyrime taikomas FR (angl. *full reference*) algoritmo variantas, vertinantis švaraus ir iš triukšmingo signalo atkurto signalo skirtumus. Atliekant vertinimą, remiantis kūrėjų rekomendacija, signalas perkvantuojamas į 8 kHz. Šios algoritmo galimos vertės svyruoja nuo -0,5 (blogai atkurtas signalas) iki 4,5 (tobulai atkurtas signalas).

STOI – objektyviai apskaičiuotinas kalbos signalų atkūrimo įvertis, skirtas aukštesnio dažnio signalams. Šis įvertis stipriai koreliuoja ($\rho = 0,95$) su subjektyviu supratimu. Jo galimos vertės svyruoja nuo 0 (nesuprantama kalba) iki 1 (tobulai suprantama kalba).

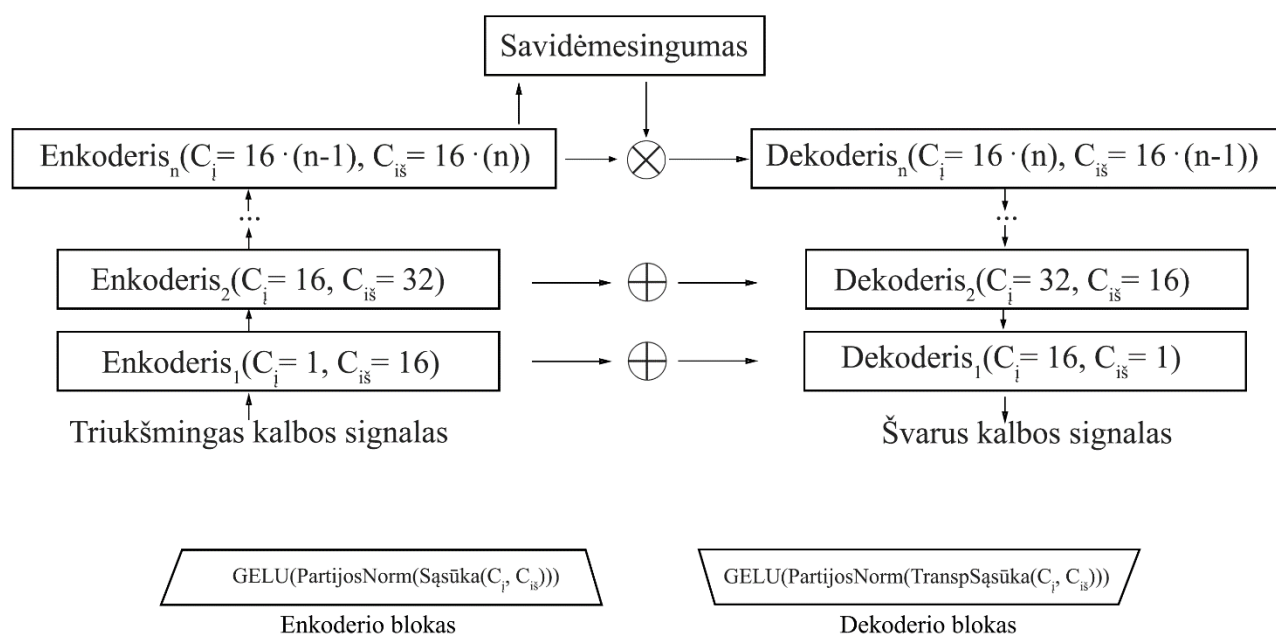
Tyrimo metu, naudojant testinį duomenų rinkinį, kurio modelis mokymo metu nemato, skaičiuojami PESQ ir STOI kriterijų aritmetiniai vidurkliai ir standartinis nuokrypis.

3.3. Tirti parametrai

Šiame poskyryje aprašomi vykdyti eksperimentai ir jų eiga. Prieš vykdant kiekvieną eksperimentą, remiantis 2.4.2, nustatomas optimalus mokymosi koeficientas. Partijos dydis parenkamas taip, kad užpildytų visą GPU atmintį. Tiriant atskirus parametrus daroma prielaida, kad jie nesusiję. Ištyrus vieną parametą, kituose tyrimuose naudojamas geriausias praėjusio eksperimento rezultatas.

Tiriami parametrai:

- Autoenkoderio gylis;
- Enkoderio ir dekoderio filtrų kiekis;
- Savidėmesingumo filtrai ir jų kiekis;
- Dekoderio filtrų tipas;
- Klaidos funkcija;
- Mokymo algoritmas.



3.1 pav. Atliekant tyrimą naudotas modelis

3.3.1. Modelio sluoksnių kiekis

Pasak bendrosios aproksimavimo teoremos, minimalus sluoksnių kiekis, reikalingas bet kokiai funkcijai aproksimuoti – du. Tačiau literatūroje, skirtoje garso atkūrimui triukšmingoje aplinkoje, minimi enkoderiai iki vienuolikos sluoksnių. Kadangi tyrime kuriamas simetriškas autoenkoderis – vienuolikos sluoksnių enkoderis reiškia dvidešimt dviejų sluoksnių modelį. Atliekant tyrimą nagrinėjamas pats mažiausias įmanomas – dviejų sluoksnių autoenkoderis, didžiausias naudojamas

literatūroje – dvidešimt dviejų sluoksnių autoenkoderis ir vidurinės jų gylio vertės – dvylikos sluoksnių autoenkoderis.

Tyrimo parametrai:

- Modelio gylis – 2, 12 ir 22;
- Sąsūkos filtrai – 32 verčių, poslinkis – 16;
- Aktyvacijos funkcijos – GELU;
- Savidėmesingumo filtrai netaikomi;
- Dekoderyje naudojami transponuotųjų sąsūkų filtrai;
- Naudojama partijos normalizacija;
- Enkoderio duomenys perduodami dekoderiui sudedant atitinkamų sluoksnių elementus;
- Klaidos funkcija – MSE;
- Mokymo algoritmas – *AdamW*, nykimo parametras – 0.1;
- Mokymas stabdomas klaidos funkcijai nustojus mažėti;
- 2 sluoksnių tinklo mokymo metu taikyti mokymo koeficientai tarp 0,03 ir 0,01;
- 12 sluoksnių tinklo mokymo metu taikyti mokymo koeficientai tarp 0,01 ir 0,03;
- 22 sluoksnių tinklo mokymo metu taikyti mokymo koeficientai tarp 0,001 ir 0,003.

3.1 lentelė. Modelio sluoksnių kiekio tyrimo rezultatai

Sluosnių kiekis	Parametrų kiekis	PESQ	STOI
2	1,058	2,85 ± 0,43	0,899 ± 0,064
12	1,173,986	3,34 ± 0,44	0,925 ± 0,06
22	7,213,858	3,15 ± 0,443	0,921 ± 0,06

Dviejų sluoksnių modelis per 15 epochų pasiekė 0,00032 klaidos funkcijos vertę, kuri toliau nekito. Dvylikos sluoksnių modelis per 46 epochas pasiekė beveik 50 % mažesnę mokymo klaidą – 0,000171. Giliausias, dvidešimt dviejų sluoksnių, modelis pasiekė 0,0003 mokymo klaidą per 53 epochas.

Pastebėtina, kad aukščiausius kriterijus pasiekė dvylikos sluoksnių tinklas, turintis ~7 kartus mažiau parametrų negu giliausias – dvidešimt dviejų sluoksnių tinklas. Remiantis šiuo rezultatu, tolimesniuose tyrimuose naudojamas dvylikos sluoksnių neuroninis tinklas.

3.3.2. Enkoderio filtrai

Enkoderio filtrų dydis daro įtaką modelio jutimui laukui. Didėjant jutimui laukui, modeliui gali daryti vis sunkiau išmokti reikiamas filtrų vertes. Dažniausiai literatūroje naudojami 32 verčių sąsūkų filtrai. Šiame tyrime jie ir tiriami. Taip pat tiriami ir dvigubai didesni – 64 verčių sąsūkų filtrai ir dvigubai mažesni – 16 verčių sąsūkų filtrai. Visų tyrimų metu naudojama pusės sąsūkos filtrų verčių ilgio poslinkis (angl. *stride*).

Tyrimo parametrai:

- Modelio gylis – 12;
- Sąsūkos filtrai – 64 verčių (poslinkis 32), 32 verčių (poslinkis – 16), 16 verčių (poslinkis 8);
- Aktyvacijos funkcijos – GELU;
- Savidėmesingumo filtrai netaikomi;
- Dekoderyje naudojami transponuotųjų sąsūkų filtrai;
- Naudojama partijos normalizacija;
- Enkoderio duomenys perduodami dekoderiui sudedant atitinkamų sluoksnių elementus;
- Klaidos funkcija – MSE;
- Mokymo algoritmas – *AdamW*, nykimo parametras – 0,1;
- Mokymas stabdomas klaidos funkcijai nustojus mažėti;
- 64 verčių sąsūkos filtrų tinklo mokymo metu taikyti mokymo koeficientai tarp 0,04 ir 0,08;
- 32 verčių sąsūkos filtrų tinklo mokymo metu taikyti mokymo koeficientai tarp 0,01 ir 0,03;
- 16 verčių sąsūkos filtrų tinklo mokymo metu taikyti mokymo koeficientai tarp 0,02 ir 0,04.

3.2 lentelė. Enkoderio filtrų dydžio tyrimo rezultatai

Filtrų dydis	Parametrų kiekis	PESQ	STOI
16	575,106	3,12 ± 0,41	0,91 ± 0,07
32	1,173,986	3,34 ± 0,44	0,925 ± 0,06
64	2,296,962	3,10 ± 0,50	0,89 ± 0,08

Tiriant filtrų dydį 16 ir 64 verčių dydžio sąsūkomis grįsti modeliai per 30 ir 60 epochų pasiekė 0.003 klaidą, kuri toliau nekito. Geriausius rezultatus rodė 32 dviejų verčių filtras, kuris ir pasirenkamas tolimesniam tyrimui.

3.3.3. Savidėmesingumo moduliai

Dėmesingumo modulių esmė – filtruoti signalą naudojant viso jo dydžio jutiminio lauko filtrais. Šis filtravimo metodas įgalina filtrus modeliuoti signalą taip, kad pradžioje matomas triukšmas gali daryti įtaką signalo pabaigoje esančiam triukšmui ir atvirkščiai. Tačiau savidėmesingumo filtrai reikalauja itin didelio parametrų kiekio. Tyrinėjami vieną ir aštuonis savidėmesingumo filtrus turintys modeliai. Jie lyginami su tokiu pat, tik dėmesingumo filtrų neturinčiu modeliu.

Tyrimo parametrai:

- Modelio gylis – 12;
- Sąsūkos filtrai – 32 verčių, poslinkis – 16;
- Aktyvacijos funkcijos – GELU;
- Savidėmesingumo filtrų kiekis 1 ir 8;
- Dekoderyje naudojami transponuotųjų sąsūkų filtrai;
- Naudojama partijos normalizacija;
- Enkoderio duomenys perduodami dekoderiui sudedant atitinkamų sluoksnių elementus;
- Klaidos funkcija – MSE;
- Mokymo algoritmas – *AdamW*, nykimo parametras – 0,1;
- Mokymas stabdomas klaidos funkcijai nustojus mažėti;
- 1 savidėmesingumo filtrą turinčio modelio mokymo metu taikyti mokymo koeficientai tarp 0,01 ir 0,001;
- 8 savidėmesingumo filtrą turinčio modelio mokymo metu taikyti mokymo koeficientai tarp 0,003 ir 0,0001.

3.3 lentelė. Savidėmesingumo filtrų kiekio tyrimo rezultatai

Savidėm. filtrų kiekis	Parametrų kiekis	PESQ	STOI
0	1,173,986	3,34 ± 0,44	0,925 ± 0,06
1	1,176,992	3,21 ± 0,45	0,921 ± 0,06
8	1,446,368	3,21 ± 0,45	0,924 ± 0,06

Tiriant optimalų savidėmesingumo filtrų įtaką triukšmingo signalo modeliavimui pastebėta, kad jų naudojimas nežymiai, tačiau suprastina neuroninio tinklo modeliavimo galimybes. Taip pat

savidėmesingumo filtrai sulėtina modelio mokymą ir padidina modelio užimamos atminties kiekį. Remiantis šiais rezultatais, tolimesniuose tyrimuose savidėmesingumo filtrai nenaudojami.

3.3.4. Dekoderio parametrai

Dekoderio tikslas – iš enkoderio suglaudinto signalo atkurti švarų garso signalą. Tam dažniausiai naudojamos transponuotosios sąsūkos, tačiau kompiuterinės regos literatūroje jos dažnai keičiamos įprastomis sąsūkomis, prieš tai padidinus signalą. Transponuotųjų sąsūkų pakeitimas įprastomis įgalina modelį lengviau atkurti švarų signalą, sintezavimo metu nesukuriant muzikinio triukšmo.

Neuroninių tinklų gyliui didėjant, gradientai gali išnykti, taip nepasiekiant enkoderio. Gradientų nykimo problemai naudojamos praleidimo jungtys (angl. *skip-connections*). Jų tikslas – perduoti informaciją iš enkoderio į dekoderį ir klaidas iš dekoderio į enkoderį. Tai įgalina kompleksiškesnį signalų modeliavimą ir užkerta kelią gradientų nykimui.

Tiriant dekoderio parametrus naudojamos skirtingų tipų sąsūkos. Taip pat tiriami skirtingi praleidimo jungčių metodai – elementų sudėtis, kanalų sudėtis ir jokių praleidimo jungčių.

Tyrimo parametrai:

- Modelio gylis – 22;
- Sąsūkos filtrai – 32 verčių, poslinkis – 16;
- Aktyvacijos funkcijos – GELU;
- Savidėmesingumo filtrai netaikomi;
- Dekoderyje naudojami transponuotųjų ir įprastų sąsūkų filtrai;
- Naudojama partijos normalizacija;
- Informacija iš enkoderio, dekoderiui perduodama:
 - a. Neperduodama;
 - b. Sudedant atitinkamų signalų kanalus;
 - c. Sudedant atitinkamų signalų elementus.
- Klaidos funkcija – MSE;
- Mokymo algoritmas – *AdamW*, nykimo parametras – 0,1;
- Mokymas stabdomas klaidos funkcijai nustojus mažėti;
- Visų tyrimų metu taikyti mokymosi koeficientai tarp 0,01 ir 0,03.

3.4 lentelė. Dekoderio parametrų tyrimo rezultatai

Sujungimas	Dekoderio filtrų tipas	Parametrų kiekis	PESQ	STOI
Nėra	Transponuotosios sąsūkos	1,173,986	$2,55 \pm 0,19$	$0,773 \pm 0,056$
Nėra	Didinimas ir sąsūkos	1,149,058	$2,48 \pm 0,23$	$0,762 \pm 0,061$
Kanalų sudėtis	Transponuotosios sąsūkos	2,870,370	$3,10 \pm 0,43$	$0,920 \pm 0,062$
Kanalų sudėtis	Didinimas ir sąsūkos	2,377,858	$3,01 \pm 0,53$	$0,90 \pm 0,053$
Elementų sudėtis	Transponuotosios sąsūkos	1,173,986	$3,34 \pm 0,44$	$0,925 \pm 0,06$
Elementų sudėtis	Didinimas ir sąsūkos	1,025,698	$3,13 \pm 0,41$	$0,901 \pm 0,057$

Jokių praleidimo jungčių neturintys modeliai pasiekė prasčiausias metrikų vertes. Galimai, tai nutiko dėl gradientų nykimo. Klaidos funkcijos vertės, nenaudojant praleidimo jungčių, visuomet buvo bent dvigubai mažesnės, lyginant su praleidimo jungtis turinčiais modeliais. Kanalų sudėties metodas įgalino lengvesnį gradientų sklaidimą modelyje, tačiau, nors ir turintys daugiau parametrų, šie modeliai nepagerino elementų sudėtį naudojančių modelių rezultatų. Transponuotųjų sąsūkų pakeitimas signalo išplėtimu ir įprastinėmis sąsūkomis nepagerino rezultatų.

3.3.5. Normalizacijos

Normalizacijos tikslas paskirstyti tarp modelio sluoksnių esančias ypatybes. Šiame tyrime tyrinėjami du normalizacijos metodai – partijos ir sluoksnio normalizacijos.

Tyrimo parametrai:

- Modelio gylis – 12;
- Sąsūkos filtrai – 32 verčių, poslinkis – 16;
- Aktyvacijos funkcijos – GELU;
- Savidėmesingumo filtrai netaikomi;
- Dekoderyje naudojami transponuotųjų sąsūkų filtrai;
- Naudojamos partijos ir sluoksnių normalizacijos;
- Klaidos funkcija – MSE;
- Mokymo algoritmas – *AdamW*, nykimo parametras – 0,1;
- Mokymas stabdomas klaidos funkcijai nustojus mažėti;
- Visų tyrimų metu taikyti mokymo koeficientai tarp 0,01 ir 0,03.

3.5 lentelė. Normalizacijos tyrimo rezultatai

Klaidos funkcija	PESQ	STOI
Partijos	3,34 ± 0,44	0,925 ± 0,06
Sluoksnių	3,23 ± 0,48	0,916 ± 0,067

Tyrimo metu per 30 epochų sluoksnių normalizaciją naudojantis modelis pasiekė 0,0002 klaidos funkcijos vertę. Sluoksnių normalizaciją naudojantis modelis nežymiai, tačiau prasčiau, geba modeliuoti triukšmingų garsų atkūrimą.

3.3.6. Aktyvavimo funkcijos

Aktyvacijos funkcijos įgalina neuroninius tinklus modeliuoti netiesines funkcijas. Šios funkcijos transformuoja kiekvieno sluoksnio ypatybes. Tačiau skirtingoms problemoms spręsti, naudojamos skirtingos funkcijos. Šio tyrimo metu tiriama kompiuterinės regos literatūroje dažnai naudojama ReLU funkcija, garso atkūrimo triukšmingoje aplinkoje literatūroje naudojama PReLU funkcija ir daug žadančius rezultatus rodanti, tačiau garso atkūrimo problemai spręsti nenaudota – GELU.

Tyrimo parametrai:

- Modelio gylis – 12;
- Sąsūkos filtrai – 32 verčių, poslinkis – 16;
- Aktyvacijos funkcija – ReLU, PReLU ir GELU;
- Savidėmesingumo filtrai netaikomi;
- Dekoderyje naudojami transponuotųjų sąsūkų filtrai;
- Klaidos funkcija – MSE;
- Mokymo algoritmas – *AdamW*, nykimo parametras – 0,01;
- Mokymas stabdomas klaidos funkcijai nustojus mažėti;
- Tyrimų metu taikyti mokymo koeficientai tarp 0,01 ir 0,03.

3.6 lentelė. Aktyvavimo funkcijų tyrimo rezultatai

Mokymo algoritmas	PESQ	STOI
ReLU	2,81 ± 0,4	0,90 ± 0,064
PReLU	3,15 ± 0,43	0,92 ± 0,43
GELU	3,34 ± 0,44	0,925 ± 0,06

Modelis, naudojantis ReLU aktyvavimo funkciją, per 60 epochų, pasiekė prasčiausią klaidos funkcijos (0,00093) ir objektyvių įverčių vertes. Galima to priežastis – ReLU pašalina neigiamas vertes, taip sunaikinant dalį atkuriamo signalo. PReLU aktyvavimo funkcija, nenaikinant neigiamų verčių, per 63 epochas pasiekė 0,00023 klaidos funkcijos vertę. Geriausių rezultatų pasiekė ankstesniuose tyrimuose naudota GELU aktyvavimo funkcija.

3.3.7. Klaidos funkcijos

Klaidos funkcija – esminis neuroniniais tinklais grįstų modelių elementas. Jos tikslas apskaičiuoti skirtumą tarp modelio nuspėjamo ir tikro signalų. Naudojant šią klaidos vertę, atnaujinami visi neuroninio tinklo parametrai. Tiriama dažniausiai tyrimuose naudojama – MSE funkcija, kuri yra labai paprasta. Taip pat tiriama ir nedaug tyrinėta, būtent garso atkūrimo triukšmingoje aplinkoje problemai spręsti, funkcija *weightedSDR*.

Tyrimo parametrai:

- Modelio gylis – 2, 12 ir 22;
- Sąsūkos filtrai – 32 verčių, poslinkis – 16;
- Savidėmesingumo filtrai netaikomi;
- Dekoderyje naudojami transponuotųjų sąsūkų filtrai;
- Klaidos funkcija – MSE, *weightedSDR*;
- Mokymo algoritmas – *AdamW*, nykimo parametras – 0,1;
- Mokymas stabdomas klaidos funkcijai nustojus mažėti;
- Tyrimų metu taikyti mokymo koeficientai tarp 0,01 ir 0,03.

3.7 lentelė. Klaidos funkcijų tyrimo rezultatai

Klaidos funkcija	PESQ	STOI
MSE	3,34 ± 0,44	0,925 ± 0,06
WeightedSDR	3,25 ± 0,44	0,92 ± 0,06

Abiejų klaidos funkcijų rezultatai stipriai nesiskyrė. Tačiau *weightedSDR* klaidos funkcija reikalauja daugiau skaičiavimų negu MSE. Remiantis šiuo rezultatu kituose tyrimuose minimizuojama MSE klaidos funkcija.

3.3.8. Mokymo algoritmai

Mokymo algoritmo tikslas – remiantis klaidos funkcijos apskaičiuota verte, pakeisti modelio parametrus taip, kad klaidos funkcija būtų kuo mažesnė. Šie algoritmai nepriklauso nuo sprendžiamos problemos. Tyrime naudojamas *AdamW* mokymo algoritmas, gebantis mokyti kiekvieną parametą naudojant skirtingą mokymo koeficientą. Taip pat tiriamas, garso atkūrimo literatūroje dažnai naudojamas, *RMSProp* mokymo algoritmas.

Tyrimo parametrai:

- Modelio gylis – 12;
- Sąsūkos filtrai – 32 verčių, poslinkis – 16;
- Savidėmesingumo filtrai netaikomi;
- Dekoderyje naudojami transponuotųjų sąsūkų filtrai;
- Klaidos funkcija – MSE;
- Mokymo algoritmas – *AdamW*, nykimo parametras – 0,1, *RMSProp*;
- Mokymas stabdomas klaidos funkcijai nustojus mažėti;
- Tyrimų metu taikyti mokymo koeficientai tarp 0,01 ir 0,03.

3.8 lentelė. Mokymo algoritmo tyrimo rezultatai

Mokymo algoritmas	PESQ	STOI
<i>AdamW</i>	3,34 ± 0,44	0,925 ± 0,06
<i>RMSProp</i>	3,19 ± 0,49	0,923 ± 0,063

RMSProp mokymo algoritmas nepagerino ankstesniuose tyrimuose gautų rezultatų. Tačiau STOI įvertis yra *AdamW* algoritmą naudojant gautų rezultatų klaidos ribose. *RMSProp* naudojantis modelis greičiau konvergavo, per 38 epochas pasiekdamas 0,000182 klaidos funkcijos vertę. Kadangi šie algoritmai įgalina labai panašius rezultatus, tinkamas mokymo algoritmas reikalauja išsamesnių tyrimų.

3.3.9. Optimalūs algoritmo parametrai

Tyrimo metu rasti parametrai, maksimizuojantys duomenų rinkinio objektyvių įverčių (PESQ ir STOI) vertes:

- Modelio gylis – 12;
- Sąsūkos filtrai – 32 verčių, poslinkis – 16;
- Netaikyti savidėmesingumo filtrai;
- Taikyta partijos normalizacija;
- Dekoderyje taikyti transponuotųjų sąsūkų filtrai;
- Dekoderio ir enkoderio duomenų sujungimas sudedant atitinkamus signalų elementus;
- Klaidos funkcija – MSE;
- Mokymo algoritmas – *AdamW*, nykimo parametras – 0,1;
- Mokymo koeficientas tarp 0,01 ir 0,03.

3.4. Rezultatų palyginimas

Rezultatų palyginimui naudojamas geriausias rezultatus pasiekęs tyrimo rezultatas. Jis lyginamas su rekurentiniais sluoksniais ir sąsūkomis grįžtais neuroniniais tinklais. Kartu su algoritmais, pateikiami objektyvūs, triukšmingo algoritmo, įverčiai.

3.9 lentelė. Tyrime įgyvendinto ir kitų algoritmų palyginimo rezultatai

Algoritmas	PESQ	STOI
Triukšmingas signalas	$2,95 \pm 0,43$	$0,90 \pm 0,1$
RNNoise	$2,78 \pm 0,78$	$0,89 \pm 0,2$
SEGAN	$2,27 \pm 0,56$	$0,87 \pm 0,3$
Šis tyrimas	$3,34 \pm 0,44$	$0,925 \pm 0,06$

Pastebėtina, kad tyrimo metu pasiekti rezultatai, naudojant pasirinktą duomenų rinkinį, – geresni negu kitų algoritmų. Tai geriausias šiuo metu esantis rezultatas (angl. *state-of-the-art*).

3.5. Trumpas skyriaus apibendrinimas

Tirti esminiai sąsūkomis grįžtų neuroninių tinklų parametrai – modelio gylis, sąsūkos filtrų dydis, savidėmesingumo filtrai, transponuotųjų sąsūkų pakeitimas įprastomis, enkoderio ir dekoderio duomenų sujungimo metodai, skirtingos aktyvavimo funkcijos, normalizavimo metodai, klaidos funkcijos ir mokymo algoritmai. Stipriausiai modelio tikslumui daro įtaką tinkamas modelio gylis, sąsūkos filtrų verčių kiekis ir tinkamas aktyvavimo funkcijos parinkimas.

APIBENDRINIMAS

Atlikta garso atkūrimo triukšmingoje aplinkoje algoritmų analizė. Apžvelgti patys seniausi ir paprasčiausi – statistiniai ir naujesni – neuroniniais tinklais grįsti metodai. Aptarti abiejų metodų skirtumai, prielaidos, panaudojimo galimybės bei kitos ypatybės.

Remiantis apžvelgtais algoritmais – įgyvendintas ir aprašytas naujas algoritmas, skirtas garso atkūrimo triukšmingoje aplinkoje problemai spręsti. Algoritmo įgyvendinimui naudotas modelis – sąsūkomis grįstas autoenkoderio tipo neuroninis tinklas. Modelio optimizacijai naudotas triukšmingos ir švarios kalbos duomenų rinkinys [10]. Algoritmas įgyvendinamas naudojant dažniausiai apžvelgtoje literatūroje naudojamais elementais – sąsūkomis, transponuotosiomis sąsūkomis, aktyvacijos ir normalizacijos funkcijomis.

Garso atkūrimo triukšmingoje aplinkoje tikslumui vertinti pasirinkti du objektyvūs įverčiai – PESQ ir STOI. Atliekant eksperimentus parenkama didžiausias objektyvias vertes pasiekianti modelio konfigūracija.

Atliktas tyrimas nagrinėja skirtingų neuroninio tinklo konfigūracijos įtaką minėtiems objektyviems įverčiams. Nagrinėti parametrai:

- Modelio gylis: tirti 2, 12 ir 22 sluoksnių konfigūracijos;
- Enkoderio sąsūkos filtrų dydis, jų poslinkis: tirti 16, 32 ir 64 filtrų verčių sąsūkų filtrai;
- Savidėmesingumo filtrai (angl. *self-attention*): tirti 1 ir 8 savidėmesingumo filtrus turintys modeliai;
- Transponuotųjų sąsūkų bei įprastų sąsūkų taikymas signalo atkūrimui;
- Enkoderio duomenų perdavimas dekoderiui: tirtas duomenų sujungimas sudedant atitinkamų signalų elementus, jų kanalus, lyginta su jokio duomenų sujungimo neturinčiais modeliais;
- Aktyvavimo funkcijos: tirtos ReLU, PReLU ir GELU funkcijos;
- Normalizacijos metodai: tirtos partijos ir sluoksnių normalizacijos;
- Klaidos funkcijos: tirtos MSE ir *weightedSDR* funkcijos;
- Mokymo algoritmas: tirti *AdamW* ir *RMSProp* algoritmai;

Iš viso atlikta 16 eksperimentų, kurie vidutiniškai truko 3 valandas. Šių eksperimentų metu atrasta modelio konfigūracija, maksimizuojanti duomenų rinkinio objektyvius įverčius. Didžiausią tikslumą modeliui suteikia jo gylis, tačiau per gilūs modeliai negeba optimaliai apsimokyti. Eksperimentų metu geriausius rezultatus įgalino dvylikos sluoksnių neuroninis tinklas. Optimalus enkoderio sąsūkų filtrų dydis 16 kHz garso signalams – 32 vertės. Savidėmesingumo filtrų naudojimas, nors ir padidinant parametų kiekį, nepadėjo modeliui geriau atkurti triukšmingų

signalų. Efektyviausia naudota normalizacija – partijos. Geriausius rezultatus leidusi pasiekti aktyvavimo funkcija – GELU. Tyrinėtas transponuotųjų sąsūkų pakeitimas įprastomis sąsūkomis žymaus pagerėjimo nesuteikė. Geriausių rezultatų pasiekė modelis, kurio enkoderio ir dekoderio signalai sujungiami sudedant atitinkamus elementus. Naudota nauja klaidos funkcija (*weightedSDR*) nepagerino modelio veikimo, geresnius rezultatus įgalino MSE funkcija. Išbandytų mokymo algoritmų – geriausius rezultatus leido pasiekti *AdamW* optimizavimo algoritmas.

Minėtas parametrų rinkinys lyginamas su kitais, neuroniniais tinklais grįžtais, algoritmais – SEGAN [29] ir *RNNoise* [15]. Įgyvendinus algoritmą, remiantis duomenų rinkinio objektyviais kriterijais, pastebėta, kad jis geba geriau modeliuoti garso atkūrimo triukšmingoje aplinkoje problemą.

Tolimesnių tyrimų metu galima būtų nagrinėti kitokių negu balso garso signalų atkūrimą triukšmingoje aplinkoje. Įgyvendintas algoritmas turėtų veikti ir kitiems, laiko ašyje esantiems signalams. Taip pat įgyvendinto modelio architektūra gali įgalinti ir kitų problemų, pavyzdžiui dažnio didinimo ar konkretaus kalbėtojo garso atkūrimo (angl. *speaker dependent speech enhancement*), sprendimą.

Nagrinėjant tolimesnius modelio architektūros patobulinimus derėtų atsižvelgti į dėmesingumo modulių parametrų mažinimą. Šie moduliai turi itin daug parametrų ir tik nežymiai pagerina modelio tikslumą. Turint didesnę kiekį skaičiavimo resursų įmanoma sukurti neuroninio tinklo modelį remiantis vien dėmesingumo moduliais. Šio tyrimo metu tolimesnio jų nagrinėjimo atsisakyta būtent dėl skaičiavimo resursų stokos. Nagrinėtina ir pamatinė – autoenkoderio architektūra. Šiame tyrime ji – simetrinė. Tai įgalina lengvą algoritmo įgyvendinimą, tačiau nėra daug tyrimų, tyrinėjančių asimetrinius autoenkoderius.

LITERATŪRA

- [1] D. Wang ir J. Chen, „Supervised speech separation based on deep learning: An overview“, *IEEE/ACM Transactions on Audio Speech and Language Processing*, t. 26, nr. 10. Institute of Electrical and Electronics Engineers Inc., p. 1702–1726, spal. 01, 2018, doi: 10.1109/TASLP.2018.2842159.
- [2] Y. Xu, J. Du, L. R. Dai, ir C. H. Lee, „An experimental study on speech enhancement based on deep neural networks“, *IEEE Signal Process. Lett.*, t. 21, nr. 1, p. 65–68, saus. 2014, doi: 10.1109/LSP.2013.2291240.
- [3] Y. Isik, J. Le Roux, Z. Chen, S. Watanabe, ir J. R. Hershey, „Single-channel multi-speaker separation using deep clustering“, *Proc. Annu. Conf. Int. Speech Commun. Assoc. INTERSPEECH*, t. 08-12-Sept, p. 545–549, 2016, doi: 10.21437/Interspeech.2016-1176.
- [4] D. Yu, M. Kolbaek, Z. H. Tan, ir J. Jensen, „Permutation invariant training of deep models for speaker-independent multi-talker speech separation“, *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, birž. 2017, p. 241–245, doi: 10.1109/ICASSP.2017.7952154.
- [5] Y. Luo ir N. Mesgarani, „Conv-TasNet: Surpassing Ideal Time-Frequency Magnitude Masking for Speech Separation“, *IEEE/ACM Trans. Audio Speech Lang. Process.*, t. 27, nr. 8, p. 1256–1266, 2019, doi: 10.1109/TASLP.2019.2915167.
- [6] Z. Q. Wang, J. Le Roux, D. L. Wang, ir J. R. Hershey, „End-to-end speech separation with unfolded iterative phase reconstruction“, *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, 2018, t. 2018-September, p. 2708–2712, doi: 10.21437/Interspeech.2018-1629.
- [7] D. W. Griffin ir J. S. Lim, „SIGNAL ESTIMATION FROM MODIFIED SHORT-TIME FOURIER TRANSFORM.“, *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 1983, t. 2, p. 804–807, doi: 10.1109/icassp.1983.1172092.
- [8] J. Le Roux, N. Ono, ir S. Sagayama, „Explicit consistency constraints for STFT spectrograms and their application to phase reconstruction“, 2008. Žiūrėta: bal. 27, 2021. [Interaktyvus]. Available at: <http://www.isca-speech.org/archive>.
- [9] X. Lu, Y. Tsao, S. Matsuda, ir C. Hori, „Speech enhancement based on deep denoising autoencoder“, 2013.
- [10] C. Valentini-Botinhao, X. Wang, S. Takaki, ir J. Yamagishi, „Investigating RNN-based speech enhancement methods for noise-robust Text-to-Speech“, *9th ISCA Speech Synthesis Workshop*, 2016, t. 2016, p. 146–152, doi: 10.21437/ssw.2016-24.

- [11] S. S. Stevens ir J. Volkmann, „The Relation of Pitch to Frequency: A Revised Scale“, *Am. J. Psychol.*, t. 53, nr. 3, p. 329, liep. 1940, doi: 10.2307/1417526.
- [12] G. T. Fechner, „Elements of psychophysics, 1860.“, *Readings in the history of psychology.*, Appleton-Century-Crofts, 2007, p. 206–213.
- [13] J. Andén ir S. Mallat, „Deep scattering spectrum“, *IEEE Trans. Signal Process.*, t. 62, nr. 16, p. 4114–4128, rugpj. 2014, doi: 10.1109/TSP.2014.2326991.
- [14] A. Herrera ir F. Del rio, „Frequency bark cepstral coefficients extraction for speech analysis by synthesis.“, *J. Acoust. Soc. Am.*, t. 128, nr. 4, p. 2290–2290, spal. 2010, doi: 10.1121/1.3508042.
- [15] J. M. Valin, „A hybrid DSP/Deep learning approach to real-time full-band speech enhancement“, lapkr. 2018, doi: 10.1109/MMSP.2018.8547084.
- [16] S. F. Boll, „Suppression of Acoustic Noise in Speech Using Spectral Subtraction“, *IEEE Trans. Acoust.*, t. 27, nr. 2, p. 113–120, 1979, doi: 10.1109/TASSP.1979.1163209.
- [17] J. S. Erkelens ir R. Heusdens, „Tracking of nonstationary noise based on data-driven recursive noise power estimation“, *IEEE Trans. Audio, Speech Lang. Process.*, t. 16, nr. 6, p. 1112–1123, 2008, doi: 10.1109/TASL.2008.2001108.
- [18] R. Martin, „Noise power spectral density estimation based on optimal smoothing and minimum statistics“, *IEEE Trans. Speech Audio Process.*, t. 9, nr. 5, p. 504–512, liep. 2001, doi: 10.1109/89.928915.
- [19] R. C. Hendriks, T. Gerkmann, ir J. Jensen, „DFt-domain based single-microphone noise reduction for speech enhancement: A survey of the state of the art“, *Synth. Lect. Speech Audio Process.*, t. 11, p. 1–80, saus. 2013, doi: 10.2200/S00473ED1V01Y201301SAP011.
- [20] Y. Ephraim ir D. Malah, „Speech enhancement using a minimum mean-square error-log-spectral amplitude estimator“, *IEEE Trans. Acoust.*, t. 33, nr. 2, p. 443–445, 1985, doi: 10.1109/TASSP.1985.1164550.
- [21] J. S. Lim ir A. V. Oppenheim, „All-Pole Modeling of Degraded Speech“, *IEEE Trans. Acoust.*, t. 26, nr. 3, p. 197–210, 1978, doi: 10.1109/TASSP.1978.1163086.
- [22] by Marvin Minsky, S. Papert, ir H. D. Block, „A Review of ,Perceptrons: An Introduction to Computational Geometry““, 1970.
- [23] K. Hornik, M. Stinchcombe, ir H. White, „Multilayer feedforward networks are universal approximators“, *Neural Networks*, t. 2, nr. 5, p. 359–366, saus. 1989, doi: 10.1016/0893-6080(89)90020-8.
- [24] R. Rojas, „Networks of Width One Are Universal Classifiers“, *Proceedings of the International Joint Conference on Neural Networks*, 2003, t. 4, p. 3124–3127, doi: 10.1109/ijcnn.2003.1224071.

- [25] K. He, X. Zhang, S. Ren, ir J. Sun, „Deep residual learning for image recognition“, *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, gruodž. 2016, t. 2016-December, p. 770–778, doi: 10.1109/CVPR.2016.90.
- [26] I. Goodfellow, Y. Bengio, ir A. Courville, *Deep Learning*. 2017.
- [27] S. Hochreiter ir J. Schmidhuber, „Long Short-Term Memory“, *Neural Comput.*, t. 9, nr. 8, p. 1735–1780, lapkr. 1997, doi: 10.1162/neco.1997.9.8.1735.
- [28] K. Cho *et al.*, „Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation“, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, p. 1724–1734, doi: 10.3115/v1/D14-1179.
- [29] S. Pascual, A. Bonafonte, ir J. Serra, „SEGAN: Speech enhancement generative adversarial network“, *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, 2017, t. 2017-Augus, p. 3642–3646, doi: 10.21437/Interspeech.2017-1428.
- [30] A. Odena, V. Dumoulin, ir C. Olah, „Deconvolution and Checkerboard Artifacts“, *Distill*, t. 1, nr. 10, p. e3, saus. 2017, doi: 10.23915/distill.00003.
- [31] J. L. Ba, J. R. Kiros, ir G. E. Hinton, „Layer Normalization“, liep. 21, 2016. <http://arxiv.org/abs/1607.06450> (žiūrėta geg. 06, 2021).
- [32] R. Pascanu, T. Mikolov, ir Y. Bengio, „On the difficulty of training recurrent neural networks“, 2013.
- [33] D. Mahajan *et al.*, „Exploring the Limits of Weakly Supervised Pretraining“, *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, t. 11206 LNCS, p. 185–201, geg. 2018, Žiūrėta: bal. 27, 2021. [Interaktyvus]. Available at: <http://arxiv.org/abs/1805.00932>.
- [34] ITU, „P.56 recommendation: Objective measurement of active speech level“, 2011. <https://www.itu.int/rec/T-REC-P.56-201112-I/en> (žiūrėta bal. 27, 2021).
- [35] K. He, X. Zhang, S. Ren, ir J. Sun, „Delving deep into rectifiers: Surpassing human-level performance on imagenet classification“, 2015. doi: 10.1109/ICCV.2015.123.
- [36] M. Ravanelli ir Y. Bengio, „Speaker Recognition from Raw Waveform with SincNet“, *2018 IEEE Spoken Language Technology Workshop, SLT 2018 - Proceedings*, vas. 2019, p. 1021–1028, doi: 10.1109/SLT.2018.8639585.
- [37] Y. LeCun, L. Bottou, Y. Bengio, ir P. Haffner, „Gradient-based learning applied to document recognition“, 1998. doi: 10.1109/5.726791.
- [38] D. Garcia-Romero, G. Sell, ir A. Mccree, „MagNetO: X-vector Magnitude Estimation Network plus Offset for Improved Speaker Recognition“, geg. 2020, p. 1–8, doi: 10.21437/odyssey.2020-1.

- [39] M. Ravanelli *et al.*, „Multi-Task Self-Supervised Learning for Robust Speech Recognition“, *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, geg. 2020, t. 2020-May, p. 6989–6993, doi: 10.1109/ICASSP40776.2020.9053569.
- [40] S. Ioffe ir C. Szegedy, „Batch normalization: Accelerating deep network training by reducing internal covariate shift“, PMLR, birž. 2015. Žiūrėta: bal. 27, 2021. [Interaktyvus]. Available at: <http://proceedings.mlr.press/v37/ioffe15.html>.
- [41] J. L. Ba, J. R. Kiros, ir G. E. Hinton, „Layer Normalization“, 2016. Žiūrėta: bal. 27, 2021. [Interaktyvus]. Available at: <http://arxiv.org/abs/1607.06450>.
- [42] A. Krizhevsky, I. Sutskever, ir G. E. Hinton, „ImageNet classification with deep convolutional neural networks“, 2017. doi: 10.1145/3065386.
- [43] D. A. Clevert, T. Unterthiner, ir S. Hochreiter, „Fast and accurate deep network learning by exponential linear units (ELUs)“, 2016.
- [44] H. Zhang, I. Goodfellow, D. Metaxas, ir A. Odena, „Self-Attention Generative Adversarial Networks“, PMLR, geg. 2019. Žiūrėta: bal. 27, 2021. [Interaktyvus]. Available at: <https://github.com/>.
- [45] A. Vaswani *et al.*, „Attention is all you need“, *31st Conf. Neural Inf. Process. Syst. NIPS 2017*, 2017.
- [46] K. Xu *et al.*, „Show, attend and tell: Neural image caption generation with visual attention“, *32nd International Conference on Machine Learning, ICML 2015*, 2015, t. 3, p. 2048–2057.
- [47] T. Luong, H. Pham, ir C. D. Manning, „Effective Approaches to Attention-based Neural Machine Translation“, *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015, p. 1412–1421, doi: 10.18653/v1/D15-1166.
- [48] W. Shi *et al.*, „Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network“, *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, gruodž. 2016, t. 2016-December, p. 1874–1883, doi: 10.1109/CVPR.2016.207.
- [49] C. Dong, C. C. Loy, K. He, ir X. Tang, „Image Super-Resolution Using Deep Convolutional Networks“, *IEEE Trans. Pattern Anal. Mach. Intell.*, t. 38, nr. 2, p. 295–307, vas. 2016, doi: 10.1109/TPAMI.2015.2439281.
- [50] N. Mulliqi *et al.*, „The Importance of Skip Connections in Encoder-Decoder Architectures for Colorectal Polyp Detection“, *Proceedings - International Conference on Image Processing, ICIP*, spal. 2020, t. 2020-October, p. 380–384, doi: 10.1109/ICIP40778.2020.9191310.
- [51] Y. le Cun, „A theoretical framework for Back-Propagation“, *Proc. 1988 Connect. Model.*

- Summer Sch.*, p. 21–28, 1988, Žiūrėta: geg. 06, 2021. [Interaktyvus]. Available at: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.28.5453>.
- [52] H. S. Choi, J. H. Kim, J. Huh, A. Kim, J. W. Ha, ir K. Lee, „Phase-aware speech enhancement with Deep Complex U-Net“, 2019. Žiūrėta: geg. 06, 2021. [Interaktyvus]. Available at: <http://kkp15.github.io/DeepComplexUnet>.
- [53] L. Bottou, „Large-scale machine learning with stochastic gradient descent“, *Proceedings of COMPSTAT 2010 - 19th International Conference on Computational Statistics, Keynote, Invited and Contributed Papers*, 2010, p. 177–186, doi: 10.1007/978-3-7908-2604-3_16.
- [54] A. Graves, „Generating Sequences With Recurrent Neural Networks“, 2013. Žiūrėta: geg. 06, 2021. [Interaktyvus]. Available at: <http://arxiv.org/abs/1308.0850>.
- [55] I. Loshchilov ir F. Hutter, „Decoupled weight decay regularization“, *arXiv*, rugs. 27, 2017. <https://github.com/loshchil/AdamW-and-SGDW> (žiūrėta geg. 06, 2021).
- [56] L. N. Smith, „Cyclical learning rates for training neural networks“, *Proceedings - 2017 IEEE Winter Conference on Applications of Computer Vision, WACV 2017*, geg. 2017, p. 464–472, doi: 10.1109/WACV.2017.58.
- [57] S. W. Fu, C. F. Liao, ir Y. Tsao, „Learning with Learned Loss Function: Speech Enhancement with Quality-Net to Improve Perceptual Evaluation of Speech Quality“, *IEEE Signal Process. Lett.*, t. 27, p. 26–30, 2020, doi: 10.1109/LSP.2019.2953810.
- [58] A. W. Rix, J. G. Beerends, M. P. Hollier, ir A. P. Hekstra, „Perceptual evaluation of speech quality (PESQ) - A new method for speech quality assessment of telephone networks and codecs“, *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 2001, t. 2, p. 749–752, doi: 10.1109/icassp.2001.941023.
- [59] C. H. Taal, R. C. Hendriks, R. Heusdens, ir J. Jensen, „An algorithm for intelligibility prediction of time-frequency weighted noisy speech“, *IEEE Trans. Audio, Speech Lang. Process.*, t. 19, nr. 7, p. 2125–2136, 2011, doi: 10.1109/TASL.2011.2114881.
- [60] „dvisockas/norse: NORSE: NOn Recurrent Speech Enhancement“. <https://github.com/dvisockas/norse> (žiūrėta geg. 17, 2021).

PRIEDAI

Prieduose paprastai pateikiama didesnės apimties medžiaga (paveikslai, schemos, charakteristikos, nedideli aprašymai, programos kodai ir t. t.), į kurią aiškinamajame rašte yra nuorodos.

Priedai gali būti segami kartu su aiškinamuoju raštu, jeigu jų apimtis yra sąlyginai nedidelė. Priedus rekomenduojama susegti atskirai, jeigu jų apimtis viršija $\sim 1/3$ aiškinamojo rašto apimties. Pavyzdžiui, jeigu aiškinamajame rašte iki santraukos imtinai yra 50–60 psl. Kai priedai sudaro virš 20 psl., juos patogiau susegti į atskirą segtuvą, kurio viršelis yra analogiškas kaip šios elektroninės formos, tačiau vietoje „Magistro baigiamasis darbas“ rašoma „Magistro baigiamojo darbo priedai“.

Jeigu prieduose pateikiama medžiaga yra įvairi (pvz., integrinių grandynų charakteristikos, nuotraukos, programų kodai), tuomet jie skaidomi į dalis ir numeruojami: 1 priedas, 2 priedas ir pan.

A priedas. Pasiūlyto algoritmo programos išeities kodas

```
import torch
from torch import nn

UPSAMPLE_TYPES = ['transpose', 'conv']
ATTENTION_TYPES = ['SAP', 'ASP']
SKIP_OP_TYPES = ['add', 'cat', 'none']
NORM_TYPES = ['batch', 'layer']

class Autoencoder(nn.Module):
    def __init__(
        self,
        pay_attention = 0,
        depth = 6,
        channels_in = 1,
        channels_out = 1,
        growth_factor = 16,
        kernel_size = 16,
        upsample_type = 'transpose',
        skip_op = 'add',
        norm = 'batch'
    ):
        self.attn = pay_attention
        self.depth = depth
        self.growth_factor = growth_factor

        self.kernel_size = kernel_size
        self.norm = norm
        assert(norm in NORM_TYPES)

        self.upsample_type = upsample_type
        assert(upsample_type in UPSAMPLE_TYPES)

        self.padding_mode = 'reflect'

        assert(skip_op in SKIP_OP_TYPES)
        self.skip_op = skip_op
```

```

super(Autoencoder, self).__init__()
self.bootstrap()

def bootstrap(self):
    self.encoder = nn.ModuleList()
    self.decoder = nn.ModuleList()
    self.attention = nn.ModuleList()

    for index in range(self.depth):
        encoder_ch_in = max(self.growth_factor * (index), 1)
        encoder_ch_out = self.growth_factor * (index + 1)

        if self.norm == 'batch':
            enc_norm = nn.BatchNorm1d(encoder_ch_out)
        else:
            enc_norm = nn.LayerNorm((encoder_ch_out))

        #activation = nn.PReLU()
        #activation = nn.ReLU()
        activation = nn.ReLU()

        stride = self.kernel_size // 2

        encode = []
        encode += [
            nn.Conv1d(
                encoder_ch_in, encoder_ch_out, self.kernel_size,
                stride=stride, padding=15, padding_mode=self.padding_mode, bias=False,
            ),
            nn.BatchNorm1d(encoder_ch_out),
            activation,
        ]

        self.encoder.append(nn.Sequential(*encode))

        channel_multiplier = 2 if (self.skip_op == 'cat' and index != 0)
    else 1

    growth = self.growth_factor * channel_multiplier

    decoder_ch_in = growth * (self.depth-index)
    decoder_ch_out = (growth * (self.depth-index-1))
    if self.skip_op == 'cat' and index != 0: decoder_ch_out =
decoder_ch_out // 2
    decoder_ch_out = max(decoder_ch_out, 1)

    decode = []
    if self.upsample_type == 'transpose':
        decode += [
            nn.ConvTranspose1d(
                decoder_ch_in, decoder_ch_out, self.kernel_size,
                stride=stride, padding=15, bias=False,
            )
        ]
    elif self.upsample_type == 'conv':
        decode += [
            nn.Upsample(None, decoder_ch_in),
            nn.Conv1d(decoder_ch_in, decoder_ch_out,
                self.kernel_size, stride=stride, padding=15, bias=False)
        ]

    deactivation = nn.Tanh() if decoder_ch_out == 1 else nn.ReLU()

```

```

        if decoder_ch_out > 1:
            decode.append(nn.BatchNorm1d(decoder_ch_out))
            decode.append(deactivation)

        self.decoder.append(nn.Sequential(*decode))

    attn_dims = self.growth_factor * self.depth

    for i in range(self.attn):
        one_filter = [
            nn.Conv1d(attn_dims, attn_dims, 1),
            nn.Conv1d(attn_dims, attn_dims, 1),
            nn.Conv1d(attn_dims, attn_dims, 1)
        ]

        self.attention.append(nn.Sequential(*one_filter))
    if self.attn > 1:
        self.attention_resolver = nn.Conv1d(attn_dims * self.attn,
attn_dims, 1, 1)

    self.resolver = nn.Sequential(
        nn.Conv1d(1, 1, kernel_size=1, bias=True),
        nn.Tanh(),
    )

    self.init_weights()

    def init_weights(self):
        for m in self.modules():
            if isinstance(m, nn.Conv1d) or isinstance(m,
nn.ConvTranspose1d):
                nn.init.xavier_normal_(m.weight.data)

    def attend(self, x):
        attention_states = []
        for attention_block in self.attention:
            f = attention_block[0](x)
            g = attention_block[1](x)
            h = attention_block[2](x)
            combine = nn.Softmax(dim=1)(f * g)
            attention_states.append(h * combine)
        attention_states = torch.cat(attention_states, dim=1)

        if self.attn > 1:
            return self.attention_resolver(attention_states)
        else:
            return attention_states[0]

    def resolve_skip_op(self, skip_in, another_skip_in):
        if self.skip_op == 'add':
            return skip_in + another_skip_in
        elif self.skip_op == 'cat':
            return torch.cat((skip_in, another_skip_in), dim=1)
        elif self.skip_op == 'none':
            return another_skip_in

    def forward(self, x):
        saved = [x]

        for encode in self.encoder:
            x = encode(x)
            saved.append(x)

        if self.attn > 0:

```

```
x = self.attend(x)

for index, decode in enumerate(self.decoder):
    encoder_output = saved.pop(-1)
    layer_in = encoder_output
    layer_in = self.resolve_skip_op(encoder_output, x)
    x = decode(layer_in)

x = self.resolver(x)

return x
```

Pagalbinis kodas patalpintas internetinėje talpykloje [60].

B priedas. Pranešimo 24-oje Lietuvos jaunųjų mokslininkų konferencijoje medžiaga

Garso atkūrimo triukšmingoje aplinkoje tyrimas

Danielius Visockas, VGTU, 202

Kalbos signalai yra ypač triukšmingi

Triukšmai:

- Aplinkos triukšmas
- Sensoriaus triukšmas
- Signalo suspaudimo padariniai

Pašalinus triukšmą:

- Gerėja kalbos atpažinimas (ypač sunkiau šnekantiems)
- Gerėja biometrija
- Pinga sensoriai

2

Tikslai

- Susilpninti triukšmo signalą
- Išlaikyti garso signalą
- Pagerinti žmogaus supratimą

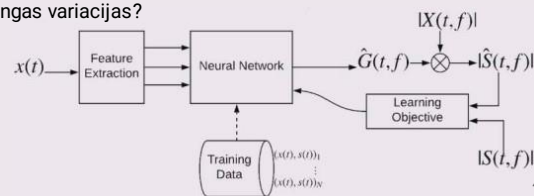
Prielaidos

- Triukšminga kalba = ideali kalba + triukšmas
- Triukšmo signalas nėra stipresnis už kalbos

3

Neuroninių tinklų mokymas

- Kokie duomenys bus naudojami?
- Kokios ypatybės bus apskaičiuotos?
- Kokia neuroninio tinklo architektūra?
- Koks tinklo mokymo tikslas?
- Kaip vertinti skirtingas variacijas?



4

Duomenys

Naudotas duomenų rinkinys, kurį naudoja kiti tyrėjai. Taip galima palyginti rezultatus.

Duomenų rinkinys susideda iš idealios kalbos ir triukšmingos kalbos porų.

Atskirti duomenys, kuriuos galima naudoti mokyme ir kuriais reikia testuoti rezultatus.

Kalbėtojai: 42 moterys, 42 vyrai

Kalba: skirtingų akcentų anglų

Triukšmas: natūralus, pridėtinis

5

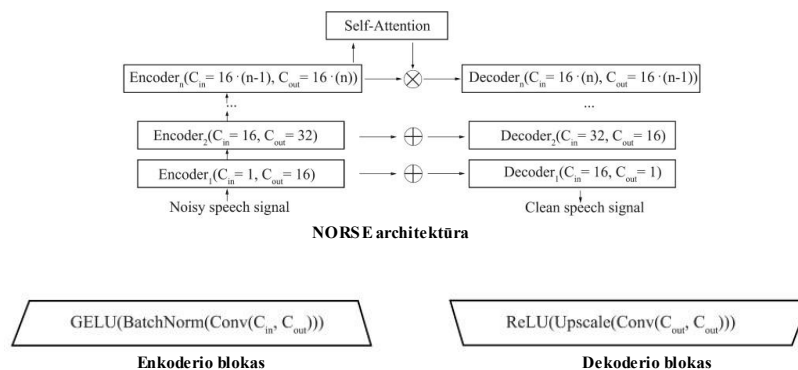
Ypatybės

Naudojamas nepakeistas signalas laiko ašyje

Neuroninis tinklas pats gali išmokti reikiamas transformacijas ir prisitaikyti prie skirtingų situacijų.

6

Neuroninio tinklo architektūra



7

Mokymo tikslas

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2.$$

$$\text{loss}_{wSDR}(x, y, \hat{y}) := \alpha \text{loss}_{SDR}(y, \hat{y}) + (1 - \alpha) \text{loss}_{SDR}(z, \hat{z}) \quad [1]$$

$$\text{loss}_{SDR}(y, \hat{y}) := -\sqrt{\frac{\text{loss}_{ven}}{\|y\|^2}} = -\frac{\langle y, \hat{y} \rangle}{\|y\| \|\hat{y}\|} \quad \hat{z} = x - \hat{y} \quad \alpha = \|y\|^2 / (\|y\|^2 + \|\hat{z}\|^2)$$

$$\text{loss}_{ven}(y, \hat{y}) := -\frac{\langle y, \hat{y} \rangle^2}{\|\hat{y}\|^2}$$

[1] Phase -aware speech enhancement. Choi et al. 2019

[2] SEGAN. Speech enhancement Generative Adversarial Network. Pascual et al. 2018

8

Vertinimas

Naudojama objektyvi PESQ^[3] metrika, modeliuojanti subjektyvią (MOS).

[3] Wideband extension to Recommendation P.862 for the assessment of wideband telephone networks and speech codecs, ITU-T, 2007.

9

Rezultatai

Metodas	PESQ
SEGAN (Pascual et al. 2018)	2.27
RNNoise (Valin et al. 2017)	2.78
SASEGAN (Phan et al. 2020)	2.36
NORSE (Visockas 2021)	3.34

10

Ateityje...

- Testuoti kaip veikia ne anglų kalbai
- Biometrija grįstas gryninimas
- Išbandyti ne aplinkos triukšmus

11

Ačiū už dėmesį.
Ar turite klausimų?

12

XXIV-toji Lietuvos jaunųjų mokslininkų konferencija
„MOKSLAS – LIETUVOS ATEITIS“

ELEKTRONIKA IR ELEKTROTECHNIKA

PAŽYMA
Danielius Visockas

Signalų technologijų (T121) sekcijoje perskaitė pranešimą

**GARSO ATKŪRIMO TRIUKŠMINGOJE
APLINKOJE TYRIMAS**

Mokslo komiteto pirmininkas

prof. habil. dr. Romanas Martavičius

Sekcijos pirmininkas

dr. Eldar Šabanovič



2021 m. kovo 19 d.
Vilnius