Tallinna Tehnikaülikool
Infotehnoloogia teaduskond

Charlie Karniol
164214 IAPB

HELISÄMPLITE KLASSIFITSEERIMINE KONVOLUTSIOONILISTE NÄRVIVÕRKUDEGA

Bakalaureusetöö

Juhendaja: Tanel Alumäe
doktorikraad

Tallinn 2019

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Charlie Karniol

22. mai 2019

Annotatsioon

Muusikalise sisuga helifaile ehk helisämpleid iseloomustab nimetusskeemide rohkus ning metaandmete puudulikkus. Käesoleva töö eesmärgiks on uurida helisämplite automaatset kategoriseerimist konvolutsiooniliste närvivõrkudega. Töökindlaid helide klassifitseerijaid saab kasutada helifailide haldamise rakendustes, mis sobiksid muusikavaldkonna spetsialistidele.

Helide klassifitseerimise kontekstis on närvivõrkude kasutamist uuritud mitmetes valdkondades, kusjuures häid tulemusi on andnud konvolutsiooniliste närvivõrkude kasutamine. Senini ei leidu töid, mis käsitleksid muusikaliste helide klassifitseerimist suure varieeruvusega andmestikul. Sobiva andmestiku puudumise tõttu koostatakse uus märgendatud sämplite andmestik, mida kasutatakse klassifitseerija arendamisel ja hindamisel.

Eksperimentaalse töö tulemusel pakutakse välja konvolutsioonilise närvivõrgu arhitektuuril põhinev helisämplite klassifitseerija, mis saavutab sarnaste lahenduste võrdluses paremaid tulemusi. Valdkonna ekspertidega korraldatud katses saavutas mudel ekspertide hinnangutega sarnaseid tulemusi. Teostatud eksperimentide ja parima mudeli analüüsi põhjal järeldatakse, et konvolutsioonilised närvivõrgud sobivad helisämplite klassifitseerimise probleemi lahendamiseks.

Töös pakutud lahendust on võimalik edasi arendada. Põhjalikumalt tuleks uurida erinevaid helisämplite kategoriseerimise skeeme, mis oleksid laiendatavad. Klassifitseerimistulemusi parandaks heli- ja failinime-analüüsi tulemuste sidumine.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 50 leheküljel, 4 peatükki, 25 joonist, 21 tabelit.

Abstract

Classification of Audio Samples Using Convolutional Neural Networks

Audio files with musical content, commonly referred to as audio samples in the music production industry, are often characterized by nondescript filenames and a lack of metadata. The intention of this work is to study the automatic categorization of audio samples using convolutional neural networks. Reliable audio classifiers may be used in conjunction with audio file browsers that are suitable for industry specialists.

In the context of audio classification, neural networks have been studied in many subfields. Recently, convolutional neural networks have been found to yield notable results for solving various audio classification problems. However, no available works have been found to study musical audio sample classification using a dataset of high variability. Due to this reason, a labeled audio sample dataset is compiled which can be used to develop and evaluate new classifiers.

In this work, an audio sample classifier is presented which is based on a convolutional neural network architecture. The proposed classifier achieves better results in comparison with similar solutions. To verify the results, a human evaluation experiment was conducted, in which the best model achieved similar results compared to the results of industry experts. On the basis of conducted experiments and thorough analysis of the best model, it is concluded that convolutional neural networks are suitable for audio sample classification tasks.

The further development of the solution proposed in this work is desirable. Foremost, exhaustive research has to be done on the subject of audio sample categorization schemes. Additionally, the results of the proposed classifier architecture may be improved by combining both audio and filename analysis.

The thesis is in Estonian and contains 50 pages of text, 4 chapters, 25 figures, 21 tables.

Lühendite ja mõistete sõnastik

ahenduskiht ingl k *pooling layer*; kiht konvolutsioonilises närvivõrgus, mis vähendab sisendi mõõtmeid, kombineerides mitu sisendit üheks väljundiks

aktiveerimisfunktsioon ingl k *activation function*; funktsioon, mis defineerib neuroni väljundi ja mida rakendatakse enne neuroni väljundi edastamist järgmisele neuronile

arhitektuur tehisnärvivõrgu kirjeldus

conv vt konvolutsiooniline kiht

dense vt täielikult ühendatud kiht

diskreetimissagedus ingl k *sample rate*; helisignaali diskreetimisel saadud üksikväärtuste ehk diskreetide arv sekundis

dropout vt väljajätumeetod

epohh üks treeningandmestiku täielik esitus treenitavale tehisnärvivõrgule

F_1-skoor kordustäpsuse ja saagise harmooniline keskmine

filter (tehisnärvivõrkude kontekstis) konvolutsiooniline neuron

helisignaal helilise informatsiooni esitusviis

hüperparameeter tehisnärvivõrgu arhitektuuri või treenimisalgoritmi parameeter, mille väärtus määratakse enne treenimisprotsessi alustamist

kaofunktsioon ingl k *loss function*; funktsioon mudeli vea mõõtmiseks, mille sisendiks on mudeli väljund ja oodatud väljund

konvolutsioon operatsioon, mille käigus tensorit teisendatakse selle mõõtmetest väiksemate mõõtmetega filtri abil

konvolutsiooniline kiht kiht tehisnärvivõrgus, mis kasutab väljundi arvutamiseks konvolutsiooni

konvolutsiooniline närvivõrk närvivõrk, mis koosneb konvolutsioonilistest kihtidest, ahenduskihtidest ning täielikult ühendatud kihtidest

kordustäpsus ingl k *precision*; klassifitseerimisprobleemis tõsipositiivsete märgendite arvu ja kõigi positiivsete märgendite arvu suhe

kvantimissügavus ingl k *bit depth*; bittide arv, mida kasutatakse helisignaali kvantimistasemete väärtuste esitamiseks

lainekuju ingl k *waveform*; helisignaali visuaalne esitus aja funktsioonina

loop tsükliline ehk korduvalt esitatav sämpel, mille lõpp lõimub sujuvalt algusega

max-pool vt ahenduskiht

mel-spektrogramm spektrogramm, mis kasutab sageduste esitamiseks inimese helikõrguste tajumise eripärasid arvestavat sageduste skaalat

miniplokk ingl k *minibatch*; treeningandmestiku jaotis

mittenegatiivne lineaarfunktsioon ingl k *rectified linear unit*, $ReLU$; funktsioon, mis on defineeritud kui $\varphi(x) = \max(0, x)$

mudel närvivõrgu arhitektuuri isend, mis saadakse treenimise tulemusel

märgend (käesoleva töö kontekstis) klassifitseerija hinnang andmepunkti kohta

neuron tehisnärvivõrgu arvutuslik element, millega on seotud vabaliige, üks või mitu kaalutud sisendit ja üks väljund

one-shot mittetsükliline ehk ühekordselt esitatav sämpel

parameeter neuroni väljundit mõjutav treenitav muutuja

regulariseerimine tehisnärvivõrgu arhitektuuri, treeningalgoritmi või treeningandmestiku muudatus, mille eesmärk on vähendada mudelite üldistusviga

ReLU vt mittenegatiivne lineaarfunktsioon

ristentroopiakadu ingl k *cross entropy loss*; kaofunktsioon, mis mõõdab viga klassifitseerimismudelil, mille väljundiks on kategooriate tõenäosusväärtused lõigul $[0, 1]$

saagis ingl k *recall*; klassifitseerimisprobleemis tõsipositiivsete märgendite arvu ja kõigi asjakohaste märgendite arvu suhe

SGD ingl k *stochastic gradient descent*; stohhastiline gradientlaskumine

spekter helisignaali koosseisu kuuluvate erineva sageduse ja amplituudiga siinussignaalide visuaalne esitus

spektrogramm spektri muutumise visuaalne esitus ajas

sämpel ingl k *sample*, helisämpel; muusikaline või muusika loomiseks kasutatav helifail

tehisnärvivõrk e närvivõrk; arvutuslik süsteem, mis loob raamistiku keeruliste andmesisendite töötlemiseks erinevate masinõppe algoritmide ja meetodite koostööl

tensor (arvutusteaduse kontekstis) mitmemõõtmeline massiiv, mida saab kasutada arvutuste paralleliseerimiseks graafikaprotsessoritel

tunnuskaart ingl k *feature map*; konvolutsioonilise neuroni väljund

täielikult ühendatud kiht ingl k *fully connected layer*, *dense layer*; kiht närvivõrgus, mille neuronid saavad oma sisendid kõikide eelmise kihi neuronite väljunditest

tämber muusikaliste parameetrite kogumit, mille erinevate väärtuste korral tajub kuulaja sama helivältuse, helikõrguse ja helitugevusega helisid erinevalt

täpsus ingl k *accuracy*; klassifitseerimisprobleemis tõsipositiivsete ja tõsinegatiivsete märgendite arvu ja kõigi märgendite arvu suhe

viga ingl k *error rate*; klassifitseerimisprobleemis väärpositiivsete ja väärnegatiivse märgendite arvu ja kõigi märgendite arvu suhe

väljajätumeetod ingl k *dropout method*; regulariseerimise meetod, mis vähendab ülesobitumist, vältides neuronite omavahelist adapteerumist

õpisamm ingl k *learning rate*; hüperparameeter, mis defineerib tehisnärvivõrgu parameetrite muutumise määra optimiseerimisalgoritmi ühel iteratsioonil

Sisukord

Sissejuhatus **12**

1 Taust **13**
1.1 Heli ja helitaju 13
1.2 Helisignaalid 14
1.2.1 Ajaesitus 14
1.2.2 Spekter 15
1.2.3 Spektrogramm 16
1.2.4 Tunnused 17
1.3 Digitaalheli 18
1.4 Helisämplid 20
1.4.1 Metaandmed ja kasutamine 20
1.5 Klassifitseerimine 22
1.5.1 Statistilised kriteeriumid 22
1.6 Tehisnärvivõrgud 24
1.6.1 Konvolutsioonilised närvivõrgud 26
1.6.2 Treenimine 27
1.6.3 Regulariseerimine 28
1.7 Kasutatud tehnoloogiad 30
1.7.1 PyTorchiga seotud mõisted 30

2 Olemasolevad tööd **32**
2.1 Uurimistööd ja andmestikud 32
2.2 Arhitektuurid 33
2.3 Rakendused 34

3 Eksperimendid **36**
3.1 Helifailide algandmestik 36
3.2 Arendusandmestikud 36
3.3 Arhitektuuri arendamine 39
3.3.1 Baasarhitektuur 41
3.3.2 Konvolutsiooniline baasarhitektuur 41
3.3.3 Varane lõpetamine 43
3.3.4 Ristkülikukujulised filtrid 44

3.3.5 Paralleelarhitektuur . . . 45
3.3.6 Lisatunnuste kasutamine . . . 47
3.3.7 Rikastatud andmete kasutamine . . . 47
3.3.8 *Mix-up* meetodi kasutamine . . . 50
3.3.9 Rikastatud andmete ja *mix-up* meetodi üheaegne kasutamine . 50
3.4 Pakutud arhitektuur . . . 51

4 Analüüs . . . 52
4.1 Vigade uurimine . . . 52
4.2 Võrdlused . . . 54
4.2.1 SampleCNN ja SwishNet . . . 54
4.2.2 Sononym ja Samplism . . . 55
4.2.3 Valdkonna eksperdid . . . 57
4.3 Edasiarendused . . . 58
4.3.1 Kategooriate hierarhia . . . 58
4.3.2 Andmestiku laiendamine . . . 59
4.3.3 Arhitektuuri parendamine . . . 59
4.3.4 Rakendus . . . 60

Kokkuvõte . . . 61

Kasutatud allikad . . . 62

Joonised

1 Analoog-helisignaali ajaesitus 15
2 Digitaalse helisignaali ajaesitus 15
3 Erineva sagedusega siinuslained 15
4 Siinussignaali ajaesituses ja sagedusesitus 16
5 Keerulisema helisignaali ajaesitus ja sagedusesitus 16
6 Mel-filtripank 40 filtriga kuni sageduseni 4000 Hz 17
7 Pika helisignaali mel-spektrogramm 17
8 Lühikese helisignaali mel-spektrogramm 18
9 Signaali diskreetimine kahe erineva diskreetimissagedusega 19
10 Siinussignaali kvantimine kolme erineva kvantimissügavusega 20
11 Klassifitseerimisega seotud mõisted 23
12 Neuron 24
13 Lihtne tehisnärvivõrgu struktuur 25
14 Kahemõõtmelise konvolutsiooni rakendamine 26
15 *Max*-ahenduskihi rakendamine 27
16 *DNN-Baseline* mudeli kaofunktsiooni väärtuste graafik 42
17 *CNN-Baseline* mudeli kaofunktsiooni väärtuste graafik 43
18 *CNN-EarlyStop* mudeli kaofunktsiooni väärtuste graafik 44
19 *CNN-Rectangular* mudeli kaofunktsiooni väärtuste graafik 45
20 *CNN-ParallelRectangular* mudeli kaofunktsiooni väärtuste graafik 46
21 *CNN-ExtraParallelRectangular* mudeli segadusmaatriks 48
22 *CNN-ExtraParallelRectangular* mudeli kaofunktsiooni väärtuste graafik . 49
23 *CNN-MixupParallel* mudeli kaofunktsiooni väärtuste graafik 50
24 Pakutud arhitektuuril treenitud mudeli segadusmaatriks 53
25 Kategooriate hierarhia näide 59

Tabelid

1 Helisämplite algandmestiku kategooriline jaotus 37
2 *DNN-Baseline* arhitektuur 41
3 *DNN-Baseline* mudelite koondtulemused 41
4 *CNN-Baseline* arhitektuur 42
5 *CNN-Baseline* mudelite koondtulemused 42
6 *CNN-EarlyStop* mudelite koondtulemused 43
7 *CNN-Rectangular* arhitektuur 44
8 *CNN-Rectangular* mudelite koondtulemused 45
9 *CNN-ParallelRectangular* arhitektuur 46
10 *CNN-ParallelRectangular* mudelite koondtulemused 46
11 *CNN-ExtraParallelRectangular* arhitektuur 47
12 *CNN-ExtraParallelRectangular* mudelite koondtulemused 48
13 *CNN-ParallelRectangular* mudelite koondtulemused andmestikuga S_a . . 49
14 *CNN-ParallelRectangular* mudelite koondtulemused andmestikuga S_A . . 49
15 *CNN-MixupParallel* mudelite koondtulemused 50
16 *CNN-MixupParallel* mudelite koondtulemused andmestikuga S_A 51
17 Konvolutsioonilise arhitektuuri paranemine eksperimentide kaupa 51
18 Pakutud arhitektuuri tulemuste paranemine võrreldes baasarhitektuuridega 51
19 Kolme arhitektuuri mudelite koondtulemuste võrdlus 55
20 Mudeli ja võrreldavate rakenduste klassifitseerimistulemused 56
21 Mudeli ja ekspertide klassifitseerimistulemused 57

Sissejuhatus

Helikunstiga tegelevad professionaalid ning asjaarmastajad kasutavad oma tööks helisämpleid ehk muusikalise sisuga helifaile. Sageli on failide nimed eksitavad ning metaandmed puudulikud, mistõttu on failide otsimine ja kasutamine raskendatud. Kui helifailide arv küündib tuhandetesse ja miljonitesse, siis on helide otsinguks vajalike metaandmete käsitsi loomine ja redigeerimine tülikas ja aeganõudev. Seetõttu oleks vaja tarkvaralahendust sämplite otsimiseks automaatselt tuvastatavate kategooriate ja omaduste järgi ning sämplite omavahelise sarnasuse järgi.

Käesoleva töö probleemistikuks on helisämplite automaatne kategoriseerimine ehk klassifitseerimine. Helide klassifitseerimise üheks võimaluseks on kasutada tehisnärvivõrke. Heli kontekstis on närvivõrkude kasutamist põhjalikult uuritud mitmetes valdkondades, näiteks instrumentide tuvastamisel ning keskkonnahelide klassifitseerimisel, kusjuures häid tulemusi on andnud konvolutsioonilised närvivõrgud. Helisämplite klassifitseerimist käsitlevaid töid on aga vähe. Edasiminekud selles valdkonnas on eelduseks eelpool kirjeldatud tarkvaralise lahenduse realiseerimisele.

Eesmärgiks on uurida konvolutsiooniliste närvivõrkude kasutamise võimalusi ja võimekust helisämplite klassifitseerimisel. Probleemi lahendamiseks on seatud alamülesanded. Kõigepealt koostatakse uus märgendatud helisämplite andmestik. Seejärel eraldatakse sämplitest tunnused, mis võimaldaksid sämpleid klassifitseerida. Järgmisena arendatakse eksperimentaalselt välja konvolutsioonilise närvivõrgu arhitektuuril põhinev helisämplite klassifitseerija. Töö tulemusena valmib uus lahendus, mida analüüsitakse lähtudes erinevatest aspektidest.

Töö koosneb neljast osast. Esimeses osas kirjeldatakse valdkonna teoreetilist tausta. Teises osas antakse ülevaade kirjandusest ja olemasolevatest lahendustest. Kolmas osa on töö tuum: selgitatakse eksperimentide metoodikat ja esitatakse töö arengukäik alates andmestiku ja baasmudeli loomisest kuni lõpliku mudeli loomiseni ja valideerimiseni. Neljandas osas analüüsitakse mudeli tulemuslikkust helide klassifitseerimisel ning kirjeldatakse võimalusi töö edasiarendamiseks.

1 Taust

Selles peatükis antakse ülevaade uurimistöös kasutatud mõistetest, meetoditest ja tehnoloogiatest, mis aitab paremini mõista töö eksperimentaalset osa.

1.1 Heli ja helitaju

Füüsikalises tähenduses on heli elastses keskkonnas leviv elastsuslaine, mida võib nimetada ka helilaineks. Heli iseloomustavate füüsikaliste omaduste hulka kuuluvad võnkumise kestus, võnkesagedus, võnkeamplituud ja helirõhk[61][78].

Heliks nimetatakse ka helilainete poolt kuulmiselunditele avaldatud rõhu tajumist, mis tekitab kuulmisaistingu[61][78].

Inimese poolt tajutav minimaalne helirõhu amplituud ehk kuuldelävi on kokkuleppeliselt 20 µPa[65] ja kuulmise valulävi on erinevate allikate põhjal umbes 10 Pa kuni 200 Pa[16][85]. Kuulmisaistingu tugevust väljendatakse helirõhkude logaritmilise suhtena detsibellides ehk helirõhutasemena:

$$L_p = 20 \log_{10} \left(\frac{p}{p_0} \right) \text{ dB}, \tag{1}$$

kus p on võrreldava helirõhu ruutkeskmine ja p_0 on helirõhu lähtesuurus, milleks on tavaliselt kuuldelävi[79]. Võttes valuläviks 20 Pa, on inimese helirõhu tajumise dünaamiliseks ulatuseks 120 dB.

Inimesed on suutelised kuulma helilaineid, mille sagedus jääb vahemikku 20 Hz kuni 20 kHz. Kahe kõrva olemasolu võimaldab inimesel tajuda heliallika asukohta ruumis[16].

Kuna inimene ei ole suuteline eristama väikseid muutusi heli füüsikalistes omadustes, siis on defineeritud heli muusikalised omadused, mis võimaldavad kirjeldada heli sellisena, nagu inimene seda tajub. Heli muusikalised omadused on helivältus, helikõrgus, helitugevus ja tämber ehk kõlavärv[61].

Tämber on muusikaliste parameetrite kogum, mille erinevate väärtuste korral tajub kuulaja sama helivältuse, helikõrguse ja helitugevusega helisid erinevalt[63]. Tämbrit määravateks muusikalisteks parameetriteks on näiteks helitugevuse kõvera mähisjoon (ingl k *amplitude envelope* või *ADSR*), helispektri mähisjoon (ingl k *spectral envelope*) ja selle muutumine ajas, vibraato (helikõrguse perioodiline muutumine), tremolo (helitugevuse perioodiline muutumine)[86].

Igal inimhäälel ja muusikainstrumendil on oma unikaalne tämber. Tämber on helide eristamisel kõige olulisem muusikaline omadus.

1.2 Helisignaalid

Helisignaal on helilise informatsiooni esitusviis. Helisignaale on võimalik sünteesida, salvestada ja esitada nii analoogsignaalina ehk pideva signaalina kui ka digitaalse signaalina ehk diskreetse signaalina[68]. Helisignaali esitavaks suuruseks võib olla näiteks helirõhk, vahelduvvoolu pinge või võnkeamplituudi kodeeringu väärtus.

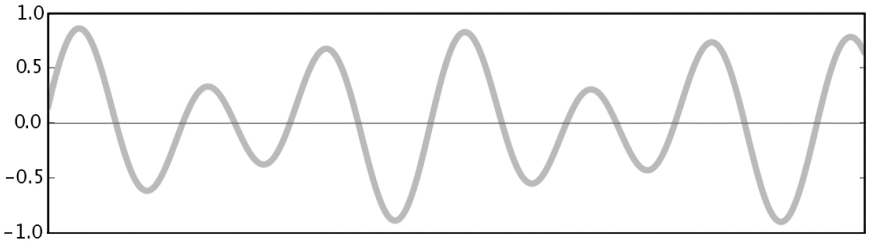
Alates fonograafi leiutamisest on helilaineid võimalik jäädvustada ja taasesitada füüsilise kandja abil. Analoog-helisignaale kandvad meediumid, näiteks fonograafplaadid ning magnetlindiga kassetid, talletavad heli lainekuju pidevana, kasutades selleks plaadi helivagu sügavust või kassetilindi magneetumise ulatust[2]. Samuti on võimalik analoog-helisignaale sünteesida muutuva pingega elektrisignaalina, kasutades analoogsüntesaatoreid.

Digitaalseid helisignaale kandvad meediumid, näiteks CD-plaadid ja kõvakettad, talletavad heli lainekuju diskreetsel kujul, kahendarvuliste üksikväärtuste jadana[2]. Digitaalne signaal saadakse analoog-signaali diskreetimisel ja kvantimisel, kasutades analoog-digitaalkonverterit. Digitaalsignaale on võimalik ka sünteesida, kasutades digitaalsüntesaatoreid või arvuteid. Digitaalheli kirjeldatakse alapeatükis 1.3.

1.2.1 Ajaesitus

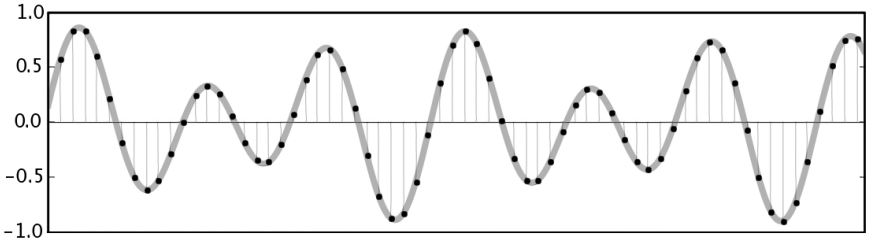
Helisignaali iseloomustab **lainekuju** (ingl k *waveform*) ehk helisignaali visuaalne esitus aja funktsioonina[88], mida võib nimetada ka helisignaali ajaesituseks.

Joonisel 1 on kujutatud analoog-helisignaali ajaesitus. Ajaesituse horisontaalteljel on kujutatud aega ning vertikaalteljel signaali esitavat suurust. Joonisel ei arvestata tegelike aja ja signaali esitava suuruse suhetega, mistõttu on signaali esitava suuruse mõõtkava kujutatud valikuliselt vahemikus $[-1, 1]$ ning ajatelje mõõtkava on jäetud määramata.



Joonis 1. Analoog-helisignaali ajaesitus[2].

Joonisel 2 on kujutatud analoog-helisignaali ajaesitus koos sellele vastava digitaalse helisignaali ajaesitusega. Punktid analoog-helisignaali lainekuju peal tähistavad digitaalse helisignaali üksikväärtuseid.

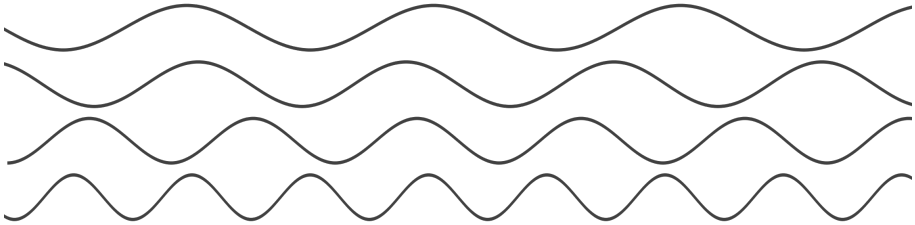


Joonis 2. Digitaalse helisignaali ajaesitus[2].

1.2.2 Spekter

Matemaatiliselt kirjeldatakse helilaineid siinuseliste tasalainete kontekstis, mis tähendab, et keerulisi helilaineid tõlgendatakse siinuslainete kombinatsioonina[78].

Joonisel 3 on kujutatud erineva sagedusega siinuslaineid. Ülalpool on madalama sagedusega siinuslained ja allpool on kõrgema sagedusega siinuslained. Horisontaalsihis on kujutatud aega.

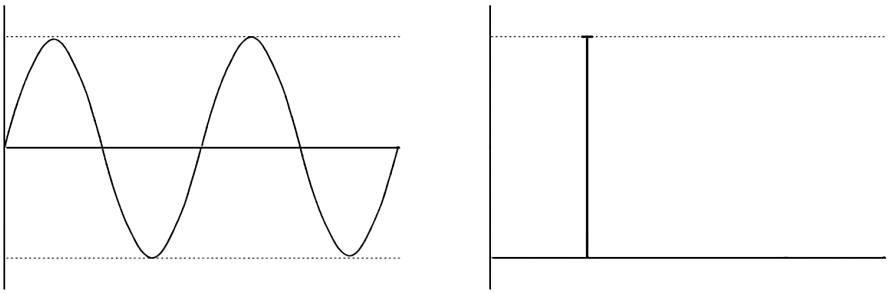


Joonis 3. Erineva sagedusega siinuslained[78].

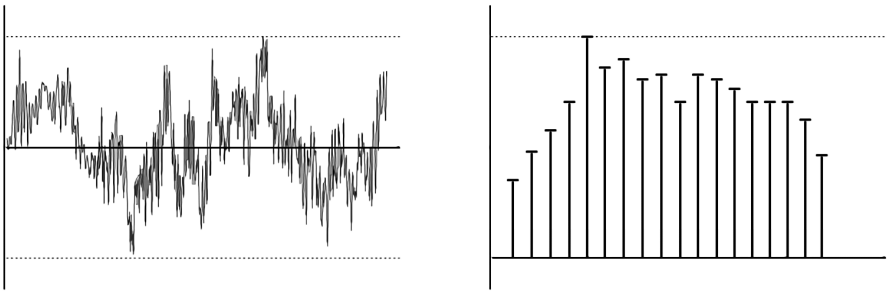
Fourier' teisendus võimaldab esitada signaale erineva sageduse, faasi ja amplituudiga siinusfunktsioonide summana ehk signaali spektrina[49][72][81].

Helisignaali **spekter** on helisignaali koosseisu kuuluvate erineva sageduse ja amplituudiga siinussignaalide visuaalne esitus, mida võib nimetada ka helisignaali sagedusesituseks.

Joonistel 4 ja 5 on kujutatud kahe helisignaali ajaesitust (vasakul) ja sagedusesitust (paremal). Tulbad sagedusesitusel kujutavad helisignaali koosseisu kuuluvaid siinussignaale. Sagedusesituse horisontaalteljel on kujutatud siinussignaali sagedust ning vertikaalteljel siinussignaali amplituudi.



Joonis 4. Siinussignaali ajaesituses ja sagedusesitus[18].



Joonis 5. Keerulisema helisignaali ajaesitus ja sagedusesitus[18].

1.2.3 Spektrogramm

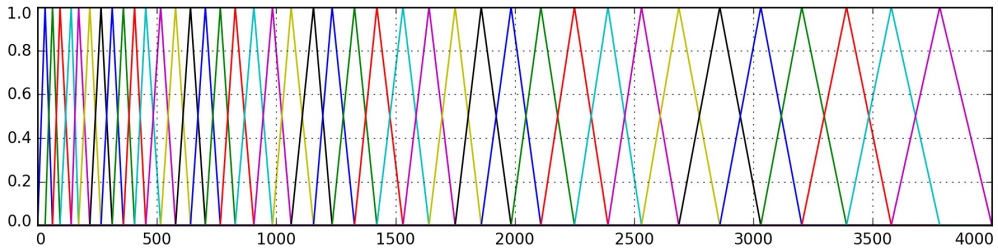
Heli omaduste muutumine tingib ka helisignaali spektri muutumise. **Spektrogramm** on signaali spektri muutumise visuaalne esitus ajas[82].

Heli analüüsimisel on osutunud kasulikuks kasutada **mel-spektrogramme** ehk spektrogramme, mis kasutavad sageduste esitamiseks mel-skaalat ehk inimese helikõrguste tajumise eripärasid arvestavat sageduste skaalat[73].

Mel-spektrogrammi saamiseks vähendatakse Fourier' teisenduste tulemusena saadud spektrogrammis sisalduvat sageduslikku informatsiooni filtrite abil. Filtripank määrab filtrid ehk sagedusvahemikud, millesse jäävad siinussignaalide amplituudväärtused kombineeritakse igal ajahetkel üksikuks amplituudväärtuseks. Kui filtripanga

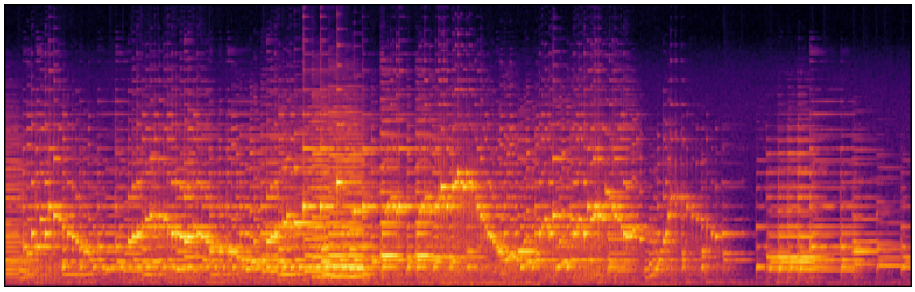
filtrid on asetatud üksteisest võrdsetele kaugustele mel-skaala järgi, siis võib seda filtripanka nimetada mel-filtripangaks.

Joonisel 6 on kujutatud mel-filtripanka 40 filtriga kuni sageduseni 4000 Hz. Horisontaalteljel on kujutatud sagedust ning vertikaalteljel amplituudi. Madalatel sagedustel suudab inimene helikõrgusi eristada paremini, mistõttu on filtrid täpsemad (kitsamad) madalamatel sagedustel.



Joonis 6. Mel-filtripank 40 filtriga kuni sageduseni 4000 Hz[21].

Joonisel 7 on kujutatud pikema helisignaali mel-spektrogrammi. Horisontaalteljel on kujutatud aega ning vertikaalteljel filtrite sageduslikke keskväärtusi. Värvid kujutavad vastava filtri võimsust. Joonisel tekkivaid heledamaid mustreid võib tõlgendada meloodiate või harmooniatena. Tumedamad alad tähistavad madalat võimsust vastaval kesksagedusel paiknevas filtris.

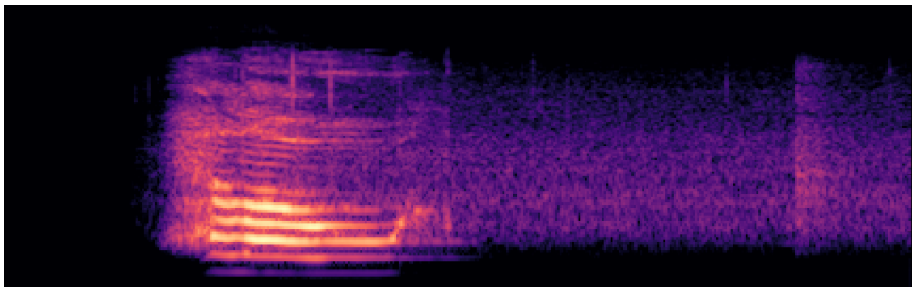


Joonis 7. Pika helisignaali mel-spektrogramm[31].

Joonisel 8 on kujutatud lühema helisignaali mel-spektrogrammi. Joonisel tekkivaid heledamaid jooni võib tõlgendada instrumendi või hääle esinemisena. Heledatele joontele järgnevaid vähem heledamaid alasid võib tõlgendada mürana. Tumedad alad tähistavad signaali puudumist vastaval sagedusel.

1.2.4 Tunnused

Lisaks aja- ja sagedusesitusele on helisignaale võimalik kirjeldada ka erinevate ajaliste ja spektraalsete tunnuste abil. Ajalisi tunnuseid tuletatakse helisignaali ajaesitusest ning spektraalseid tunnuseid tuletatakse helisignaali sagedusesitusest.



Joonis 8. Lühikese helisignaali mel-spektrogramm[55].

Nullist üleminekute sagedus (ingl k *zero-crossing rate*) on helisignaali ajaline tunnus, mis kirjeldab signaali väärtuse märgi muutumiste arvu sekundis. Nullist üleminekute sagedus on tunnus, mis võib osutuda kasulikuks löökpillide klassifitseerimisel[84].

Spektraaltsentroid (ingl k *spectral centroid*) on helisignaali spektraalne tunnus, mis kirjeldab spektri niinimetatud keskpunkti. Spektraaltsentroidi abil on võimalik kirjeldada heli tämbri "heledust" ehk "teravust" ja "tumedust" ehk "pehmust"[80].

Spektraalne lamedus (ingl k *spectral flatness*) on helisignaali spektraalne tunnus, mis kirjeldab heli tämbri tonaalsust ja mürasust[36].

Spektraalne murdesagedus (ingl k *spectral roll-off frequency*) on helisignaali spektraalne tunnus, mis kirjeldab sagedust, alla mille jääb valikuliselt määratud osa kogu spektri energiast[37]. Spektraalse murdesageduse abil saab tuvastada spektri ekstremaalseid sagedusi ning eristada harmoonilisi helisid ja mürahelisid[37][41].

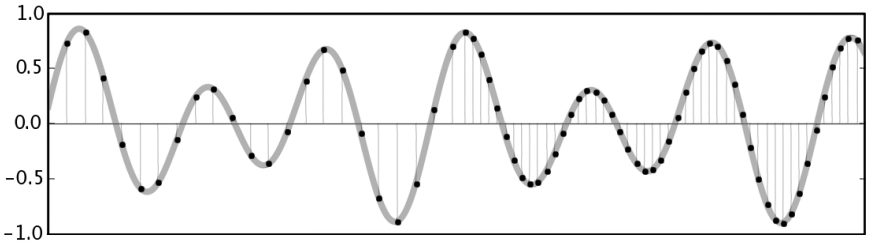
1.3 Digitaalheli

Käesolevas töös tegeletakse digitaalse heliga. Digitaalne heli on kahendarvude jadana kodeeritud helisignaal, mis saadakse analoog-helisignaali diskreetimisel ja kvantimisel või digitaalse helisünteesi tulemusena. Digitaalset heli iseloomustavad peamiselt kaks suurust: **diskreetimissagedus** (ingl k *sample rate*) ja **kvantimissügavus** (ingl k *bit depth*)[70]. Mäluseadmetele salvestatud digitaalset heli talletatakse erineva vorminguga helifailides. Helifaile käsitletakse lähemalt peatükis 1.4.

Diskreetimine on protsess, mille käigus pidev signaal teisendatakse diskreetseks signaaliks. Diskreetimissagedus määrab signaali diskreetimisel saadud üksikväärtuste ehk diskreetide arvu sekundis. Mida suurem on diskreetimissagedus, seda täpsem on salvestatud signaal[2].

Vastavalt Nyquist-Shannoni teoreemile on signaali kadudeta salvestamine ja taasesitamine võimalik juhul, kui diskreetimissagedus on vähemalt kaks korda suurem kui kõrgeim signaalis esinev sagedus[75]. Heliplaatidele salvestatud digitaalsete signaalide levinuim diskreetimissagedus on 44.1 kHz[2], mis võimaldab taasesitada inimesele kuuldavaid helisid täies ulatuses.

Joonisel 9 on kujutatud digitaalse helisignaali üksikväärtuseid nii, et joonise vasakus osas on digitaalsignaalil väike diskreetimissagedus ning paremas osas suur diskreetimissagedus. Suurema diskreetimissagedusega digitaalne signaal esitab esialgset (pidevat) signaali täpsemini.

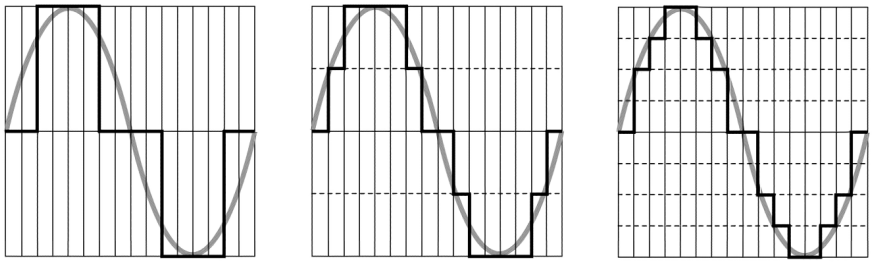


Joonis 9. Signaali diskreetimine kahe erineva diskreetimissagedusega[2].

Kvantimine on protsess, mille käigus diskreetsignaali mõõtepiirkond jaotatakse lõplikuks arvuks vahemikeks ehk kvantimistasemeteks. Kvantimissügavus on bittide arv, mida kasutatakse kvantimistasemete väärtuste esitamiseks[62]. Mida suurem on kvantimissügavus, seda täpsem on salvestatud signaal. Kvantimissügavus määrab ka salvestatava signaali dünaamilise ulatuse ehk vahe madalaima ja kõrgeima amplituudiga punkti vahel. Heliplaatidele salvestatud digitaalsete signaalide levinuim bitisügavus on 16 bitti, mis võimaldab salvestada diskreetide väärtuseid dünaamilise ulatusega kuni 96 dB[2], sarnanedes ligikaudu inimese poolt tajutavale helide dünaamilisele ulatusele.

Joonisel 10 on kujutatud konstantse diskreetimissagedusega digitaalset helisignaali erinevatel kvantimissügavustel. Kvantimistasemete arv suureneb vasakult paremale: esimesel joonisel on tasemeid 3, teisel joonisel 5 ning kolmandal joonisel 9. Suurema kvantimistasemete arvuga digitaalne signaal esitab esialgset (pidevat) signaali täpsemini.

Lisaks eelmainitule on digitaalse heli omaduseks ka helikanalite arv. Helikanal esitab punkti ruumis, kust heli pärineb. Sõltuvalt helifaili salvestatud kanalite arvust öeldakse, et helifail on kas monos (ühe kanaliga), stereos (kahe kanaliga) või mitme kanaliga (rohkem kui kahe kanaliga). Ruumilise heli taasesitamiseks on vajalik, et digitaalsel helil oleks vähemalt kaks kanalit. Heliplaatidele salvestatud helifailid on üldjuhul kahe kanaliga.



Joonis 10. Siinussignaali kvantimine kolme erineva kvantimissügavusega[29].

1.4 Helisämplid

Käesoleva töö kontekstis on **sämpel** muusikaline või muusika loomiseks kasutatav helifail. Sämplite hulka kuuluvad seega nii akustiliste, elektrooniliste kui ka sünteesitud instrumendide salvestused, vokaali- ehk laulusalvestused ja muud salvestused. Üldjuhul ei nimetata sämpliks muusikateost, kuid sämpel võib olla lõik muusikateosest, mida on võimalik kasutada helilise loometöö käigus.

Sämpli näideteks võib tuua: lauldud fraas või osa kõnest; basstrummi, soolotrummi või taldriku löök; elektriklaveri või -oreli akord; süntesaatoriheli; salvestatud müra või keskkonnaheli; lühike lõik mõne heliteose trummipartiist.

Sämplite kasutamine on helikunstiga tegelevate professionaalide ja asjaarmastajate igapäevase töö osa. Heliloojad, -produtsendid, -disainerid ja muusikud kasutavad sämpleid digitaalsete heliteoste osadena.

Sämpleid on võimalik osta füüsilise koopiana andmekandjal või osta ja tasuta alla laadida erinevatest internetiallikatest. Mõnedeks sämplite allalaadimise veebikeskkondadeks on Splice[1], Soundsnap[2], LANDR[3] ja Freesound[4].

1.4.1 Metaandmed ja kasutamine

Metaandmed on andmed, mis kirjeldavad teisi andmeid. Failide puhul on võimalik metaandmeid salvestada faili enese sisse[74].

Failihaldurid kuvavad kasutajale helifailide tehnilisi metaandmeid, mille hulka kuuluvad näiteks faili suurus, vorming, diskreetimissagedus ja kvantimissügavus. Tehnilised metaandmed on failist tuletatavad. Sämplite kasutamise seisukohalt on aga olulisemad kirjeldavad metaandmed, mis võimaldavad faile heli omaduste järgi ot-

[1] https://splice.com/
[2] https://soundsnap.com/
[3] https://samples.landr.com/
[4] https://freesound.org/

sida ja tuvastada. Sämpleid kirjeldavate metaandmete hulka kuuluvad näiteks heli kestvus, tempo, helistik, žanr ja kategooriad.

Kategooriate abil on võimalik väljendada helisämpli instrumenti või üldisi tämbrilisi omadusi. Sämplite kategoriseerimise võimalusi on erinevaid. Käesolevas töös lähtutakse valdkonnas levinud dokumenteerimata tavadest, mida käsitletakse lähemalt alapeatükis 3.1.

Metaandmete rohkus ja varieeruvus tingib vajaduse nende struktureerituse järele. Levinuimaks helifailide metaandmete konteineriks on ID3[27], mida kasutatakse muusika jaoks ning mis üldjuhul ei sobi sämplite jaoks. Helisämplite kontekstis levinumad WAVE- (*Waveform Audio File Format*) ja AIFF-vormingud (*Audio Interchange File Format*) toetavad lisaks ID3-konteinerile ka XMP-standardi (*Extensible Metadata Platform*) järgsete andmete salvestamist[67][87], mis võimaldab väljendada suvalise struktuuriga metaandmeid[71]. Digitaalheli tööjaamad (ingl k *digital audio workstation*) ja helifailide haldamise tarkvaraprogrammid salvestavad metaandmeid tihti omas vormingus ja eraldi failidena.

Valdavalt on kirjeldavad metaandmed helisämplite kohta puudulikud. ID3-konteiner on sämplite kontekstis piiratud struktuuriga ning selles sisaldub tihti ainult sämplikogumiku nimi või veebiaadress, millest ei ole tavaliselt kasu sämplite otsimisel nende helilise sisu järgi. XMP-metaandmeid kasutatakse harva. Levinud on sämplite haldamistarkvara abil käsitsi või pool-automaatselt loodud metaandmete kasutamine, kuid kindel struktuur metaandmete salvestamiseks puudub.

Sämplite kasutamisel on kõige olulisem faili nimi: see on esimene, mida kasutaja sämpleid sirvides näeb, ning peamine, mille järgi sämpleid failisüsteemist otsitakse. Olulisuselt järgmine on sämpli paiknemine failisüsteemi hierarhias, sest kataloogi nimi võib anda informatsiooni sämpli kategooria kohta. Sämpleid loovatel autoritel ja sämplite kasutajatel on oma isiklikud moodused failide nimetamiseks ja korrapärastamiseks. Professionaalseid ja tasulisi sämplikogumikke eristab teistest peamiselt asjaolu, et neis sisalduvate sämplite nimed ja paigutus on ühtlase skeemiga ning nimedes sisaldub rohkem kirjeldavat informatsiooni, milleks on peamiselt sämpli kategooria, helistik ja tempo ning vahel ka tämbri paarisõnaline kirjeldus. Suure osa kättesaadavate sämplite puhul kirjeldab faili nimi sämpli kategooriat, kuid kategooriate liigitus ja nimetused varieeruvad. Internetist tasuta allalaaditavatel sämplitel ei pruugi faili nimes kirjeldavat informatsiooni sisalduda.

Kokkuvõtlikult võib öelda, et helisämpleid valdab nimetus- ja paigutusskeemide rohkus ja metaandmete puudulikkus, mis muudab sämplite tuvastamise, korrapärastamise, otsimise ja kasutamise keeruliseks.

1.5 Klassifitseerimine

Klassifitseerimine on probleem masinõppe ja statistika valdkonnas, mis seisneb uue vaatlustulemuse kategooria määramises teadaolevate vaatlustulemuste ehk treeningandmestiku põhjal. Algoritmi, mis lahendab mõnda konkreetset klassifitseerimise probleemi, nimetatakse klassifitseerijaks (ingl k *classifier*). Klassifitseerijat võib matemaatiliselt vaadelda ka kui kujutust või funktsiooni, mis seab sisendandmetega vastavusse kategooria[83].

Helisämplite automaatse klassifitseerimise probleem seisneb sellise klassifitseerija arendamises, mis suudaks määrata sämpli kategooria, olenemata faili nimest või paiknemisest failisüsteemis.

1.5.1 Statistilised kriteeriumid

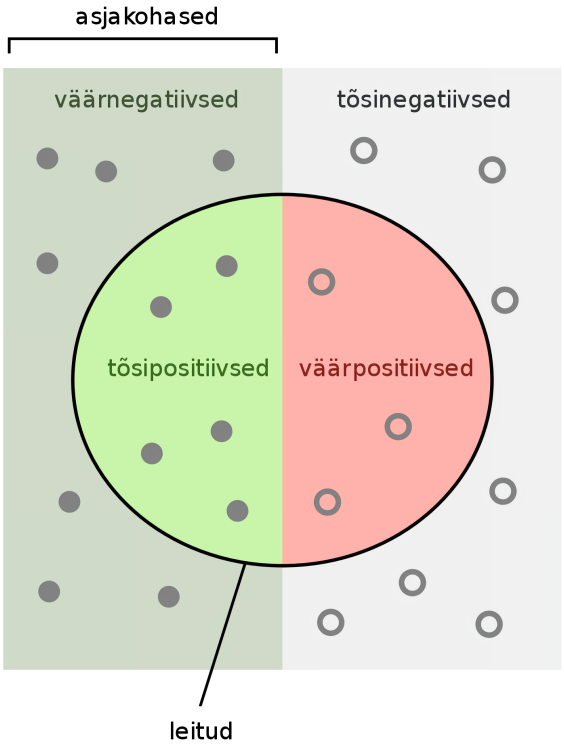
Klassifitseerimisülesannete jaoks on kasutusele võetud mõisted, mille abil on võimalik võrrelda klassifitseerija hinnanguid teadaolevate hinnangutega. Neid mõisteid kujutatakse joonisel 11.

Mõisted "positiivne" ja "negatiivne" on seotud nii klassifitseerija hinnanguga kui ka teadaoleva hinnanguga. Positiivne hinnang teatud kategooria suhtes tähendab, et vaatlustulemus on klassifitseeritud sellesse kategooriasse. Negatiivne hinnang teatud kategooria suhtes tähendab, et vaatlustulemus ei ole klassifitseeritud sellesse kategooriasse. Mõisted "tõene" ja "väär" on seotud klassifitseerija hinnanguga teadaoleva vaatlustulemuse kohta. Tõene hinnang tähendab, et klassifitseerija on uue vaatlustulemuse klassifitseerinud õigesti, ja väär hinnang, et valesti[64][76].

Klassifitseerija hinnangut uue vaatlustulemuse kohta võib nimetada märgendiks (ingl k *label*). Tõsipositiivsed märgendid on õigesti positiivsena klassifitseeritud. Tõsinegatiivsed märgendid on õigesti negatiivsena klassifitseeritud. Väärpositiivsed märgendid on valesti positiivsena klassifitseeritud. Väärnegatiivsed märgendid on valesti negatiivsena klassifitseeritud. Leitud märgendid on märgendid, mis on nii õigesti kui ka valesti positiivsena klassifitseeritud. Asjakohased märgendid on märgendid, mis on klassifitseeritud tõsipositiivsete ja väärnegatiivsetena[64][76].

Klassifitseerija hindamiseks on kasutusel erinevad statistilised kriteeriumid: viga, täpsus, kordustäpsus, saagis ja F_1-skoor.

Täpsus (ingl k *accuracy*) on tõsipositiivsete ja tõsinegatiivsete märgendite arvu ja kõigi märgendite arvu suhe. Täpsus väljendab klassifitseerija tõenäosust teha tõene hinnang. Täpsus on vea täiendtõenäosus.



Joonis 11. Klassifitseerimisega seotud mõisted[64].

Viga (ingl k *error rate*) on väärpositiivsete ja väärnegatiivse märgendite arvu ja kõigi märgendite arvu suhe. Viga väljendab klassifitseerija tõenäosust teha väär hinnang. Viga on täpsuse täiendtõenäosus.

Kordustäpsus (ingl k *precision*) on tõsipositiivsete märgendite arvu ja kõigi positiivsete, nii tõeste kui väärate, märgendite arvu suhe. Kordustäpsus väljendab klassifitseerija tõenäosust teha tõene positiivne hinnang[76].

Saagis (ingl k *recall*) on tõsipositiivsete märgendite arvu ja asjakohaste märgendite arvu suhe. Saagis väljendab klassifitseerija tulemuste asjakohasust ehk tõenäosust hinnata teadaolevalt positiivset vaatlustulemust tõeselt[76].

***F*$_1$-skoor** on kordustäpsuse ja saagise harmooniline keskmine. Kõrge F_1-skoor väljendab madalat väärpositiivsete ja väärnegatiivsete osakaalu klassifitseerimistulemustes. F_1-skoori üldistus on F_β-skoor, mis võimaldab hinnangu arvutamisel määrata kordustäpsuse ja saagise omavahelise osakaalu[76].

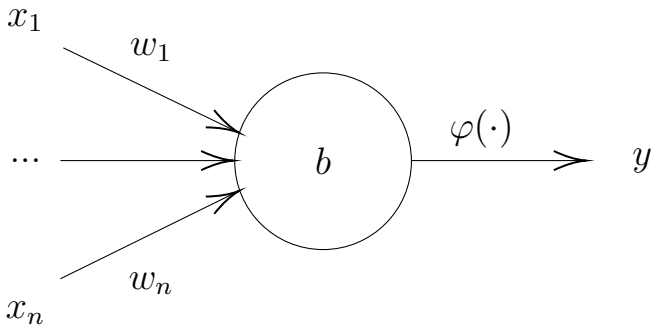
Tihti kasutatakse klassifitseerija tulemuste esitamiseks **segadusmaatriksit**, mis näitab seoseid klassifitseerija hinnangute ja teadaolevate hinnangute vahel.

1.6 Tehisnärvivõrgud

Tehisnärvivõrk ehk närvivõrk on arvutuslik süsteem, mis loob raamistiku andmesisendite töötlemiseks erinevate masinõppe algoritmide ja meetodite koostööl. Närvivõrke treenitakse ära tundma mustreid, tegema järeldusi ning lahendama ülesandeid olemasolevate andmete põhjal[66]. Treenimist käsitletakse alapeatükis 1.6.2.

Närvivõrgud koosnevad omavahel ühendatud arvutuslikest elementidest: **neuronitest**, millel on üks või mitu sisendit ja üks väljund. Harilikult esitatakse sisendeid, väljundit ning ühendusi neuronide vahel reaalarvudena. Ühendusi neuronide vahel moodustatakse ühe neuroni väljundi suunamisel teise neuroni sisendisse. Igale sisendile on määratud kaal, mis väljendab sisendi olulisust neuroni väljundi arvutamisel. Neuronil on vabaliige, millega saab lisanduvalt mõjutada väljundi väärtust. Neuroni väljundit mõjutavaid muutujaid (sisendite kaale ja vabaliiget) nimetatakse kokkuvõtvalt **parameetriteks**[44][66].

Neuroni väljund arvutatakse **aktiveerimisfunktsiooniga**, mis on tavaliselt mittelineaarne. Aktiveerimisfunktsiooni mittelineaarsus võimaldab modelleerida mittelineaarseid süsteeme[66]. Aktiveerimisfunktsiooniks võib olla näiteks **mittenegatiivne lineaarfunktsioon** ($ReLU$): $\varphi(x) = \max(0, x)$.



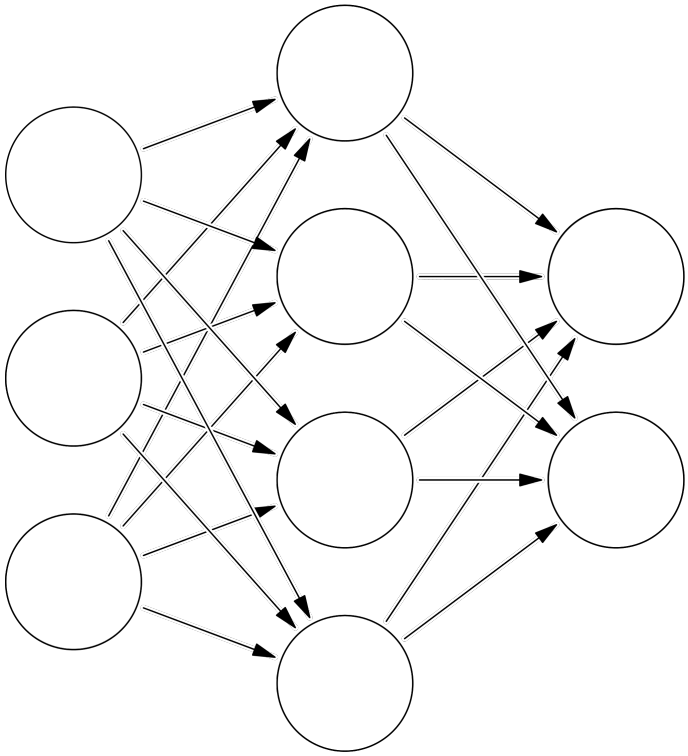
Joonis 12. Neuron.

Joonisel 12 on kujutatud tehisneuronit. Olgu $x_1, \ldots, x_n$ neuroni sisendid; $w_1, \ldots, w_n$ sisenditele vastavad kaalud; b vabaliige ning φ aktiveerimisfunktsioon. Neuroni väljund y avaldub kui

$$y = \varphi\left(b + \sum_{i=1}^{n} w_i x_i\right). \tag{2}$$

Neuroneid kombineeritakse kihtideks, mis võimaldab luua keerukaid arvutuslikke struktuure. Joonisel 13 on kujutatud lihtsamat tehisnärvivõrgu struktuuri. Antud juhul on neuronid kombineeritud **täielikult ühendatud kihtidesse** (ingl k *fully*

connected layer, lühendatult *dense*): iga neuroni sisenditeks on *kõikide* eelmise kihi neuronite väljundid. Ringid tähistavad neuroneid ning nooled tähistavad ühe neuroni väljundi ühendamist teise neuroni sisendiga. Igal neuronil on üks väljund, kuid seda korratakse iga järgneva kihi neuroni sisendi jaoks. Andmed läbivad sisendkihi (vasakul), ühe peidetud kihi (keskel) ja väljundkihi (paremal). Väljundkihi väljundid moodustavad närvivõrgu järelduse sisendandmete põhjal. Närvivõrgu järeldus sõltub närvivõrgu parameetritest. Täpne närvivõrgu struktuur sõltub lahendatavast probleemist.



Joonis 13. Lihtne tehisnärvivõrgu struktuur[66].

Närvivõrkude kontekstis saab eristada arhitektuuri ja mudelit. **Arhitektuur** on närvivõrgu kirjeldus, mis sisaldab informatsiooni lahendatava probleemi, sisendi ja väljundi, kihtide ja nende seotuse ning muude omaduste ja kasutatud meetodite kohta. **Mudel** on arhitektuuri isend ehk närvivõrgu instants, mis saadakse treenimise tulemusel.

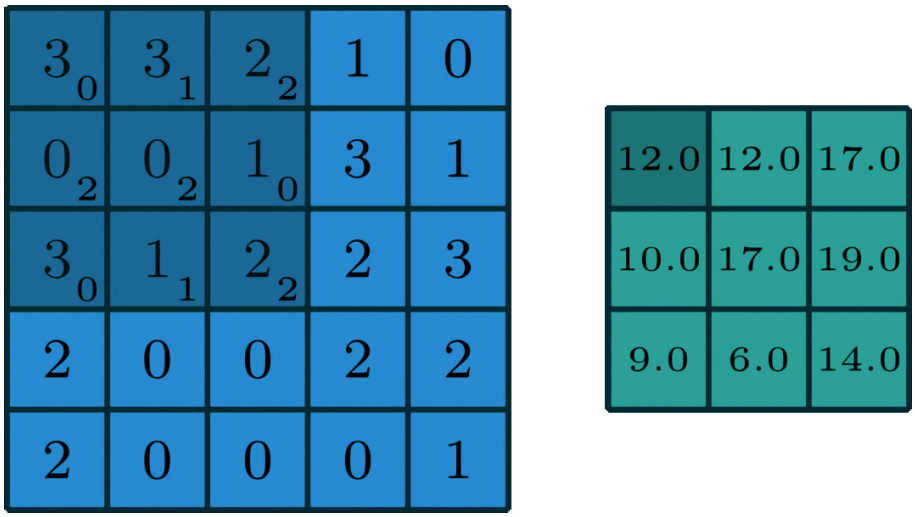
Käesolevas töös kasutatakse närvivõrke klassifitseerija arendamisel. Närvivõrgud treenitakse tuvastama mustreid sämplite ajaesituses ning mel-spektrogrammides, mille põhjal on võimalik anda hinnanguid heli kategooria kohta.

1.6.1 Konvolutsioonilised närvivõrgud

Konvolutsiooniline närvivõrk koosneb konvolutsioonilistest kihtidest, ahenduskihtidest ning täielikult ühendatud kihtidest. Konvolutsioonilise närvivõrgu sisendiks on tüüpiliselt n-mõõtmelise vektorina või $(m \times n)$-mõõtmelise maatriksina esitatavad andmed[69].

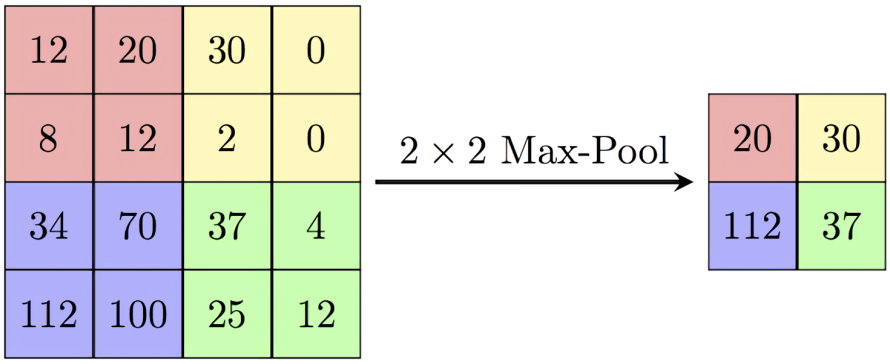
Konvolutsiooniline kiht (ingl k *convolutional layer*, lühendatult *conv*) kasutab väljundi arvutamiseks konvolutsiooni: operatsiooni, mille käigus fikseeritud mõõtmetega sisendeid teisendatakse konvolutsioonilise neuroni ehk **filtri** abil, kusjuures filter on väiksemate mõõtmetega kui sisend. Teisendamisel "libistatakse" filtrit samm-sammu haaval üle sisendi ning väljund saadakse sarnaselt valemiga (2): igal sammul arvutatakse kaalutud sisendelementide ja vabaliikme summa ning vastav väljundelement arvutatakse saadud summast aktiveerimisfunktsiooniga. Filtri väljundit nimetatakse ka **tunnuskaardiks** (ingl k *feature map*). Konvolutsioonilisel kihil võib olla mitu filtrit: sel juhul vastab igale filtrile üks tunnuskaart ning öeldakse, et konvolutsioonilisel kihil on mitu kanalit[43][56][69].

Joonisel 14 on kujutatud kahemõõtmelist konvolutsiooni maatriksina esitataval sisendil. Vasakul on kujutatud sisendit mõõtmetega (5×5), mida teisendatakse filtriga, mille mõõtmed on (3×3). Filtri kaalud on märgitud väiksemate arvudega. Paremal on kujutatud tunnuskaarti, mis saadakse filtri rakendamisel üle sisendi. Lihtsuse mõttes on filtri vabaliige 0 ning kasutatav aktiveerimisfunktsioon $\varphi(x) = x$. Analoogselt on võimalik rakendada ühemõõtmelist konvolutsiooni vektorina esitataval sisendil.



Joonis 14. Kahemõõtmelise konvolutsiooni rakendamine[52].

Ahenduskiht (ingl k *pooling layer*) vähendab sisendi mõõtmeid, kombineerides mitu sisendit üheks väljundiks. Protsess on sarnane konvolutsioonile: ka siin kasutatakse filtreid. Joonisel 15 on kujutatud näidet *max*-ahenduskihi (ingl k lühendatult *max-pool*) rakendamisest filtri mõõtmetega (2×2) sisendile mõõtmetega (4×4). *Max*-ahenduskihi väljundid saadakse sisendi (2×2)-suurustest mittekattuvatest aladest, valides igast alast maksimaalse väärtuse[10][43].



Joonis 15. *Max*-ahenduskihi rakendamine[10].

Konvolutsioonilised närvivõrgud säilitavad andmete ruumilisi ja ajalisi seoseid paremini kui ainult täielikult ühendatud kihtidest koosnevad närvivõrgud[69].

1.6.2 Treenimine

Närvivõrgu treenimine on protsess, mille käigus kohandatakse mudeli parameetreid selliselt, et mudeli tulemuslikkus probleemi lahendamisel paraneks. Treenimist teostatakse iteratiivse treeningalgoritmiga, esitades mudelile treeningandmestikku kuuluvaid andmepunkte koos oodatud väljunditega.

Mudeli vea mõõtmiseks defineeritakse **kaofunktsioon**, mille sisendiks on mudeli väljund ja oodatud väljund. Kui viga mõõdetakse klassifitseerijal, mille väljundiks on vektor kategooriate tõenäosusväärtustega lõigul $[0, 1]$, siis kasutatakse **ristentroopiakadu** (ingl k *cross-entropy loss*). Esitades klassifitseerijale vaatlustulemuse o, avaldub ristentroopiakadu järgmiselt:

$$\mathcal{L}_o\Big(\left(p_{o,1}, \ldots, p_{o,C}\right), \left(y_{o,1}, \ldots, y_{o,C}\right) \Big) = -\sum_{c=1}^{C} y_{o,c} \ln \left(p_{o,c}\right), \tag{3}$$

kus C on kategooriate arv; $p_{o,c}$ on klassifitseerija tõenäosusväärtus kategooria c jaoks vaatlustulemuse o põhjal; $y_{o,c}$ on binaarväärtus, mis tähistab vaatlustulemuse o teadaolevat kuuluvust kategooriasse c (0 ehk "ei kuulu" või 1 ehk "kuulub")[39].

Treenimine seisneb kaofunktsiooni minimeerimises treeningalgoritmiga. Treenimine jaotatakse kaheks faasiks – treeningfaasiks ja valideerimisfaasiks – ning treeningandmestikust eraldatakse osa, mida nimetatakse valideerimisandmestikuks. Treeningfaasis esitatakse mudelile treeningandmestikku kuuluvaid andmepunkte ning toimub mudeli parameetrite kohandamine ja kaofunktsiooni minimeerimine. Valideerimisfaasis hinnatakse mudeli kaofunktsiooni väärtust valideerimisandmestikul. Treenimise eesmärk on luua mudel, mis annaks häid tulemusi ka uute andmepunktide korral ehk millel oleks madal üldistusviga. Tavaliselt jätkatakse treenimist kuni kaofunktsiooni (lokaalse) miinimumi saavutamiseni valideerimisandmestikul. Mudeli üldistusvõimet hinnatakse testandmestikul, millel puudub ühisosa treening- ja valideerimisandmestikuga.

Minimeerimisprobleemi lahendamiseks on mitmeid algoritme. Kõige levinum optimiseerimisalgoritm on stohhastiline gradientlaskumine (ingl k *stochastic gradient descent*, lühendatult SGD). Treenitavale mudelile antakse sisendina juhuslikus järjekorras treeningandmestikku kuuluvaid andmepunkte. Teades mudeli väljundit ning oodatud väljundit, on võimalik arvutada kaofunktsiooni gradient mudeli parameetrite suhtes. Kaofunktsiooni minimeerimiseks muudetakse mudeli parameetreid gradiendile vastavalt. **Õpisamm** (sümbolina Δ) defineerib mudeli parameetrite muutumise määra optimiseerimisalgoritmi ühel iteratsioonil[9][44]. Leidub ka optimiseerimisalgoritme, näiteks Adagrad või Adadelta, mis kohandavad õpisammu vastavalt närvivõrgu erinevatele parameetrigruppidele.

Treeningalgoritmi igal iteratsioonil on treeningandmestikku kuuluvaid andmepunkte võimalik esitada kolmel moel: ükshaaval, miniplokkidena või korraga. Igal moodusel on oma eelised ja puudused. **Miniplokk** on treeningandmestiku jaotis, mille suurus on vabalt valitav. **Epohh** on üks treeningandmestiku täielik esitus treenitavale tehisnärvivõrgule[7].

Enne treenimist määratakse **hüperparameetrid**: tehisnärvivõrgu arhitektuuri või treenimisalgoritmi muutujad, mille hulka kuuluvad näiteks valitud kaofunktsioon, õpisamm, epohhide arv ja miniploki suurus. Hüperparameetrite muutmine võib oluliselt parandada treenimise kiirust ja närvivõrgu tulemuslikkust püstitatud probleemi lahendamisel.

1.6.3 Regulariseerimine

Kui treenimise käigus hakkab kaofunktsiooni väärtus valideerimisandmestikul suurenema, siis mudeli üldistusvõime väheneb ja öeldakse, et tegemist on ülesobitumisega treeningandmestikul. Ülesobitumise vältimiseks suurendatakse treeningandmestikku

või kasutatakse **regulariseerimist**: tehakse närvivõrgu arhitektuuri, treeningalgoritmi või -andmestiku muudatus, mille eesmärk on vähendada üldistusviga[60].

Varane lõpetamine (ingl k *early stopping*) seisneb treenimise lõpetamises, kui kaofunktsiooni väärtus valideerimisandmestikul ei ole langenud viimase valitud arvu epohhide jooksul. Seda arvu nimetatakse kannatlikkuseks (ingl k *patience*). Treenimise käigus salvestatakse mudel, mis saavutas madalaima kaofunktsiooni väärtuse valideerimisandmestikul.

Väljajätumeetod (ingl k *dropout method*, lühendatult *dropout*) vähendab ülesobitumist, vältides neuronite omavahelist adapteerumist. Treenimise käigus eiratakse juhuslikult valitud osa neuronitest, $p \in (0, 1)$, mis ei osale väljundi arvutamisel ja mille parameetreid ei kohandata[8].

Andmete rikastamine (ingl k *data augmentation*) suurendab treeningandmestikku. Rikastamisel muundatakse treeningandmestikku kuuluvaid andmepunkte. Sõltuvalt valitud meetodist muundatakse ka andmepunktile vastavat oodatud väljundit või see jäetakse samaks.

***Mix-up* meetod** on andmete rikastamise meetod, mille käigus kaks andmepunkti kombineeritakse üheks vastavalt juhuslikult valitud kaalule. Seda meetodit kasutatakse klassifitseerijate treenimisel. Matemaatiliselt võib seda meetodit kirjeldada järgmiselt:

$$\begin{aligned} \tilde{x} &= \lambda x_i + (1 - \lambda)\, x_j, \\ \tilde{y} &= \lambda y_i + (1 - \lambda)\, y_j, \end{aligned} \tag{4}$$

kus x on vaatlustulemus ja y sellele vastav oodatud väljund tõenäosusvektorina: (x_i, y_i) ja (x_j, y_j) on kaks kombineeritavat andmepunkti, $(\tilde{x}, \tilde{y})$ on kombineeritud andmepunkt ja $\lambda \in [0, 1]$ on beeta-jaotuse järgi juhuslikult valitud kaal[57].

Õpisammu ajastamine (ingl k *learning rate scheduling*) seisneb õpisammu muutmises treeningfaasis, mis võimaldab leida soodsamaid kaofunktsiooni lokaalseid miinimume ning millega välditakse parameetrite üleliigset muutumist lokaalse miinimumi juures.

Ülesobitumise vastandiks on alasobitumine, mis tähendab, et mudeli täpsust valideerimisandmestikul saaks veel suurendada. Alasobitumise vältimiseks on võimalik suurendada epohhide arvu, vähendada regulariseerimist või teha närvivõrgu arhitektuuri muudatus.

1.7 Kasutatud tehnoloogiad

Käesolevas töös kasutatakse programmeerimiskeelt Python 3, rakendust Jupyter Notebook, masinõppeteeki PyTorch ja helianalüüsiteeki LibROSA.

Python 3 on platvormiülene, interpreteeritud, dünaamilise tüüpimisega ja objektorienteeritud programmeerimiskeel[47]. Laialdane valik masinõppe-, andmetöötlus- ja helianalüüsiteeke teeb Pythonist hea valiku tehisnärvivõrkude arhitektuuride arendamiseks.

Jupyter Notebook on interaktiivne Pythoni arenduskeskkond koodi jooksutamiseks ja dokumenteerimiseks ning töö tulemuste salvestamiseks. Jupyter Notebook'i dokumentidesse (*notebook*'idesse) saab üheaegselt salvestada nii programmi lähtekoodi, kommentaare kui ka multimeediat, näiteks helifaile[28].

PyTorch on Pythoni masinõppeteek, mille iseloomulik funktsioonaalsus on automaatne diferentseerimine tensoritel. PyTorchi kontekstis on **tensorid** mitmemõõtmelised massiivid, mida on võimalik kasutada CUDA-toetusega graafikaprotsessoritel arvutuste paralleliseerimiseks[77]. CUDA on paralleelarvutuse platvorm ja programmeerimismudel, mis võimaldab rakendada graafikaprotsessoreid üldotstarbeliseks andmete töötlemiseks[45]. Automaatne diferentseerimine (ingl k *autograd*) on meetod funktsiooni gradiendite arvutamiseks, salvestades tensoritele rakendatud matemaatilised operatsioonid nende rakendamise järjekorras ning arvutades tensorite tuletisi salvestatule vastupidises järjekorras[48]. PyTorch loob tensorite ja automaatse diferentseerimise abil abstraktsioonikihi tehisnärvivõrkude defineerimiseks. Nii on võimalik teegi kasutajal luua keerukaid närvivõrkude arhitektuure, keskendumata matemaatilistele detailidele.

LibROSA on Pythoni helianalüüsiteek, mis pakub tööriistu muusikalise teabe otsingu süsteemide loomiseks[35]. LibROSA võimaldab hõlpsasti eraldada helisignaalide mel-spektrogramme ning ajalisi ja spektraalseid tunnuseid.

Valitud tehnoloogiate koostöö võimaldab kiiret prototüüpimist, säilitades paindlikkuse edasisteks laiendusteks.

1.7.1 PyTorchiga seotud mõisted

n-järku tensoreid esitatakse kujul $(t_0, t_1, \ldots, t_n)$, kus $t_i \mid i \in [0, n]$ on i-nda järgu massiivi pikkus. Vektori "mõõtme", maatriksi "järgu" või "mõõtmete" ja tensori "järgu" asemel kasutatakse ka sõna **"kuju"**.

Lamendamine (ingl k *flattening*, lühendatult *flatten*) on teisendus, mis loob n-järku tensori vektoresituse ehk 1-järku tensori. Tüüpiliselt kasutatakse seda teisendust pärast konvolutsioonilisi kihte ja enne täielikult ühendatud kihte.

Ühendamine ("konkatenatsioon", ingl k *concatenation*, lühendatult *concat*) on teisendus, mis liidab kokku mitu n-järku tensorit ja tagastab n-järku tensori. Ühendamisel valitakse tensori mõõde, mille järgi ühendamine toimub. Tüüpiliselt kasutatakse seda teisendust enne täielikult ühendatud kihte, et liita kokku mitu väljundit eelmistest kihtidest.

2 Olemasolevad tööd

Selles peatükis tuuakse näiteid helide automaatse klassifitseerimisega seotud varasematest uurimistöödest ja arhitektuuridest. Kirjeldatakse ka olemasolevaid rakendusi, mis lahendavad käesolevas töös püsitatuga sarnast probleemi.

2.1 Uurimistööd ja andmestikud

Heliga seonduvalt on närvivõrkude kasutamist uuritud kõne[59], kõneleja[46], heliteoste[11], heliteoste meeleolu[13], helistike[30], muusikažanrite[15][38] ja instrumentide[25][51] tuvastamiseks ning keskkonnahelide[24][58] ja üldiste helide[32][89] klassifitseerimiseks.

Helisämplite klassifitseerimist käsitlevad tööd ei ole kasutanud närvivõrke ning on piirdunud vaid trummihelide klassifitseerimisega. Herrera *et al* uurisid trummihelide klassifitseerimist kasutades k-lähima naabri meetodit, statistilist modelleerimist ja otsustuspuid[23]. Deolekar ja Abraham uurisid tablalöökide klassifitseerimist otsustuspuu ja otsustusmetsa abil[14]. Shier *et al* uurisid basstrummi ja soolotrummi sämplite klassifitseerimist statistiliste tunnuste alusel, kasutades peakomponentide analüüsi[53].

Suur osa vabalt kättesaadavaid heliandmestikke kuulub samuti eelnevalt mainitud kategooriatesse: kõne[4][40][42], üldised helid[19][20], keskkonnahelid[22] ja heliteosed[6][12].

Ühe erandina võib välja tuua Magenta NSynth andmestiku, mis koosneb üksikutest muusikalistest nootidest ja sisaldab näiteid üle tuhande erineva instrumendi kohta[17]. Käesolevas töös esitatud definitsiooni järgi koosneb NSynth andmestik küll sämplitest, kuid andmestikus ei sisaldu näiteks vokaale, rütmiinstrumente ega heliefekte, mistõttu jääb see püstitatud ülesande kontekstis piiratuks.

Üldiselt võib öelda, et helisämplite automaatne klassifitseerimine on vähe uuritud valdkond. Puudu on ka suure kategooriate arvuga ja suure variatsiooniga helisämplite andmestik, mille alusel klassifitseerijaid võrrelda.

2.2 Arhitektuurid

Konkreetselt helisämplite klassifitseerimise probleemi lahendavaid närvivõrkude arhitektuure ei leitud. Eeskuju on võimalik võtta arhitektuuridest, mis lahendavad keskkonnahelide või üldiste helide klassifitseerimise probleemi või mis suudavad samaaegselt klassifitseerida muusikat, kõne ja müra.

(a) Lee *et al* uurisid kahe erineva ühemõõtmelist konvolutsiooni rakendava arhitektuuri võimekust muusika-, kõne- ja keskkonnahelide klassifitseerimisel. Ühemõõtmelist konvolutsiooni rakendati helide ajaesitusele, kasutades väiksemõõtmelisi filtreid. Autorite poolt pakutud arhitektuur SampleCNN saavutas kaasaegse parima lahendusega võrreldavaid tulemusi[33][34].

(b) Hussain ja Haque pakkusid varieeruva kestusega kõne, muusika ja mürade klassifitseerimiseks ja segmenteerimiseks arhitektuuri SwishNet, mis kasutab ühemõõtmelisi konvolutsioone helide mel-sageduse kepstri kordajatel[26].

(c) Schindler *et al* uurisid keskkonnahelide klassifitseerimist paralleelarhitektuuril põhineva konvolutsioonilise närvivõrguga, kasutades sisendina erinevate resolutsioonidega mel-spektrogramme. Spektrogrammidele rakendati paralleelselt erinevate mõõtmetega filtreid. Mitme resolutsiooni kasutamine spektrogrammides ja erinevate mõõtmetega filtrite kasutamine võimaldas paremini tuvastada helide tunnuseid[50].

(d) Xu, Zhu *et al* uurisid ansambelõppe ja statistiliste tunnuste kasutamist üldiste helide klassifitseerimisel. Pakutud arhitektuur kasutas seitset konvolutsioonilist arhitektuuri, helide mel-spektrogramme ja mel-kepstri kordajaid ning ajaesitust koos selle statistiliste tunnustega. Statistiliste tunnuste kasutamine saavutas paremaid tulemusi kui nende mittekasutamine[89].

(e) Abdoli *et al* uurisid ühemõõtmelise konvolutsiooni kasutamist keskkonnahelide klassifitseerimisel, kasutades treenitavaid filtreid. Pakutud arhitektuur on suuteline töötlema varieeruva kestvusega ajaesitusi, kasutades niinimetatud libisevat akent, mis jaotab sisendi fikseeritud mõõtmetega osadeks. Kasutades erinevaid kombinatsioonreegleid, leitakse üksikute osade klassifitseerimistulemustest esialgse sisendi lõplik tulemus. Töös uuriti heli kestuse mõju klassifitseerija tulemuslikkusele: ühesekundilised sisendid saavutasid sama häid tulemusi kui kasutatud andmestiku täiskestusega sisendid, mis olid ligikaudu kolmesekundilised. Pakutud lahendus saavutas parima tulemuse võrreldes teiste sama probleemi lahendavate mudelitega[1].

(f) Lederle ja Wilhelm uurisid konvolutsioonilist arhitektuuri üldiste helide klassifitseerimiseks, mis kombineeris helide ajaesitused ja mel-spektrogrammid. Kombineeritud sisendiga arhitektuur andis paremaid tulemusi kui ainult ajaesituse või mel-spektrogrammide kasutamine[32].

(g) Zhang, Xu *et al* uurisid keskkonnahelide klassifitseerimist sügava konvolutsioonilise närvivõrguga, kasutades *mix-up* meetodil rikastatud mel-spektrogramme. Sisendite rikastamine võimaldas treenida täpsema klassifikaatori[58].

(h) Huang ja Leanos pakkusid keskkonnahelide klassifitseerimiseks täielikult konvolutsioonilisel närvivõrgul põhineva arhitektuuri AclNet, mis saavutas kasutatud andmestiku klassifitseerimisel parema täpsuse kui inimene[24].

Konvolutsioonilisi närvivõrke on võimalik edukalt kasutada helide klassifitseerimiseks. Sõltuvalt püstitatud probleemist annab paremaid tulemusi varieeruvate mõõtmetega filtrite kasutamine, varieeruva resolutsiooniga mel-spektrogrammide kasutamine, ajaesituse ja mel-spektrogrammide paralleelne kasutamine, statistiliste tunnuste kasutamine, andmete rikastamine ja *mix-up* meetodi kasutamine.

Käesolevas töös esitatud lahenduse klassifitseerimistulemusi võrreldakse mõne eelpool kirjeldatud arhitektuuril põhineva klassifikaatori tulemustega *kvantitatiivselt*.

2.3 Rakendused

Leidub sämplite haldamise rakendusi, mis lahendavad sämplite klassifitseerimise probleemi, kuid mis on suletud lähtekoodiga ja tasulised. Kättesaadavat informatsiooni rakendustes kasutatud klassifitseerijate realisatsioonide kohta ei leidu.

(a) Üheks selliseks rakenduseks on Sononym[1], mis "kasutab masinõppe meetodeid helisämplite organiseerimiseks ja otsimiseks"[54]. Programmi uurides ei ole võimalik teha oletusi meetodite kohta, mida Sononym kasutab sämplite klassifitseerimiseks. Küll aga võib rakenduse krüpteerimata SQLite andmebaasifaili ja logifailide põhjal oletada, et rakendus kasutab kahte mudelit. Esimene mudel eristab ***loop***'e (pikemaid või tsüklilisi sämpleid) ja ***one-shot***'e (lühemaid või mittetsüklilisi sämpleid) ning teine klassifitseerib *one-shot*'e. Andmebaasi tabelitesse on salvestatud BLOB'e ehk binaarandmeid, mis viitavad helifailide täiendavale töötlemisele ja tunnuste eraldamisele. Iga sämpli klassifitseerimistulemus salvestatakse kategooriate tõenäosusvektorina. Sämp-

[1] https://www.sononym.net/

lile määratavad kategooriad valitakse tõenäosusvektori järgi, kui kategooriale vastav tõenäosusväärtus ületab fikseeritud lävendit.

(b) Teine leitud rakendus on Samplism[1], mille peamiseks eripäraks on kolmanda osapoole sämplite ja tehnoloogiate toetus: programm suudab märgendada sämpleid, mis kuuluvad tuntud sämplikogumitesse või mis on salvestatud vähemlevinud vormingutes. Erinevalt Sononymist, mis on platvormiülene, toetab Samplism vaid macOS operatsioonsüsteemi[3]. Samplism tekitab samuti sämplite analüüsimise tulemusena krüpteerimata SQLite andmebaasifaili. Andmebaasis sisalduvad lisaks helifailide tehnilistele metaandmetele vaid tundmatud arvulised märgendid, mistõttu ei ole võimalik andmebaasifaili põhjal teha oletusi Samplismi poolt kasutatavate klassifitseerimismeetodite kohta. Programmi kasutades ja sämplitele määratud kategooriaid uurides võib arvata, et Samplism kasutab klassifitseerimisel faili nimes sisalduvaid märksõnu. Märksõnadeta failinimedega sämpleid laadides ei suuda programm kategooriaid määrata.

Käesolevas töös esitatud lahenduse klassifitseerimistulemusi võrreldakse eelpool kirjeldatud rakenduste tulemustega *kvalitatiivselt*.

[1] https://audiohelperproject.com/products/samplism/

3 Eksperimendid

Selles peatükis esitatakse töö eksperimentaalne osa. Töö eesmärgiks on luua fikseeritud sisendmõõtmetega ja ühe väljund-kategooriaga helisämplite klassifitseerija. Probleemi lahendamiseks arendatakse välja konvolutsioonilise närvivõrgu arhitektuur.

3.1 Helifailide algandmestik

Suure variatsiooni ja kategooriate arvuga helisämplite andmestiku puudumise tõttu oli vajalik koostada uus andmestik. Andmestiku koosseisu kuuluvad nii tasuta kättesaadavad sämplid kui ka tasulistesse kogumikesse kuuluvad sämplid. Kokku koguti $n = 20000$ sämplit, mis märgendati käsitsi $C = 31$ kategooriasse. Sämplite kategoriseerimisel lähtuti valdkonnas levinud tavadest: valdavalt on eeskuju võetud tasulistes kogumikes kasutatud sämplite liigendamise ja nimetamise skeemidest. Andmestikku kuuluvate sämplite jaotus kategooriate järgi peegeldab tegelikku jaotust kasutatud kogumikes. Valdavalt kuuluvad andmestikku erinevat tüüpi rütmiinstrumendid. Andmestiku kogukestus on 10 h 12 min 48 s.

Tabelis 1 on loetletud andmestikus esindatud kategooriad koos selgituste ja indeksitega c_i. Tabeli parempoolses tulbas on kategooriaga c_i seotud sämplite arv n_{c_i}.

Andmestikku kuuluvate sämplikogumike kasutuslitsentsides sätestatud piirangute tõttu ei ole autoril õigust jagada algseid helifaile kolmanda osapoolega.

3.2 Arendusandmestikud

Klassifikaatorite arendamiseks koostatakse hulk erinevaid andmestikke.

Põhiandmestik koosneb algandmestiku sämplitest, mida töödeldakse andmemahu ja edasise arvutusmahu vähendamise eesmärgil. Esiteks fikseeritakse helisämplid diskreetimissagedusele 22 050 Hz ja kvantimissügavusele 16 bitti. Need teisendused ei vähenda helide subjektiivset eristusvõimet, kuna valdav osa muusikaliste helide sageduslikust energiast asub allpool ~ 11 kHz ning 16 bitti on sämplite helivaljususe

Tabel 1. Helisämplite algandmestiku kategooriline jaotus.

c_i	**Kategooria**	**Selgitus**	n_{c_i}
0	*atmosphere*	Sünteesitud ja salvestatud "meeleoluhelid"	264
1	*bongo conga*	Akustilised ja elektroonilised bongo- ja konga-trummid	364
2	*breaks*	Salvestatud trummipartiid ehk "breigid"	272
3	*cajon*	Akustilised cajón-trummid	19
4	*chimes*	Akustilised helinad ja kellamäng	20
5	*clap*	Salvestatud, sünteesitud ja muundatud plaksud	1519
6	*cowbell*	Akustilised ja elektroonilised lehmakellad	49
7	*crash*	Akustilised ja elektroonilised *crash*-taldrikud	272
8	*djembe*	Akustilised džembe-trummid	78
9	*electric bass*	Elektrilised basskitarrid	298
10	*electric piano*	Elektrilised klaverid	89
11	*guitar*	Akustilised ja elektrilised kitarrid	693
12	*hihat*	Akustilised ja elektroonilised *hi-hat*-taldrikud	3232
13	*impact*	Elektroonilised impulss-heliefektid	195
14	*kick*	Elektroonilised basstrummid	5966
15	*organ*	Elektrilised orelid	20
16	*pads*	Efektidega süntesaatorihelid	301
17	*perc*	Muud kategoriseerimata rütmiinstrumendid	1302
18	*piano*	Akustilised klaverid	298
19	*rhodes*	Rhodes-tüüpi elektrilised klaverid	50
20	*ride*	Akustilised ja elektroonilised *ride*-taldrikud	329
21	*saxophone*	Salvestatud saksofonid	27
22	*shaker*	Akustilised ja elektroonilised *shaker*'id	342
23	*snap*	Salvestatud ja muundatud nipsud	250
24	*snare*	Akustilised ja elektroonilised soolotrummid	2492
25	*strings*	Akustilised keelpillid	296
26	*tabla*	Salvestatud tabla-trummid	81
27	*tambourine*	Sünteesitud ja salvestatud tamburiinid	86
28	*tom*	Akustilised, sünteesitud ja muundatud *tom*-trummid	656
29	*trumpet*	Salvestatud ja võimendamata trompetid	30
30	*vocal*	Kõne, laul ja vokaalefektid	110
Kokku			20000

dünaamika esitamiseks piisav. Teiseks summeeritakse kanalid monosse, kuna valdav osa sämpleid sisaldab vähe ruumilise heli informatsiooni, mis ei paranda subjektiivset eristusvõimet. Viimaks normaliseeritakse sämplid amplituudi järgi, et need oleksid ühtlasema valjususega. Selliselt töödeldud sämplid moodustavad põhiandmestiku, mis koosneb sämplite ajaesitustest ja nendele vastavatest täisarvulistest kategooriatest. Kategooriad on esitatud täisarvulise kodeeringuna, kuna PyTorchi vaikimisi ristentroopia-kaofunktsioon nõuab oodatud väljundina täisarve.

Põhiandmestiku uurimisel selgus, et üle kolmveerandi sämplitest on kestusega poolteist sekundit või vähem. Seetõttu kasutatakse edaspidi fikseeritud kestusega $d = 1.5\,\mathrm{s}$ sämplite ajaesituste andmestikku D. Kui sämpli $d < 1.5\,\mathrm{s}$, lisatakse ajaesituse lõppu vaikust kuni soovitud kestuseni. Kui sämpli $d > 1.5\,\mathrm{s}$, valitakse lõik ajaesituse algusest kuni soovitud kestuseni.

Põhiandmestiku andmepunktide indeksid $i \in [0, n-1]$ jaotatakse kolmeks hulgaks indeksiteks, mis vastavad treening- (indeksiga $train$), valideerimis- ($valid$) ja testandmestikkudele ($test$). Treeningandmestik D_{train} moodustab 60 protsenti ($|D_{train}| =$ 12 000 sämplit), valideerimisandmestik D_{valid} 20 protsenti ($|D_{valid}| = 4000$) ja testandmestik D_{test} 20 protsenti ($|D_{test}| = 4000$) põhiandmestikust D. Igas andmestikus on olemas sämpleid igast kategooriast.

Klassifitseerijate sisendina kasutatakse mel-spektrogrammide andmestikku S, mis luuakse fikseeritud kestusega sämplite D põhjal. Spektrogrammide saamiseks kasutatakse lühikese kestusega diskreetset Fourier' teisendust (ingl k *Short Time Fourier Transform*, STFT), mille akna suuruseks (ingl k *window size*) valitakse 20 ms ehk 441 diskreeti ja hüppe pikkuseks (ingl k *hop length*) 10 ms ehk 220 diskreeti. Spektrogrammide sagedusliku informatsiooni teisendamisel kasutatakse mel-filtripanka 64 filtriga. Tulemuseks on $(1, 64, 151)$-kujuliste tensoritena esituvad ühe kanaliga mel-spektrogrammid. Suuruste valimisel lähtuti teadmisest, et sämplid on lühikesed: hea ajalise eraldusvõime saavutamiseks on kasutusel väikse kestusega STFT-aknad ning hea sagedusliku eraldusvõime saavutamiseks suur arv mel-filtripankasid.

Hiljem proovitake klassifitseerimistulemuste parandamiseks kasutada sämplite ajaliste ja spektraalsete tunnuste andmestikku E. Iga fikseeritud kestusega sämpli kohta arvutatakse 4 tunnust: nullist üleminekute sageduse muutumine ajas, spektraaltsendroidi muutumine ajas, spektraalse lameduse muutumine ajas ja spektraalse murdesageduse muutumine ajas. Spektraalsete tunnuste eraldamisel valitakse STFT-akna suuruseks 2048 diskreeti. Tunnuseid arvutatakse perioodiliselt 16 korda sekundis. Kuna sämplid on fikseeritud kestusega $d = 1.5\,\mathrm{s}$, on tulemuseks on tunnus-

te $(4, 24)$-kujulised tensorid. Suuruste valimisel lähtuti oletusest, et tunnuste täpsem eraldamine ei ole oluline, kuna tunnused varieeruvad lühikeste sämplite piires vähe.

Klassifitseerija tulemuste parandamiseks proovitakse kasutada ka D_{train} rikastatud sämplite andmestikku (indeksiga a, $|D_a| = 6806$). Helisämpleid muundatakse juhuslikult neljal erineval moel. Esimene meetod lisab sämplitele Gaussi müra amplituudiga vahemikus $[0.002, 0.005]$. Teine meetod venitab sämpleid kordajaga vahemikus $[0.9, 1.1]$, muutmata helikõrgust ning fikseerides uuesti sämpli kestuse $d = 1.5\,\text{s}$. Kolmas meetod muudab sämpli helikõrgust vahemikus $[-2, 2]$ pooltooni, fikseerides uuesti sämpli kestuse $d = 1.5\,\text{s}$. Neljas meetod nihutab sämpli ajaesituse diskreetide väärtuseid vahemikus $[0\%, 50\%]$ sämpli täiskestusest. Iga kategooria c_i kohta luuakse $\max(0, \frac{|D_{train}|}{C} - n_{c_i,train})$ rikastatud sämplit, kus $n_{c_i,train}$ on kategooriale c_i vastav sämplite arv treeningandmestikus. Iga kategooria jaoks luuakse juurde nii mitu sämplit, kui palju on puudu kategooriaülesest sämplite arvu aritmeetilisest keskmisest. Kui mõnesse kategooriasse kuulub keskmine või üle keskmise arv sämpleid, siis selle kategooria jaoks rikastatud sämpleid ei looda. Sarnaselt eelmainitule luuakse rikastatud sämplite mel-spektrogrammide andmestik S_a.

Kuna eelpool kirjeldatud rikastatud andmestik on suhteliselt väiksemahuline, suurendades treeningandmestikku vaid ligikaudu poole võrra, siis luuakse ka suuremahulisem rikastatud andmestik (indeksiga A, $|D_A| = 98\ 980$). Rakendatud meetodid on sarnased eelmainitule, kuid parameetrid on erinevad. Esimene meetod lisab sämplitele Gaussi müra amplituudiga vahemikus $[0.003, 0.005]$. Teine meetod venitab sämpleid kordajaga vahemikus $[0.8, 1.2]$. Kolmas meetod muudab sämpli helikõrgust vahemikus $[-3, 3]$ pooltooni. Neljas meetod nihutab sämpli ajaesituse diskreetide väärtuseid vahemikus $[0\%, 25\%]$ sämpli täiskestusest. Iga kategooria c_i kohta luuakse $\max_j(n_{c_j,train}) - n_{c_i,train}$ rikastatud sämplit, kus $\max_j(n_{c_j,train})$ on treeningandmestiku suurima sämplite arvuga kategooriasse kuuluvate sämplite arv. Iga kategooria jaoks luuakse juurde nii mitu sämplit, kui palju on puudu maksimaalse sämplite arvuga kategooria sämplite arvust. Tulemusena on igas kategoorias võrdne arv sämpleid. Sarnaselt eelmainitule luuakse rikastatud sämplite mel-spektrogrammide andmestik S_A.

3.3 Arhitektuuri arendamine

Konvolutsioonilise arhitektuuri väljatöötamine on iteratiivne protsess, mis seisneb vaheldumisi uute mudelite treenimises ja tulemuste võrdlemises baasmudeli ja vahepealsete mudelite tulemustega.

Mudelite treenimisel kasutatakse miniplokktreeningut: iga treenitava mudeli sisendiks on treeningandmestiku jaotised suurusega 32 andmepunkti. Minimeeritav kaofunktsioon on ristentroopiakadu. Kõikide mudelite treeningalgoritm jääb põhiosas samaks, põhinedes baasmudeli treeningalgoritmil.

Treeningalgoritm on jaotatud kahte faasi: treening- ja valideerimisfaasi. Treeningfaasis toimub närvivõrgu kaalude kohandamine ja kaofunktsiooni minimeerimine. Valideerimisfaasis hinnatakse kaofunktsiooni väärtust valideerimisandmestikul. Jaotus kaheks faasiks on oluline, sest treenimise käigus saavutatud tulemused valideerimisandmestikul annavad aimu närvivõrgu tulemuslikkusest klassifitseerimisülesande lahendamisel. Sõltuvalt arhitektuurist kasutatakse treenimisel erinevaid optimiseerimise algoritme ja regulariseerimise meetodeid, mistõttu täieneb ka treeningalgoritm. Treenitud mudelite hindamisel kasutatakse testandmestikku.

Arhitektuure kirjeldavates tabelites kasutatakse lühendeid ja sümboleid, mis on defineeritud töö tausta kirjeldavas peatükis. Tabeli ülemisse ossa on märgitud kasutatud sisendandmestikud. Tabeli keskosas on loetletud arhitektuuri kihid ja nende väljundite kujud. Kihtide väljundite kujud aitavad mõista kihtide funktsioone sisendandmete töötlemisel. Tensorite voog toimub ülevalt alla ehk andmed läbivad kihte selles järjekorras, nagu kihid on loetletud. Tabelisse on märgitud ka arhitektuuri treenitavate parameetrite arv.

Mudeleid kirjeldavatesse tabelitesse märgitakse kasutatud optimiseerija ja selle parameetrid, treenitud epohhide arv, treenimise aeg ja testandmestikul saavutatud klassifitseerimistulemuste hinnangud. Statistiliste kriteeriumidena on kasutusel täpsus (ingl k *accuracy*), kategoorilise kordustäpsuse keskmine (ingl k *macro precision*, lühendatult k-täpsus), kategoorilise saagise keskmine (ingl k *macro recall*, k-saagis) ja kategoorilise F_1-skoori keskmine (ingl k *macro F1-score*, k-F_1). Kategoorilise kriteeriumi keskmine saadakse vastava kriteeriumi väärtuse arvutamisel iga kategooria jaoks eraldi, misjärel võetakse saadud väärtuste aritmeetiline keskmine. Sellise keskmise alusel on võimalik hinnata mudeli tulemuslikkust, arvestamata andmepunktide ebaühtlast jaotust üle kategooriate. Täpsus väljendab mudeli tulemuslikkust ebaühtlase kategoorilise jaotusega andmestikul; sellisel, mis on kasutusel ka käesolevas töös. Kuna eksperimentides on treenitavaid mudeleid mitu, siis tuuakse tabelis välja mitme mudeli koondtulemused ehk statistiliste kriteeriumide aritmeetiline keskmine ja standardhälve.

Mudelite treenimiseks kasutatakse personaalarvutit protsessoriga AMD Ryzen 7 2700X ja graafikakaardiga Nvidia GeForce RTX 2080.

3.3.1 Baasarhitektuur

Käesoleva töö baasarhitektuuriks on valitud sügav närvivõrk *DNN-Baseline*. Selle arhitektuuri põhjal treenitud mudelid annavad algse tulemuse, mille põhjal hilisemaid tulemusi võrrelda.

DNN-Baseline arhitektuuri kirjeldab tabel 2. Arhitektuur koosneb ühest lamendavast kihist ja neljast täielikult ühendatud kihist. Lamendavas kihis teisendatakse mel-spektrogrammide 3-järku tensorid 1-järku tensoriteks. Aktiveerimisfunktsioonina kasutatakse $ReLU$.

Mudeleid treeniti kolm korda 50 epohhiga, kasutades SGD optimiseerijat õpisammuga $\Delta = 0.001$. Mudelite koondtulemused on toodud tabelis 3. Ühe mudeli kaofunktsiooni väärtuste graafik on esitatud joonisel 16. Must joon tähistab kaofunktsiooni väärtust treeningandmestikul, punane joon valideerimisandmestikul.

Tabel 2. *DNN-Baseline* arhitektuur.

Sisend	S_{train}
Kiht	**Väljund**
flatten	(9664)
dense1, relu	(1024)
dense2, relu	(1024)
dense3, relu	(1024)
dense4	(C)
Parameetrite arv	12 027 935

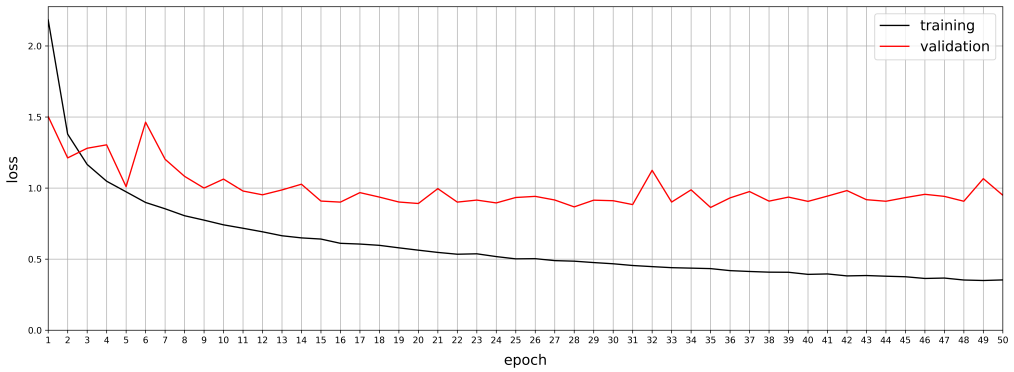
Tabel 3. *DNN-Baseline* mudelite koondtulemused.

Optimiseerija			SGD, $\Delta = 0.001$
Epohhe			50
k-Täpsus	**k-Saagis**	**k-F_1**	**Täpsus**
$48.3 \pm 2.2\%$	$44.6 \pm 1.5\%$	$44.6 \pm 1.0\%$	$74.8 \pm 1.4\%$

3.3.2 Konvolutsiooniline baasarhitektuur

Algseks konvolutsiooniliseks arhitektuuriks luuakse *CNN-Baseline*, mida kirjeldab tabel 4. Arhitektuur koosneb kahest konvolutsioonilisest kihist, kahest ahenduskihist ja kolmest täielikult ühendatud kihist. Selle arhitektuuri ja ka järgmiste arhitektuuride sisendiks on mel-spektrogrammide 3-järku tensorid.

Sarnaselt baasmudelitele treeniti *CNN-Baseline* mudeleid kolm korda 50 epohhiga. Mudelite koondtulemused on toodud tabelis 5. Tulemused on võrreldes *DNN-*



Joonis 16. *DNN-Baseline* mudeli kaofunktsiooni väärtuste graafik.

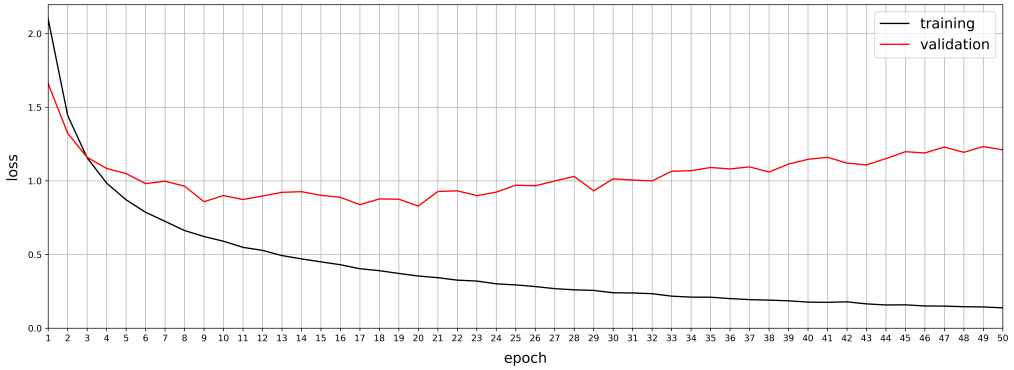
Baseline baasmudelite tulemustega kehvemad. Ühe mudeli kaofunktsiooni väärtuste graafik on esitatud joonisel 17. Graafikut uurides võib täheldada, et mudel hakkas ülesobituma umbes pärast 20. epohhi. 50. epohhi ajaks oli kaofunktsiooni väärtus jõudnud kahaneda oma miinimumini ning kasvada esimestel epohhidel saavutatud kaofunktsiooni väärtuseni. Kaofunktsiooni väärtus treeningandmestikul kahanes monotoonselt. Tegemist on ülesobitumisega treenimisandmestikul. Ülesobitumise vältimiseks on vaja suurendada treeningandmestikku või kasutada regulariseerimismeetodeid.

Tabel 4. *CNN-Baseline* arhitektuur.

Sisend		S_{train}
Kiht	**Filter**	**Väljund**
conv1, relu	$(5,5) \times 6$	$(6, 60, 147)$
max-pool	$(2,2)$	$(6, 30, 73)$
conv2, relu	$(5,5) \times 16$	$(16, 26, 69)$
max-pool	$(2,2)$	$(16, 13, 34)$
flatten		(7072)
dense1		(120)
dense2		(84)
dense3		(C)
Parameetrite arv		864 131

Tabel 5. *CNN-Baseline* mudelite koondtulemused.

Optimiseerija			SGD, $\Delta = 0.001$
Epohhe			50
Treenimise aeg			$4\,\text{min}\ 44\,\text{s} \pm 1\,\text{s}$
k-Täpsus	**k-Saagis**	**k-F_1**	**Täpsus**
$46.0 \pm 1.8\%$	$44.5 \pm 1.5\%$	$43.6 \pm 1.2\%$	$73.9 \pm 1.1\%$



Joonis 17. *CNN-Baseline* mudeli kaofunktsiooni väärtuste graafik.

3.3.3 Varane lõpetamine

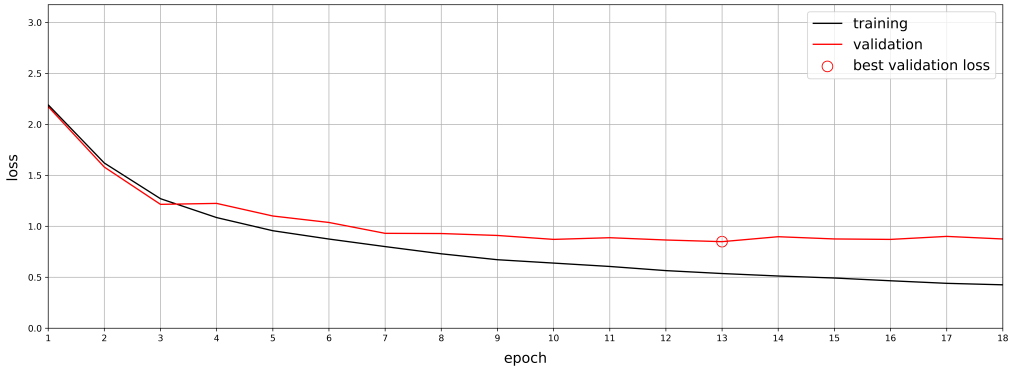
CNN-EarlyStop on konvolutsioonilise baasarhitektuuri edasiarendus, mis kasutab treeningalgoritmis varast lõpetamist kannatlikkusega 5 epohhi. Treenimise lõppedes salvestatakse valideerimisandmestikul madalaima kaofunktsiooni väärtuse saavutanud mudel.

CNN-EarlyStop kolme mudeli koondtulemused on toodud tabelis 6. Treenimine lõpetati enne sätestatud maksimaalset epohhide arvu, millega saavutati treenimisaja kokkuhoid, kusjuures treenitud mudeli täpsus oli sarnane konvolutsioonilise baasmudeliga. Teised tulemused on võrreldes baasmudelite tulemustega kehvemad. Tulemusi võiks parandada kannatlikkuse suurendamine. Kõikides järgnevates arhitektuurides kasutatakse varast lõpetamist.

Tabel 6. *CNN-EarlyStop* mudelite koondtulemused.

Optimiseerija			SGD, $\Delta = 0.001$
Kannatlikkus			5
Epohhe			16 ± 3
Treenimise aeg			2 min 33 s $\pm$ 12 s
k-Täpsus	**k-Saagis**	**k-F_1**	**Täpsus**
$43.2 \pm 2.1\%$	$39.3 \pm 2.9\%$	$39.4 \pm 2.2\%$	$73.1 \pm 1.8\%$

Joonisel 18 on kujutatud ühe mudeli kaofunktsiooni väärtuste graafikut. Punane ring tähistab epohhi, mille lõppedes saavutas mudel madalaima kaofunktsiooni väärtuse valideerimisandmestikul.



Joonis 18. *CNN-EarlyStop* mudeli kaofunktsiooni väärtuste graafik.

3.3.4 Ristkülikukujulised filtrid

CNN-Rectangular on uuendatud arhitektuur, mis kasutab ristkülikukujulisi (3×4)-filtritega konvolutsioonilisi kihte ja (2×2)-ahenduskihte vaheldumisi. Järjestikuste konvolutsiooniliste kihtide arv ja kanalite arv on seatud suuremaks. Suurema arvu kanalite kasutamine peaks võimaldama närvivõrgul paremini õppida seoseid erinevate sämplite mel-spektrogrammides. Ristkülikukujuliste filtrite kasutamine peaks suutma püüda rohkem ajalisi tunnuseid. *Max*-ahenduskihtide kasutamine konvolutsiooniliste kihtide järel vähendab andmete mõõtmeid ja peaks vähendama ajalissageduslikku tundlikkust tunnuste eraldamisel. Regulariseeriva kihina on lisatud ka väljajätukiht. Arhitektuuri on kujutatud tabelis 7.

Tabel 7. *CNN-Rectangular* arhitektuur.

Sisend		S_{train}
Kiht	**Filter**	**Väljund**
conv1, relu	$(3, 4) \times 32$	$(32, 62, 148)$
max-pool	$(2, 2)$	$(32, 31, 74)$
conv2, relu	$(3, 4) \times 64$	$(64, 29, 71)$
max-pool	$(2, 2)$	$(64, 14, 35)$
conv3, relu	$(3, 4) \times 128$	$(128, 12, 32)$
max-pool	$(2, 2)$	$(128, 6, 16)$
conv4, relu	$(3, 4) \times 128$	$(128, 4, 13)$
max-pool	$(2, 2)$	$(128, 2, 6)$
flatten		(1536)
dense1, relu		(256)
dropout, $p = 0.25$		(256)
dense2		(C)
Parameetrite arv		721 663

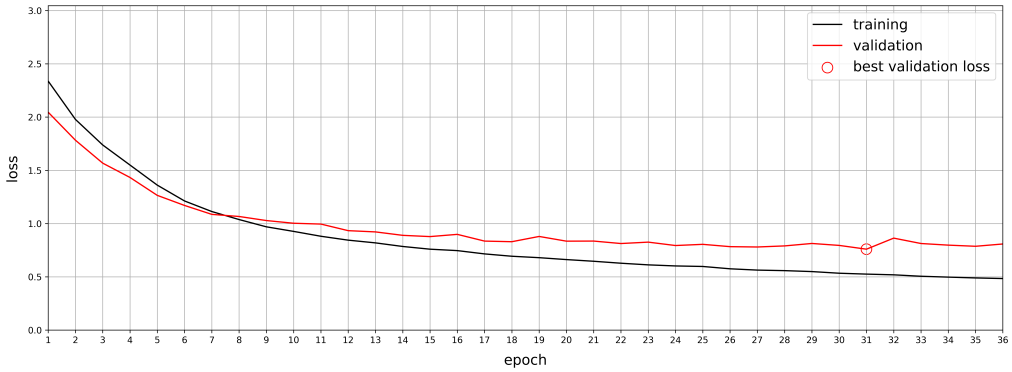
Tabelis 8 on märgitud kolme mudeli koondtulemused ja joonisel 19 on kujutatud ühe mudeli kaofunktsiooni väärtuste graafikut. Võrreldes arhitektuuridega *CNN-*

Baseline ja *CNN-EarlyStop* on kõikide kriteeriumide väärtused paranenud, v.a k-saagise väärtused, mis olid kõrgemad *CNN-Baseline* mudelitel.

Tabel 8. *CNN-Rectangular* mudelite koondtulemused.

Optimiseerija	SGD, $\Delta = 0.001$
Kannatlikkus	5
Epohhe	35 ± 5
Treenimise aeg	7 min 51 s ± 1 min 5 s

k-Täpsus	**k-Saagis**	**k-F_1**	**Täpsus**
$50.5 \pm 2.1\%$	$41.5 \pm 1.1\%$	$43.3 \pm 0.9\%$	$75.3 \pm 0.4\%$



Joonis 19. *CNN-Rectangular* mudeli kaofunktsiooni väärtuste graafik.

3.3.5 Paralleelarhitektuur

Erikujuliste filtrite kasutamine sama sisendi töötlemisel peaks andma paremaid tulemusi nii ajaliste kui ka sageduslike tunnuste eraldamisel mel-spektrogrammist. *CNN-ParallelRectangular* arhitektuur on ristkülikukujuliste filtrite kasutamise idee edasiarendus, mis töötleb spektrogramme paralleelselt. Üks osa arhitektuurist kasutab (3×4)-filtritega konvolutsioonilisi kihte ja teine osa kasutab (4×3)-filtritega konvolutsioonilisi kihte. Kahe osa ühendatud väljundid suunatakse täielikult ühendatud kihtidesse. Paralleelarhitektuuri kirjeldab tabel 9.

Kolme mudeli koondtulemused on märgitud tabelisse 10 ning joonisel 20 on kujutatud ühe mudeli kaofunktsiooni väärtuste graafikut. Mudelite statistiliste kriteeriumite väärtused on võrreldes eelnevate konvolutsiooniliste mudelitega selgesti paranenud. Võrreldes eelnevate mudelitega on k-täpsus oluliselt kõrgem.

Tabel 9. *CNN-ParallelRectangular* arhitektuur.

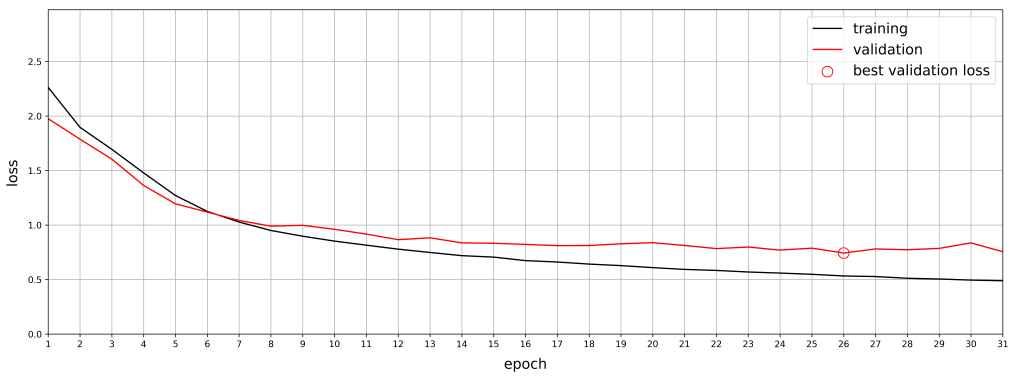
Sisend					S_{train}
Kiht	**Filter**	**Väljund**	**Kiht**	**Filter**	**Väljund**
conv1-1, relu	$(3,4) \times 128$	$(128, 62, 148)$	conv2-1, relu	$(4,3) \times 128$	$(128, 61, 149)$
max-pool	$(2,3)$	$(128, 31, 49)$	max-pool	$(2,3)$	$(128, 30, 49)$
conv1-2, relu	$(3,4) \times 128$	$(128, 29, 46)$	conv2-2, relu	$(4,3) \times 128$	$(128, 27, 47)$
max-pool	$(2,3)$	$(128, 14, 15)$	max-pool	$(2,3)$	$(128, 13, 15)$
conv1-3, relu	$(3,4) \times 256$	$(256, 12, 12)$	conv2-3, relu	$(4,3) \times 256$	$(256, 10, 13)$
max-pool	$(2,3)$	$(256, 6, 4)$	max-pool	$(2,3)$	$(256, 5, 4)$
conv1-4, relu	$(3,4) \times 512$	$(512, 4, 1)$	conv2-4, relu	$(4,3) \times 512$	$(512, 2, 2)$
max-pool	$(2,3)$	$(512, 2, 1)$	max-pool	$(2,3)$	$(512, 1, 1)$
flatten		(1024)	flatten		(512)

Kiht	**Väljund**
concat	(1536)
dense1, relu	(512)
dropout, $p = 0.25$	(512)
dense2	(C)
Parameetrite arv	5 133 343

Tabel 10. *CNN-ParallelRectangular* mudelite koondtulemused.

Optimiseerija	SGD, $\Delta = 0.001$
Kannatlikkus	5
Epohhe	27 ± 3
Treenimise aeg	23 min 18 s $\pm$ 2 min 24 s

k-Täpsus	**k-Saagis**	**k-F_1**	**Täpsus**
$53.4 \pm 1.5\%$	$44.4 \pm 1.6\%$	$45.6 \pm 0.9\%$	$75.7 \pm 0.6\%$



Joonis 20. *CNN-ParallelRectangular* mudeli kaofunktsiooni väärtuste graafik.

3.3.6 Lisatunnuste kasutamine

CNN-ExtraParallelRectangular arhitektuur on *CNN-ParallelRectangular* alternatiivne variant, mis töötleb sämplite ajalisi ja spektraalseid tunnuseid. Selle jaoks on lisatud paralleelselt paar täielikult ühendatud kihti. Arhitektuuri kirjeldus on toodud tabelis 11. Tabeli vasakpoolses osas on kujutatud *CNN-ParallelRectangular* arhitektuuri kuni väljajätukihini. Tabeli parempoolsesse ossa on märgitud kihid, mis töötlevad lisatunnuseid.

Tabel 11. *CNN-ExtraParallelRectangular* arhitektuur.

Sisend	S_{train}	Sisend	E_{train}
CNN-ParallelRectangular		**Kiht**	**Väljund**
...		flatten	(96)
...		dense2	(512)
dropout, $p = 0.25$	(512)	dense3	(48)
Kiht			**Väljund**
concat			(560)
dense4			(C)
Parameetrite arv			5 209 119

Kolme mudeli tulemused on toodud tabelis 12 ja joonisel 20 on kujutatud ühe mudeli kaofunktsiooni väärtuste graafikut. Lisatunnuste kasutamine sellisel moel ei parandanud klassifitseerimistulemusi. Probleemiks võib olla arhitektuuri kihtide ülesehitus. Samuti võis tulemusi mõjutada lisatunnuste normaliseerimata jätmine.

Mudeli vigadest arusaamiseks analüüsitakse segadusmaatriksit joonisel 21. Ühe näitena võib välja tuua, et mudel kaldub suure tõenäosusega klassifitseerima kategooriasse *kick* selliseid andmepunkte, mis tegelikult kuuluvad kategooriatesse *bongo conga*, *djembe*, *rhodes* ja *tom*. Mainitud kategooriatesse kuuluvate sämplite oluliseks tunnuseks on suhteliselt suure amplituudiga madalsageduslikud osahelid. See asjaolu kajastub ka madala spektraaltsentroidi väärtusena. Kuna treeningandmestikku kuuluvad valdavalt kategooriasse *kick* kuuluvad sämplid, siis arvatavasti seostab mudel sämplite spektraaltsentroidi madalaid väärtuseid kategooriaga *kick*.

Kuna sobivaid arhitektuurilisi lahendusi lisatunnuste andmestiku töötlemiseks ei leitud, siis lisatunnuseid arhitektuuri edasisel arendamisel ei kasutata.

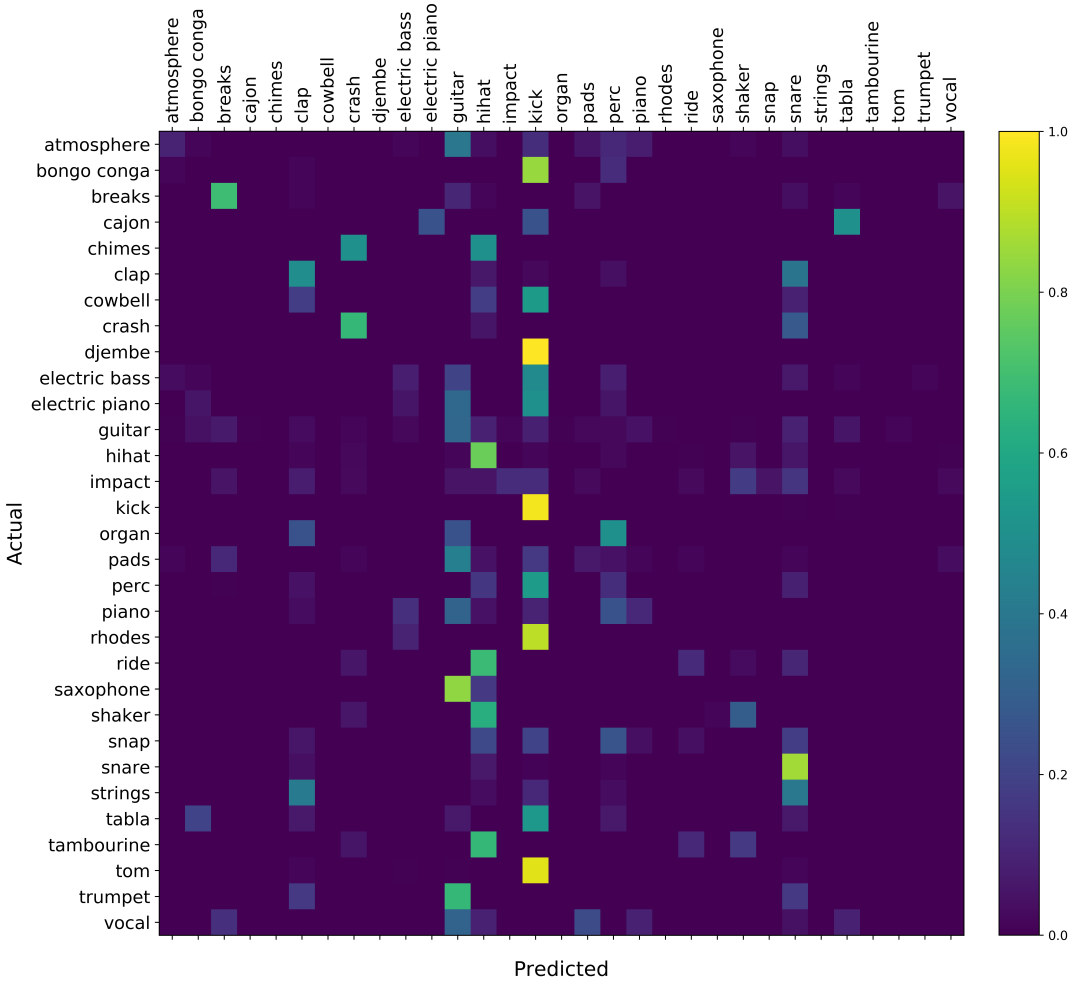
3.3.7 Rikastatud andmete kasutamine

Parima tulemuse on saavutanud konvolutsiooniline paralleelarhitektuur, mis töötleb ainult sämplite mel-spektrogramme. Edasi katsetatakse rikastatud sämplite kasuta-

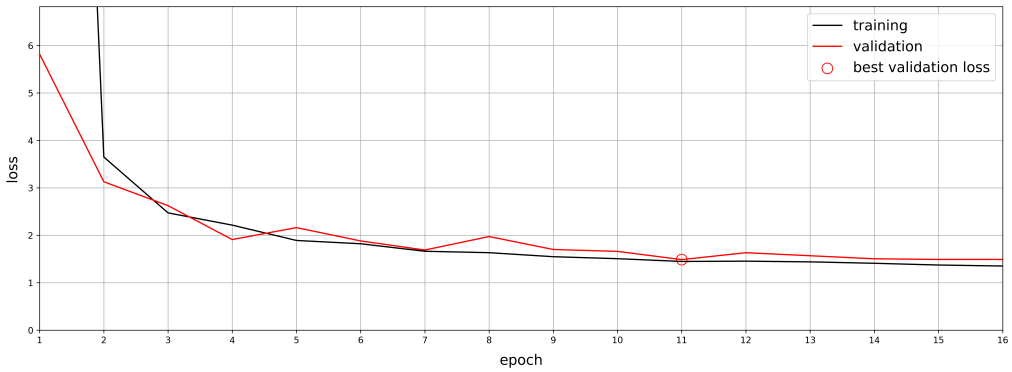
Tabel 12. *CNN-ExtraParallelRectangular* mudelite koondtulemused.

Optimiseerija	SGD, $\Delta = 0.0001$
Kannatlikkus	5
Epohhe	18 ± 11
Treenimise aeg	17 min 13 s $\pm$ 8 min 26 s

k-Täpsus	**k-Saagis**	**k-F_1**	**Täpsus**
$24.6 \pm 3.1\%$	$21.0 \pm 2.6\%$	$21.0 \pm 3.2\%$	$63.0 \pm 2.2\%$



Joonis 21. *CNN-ExtraParallelRectangular* mudeli segadusmaatriks.



Joonis 22. *CNN-ExtraParallelRectangular* mudeli kaofunktsiooni väärtuste graafik.

mist paralleelarhitektuuriga. Esimeses katses kasutatakse lisaks treeningandmestikule S_{train} rikastatud treeningandmestikku S_a ning teises katses kasutatakse rikastatud treeningandmestikku S_A.

Tabelites 13 ja 14 on toodud kolme mudeli koondtulemused kummagi katse jaoks. Rikastatud treeningandmestike kasutamine parandas klassifitseerimistulemusi. S_a kasutamine suurendas peamiselt k-saagist. S_A kasutamine suurendas kõiki statistilisi kriteeriume, taaskord on suurenenud k-saagise väärtus.

Tabel 13. *CNN-ParallelRectangular* mudelite koondtulemused andmestikuga S_a.

Optimiseerija			SGD, $\Delta = 0.001$
Kannatlikkus			5
Epohhe			19 ± 9
Treenimise aeg			26 min 24 s ± 9 min 24 s
k-Täpsus	**k-Saagis**	**k-F_1**	**Täpsus**
$54.2 \pm 4.9\%$	$53.2 \pm 4.5\%$	$50.7 \pm 3.9\%$	$75.2 \pm 1.5\%$

Tabel 14. *CNN-ParallelRectangular* mudelite koondtulemused andmestikuga S_A.

Optimiseerija			SGD, $\Delta = 0.002$
Kannatlikkus			5
Epohhe			5 ± 1
Treenimise aeg			1 h 1 min 20 s ± 3 min 16 s
k-Täpsus	**k-Saagis**	**k-F_1**	**Täpsus**
$54.4 \pm 3.4\%$	$55.0 \pm 2.2\%$	$52.3 \pm 2.1\%$	$76.0 \pm 0.9\%$

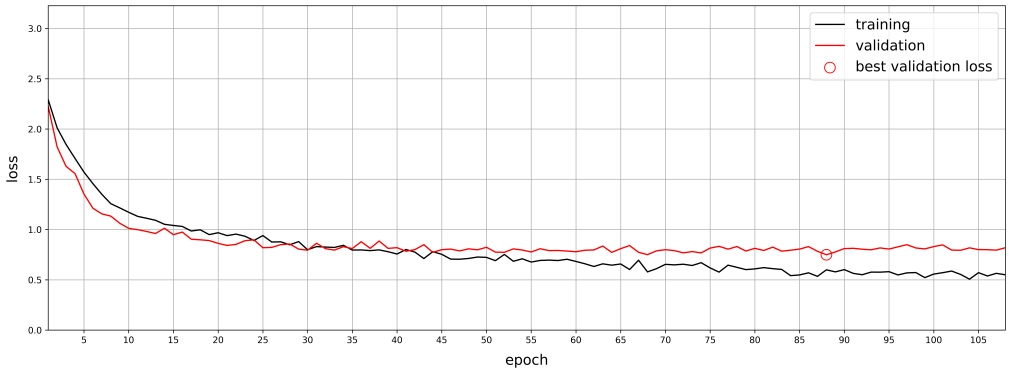
3.3.8 *Mix-up* meetodi kasutamine

CNN-MixupParallel on esialgse paralleelarhitektuuri edasiarendus, mille treenimisalgoritmis kasutatakse *mix-up* meetodit. *Mix-up* meetodit rakendati otse miniplokil. Selleks seati miniploki iga andmepunktiga vastavusse miniplokist juhuslikult valitud teine andmepunkt. Eksperimentide käigus selgus, et *mix-up* meetodil oli suur regulariseeriv mõju, mistõttu eemaldati täielikult ühendatud kihtide vahelt väljajätukiht. Kirjeldavat tabelit ei lisata, kuna arhitektuur on põhiosas sama oma vanemaga.

Tabelis 15 on toodud kolme mudeli tulemused ja joonisel 23 on kujutatud ühe mudeli kaofunktsiooni väärtuste graafikut. *Mix-up* meetodi kasutamine on andnud siiani parima täpsuse, kuid rikastatud andmete mittekasutamine vähendas k-saagist võrreldes eelmise katsega. Mudelite treenimisel jäi esimeste epohhide jooksul kaofunktsiooni väärtus treeningandmestikul kõrgemaks kaofunktsiooni väärtusest valideerimisandmestikul, mistõttu otsustati kannatlikkust suurendada.

Tabel 15. *CNN-MixupParallel* mudelite koondtulemused.

Optimiseerija			SGD, $\Delta = 0.001$
Kannatlikkus			20
Epohhe			66 ± 20
Treenimise aeg			1 h 2 min 30 s ± 13 min 25 s
k-Täpsus	**k-Saagis**	**k-F_1**	**Täpsus**
$55.8 \pm 2.2\%$	$48.6 \pm 1.7\%$	$50.2 \pm 1.8\%$	$78.1 \pm 0.5\%$



Joonis 23. *CNN-MixupParallel* mudeli kaofunktsiooni väärtuste graafik.

3.3.9 Rikastatud andmete ja *mix-up* meetodi üheaegne kasutamine

Edasi katsetatakse *CNN-MixupParallel* mudelite treenimist rikastatud spektrogrammide treeningandmestikuga S_A. Samuti kasutatakse kaofunktsiooni parema lokaalse miinimumi saavutamiseks valideerimisandmestikul õpisammu ajastamist. Õpi-

sammu ajastatakse kannatlikkusega 5 epohhi, mille järel vähendatakse õpisammu 5-kordselt. Tabelis 16 on toodud kolme mudeli koondtulemused. Mainitud meetodite üheaegne kasutamine parandas klassifitseerimistulemusi märgatavalt.

Tabel 16. *CNN-MixupParallel* mudelite koondtulemused andmestikuga S_A.

Optimiseerija			SGD, $\Delta = 0.001$
Kannatlikkus			20
Epohhe			23 ± 2
Treenimise aeg			3 h 21 min 29 s ± 21 min 15 s
k-Täpsus	**k-Saagis**	**k-F_1**	**Täpsus**
$57.2 \pm 0.1\%$	$57.6 \pm 0.3\%$	$55.8 \pm 0.3\%$	$78.7 \pm 0.5\%$

3.4 Pakutud arhitektuur

Pakutud sämplite klassifitseerimise probleemi lahendavaks konvolutsioonilise närvivõrgu arhitektuuriks on *CNN-MixupParallel*, mis kasutab õpisammu ajastamist, rakendab *mix-up* meetodit ja kasutab sisendina mel-spektrogramme, mida on rikastatud sarnaselt andmestikuga S_A.

Eksperimentide põhjal saavutati selle arhitektuuriga statistiliste kriteeriumide parimad väärtused. Tabelis 17 on näidatud tulemuste paranemisi eksperimentide kaupa. Tabelis 18 on näidatud pakutud arhitektuuri statistilise kriteeriumide suhtelisi paranemisi võrreldes baasarhitektuuridega.

Tabel 17. Konvolutsioonilise arhitektuuri paranemine eksperimentide kaupa.

Eksperiment	**Täpsus**	**Viga**	**k-F_1**
CNN-Baseline	73.9%	26.1%	43.6%
CNN-Rectangular	75.3%	24.7%	43.3%
CNN-ParallelRectangular	75.7%	24.3%	45.6%
+ rikastatud andmestik S_a	75.2%	24.8%	50.7%
+ rikastatud andmestik S_A	76.0%	24.0%	52.3%
CNN-MixupParallel	78.1%	21.9%	50.2%
+ rikastatud andmestik S_A	**78.7**%	**21.3**%	**55.8**%

Tabel 18. Pakutud arhitektuuri tulemuste paranemine võrreldes baasarhitektuuridega.

Baasarhitektuur	**Δ Täpsus**	**Δ Viga**	**Δ k-F_1**
DNN-Baseline	+5.2%	−15.5%	+25.0%
CNN-Baseline	+6.5%	−18.4%	+28.2%

4 Analüüs

Selles peatükis analüüsitakse pakutud lahendust ning tehakse võrdlusi olemasolevate arhitektuuride ja tarkvaralahendustega. Analüüsi põhjal osutatakse võimalustele töö edasiseks arendamiseks.

4.1 Vigade uurimine

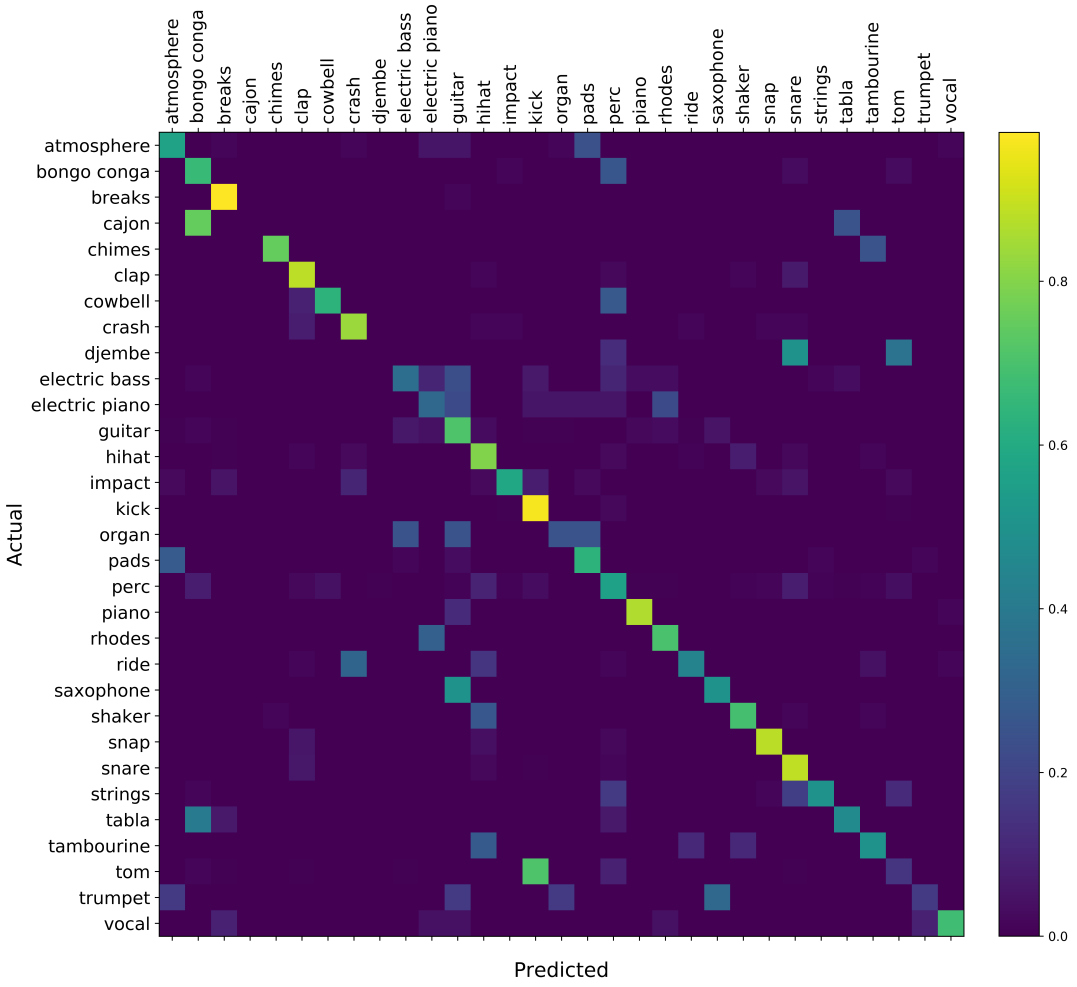
Mudeli vigade hindamiseks valitakse juhuslik hulk testandmestiku sämpleid, mis antakse sisendina pakutud arhitektuuri parimale mudelile. Mudeli hinnanguid võrreldakse subjektiivsete hinnangutega. Samuti kasutatakse mudeli segadusmaatriksit joonisel 24, mida uurides on võimalik paremini mõista mudeli poolt tehtavaid vigu.

Maatriksit uurides köidab esimesena tähelepanu kategooria *kick*, millesse kuuluvaid sämpleid ei klassifitseerita peaaegu kunagi valesti. Seda asjaolu võib peamiselt seostada *kick*-sämplite rohkusega treeningandmestikus. Samas on *kick*-sämplid oma tämbrilistelt omadustelt üsna ühetaolised.

Mudel teeb palju vigu *tom*-sämplite klassifitseerimisel, arvates need kategooriasse *kick*. Kummagi kategooria sämpleid iseloomustab kõrge amplituudiga ja madala sagedusega osahelide esinemine. Üldiselt ei ole selline viga soovitav, sest subjektiivsel hindamisel on kahe kategooria helid tihti eristatavad.

Trummihelisid kategooriatest *tabla* ja *cajon* klassifitseeritakse kui *bongo conga*. Subjektiivsel hindamisel leidub sämpleid, mille kategooriat on keeruline määrata, kuid valdavalt on erinevate kategooriate sämplid eristatavad. Algandmestikus on kategooriasse *bongo conga* kuuluvaid erinevaid sämpleid üle kolme korra rohkem kui *tabla*- ja *cajon* sämpleid kokku, mis võib olla mudeli vigade põhjuseks.

Sämpleid kategooriast *djembe* klassifitseeritakse peamiselt kategooriatesse *snare* ja *tom*. Subjektiivsel hindamisel on nendesse kategooriatesse kuuluvad sämplid üldjoontes eristatavad: vaid mõne üksiku sämpli korral oleks mudeli poolt *djembe*-sämplile määratud kategooria *tom* aktsepteeritav.



Joonis 24. Pakutud arhitektuuril treenitud mudeli segadusmaatriks.

Kategooriate *electric bass*, *electric piano* ja *guitar* puhul teeb mudel palju vigu. Subjektiivsel hindamisel on instrumentide sämplid eristatavad.

Mudel teeb ka aktsepteeritavaid vigu. Selliseks veaks on näiteks *atmosphere*-sämplite klassifitseemine *pad*-sämplitena ja vastupidi. Nendesse kategooriatesse kuuluvad sämplid jäävad subjektiivsel hindamisel tihti eristamatuteks. Andmestiku koostamisel lähtuti sämplikogumikes levinud kategoriseerimise skeemidest, kuid suures osas võib eelmainitud kategooriatesse kuuluvaid sämpleid kategoriseerida nii ühte kui ka teise või mõlemasse kategooriasse, sest nende funktsioon muusikas on sarnane. Sarnaseid vigu leidub ka näiteks kategooriate *hihat* ja *shaker* vahel ning kategooriate *electric piano* ja *rhodes* vahel.

Klassifitseerimistulemuste ülevaatamisel selgus ühe huvitava leiuna, et mudel leidis vea andmestikus endas. Üks sämpel oli andmestikus märgendatud kategooriasse *trumpet*, ent mudeli hinnang oli *saxophone*. Sämpli failinimes sisaldus märksõna, mis viitas *trumpet* kategooriale, kuid subjektiivsel taashindamisel leiti, et tegemist oli tõepoolest *saxophone*-sämpliga. See juhtum viitab algandmestiku märgendamise vigadele, mis võisid mingil määral mõjutada statistiliste kriteeriumide väärtuseid arhitektuuri arendamisel.

Mudeli klassifitseerimistulemustes leidub ka vähem olulisi vigu, mida käesolevas töös lähemalt ei käsitleta. Üldjoontes käitub mudel ootuspäraselt. Ühte osa vigadest aitaks parandada andmestiku täiendamine ja parandamine. Kindlasti vajab põhjalikku ülevaatust sämplite klassifitseerimiseks kasutatav kategooriate skeem.

4.2 Võrdlused

Eksperimentaalses osas saadud tulemusi võrreldakse kahe kirjanduses pakutud arhitektuuri tulemustega. Samuti tehakse võrdlusi arhitektuuri parima mudeli ja kahe helisämpleid kategoriseeriva tarkvararakenduse automaatsete klassifitseerimistulemuste vahel. Parima mudeli võrdlusesse võetakse ka kahe valdkonna eksperdi hinnangud. Arhitektuuride võrdlemisel lähtutakse statistilistest kriteeriumidest. Rakenduste ja ekspertide klassifitseerimistulemusi võrreldakse pakutud arhitektuuri parima mudeli hinnangutega kvalitatiivselt.

4.2.1 SampleCNN ja SwishNet

Olemasolevate arhitektuuride võrdlusesse on valitud kaks konvolutsioonilist arhitektuuri: SampleCNN [2.2, (a)] ja SwishNet [2.2, (b)]. Kuna mainitud arhitektuurid

olid oma valdkonna probleemi lahendamisel edukad, siis pakub huvi nende võimekus lahendada helisämplite klassifitseerimise probleem.

Võrdlus võib anda aimu ka erinevate sisendesituste kasutamise efektiivsusest sämplite klassifitseerimisel. SampleCNN kasutab sisendandmetena fikseeritud kestusega ajaesitusi ja SwishNet kasutab sisendandmetena mel-sageduse kepstri kordajaid. SampleCNN'i treenimiseks kasutatakse andmestikku D_{train}. Andmestiku D põhjal luuakse helisämplite mel-sageduse kepstri kordajate andmestik M andmepunkti kujuga $(20, 151)$: spektrogrammi arvutamise parameetrid on identsed eksperimentaalses osas kirjeldatule ja kordajate arvuks on valitud 20. SwishNet'i treenimiseks kasutatakse andmestikku M_{train}. Mudeleid treenitakse allikates kirjeldatu järgi.

Iga arhitektuuri jaoks treenitakse kolm mudelit. Tabelis 19 on toodud mudelite koondtulemuste võrdlus arhitektuuriti.

Tabel 19. Kolme arhitektuuri mudelite koondtulemuste võrdlus.

Arhitektuur	**k-Täpsus**	**k-Saagis**	**k-F_1**	**Täpsus**
Pakutud	$57.2 \pm 0.1\%$	$57.6 \pm 0.3\%$	$55.8 \pm 0.3\%$	$78.7 \pm 0.5\%$
SampleCNN	$39.8 \pm 3.3\%$	$32.6 \pm 1.8\%$	$33.6 \pm 2.0\%$	$73.7 \pm 1.9\%$
SwishNet	$33.5 \pm 1.0\%$	$31.7 \pm 0.5\%$	$32.2 \pm 0.7\%$	$68.3 \pm 1.2\%$

SampleCNN täpsus helisämplite klassifitseerimisel on pisut madalam võrreldes käesoleva töö baasarhitektuuridega. Ilmselt sobivad mel-spektrogrammid sämplite klassifitseerimiseks paremini, sest nende kasutamine tavalises, sügavas närvivõrgus andis paremaid tulemusi.

SwishNeti tulemused on võrreldes pakutud arhitektuuriga märgatavalt kehvemad. On tõenäoline, et vastavas allikas soovitatud kepstri kordajate arv jääb helisämplite klassifitseerimisel madalaks. Edaspidi tasuks lähemalt uurida mel-sageduse kepstri kordajate kasutamist helisämplite klassifitseerimisel.

4.2.2 Sononym ja Samplism

Olemasolevate rakenduste võrdlusesse on valitud kaks helisämplite haldamise tarkvaraprogrammi – Sononym [2.3, (a)] ja Samplism [2.3, (b)] –, mis suudavad helisämpleid automaatselt kategoriseerida.

Vaatluse põhjal lahendab Sononym sarnast probleemi, kategoriseerides sämpleid failinimest või faili paiknemisest sõltumata. Samplismi poolt pakutav automaatse kategoriseerimise funktsionaalsus sõltub täielikult helifailide nimedest. Samplism ei

lahenda Sononymi ega käesolevas töös pakutud arhitektuuriga sarnast ülesannet, kuid sellele vaatamata võib olla huvitav näha ka selle rakenduse tulemusi.

Tabelis 20 on toodud klassifitseerimistulemused pakutud arhitektuuril treenitud parima mudeli ja kummagi rakenduse jaoks. Võrdlusesse püüti leida võimalikult erinevate klassifitseerimistulemustega sämpleid. Rakendustes esitatud kategooriate nimetusi on lühendatud. Mõttekriips tähistab kategooria puudumist.

Tabel 20. Mudeli ja võrreldavate rakenduste klassifitseerimistulemused.

i	**Failinimi**	**Algandmestik**	**Mudel**	**Sononym**	**Samplism**
1	`weow.wav`	*hihat*	*clap*	*OneShot*	—
2	`SSH_Closed_Hat_73.wav`	*hihat*	*clap*	*Hats & Shakers*	*Drum: Closed Hihat*
3	`VI2PIZZ E5.wav`	*strings*	*perc*	*Snares*	*Techniques: Pizzicato*
4	`TablaHit-0020.wav`	*tabla*	*bongo conga*	*Wood Hits*	*Percussion: Tabla*
5	`PB Electric Slide E 06.wav`	*guitar*	*electric bass*	*Loop*	*Genre: Electric*
6	`SF Acoustic Strum 31.wav`	*guitar*	*guitar*	*Loop*	*Genre: Acoustic*
7	`SR16 Kik_Dry12.wav`	*kick*	*kick*	*Kicks*	*Effects: Dry*
8	`SR16 Kik_Hall 4.wav`	*kick*	*kick*	*Kicks*	—

Sämpli 1 failinimi iseloomustab töös käsitletavat probleemi ilmekalt: failinimes puudub informatsioon helilise sisu kohta. Kuna Samplism kasutab kategoriseerimisel failinimes sisalduvaid märksõnu ja kuna ilmselt märksõna `weow` rakenduse andmebaasis ei leidu, siis ei suuda rakendus sämplit klassifitseerida. Sononym suutis sämpli kategoriseerida ainult üldisesse kategooriasse *OneShot*.

Sämplit 2 ei suutnud mudel õigesti kategoriseerida. Sononym määrab sämplile laiema, ent aktsepteeritava, kategooria. Samplism suudab failinimest tuletada õige ja täpse kategooria: lisaks taldriku tüübile on ära märgitud ka mängutehnika (*closed*).

Sämpli 3 failinimes sisaldub viide *pizzicato* mängutehnikale, mille Samplism suudab õigesti tuvastada. Küll aga ei ole Samplism tuletanud, et tegu on keelpilliheliga, kuigi *pizzicato* tehnikat kasutatakse ainult keelpillidega. Mudel ega Sononym sämpli kategooriat õigesti ei määranud.

Sämpli 4 klassifitseeris õigesti vaid Samplism. Sononymi poolt määratud kategooria ei ole seotud tabla-trummidega.

Sämplid 5 ja 6 näitlikustavad pakutud arhitektuuril treenitud mudeli paremaid tulemusi *guitar*-sämplite klassifitseerimisel. Sononym ei lisa *Loop*-tüüpi sämplitele lisanduvaid kategooriaid. Samplism tuletab failinimest "žanri", mis on põhimõtteliselt ebakorrektne, sest nimetused *Electric* ja *Acoustic* viitavad instrumendi tüübile, mitte muusikalisele žanrile.

Sämplid 7 ja 8 kategoriseeriti mudeli ja Sononymi poolt õigesti. Samplism ei arvesta instrumentide nimede variatsioonidega: `kik` on sarnane sõnele `kick`.

Mudeli ja rakenduste tulemuste võrdluse põhjal võib oletada, et parimaid klassifitseerimistulemusi annab sämpli helilise sisu ja failinime üheaegne kasutamine. Sononym määrab sämplitele tihti üldiseid kategooriaid nagu *OneShot* või *Loop*. Samplism suudab failinimest tuletada sämpli instrumendi mõnedel juhtudel, mil mudel ega Sononym õiget kategooriat ei määranud.

4.2.3 Valdkonna eksperdid

Sämpleid klassifitseeriva mudeli tulemusi on huvitav võrrelda valdkonna ekspertide tulemustega. Selleks viiakse läbi katse kahe subjektiga. Eksperimentaalses osas kirjeldatud testandmestikust eraldatakse 100 juhuslikult valitud sämplit, mis moodustavad katseandmestiku. Katseandmestikku kuuluvaid sämpleid esitatakse mudelile ja kahele eksperdile, kes määravad kuuldud sämpli kategooria vastavalt algandmestikus kasutatud kategoriseerimise skeemile. Seejärel võrreldakse katses osalenute hinnanguid parima mudeli tulemustega kvalitatiivselt. Katse aitab mõista mudeli tulemuslikkust võrreldes valdkonna asjatundjatega.

Tabelis 21 on toodud klassifitseerimistulemused pakutud arhitektuuril treenitud parima mudeli ja kummagi eksperdi jaoks. Katses osalenud eksperdid ei olnud teadlikud failinimedest: need on tabelisse lisatud illustratiivsel eesmärgil. Võrdlusesse püüti valida võimalikult erinevaid hinnanguid saanud katseandmestiku sämplid.

Tabel 21. Mudeli ja ekspertide klassifitseerimistulemused.

i	**Failinimi**	**Algandmestik**	**Mudel**	**Ekspert A**	**Ekspert B**
1	`shaker.wav`	*shaker*	*hihat*	*hihat*	*shaker*
2	`SR16 Hat_Hard.wav`	*hihat*	*perc*	*perc*	*perc*
3	`Lev_Djembe_Low_09.wav`	*djembe*	*tom*	*djembe*	*djembe*
4	`Your Pad.wav`	*pads*	*pads*	*atmosphere*	*pads*
5	`SPATS - Pad 5.wav`	*pads*	*atmosphere*	*atmosphere*	*atmosphere*
6	`trumpet step.wav`	*trumpet*	*saxophone*	*saxophone*	*saxophone*
7	`HM_124_GF_EPiano20.wav`	*electric piano*	*guitar*	*rhodes*	*rhodes*
8	`PB Electric Slide E 06.wav`	*guitar*	*electric bass*	*guitar*	*guitar*
9	`VMH2 Toms & Deep Tones 109.wav`	*tom*	*kick*	*kick*	*kick*
Täpsus katseandmestikul			75%	73%	78%

Sämpli 1 hindamisel tegi vea nii mudel kui ka üks ekspertidest. Kategooriatesse *hihat* ja *shaker* kuuluvate sämplite tämber ja funktsioon muusikas on sarnane, mis võib olla vigade põhjuseks.

Sämpel 2 kuulub kategooriasse *hihat*, kuid see klassifitseeriti nii mudeli kui ka ekspertide poolt kategooriasse *perc*. Sämpel 2 ei ole sarnane tüüpilistele *hihat*-sämplitele.

Sämpli 3 tulemused ilmestavad mudeli viga võrreldes ekspertidega, kes suutsid tuvastada *djembe*-sämpli.

Sämplid 4 ja 5 näitlikustavad kategooriatesse *atmosphere* ja *pads* kuuluvate sämplite sarnasusi, sest võrreldes algandmestikuga teevad vigu nii mudel kui ka mõlemad katses osalenud eksperdid.

Sämpel 6 on saksofoniheli, mis on algandmestikus vigaselt kategoriseeritud kui *trumpet*. Nii mudel kui ka eksperdid kategoriseerisid sämpli kui *saxophone*, mis oleks pidanud olema selle sämpli tegelik kategooria.

Sämpel 7 on ekspertide poolt klassifitseeritud kui *rhodes*, mis on ühte tüüpi elektriklaver ning on seetõttu seotud kategooriaga *electric piano*. Mudel klassifitseerib selle sämpli vigaselt kategooriasse *guitar*.

Sämpli 8 puhul on näha, et mudel leidis õigele kategooriale *guitar* lähedase kategooria *electric bass*.

Sämpli 9 tulemused ilmestavad helide kategoriseerimise mõnetist subjektiivsust. Failinimi ja selle heliline sisu on algselt seotud kategooriaga *tom*, kuid mudel ja eksperdid klassifitseerisid sämpli üheselt kategooriasse *kick*.

Mudel lahendab püstitatud probleemi valdkonna ekspertidele sarnaselt. Subjektid mainisid algandmestikus kasutatud kategoriseerimise skeemi piiratust: mõni sämpel võis nende hinnangul kuuluda mitmesse kategooriasse.

4.3 Edasiarendused

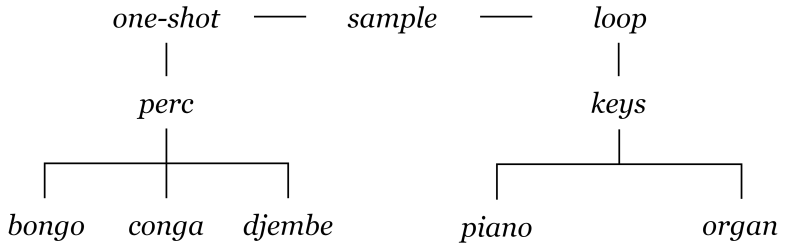
Töös pakutud lahendust saaks edasi arendada mitmel moel. Esiteks oleks oluline luua *hierarhiline* kategooriate skeem ja märgendada andmestikku kuuluvaid sämpleid selle skeemi järgi. Teiseks, kuna eksperimentide käigus selgus, et andmete rikastamine parandas mõnevõrra klassifitseerimistulemusi, siis oleks kasu treeningandmestiku suurendamisest ja erinevate sämplite rikastamise meetoditega eksperimenteerimisest. Kolmandaks tasuks edasi arendada arhitektuuri, muutes kihtide ülesehitust ja hüperparameetreid, ning täiendada treeningalgoritmi. Heade tulemustega klassifitseerivat mudelit on võimalik kasutada helisämplite haldamise rakendustes.

4.3.1 Kategooriate hierarhia

Sämpleid saab kategoriseerida erinevalt. Käesolevas töös kasutatud kategooriate liigitus on välistav, st ühte kategooriasse kuulumine välistab kuulumise kõikidesse

teistesse kategooriatesse. Käesolevas töös kasutatud liigitust saab laiendada, luues kategooriate hierarhia. Sel juhul määratakse igale sämplile nii üldisemad kui ka täpsemad kategooriad, mis üksteist ei välista.

Kategooriate hierarhiat on kujutatud joonisel 25. Ühe näitena võib välja tuua käesolevas töös kasutatud kategooriad *bongo conga* ja *djembe*, mis võiksid kuuluda ülemkategooriasse *perc*, mis võiks omakorda kuuluda ülemkategooriasse *one-shot*.



Joonis 25. Kategooriate hierarhia näide.

4.3.2 Andmestiku laiendamine

Andmestiku koostamisel tasuks lähtuda loodavast kategooriate hierarhiast. Kuna eksperimentaalses osas selgus, et andmete rikastamine parandas klassifitseerimistulemusi, siis oleks oluline ka koguda uusi sämpleid selliste kategooriate jaoks, millesse kuuluvaid olemasolevaid sämpleid on vähem.

Sämplite eeltöötlemisel säilitati helide subjektiivne eristusvõime, kuid vähendatud diskreetimissagedus ja mitme kanali üheks summeerimine võis siiski mõjutada konvolutsiooniliste mudelite klassifitseerimistulemusi. Seetõttu tasuks proovida kõrgemat diskreetimissagedust ja jätta kanalid monosse summeerimata.

Mel-spektrogrammide arvutamisel valitud STFT ja mel-filtripanga parameetrid valiti helisämplite eripärasid meeles pidades, kuid täpsemaid eksperimente eelnevalt teostamata. Kuna valdav osa sämpleid on lühikesed, siis tasuks konvolutsioonilise arhitektuuri sisendina kasutatavate spektrogrammide arvutamisel eksperimenteerida väiksema akna suuruse ja hüppe pikkusega. Samuti tasuks katsetada mel-filtripanga filtrite arvu suurendamist. Kirjanduse põhjal võib häid tulemusi anda ka varieeruva resolutsiooniga mel-spektrogrammide kasutamine.

4.3.3 Arhitektuuri parendamine

Pakutud arhitektuur ei pruugi olla optimaalne. Konvolutsioonilise arhitektuuri mõistmiseks tasuks lähemalt uurida treenitud mudelite konvolutsiooniliste kihtide filtrite kaalusid, mis võib anda informatsiooni kanalite arvu ja filtrite kuju optimaalsemaks

sättimiseks. Kirjanduse põhjal võib klassifitseerimisülesande lahendamiseks häid tulemusi anda ka ajaesituse ja spektrogrammide paralleelne kasutamine.

Töö edasisel arendamisel võiks luua hierarhilise klassifitseerija, mis ühendab mitu alamklassifitseerijat. Kategooriate hierarhias võiks olla üks klassifitseerija iga hargnemise kohta. Nii liigitatakse sämplid esmalt üldisematesse kategooriatesse ning seejärel täpsematesse kategooriatesse. Hüpotees on, et alamandmestikel treenitud mudelid võivad töötada täpsemate kategooriate määramisel paremini kui üks monoliitne mudel.

Hiljutise kirjanduse põhjal võib häid tulemusi anda pool-juhendatud õppe rakendamine *MixMatch* meetodil, mis võimaldab treenimise ajal kasutada samaaegselt nii märgendatud kui ka märgendamata andmeid[5].

4.3.4 Rakendus

Edasiarendatud helisämplite klassifitseerijat saaks kasutada ühe osana tarkvaralahendusest, mille abil oleks võimalik sämpleid korrapärastada, analüüsida ning automaatselt kategoriseerida. Kasutajale võiks kuvada rakendusse laetud sämplite helianalüüsi informatsiooni ning klassifitseerija poolt määratud kategooriaid. Kasutajal peaks olema võimalik otsida sämpleid sarnasuse järgi: valides ühe sämpli, kuvaks programm kasutaja poolt valitud otsingukriteeriumide järgi teisi helisämpleid. Rakendus võiks siduda heli- ja tekstianalüüsi automaatsete klassifitseerimistulemuste parendamiseks. Samuti oleks kasutajal võimalik muuta helifaili metaandmeid, sealhulgas sämplile automaatselt määratud kategooriat, mis võimaldaks klassifitseerijal kasutaja isiklike eelistustega kohaneda.

Kokkuvõte

Helisämplite automaatne kategoriseerimine on vähe uuritud valdkond. Senini ei ole leidunud töid, mis oleksid püüdnud arendada sämplite klassifitseerijat suure varieeruvuse ja kategooriate arvuga andmestikul. Käesoleva töö põhieesmärgiks oli luua konvolutsioonilise närvivõrgu arhitektuuril põhinev helisämplite klassifitseerija, mis suudaks kategoriseerida sämpleid sõltumata failinimest või paiknemisest failisüsteemis. Eesmärgi saavutamiseks loodi uus helisämplite andmestik ning arendati eksperimentaalselt välja klassifitseeriv mudel, mida võrreldi sarnaste lahendustega.

Konvolutsioonilised närvivõrgud sobivad sämplite klassifitseerimise probleemi lahendamiseks. Pakutud arhitektuuril põhinevad mudelid saavutasid testandmestikul keskmise täpsuse 78.7%. Eksperimentide käigus õnnestus saavutada konvolutsioonniliste mudelite 15.5-protsendiline suhteline vigade vähenemine võrreldes sügaval närvivõrgul põhinevate baasmudelitega.

Sarnaste konvolutsiooniliste arhitektuuride kvantitatiivses võrdluses saavutasid pakutud arhitektuuri mudelid paremaid tulemusi. Helifailide automaatse kategoriseerimise funktsionaalsust pakkuva rakenduse kvalitatiivses võrdluses saavutas töö parim mudel rakendusega sarnaseid tulemusi. Valdkonna ekspertidega korraldatud katses saavutas mudel ekspertide hinnangutega samuti sarnaseid tulemusi.

Töö tulemust saab edasi arendada mitmel moel. Esmalt oleks vaja üksikasjalikumalt uurida helide kategoriseerimise võimalusi ja skeeme, mis ei oleks piiratud ning mida oleks võimalik vajadusel laiendada. Samuti on oluline luua põhjalikult märgendatud ja vabalt kättesaadav helisämplite andmestik. Seejärel on võimalik töös pakutud arhitektuuri täiendada selliselt, et mudelite tulemused paraneksid. Üheks tulemusi parandavaks meetodiks võib olla heli- ja tekstianalüüsi sidumine. Töökindlaid klassifitseerijaid saab kasutada sämplite haldamise rakendustes, mis sobiksid heli- ja muusikavaldkonna spetsialistidele.

Kasutatud allikad

[1] Abdoli, S., Cardinal, P., Koerich, A. L. End-to-End Environmental Sound Classification using a 1D Convolutional Neural Network. 2019. [*Online*] arXiv: 1904.08990.

[2] Audacity Development Team. Digital Audio Fundamentals. https://alphamanual.audacityteam.org/m/index.php?title=Digital_Audio&oldid=63050 (25.04.2019).

[3] Audio Helper Project. Samplism. https://audiohelperproject.com/products/samplism/ (15.04.2019).

[4] Baumann, T., Köhn, A., Hennig, F. The Spoken Wikipedia Corpus collection: Harvesting, alignment and an application to hyperlistening. *Language Resources and Evaluation*. 2018. [DOI] 10.1007/s10579-017-9410-y.

[5] Berthelot, D. *et al.* MixMatch: A Holistic Approach to Semi-Supervised Learning. 2019. [*Online*] arXiv: 1905.02249.

[6] Bertin-Mahieux, T. *et al.* The Million Song Dataset. *Proceedings of the 12th International Conference on Music Information Retrieval*. 2011. https://www.ee.columbia.edu/~dpwe/pubs/BertEWL11-msd.pdf (29.04.2019).

[7] Brownlee, J. A Gentle Introduction to Mini-Batch Gradient Descent and How to Configure Batch Size. https://machinelearningmastery.com/gentle-introduction-mini-batch-gradient-descent-configure-batch-size/ (11.05.2019).

[8] Brownlee, J. Dropout Regularization For Neural Networks. https://machinelearningmastery.com/dropout-regularization-deep-learning-models-keras/ (11.05.2019).

[9] Bushaev, V. How do we 'train' neural networks? https://towardsdatascience.com/how-do-we-train-neural-networks-edd985562b73 (11.05.2019).

[10] Computer Science Wiki. Max-pooling / Pooling. https://computerscience wiki.org/index.php?title=Max-pooling_/_Pooling&oldid=7839 (11.05.2019).

[11] Correya, A. A., Hennequin, R., Arcos, M. Large-Scale Cover Song Detection in Digital Music Libraries Using Metadata, Lyrics and Audio Features. 2018. [*Online*] arXiv: 1808.10351.

[12] Defferrard, M. *et al.* FMA: A Dataset For Music Analysis. 2016. [*Online*] arXiv: 1612.01840.

[13] Delbouys, R. *et al.* Music Mood Detection Based On Audio And Lyrics With Deep Neural Net. 2018. [*Online*] arXiv: 1809.07276.

[14] Deolekar, S., Abraham, S. Tree based classification of tabla strokes. 2018. [*Online*] arXiv: 1801.01712.

[15] Dong, M. Convolutional Neural Network Achieves Human-level Accuracy in Music Genre Classification. 2018. [*Online*] arXiv: 1802.09697.

[16] Encyclopædia Britannica. Sound. https://www.britannica.com/science/sound-physics (27.04.2019).

[17] Engel, J. *et al.* Neural Audio Synthesis of Musical Notes with WaveNet Autoencoders. 2017. [*Online*] arXiv: 1704.01279.

[18] Erzetich. Time and Frequency Domain. https://www.erzetich-audio.com/knowledgebase-05-time-vs-frequency.html (06.05.2019).

[19] Fonseca, E. *et al.* Freesound Datasets: a platform for the creation of open audio datasets. *Proceedings of the 18th International Society for Music Information Retrieval Conference*. Suzhou, China, 2017, 486–493. https://repositori.upf.edu/bitstream/handle/10230/33299/fonseca_ismir17_freesound.pdf (29.04.2019).

[20] Gemmeke, J. F. *et al.* Audio Set: An ontology and human-labeled dataset for audio events. *Proceedings of IEEE ICASSP*. New Orleans, LA, 2017. https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/45857.pdf (29.04.2019).

[21] Haytham Fayek. Speech Processing for Machine Learning: Filter banks, Mel-Frequency Cepstral Coefficients (MFCCs) and What's In-Between. https://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html (06.05.2019).

[22] Heittola, T. Datasets: Environmental Sounds. http://www.cs.tut.fi/~heittolt/datasets (15.04.2019).

[23] Herrera, P., Yeterian, A., Gouyon, F. Automatic Classification of Drum Sounds: A Comparison of Feature Selection Methods and Classification Techniques. *Music and Artificial Intelligence*. Springer: Berlin, Heidelberg, 2002, 69–80. [DOI] 10.1007/3-540-45722-4_8.

[24] Huang, J. J., Leanos, J. J. A. AclNet: efficient end-to-end audio classification CNN. 2018. [*Online*] arXiv: 1811.06669.

[25] Hung, Y.-N., Yang, Y.-H. Frame-level Instrument Recognition by Timbre and Pitch. 2018. [*Online*] arXiv: 1806.09587.

[26] Hussain, M. S., Haque, M. A. SwishNet: A Fast Convolutional Neural Network for Speech, Music and Noise Classification and Segmentation. 2018. [*Online*] arXiv: 1812.00149.

[27] ID3 Contributors. What is ID3v2? http://id3.org/ID3v2Easy (06.05.2019).

[28] Jupyter Team. The Jupyter Notebook. https://jupyter-notebook.readthedocs.io/en/stable/notebook.html (11.04.2019).

[29] Karagioza. Digital representation of the sound. https://karagioza.com/?p=58 (06.05.2019).

[30] Korzeniowski, F., Widmer, G. Genre-Agnostic Key Classification With Convolutional Neural Networks. 2018. [*Online*] arXiv: 1808.05340.

[31] Kozakowski, P., Michalak, B. Music Genre Recognition. http://deepsound.io/music_genre_recognition.html (06.05.2019).

[32] Lederle, M., Wilhelm, B. Combining High-Level Features of Raw Audio Waves and Mel-Spectrograms for Audio Tagging. 2018. [*Online*] arXiv: 1811.10708.

[33] Lee, J. *et al.* Raw Waveform-based Audio Classification Using Sample-level CNN Architectures. 2017. [*Online*] arXiv: 1712.00866.

[34] Lee, J. *et al.* Sample-level Deep Convolutional Neural Networks for Music Auto-tagging Using Raw Waveforms. 2017. [*Online*] arXiv: 1703.01789.

[35] Librosa Development Team. LibROSA — librosa 0.6.3 documentation. https://librosa.github.io/librosa/index.html (28.04.2019).

[36] Librosa Development Team. librosa.feature.spectral_flatness — librosa 0.6.3 documentation. https://librosa.github.io/librosa/generated/librosa.feature.spectral_flatness.html (28.04.2019).

[37] Librosa Development Team. librosa.feature.spectral_rolloff — librosa 0.6.3 documentation. https://librosa.github.io/librosa/generated/librosa.feature.spectral_rolloff.html (28.04.2019).

[38] Liu, C. *et al.* Bottom-up Broadcast Neural Network For Music Genre Classification. 2019. [*Online*] arXiv: 1901.08928.

[39] Machine Learning Cheatsheet. Loss Functions. https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html#cross-entropy (11.05.2019).

[40] Mozilla. Common Voice Datasets. https://voice.mozilla.org/en/datasets (15.04.2019).

[41] Music Technology Group of Universitat Pompeu Fabra. Algorithm reference: RollOff — Essentia 2.1-beta5-dev documentation. https://essentia.upf.edu/documentation/reference/streaming_RollOff.html (28.04.2019).

[42] Nagrani, A., Chung, J. S., Zisserman, A. VoxCeleb: a large-scale speaker identification dataset. 2017. [*Online*] arXiv: 1706.08612.

[43] Nielsen, M. A. Neural networks and deep learning — Deep learning. http://neuralnetworksanddeeplearning.com/chap6.html (29.04.2019).

[44] Nielsen, M. A. Neural networks and deep learning — Using neural nets to recognize handwritten digits. http://neuralnetworksanddeeplearning.com/chap1.html (29.04.2019).

[45] NVIDIA Developer. CUDA Zone. https://developer.nvidia.com/cuda-zone (28.04.2019).

[46] Piel, L. K. Speech-Based Identification of Children's Gender and Age with Neural Networks. Master's Thesis. Tallinn: Tallinn University of Technology, 2018. [*Online*] TTÜR Digikogu: 10757.

[47] Python Software Foundation. The Python Tutorial — Python 3.7.3 Documentation. https://docs.python.org/3.7/tutorial/index.html (11.04.2019).

[48] PyTorch. Autograd: Automatic Differentiation. https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html (12.04.2019).

[49] Scavone, G. P. Signal Spectra. https://www.music.mcgill.ca/~gary/307/week1/spectra.html (27.04.2019).

[50] Schindler, A., Lidy, T., Rauber, A. Multi-Temporal Resolution Convolutional Neural Networks for Acoustic Scene Classification. 2018. [*Online*] arXiv: 1811.04419.

[51] Seipel, F. Music Instrument Identification using Convolutional Neural Networks. Master's Thesis. Berlin: Technical University of Berlin, 2018. https://www2.ak.tu-berlin.de/~akgroup/ak_pub/abschlussarbeiten/2018/Seipel_MasA.pdf (29.04.2019).

[52] Shafkat, I. Intuitively Understanding Convolutions for Deep Learning. https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1 (11.05.2019).

[53] Shier, J., McNally, K., Tzanetakis, G. Sieve: A Plugin for the Automatic Classification and Intelligent Browsing of Kick and Snare Samples. *Proceedings of the 3rd Workshop on Intelligent Music Production*. Salford, UK, 2017. http://www.semanticaudio.co.uk/wp-content/uploads/2017/09/WIMP2017_ShierEtAl.pdf (29.04.2019).

[54] Sononym. About Sononym. https://www.sononym.net/about/sononym/ (15.04.2019).

[55] Stack Overflow. Way to embed fixed length spectograms to tensor with CNN. https://stackoverflow.com/questions/43914136/way-to-embed-fixed-length-spectograms-to-tensor-with-cnn-perhaps (06.05.2019).

[56] Stanford CS231n Course Material Contributors. Convolutional Neural Networks for Visual Recognition. http://cs231n.github.io/convolutional-networks (29.04.2019).

[57] Zhang, H. *et al.* mixup: Beyond Empirical Risk Minimization. 2017. [*Online*] arXiv: 1710.09412.

[58] Zhang, Z. *et al.* Deep Convolutional Neural Network with Mixup for Environmental Sound Classification. 2018. [*Online*] arXiv: 1808.08405.

[59] Talimets, M. End-to-end speech recognition for Estonian. Master's Thesis. Tallinn: Tallinn University of Technology, 2018. [*Online*] TTÜR Digikogu: 10607.

[60] Tensorflow. Explore overfitting and underfitting. https://www.tensorflow.org/tutorials/keras/overfit_and_underfit (11.05.2019).

[61] Vikipeedia. Heli. https://et.wikipedia.org/w/index.php?title=Heli&oldid=5294773 (25.04.2019).

[62] Vikipeedia. Kvantimine. https://et.wikipedia.org/w/index.php?title=Kvantimine&oldid=4900334 (27.04.2019).

[63] Vikipeedia. Tämber. https://et.wikipedia.org/w/index.php?title=T%C3%A4mber&oldid=5191839 (26.04.2019).

[64] Vikipeedia. Täpsus ja saagis. https://et.wikipedia.org/w/index.php?title=T%C3%A4psus_ja_saagis&oldid=5292700 (08.05.2019).

[65] Wikipedia. Absolute threshold of hearing. https://en.wikipedia.org/w/index.php?title=Absolute_threshold_of_hearing&oldid=893814294 (27.04.2019).

[66] Wikipedia. Artificial neural network. https://en.wikipedia.org/w/index.php?title=Artificial_neural_network&oldid=893208691 (29.04.2019).

[67] Wikipedia. Audio Interchange File Format. https://en.wikipedia.org/w/index.php?title=Audio_Interchange_File_Format&oldid=891174059#Metadata (07.05.2019).

[68] Wikipedia. Audio signal. https://en.wikipedia.org/w/index.php?title=Audio_signal&oldid=891891985 (26.04.2019).

[69] Wikipedia. Convolutional neural network. https://en.wikipedia.org/w/index.php?title=Convolutional_neural_network&oldid=893419085 (29.04.2019).

[70] Wikipedia. Digital audio. https://en.wikipedia.org/w/index.php?title=Digital_audio&oldid=884238956 (27.04.2019).

[71] Wikipedia. Extensible Metadata Platform. https://en.wikipedia.org/w/index.php?title=Extensible_Metadata_Platform&oldid=894016437 (07.05.2019).

[72] Wikipedia. Fourier analysis. https://en.wikipedia.org/w/index.php?title=Fourier_analysis&oldid=887020977 (27.04.2019).

[73] Wikipedia. Mel scale. https://en.wikipedia.org/w/index.php?title=Mel_scale&oldid=889156041 (27.04.2019).

[74] Wikipedia. Metadata. https://en.wikipedia.org/w/index.php?title=Metadata&oldid=895258566 (06.05.2019).

[75] Wikipedia. Nyquist–Shannon sampling theorem. https://en.wikipedia.org/w/index.php?title=Nyquist%E2%80%93Shannon_sampling_theorem&oldid=890776007 (27.04.2019).

[76] Wikipedia. Precision and recall. https://en.wikipedia.org/w/index.php?title=Precision_and_recall&oldid=893227571#Definition_(classification_context) (08.05.2019).

[77] Wikipedia. PyTorch. https://en.wikipedia.org/w/index.php?title=PyTorch&oldid=890207605 (12.04.2019).

[78] Wikipedia. Sound. https://en.wikipedia.org/w/index.php?title=Sound&oldid=891990566 (27.04.2019).

[79] Wikipedia. Sound pressure. https://en.wikipedia.org/w/index.php?title=Sound_pressure&oldid=894066216 (27.04.2019).

[80] Wikipedia. Spectral centroid. https://en.wikipedia.org/w/index.php?title=Spectral_centroid&oldid=843132624 (28.04.2019).

[81] Wikipedia. Spectral density. https://en.wikipedia.org/w/index.php?title=Spectral_density&oldid=892709202 (27.04.2019).

[82] Wikipedia. Spectrogram. https://en.wikipedia.org/w/index.php?title=Spectrogram&oldid=893824963 (27.04.2019).

[83] Wikipedia. Statistical classification. https://en.wikipedia.org/w/index.php?title=Statistical_classification&oldid=889812528 (08.05.2019).

[84] Wikipedia. Zero-crossing rate. https://en.wikipedia.org/w/index.php?title=Zero-crossing_rate&oldid=879099113 (28.04.2019).

[85] Wikipedia. Threshold of pain. https://en.wikipedia.org/w/index.php?title=Threshold_of_pain&oldid=848115906 (27.04.2019).

[86] Wikipedia. Timbre. https://en.wikipedia.org/w/index.php?title=Timbre&oldid=894045877 (27.04.2019).

[87] Wikipedia. WAV. https://en.wikipedia.org/w/index.php?title=WAV&oldid=895345023#Metadata (07.05.2019).

[88] Wikipedia. Waveform. https://en.wikipedia.org/w/index.php?title=Waveform&oldid=892395776 (26.04.2019).

[89] Xu, K. *et al.* General audio tagging with ensembling convolutional neural network and statistical features. 2018. [*Online*] arXiv: 1810.12832.