

RĪGAS TEHNISKĀ UNIVERSITĀTE

Datorzinātnes un informācijas tehnoloģijas fakultāte

Lietišķo datorsistēmu institūts

Mākslīgā intelekta un sistēmu inženierijas katedra

Dmitrijs Odinokijs

Akadēmiskās bakalaura studiju programmas „Intelektuālās robotizētas sistēmas” students

(stud. apl. nr. 171RDB285)

**SKAŅAS AVOTU TEMBRU
ATPAZĪŠANA, IZMANTOJOT DZIĻAS
MAŠĪNAPMĀCĪBAS METODES**

Bakalaura darbs

Zinātniskais vadītājs
ĒVALDS URTĀNS
doktorants

Rīga 2020

DARBA IZPILDES UN NOVĒRTĒJUMA LAPA

Bakalaura darbs izstrādāts *Mākslīgā intelekta un sistēmu inženierijas katedrā*.

Ar parakstu apliecinu, ka visi izmantotie materiāli ir norādīti literatūras sarakstā un iesniegtais darbs ir oriģināls.

Darba autors:

stud. **D.Odinokijs**.....
(paraksts, datums)

Bakalaura darbs ieteikts aizstāvēšanai:

Zinātniskais vadītājs:

Ē.Urtāns.....
(paraksts, datums)

Bakalaura darbs pielaists aizstāvēšanai:

Mākslīgā intelekta un sistēmu inženierijas katedras vadītājs:

Dr.habil.sc.ing., prof. **A.Nikitenko**.....
(paraksts, datums)

Bakalaura darbs aizstāvēts Mākslīgā intelekta un sistēmu inženierijas katedras Gala

pārbaudījumu komisijas 2020 gada.....sēdē un novērtēts ar atzīmi ().
(gads) (datums, mēnesis)

Mākslīgā intelekta un sistēmu inženierijas katedras Gala pārbaudījumu komisijas

sekretāre.....
(uzvārds, paraksts)

ANOTĀCIJA

[dziļas mašīnmācības, neironu tīkli, datu kopas, digitālā signāla apstrāde, lineārā algebra]

Dotajā darbā ir publicēti pētījuma rezultāti, kas salīdzina īso audio ierakstu klasificēšanas metodes, izmantojot dziļās mašīnāpmācības algoritmus. Šādu algoritmu pielietošana attēlu klasifikācijas/atpazīšanas uzdevumiem pēdējos desmit gados ir kļuvusi par izplatītu praksi. Arī skaņas ierakstu datu kopas ir iespējams klasificēt līdzīgi attēlam, izmantojot digitālā signāla apstrādes metodes. Taču skaņas ierakstu klasificēšanā pastāv aspekti, kas padara šo specifisko uzdevumu par izaicinošu un netriviālu. Tajā pašā laikā aug publiski pieejamo akadēmisko datu kopu piedāvājums - gan audio ierakstu, gan saistīto pētījumu daudzums.

Šis darbs apskata vairākus dziļās mašīnāpmācības algoritmus (mākslīgo neironu tīklu uzbūves arhitektūras), ar mērķi paaugstināt klasificēšanas modeļu darbības precizitāti. Pētījuma gaitā tiek uzbūvētas bieži izmantots neironu tīklu arhitektūru modifikācijas, pielietotas uz datu kopas FSD (Fonseca et al. 2018), kas bija izmantota starptautiskās sacensībās resursā kaggle.com 2018. gadā.

Darba pamattekstā ir 53 lappuses, 19 attēli, 6 tabulas, 40 informācijas avoti un 5 pielikumi.

ABSTRACT

[deep learning, neural networks, datasets, digital signal processing, linear algebra]

The current paper contains short audio file classification methods research results, being focused on the deep learning algorithms. Such algorithms became widely used for image recognition over last decade. Audio files might be classified in the similar manner, when being represented by applying digital signal processing methods. However, there are specific aspects, which turns audio file classification into challenging and non-trivial task to solve. At the same time, the amount of publicly available audio file datasets is growing and there are more and more dedicated research publications available.

This research is focused on sound classification related specific aspects and describes several deep learning methods and data pre-processing techniques, aiming to build the best possible prediction model, based on the features extracted from the audio files. During the research the author implemented popular deep artificial neural network architectures, training, testing and validating them by applying the FSD (Fonseca et al. 2018) dataset, which was used during Freesound General-Purpose Audio Tagging Challenge in 2018.

Paper includes 53 pages as main text, 19 pictures, 6 tables, 40 references and 5 appendix pages.

АНОТАЦИЯ

[глубокое машинное обучение, нейронные сети, коллекции данных, цифровая обработка сигнала, линейная алгебра]

Данная работа содержит результаты исследования о методах классификации коротких аудио файлов, с акцентом на методы глубокого машинного обучения. Такие методы приобрели популярность для работы с распознаванием изображений за последние десятилетия. Аудио файлы могут быть распознаваемы применяя схожий подход, если задействуются методы цифровой обработки сигнала. Однако, существуют специфические аспекты работы с аудио, что делает задачу сложной и нетривиальной. В то же время появляется всё больше и больше публично доступных коллекций с данными, и исследований на эту тему.

Данное исследование сосредоточено на специфических аспектах классификации аудио и описывает несколько методов глубокого машинного обучения, а также способов предварительной обработки данных, с целью построить наиболее эффективную модель для классификации на базе извлечённых из аудио файлов признаков. По ходу исследования автор использовал популярные архитектуры глубоких нейронных сетей, обучая, проводя валидацию и тестируя их на основе коллекции данных FSD (Fonseca et al. 2018), который был использован для конкурса Freesound General-Purpose Audio Tagging Challenge в 2018 году.

Работа включает в себя 53 страницы основного текста, 19 изображений, 6 таблиц, 40 источников и 5 приложений.

SATURS

DARBA IZPILDES UN NOVĒRTĒJUMA LAPA	2
ANOTĀCIJA.....	3
ABSTRACT	4
АНОТАЦИЯ.....	5
SATURS.....	6
SAĪSINĀJUMU UN TERMINU VĀRDNĪCA	8
IEVADS	10
1. TEORĒTISKAIS PAMATOJUMS.....	14
1.1. Skaņas avotu klasifikācija un atpazīšana	14
1.2. Klasiskās metodes.....	15
1.3. Dziļas mašīnāpmācības metodes.....	16
1.3.1. Klasiskie neironu tīkli.....	17
1.3.2. Kļūdas atpakaļ-izplatīšanas algoritms	18
1.3.3. Gradienta numeriskās nestabilitātes problēma	21
1.3.4. Konvolūciju kodola funkcija	21
1.3.5. Retas attēlu iezīmes	23
1.3.6. Apmācības parametri.....	23
1.3.7. Kodola izmērs un pazīmju daudzums.....	24
1.3.8. ConvNets (Konvolūcijas neironu tīkli)	25
1.3.9. ResNets (noturīgie neironu tīkli)	25
2. METODOLOĢIJA	27
2.1.1. Izmantotās bibliotēkas un programmatūra	27
2.1.2. Izmantotās datu kopas	28
2.1.3. Datu nolasīšana un priekšapstrāde.....	30
2.1.4. Aktivācijas funkcija.....	31
2.1.5. Kļūdas funkcija.....	32
2.1.6. Softmax funkcija	33
2.1.7. Modeļu optimizēšanas metode	34
2.1.8. Iezīmju grupēšana kopās	35
2.1.9. Normalizēšana	35
2.1.10. Modeļu arhitektūra: ConvNet.....	35

2.1.1.	Modeļu arhitektūra: ResNet18.....	36
2.1.2.	Rezultātu analīzes metodoloģija (metrikas)	36
3.	EKSPERIMENTĀLĀ DAĻA	39
3.1.1.	Audio ierakstu priekšapstrādē un analīze	39
3.1.2.	Modeļu apmācība	41
3.1.3.	Validācija.....	42
3.1.4.	Rezultāti un vērtēšana.....	42
3.1.5.	Uzbūvēto arhitektūru salīdzinājums	43
4.	TALĀKIE PĒTĪJUMI	49
5.	SECINĀJUMI	52
6.	IZMANTOTĀ LITERATŪRA	54
	PIELIKUMI.....	57

SAĪSINĀJUMU UN TERMINU VĀRDNĪCA

VR – Virtual reality (angl): Virtuālā realitāte

AR – Augmented reality (angl): Paplašinātā realitāte

AMIR - Automatic Music Instrument Recognition (angl.): Automātiskā mūzikas instrumentu atpazīšana

SVM - Support Vector Machines (angl): Atbalstu vektora mašīna

PCA – Principal Component Analysis (angl): Principiālo komponentu analīze

LPC - Linear Prediction Coefficients (angl): Lineārās prognozes koeficienti

MFCC - Mel-Frequency Cepstral, angl: Mel frekvences cepstrālie koeficienti

ANN - Artificial Neural Networks, angl: Mākslīgie neironu tīkli

GPU - Graphics Processing Unit (angl): Grafikas apstrādes procesors

SGD - Stochastic Gradient Descent (angl): Stohastiskā gradienta nolaišanās

ADAM – Adaptive Moment (angl): Adaptīvs Moments

RADAM – Rectified ADAM: Taisnstūra ADAM

NaN – not-a-number (angl): Neskaitlisks mainīgā stāvoklis

CUDA – Compute Unified Device Architecture (angl): Vienotā skaitļošanas ierīču arhitektūra

API – An application programming interface (angl): lietojuma programmēšanas saskarsme

Terminu tulkojums

Attack and decay time (angl): Uzbrukuma un rimšanas laiks

Fully connected (angl): Pilnībā savienots

Bias (angl): Nobīde

Learning rate (angl): Apmācības ātrums

Kernel (angl): Konvolūcijas kodola funkcija

Batch size (angl): Paketes izmērs

Zero padding (angl): Matricas papildinājums ar 0

BatchNorm: Paketes normalizēšana

Dropout: Pārtraukums

Pooling: Iezīmju grupēšana kopās

Max Pooling: Iezīmju grupēšana kopās pēc maksimālas vērtības.

Adaptive Pooling (angl): Adaptīva iezīmju grupēšana kopās

Average Pooling: Iezīmju grupēšana kopās pēc vidējas vērtības

Root Mean Square error (angl): Vidējā kvadrātiskā kļūda

One-hot encoded vector (angl): “Karsti” iekodēts vektors

Fast Fourier Transform (angl): Ātrā Furjē transformācija

Categorical Cross-Entropy (angl): Kategoriska Cross-Entropija

Precision (angl): Precizitāte

Recall (angl): Jūtība

Confusion Matrix (angl): Sajukuma matrica

Overfitting (angl): Pārprielāgošanās

IEVADS

Problēmas aktualitāte un pielietojšanas piemēri

Digitālo audio ierakstu atpazīšana visbiežāk ir asociēta ar runas klasifikāciju un balss asistentiem - mākslīgo intelektu, kas spēj komunicēt ar cilvēku, izmantojot dzimto valodu, balsi un intonācijas. Šāda tipa sistēmām jāatpazīst cilvēka balss (vai arī citi runas avoti) trokšņainā fonā un jāspēj pareizi interpretēt informācija, ko pārnes runas avots.

Šādu sistēmu realizācija spētu, piemēram, izolēt klausītāju viņa austiņās trokšņainās telpās vai izolēt nevēlamus trokšņu avotus no ārējās vides. Audio ir neatņemama daļa no virtuālās un paplašinātās realitātes. Strauji attīstās paplašinātās realitātes implementācijas, kur reālās pasaules skaņas tiek apstrādātas pseido-reālā laikā¹.

Muzikālo audio ierakstu sadalīšana komponentēs (kas ir mūzikas instrumentu atsevišķas partijas) ir plaši pētīta mūzikas radīšanas programmatūras izstrādei, multimediju un izklaides industriju vajadzībām.

Es koncentrēšos tieši uz pēdējā aspekta, jo mana personīgā interese saistās ar mūzikas instrumentu izveidi un cilvēka emociju pētīšanu, kas rodas saistībā ar mūziku.

Audio atpazīšanas vēsturisks kopsavilkums

Mākslīgie neironu tīkli piesaistīja zinātnieku uzmanību trīs stadijās:

- Perceptrona algoritma izveide, 1957. g.².
- Kļūdu atgriezeniskas izplatīšanas algoritma izveide, 1986. g. (Alber, 2018)
- Dziļo mašīnāpmācību sasniegumi runāšanas un attēlu atpazīšanas jomās (Krizhevsky, Sutskever, and Hinton 2017)

Pēdējais notikums bija relatīvi nesen un ierosināja renesansi mākslīgo neironu tīklu pētīšanas jomā. Tad izcēlās divas algoritma modifikācijas: konvolūcijas neironu tīkli (Kim

¹ Sennheiser launches “AMBEO AR One” in-ears for Magic Leap One glasses (VR headset): <https://en-uk.sennheiser.com/sennheiser-launches-ambeo-ar-one-in-ears-for-magic-leap-one-glasses>

² <https://web.csulb.edu/~cwallis/artificialn/History.htm>

2014; Krizhevsky, Sutskever, and Hinton 2017) un “long short-term memory” vai LSTM (Schmidhuber, Gers, and Eck 2002). Šādā “dziļā” paradigmā, konstrukcijas ar vairāku parametru daudzumu ir apmācītas, izmantojot kļūdu atgriezenisko izplatīšanas algoritmu (Skansi 2018, para. 4.5), aprēķinot katra parametra ieguldījuma pakāpi kopīgā kļūdā un mēģinājumu veikšanā, mainot parametrus tā, lai sasniegtu vislabāko rezultātu. Šai pieejai ir nepieciešams kritiski liels datu apjoms, kas ir uzkrājies interneta (un sociālo tīklu³) izplatīšanās dēļ. Papildus tam, attīstījās paralēlas skaitļošanas rīki, tai skaitā grafiskās kartes⁴. Pielietojums algoritmiem uzreiz atradās digitālo signālu apstrādes jomā, bieži apsteidzot tradicionālas metodes (O’Mahony et al. 2020).

Tomēr sarežģītā atklādošana uzreiz izraisīja neuzticību ekspertu starpā, paradigma tika nosaukta par “melno kasti” un vēl joprojām nav skaidrs, kā iegūt cilvēkam saprotamu pamatojumu daudzslāņu arhitektūras pieņemtajiem lēmumiem. Taču šobrīd veidojas pētniecības joma, kas izmanto pašas neironu tīklu arhitektūru par pētījumu objektu (Buhrmester, Münch, and Arens 2019).

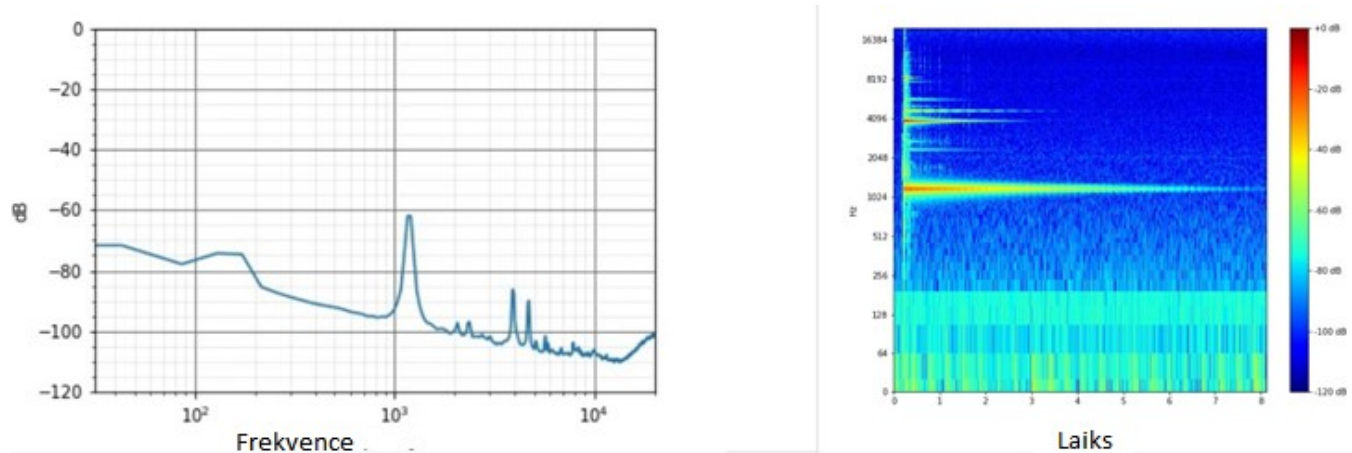
Šobrīd dziļās mašīnāpmācības sistēmas atrod sev pielietojumu dažādās jomās, to skaitā: fizikā, ķīmijā, astronomijā, valodas apstrāde un ekspert sistēmās.

Esošā nozares stāvokļa kopsavilkums

Uz doto brīdi iepriekš izmantotās metodes, tādas kā Gaussian mixture modeļi / slēptie Markova modeļi (Bilmes 2008; Marques and Moreno, 1999.) un Atbalstu Vektora Mašīnas (Zhao and Wang, 2019) nedod tik iespaidīgus rezultātus, salīdzinot ar dziļās mašīnāpmācības metodēm. Tās tiek attīstītas salīdzinoši lēni, tāpēc mēs nekoncentrēsimies uz tām šī darba ietvaros. Aktīvi un padziļināti tika pētītas iespējas lietot dažādas mašīnāpmācības metožu kombinācijas (piemēram, konvolūciju mākslīgos neironu tīklus un atbalstu vektora mašīnas) (Singh et al. 2019).

³ Sociālie tīkli un datu kopas: <https://ourworldindata.org/rise-of-social-media>

⁴ Grafisko procesoru vēsturiskā analīze: <https://www.techspot.com/article/2008-gpu-efficiency-historical-analysis/>



Attēls 1.1. Zvaniņa skaņas (notis **D₆**, 1174.66 Hz) vidējās jaudas spektrs un spektrogramma. Gan skaļums, gan frekvences veido nelineāro asi. Tā skaļums tiek mērīts relatīvi, decibelos. Bet frekvence pārveidota uz cilvēkam saprotamo hromatisko skatu, kur ir 12 pustoņi un oktāva. Tas arī darīts ar bildēm, lai tie būtu vizuāli pārskatāmi. Audio failu dati prasa specifisko analīzi un priekšapstrādi, ja mēs gribam tos klasificēt.

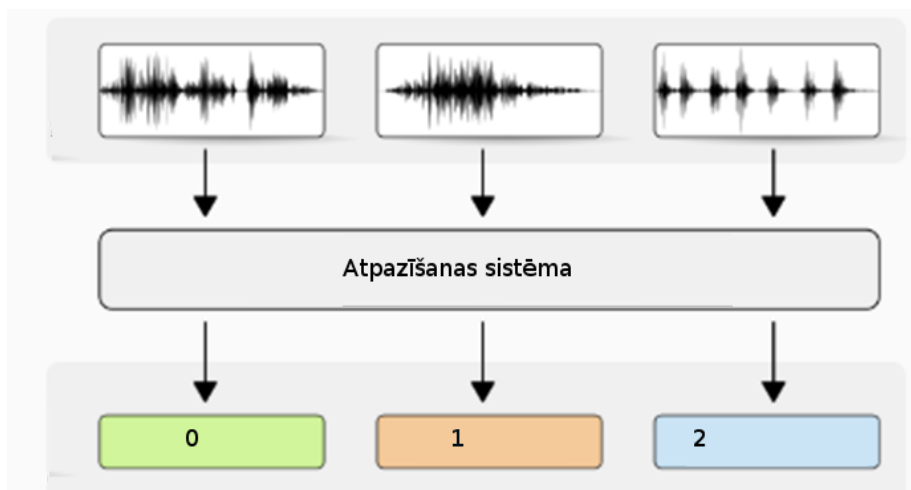
Vairākas dziļās mašīnāpmācības metodes tika attīstītas pateicoties attēlu apstrādes jomai. Pastāv ievērojamas atšķirības starp to pielietojumu darbam ar audio. Sākotnēji, audio veido vienas dimensijas signālu, kamēr attēli ir vismaz divdimensionāli. Audiosignāls bieži tiek transformēts vairāku dimensiju reprezentācijās darbam ar neironu tīkliem. Bet laika un frekvences asis nebūs homogēnas kā attēla gadījumā, piemēram, laiks ir lineārs, bet frekvenci bieži jāattēlo logaritmiskā asī. Bilde ir statiskas un parāda kādu objektu vai objektu kopu, bet audio bieži jāanalizē hronoloģiskā secībā un tam ir informatīvā nozīme. Tas liecina par to, ka ir jābūt specifiskākiem risinājumiem tieši audio klasifikācijas uzdevumiem.

Bakalaura darba mērķis:

Iegūt efektīvāku dziļās mašīnāpmācības metodi mūzikas instrumentu atpazīšanai no audio ieraksta.

Bakalaura darba uzdevumi:

- Uzbūvēt uz Python valodas pamata grafisko attēlu klasificēšanas modeļus un apmācību/testēšanas skriptus.
- Pārbaudīt modeļus uz Fashion MNIST datu kopas⁵, sasniegt precizitātes un F1 (2.1.2) metriku rādītājus tuvu 0.85.
- Realizēt kodā konvolūcijas neironu tīklus audio failu atpazīšanai. Izmantot datu izlasi ar dažādu tembru mūzikas instrumentu ierakstiem.
- Izveidot modeli, kas spēj atpazīt mūzikas instrumentus no FSD Kaggle⁶ (Fonseca, Plakal, Font et al. 2018) datu kopas.
- Formulēt audio klasifikācijas algoritma vērtēšanas parametrus un realizēt kodā to darbību.
- Salīdzināt vismaz 2 dažādu modeļu arhitektūras apmācības rezultātus. Novērtēt, izmantojot F1 metriku.
- Sagatavot prezentāciju ar salīdzinājuma rezultātiem, informatīvo grafiku un secinājumiem.



Attēls 1.2.
Projektējamas
sistēmas shematisks
zīmējums (aizgūts no
Fonseca, Plakal, Font
et al. 2018)

⁵ Apģērbu attēlu 10 klašu melnbalta datu kopa: <https://www.kaggle.com/zalando-research/fashionmnist>

⁶ Skaņas avotu datu kopa, samazinātā versija: <https://zenodo.org/record/2552860#.Xq6Y6qgzZhF>

1. TEORĒTISKAIS PAMATOJUMS

1.1. Skaņas avotu klasifikācija un atpazīšana

Matemātisks aparāts un algoritmu kopa, kas ļautu nolasīt, saglabāt un interpretēt lietderīgu informāciju no audio ierakstiem, atrod daudz pielietojumu praksē. Pieminot tikai dažus, var pateikt par:

- Automātiskām balss atpazīšanas sistēmām (Chiu et al. 2018),
- Automatizētām mūzikas analīzes sistēmām (Segal, 2016)
- Virtuālās un paplašinātās realitātes risinājumiem, VR & AR⁷.

Pirmais ļautu veidot datoru interfeisu tāda veidā, lai lietotājs būtu spējīgs komunicēt ar programmatūru izmantojot balsi, intonācijas un dzimto valodu. Otrais – projektēt mūzikas instrumentus un programmatūru ar paplašinātām iespējām, attīstītām funkcijām. Trešais – kombinēt akustiskās skaņas no reālās pasaules un virtuālus skaņas avotus AR vidē. Saraksts, protams, var būt paplašināts.

Mēs šī darba ietvaros bieži izmantosim terminu AMIR⁸. Kā jau minēts, šajā jomā nesen tika sasniegts ievērojams progress, un tas ir primāri saistīts ar dziļo neironu tīklu izpēti un metodoloģiju attīstīšanu, par ko mēs vēl runāsim šajā nodaļā.

Neskatoties uz progresu, AMIR algoritmi vēl ir jāpilnveido, lai tie būtu spējīgi tikpat labi (vai labāk) kā cilvēki atpazīt un klasificēt skaņas avotus. Salīdzinot AMIR tipa klasifikācijas uzdevumus ar citiem, piemēram, no fizikas vai bioloģijas jomas, tie varētu izskatīties daudz vienkāršāki, taču tā tas nav. Cilvēks izmanto dzirdi komunikācijai, navigācijai vidē un dara to nepārtraukti visas dzīves gaitā. Dzirdes un runas aparāti ir cieši integrēti centrālajā un perifērijas nervu sistēmās. Tas ir pārdomāts, ļoti precīzs evolūcijas brīnums, kurš ir pilns ar nelineāriem

⁷ Paplašinātās realitātes audio produkti, kuru izstrādei lieto dziļas mašīnāpmācības: <https://en-us.sennheiser.com/ambeo>

<https://en-us.sennheiser.com/ambeo-application-armr>

⁸ Automatic Music Instrument Recognition, angl

posmiem un atgriezeniskām saitēm. Tāpēc modelēt to, pilnībā izmantojot datoru, vēl joprojām ir liels izaicinājums.

1.2. Klasiskās metodes

AMIR kontekstā tika izstrādāti un pētīti daudzi algoritmi. Taču vairākas dekādes jomā dominēja algoritmi, kuros izmantotās klasificēšanas pazīmes tika ģenerētas ar cilvēku/ekspertu palīdzību. Piemēram, 90-to gadu beigās plaši pielietoja SVM⁹ klasifikatoru.

SVM (atbalstu vektoru mašīna) ir izplatīts un plaši pielietots algoritms datu klasifikācijai. Dažādi paraugu pazīmes raksturojošie parametri tiek attēloti kā vektoru kopa. Piemēram, paraugu augums un platums (divas pazīmes) var būt aprakstīti ar diviem skaitļiem Dekartu koordinātu sistēmā. Ģeometriski, gadījumā ar tikai diviem parametriem paraugi tiek attēloti kā punkti plaknē. Klasifikācijas uzdevums tādā piemērā ir tādas funkcijas atrašana, kas atbilstu līnijai, kas vislabāk atdalītu plaknē viena klases paraugus no citas klases. Pieaugot pazīmju daudzumam, tāda modeļi pieaug arī dimensiju skaits. Bet būtība nemainās – algoritms meklēs vislabāk klasificējošo funkcijas vienādojumu. Lai reducēt dimensiju skaitu un atrast tas dimensijas, kas vislabāk der klasifikācijai konkrētajā gadījumā, izmanto principiālo komponentu analīzes (PCA) algoritmu (Marques and Moreno, 1999).

Tā Marques and Moreno rakstā tiek apspriestas klasifikācijas pazīmes: lineārie prognozēšanas koeficienti LPC¹⁰, cepstrālie koeficienti un Mel-frekvences cepstrālie koeficienti¹¹ (2.1.3). Rezultātā aptuveni 70% audio fragmentu tika klasificēti pareizi. Eronens and Klapurijs attīstīja citu pieeju, kurā viņi izstrādāja vairāk nekā 20 pazīmes (Eronen and Klapuri 2000). Tās tiek veidotas savstarpēji kombinējot MFCC, spektrālo centroīdi¹², uzbrukuma un rimšanas laiku¹³, kā arī vairākas citas pazīmes. Tās jau tika balstītas gan uz spektrālām īpašībām, gan laika-dinamikas īpašībām.

⁹ Support Vector Machines, angl: <https://www.mathworks.com/discovery/support-vector-machine.html>

¹⁰ Linear Prediction Coefficients, angl: <https://www.mathworks.com/help/signal/ref/lpc.html>

¹¹ Mel Frequency Cepstral Coefficients (De La Calle Silos 2017, chap. 3), angl.

¹² Spectral Centroid, angl: <https://www.mathworks.com/help/audio/ref/spectralcentroid.html>

¹³ Attack and decay, angl.

Bet, neskatoties uz pazīmju daudzveidību un skaitu, modeļa rezultāti bija daudz neprecīzāki nekā uzdevums padodas cilvēkiem: individuālie skaņas avoti bija pareizi klasificēti tikai 80% gadījumu.

Pazīmju formulēšanas paradigma tika izmantota vēl vairākos pētījumos. Taču tikai pēdējos 10 gados atkal aktivizējās dziļās mašīnāpmācības pētniecība AMIR jomā. Pēc Choi, Fazekas un Sandler (Choi et al, 2016) dziļās mašīnāpmācības algoritmi de-facto kļuva par standartu datorredzes jomā. Algoritmi tiek pielietoti arī audio atpazīšanā, rādot ievērojami labākus rezultātus nekā kanoniskās mašīnāpmācības metodes. Nākošie paragrāfi būs veltīti tam, kas ir mākslīgie neironu tīkli un dziļās mašīnāpmācības metodes.

Darba ietvaros nav plānots iedziļināties klasiskās metodēs. Kā ir aprakstīts vēlāk, mūsdienu industrijā audio failu klasifikācijas uzdevumiem dominē dziļās mašīnāpmācības metodes (Segal, n.d.; Zhao and Wang 2019). Tāpēc es koncentrēšos tieši tajā jomā.

1.3. Dziļās mašīnāpmācības metodes

Mākslīgie neironu tīkli (ANN¹⁴) ir skaitļošanas modeļu kategorija. ANN izpēti iedvesmoja bioloģiskie prototipi, it īpaši dzīvnieku smadzeņu struktūra, to spēja mācīties un risināt sarežģītas problēmas (Basheer and Hajmeer 2000). Nosaukums “neironu tīkli” atbilst to struktūras īpašībai: datu glabāšanas vai apstrādes vienības (grafu virsotnes, kas satur konstantes, mainīgos vai funkcijas) tiek organizētas slāņos. Virsotnes veido savienojumu ar citu slāņu virsotnēm caur lokiem, kuriem piešķirti svāri. ANN pētnieki izmanto šādas struktūras, lai risinātu sarežģītas problēmas, kur citu metožu izmantošana nav iespējama.

Precizējot, mūs lielākoties interesē dziļie neironu tīkli ANN ar vairākiem “slēptiem” starpslāņiem. Tā kā tie kļuva īpaši populāri sākot no 2012. gada ar AlexNet arhitektūras uzvaru ImageNet sacensībās¹⁵ (Krizhevsky, Sutskever, and Hinton 2012), cilvēki par to bieži runā kā par jaunām tehnoloģijām. Taču ideja, kas stāv aiz tā visa paradās publiskajā diskusijā jau kopš 20. g.s. 1940-tajiem gadiem (McCulloch and Pitts, 1943).

Atjaunota interese par šo jomu parasti tiek saistīta ar diviem faktoriem. Pirmkārt, patiecoties sociālajiem tīkliem un informācijas izplatībai mūsdienu pasaulē, tika uzkrāti milzīgi

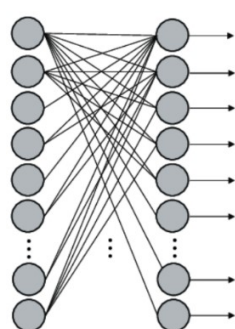
¹⁴ Artificial Neural Networks, angl.

¹⁵ Sacensību mājas lapa: <http://www.image-net.org/>

datu apjomi un to uzkrāšanas ātrums aug joprojām. Datu kopas apjoms ir primārais nosacījums dziļo mākslīgo neironu tīklu darbībai, un šādi apjomi nebija pieejami nekad iepriekš. Milzīgā datu apjoma apstrāde prasa vērienīgus skaitļošanas resursus. Otrs iemesls dziļās mašīnāpmācības popularitātes pieaugumam ir saistīts ar šo faktu. Tieši pēc 2010. gada tirgū parādījās ļoti efektīvi grafikas apstrādes procesori (GPU¹⁶). Patiecoties videospēļu industrijai, GPU tika pietiekami labi optimizēti un sasniedza relatīvi zemas ražošanas izmaksas, kas ietekmēja to tirgus cenu un pieejamību. Tika izveidotas un kļuva publiski pieejamas zema līmeņa bibliotēkas, tādas kā Tensorflow (Abadi et al. 2016) un PyTorch¹⁷, kas ļauj izmantot GPU sarežģīto modeļu aprēķiniem.

Kā jau minēts, neironi (virsotnes) mākslīgos neironu tīklos tiek sagrupēti slāņos. Pastāv trīs slāņu tipi: ieejas, izejas un slēptie. Ieejas slānis paredzēts ieejas datu ielādēšanai apmācību vai testēšanas procesā. Piemēram, katrs melnbalta attēla pikselis būs reprezentēts ar nenegatīvo decimālo skaitli, kas tiek saglabāts vienā no ieejas neironiem. Slēptie slāņi tiek izmantoti, lai apkopotu ieejošos datus. Jo dziļāk būs tīkls, jo sarežģītākas klasifikācijas pazīmes tas spēs ģenerēt un izmantot. Piemēram, klasificējot automobiļus, pirmais slēptais slānis spēj atšķirt kontūras, otrais – līniju krustojumus, trešais - krāsu likumsakarības. Dziļākie slāņi vairāk darbojas uz abstraktām formām, kurām piemīt formu un konfigurāciju daudzveidība. Visa sistēma mācību procesā automātiski veido slāņu konfigurāciju un pēc apmācības spēj patstāvīgi pieņemt klasifikācijas lēmumus.

1.3.1. Klasiskie neironu tīkli



Tradicionālie mākslīgie neironu tīkli ir savienoti¹⁸: katrs neirons ir savienots ar pilnīgi visiem neironiem blakus stāvošā slānī (tieši vai netieši). Kad mēs sākam “neirons ir savienots ar citu neironu”, mēs gribam pateikt, ka viena neirona izejas signāls tiek padots kā ieejas signāls citam neironam. Kā mēs aprēķinām izejas vērtību? Matemātiski,

Attēls 1.3. Pilnībā savienots mākslīgais neironu tīkls. (aizgūts no deeplarning.ai, 2018)

¹⁶ Graphics Processing Unit, angl.

¹⁷ Atvērtā koda vides, kas ir plaši pielietotas mašīnāpmācības uzdevumu risināšanai: <https://www.tensorflow.org/>, <https://pytorch.org/>

¹⁸ Fully connected, angl.

loka (savienojuma) svars grafā tiek reizināts ar ieejas vērtību, tiek pieskaitīta nobīde¹⁹ un rezultāts tiek pasniegts kā arguments kādai no nelineārām funkcijām (aktivācijas funkcijas)²⁰. Tā kā runa bieži iet par milzīgiem grafiem, daudz ērtāk ir runāt par šīm operācijām kā par matricu reizināšanu (visu svaru matrica ar visu ieejas vērtību matricu. Mēs varam apskatīt sekojošo vienādojumu:

$$y^j = \sigma(w^j y^{j-1} + b^j) \quad (1.1)$$

Izteiksmē tiek aprēķināta slāņa izeja y^j . Mēs ņemam iepriekšējā slāņa $y - 1$ izeju un reizinām to ar savienojamo loku svaru matricu (Goodfellow, Bengio, and Courville 2016, chap. 9). Pievienojam nobīdi un padodam tālāk uz aktivācijas funkciju σ . Aktivācijas funkcijas būs apskatītas tālāk tekstā.

Tālāk padotais rezultējošais vektors ir neironu tīkla darbības rezultāts un bieži nozīmē lēmumu, klasificēšanas rezultātu. Lēmums var būt absolūti pareizs vai nepareizs. Var atrasties arī starp pareizo un nepareizo, tad to var novērtēt ar pareiza lēmuma pieņemšanas varbūtību. Lai novērtētu darbības rezultātu un liktu neironu tīklam salabot savus iekšējos parametrus, izmanto kādu no kļūdas funkcijām (2.1.5). Pastāv iespēja likt neironu tīklam mainīt savus svarus un nobīdes vērtības tā, lai ātrāk tiktu pie pareizā rezultāta un tādā veidā mācītos no piemēriem (2.1.7).

Par tradicionāliem neironu tīkliem var uzskatīt “nedziļus” grafus ar tikai vienu slēpto slāni. Bet, tā kā literatūrā pastāv noteikts terminoloģisks sajukums, arī pilnībā savienotus dziļos tīklus mēdz saukt par tradicionāliem.

1.3.2. Kļūdas atpakaļ-izplatīšanas algoritms

Atpakaļ-izplatīšanas algoritmā realizācija nodrošina mākslīgo neironu tīklu spēju mācīties no piemēriem. Pats termins pirmo reizi minēts 1986. gadā (Rumelhart, Hinton, and Williams, 1986), bet pati metode izmantota kopš 1960. gadiem (Henry J. Kelly, Stuart Dreyfus,

¹⁹ Bias, angl: Nobīde

²⁰ Aktivācijas funkcijas definē virsotnēs izeju, pēc argumenta kopas padošanas uz ieeju (Goodfellow, Bengio, and Courville 2016)

Bryson & Ho un citi). Pati vienkāršākā algoritma versija (Stohastiskā Gradianta Nolaišanās, SGD²¹) sastāv no sekojošiem posmiem:

- 1) Ieejas datu kopa tiek nodota caur mākslīgo neironu tīklu (katra nākama slāņa izejas skaitļošana tiek aprakstīta ar formulu 1.1. Rezultāta tiek iegūta prognozētā vērtība.
- 2) Prognozētā vērtība tiek salīdzināta ar pareizo atbildi (apmācības paraugu), izmantojot kļūdas funkciju (2.1.5). Atgrieztā vērtība parāda, par cik uzģenerētā vērtība nesakrīt ar paraugu un cik daudz to pārsniedz/nesasniedz.
- 3) Iepriekšējam slānim tiek aprēķināts katra svara un nobīdes ieguldījums kopīgajā kļūdā – tiek rēķināts parciāls atvasinājums un to summa veido gradientu (Bottou 2012)

Visvienkāršākā kļūdas funkcija tiek definēta kā novirze no sagaidītās vērtības:

$$C_0 = \sum_{j=0}^{n-1} (a_j^{(L)} - y_j)^2 \quad (1.2)$$

Kur $a_j^{(L)}$ ir gaidāmā vērtība virsotnē j slānī L. Mēs varam uzskatīt to par visu iepriekšējo slāņu virsotņu kļūdas summu.

Katras kartējās virsotnes izeja slānī L tiek pierakstītā kā:

$$z_j^{(l)} = \sum_{k=0}^{n-1} w_{jk}^{(l)} a_k^{(l-1)} \quad (1.3)$$

Pie tam aktivācijas funkcijas g (2.1.4) izeja $a_j^{(l)}$ tiek pierakstītā kā:

$$a_j^{(l)} = g^{(l)}(z_j^{(l)}) \quad (1.4)$$

Aprakstot katras atsevišķas virsotnes hipotētisko kļūdu veidojas ķēde:

$$C_{0j} = C_{0j}(a_j^{(L)}(z_j^{(L)}(w_j^{(L)}))) \quad (1.5)$$

Algoritma mērķis ir iteratīvi modificēt svarus (un nobīdes) tā, lai minimizētu kļūdu un pietuvotos gaidāmajai vērtībai (nobīdes vērtības šobrīd ir izņemtas no izteiksmēm, lai vienkāršotu notācību).

²¹ Stochastic Gradient Descent (Bottou 2012), angl.

Apskatīsim kādu svaru, kurš savieno virsotni i slānī $L-1$ ar virsotni j slānī L . Šāds svars tiek pierakstīts kā $w_{ij}^{(L)}$, un C_0 atvasinājums pēc šī svara ir $\frac{\partial C_0}{\partial w_{ij}^{(L)}}$.

Tā kā visi mainīgie ir atkarīgi viens no otra atbilstoši izteiksmēm 1.5, mēs pielietosim atvasinājumu ķēdes likumu²² un, lai atvasinātu C_0 pēc svara $w_{ij}^{(L)}$, drīkst paņemt parciālo atvasinājumu reizinājumu pēc katra no šiem mainīgajiem:

$$\frac{\partial C_0}{\partial w_{ij}^{(L)}} = \left(\frac{\partial C_0}{\partial a_i^{(L)}} \right) \left(\frac{\partial a_i^{(L)}}{\partial z_i^{(L)}} \right) \left(\frac{\partial z_i^{(L)}}{\partial w_{ij}^{(L)}} \right) \quad (1.6)$$

Rēķinot parciālos atvasinājumus tiek iegūts:

$$\frac{\partial C_0}{\partial w_{ij}^{(L)}} = 2 \left(a_i^{(L)} - y_i \right) \left(g'^{(L)} \left(z_i^{(L)} \right) \right) \left(a_j^{(L-1)} \right) \quad (1.4)$$

Gadījumā ar vairākiem apmācību piemēriem, kopīgais atvasinājums tiek rēķināts kā vidējais aritmētiskais no visu piemēru kopas, un procedūra tiek atkārtota, lai aprēķinātu C atvasinājumu pēc katra no svariem:

$$\frac{\partial C}{\partial w_{ij}^{(L)}} = \frac{1}{n} \sum_{i=0}^{n-1} \frac{\partial C_i}{\partial w_{ij}^{(L)}} \quad (1.5)$$

Rezultātā aprēķinātais atvasinājums tiek reizināts ar mācīšanās koeficientu η (learning rate, angl.) un atņemts no apskatītā svara, lai iegūtu jaunu vērtību (Goodfellow, Bengio, and Courville 2016, chap. 4.3).

$$\tilde{w}_{ij}^{(L)} = w_{ij}^{(L)} - \eta \frac{\partial C_0}{\partial w_{ij}^{(L)}} \quad (1.6)$$

Procedūra tiek atkārtota ar visiem svariem (un nobīdēm), visos slāņos. Tad sākas nākamā iterācija un nākamais apmācības cikls.

²² Parciālo atvasinājumu ķēdes likums: <http://mathonline.wikidot.com/derivative-chain-rule> ("Derivative Chain Rule - Mathonline")

1.3.3. Gradianta numeriskās nestabilitātes problēma

Palielinot slāņu daudzumu, mēs neizbēgami saskaramies ar problēmu, kad kļūdas atpakaļ-izplatīšanas algoritmā aprēķinātais gradients kļūst nestabils (ir spējīgs neprognozējami paņemt ļoti mazas vai ekstremāli lielās vērtības).

Gradients dziļās mašīnāpmācības kontekstā ir kļūdas funkcijas parciāls atvasinājums pēc katra no sistēmas svāriem, kas tika apspriests iepriekšējā nodaļā. Aiz tā stāv intuīcija, ka algoritms cenšas atjaunot visus parametrus (svarus un nobīdes) tā, lai minimizētu kļūdu. Tas ir iespējams, pielietojot atvasinājumu ķēdes likumu. Taču, pieaugot slāņu daudzumam, atvasinājumu ķēdes paliek ļoti sarežģītas, grādiņa vērtība var tikt tuvu 0 un tad, aprēķinot ķēdi, kopējais reizinājums arī tiecas uz 0. Tas, savukārt, atjauno svarus uz nenoīmīgu vērtību. Ar katru jaunu apmācību ciklu situācija tikai pasliktinās, paralizējot sistēmas spēju mācīties²³.

Pretējas gadījums ir saistīts ar numeriski nestabilo gradientu. Ja algoritma darbības rezultātā kāds no parametriem pārsniedz 1, veidojas tendence bezgalīgam grādiņa pieaugumam.

1.3.4. Konvolūciju kodola funkcija

Konvolūcija ir viena no centrālajām operācijām signāla apstrādes teorijā. Tā tiek izmatota audio analīzē, attēlu apstrādē un, kā redzēsīm tālāk, neironu tīklos. Ideja, kas slēpjas aiz konvolūcijas ir secīga filtra pielietošana vienas vai vairāku asu garumā ieejas signāla $x(t)$ nolasēm. Pie katra ieejas soļa mēs to reizinām ar filtra vērtību un summējam, tādā veidā iegūstot izejas nolasi. Vienas dimensijas konvolūcijas operācija tiek aprakstīta ar sekojošo formulu:

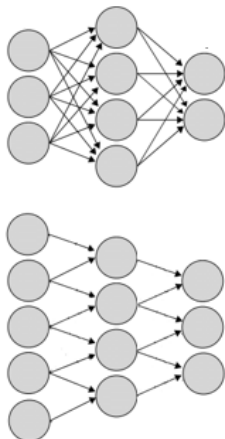
$$(x * h)[n] = \sum_{m=-\infty}^{\infty} x[m]h[n - m] \quad (1.2)$$

Kur x ir ieejas signāls un h ir filtrs. Konvolūcija bieži tiek izmantota, lai pievienotu efektus signālam, tādus kā nogludināšana un saasināšana. Bieži tiek pielietota arī vairāku dimensiju konvolūciju izmantošana. Piemēram, apstrādājot digitālus attēlus mēs varam pielietot 2D filtru un nobīdīt to divu asu garumā (melnbalta attēla gadījumā).

²³ Nestabilā grādiņa problēma: <https://ru.coursera.org/lecture/deep-neural-network/vanishing-exploding-gradients-C9iQO> ("Vanishing / Exploding gradients - Practical aspects of Deep Learning" n.d.)

1.3.5. Retas attēlu iezīmes

Svarīgas pazīmes nereti ir relatīvi mazas un aizņem tikai daļu no visa attēla. Piemēram, tajā var būt tūkstošiem pikseļu, bet, attēlotā dzīvnieka acis ir veidotas tikai no nelielas to daļas. Konvolūcija ļauj detektēt un atdalīt tādas pazīmes caur konvolūciju un tās kodola funkciju²⁴. Kodols šāda tipa neironu tīklos var būt ļoti mazs, kā 3x3 vai 5x5 pikseļi. Runājot par aprēķinu sarežģītību, tas viennozīmīgi pilnveido kanonisko pieeju. Attēlā 1.6 tiek parādīta arhitektūras struktūras shēma. Kā var redzēt, grafā ir daudz mazāk loku, nekā attēla 1.1. shēmā.



Attēls 1.6. Pilnībā savienots tradicionāls neironu tīkls (augšā) un konvolūcijas neironu tīkls (lejā). (aizgūts no deeplarning.ai, 2018)

Izmantojot matricu reizināšanu sistēmā ar M ieejām un N izejām, nepieciešami $M \times N$ parametri, bet aprēķinu sarežģītība būs $O(M \times N)$ uz vienu apmācību piemēru. Ierobežojot savienojumu skaitu atbilstoši mūsu konvolūcijas kodola izmēram K (kur K ir ievērojami mazāks par M), sistēma ietver tikai $K \times N$ savienojumu. Rezultātā samazināsies atmiņas patēriņš. Operāciju skaits uz vienu datu piemēru arī samazinās, kas ļauj ievērojami ietaupīt kopīgo aprēķinu laiku (Goodfellow, Bengio, and Courville 2016, chap. 9.8).

0	1	2
3	4	5
6	7	8

*

0	1
2	3

=

19	25
37	43

Attēls1.1.
Konvolūcijas operācijas piemērs. Tās ieeja, kodols un rezultāts.

1.3.6. Apmācības parametri

Konvolūcijas izmantošana matricu reizināšanas vietā noved pie vēl viena svarīga paņēmiena – parametru koplietošanas. Izpildot konvolūciju, mēs nobīdam kodolu dimensijas garumā, lai aprēķinātu izejas vērtības. Varam nosaukt katru kodola elementu par parametru. Tādā veidā kodols ar izmēru 3 uz 3 elementiem satur sevī 9 parametrus, bet 5 uz 5 elementiem satur 25 parametrus. Katru reizi, kad mēs bīdam kodolu un izmantojam to, lai aprēķinātu izejas

²⁴ Kernel, angl.

vērtības, mēs ņemam vienus un tos pašus parametrus. Tas ir ne tikai viens no ātrdarbības iemesliem, bet arī ļauj sistēmai atdalīt pazīmes neatkarīgi no tā, kurā vietā datu fragmentā tās atrodas. Pazīmju padarīšana par neatkarīgām no izvietojuma ir viena no pamata priekšrocībām konvolūcijas neironu tīklu izmantošanai, salīdzinot ar citiem veidiem²⁵.

1.3.7. Kodola izmērs un pazīmju daudzums

Viens no konvolūcijas kodola globālajiem parametriem ir tā izmērs. Pareiza izmēra izvēle vienmēr ietekmē rezultātu, jo nosaka pazīmju izmēru, ko atdalīs sistēma. Ja kodols ir pārāk mazs, sarežģītas struktūras netiks pamanītas. Ja pārāk liels – pazīmes var kļūt vispārīgas.

Vēl viens nozīmīgs parametrs ir pazīmju daudzums. Pārāk daudz parametru var ievest pārāk lielo redundanci, bet pārāk maz varētu kavēt spēju atpazīt sarežģītas likumsakarības.

Svarīgākie parametri konvolūcijas neironu tīkliem ir:

- Datu paketes izmērs²⁶
- Kodola funkcijas izmērs²⁷
- Papildinājums vai papildus rindas un kolonnas ar 0 vērtībām apkārt matricai ar datiem
- Solis, nobīdes solis, uz kuru pārvietojas kodola funkcija²⁸

0	0	0	0	0
0	0	1	2	0
0	3	4	5	0
0	6	7	8	0
0	0	0	0	0

 \star

0	1
2	3

 $=$

0	3	8	4
9	19	25	10
21	37	43	16
6	7	8	0

Attēls 1.7. Matricas papildinājums ar 0 - papildus šūnas ar 0 vērtībām apkārt matricai ar datiem. Lai būtu iespējama konvolūcijas operācija.

²⁵ <https://cs231n.github.io/convolutional-networks/> ("CS231n Convolutional Neural Networks for Visual Recognition", 2020)

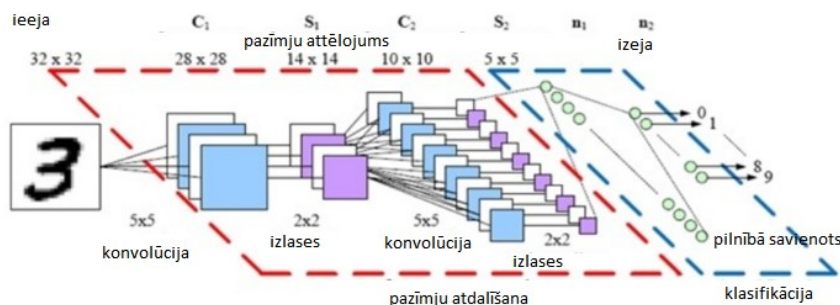
²⁶ Batch size, angl.

²⁷ Kernel size, angl.

²⁸ Stride, angl. https://d2l.ai/chapter_convolutional-neural-networks/padding-and-strides.html ("6.3. Padding and Stride — Dive into Deep Learning 0.7.1 Documentation", 2018)

1.3.8. ConvNets (Konvolūcijas neironu tīkli)

Skaidrojot vienā teikumā, konvolūcijas mākslīgie neironu tīkli (ConvNets) izmanto divu dimensiju konvolūciju svaru aprēķinam (nevis matricu reizināšanu) (Krizhevsky,



Attēls 1.4. AlexNet arhitektūra priekš rokraksta ciparu simbolu atpazīšanas (aizņemts Krizhevsky, Sutskever, and Hinton 2012)

Sutskever, and Hinton 2012). Šī atšķirība no tradicionālās arhitektūras ievērojami ietekmē aprēķina rezultātu. Kā jau aprakstīts, tradicionālie

mākslīgie neironu tīkli ir pilnībā savienoti (1.3.1).

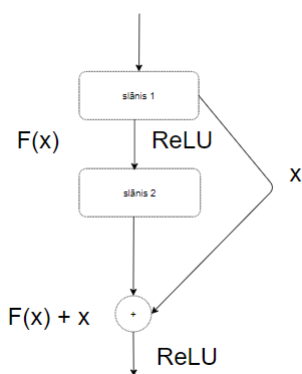
Daudzu virsotņu gadījumos grafu aprēķins prasa ievērojamus skaitļošanas resursus. Izmantojot konvolūciju matricu reizināšanas vietā ir iespējams optimizēt šo procedūru²⁹.

Detalizēts konvolūcijas apraksts nav apskatāms šī darba ietvaros, taču tālāk mēs mēģināsim aprakstīt vismaz pašus pamatus, lai beigās labāk saprastu praktiskajā daļā realizēto.

Jau klasikā realizācija ir tieši AlexNet (1.3) – arhitektūrā, kuras uzvara tagad bieži asociējas ar modernā perioda sākumu dziļās mašīnāpmācības jomā (attēls 1.2).

1.3.9. ResNets (noturīgie neironu tīkli)

Konvolūcijas mākslīgo neironu tīklu veiksmīga pielietošana sākot no 2012. gada ierosināja vairākus pētījumus³⁰, kuros tika veikti eksperimenti ar tīklu topoloģijām. Arhitektūras kļuva aizvien sarežģītākas un paslēpto slāņu daudzums palielinājās. Viena no dziļo mākslīgo neironu tīklu priekšrocībām ir spēja aproksimēt ļoti sarežģītas funkcijas. Šī spēja ir atkarīga no paslēpto slāņu daudzuma.



Attēls 1.8. ResNet shematisks skaidrojums.

Kā jau minēts, viens no nozīmīgākiem šķēršļiem dziļo topoloģiju pielietošanai ir

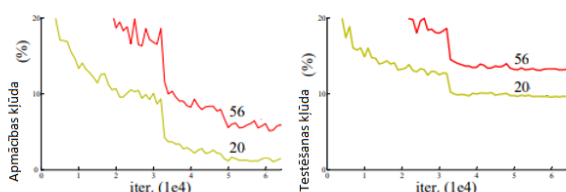
²⁹ <https://hackernoon.com/-understanding-convolution-neural-networks-cnn-the-eli5-way-photo-by-efe-kurnaz-on-unsplash-u-pali327j> ("What Are Convolution Neural Networks? [ELI5] | Hacker Noon", 2019)

³⁰ https://hai.stanford.edu/sites/default/files/ai_index_2019_report.pdf

gradienta numeriskās nestabilitātes problēma (1.3.3), kas izpaužas nepieciešamā skaitļošanas resursa un modeļu apmācībai nepieciešamā laika palielināšanās.

Risinājums, ko piedāvāja 2015. gadā Microsoft Research komanda³¹ un ieguva uzvaru ImageNet sacensībās ir sekojošs. Tika piedāvāts saglabāt sākotnējos datus no ienākoša slāņa un nodot tos tālāk cauri vairākiem slāņiem, summējot tos pie ienākošo ieeju vērtībām. Kopumā, to var izteikt ar formulu:

$$y = \mathcal{F}(x, \{W_i\}) + x \quad (1.3)$$



Kur x ir ieejas un y izejas vektori atbilstošos slāņos, funkcija $\mathcal{F}(x, \{W_i\})$ nozīmē starpības attēlu. Piemēram, attēlā 1.4, $\mathcal{F} = W_2 \sigma(W_1 x)$. σ apzīmē ReLU (2.1.4) funkciju un nobīdēs (bias, angl.) vērtības netiek radītas, lai vienkāršotu

pierakstu. Operācija $\mathcal{F} + x$ apzīmē saīsnī, kas arī ir parādīta Attēlā 1.4.

Rezultātā Microsoft Research pētniekiem izdevās ievērojami optimizēt algoritma darbību, kas tiek parādīts Attēla 1.8 grafikos (He, Zhang, Ren, Sun. 2015):

Attēls 1.9. Apmācības un testēšanas kļūdas samazināšanas grafiki 20 un 50 slāņu arhitektūras gadījumos. Attēlā tiek apskatītas ResNet arhitektūru modifikācijas attiecībā uz tradicionālās konvolūcijas modeļiem. Apmācības notika uz CIFAR-10³² datu kopas. (Aizgūts no He, Zhang, Ren, Sun, 2015)

Ir jāņem vērā, ka, lai ResNet saīsrē vispār būtu iespējama, ir jāsakrīt x un \mathcal{F} izeju izmēriem³³. Ja nosacījums ir ievērots, ir iespējams eksperimentēt ar saīsnēm cauri vairākiem slāņiem, veidojot dziļas un sazarotas tīklu topoloģijas, kas arī tiek pielietots mūsdienas. Piemēram, audio klasificēšanas uzdevumu risināšanai.

³¹ <https://blogs.microsoft.com/ai/microsoft-researchers-win-imagenet-computer-vision-challenge/>

³² Datorredzes modeļu apmācības datu kopa, <https://www.kaggle.com/c/cifar-10/overview>

³³ <https://medium.com/@14prakash/understanding-and-implementing-architectures-of-resnet-and-resnext-for-state-of-the-art-image-cf51669e1624> (Jay, 2018)

2. METODOLOĢIJA

2.1.1. Izmantotās bibliotēkas un programmatūra

PyTorch ir programmēšanas bibliotēku kopums, kas ir balstīts uz Python³⁴ valodas un ir paredzēts zinātniskai skaitļošanai. PyTorch pamatuzdevumi ir:

- Nodrošināt augsta līmeņa rīkus grafisko procesoru izmantošanas aprēķinu veikšanai
- Dziļās mašīnāpmācības pētīšanas platforma, kas nodrošinās maksimālu elastību un ātrumu³⁵

PyTorch labi integrējas ar citam pazīstamam Python bibliotēkām, tādām kā NumPy³⁶, Pandas³⁷, Sci-Py³⁸ un Librosa³⁹. Tas atvieglo to apgūšanu un ļauj izmantot Python valodas priekšrocības jebkurā datorā, kas aprīkots ar grafisko procesoru rīku mākslīgo neironu tīklu izmeklēšanai un praktiskiem pielietojumiem.

Bibliotēku pamatā ir tensora jēdziens (Dullemond and Peeters, 2010) – matemātisks objekts, kas pārveido vienas lineārās telpas elementus uz citu telpu elementiem. Pēc būtības, tā ir daudzdimensionāla matrica un var būt aprakstītā kā matricu reizinājums.

PyTorch tensori ļauj programmiski veidot tādus objektus, un tie uzreiz manto vairākas īpašības, kas ir noderīgas grafu uzbūvei un konkrēti eksperimentiem, kas apskata dziļas mašīnāpmācības. Piemēram, jau uzreiz ir iebūvēta iespēja izplatīt apmācības datu kopu “uz priekšu” (veikt prognozi) un veikt kļūdas atpakaļ-izplatīšanos. Bibliotēkas satur spējas ērti veidot daudzveidīgas arhitektūras, piedāvājot jau gatavus moduļus ar realizācijām (kā piemērs ir kļūdas un aktivācijas funkcijas, modeļu klases). Mēs izmantosim vairākus gadījumus, dažus elementus būvēsim patstāvīgi, izmantojot elementāras darbības, formulas un NumPy.

³⁴ <https://www.python.org/>

³⁵ https://pytorch.org/tutorials/beginner/blitz/tensor_tutorial.html

³⁶ <https://numpy.org/>

³⁷ <https://pandas.pydata.org/>

³⁸ <https://www.scipy.org/>

³⁹ <https://librosa.github.io/librosa/>

Runājot par mašīnāpmācības platformām, svarīgi pieminēt arī Tensorflow⁴⁰. To var uzskatīt par galveno alternatīvu PyTorch. Tensorflow bija izstrādāts Google Brain⁴¹ pētījuma ietvaros un piedāvā vairākas bibliotēkas modeļu būvēšanai, apmācībai (tai skaitā uz grafiskajiem procesoriem) un testēšanai. Tajā ir iekļauti vairāki augsta līmeņa rīki, un kopumā, man personīgi vide liekas nedaudz vienkāršāka un saprotamāka iesācējiem. Parasti Tensorflow tiek lietots ar Keras API⁴², kurā tiek piedāvāti vairāki bloki un funkcijas mašīnāpmācības modeļu izveidei. Tomēr darbs ar augsta līmeņa bibliotēkām bieži neļauj līdz galam saprast grafu programmēšanas un dziļas mašīnāpmācības principus. Tāpēc, jau sākot rakstīt šo darbu, radās jautājums, kurai no platformām veltīt vairāk uzmanības, lai veiksmīgi pabeigtu pētījumu. Lēmums bija sekojošs: mēģināt pakāpeniski risināt dažādus klasifikācijas uzdevumus, izmantojot numpy un PyTorch un beigās mēģināt realizēt audio failu priekšapstrādes un klasificēšanas modeļus. Paralēli uzrakstīt līdzīgu sistēmu, izmantojot TensorFlow piedāvātos augstākā līmeņa rīkus. Kopumā, ar TensorFlow man izdevās sasniegt labākus rezultātus (par to tiek stāstīts nākamajās sadaļās).

2.1.2. Izmantotās datu kopas

Freesound.org⁴³ ir starptautisks projekts, izveidots audio entuziastiem un profesionāliem pētniekiem. Pēc būtības tā ir platforma, kurā cilvēki publicē audio ierakstus, pievienojot dažādus mārkērus (lokācija, skaņas avota tips, tonalitāte, dinamika un citi). 2018. gadā Kaggle organizēja programmēšanas sacensības, kurās tika piedāvāts uzlabot esošo vai uzbūvēt jaunu skaņas avotu klasificēšanas modeli datu kopai, kas tika speciāli uzbūvēta sacensībām. Datu kopas ir FSD Kaggle 2018 (Fonseca et al. 2018). Datu kopa iekļauj 20 kategorijas, kurās ir 20 mūzikas instrumenti un 24 cita veida skaņas avoti. Kopā ir pārstāvētas 44 kategorijas atbilstoši The AudioSet ontoloģijai (skaņu kategoriju koks, kas tiek piedāvāts ar Google pētniekiem)⁴⁴, taču datu kopa nav sabalansēta:

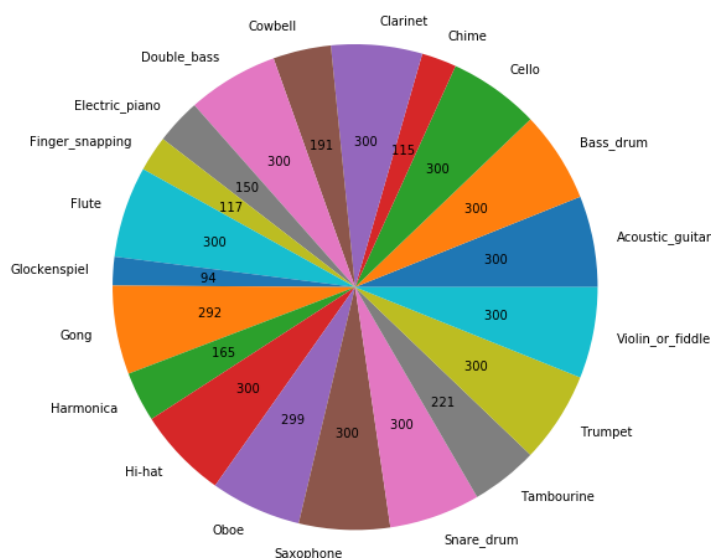
⁴⁰ Oficiālā mājaslapa: <https://www.tensorflow.org/about>

⁴¹ Google pētījums par mākslīgo intelektu: <https://research.google/teams/brain/>

⁴² Augsta līmeņa API darbam ar Tensorflow: <https://keras.io/>

⁴³ Oficiālā mājaslapa: <https://freesound.org/help/about/>

⁴⁴ Oficiālā mājaslapa: <https://research.google.com/audioset/>



Attēls 2.1. Absolūts fragmentu skaits datu kopā FSD Kaggle 2018, 20 atlasītas klases (tikai mūzikas instrumenti).

Kopā visās 44 kategorijās ir 9473 faili, un to vidējais garums ir 5.89 sekundes (minimālais – 0.32 sekundes, maksimālais - 30 sekundes). Modeļu testēšanas datu kopa satur 1600 audio failus, kas reprezentē tās pašas kategorijas, kā apmācību kopā. Tomēr faili ir atšķirīgi, un kopa tiek pielietota, lai nodrošinātu to, modelis nav “pārmācījies” (apskatīsim šo parādību vēlāk). Turpmāk darbā visas metrikas tiek skaitītas tikai priekš testēšanas datu kopas failiem.

Izstrādājot un atklūdojot modeli, ne vienmēr ir ērti strādāt ar pilnu datu kopu. Tāpēc uz tās pamata tika izveidotas samazinātas kopas:

Datu kopas/apakškopas darba nosaukums	Iekļauto kategoriju daudzums	Apmācību failu daudzums	Testēšanas failu daudzums
FSD Kaggle	41	9474	1601
“Music instruments 10”	10	3000	569
“Environment sounds 10”	10	2177	317
“Music instruments mini”	10	300	100
“Four class vs other”	5	1610	280

Tabula 2.1. Datu kopas darba modifikācijas.

2.1.3. Datu nolasišana un priekšapstrāde

Datu kopa ir jāsagatavo darbam ar PyTorch un Keras/Tensorflow bibliotēku moduļiem. Vēlams, lai modulis būtu spējīgs pēc iespējas ātrāk iegūt fragmentus no audio fragmentu cepstrogrammām⁴⁵, nosakot, kurai grupai tas pieder.

Tiek uzbūvēta speciāla datu ielādes klase, kas sagatavos datus nodošanai uz mākslīgo neironu tīkla ieejām. Pati cepstrogrammu būvēšana notiek ar `python_speech_features`⁴⁶ bibliotēkas palīdzību. Taču šeit var saskarties ar problēmu, kad izlasē tiek daļa no faila, kurā nav satura (tas var nejauši trāpīt uz pauzi vai klusumu). Vēl jāņem vērā, ka pamatinformācija par skaņas avota tembru atrodas frekvences apgabalā no 150 līdz 8000 Hz. Lai atvieglotu skaitļošanas slodzi mūsu sistēmai, pārveidosim datu kopu tā, lai diskretizācijas frekvence būtu 16000 Hz (oriģināli datu kopā ir 44100 Hz), atlasot saturu virs 8 kHz. Lai izvairītos no klusuma failos, pielietosim arī transformāciju. Pārrakstot jaunus failus, ignorēsim signāla daļas, kurās signāla RMS⁴⁷ krīt zem -80 dBFS⁴⁸. Pārsintezētus failus ierakstīsim atsevišķā direktorijā gan apmācību, gan testēšanas datu kopas daļām. Pārveidoto datu kopu es saglabāju atsevišķā direktorijā zem sākotnējiem nosaukumiem.

Datu ielādes klase būs spējīga automātiski noskenēt apstrādātās apmācību un testēšanas datu kopas, sadalīt apmācību paketes un sniegt sistēmai pareizu atbildi “karsti-iekodēta” vektora formā⁴⁹.

Darbā tiek izmantota MFCC (cepstrogrammu) datu reprezentēšanas metode. Līdzīgi kā klasiskās spektrogrammas, kas attēlo FFT⁵⁰ algoritma darbības rezultātu kā vērtību matricu, cepstrogrammas kolonnas ir Mel frekvences cepstrālo koeficientu vektori, katrs no tiem atbilst laika logam no oriģinālā audio faila. Tāda matrica attēlo audio ierakstu bides formātā ar kanālu daudzumu 1 (melnbalts), un tāds formāts der darbam ar datorredzes modeļiem.

⁴⁵ Cepstrogrammas un Mel frekvences cepstrālie koeficienti ir aprakstīti 3.1.1 sadaļā.

⁴⁶ Dokumentācija: <https://python-speech-features.readthedocs.io/en/latest/>

⁴⁷ Vidējais kvadrātiskais: <https://www.mathworks.com/help/signal/ref/rms.html>

⁴⁸ Decibelu pilnā skala: <http://www.sengpielaudio.com/calculator-db-volt.htm>

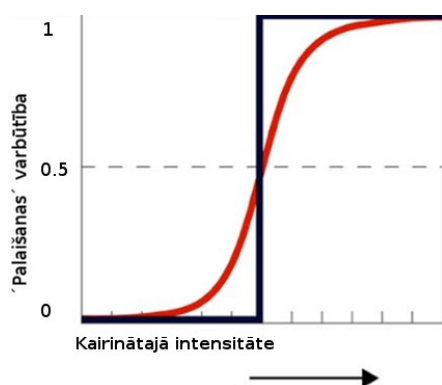
⁴⁹ One-hot encoded vector, angl <https://medium.com/bryanyang0528/deep-learning-numpy-%E5%AF%A6%E4%BD%9C-one-hot-encoding-softmax-d4d641f48258>

⁵⁰ Ātrā Furjē transformācija. Fast Fourier Transform, angl. <https://www.dspguide.com/ch12/2.htm>

2.1.4. Aktivācijas funkcija

Mākslīgos neironu tīklos aktivācijas funkcija definē konkrētā neirona izeju, kad tās arguments ir visu ienākošo ieeju un to atbilstošo svaru reizinājumu summa. Pirms tiks skaidrots, kāpēc aktivācijas funkcija vispār tiek izmantota, apskatīsim sigmoīda funkciju⁵¹:

$$S(x) = \frac{1}{1 + e^{-x}} \quad (2.1)$$



Attēls 2.2. Sigmoīda funkcija kā modelis, raksturojošs indivīda spējas atpazīt kairinātāju pēc G. Fechner teorijas. Attēlā funkcija (sarkans) ir salīdzināta ar vienības soļa funkciju (melns).

Šādu funkciju pielietošana bieži tiek saistīta ar iedvesmošanos no psihofizioloģijas, kad Gustav Fechner (1860)⁵² formulēja psihofizisko testu metodoloģijas. To pamatā ir ideja, ka ir iespējams aprakstīt dzīvas būtnes spēju atpazīt kairinātāju, izmantojot varbūtības blīvuma funkciju. Šāda funkcija apraksta varbūtību, ka subjekts uztver kairinātāju (piemēram, skaņas toni pie noteiktas frekvences un amplitūdas) testā pie noteiktas intensitātes.

Un agrīnie eksperimenti parādīja rezultātus, kas tiešām atgādināja sigmoīda funkciju⁵³, kas kļuva par pamatu vienkāršotajiem modeļiem, kas apraksta cilvēka psihofizioloģiju.

Līdzīgu principu izmantoja neirozinātnē⁵⁴, kad ar matemātiskiem modeļiem mēģina aprakstīt neirona palaišanas nosacījumus. Vienkāršotos modeļos neirona “palaišanas” varbūtība pieaug nelineāri, atbilstoši sigmoīda funkcijai atkarībā no kairinātāja intensitātes (fiziskās iedarbības, tai skaitā cita neirona izejas signāla).

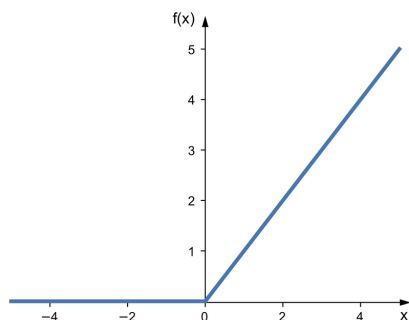
⁵¹ <https://www.sciencedirect.com/topics/computer-science/sigmoid-function>

⁵² Psihofizioloģijas zinātnes vēsture: <http://starasp.uit.yorku.ca/psycho/en/postscript.asp>

⁵³ Sigmoīda funkcija psihofizioloģijā
<https://www.cns.nyu.edu/~david/courses/perception/lecturenotes/psychophysics/psychophysics.html>

⁵⁴ Neironi kā klasifikatori: <https://www.coursera.org/lecture/computational-neuroscience/8-1-neurons-as-classifiers-and-supervised-learning-ltmfe>

Gadījumā ar neironu tīkliem, it īpaši agrīnajos mēģinājumos tos realizēt, bieži tika pieņemts “saspiest” jebkādu ienākošo argumentu apgabālā no $-\infty$ līdz $+\infty$ un atgriezt to kā vērtību starp 0 un 1. Taču mūsdienās bieži tiek pielietota cita funkcija, kas saucās ReLU⁵⁵, kas visas vērtības mazākas par 0 pārveido par 0, bet lielākas par 0 atstāj bez izmaiņām.



Attēls 2.3. Rectified Linear Unit.

ReLU funkcija

To var interpretēt tā: ja ieejas arguments ir negatīvs, tad izejas vērtība ir vienāda ar 0. Jo lielāks būs pozitīvais arguments, jo lielāka būs atgrieztā vērtība.

Tās ir tikai dažas, visbiežāk pielietotās aktivācijas funkcijas. Tieši **ReLU būs pielietots darba eksperimentālajā daļā**, realizējot apmācības modeļus.

2.1.5. Kļūdas funkcija

Kļūdas funkcijas ļauj skaitliski novērtēt, cik prognozētā izeja ir tuvu vai tālu no īstās vērtības. Pati vienkāršākā kļūdas funkcija ir **vidējā kvadrātiskā kļūda**:

$$E(\theta) = \frac{1}{n} \sum_{i=1}^n (\hat{\theta}_i - \theta_i)^2 \quad (2.2)$$

Kādreiz tā arī ir pielietojama klasificēšanas uzdevumu gadījumos. Taču šajā pētījumā mēs izmantosim citu aktivācijas funkciju, kas saucas Kategoriskā Cross-Entropija (Qin, Kim, and Gedeon 2020). Tā apstrādā vektoru ar kategoriju varbūtībām un intuitīvi ir vairāk piemērota klasifikācijām, kad apmācībā piedalās fiksēta un nemainīga datu kopa ar zināmu kategoriju daudzumu. Lai Kategoriskā Cross-Entropija izmantošana vispār būtu iespējama, pirms tam neironu tīklā tiek izvietota Softmax Funkcija (2.1.6), kuru mēs detalizētāk aprakstīsim nākamajā sadaļā. Kategoriskās Cross-Entropijas īpašība ir tās pielietošana gadījumos ar ļoti lielu kategoriju daudzumu. Tās darbība ir balstītā uz varbūtību vektora salīdzināšanu ar “patieso

⁵⁵ ReLU funkcija: <https://medium.com/@danqing/a-practical-guide-to-relu-b83ca804f1f7>

atbilžu” vektoru, kas sastāv ar 0 un vienas vienīgas 1 vērtības (kas reprezentē pareizo atbildi, īsto klasi klasifikācijā). Tātad, pati funkcija ir izteikta sekojoša veidā:

$$D(S, L) = -\sum L_i \log(S_i) \quad (2.3)$$

Kur S ir Softmax funkcijas izejošais klasifikācijas varbūtības vektors, L ir “īsto” atbilžu vektors⁵⁶.

Nozīmīgākā modifikācija kļūdas funkcijai, kas tiek izmantota darba praktiskās daļas realizācijā, ir svaru aprēķins un pielietošana kļūdas funkcijā. Mūsu datu kopas datu daudzums dažādās klasēs nav sabalansēts. Runa ir par to, cik daudz ierakstīto laiku kopumā ir katrā no kategorijām.

Mēs aprēķināsim īpatsvaru katrai kategorijas datu kopai un uz tā pamatā izveidosim vektoru ar svāriem. Svāri tiek skalāri sareizināti ar atbilstošiem S_i vektora elementiem formulā 2.3 un tikai pēc tam tiek rēķināts $D(S, L)$.

2.1.6. Softmax funkcija

Softmax funkcija garantē, ka uz Kategoriskas Cross-Entropijas funkciju nekad netiks 0 vērtība, kas atgrieztu NaN⁵⁷ vērtību un sajauktu aprēķinus. Tā tiek noteikta ar formulu⁵⁸:

$$f_i(\vec{a}) = \frac{e^{a_i}}{\sum_{k=1}^K e^{a_k}} \quad (2.4)$$

Kur \vec{a} ir vektors, kas tiek saukts par logitu un ir lineārā modeļa darbības rezultāts (formula 1.1). Bieži literatūrā šai konstrukcijai piešķir nosaukumu Multinomiālā Loģistiskā Klasifikācija (El-Habil 2012). To mēs izmantosim skaņas avotu klasificēšanai.

⁵⁶ Softmax funkcija atgriež varbūtību sadalījuma vektoru, kura garums atbilst klašu daudzumam sistēmā.

⁵⁷ Not-a-number, angl.

⁵⁸ <https://www.machinecurve.com/index.php/2020/01/08/how-does-the-softmax-activation-function-work/#> (“How Does the Softmax Activation Function Work?” 2020)

2.1.7. Modeļu optimizēšanas metode

Klasiskais veids, lai atjaunotu mākslīgā neironu tīkla svarus ir Stohastiskās Gradianta Nolaishanās vai SGD (Bottou 2012). Katrs slāņa svars tiek atjaunots, pamatojoties uz tā ieguldījumu kļūdas funkcijā (formula 1.3). Tam ir virkne ar problēmām: neiespējamība atrast globālo minimumu, samazināta precizitāte ar fiksēto apmācību faktoru⁵⁹.

Pēdējos gados tiek plaši izmantots optimizēšanas paņēmiens, ko sauc par ADAM optimizētājs (Kingma and Ba 2017). Šajā laika posmā vairākas reizes bija parādīts, ka tas daudz labāk darbojas konvolūciju neironu tīklu gadījumos. Arī svaru atjaunošanas metode balstās uz gradienta un apmācības ātruma η . Bet vēl tiek izmantoti papildus parametri, kas ņem vērā to, kā notika apmācība iepriekšējās iterācijās. Detalizēti neapskatīsim pamatojumu un teoriju, kas stāv aiz tā⁶⁰, bet pieminēsim, ka autori kombinēja divus iepriekš plaši izmantotus algoritmus:

- AdaGrad⁶¹, kas tiek rēķināts, pamatojoties uz apmācību ātruma pēdējās iterācijās un paaugstina apmācību ciklu ātrdarbību
- RMSProp⁶², kas balstās uz gradienta izmaiņas ātruma

ADAM aprēķina eksponenciālo peldošo vidējo gradienta vērtību un gradienta kvadrātu. Divi parametri β_1 un β_2 kontrolē rimšanas ātrumu šīm peldošajām vidējām vērtībām. Sakontējās β_1 un β_2 vērtības ir tuvu 1 un tiek nobīdītas 0 virzienā apmācību gaitā. Nosaukumā ADAM ir iekodēts “Adaptive Moment”, kas novērtē un optimizē gradienta nolaishanas momentu. Šajā darbā tiek izmantota ADAM algoritma modifikācija, kuras nosaukums ir RADAM (Liu et al. 2020). Algoritma dizains un detalizēts ieskats būtu ārpus šī darba mēroga, tomēr tā efektivitāte salīdzinājumā ar citām metodēm un augošā popularitāte ir pamatojums tieši šī algoritma izvēlei modeļa izstrādē. Svarīgi pieminēt, ka RADAM ietekmē apmācības ātrumu η un testējot modeli, nav nepieciešamības pārbaudīt to vairākās η kombinācijās ar citiem sistēmas parametriem.

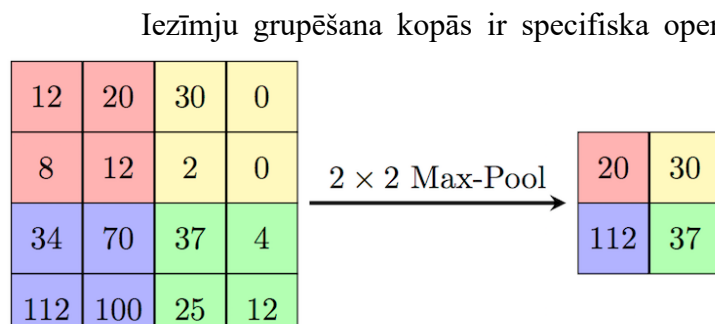
⁵⁹ “The ‘Terpret Problem’ and the Limits of SGD”, 2018: <https://dselsam.github.io/the-terpret-problem/>

⁶⁰ Par ADAM optimizāciju: <https://mlfromscratch.com/optimizers-explained/>

⁶¹ Adaptive Gradient Algorithm, angl. Adaptīva Gradianta Algoritms

⁶² Root Mean Square Propagation, angl. Vidējas kvadrātiskās kļūdas izplatīšana

2.1.8. Iezīmju grupēšana kopās



Attēls 2.4. Iezīmju grupēšana kopās pēc maksimālas vērtības (aizgūts no <https://computersciencewiki.org/>, 2017)

izmērus būvējot modeli. Arī tā izmanto kodola funkciju, tikai, atšķirībā no konvolūcijas, šī funkcija atgriež maksimālo vai vidējo aritmētisko no ieejas vērtībām⁶³. Operācijai ir nepieciešama ienākošās un izejošās matricas izmēru

2.1.9. Normalizēšana

2.1.10. Modeļu arhitektūra: ConvNet

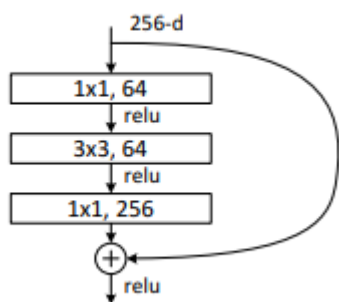
⁶³ Max Pooling, Average Pooling: <https://www.machinecurve.com/index.php/2020/01/30/what-are-max-pooling-average-pooling-global-max-pooling-and-global-average-pooling/>

Uz ieejas padotas trenēšanas paraugu matricas (1 kanāls). Slāņos pakāpeniski pieaug kanālu daudzums un samazinās matricu izmērs (sk. 1. pielikums).

Pēc konvolūcijas slāņiem seko adaptīva iezīmju grupēšana. Tā ir iezīmju grupēšana kopās, kas automātiski aprēķina ieeju skaitu pie zināma izeju daudzuma. Tas dod mums iespēju pārveidot modeli par neatkarīgu no datu fragmenta matricu izmēriem.

Rezultāts tiek padots uz lineāro slāni, Softmax funkciju un tālāk tiek salīdzināts ar pareizo atbildi, izmantojot Kategorisko Cross-Entropijas funkciju.

2.1.1. Modeļu arhitektūra: ResNet18



Attēls 2.5. “Bottleneck” bloks
(Aizgūts pie Zhang, et al, 2015)

paņēmiens (Dropout) ar izslēgšanas varbūtību 0.1. Modelis arī tiek apmācīts izmantojot Kategorisko Cross-Entropijas kļūdas funkciju. **ResNet18** un tā padziļinātājam versija **ResNet50** (2. pielikums) ir raksturīgi ar modificēti ResNet bloki, kuras iekšējos slāņos atšķiras ieeju un izeju kanālu daudzums (nosaukts par “bottleneck”). Struktūra vienkāršoti ir parādīta 2. pielikumā.

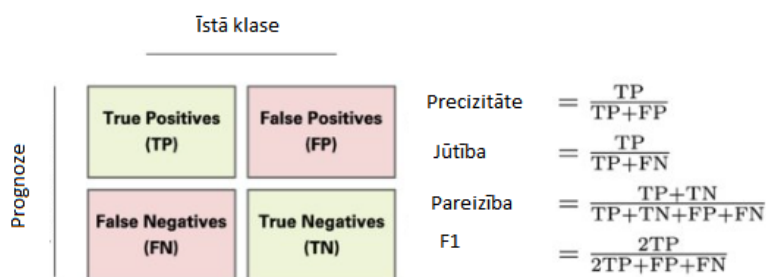
2.1.2. Rezultātu analīzes metodoloģija (metrikas)

Metrika ir rādītājs, kas ļauj skaitliski novērtēt modeļa apmācības un darbības efektivitāti. Parasti tiek analizētas vairākas metrikas, lai saņemtu pilnīgāku priekšstatu par modeli. Pirmā metrika tika apspriesta iepriekšējās nodaļās, tā ir kļūdas funkcijas vērtība. Tās

⁶⁴ “Bottleneck” bloks (Attēls 2.5)

primārais pielietojums ir paša modeļa apmācība, un skaitlis pats pa sevi cilvēkam neko neizsaka, jo ir atkarīgs no modeļa uzbūves un datu kopas. Tomēr, skatoties uz kļūdas funkcijas vērtībām, pētnieks var secināt, vai apmācības vispār notiek (modelis konverģē). Tas notiek, ja kļūdas funkcijas (mūsu gadījumā Kategoriskā Cross-Entropija) samazinās.

Divas svarīgas metrikas ir precizitāte un jutība. Precizitāte raksturo modeļa spēju atkārtoti klasificēt konkrēto paraugu, bet neraksturo klasifikācijas rezultāta sakritumu ar referenci – pareizo atbildi. Tā ir pareizo prognožu daudzums pret visām pozitīvajām prognozēm. Taču pareizība vienmēr ir apstiprināto paraugu un noraidīto paraugu summa pret visiem apstiprinājumiem un noraidījumiem (pareiziem un nē). Mūs interesē arī skats no otras puses: cik daudzi relevanti elementi tiek izvēlēti attiecībā uz visiem relevantiem elementiem? Tā ir modeļa jutība. Abas metrikas tiek paskaidrotas attēlos 1 un 2. Kombinējot precizitāti un jutību vienā parametrā, tiek rēķināta F1 metrika, kas ir harmoniskais vidējais starp precizitāti un jutību (Powers, 2011):

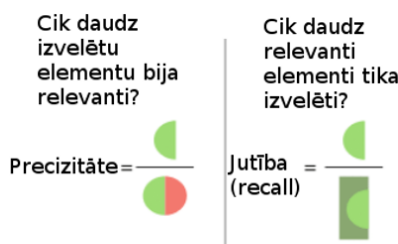
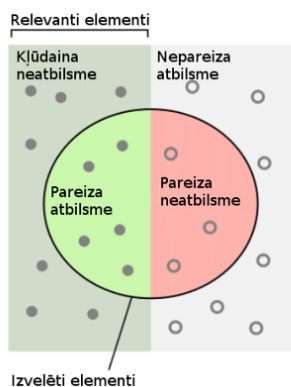


Attēls 2.6. Klasifikācijas metrikas. TP – pareizi apstiprināts, FP – nepareizi apstiprināts, TN – pareizi noraidīts, FN – nepareizi noraidīts (aizgūts pie Bitttrrich, Kaden et al, 2019)

F1 metrika tiek aprēķināta pēc formulas:

$$F_1 = \frac{2}{\frac{1}{J} + \frac{1}{P}} \quad (2.5)$$

Kur P ir precizitāte, J ir jutība.



Attēls 2.7. Precizitāte, jutība un F1 metrikas

Parasti, vērtējot klasifikācijas uzdevumu darbību, tiek izmantota apjukuma matrica. Tā ir kvadrātiska matrica, kas reprezentē modeļa prognozēto klašu apzīmējumu kopu attēlu uz īsto klašu apzīmējumu kopas (parasti, validētu ar cilvēka piesaisti). Matricas 0 ass atbilst īstam klašu apzīmējumiem, 1 ass atbilst prognozētiem klašu apzīmējumiem. Galvenā diagonāle attēlo visus gadījumus, kad prognoze sakrīt ar īsto vērtību. Vērtības zem galvenās diagonāles atbilst gadījumiem, kad modelis nepareizi prognozēja, ka paraugs pieder citai klasei. Vērtības virs galvenās diagonāles atbilst gadījumiem, kad modelis kļūdaini noraidīja īsto klasi savā prognozē.

Ir laba prakse – veikt gala pārbaudi uz validācijas datu kopas. Ir plānots to izdarīt, kad pēc eksperimentiem tiks atlasīts visefektīvākais modelis un tā parametri. Validācijas datu kopa sastāv no failiem, kas nepieder trenēšanas un testēšanas kopām. Tas ļautu pārbaudīt, cik veiksmīgi sistēma klasificē dabā ierakstītus piemērus. Uz tās pamata mēs veiksime prognozes un rezultātu sadaļā aprēķināsim augstāk minētās metrikas.

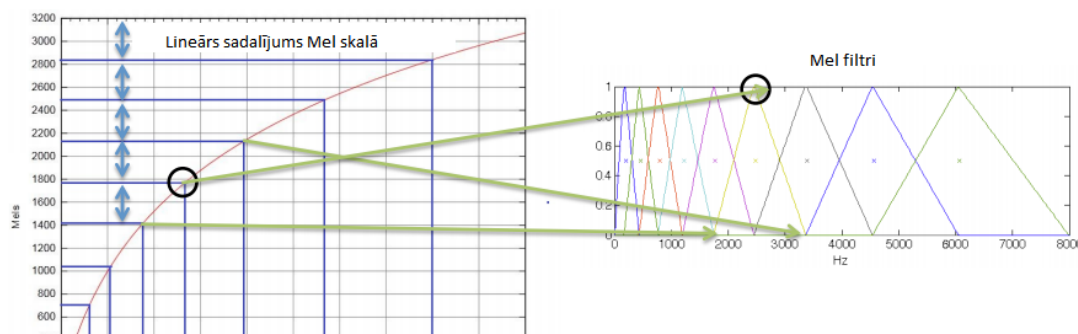
3. EKSPERIMENTĀLĀ DAĻA⁶⁵

3.1.1. Audio ierakstu priekšapstrādē un analīze

Datu kopas sagatavošana ir viens no būtiskākajiem aspektiem, kas ietekmē modeļa spēju no tā mācīties. No vienas puses, mūsu pētījuma uzdevums ir nekas cits kā attēlu klasifikācija. Tomēr pazīmes, kas labi darbojas attēlu atpazīšanā ar uzzīmētiem objektiem vai simboliem (līniju forma, pikseļu intensitātes blīvums u.t.t.) nevar būt tik vienkārši pārnestas uz gadījumu ar audio signāla spektrogrammu. Citiem vārdiem runājot, tur darbojas citas likumsakarības. Klausoties atskaņotus audio fragmentus, cilvēkam nav nekādu problēmu atpazīt mūzikas instrumentu. Taču grūtības veidojas brīdī, kad cilvēks skatās uz spektrogrammu. Atpazīšana, izmantojot redzes uztveres kanālu, arī cilvēkam prasa pieredzi un trenēšanos, un nav tik uzticama. Tomēr modelis var iemācīties pazīmes, kas nav acīmredzamas cilvēkam.

Lielākā daļa no paņēmieniem, kas tiks pielietota šī pētījuma ietvaros, ir mēģinājums atdarināt metodes, kas ir visbiežāk sastopamas audio failu klasificēšanas pētījumos (Segal, 2016) (Gururani, Sharma, and Lerch 2019).

Apskatīsim audio datu reprezentēšanas metodes. Kā jau bija minēts, darbā tiek izmantotas Mel frekvences cepstrālo koeficientu spektrogrammas – cepstrogrammas. Tāda tipa reprezentēšana ņem vērā specifiskās pazīmes, kas ievēro cilvēka dzirdes sistēmas bioloģiskās īpašības. Tas modelē dzirdes sistēmu kā vairāku pārkļājušos trīsstūrveidīgo filtru kopu, kuru platums samazinās ar frekvences pieaugumu. Tas veido cilvēka spēju viņam specifiskā veidā



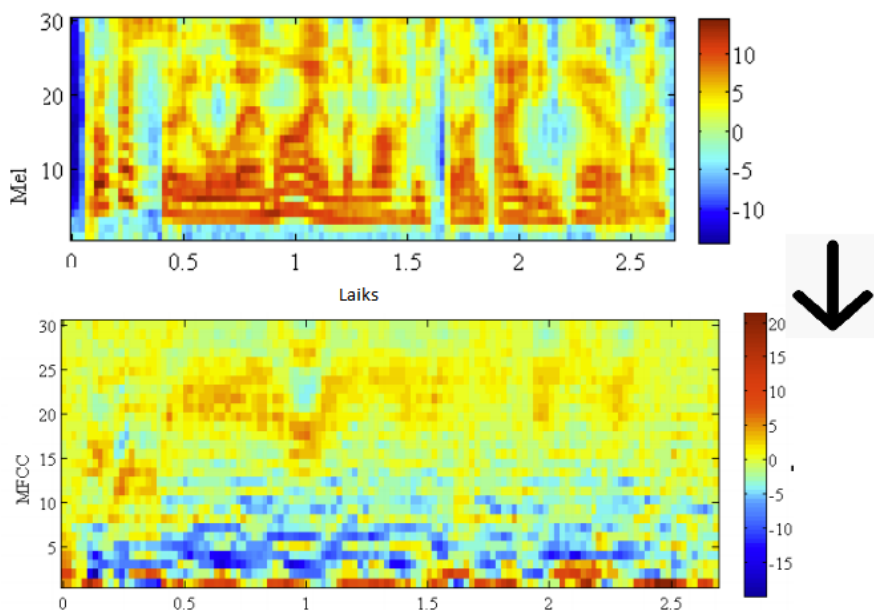
Attēls 3.1. Mel frekvences skalas saistība ar lineāro frekvences skalu un Mel filtriem (aizgūts no Katariina Mahkonen. TUT kurss "PuheenkäsiMelyn menetelmät", 2013)

⁶⁵ Projekta kods ir pieejams repozitorijā: https://github.com/DmitryOdinoky/audio_files_classification_project

atpazīt izmaiņas zemā frekvences diapazonā (ņemot vērā, ka cilvēkam dzirdamais diapazons atrodas no 20 Hz līdz 20 000 Hz).

Šī darba ietvaros tiek izmantota `python_speech_features` bibliotēkas iebūvētā funkcija `mfcc()`⁶⁶, kas atgriež divu dimensiju masīvu ar cepstrogrammu. Librosa arī tiek izmantota, lai realizētu failu ielādi un transformācijas failu priekšapstrādes stadijā.

Veicot FFT katrā Mel frekvences filtrā, vērtējot tā enerģijas izmaiņas laikā un pielietojot



specifisko matemātisko aparātu filtru savstarpējai dekorrelācijai⁶⁷ (detaļas paliek ārpus šī darba mēroga), tiek aprēķināti Mel cepstrālie koeficienti. Koeficienti tiek aprēķināti katram laika intervālam, un izejošo vektoru kopa tiek apvienota matricā, kas tiek attēlota spektrogrammas veidā.

Attēls 3.2. Mel frekvences spektrogramma un tam atbilstošs Mel Frekvences Koeficientu cepstrogramma (MFCC spektrogramma).

Tieši uz šīs metodes es koncentrēšos šajā darbā, jo tās minētās pazīmes ir tik plaši apspriestas pēdējo gadu publikācijās

par tembru un balss atpazīšanu (Eringis and Tamulevičius, 2015), ka es pieņēmu lēmumu pagaidām uzticēties plaši izplatītajai praksei.

Pirms cepstrogrammu būvēšanas no audio failiem tiek izņemti laika fragmenti, kuros signāla jauda krīt zemāk par 80 dB no maksimālas vērtības (pieņemot, ka tie nesatur tembrālo informāciju, bet tikai klusumu ar trokšņiem). Visu fragmentu diskretizācijas frekvence tiek

⁶⁶ Dokumentācija: <https://python-speech-features.readthedocs.io/en/latest/>

⁶⁷ Mel frekvence un cepstrālie koeficienti: <http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/>

pārveidota no oriģinālās 44100 Hz uz 16000 Hz, un maksimālā frekvence ir nofiltrēta līdz 8000 Hz, atbilstoši.

Lai modelis būtu darbspējīgs, ir jānodrošina pietiekami liels datu kopas apjoms un vienādi trenēšanas paraugu izmēri. Datu priekšapstrādes solī pēc spektrogrammu uzbūves, tās sākumā tiek atkārtotas 2 reizes laika asī un pēc tam sadalītas mazākos vienāda garuma fragmentos - 40 soļu garumā ar pārklājumu 20 soļi⁶⁸. Katram fragmentam tiek piešķirts apzīmējums, kas atbilst pareizai skaņas avota kategorijai. Datu ieladēšanai tiek izmantota PyTorch speciālā klase, kas sajauc paraugus nejaušā secībā un veido paketes (katram paraugam paketes tensorā atbilst pareizas klases apzīmējums identifikācijas tensorā. Abi tiek padoti uz modeļa ieejam apmācības gaitā). Pilns apmācību cikls beidzas, kad tiek izmantotas visas paketes paraugu kopā. Pēc tam notiek testēšana, jau ar citu kopu.

Datu sagatavošanas procedūra Tensorflow realizācijai būtiski atšķiras. Apmācību gaitā speciālā funkcija nejauši izvēlās kategoriju, audio failu no tās un soli 0.1 sekundes garā fragmentā no tā. Kopīgs fragmentu daudzums tiek rēķināts pēc formulas:

$$n = 2 * \frac{\sum_{i=0}^N t_i}{s} \quad (3.1)$$

Kur n ir kopīgs fragmentu daudzums, N ir kopīgs failu daudzums datu kopā, t ir i-taja faila garums sekundēs un s ir solis, nolases daudzums⁶⁹. Tāds pats princips tiek izmantots testēšanai apmācību gaitā. Validācijai tiek paņemts viens vienīgs nejauši izvēlēts fragments no validācijas kopas visiem paraugiem pēc kārtas.

3.1.2. Modeļu apmācība

Visi eksperimenti tika veikti izmantojot portatīvo datoru Lenovo Y700. Processors i7 (6th gen), 8GB RAM. Datorā ir uzinstalēta NVIDIA GeForce GTX 960M 4GB GDDR5 grafiskā karte un tiek izmantota paralēlas skaitļošanas platforma CUDA⁷⁰. PyTorch un Keras

⁶⁸ Datu sagatavošanas koda daļa PyTorch realizācijai:

https://github.com/DmitryOdnokov/ConvNets_audioClassification/blob/master/data_class.py

⁶⁹ Datu sagatavošanas koda daļa Tensorflow realizācijai:

https://github.com/DmitryOdnokov/TF_based_models/blob/master/model.py

⁷⁰ Compute Unified Device Architecture, angl: <https://developer.nvidia.com/cuda-downloads>

dod iespēju relatīvi vienkārši izmantot CUDA. Tas kļūst īpaši aktuāli, kad runa ir par Torchvision piedāvāto dziļo modeļu lietošanu, kas ir ļoti resursu ietilpīgi. Šādu modeļu lietošana kļūst par vairāku stundu (un pat dienu) jautājumu uz vienkārša personīgā datora (atkarībā no datu kopas izmēriem), pat izmantojot tā grafisko karti.

3.1.3. Validācija

Svarīgs nosacījums šī darba mērķa sasniegšanai ir modeļa spēja pareizi klasificēt audio failus, ar kuriem tas pirms tam nav saskaries. Lai automatizētu testēšanas procedūru un varētu novērtēt, cik precīzi modelis veic klasifikāciju pēc katras apmācību epohas, sākotnēji tika pievienots validācijas fāzes cikls. Tā gaitā dati tiek paņemti no testēšanas kopas un sagatavoti līdzīgā veidā, kā apmācību stadijā. Vienīgā atšķirība ir tāda, ka dati neietekmē pašu neironu tīklu (nenotiek kļūdas atpakaļ-izplatīšanās) un tiek skaitīts, cik precīzi modelis spēj veikt prognozes ar failiem, kas pirms tam nepiedalījās apmācībā.

Apmācīts modelis tiek pārbaudīts ar testēšanas datu kopu, kas sastāv no failiem, kas nepiedalījās validācijas un apmācības stadijās. Kad darba noslēgumā izvēlētais modelis tiks apmācīts uz lielas datu kopas, es sadalīšu apmācības datu kopu un izmantošu 0.3 no visiem apmācības datiem validācijai. Bet validācijas kopa tiks izmantota testēšanai. Tas ļaus spriest, cik veiksmīgi modelis klasificētu reālus ierakstus, kas, piemēram, varētu būt ierakstītas ārējā vidē.

Šī darba ietvaros es visādos veidos sadalīšu to apakškopās (Tabula 2.1). Tas ir saistīts ar laika ierobežojumiem (samazināts datu apjoms samazina apmācības laiku. Bet, protams, samazina arī apmācības efektivitāti) un pētnieciskiem uzdevumiem (piemēram pārbaudei, kā klašu daudzums un datu apjoms ietekmē apmācības radītājus).

3.1.4. Rezultāti un vērtēšana

Ir jāatzīst, ka esošais mēģinājums ir mans pirmais solis gan audio ierakstu klasifikācijas jomā, gan dziļas mašīnāpmācības jomā. Mēģinājums realizēt efektīvu modeli, izmantojot PyTorch bibliotēkas, nebija tik veiksmīgs, tomēr izdevās panākt, ka modelis tiešām mācās, kas jau ļauj salīdzināt gan dažādus modeļus, gan datu priekšapstrādes un audio ierakstu reprezentācijas metodes.

Tomēr, lai sasniegtu rezultātu, es uztaisīju otru realizāciju Tensorflow platformā, izmantojot Keras API. Ar to izdevās sasniegt labākus rādītājus un validēt rezultātus. Bet divas

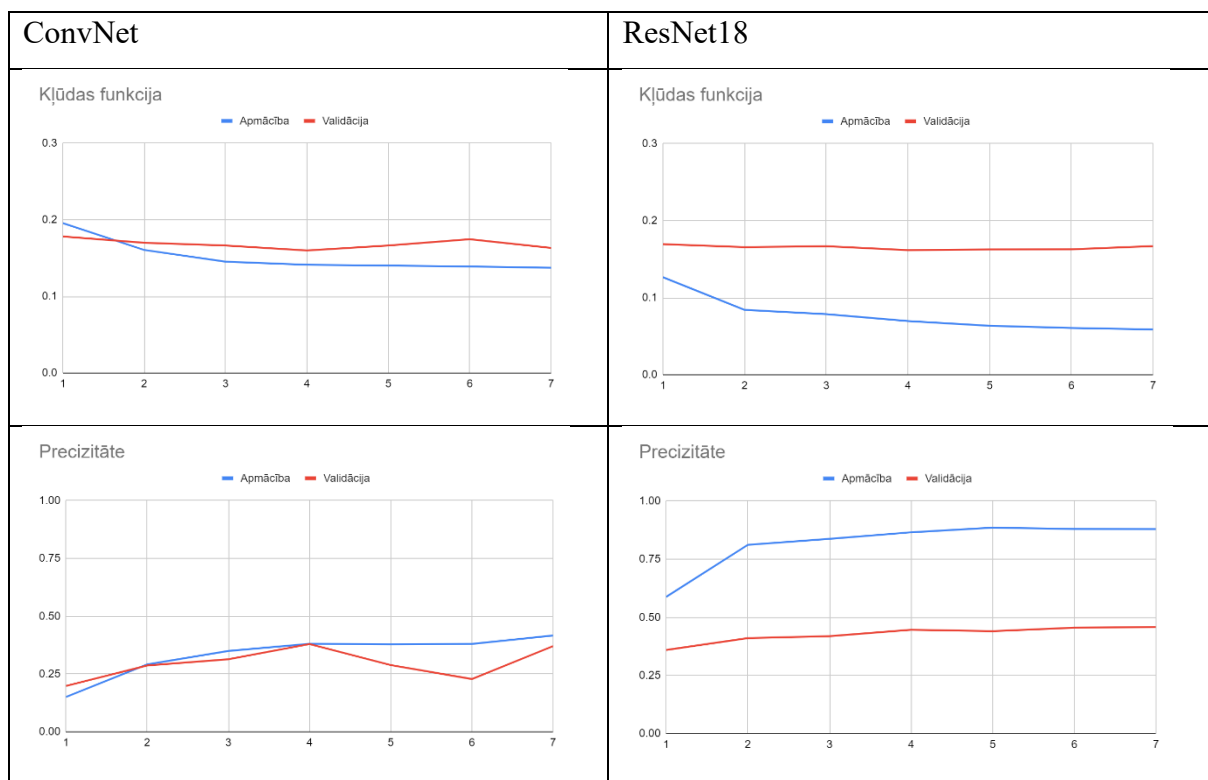
realizācijas nevar nosaukt par ekvivalentām (un pat salīdzināt savā starpā), jo realizācijas atšķiras: nav identiska modeļu arhitektūra, datu ielādes un priekšapstrādes metodes.

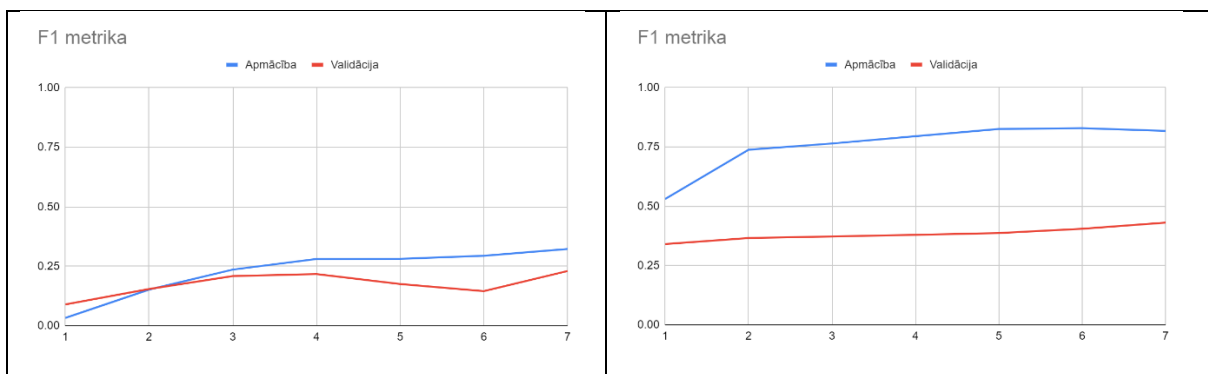
Eksperiments tika veikts, lai sasniegtu bakalaura darba mērķi: **iegūt efektīvāku dziļās mašīnāpmācības metodi mūzikas instrumentu atpazīšanai no audio ieraksta**. Efektivitāti es mērīšu izmantojot **precizitāti, pareizību** un **F₁** metriku, kas kopumā vērtē modeļu spēju pareizi klasificēt paraugu no ierobežotas klašu kopas.

3.1.5. Uzbūvēto arhitektūru salīdzinājums

Pirmkārt, tiek salīdzinātas divas modeļu versijas: ConvNet un vienkāršots ResNet18. Pārbaude tika veikta uz “Music Instruments Mini” datu kopas (Tabula 2.1). Tiek salīdzinātas: kļūdas funkcijas vērtības, precizitāte un F₁ metrikas pēc 7 pilniem apmācību cikliem - epohām. Datu reprezentēšanas veids ir MFCC cepstrogrammas, jo tās bija visbiežāk sastopamas atrastajās publikācijās par līdzīgu uzdevumu risināšanu (Eringis and Tamulevičius, 2015). Cepstrālo koeficientu daudzums ir 42, audio faili no datu kopas tiek sagriezti fragmentos, kuru garums ir 40 soli un pārklājuma logs ir 20 soli.

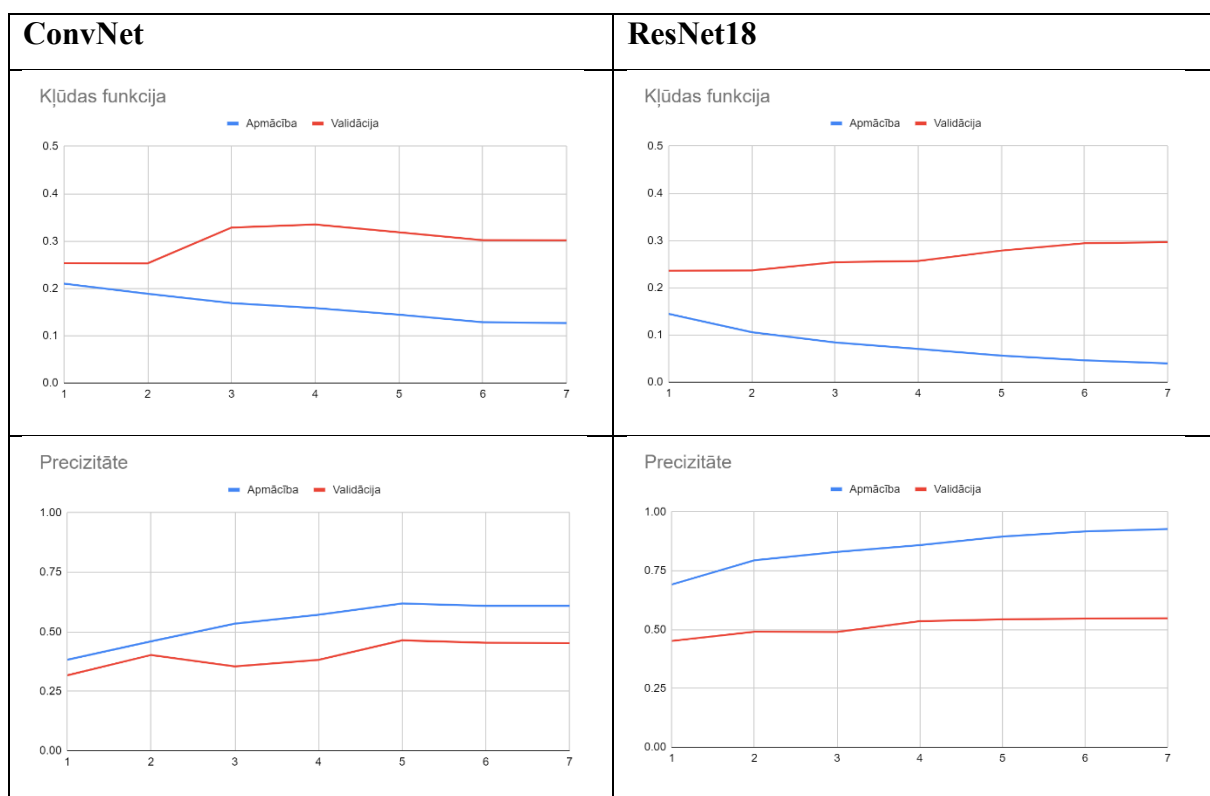
MFCC spektrogrammas ir veidotas no 13 koeficientiem uz 26 filtriem.

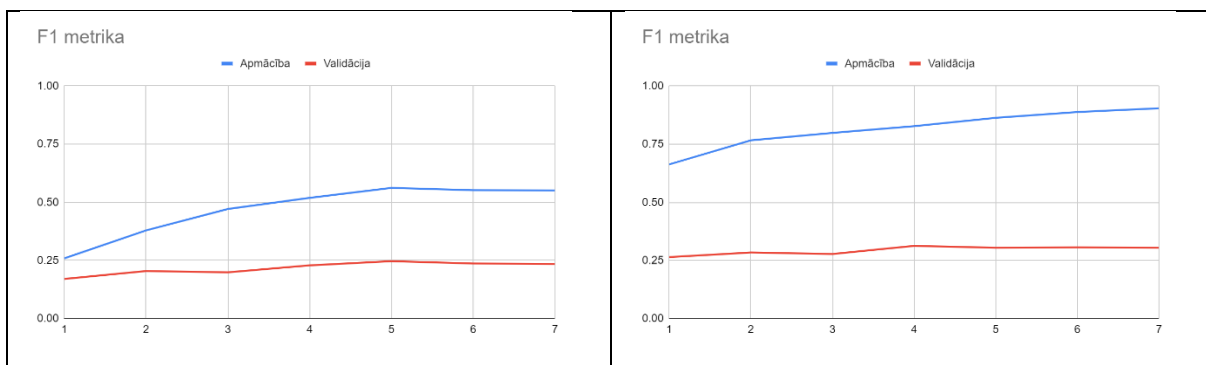




Tabula 3-1. Pirmais eksperimenta posms. Divu modeļu salīdzinājums uz datu kopas “Music Instruments Mini”.

Tālāk modeļi tika pārbaudīti uz lielākas datu kopas, kurā ietvertas tikai 5 klases (lielas bungas, hi-hat šķīvis, flauta, trompete un “cits”. “Four Class vs Other” tabulā 2.1). Klase “cits” ir nejaušs sajaukums no visām citām pilnās datu kopas skaņām.



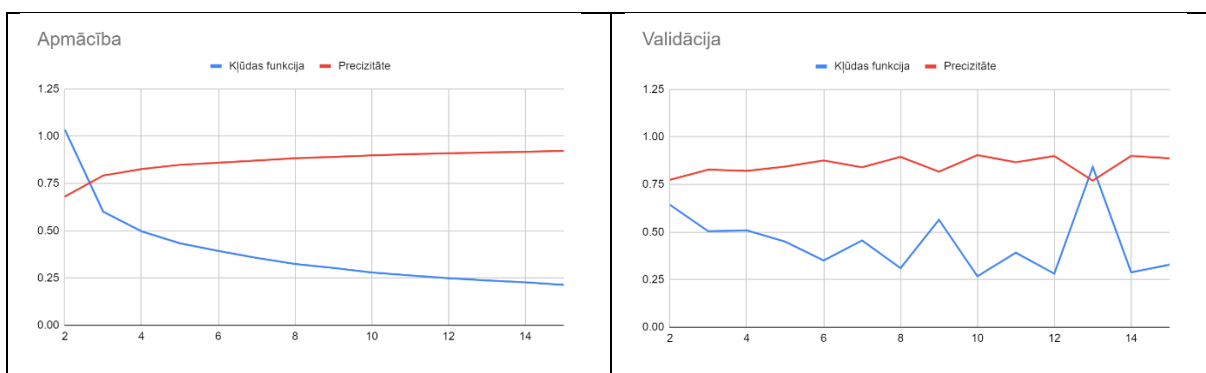


Tabula 3-2. Otrais eksperimenta posms. Divu modeļu salīdzinājums uz datu kopas “Four Class vs Other”.

Rezultāti apstiprina teorētiskajā daļā aprakstīto vielu. ResNet arhitektūra abu samazināto datu kopu gadījumos parādīja labākus rezultātus. Tomēr rezultāts kopumā nav labs, jo visos gadījumos pēc pirmās epohas sāk augt apmācību kļūdas funkcija, kas var būt saistīts ar kļūdām, pieļautām datu sagatavošanas\priekšapstrādes kodā.

Diemžēl, ar PyTorch man neizdevās panākt rezultātu, kas būtu lietojams reālajā dzīvē. Tam varētu būt vairāki iemesli, apskatīsim tos secinājumos. Bet, kā arī bija minēts, es izdarīju mēģinājumu apmācīt modeli uz Tensorflow platformas pamata, izmantojot Keras API. Iekš Keras es neatradu vienkāršas ResNet18 realizācijas, bet atradu ResNet50, kas ir padziļinātā versija (arhitektūras shematiskais skaidrojums ir atrodams 2. pielikumā).

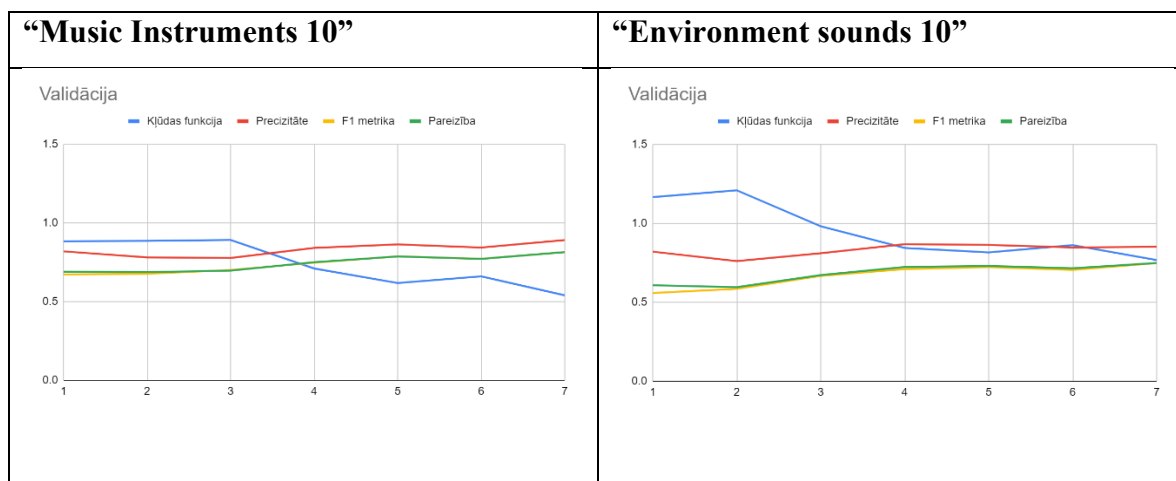
Pirms palaist apmācības procedūru uz lielas datu kopas, sākumā es notestēju sistēmu apmācot vienkāršo ConvNet modeli (jo tas apmācas ātrāk – mazāk laika vienai pilnai epohai), lai pārbaudītu, vai tas vispār darbojas.



Attēls 3.3. ConvNet uz Tensorflow, “Music Instruments Mini” datu kopa

Tā kā ResNet arhitektūra dod labākus radītājus, sākotnējais plāns bija apmācīt ResNet50 modeli uz pilnās 41 klašu datu kopas. 30% no visiem failiem atlasīti kā testēšanas apakškopa.

Atlikušos failus es plānoju izmantot vēlāk validācijai. Bet ar vienu GPU 15 epohu izpilde aizņēmtu 30 stundas. Tāpēc es izvēlos sadalīt datu kopu divās daļās: 10 klases ar mūzikas instrumentu skaņām un 10 klases ar arējas vides skaņām, kas nav mūzikas instrumenti. Ideja bija pārbaudīt, cik veiksmīgi vienkāršā ConvNet arhitektūra apmācīsies un būs spējīgā klasificēt šīs divas apakškopas, ja datu reprezentācijai tiek izmantoti Mel frekvences cepstrālie koeficienti (datu priekšapstrādes un datu ielādēs metodes ir aprakstītas sadaļā 3.1.1).



Attēls 3.4. ConvNet uz Tensorflow, apmācīts uz divām desmit klašu datu kopām.

Nākamajā solī tiek pārbaudīts, cik precīzi modelis klasificē nezināmus failus no testēšanas kopas (Tabula 4.3).

“Music instruments 10”				“Environment sounds 10”			
Klase	Precizitāte	Jūtība	F1	Klase	Precizitāte	Jūtība	F1
Acoustic_guitar	0.81	0.87	0.84	Applause	0.82	1	0.9
Bass_drum	1	0.89	0.94	Bark	0.54	0.89	0.68
Cello	0.92	0.85	0.88	Computer_keyboard	0.43	0.58	0.49
Clarinet	0.83	0.98	0.9	Fireworks	0.53	0.59	0.56
Double_bass	0.92	0.9	0.91	Knock	0.76	0.49	0.59
Flute	0.72	0.95	0.82	Meow	0.67	0.62	0.64
Hi-hat	1	0.97	0.99	Scissors	0.44	0.44	0.44
Saxophone	0.98	0.87	0.92	Squeak	0.25	0.21	0.23
Snare_drum	1	0.53	0.69	Telephone	0.93	0.52	0.67
Violin_or_fiddle	0.9	0.95	0.92	Writing	0.48	0.55	0.52
Pareizība			0.89	Pareizība			0.59
Vidējais	0.91	0.88	0.88	Vidējais	0.58	0.59	0.57
Svērtais vidējais	0.91	0.89	0.89	Svērtais vidējais	0.62	0.59	0.58

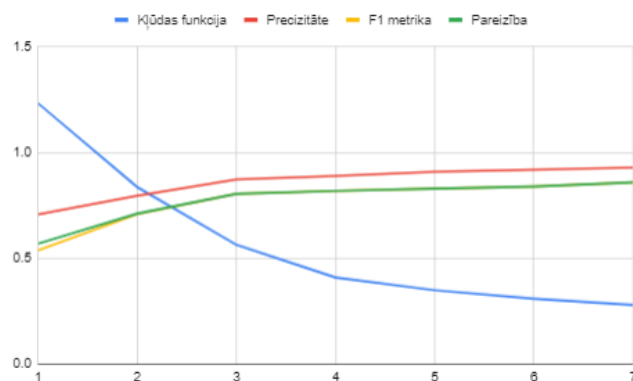
Tabula 3-3. Trešais eksperimenta posms. ConvNet apmācīts uz divām datu kopām, 7 epohas (atbilstošās apjukuma matricas var apskatīt 3. pielikumā). Tabulā ir testēšanas rezultāti uz datu kopas, kas nepiedalījās apmācībā. Klašu nosaukumi ir angļu valodā, atbilst FSD datu kopas formātam (apjukuma matricas ir 3. pielikumā).

Skatoties uz metrikām es varu izdarīt secinājumu, ka modelis rezultatīvāk apmācas uz mūzikas instrumentiem. Taču jāatceras, ka arī failu daudzums (un kopējs ierakstu garums) gadījumā ar mūzikas instrumentiem bija lielāks.

Rezultāti izskatās daudz optimistiskāk salīdzinot ar iepriekšējo realizāciju. Tomēr abu realizāciju rezultātus nevar salīdzināt, jo tiek izmantots cits datu ielādēšanas mehānisms. Arhitektūra sakrīt, un abas realizācijas tika notestētas uz “Music Instruments Mini” datu kopas. Failu priekšapstrādes metode gan paliek nemainīga. Abos gadījumos failu spektrs tiek nofiltrēts līdz 8000 Hz, bet diskretizācijas frekvence samazināta līdz 16000 Hz. MFCC ģenerēšanai tiek pielietota viena un tā pati metode un parametri, izgrieztās klusās vietas un laika logs, kas griež fragmentus trenēšanai ir līdzīgs pēc garuma. Sakrīt arī optimizācijas metode (RADAM). Rodas aizdomas par programmēšanas kļūdām PyTorch realizācijā, bet tas paliek par priekšmetu turpmākai izpētei.

Nākamajā solī uz “Music instruments 10” datu apakškopas tiek apmācītā ResNet50 arhitektūra. Laika ierobežojuma dēļ apmācība notika tikai 7 epohās. Taču modelī ir vairāk slāņu nekā visos iepriekš testētajos modeļos, tāpēc bija ļoti interesanti redzēt, cik labus radītājus ir iespējams iegūt.

“Music instruments 10”			
Klase	Precizitāte	Jūtība	F1
Acoustic_guitar	0.91	0.9	0.90
Bass_drum	0.97	0.9	0.93
Cello	0.94	0.9	0.92
Clarinet	0.92	0.92	0.92
Double_bass	0.96	0.75	0.84
Flute	0.94	0.91	0.92
Hi-hat	1	0.9	0.95
Saxophone	0.94	0.92	0.93
Snare_drum	0.95	0.6	0.74
Violin_or_fiddle	0.92	0.93	0.92
Pareizība			0.90
Vidējais	0.95	0.86	0.90
Svērtais vidējais	0.91	0.86	0.90



Ilustrācija 3.5. ResNet50 validācijas radītāji apmācību gaitā 7 epohās.

Tabula 3-4. 7 epohas apmācīts ResNet50, testēšanas rezultāts izmantojot iepriekš neredzētus datus.

Jau pēc 3 epohām pārbaude uz testēšanas datiem dod rādītājus, kas ir pavisam tuvu vienkāršam ConvNet pēc 7 epohām. Rezultāts atbilst sagaidītam, jo citu pētījumu publikācijās ResNet arhitektūras arī dod augstākus rādītājus, salīdzinot ar parastiem ConvNet dziļiem neironu tīkliem(Jay 2018; He et al. 2015).

Pēdējā posmā ResNet50 un ConvNet tika apmācīts uz pilnās datu kopas, kas ietilpst jau visas 41 datu FSD datu kopas kategorijas. Lai samazinātu apmācības laiku, apmācības paketes izmērs tika palielināts no 32 līdz 256 fragmentiem (rezultāti ir attēloti tabulā 3.5 pēdējās 2 rindās).

Modelis	Datu kopa/apakškopa	Klases	Epohu daudzums	Paketes izmērs	Apmācības failu daudzums	Validācijas failu daudzums	Testēšanas failu daudzums	Vidējā testēšanas precizitāte	Vidējā testēšanas F1 metrika
ConvNet	"Music Instruments 10"	10	7	32	2100	900	569	0.91	0.88
ConvNet	"Environment sounds 10"	10	7	32	1523	653	317	0.58	0.57
ResNet50	"Music Instruments 10"	10	7	32	2100	900	569	0.95	0.90
ConvNet	FSD Kaggle	41	14	256	6631	2842	1601	0.67	0.68
ResNet50	FSD Kaggle	41	14	256	6631	2842	1601	0.66	0.67

Tabula 3.5. Tensorflow uzbūvēto modeļu realizācijas testēšanas kopsavilkums

Salīdzinot ConvNet un ResNet50 arhitektūras, nevar pateikt, ka apmācoties un klasificējot audio failus no pilnās 41 klašu FSD datu kopas ResNet50 dod labākus rezultātus (apjukuma matricas ir 4. un 5. pielikumos).

4. TALĀKIE PĒTĪJUMI

Pirmkārt, man ir jāsaprot, kāpēc neizdevās iegūt labus rezultātus pirmajā realizācijā, kas bija uztaisīta uz PyTorch. Cik es pagaidām saprotu, es saskaros ar parādību, kas parasti tiek saukta par pārprielāgošanu⁷¹. Tas notiek jau pēc pirmās epohas, un tālāk kļūdas funkcija testēšanas datu kopai tikai pieaug. Neizskatās, ka to ietekmē klašu daudzums datu kopā vai tās izmērs. Šaubos, vai problēma ir pašos modeļos, jo tie nav pārāk sarežģīti un bija testēti uz Fashion MNIST datu kopas, ar kuru veiksmīgi apmācījās. Pagaidām man ir divas versijas:

- Problēma ir saistītā ar kļūdām datu priekšapstrādes un ielādēs kodā. Varbūt audio failu fragmenti ir nepareizi sagriezti daļās, vai kaut kāda brīdī tām kļūdaini tiek piešķirts īsto klašu apzīmējums
- Rezultāts ir pareizs un izvēlētie modeļi (gan arī datu priekšapstrādes metodes) nestrādā labi ar konkrētā tipa uzdevumiem. Bet Tensorflow realizācija sniedz labus rezultātus nepareizās datu ielādes sistēmas dēļ (tiek izmantota nejaušas secības fragmentu izvēle apmācības un testēšanas gaitā)

Pirmo versiju var pārbaudīt tikai rūpīgi atklājot datu ielādēs kodu. Plāns ir iziet cauri visiem soļiem un, izvadot sagriezto fragmentu cepstrogrammas uz ekrāna, novērtēt, cik tie ir relevanti. Pārbaudīt, vai dati nav sabojāti, un, vai ierakstītais ar to asociētais kategoriju apzīmējums ir relevant.

Otro versiju var pārbaudīt pārrakstot datu ielādes metodi no Tensorflow realizācijas uz PyTorch (vai otrādi). Ja kļūdas funkcija PyTorch realizācijā testu kopai vēl joprojām nesamazināsies, būs nepieciešams meklēt citas atšķirības starp realizācijām.

Dziļo modeļu apmācība prasa ievērojamus skaitļošanas resursus, un tās prasības pieaug, līdz ar datu kopas izmēriem. Tāpēc industrijā aug pieprasījums pēc superdatoriem, kas tiek aprīkoti ar vairākiem grafiskajiem procesoriem un spēj vienlaicīgi trenēt un testēt vairākus modeļus. Tas arī ir svarīgs aspekts efektīvu modeļu izveidei – spēja pārbaudīt vairākas parametru kombinācijas un datu priekšapstrādes variācijas. Nākamajā solī ir plānots salīdzināt šajā pētījumā apskatīto modeli ar daudzslāņu arhitektūru, kas aizņēma 8. vietu Kaggle⁷²

⁷¹ Overfitting (angl)

⁷² Skaņas avotu klasificēšanas konkurss Kaggle platformā: <https://www.kaggle.com/c/freesound-audio-tagging>

sacensības⁷³. Līdz šim tas nebija iespējams, tā kā dziļākas arhitektūras uz personīgā datora darbojas ļoti lēni, pat izmantojot CUDA (realizācija izmanto dziļo arhitektūru MobileNetV2 (Sandler et al. 2019)). Tādēļ ir plāns izmantot RTU HPC⁷⁴ superdatoru, ar kura palīdzību tāds eksperiments būtu iespējams.

Eksperimenta gaitā tika pamanīts, ka, apmācot vienkāršo ConvNet modeli uz mūzikas instrumentu skaņām un atsevišķi uz arējās vides skaņām, modelis labāk mācas tieši uz mūzikas instrumentiem. Manuprāt, tā ir interesanta hipotēze un nākotnē būtu noderīgi pārbaudīt:

- Vai būs līdzīgs rezultāts, ja datu apakškopās būs līdzīgs audio failu daudzums un summārs kopas ierakstu laiks?
- Cik ļoti mainīsies rezultāts, izmantojot sarežģītākās arhitektūras (piemēram, ResNet modifikācijas)?
- Cik atšķirīgs būs rezultāts, izmantojot citas audio failu reprezentēšanas metodes (FFT spektrogrammas un Mel frekvences spektrogrammas)?

Vēl būtu interesanti atdarināt arhitektūru un datu kopas sagatavošanas metodes kādam no zināmiem pētījumiem (piemēram, Nevo Segal (Segal, 2016)). Es gribētu pārbaudīt, cik reproducējami ir pētījuma rezultāti, un vai tos iespējams uzlabot, ja ir pieejams darba pirmkods.

Parādījās ideja pamēģināt apmācīt modeli, izmantojot trīs audio ierakstu reprezentēšanas metodes vienlaicīgi (piemēram: klasiskās FFT spektrogrammas, Mel frekvences spektrogrammas un MFCC cepstrogrammas), kā trīs kanālus. Priekšapstrādes solī būs jānodrošina vienādi izmēri visu trīs kanālu matricām. Ideja ir radusies no izplatītas prakses krāsainu attēlu klasifikācijai izmantot trīs krāsu kanālus uz ieejas konvolūcijas modeļos.

Skatoties globāli, man ir interese ne tikai par skaņas avotu klasificēšanu un atpazīšanu, bet arī skaņdarbu žanru klasificēšanu. Tāpēc, tiklīdz būs eksperimentāli apstiprināta viena vai otra modeļa efektivitāte, es labprāt paplašinātu uzdevumu un klasificētu skaņdarbu fragmentus. Tas, no vienas puses, ir līdzīgs uzdevums. Tāpat kā šajā pētījumā, būs nepieciešamība reprezentēt audio ierakstus matricu veidā, izmantot dažādas metodes ierakstu priekšapstrādei, izstrādāt un notestēt vairākus modeļus ar mērķi atlasīt visefektīvākos. No otras puses, šādā

⁷³ Sainath Adappa: <https://www.kaggle.com/c/freesound-audio-tagging/discussion/64262>

⁷⁴ High Performance Computing, angl: <https://hpc.rtu.lv/kas-ir-hpc/>

uzdevumā modelim jāiemācās citas pazīmes – likumsakarības - un provizoriski var pateikt, ka tas būs sarežģītāk. Jo žanrus atšķir ne tikai izmantoto mūzikas instrumentu tembri un instrumentu sastāvs, bet arī muzikālās pazīmes – ritms, melodija, harmonija. Būtu ļoti noderīgi saprast, cik veiksmīgi var pielietot mašīnāpmācības modeļus šādu uzdevumu risināšanai.

5. SECINĀJUMI

Neskatoties uz vairākām grūtībām, ar kurām es saskaros, realizējot bakalaura darba praktisko daļu, tā ir ļoti pozitīva pieredze. Izdevās apskatīt divas populārākas industrijas platformas un abos gadījumos uzrakstīt strādājošo kodu. Realizējot projektu bija iespējams uz savas personīgas pieredzes iepazīt stiprās un vājās PyTorch un Tensorflow puses, kaut gan tas nebija darba pamatmērķis.

Manuprāt, PyTorch kodēšanas stils prasa plašākas teorētiskās zināšanas un dziļāku saprašanu par iekšējiem procesiem, kas notiek modeļu apmācīšanas un testēšanas procesā. Tāpat kā iekš Keras, tajās ir augsta līmeņa API elementi (tas pats Sequential, kas arī ļauj vienkāršotā veidā programmēt grafus). Bet, kopumā, realizācija prasa plašāku programmēšanas pieredzi un intuīciju. Vēl es konstatēju, ka man bija vieglāk sekot visām tensoru manipulācijām tieši PyTorch vidē, un tas ļoti pozitīvi ietekmēja manas spējas realizēt projektu ar Tensorflow (ar kuru es iesāku iepazīties ar dziļām mašīnāpmācībām gadu atpakaļ, bet ilgi īsti nevarēju saprast, kā modelis tiek uzbūvēts un mācās).

Pirmais pētījuma secinājums ir sekojošs: **datu ielādēšanas un priekšapstrādes metode stipri ietekmē uz eksperimenta rezultātus**. Tas ir ļoti loģiski un no tā seko, ka nemainot pašu sistēmu (kā arī klašu daudzumu), bet tikai izmantojot citas kategorijas, tiek iegūti pavisam citi apmācības radītāji (kā manā gadījuma ar mūzikas instrumentiem un citām skaņām). Tāpēc pat nelielas izmaiņas datu ielādes metodēs var krietni ietekmēt uz darba rezultātus un ir jābūt ļoti uzmanīgam.

Otrais secinājums: **modeļu kopīgais apmācības laiks arī ir svarīgs faktors, kas jāņem vērā vērtējot efektivitāti**. Patērētais resursu daudzums gadījumos ar nopietnām datu kopām un dziļiem mākslīgo neironu tīkliem var būt astronomisks. Darba mērķis bija novērtēt modeļa efektivitāti un iegūt visoptimālāko konkrētam uzdevumam. Tomēr viens no efektivitātes kritērijiem ir laika resursu racionāls patēriņš (pēc iespējas labāki validācijas radītāji visīsākā laika periodā). Es rūpīgi nesekoju līdz patērētajam laikam, veicot eksperimentus, un tā ir lielā kļūda, kas jāņem vērā turpmāk.

Trešais secinājums: **vienkārša ConvNet arhitektūra labāk klasificē mūzikas instrumentus nekā cita tipa skaņas**. Šī tēze nav viennozīmīgā un prasa atkārtotas pārbaudes. Bet salīdzinājums tabulā 4.3 norāda uz to, ka modelis apmācījās labāk tieši uz pirmās apakškopas. Tas var būt saistīts ar audio failu reprezentēšanas pieeju un faktu, ka tika izmantoti

Mel frekvences cepstrālie koeficienti. Bet apstiprināt vai noraidīt šo tēzi varētu būt svarīgi, lai labāk saprastu kādas pazīmes mūs interesē mūzikas instrumentu vai citu skaņu atpazīšanā.

Kopumā es uzskatu, ka darba mērķis ir sasniegts. Tika pārbaudītas divās arhitektūras un tika izmēģinātas pat divās programmēšanas platformās arhitektūru apmācībai un testēšanai. Bet industrijā pastāv milzīgs daudzums ar metodēm⁷⁵. Tagad es saprotu, ka vajadzētu pārbaudīt vairāku arhitektūru dažādas parametru kombinācijas. Kā arī izmēģināt dažādas datu kopas kombinācijas un datu priekšapstrādes veidus. Diemžēl tas viss palika ārpus šī darba realizācijas.

Tomēr darbs bija produktīvs no pašizglītības skatu punkta, un tagad es daudz labāk orientējos jomā. Tagad galvenais jautājums, kas man ir, cik liela mērā ir reproducējami ir pētījumi, kas tiek aprakstīti publikācijās par dziļām mašīnāpmācībām. Man šķiet, ka eksperimentu reproducēšana ir lielisks veids, kā var apgūt jaunas iemaņas šajā jomā un tieši šajā virzienā es plānoju koncentrēt savus spēkus tuvākajā nākotnē.

⁷⁵ Torchvision datorredzes bibliotēkā var atrast 12 dažādas dziļās arhitektūras. Bet tās var dalīt, kombinēt un pilnveidot.

6. IZMANTOTĀ LITERATŪRA

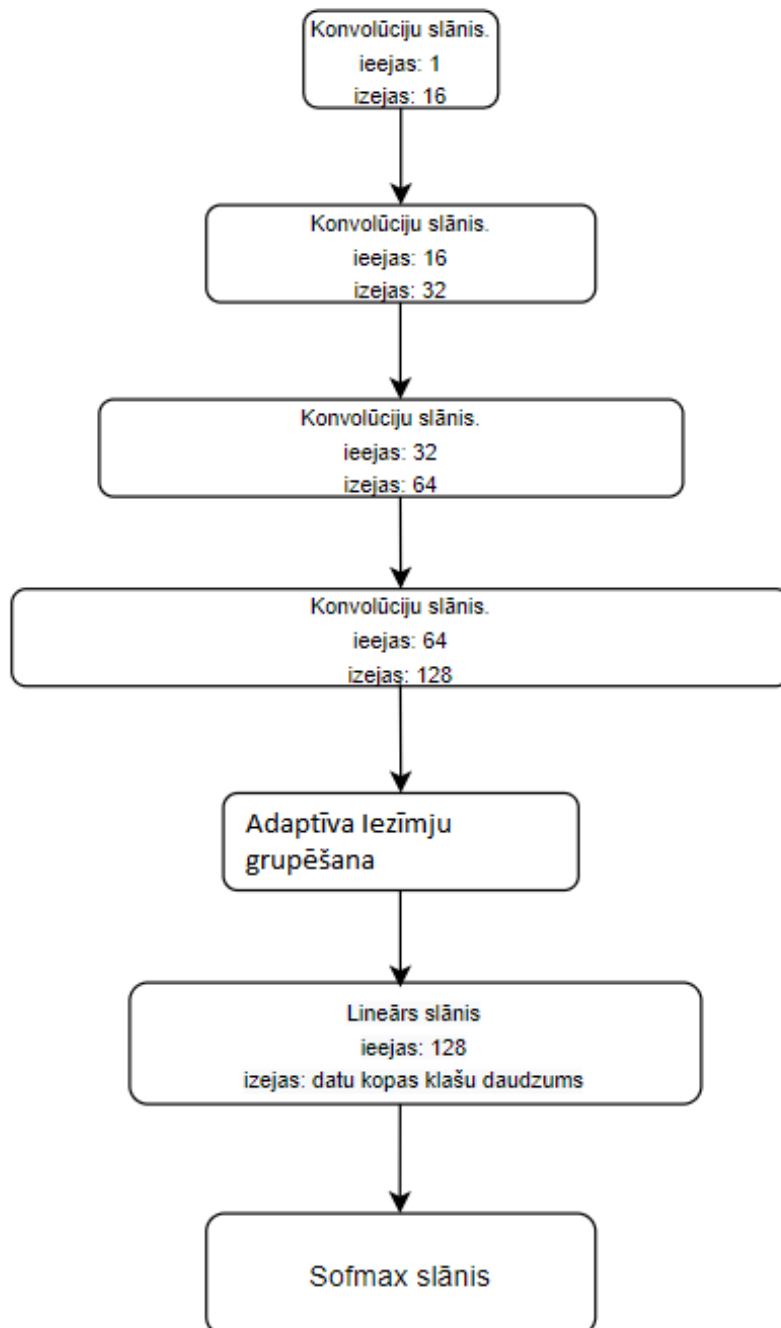
- “6.3. Padding and Stride — Dive into Deep Learning 0.7.1 Documentation.” n.d. Accessed April 29, 2020. https://d2l.ai/chapter_convolutional-neural-networks/padding-and-strides.html.
- Alber, Maximilian, Irwan Bello, Barret Zoph, Pieter-Jan Kindermans, Prajit Ramachandran, and Quoc Le. 2018. “Backprop Evolution.” *ArXiv:1808.02822 [Cs, Stat]*, August. <http://arxiv.org/abs/1808.02822>.
- Basheer, I.A, and M Hajmeer. 2000. “Artificial Neural Networks: Fundamentals, Computing, Design, and Application.” *Journal of Microbiological Methods* 43 (1): 3–31. [https://doi.org/10.1016/S0167-7012\(00\)00201-3](https://doi.org/10.1016/S0167-7012(00)00201-3).
- Bilmes, Jeff. 2008. “Gaussian Models in Automatic Speech Recognition.” In *Handbook of Signal Processing in Acoustics*, edited by David Havelock, Sonoko Kuwano, and Michael Vorländer, 521–55. New York, NY: Springer New York. https://doi.org/10.1007/978-0-387-30441-0_29.
- Bottou, Léon. 2012. “Stochastic Gradient Descent Tricks.” In *Neural Networks: Tricks of the Trade*, edited by Grégoire Montavon, Geneviève B. Orr, and Klaus-Robert Müller, 7700:421–36. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-35289-8_25.
- Buhrmester, Vanessa, David Münch, and Michael Arens. 2019. “Analysis of Explainers of Black Box Deep Neural Networks for Computer Vision: A Survey.” *ArXiv:1911.12116 [Cs]*, November. <http://arxiv.org/abs/1911.12116>.
- Chiu, Chung-Cheng, Tara N. Sainath, Yonghui Wu, Rohit Prabhavalkar, Patrick Nguyen, Zhifeng Chen, Anjuli Kannan, et al. 2018. “State-of-the-Art Speech Recognition With Sequence-to-Sequence Models.” *ArXiv:1712.01769 [Cs, Eess, Stat]*, February. <http://arxiv.org/abs/1712.01769>.
- “CS231n Convolutional Neural Networks for Visual Recognition.” n.d. Accessed April 29, 2020. <https://cs231n.github.io/convolutional-networks/>.
- David Martin Ward Powers. 2011. “Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness & Correlation.” *Journal of Machine Learning Technologies*. <https://doi.org/10.9735/2229-3981>.
- De La Calle Silos, Fernando. 2017. “Bio-Motivated Features and Deep Learning for Robust Speech Recognition,” 179.
- “Derivative Chain Rule - Mathonline.” n.d. Accessed April 29, 2020. <http://mathonline.wikidot.com/derivative-chain-rule>.
- Dullemond, Kees, and Kasper Peeters. n.d. “Introduction to Tensor Calculus,” 53.
- El-Habil, Abdalla M. 2012. “An Application on Multinomial Logistic Regression Model.” *Pakistan Journal of Statistics and Operation Research* 8 (2): 271. <https://doi.org/10.18187/pjsor.v8i2.234>.
- Eringis, Deividas, and Gintautas Tamulevičius. 2016. “Modified Filterbank Analysis Features for Speech Recognition,” 14.
- Eronen, A., and A. Klapuri. 2000. “Musical Instrument Recognition Using Cepstral Coefficients and Temporal Features.” In *2000 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.00CH37100)*, 2:II753–56. Istanbul, Turkey: IEEE. <https://doi.org/10.1109/ICASSP.2000.859069>.
- Fonseca, Eduardo, Manoj Plakal, Frederic Font, Daniel P. W. Ellis, Xavier Favory, Jordi Pons, and Xavier Serra. 2018. “General-Purpose Tagging of Freesound Audio with

- AudioSet Labels: Task Description, Dataset, and Baseline.” *ArXiv:1807.09902 [Cs, Eess, Stat]*, October. <http://arxiv.org/abs/1807.09902>.
- Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning (Adaptive Computation and Machine Learning Series)*. <https://www.deeplearningbook.org/>.
- He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. “Deep Residual Learning for Image Recognition.” *ArXiv:1512.03385 [Cs]*, December. <http://arxiv.org/abs/1512.03385>.
- “How Does the Softmax Activation Function Work?” 2020. *MachineCurve* (blog). January 8, 2020. <https://www.machinecurve.com/index.php/2020/01/08/how-does-the-softmax-activation-function-work/>.
- Ioffe, Sergey, and Christian Szegedy. 2015. “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.” *ArXiv:1502.03167 [Cs]*, March. <http://arxiv.org/abs/1502.03167>.
- Jay, Prakash. 2018. “Understanding and Implementing Architectures of ResNet and ResNeXt for State-of-the-Art Image...” Medium. April 8, 2018. <https://medium.com/@14prakash/understanding-and-implementing-architectures-of-resnet-and-resnext-for-state-of-the-art-image-cf51669e1624>.
- Kim, Yoon. 2014. “Convolutional Neural Networks for Sentence Classification.” *ArXiv:1408.5882 [Cs]*, September. <http://arxiv.org/abs/1408.5882>.
- Kingma, Diederik P., and Jimmy Ba. 2017. “Adam: A Method for Stochastic Optimization.” *ArXiv:1412.6980 [Cs]*, January. <http://arxiv.org/abs/1412.6980>.
- Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. 2017. “ImageNet Classification with Deep Convolutional Neural Networks.” *Communications of the ACM* 60 (6): 84–90. <https://doi.org/10.1145/3065386>.
- Liu, Liyuan, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. 2020. “On the Variance of the Adaptive Learning Rate and Beyond.” *ArXiv:1908.03265 [Cs, Stat]*, April. <http://arxiv.org/abs/1908.03265>.
- Marques, Janet, and Pedro J Moreno. n.d. “A Study of Musical Instrument Classification Using Gaussian Mixture Models and Support Vector Machines,” 21.
- Mcculloch, Warren S, and Walter Pitts. n.d. “A LOGICAL CALCULUS OF THE IDEAS IMMANENT IN NERVOUS ACTIVITY,” 17.
- O’Mahony, Niall, Sean Campbell, Anderson Carvalho, Suman Harapanahalli, Gustavo Velasco Hernandez, Lenka Krpalkova, Daniel Riordan, and Joseph Walsh. 2020. “Deep Learning vs. Traditional Computer Vision.” In *Advances in Computer Vision*, edited by Kohei Arai and Supriya Kapoor, 943:128–44. Advances in Intelligent Systems and Computing. Cham: Springer International Publishing. https://doi.org/10.1007/978-3-030-17795-9_10.
- Qin, Zhenyue, Dongwoo Kim, and Tom Gedeon. 2020. “Rethinking Softmax with Cross-Entropy: Neural Network Classifier as Mutual Information Estimator.” *ArXiv:1911.10688 [Cs, Stat]*, March. <http://arxiv.org/abs/1911.10688>.
- Rumelhart, D E, G E Hinton, and R J Williams. n.d. “Learning Internal Representations by Error Propagation,” 23.
- Sandler, Mark, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. 2019. “MobileNetV2: Inverted Residuals and Linear Bottlenecks.” *ArXiv:1801.04381 [Cs]*, March. <http://arxiv.org/abs/1801.04381>.
- Schmidhuber, J., F. Gers, and D. Eck. 2002. “Learning Nonregular Languages: A Comparison of Simple Recurrent Networks and LSTM.” *Neural Computation* 14 (9): 2039–41. <https://doi.org/10.1162/089976602320263980>.

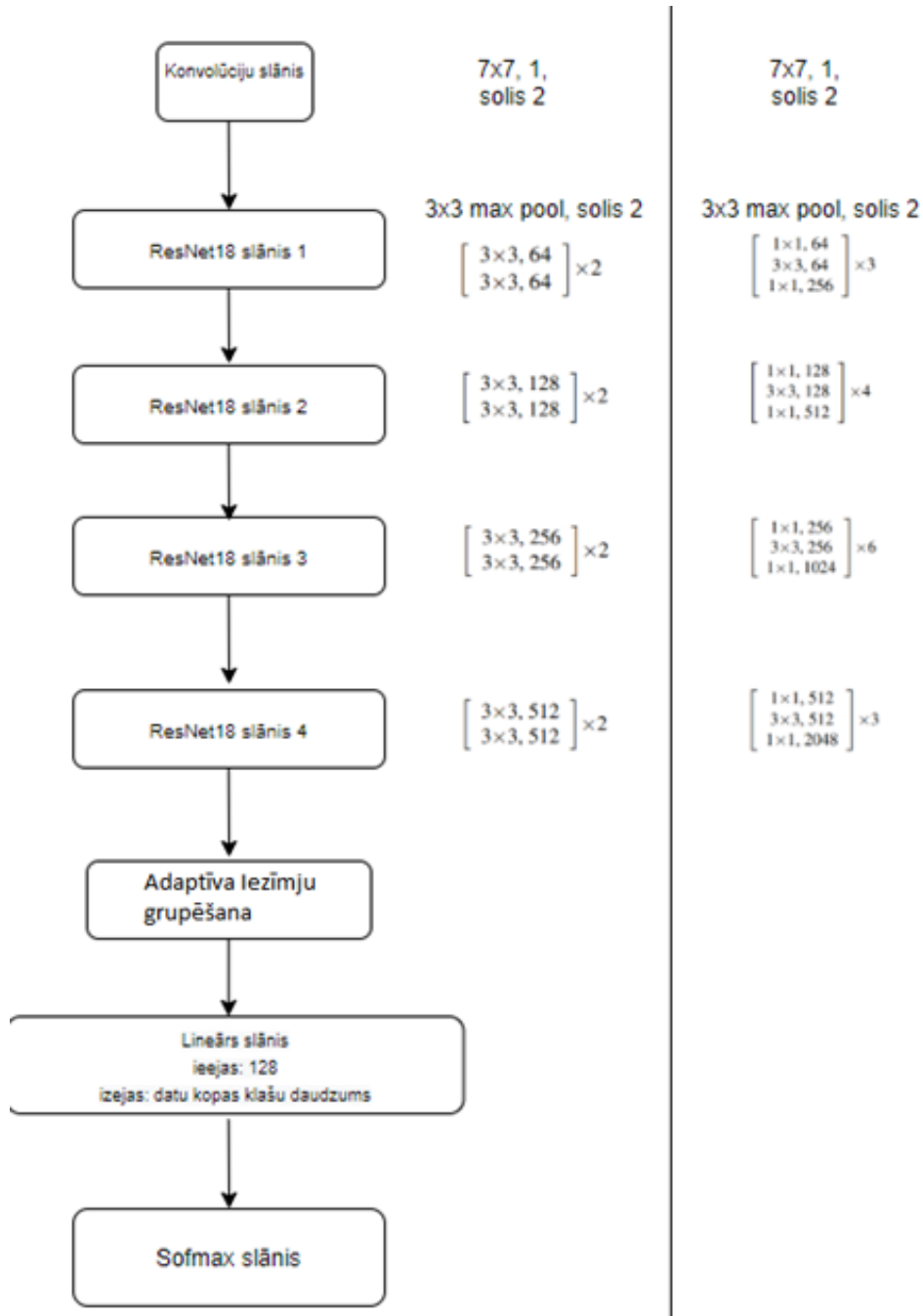
- Segal, Nevo. 2016. "Automatic Musical Instrument Recognition Using Convolutional Neural Networks," 51.
- Singh, Prabhjyot, Dnyaneshwar Bachhav, Omkar Joshi, and Nita Patil. 2019. "Musical Instrument Recognition Using CNN and SVM" 06 (03): 4.
- Skansi, Sandro. 2018. *Introduction to Deep Learning: From Logical Calculus to Artificial Intelligence*. Undergraduate Topics in Computer Science. Cham: Springer International Publishing. <https://doi.org/10.1007/978-3-319-73004-2>.
- "The 'Terpret Problem' and the Limits of SGD." 2018. Daniel Selsam. July 21, 2018. <https://dselsam.github.io/the-terpret-problem/>.
- "Vanishing / Exploding gradients - Practical aspects of Deep Learning." n.d. Coursera. Accessed April 29, 2020. <https://ru.coursera.org/lecture/deep-neural-network/vanishing-exploding-gradients-C9iQO>.
- "What Are Convolution Neural Networks? [ELI5] | Hacker Noon." n.d. Accessed April 29, 2020. <https://hackernoon.com/-understanding-convolution-neural-networks-cnn-the-eli5-way-photo-by-efe-kurnaz-on-unsplash-u-pa1i327j>.
- Zhao, Zishuo, and Haoyun Wang. 2019. "Musical Instrument Classification via Low-Dimensional Feature Vectors." *ArXiv:1909.08444 [Cs, Eess]*, September. <http://arxiv.org/abs/1909.08444>.

PIELIKUMI

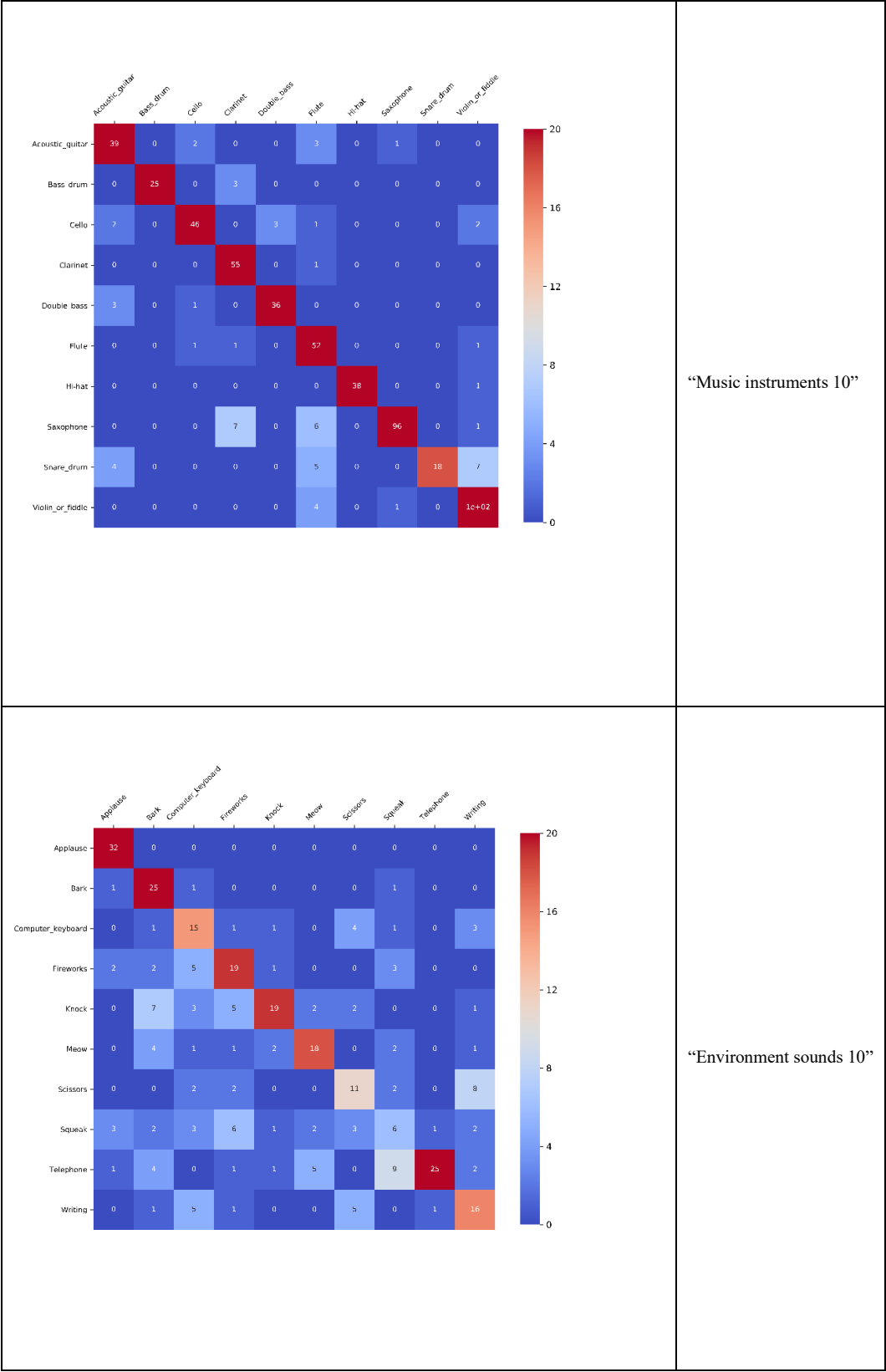
1. pielikums. ConvNet.



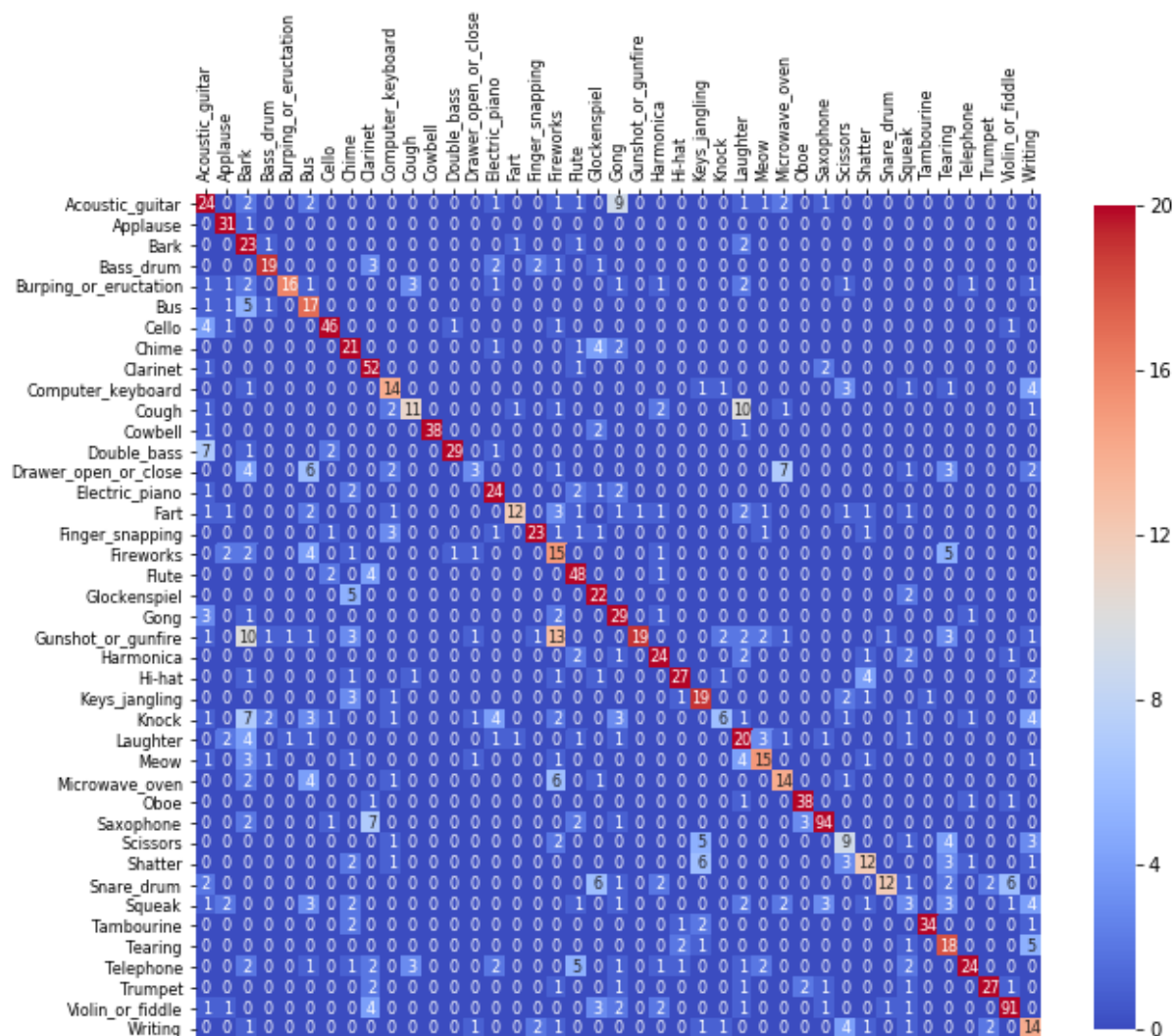
2. pielikums. ResNet18 un ResNet50



3. pielikums. Vienkāršas ConvNet apmācīts uz 10 klašu datu kopām un notestēts uz iepriekš neredzētiem datiem.



4. pielikums. ConvNet apmācīts uz 41 klašu datu kopām un notestēts uz iepriekš neredzētiem datiem



5. pielikums. ResNet50 apmācīts uz 41 klašu datu kopām un notestēts uz iepriekš neredzētiem datiem

