Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií
FIIT-182905-75095

Bc. Martin Žák

MODELOVANIE ČŔT HUDOBNÝCH AUDIO SIGNÁLOV POMOCOU ALGORITMOV STROJOVÉHO UČENIA

Diplomová práca

Študijný program:	Inteligentné softvérové systémy
Študijný odbor:	18. Informatika
Miesto vypracovania:	Ústav informatiky a softvérového inžinierstva, FIIT STU, Bratislava
Vedúci práce:	Ing. Lukáš Marták

máj 2021

STU FIIT
SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

Zadanie diplomovej práce

Meno študenta: **Bc. Martin Žák**

Študijný program: Inteligentné softvérové systémy

Študijný odbor: Softvérové inžinierstvo – hlavný študijný odbor
Umelá inteligencia – vedľajší študijný odbor

Názov práce: **Modelovanie čŕt hudobných audio signálov pomocou umelých neurónových sietí**

Samostatnou výskumnou a vývojovou činnosťou v rámci predmetov Diplomový projekt I, II, III vypracujte diplomovú prácu na tému, vyjadrenú vyššie uvedeným názvom tak, aby ste dosiahli tieto ciele:

Všeobecný cieľ:

Vypracovaním diplomovej práce preukážte, ako ste si osvojili metódy a postupy riešenia relatívne rozsiahlych projektov, schopnosť samostatne a tvorivo riešiť zložité úlohy aj výskumného charakteru v súlade so súčasnými metódami a postupmi študovaného odboru využívanými v príslušnej oblasti a schopnosť samostatne, tvorivo a kriticky pristupovať k analýze možných riešení a k tvorbe modelov.

Špecifický cieľ:

Vytvorte riešenie zodpovedajúce návrhu textu zadania, ktorý je prílohou tohto zadania. Návrh bližšie opisuje tému vyjadrenú názvom. Tento opis je záväzný, má však rámcový charakter, aby vznikol dostatočný priestor pre Vašu tvorivosť.

Riaďte sa pokynmi Vášho vedúceho.

Pokiaľ v priebehu riešenia, opierajúc sa o hlbšie poznanie súčasného stavu v príslušnej oblasti, alebo o priebežné výsledky Vášho riešenia, alebo o iné závažné skutočnosti, dospejete spoločne s Vaším vedúcim k presvedčeniu, že niečo v texte zadania a/alebo v názve by sa malo zmeniť, navrhnite zmenu. Zmena je spravidla možná len pri dosiahnutí kontrolného bodu.

Miesto vypracovania: Ústav počítačového inžinierstva a aplikovanej informatiky, FIIT STU Bratislava

Vedúci práce: **Ing. Lukáš Samuel Marták**

Termíny odovzdania:

Podľa harmonogramu štúdia platného pre semester, v ktorom máte príslušný predmet (Diplomový projekt I, II, III) absolvovať podľa Vášho študijného plánu

Predmety odovzdania:

V každom predmete dokument podľa pokynov na www.fiit.stuba.sk v časti: home > Informácie o > štúdiu > harmonogram štúdia > diplomový projekt.

V Bratislave dňa 11. 2. 2019

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
Fakulta informatiky a informačných technológií
Ilkovičova 2, 842 16 Bratislava 4
1

prof. Ing. Pavol Návrat, PhD.
riaditeľ Ústavu informatiky, informačných systémov a softvérového inžinierstva

STU FIIT
SLOVENSKÁ TECHNICKÁ
UNIVERZITA V BRATISLAVE
FAKULTA INFORMATIKY
A INFORMAČNÝCH TECHNOLÓGIÍ

Návrh zadania diplomovej práce

Finálna verzia do diplomovej práce [1]

Študent:

Meno, priezvisko, tituly: Martin Žák, Bc.
Študijný program: Inteligentné softvérové systémy
Kontakt: zakyz.martin@gmail.com

Výskumník:

Meno, priezvisko, tituly: Lukáš Samuel Marták, Ing.

Projekt:

Názov:	Modelovanie čŕt hudobných audio signálov pomocou umelých neurónových sietí
Názov v angličtine:	Modelling features of musical audio signals using artifical neural networks
Miesto vypracovania:	Ústav počítačového inžinierstva a aplikovanej informatiky, FIIT STU
Oblasť problematiky:	Spracovanie audio signálov, hlboké neurónové siete

Text návrhu zadania[2]

Metódy hlbokého učenia si našli široké uplatnenie v rôznych aplikačných doménach, ako napríklad spracovanie textu, prirodzeného jazyka, obrazu, ale aj zvuku. Vďaka rozmanitému aplikačnému výskumu existujú rôzne architektúry hlbokých neurónových sietí, ktoré sa dnes vedia vysporiadať so zašumenými či neúplnými dátami a vyextrahovať z nich dôležité črty a vzťahy. Nasvedčujú tomu aj rôzne architektúry umelých neurónových sietí, ktoré pri riešení problémov spracovania akustickej hudby vedia konkurovať donedávna veľmi často využívaným štatistickým modelom alebo iným prístupom strojového učenia.

Analyzujte existujúce prístupy k spracovaniu signálov akustickej hudby za účelom modelovania prítomných rytmických, melodických, či harmonických zložiek hudobného záznamu. Zamerajte sa na prístupy špecializované pre vybranú skupinu hudobných nástrojov a na metódy využívajúce umelé neurónové siete.

Navrhnite a zrealizujte metódu spracovania digitálneho záznamu zvukového signálu za účelom extrakcie čŕt popisujúcich zodpovedajúci hudobný obsah. Inšpirujte sa riešeniami z oblasti strojového učenia využívajúcimi hlboké neurónové siete. Pri realizácií riešenia berte do úvahy doménové črty hudobných audio signálov a pokúste sa nimi návrh svojej metódy podporiť. Riešenie overte na verejne dostupných dátach a výsledné hodnoty porovnajte s existujúcimi riešeniami.

---

[1] Vytlačiť obojstranne na jeden list papiera

[2] 150-200 slov (1200-1700 znakov), ktoré opisujú výskumný problém v kontexte súčasného stavu vrátane motivácie a smerov riešenia

Literatúra[3]

- C. Southall, R. Stables a J. Hockman, „Automatic Drum Transcription Using Bi-Directional Recurrent Neural Networks," In Proc. of the 17th International Society for Music Information Retrieval Con- ference, New York, 2016.
- R. Vogl, M. Dorfer, G. Widmer a P. Knees, „Drum transcription via joint beat and drum modeling using convolutional recurrent neural networks," In Proc. of the 18th International Society for Music Information Retrieval Conference, Suzhou, 2017.

Vyššie je uvedený návrh diplomového projektu, ktorý vypracoval(a) Bc. Martin Žák, konzultoval(a) a osvojil(a) si ho Ing. Lukáš Samuel Marták a súhlasí, že bude takýto projekt viesť v prípade, že bude pridelený tomuto študentovi.

V Bratislave dňa 8.1.2019

Podpis študenta

Podpis výskumníka

Vyjadrenie garanta predmetov Diplomový projekt I, II, III

Návrh zadania schválený: áno / ~~nie~~[4]

Dňa: 11.2.2019

Podpis garanta predmetov

[3] 2 vedecké zdroje, každý v samostatnej rubrike a s údajmi zodpovedajúcimi bibliografickým odkazom podľa normy STN ISO 690, ktoré sa viažu k téme zadania a preukazujú výskumnú povahu problému a jeho aktuálnosť (uveďte všetky potrebné údaje na identifikáciu zdroja, pričom uprednostnite vedecké príspevky v časopisoch a medzinárodných konferenciách)

[4] Nehodiace sa prečiarknite

SLOVENSKÁ TECHNICKÁ
UNIVERZITA V BRATISLAVE
FAKULTA INFORMATIKY
A INFORMAČNÝCH TECHNOLÓGIÍ

Návrh zadania diplomovej práce

Revízia č.: 1 [1]

Študent:

Meno, priezvisko, tituly:	Martin Žák, Bc.
Študijný program:	Inteligentné softvérové systémy
Kontakt:	zakyz.martin@gmail.com

Výskumník:

Meno, priezvisko, tituly:	Lukáš Samuel Marták, Ing.

Projekt:

Názov:	Modelovanie čŕt hudobných audio signálov pomocou algoritmov strojového učenia
Názov v angličtine:	Modelling features of musical audio signals using machine learning algorithms
Miesto vypracovania:	Ústav počítačového inžinierstva a aplikovanej informatiky, FIIT STU
Oblasť problematiky:	Spracovanie audio signálov, algoritmy strojového učenia

Text návrhu zadania[2]

Metódy strojového učenia si našli široké uplatnenie v rôznych aplikačných doménach, ako napriklad spracovanie textu, prirodzeného jazyka, obrazu, ale aj zvuku. Vd'aka rozmanitému aplikačnému výskumu existujú rôzne varianty algoritmov strojového učenia, ktoré sa dnes vedia vysporiadať so zašumenými či neúplnými dátami a extrahovať z nich dôležité črty a vzťahy. Príkladom sú rôzne architektúry umelých neurónových sietí alebo varianty algoritmu nezápornej maticovej faktorizácie.

Analyzujte existujúce prístupy k spracovaniu signálov akustickej hudby za cieľom modelovania prítomných rytmických, melodických, či harmonických zložiek hudobného záznamu. Zamerajte sa na prístupy špecializované pre vybranú skupinu hudobných nástrojov a na metódy využívajúce techniky strojového učenia.

Navrhnite a zrealizujte metódu spracovania digitálneho záznamu zvukového signálu za účelom extrakcie čŕt popisujúcich zodpovedajúci hudobný obsah. Inšpirujte sa riešeniami z oblasti strojového učenia využívajúcimi hlboké neurónové siete a nezápornú maticovú faktorizáciu. Pri realizácii riešenia berte do úvahy doménové črty hudobných audio signálov a pokúste sa nimi návrh svojej metódy podporiť. Riešenie overte na verejne dostupných dátach a výsledné hodnoty porovnajte s existujúcimi riešeniami.

---

[1] Vytlačiť obojstranne na jeden list papiera

[2] 150-200 slov (1200-1700 znakov), ktoré opisujú výskumný problém v kontexte súčasného stavu vrátane motivácie a smerov riešenia

Literatúra[3]

- C. Southall, R. Stables a J. Hockman, „Automatic Drum Transcription Using Bi-Directional Recurrent Neural Networks,“ In Proc. of the 17th International Society for Music Information Retrieval Con- ference, New York, 2016.
- R. Vogl, M. Dorfer, G. Widmer a P. Knees, „Drum transcription via joint beat and drum modeling using convolutional recurrent neural networks,“ In Proc. of the 18th International Society for Music Information Retrieval Conference, Suzhou, 2017.

Vyššie je uvedený návrh diplomového projektu, ktorý vypracoval(a) Bc. Martin Žák, konzultoval(a) a osvojil(a) si ho Ing. Lukáš Samuel Marták a súhlasí, že bude takýto projekt viesť.

V Bratislave dňa 6.5.2021

Podpis študenta

Podpis výskumníka

Vyjadrenie garanta predmetov Diplomový projekt I, II, III

Návrh zadania schválený: áno / nie[4]

Dňa: .....

Podpis garanta predmetov

[3] 2 vedecké zdroje, každý v samostatnej rubrike a s údajmi zodpovedajúcimi bibliografickým odkazom podľa normy STN ISO 690, ktoré sa viažu k téme zadania a preukazujú výskumnú povahu problému a jeho aktuálnosť (uveďte všetky potrebné údaje na identifikáciu zdroja, pričom uprednostnite vedecké príspevky v časopisoch a medzinárodných konferenciách)

[4] Nehodiace sa prečiarknite

Čestne vyhlasujem, že som túto prácu vypracoval samostatne, na základe konzultácií a s použitím uvedenej literatúry

Bratislava 07.05.2021

Bc. Martin Žák

Poďakovanie

Chcem sa primárne poďakovať celej mojej rodine, ktorá ma celý čas podporovala počas tvorby diplomovej práce. Veľká vďaka patrí môjmu skvelému vedúcemu práce, Lukášovi Martákovi, ktorý mi venoval nemalé množstvo času a ochotne ma sprevádzal naprieč všetkými fázami. Obzvlášť som mu vďačný za jeho podrobnú spätnú väzbu, ktorú mi dal po dopísaní práce. Na záver ďakujem všetkým profesorom, od ktorých som mohol čerpať počas štúdií na fakulte a ktorých vedomosti a princípy som aplikoval pri písaní tejto záverečnej práce.

Anotácia

Slovenská technická univerzita v Bratislave
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ
Študijný program: Informatika
Autor: Martin Žák
Diplomová práca: **Modelovanie hudobných čŕt hudobných audio signálov pomocou algoritmov strojového učenia**
Vedúci diplomovej práce: Ing. Lukáš Marták.
máj 2021

Hudba je fenomén, ktorý je medzi nami už tisícročia. Autori klasických žánrov zvykli v minulosti ich populárne diela manuálne prepisovať do notového zápisu, aby ich uchovali pre ďalšie generácie. Prepis hudby, resp. transkripcia však nie je triviálnou záležitosťou. Vždy to bol a stále je kognitívne veľmi náročný proces, ktorý je určený buď pre autorov samotných skladieb, alebo ľudí s vynikajúcim sluchom.

V súčasnosti, kedy vzniká enormné množstvo populárnych skladieb rôznych žánrov, si nie každý autor dáva záležať na tom, aby spravil poctivý notový zápis. Skladby s viacerými hudobnými nástrojmi je potom oveľa ťažšie reprodukovať. No vďaka veľkej výpočtovej sile súčasných počítačov a vzniku pokročilých algoritmov strojového učenia je tu posledné desaťročia snaha proces transkripcie automatizovať, čím sa miera reprodukovateľnosti môže zvýšiť. Keďže transkripcia hudby je veľmi komplexnou úlohou, momentálny stav vedy sa snaží riešiť tento problém na úrovni hudobných nástrojov.

V tejto práci sa zameriavame na automatickú transkripciu bicích nástrojov. Navrhli sme tri metódy, pričom dve z nich do transkripcie zakomponovávajú aj detailnejšie informácie o hracích technikách na hudobné nástroje, čo konkurenčným metódam poväčšine chýba. Túto informáciu považujeme za veľmi dôležitú, keďže výsledkom akejkoľvek transkripcie majú byť čo najdetailnejšie informácie o hre na hudobné nástroje.

Annotation

Slovak University of Technology Bratislava
FACULTY OF INFORMATICS AND INFORMATION TECHNOLOGIES
Degree Course: Informatics
Author: Martin Žák
Master thesis: **Modelling features of musical audio signals using machine learning algorithms**
Supervisor: Ing. Lukáš Marták
2021, May

Music has been a phenomenon in society for many millennia. Authors of classics used to manually notate their well-known musical compositions. They did it to preserve them for future generations. However, manual notating, respectively transcription, is not trivial task. It has always been and still is cognitively demanding process, which is intended either for the authors of the compositions or for people with excellent hearing capabilities.

Nowadays, when a huge number of popular songs of various genres are being created, not every author care about doing proper music transcription. Because of that songs with multiple musical instruments are much more difficult to reproduce. Though thanks to the great computation power of computers and the emergence of advanced machine learning algorithms, there has been an effort in the last few decades to automate whole transcription process with the aim to increase degree of reproducibility. Since the music transcription is a very complex task, current state-of-the art methods solve this problem at the level of musical instruments.

In this work, we focus on automatic drum transcription. We propose 3 methods, two of which incorporate more detailed information about instrument playing techniques into transcription, which is often lacking in state-of-the-art methods.

Obsah

Slovník pojmov a skratiek

ADT – Automatická transkripcia bicích nástrojov (angl. *Automatic drum transcription*)

AMT – Automatická transkripcia hudby (angl. *Automatic music transcription*)

MIR – Získavanie hudobných informácii (angl. *Music Information Retrieval*)

DSC – Klasifikácia zvukov bicích nástrojov (angl. *Drum Sound Classification*)

DTC – Klasifikácia bubenických hracích techník (angl. *Drum Technique Classification*)

DTD – Transkripcia bicích nástrojov z nahrávok polyfonickej hry na bicie nástroje (angl. *Drum Transcription of Drum-only Recordings*)

DTP – Transkripcia bicích nástrojov z nahrávok polyfonickej hry na bicie nástroje a ďalších rytmických nástrojov (angl. *Drum Transcription in the Presence of Additional Percussion*)

DTM – Transkripcia bicích nástrojov z nahrávok polyfonickej hry na bicie nástroje za prítomnosti melodických nástrojov (angl. *Drum Transcription in the Presence of Melodic Instruments*)

KD – úder kopákom (angl. *Kick drum*)

SD – úder na rytmičák (angl. *Snare Drum*)

SDB - úder na rytmičák pomocou metličiek (angl. *snare drum brush hit*)

SDF – úder na rytmičák oboma paličkami takmer naraz – *flam* technika (angl. s*nare drum flam hit*)

SDG – veľmi slabý úder na rytmičák, ktorý však má význam v rytme(angl. *snare drum ghost hit,* alebo *ghost note*)

SST/SDST – úder o ráfik (angl. *snare side stick*)

SDNS – úder na rytmičák, ktorý má povolené struny (angl. *snare drum no wires*)

HT – úder na vysoký tom (angl. *High Tom*)

MT/MHT – úder na stredný tom (angl. *Mid Tom*)

LT/LFT – úder na kotol (angl. *Low Tom*)

HH – úder na hi-hat činel (angl. *Hi-hat)*

CHH – úder na zavretý hi-hat činel (angl. *Closed hi-hat*)

PHH – Hi-hat činel stlačený pedálom (angl. *Pedal hi-hat*)

OHH – úder na otvorený hi-hat činel (angl. *Opened hi-hat*)

CC/CRC – úder na *Crash činelu* (angl. *Crash Cymbal*)

RC/RDC – úder na *Ride* činelu (angl. *Ride Cymbal*)

RDB – úder na vnútornú časť *Ride* činely (angl. *Ride Bell*)

CHC – úder na *China* činelu

TMB – úder na tamburínu

RNN – Rekurentné neurónové siete (angl. *Recurrent neural networks*)

BDRNN – Obojsmerná rekurentná neurónová sieť (angl. *Bidirectional recurrent neural network*)

LSTM RNN – Rekurentné neurónové siete s dlhou krátkodobou pamäťou (angl. *Long Short Term Memory recurrent neural networks*)

GRU RNN – Rekurentné neurónové siete s uzavretými opakujúcimi sa jednotkami (angl. *Gated recurrent unit recurrent neural networks*)

DFT – Diskrétna Fourierova transformácia (angl. *Discrete Fourier transformation*)

STFT – Krátkodobá Fourierova transformácia (angl. *Short time fourier transform*)

NMF – Nezáporná maticová faktorizácia (angl. *Non-negative matrix factorization)*

PFNMF – NMF s čiasotočne fixnými bázami (angl. *Partially-fixed non-negative matrix factorization*)

NMFD – Dekonvolutívna NMF (angl. *Non-negative matrix factorization deconvolution*)

MU – multiplikatívne aktualizácie (angl. *multiplicative update rules*)

PG – projektovaný zostupujúci gradient (angl. *projected gradient descent*)

1. Úvod

Automatická transkripcia hudby (angl. *Automatic Music Transcription*, ďalej už len AMT), resp. automatický prepis hudobného záznamu do notového zápisu, je veľmi náročnou úlohou v rámci výskumných skupín, ktoré sa zaoberajú spracovávaním signálu a umelou inteligenciou. Skladá sa z niekoľkých čiastkových úloh ako rozpoznanie hudobného nástroja, detekcia začiatku nástupu noty (angl. *onset detection*), koniec noty (angl. *offset detection*), odhad súčasne zahraných nôt v čase, rozpoznanie dynamiky a mnoho iných [1].

Aby automatická transkripcia mohla plnohodnotne nahradiť manuálnu transkripciu, jej výstupom musí byť detailná, vizuálna charakteristika hudby v podobe notového zápisu, ktorá zachytí všetky aspekty skladby do takej miery, aby bola reprodukovateľná. Tým sa myslia napríklad zahrané tóny, technika hrania, notový kľúč, ktorý spolu s umiestnením noty v notovej osnove definuje výšku tónu [2], dĺžka tónu, akord, ktorý je súčasťou harmónie, tempo, ktoré určuje rýchlosť skladby, dynamika, v akej je sekvencia nôt zahraná alebo takt, ktorý určuje pravidelné striedanie prízvučných a neprízvučných dôb.

Hoci už len kvalitné zautomatizovanie samotného procesu transkripcie hudby je výzvou pre súčasný výskum, dôležitým prvkom, ktorý celú úlohu sťažuje, je hudba, resp. zvuk ako taký. V princípe každý zvuk, ktorý človek počuje, sa šíri na určitej úrovni frekvencie, z ktorej vieme následne určiť tón. Ak v čase znie len jeden tón, hovoríme o monofónii. To pre súčasný výskum nepredstavuje značný problém. Ten nastáva až pri polyfónií, kedy v čase znie viacero rôznych tónov naraz. V takomto prípade je náročné nielen pre pokročilé algoritmy ale aj pre človeka určiť, ktoré noty boli práve zahrané a ktorý tón prislúcha akému hudobnému nástroju.

Súčasťou polyfonickej hudby je vždy rytmická a melodická zložka. Rytmickú zložku tvoria rytmické nástroje ako napríklad hrkálky, tamburíny či bicie nástroje. Ich úlohou je dodržať tempo, určiť rytmus a zároveň štýl skladby alebo zvýrazniť prízvučné a neprízvučné doby. Melodickú zložku tvoria melodické nástroje ako napríklad klavír, gitara, husle, alebo spev. Tie tvoria melódiu skladby, ktoré sú častokrát podporené rôznymi konsonantnými (ľubozvučné) alebo disonantnými (neľubozvučné) akordami. Vďaka melódií vie potom človek rozoznávať skladby, ktoré počuje.

V rámci výskumnej oblasti s názvom *získavanie hudobných informácii*[1], ktorá sa už roky zaoberá problémami spracovávania akustickej hudby[2], sa prevažne rieši transkripcia melodických nástrojov. Téma automatickej transkripcie bicích nástrojov zaostáva. Aby sa niekedy v budúcnosti robila automatická transkripcia hudby správne, transkripcia melodických nástrojov a bicích nástrojov musí byť na rovnakej úrovni. Vzhľadom na aktuálny výskum v tejto oblasti, kedy sa zatiaľ sa nepodarilo nájsť komplexný spôsob, ktorý by riešil obidve zložky spoločne, automatická transkripcia bicích nástrojov si vyžaduje osobitnú pozornosť.

Existujúcich riešení na automatickú transkripciu rytmických, špecificky bicích nástrojov, je pomenej a aj tie čo sú, majú stále rezervy v presnosti transkripcie [3]. Preto tu vidíme priestor na výskum.

1.1. Štruktúra dokumentu

Štruktúra diplomovej práce je nasledovná. Vo zvyšných pod sekciách kapitoly 1 bližšie poviem o svojej motivácií, prečo som sa rozhodol riešiť práve problém automatickej transkripcie bicích nástrojov, ktorú ďalej budem spomínať pod skratkou ADT (angl. *Automatic Drum Transcription*). Špecifikujem danú problematiku a objasním význam bicích nástrojov v hudbe a aj význam jednotlivých komponentov bicej sady.

V sekcii 2 sa budem venovať digitálnemu spracovaniu audio signálu a vysvetlím základné koncepty, ktoré sú často používané v aktuálne najlepších prístupoch riešiacich problémy ADT.

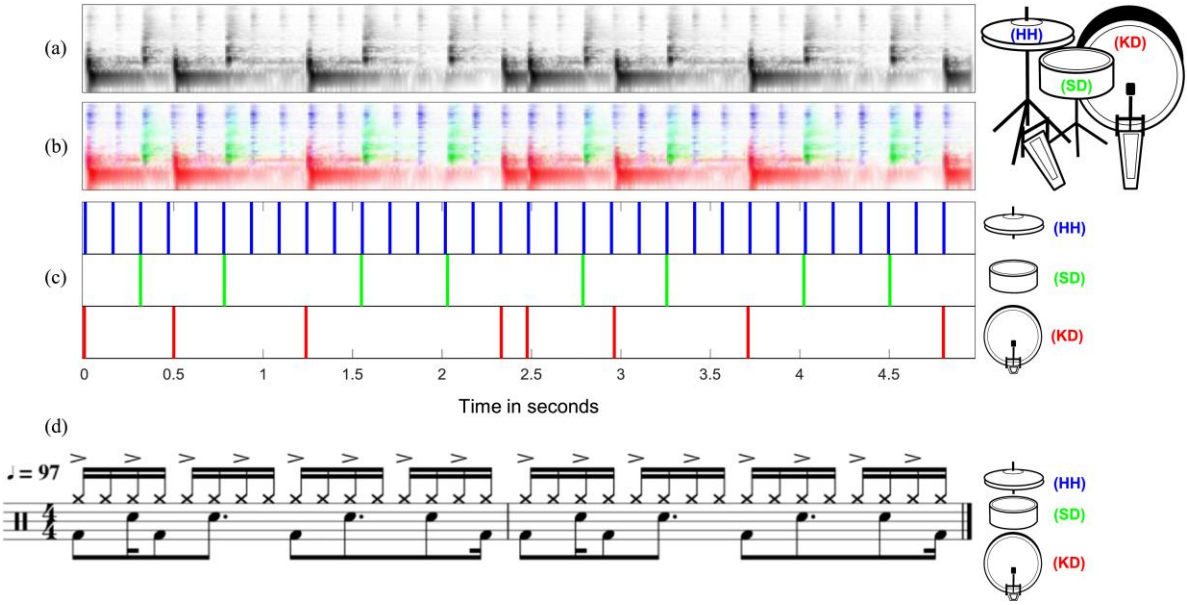
V sekcii 3 a 4 podrobnejšie rozoberiem 2 najmodernejšie prístupy, ktoré riešia problémy ADT. Nimi sú nezáporná maticová faktorizácia a hlboké neurónové siete.

V sekcii 5 poviem bližšie o aktuálnom stave v oblasti automatickej transkripcie bicích nástrojov. Rozoberiem aktuálne najmodernejšie prístupy, porovnám a pomenujem ich výhody a nevýhody. Na záver uvediem niekoľko výskumných problémov a naznačím, ktorému z nich sa budem v práci ďalej venovať. V sekcii 6 uvádzam návrhy rôznych vylepšení. V kapitole 7 opíšem, čo všetko som použil pri implementácií. V kapitole 8 vyhodnotím nami navrhnuté algoritmy a v kapitole 9 zhrniem nadobudnuté poznatky a ďalšie smerovanie práce do budúcna.

[1] Music Information Retrieval (MIR)
[2] https://www.music-ir.org/mirex/wiki/2019:Main_Page

1.2. Automatická transkripcia bicích nástrojov

ADT je jedným z podproblémov AMT. Úlohy v ADT sa zameriavajú na extrakciu tých istých čŕt ako v AMT. Teda tiež sa zaoberajú notami, dynamikou, tempom, taktami, technikou hrania. No v niektorých prípadoch však majú iný význam a zdá sa, že ADT je zjednodušený prípad AMT. Napríklad niektoré bicie sú naladené na určitý tón (konkrétne membránofóny, viac rozoberiem v sekcii 1.4). A teda problém detekcie tónu vieme pretransformovať na problém detekcie rytmického hudobného nástroja. Alebo dĺžka tónu nie je až tak dôležitá ako pri melodických nástrojoch. Je to hlavne preto, lebo rytmus sa vyjadruje skôr diskrétnymi udalosťami ako kontinuálnymi, ktoré sú záležitosťou melodických nástrojov.



Obrázok 1. Ilustrácia ADT len bicích nástrojov. a) Zvuk bicích vo forme spektrogramu. Tmavšie odtiene sivej charakterizujú vyššiu energiu. b) Rovnaká reprezentácia vo forme spektrogramu s ofarbenými príspevkami jednotlivých bicích nástrojov. c) Začiatky nástupov tónov sú zobrazené vo forme diskrétnych udalostí. d) Zápis aktivity bicích do notového záznamu. [3]

Na vyššie uvedenom obrázku si môžeme všimnúť, že rôzne udalosti hry na bicích sa prekrývajú v čase a aj vo frekvenčných pásmach, ktoré sú vyjadrené v spektrogame (viac rozoberiem v kapitole 2). Preto už len transkripcia bubnov pri polyfonickej hre na bicích bez prítomnosti iných nástrojov predstavuje náročnú úlohu [3].

Ako sme už vyššie uviedli, jednou z charakteristík bicích je zvýrazňovanie rytmu skladby. To je kvôli periodickej a opakovanej hre na jednotlivé bicie nástroje. Táto znalosť môže byť využitá pri metódach, ktoré vyplývajúc už zo svojej podstaty vedia zachytávať a prípadne uchovávať tieto periodické charakteristiky [3].

Výskumné problémy v oblasti ADT sa delia do rôznych kategórií podľa toho, na čo sa zameriavajú. Môže to byť obyčajná klasifikácia zvukov bicích (DSC – angl. *Drum Sound Classification*), hracej techniky (DTC – angl. *Drum Technique Classification*) alebo transkripcia samostatne znejúcich bicích (DTD – angl. *Drum Transcription of Drum-only Recordings*), za prítomnosti ďalších rytmických nástrojov (DTP – angl. *Drum Transcription in the Presence of Additional Percussion*) alebo melodických nástrojov (DTM – angl. *Drum Transcription in the Presence of Melodic Instruments*) [3]. Posledná z nich, DTM, je všetkým ľuďom asi najbližšia, pretože je to najčastejší prípad v produkovanej populárnej hudbe, ktorá disponuje melodickými pasážami doprevádzanými výrazným rytmom. Vyriešenie tohto problému by určite veľmi rýchlo našlo uplatnenie v praxi.

1.3. Motivácia

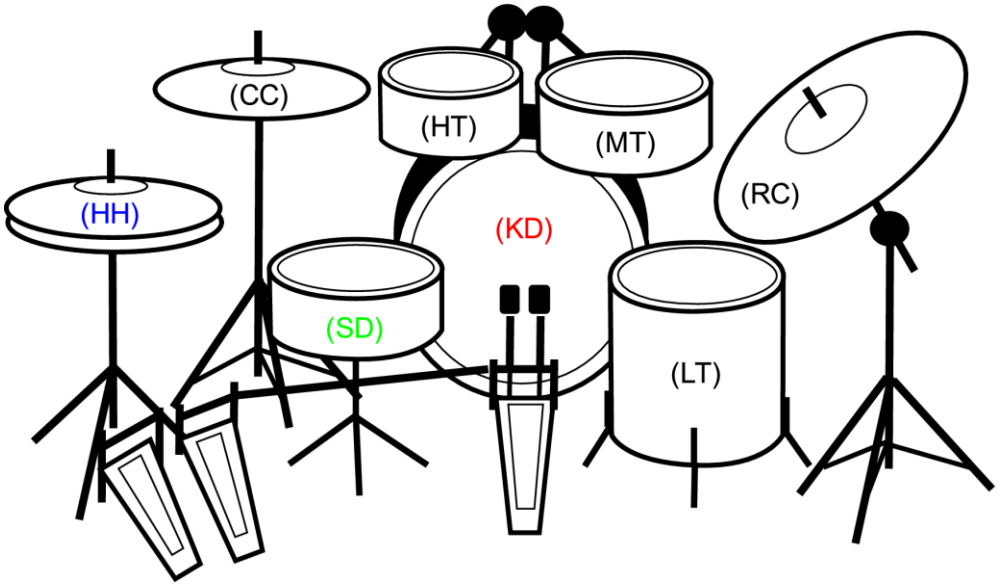
Využitie automatickej transkripcie bicích nástrojov je rozsiahle. Mohlo by byť súčasťou hudobnej náuky, v ktorej by softvér učil hru na bicie nástroje za použitia gemifikácie, čím by sa cvičenie na nich stalo atraktívnejším. Samozrejme toto je aplikovateľné na ľubovoľný hudobný nástroj. Súčasťou transkripcie bicích, ktoré znejú v polyfonickej hudbe, je ich separácia od ostatných nástrojov. Toto by sa dalo využiť pri štúdiovom nahrávaní bicích nástrojov, ktoré si vyžadujú veľké množstvo mikrofónov, kde každý sníma buď jeden špecifický bubon, činelu alebo menšiu podskupinu nástrojov. Problémom je, že jeden mikrofón zachytáva viacero bubnov. ADT by riešilo tento problém filtrovaním nežiadúcich zvukov bubnov na zvolených kanáloch. Separácia jednotlivých bicích ale aj všeobecne rôznych hudobných nástrojov by taktiež umožnila DJ-om alebo hudobným producentom meniť vlastnosti hudobných inštrumentov samostatne. Mohlo by sa zlepšiť aj vyhľadávanie hudby ak by sa automatickým spôsobom vyextrahoval rytmus, ktorý zadefinuje *groove* (viac rozoberiem v sekcii 1.4) a nájde skladby podobného žánru [3]. Okrem praktického využitia mám k téme aj osobný vzťah. Keďže sa od detstva venujem hudbe, konkrétne hre na klavíri a na bicích nástrojoch, o to väčšiu mám chuť zaoberať sa problémami spadajúcich do tejto oblasti. Navyše je tu možnosť zapojiť sa do každoročne organizovanej súťaže Music Information Retrieval Evaluation eXchange (MIREX), ktorá rieši rôzne problémy v už spomínanej výskumnej oblasti MIR. Jednou z nich je aj tzv. *drum transcription*.

1.4. Bicie nástroje ako rytmická zložka hudby

Ako sme už spomínali v úvode, polyfonická populárna, alebo alternatívna hudba sa skladá z melodickej a rytmickej zložky. Bicie nástroje zastupujú práve rytmickú zložku. Okrem toho, že udržiavajú tempo, udávajú rytmus, zvýrazňujú prízvučné a neprízvučné doby,

definujú aj *groove*. Je to hudobný prvok, ktorý vychádza z rytmu a častokrát si jeho prítomnosť uvedomíme až vtedy, keď pri opakovanom rytme v hranej skladbe zachytíme jeho charakter a začneme si ho v hlave opakovať, pohmkávať alebo vyklopkávať. Táto zložka nám vie napovedať niečo o štýle hudby, v akom je zahraná.

V zvyšku tejto práce budem často hovoriť o konkrétnych nástrojoch bicej sady. Aby sme boli v obraze, ktoré nástroje budem mať na mysli, teraz si ich rozoberieme na základnej úrovni a ďalej budeme operovať už iba s ich skratkami, ktoré sú vysvetlené v úvode práce v slovníku pojmov a skratiek.



Obrázok 2. Najdôležitejšie nástroje bicej sady používané v populárnej hudbe [3]

Bicie nástroje delíme do dvoch skupín: Blanozvučné (membranofóny) a samozvučné (idiofóny). Blanozvučné majú telo, ktoré je poväčšine z dreva a 2 blany – vrchnú a spodnú. Väčšinou sa udiera len po vrchnej. Medzi ne patrí kopák (KD), rytmičák (SD), všetky druhy prechodov, ktorými sú napríklad vysoký tom (HT), stredný tom (MT) a kotol (LT). Na samozvučné sa udiera priamo po tele nástroja, ktoré je aj zdrojom a zosilňovačom zvuku. Medzi ne patrí hi-hat činel (HH), crash činel (CC), ride činel (RC) a iné [3, 4]. Umiestnenie všetkých bubnov a činelov si môžeme na základe uvedených skratiek pozrieť na obrázku 2. Najdôležitejšími bicími nástrojmi, na ktorých je tvorená väčšina rytmov, sú rytmičák, kopák a hi-hat činel. Tie si preto teraz trochu bližšie ozrejmime.

Kopák (KD) je najväčší spomedzi všetkých bubnov. V súprave sa na neho hrá kladivkom pomocou kladky a pedálu. Má hlbokú a zároveň neurčitú výšku tónu. Zvyčajne sa jeho priemer pohybuje v rozmedzí 20 až 22 palcov. Niekedy sa medzi blany dovnútra bubna umiestni

tlmiaci materiál vo forme deky alebo vankúša, ktoré redukujú rezonanciu bubna a znižujú jeho hlasitosť [5, 6]. V hudbe je jeho úlohou zvýrazňovať prízvučné doby.

Rytmičák (SD) je jeden z najdôležitejších bubnov bicej sady. Hrá sa na ňom úderom paličky po blane. Má špecificky zvuk, pretože na jeho spodnej blane je pripnutá struna. Tú môžeme páčkou na bočnej strane povoliť. Potom rytmičak stratí svoj špecifický zvuk a bude znieť ako prechodový bubon. Pri hraní paličkou môžeme udierať na blanu, obruč, obal, kombinovať viaceré miesta úderu naraz, čím vytvoríme viacero odlišných zvukov. Zvyčajne má priemer okolo 14 palcov [6]. Rytmičák spolu s kopákom sú elementárnou jednotkou jednoduchých aj zložitých rytmov. Okrem zvýrazňovania prízvučných dôb je jeho hlavnou úlohou tvoriť základný charakter rytmu.

Hajtka (HH) je najpoužívanejším činelom zo súpravy. Skladá sa z dvoch činelov umiestnených proti sebe na špeciálnom stojane. Spodný činel je pripevnený na stojan, vrchný je pripevnený na tyčku, ktorá sa ovláda pedálom. Vďaka tejto koštrukcii sa hi-hat činel môže otvoriť a zatvoriť, čo umožňuje vytvoriť viacero rôznych zvukov pomocou pedálu a údermi paličiek [6]. Hi-hat činel je neoddeliteľnou súčasťou bicích, pretože okrem toho, že slúži na výplň rytmu, je vynikajúcou pomôckou pre bubeníkov na udržanie pravidelného rytmu.

Vo všeobecnosti sú bicie nástroje odlišné od melodických. Hoci vydávajú konkrétne tóny, ľudské ucho ich nevie tak ľahko rozoznať. A to hlavne preto, lebo úderom paličky na bubon sa vytvára prechodný zvuk, ktorý pokrýva široké spektrum frekvencii podobné šumu. Tonálne prvky sú v ňom nie tak výrazné, čo potom spôsobuje neidentifikovateľnosť tónu. Kvôli týmto charakteristikám rôzne algoritmy, ktoré sú určené pre melodické nástroje, nie sú aplikovateľné pre ADT úlohy [3].

1.5. Celkový pohľad na proces automatickej transkripcie bicích nástrojov

Keď sme si už povedali niečo o bicích nástrojoch, akú rolu zohrávajú v hudbe a prečo sa samostatne rieši problém automatickej transkripcie bicích nástrojov, môžeme zľahka načrtnúť proces automatickej transkripcie bicích, ktorý rozvinieme v nasledujúcich častiach.

Na to, aby sme z hudobnej nahrávky dostali notový zápis, v prvom rade sa musí audio signál pretransformovať do podoby s vhodnou reprezentáciou, ktorá zvýrazní dôležité črty hudby (podrobnejšie rozoberiem v časti 2). Takáto reprezentácia môže byť ešte iným spôsobom spracovaná alebo rovno použitá ako vstup do algoritmov, ktorých výstupom je detekcia

zahraných nôt, v našom prípade bubnov, s potrebnými atribútmi a ich zápis do notového záznamu. Na detekciu bicích nástrojov sa aplikovala široká paleta metód od primitívnych prístupov až po zložitejšie algoritmy strojového učenia. Jednou z najpopulárnejších metód strojového učenia, ktoré v doméne ADT aktuálne vykazujú veľmi dobré výsledky, pričom aj veľké technologické spoločnosti ako Google či Microsoft sú ochotné investovať veľké množstvo peňazí do výskumu tejto oblasti, sú hlboké neurónové siete. Okrem nich sa pri riešení problémov automatickej transkripcie hudby ujal aj o niečo jednoduchší spôsob, ktorým je nezáporná maticová faktorizácia. Tieto prístupy podrobne rozoberiem v kapitolách 3 a 4.

2. Digitálne spracovanie signálu

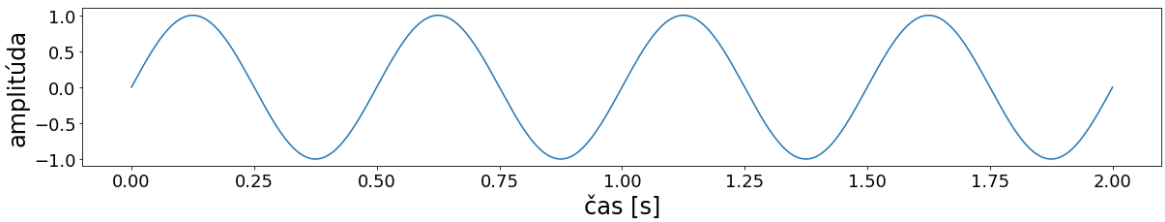
Väčšina ľudí, okrem tých čo sú sluchovo postihnutí, dennodenne prichádza do kontaktu so zvukom v rôznej podobe. Či sa jedná o hudbu, ktorú počujeme naživo alebo cez audioprehrávač, ľudskú hovorenú reč, ľubovoľné výstražné znamenia, alebo obyčajný šum, ktorý počujeme v tichu, vo všetkých prípadoch ľudské ucho reprezentuje získané informácie ako zvuk.

Zvuk je mechanický typ vlnenia. To znamená, že ide o kmitavý pohyb, ktorý prenáša energiu v určitom smere [7]. Akýkoľvek zvuk ľudské ucho spracováva pomocou membrány, tzv. *ušného bubienka*, ktorá sa rozkmitá primárne na základe jeho farby, hlasitosti a výšky. Tieto tri veličiny sa v čase môžu meniť. Vibrácie ušného bubienka sú ďalej prešírené cez nervy prostredníctvom elektrických impulzov až do mozgu, ktorý interpretuje prijaté informácie ako zvuk, vďaka čomu môžeme počuť [8].

Keďže ľudské ucho disponuje pomerne zložitým mechanizmom transformácie akustického signálu do podoby, ktorý je zrozumiteľný pre mozog, samozrejme aj počítače musia disponovať aparátom, ktorý spracuje akustický analógový signál do formy, ktorá je vhodná pre digitálny svet. Na to slúži analógovo-digitálny prevodník na zvukovej karte, ktorého funkciu si priblížime v sekcii 2.1.1.

2.1. Digitalizácia audio signálu

Audio signál je v analógovom svete spojitá funkcia, ktorá má tvar vlnenia. V prípade, že ide o rozpoznateľný zvuk alebo tón, toto vlnenie, resp. zvukový signál je periodický s určitou frekvenciou [7].



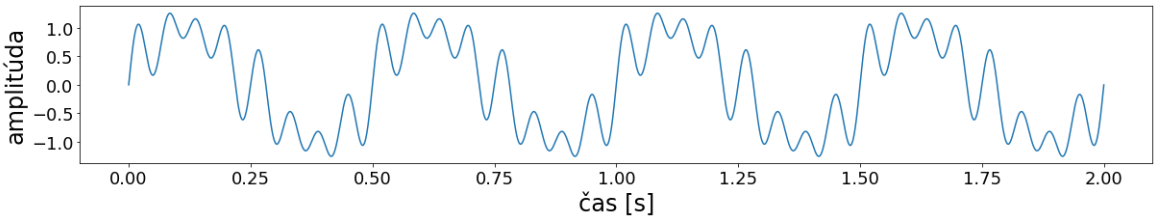
Obrázok 3. Ilustračný príklad jednoduchého periodického signálu

Na vyššie uvedenom obrázku môžeme vidieť jednoduchý zvukový signál, ktorý je periodický. Periodicitu môžeme spozorovať vďaka opakujúcej sa sínusovej vlne v čase. Tá má svoju frekvenciu (konkrétne 2Hz, pretože sa vykonajú 2 kmity za 1 sekundu), ktorá definuje výšku tónu a amplitúdu (tzv. rozdiel od rovnovážnej polohy, resp. veľkosť vlny), ktorá hovorí o hlasitosti. Matematické vyjadrenie vlnenia v čase je nasledovné:

$$y = y_m \sin(\omega t) \quad (1)$$

ω je uhlová rýchlosť, ktorá je vyjadrená frekvenciou, t je čas, v ktorom chceme pozorovať veľkosť amplitúdy y a y_m je maximálna amplitúda vlnenia.

Na vyššie uvedenom obrázku (Obrázok 3) vidíme jednoduchý signál (obyčajnú sínusoidu), s ktorým sa v reálnom svete málokedy stretneme. Keď počujeme znejúci tón hudobného nástroja, vlnenie je stále periodické no samotná vlna je komplexnejšia. Je to tak preto, lebo tón je tvorený ako sme už spomenuli hlasitosťou, výškou, farbou [7] a vzniká superpozíciou viacerých harmonických frekvencií. Takéto vlnenie môže vyzerať nasledovne.



Obrázok 4. Ilustračný príklad komplexného periodického signálu

Na tomto obrázku (Obrázok 4) môžeme vidieť, že periodicita je zachovaná no charakter vlny je oproti pôvodnému obrázku zložitejší. O tom, ako ďalej pracovať s takto komplexným signálom si opíšeme bližšie v podkapitole o Diskrétnej Fourierovej transformácií.

2.1.1. Vzorkovanie

Keďže počítače vedia pracovať len s diskrétnymi hodnotami, úlohou analógovo-digitálneho prevodníka je aproximovať spojitú funkciu (komplexný zvukový signál, napr. na obrázku 4) do diskrétnej podoby tak, aby diskrétna reprezentácia signálu bola čo najpresnejšia. Toto je možné spraviť pomocou vzorkovania s vhodne nastavenou vzorkovacou frekvenciou.

Vzorkovanie v kontexte digitalizácie audio signálu je proces, v ktorom sa zachovávajú len niektoré hodnoty spojitej vlny zvukového signálu. To, ako často sa majú uchovávať tieto hodnoty určuje vzorkovacia frekvencia, ktorá definuje počet odobratých vzoriek zo spojitého signálu za sekundu. Pri tomto prirodzene nastáva otázka, ako optimálne nastaviť hodnotu vzorkovacej frekvencie, aby sme čo najlepšie aproximovali spojitú funkciu? Nyquist-Shannon vzorkovací teorém (angl. *Nyquist-Shannon sampling theorem*) rieši tento problém a hovorí, že na perfektné zrekonštruovanie signálu je potrebný minimálne 2-násobok maximálnej frekvencie, ktorá sa nachádza vo zvukovom signále. Keďže ľudské ucho je schopné rozlišovať zvuky vo frekvenčnom rozsahu 20 – 20 000 Hz, zvyčajne je táto frekvencia nastavená aspoň na hodnotu 40 000Hz (40 kHz) [9].

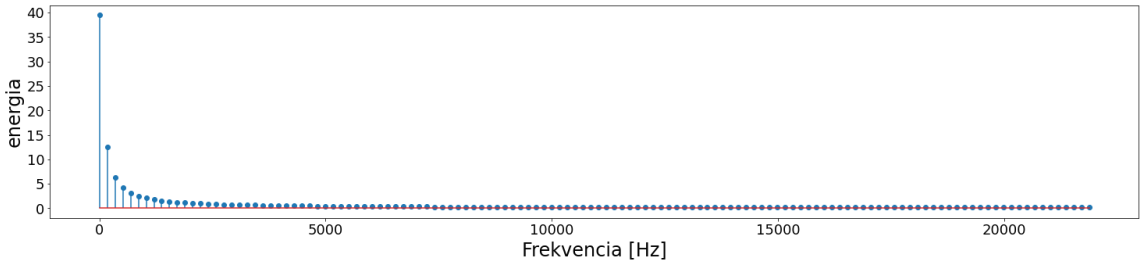
2.2. Diskrétna Fourierova transformácia

Keďže zvukový signál je zmena akustického tlaku v čase, ktorá vykazuje svoju periodicitu, vieme ho popísať zmenou amplitúdy v čase. Vtedy hovoríme, že signál je v časovej doméne. Môže byť jednoduchý (viď Obrázok 3) alebo komplexný, teda pozostávajúci z viacerých frekvenčných komponentov (viď Obrázok 4). Fourierova transformácia je mechanizmus, ktorý transformuje časovú doménu signálu na frekvenčnú [10]. Zjednodušene povedané, Fourierova transformácia zisťuje, ktoré frekvenčné komponenty, resp. frekvencie s akou intenzitou sa nachádzajú vo vymedzenom úseku zvukového signálu. Treba podotknúť, že Fourierová transformácia sa aplikuje na spojitý signál, no počítače pracujú s diskrétnymi hodnotami, preto existuje jej varianta – Diskrétna Fourierova transformácia. Matematické vyjadrenie Diskrétnej Fourierovej transformácie (ďalej už len DFT) je nasledovné

$$DFT[k] = \sum_{n=0}^{N-1} x[n] \,.\, e^{-\varphi i} \tag{2}$$

$$kde\ \varphi = k\frac{n}{N}2\pi \tag{3}$$

Ak by sme aplikovali DFT na časť komplexného signálu (viď Obrázok 4), výsledok bude nasledovný.



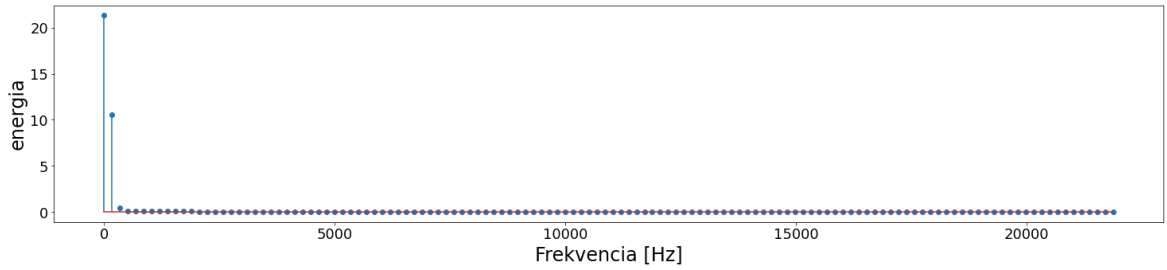
Obrázok 5. Vizualizácia diskrétnej Fourierovej transformácie časti komplexného signálu

Aktivita frekvenčných rozsahov (angl. *frequency bins*) Fourierovho spektra približne zodpovedá intenzite a frekvenciám signálov, z ktorých bol ilustračný komplexný signál vytvorený. Problémom je to, že bol zložený z troch jednoduchých signálov, pričom prvý signál má frekvenciu 2 Hz (Obrázok 3) a ďalšie sú voči nemu harmonické frekvencie. Na vyššie uvedenom obrázku (Obrázok 5) však vidíme aktivitu v oveľa väčšom počte frekvenčných rozsahov. Príčinou problému je to, že DFT je aplikovaná len na časť signálu a v prípade, že začiatok alebo koniec signálu daného výseku nezačína v neutrálnej polohe (amplitúda = 0), môžeme sledovať tzv. *Gibbsov jav*. Totižto náhla zmena v amplitúde analyzovaného úseku signálu, ktorú by bolo teoreticky možné pri Fourierovom rozklade rozložiť na nekonečné frekvenčné spektrum, sa za súčasných podmienok, kedy pracujeme s konečným

frekvenčným rozsahom, ťažko replikuje [11]. Táto neschopnosť spôsobuje spomínaný *Gibbsov jav* a preto je na obrázku 5 aktívnych viacero frekvenčných rozsahov.

2.2.1. Okienková funkcia

Na potlačenie *Gibbsovho javu* sa ešte pred DFT zvyčajne aplikuje na signál tzv. *okienková funkcia* (tzv. *window function*), ktorá zabezpečí to, aby sa v signále nevyskytovala náhla zmena. Existuje viacero typov okienok, napr. obdĺžnikové okno (angl. *rectangular window*), Hammingove okienko (angl. *hamming window*) a iné [10]. Napríklad po aplikovaní Hammingovho okienka na signál z obrázka 4 bude výsledok DFT vyzerať nasledovne



Obrázok 6. Vizualizácia diskrétnej Fourierovej transformácie po aplikovaní Hammingovho okienka na časť komplexného signálu

Na vyššie uvedenom obrázku môžeme vidieť, že aktívne frekvenčné rozsahy zodpovedajú frekvenciám, z ktorých bol komplexný signál vytvorený a teda Gibbsov jav bol potlačený.

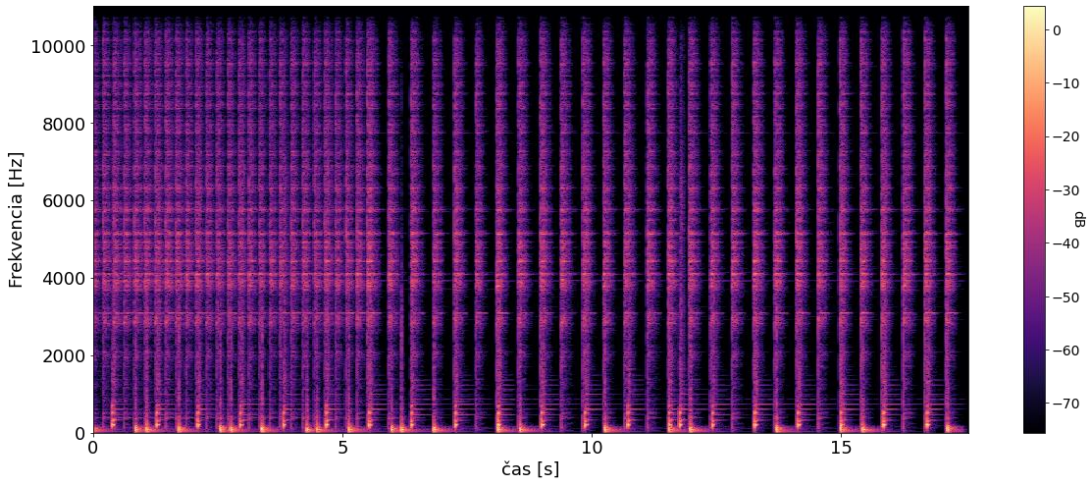
2.3. STFT spektrogram

V doméne hudby skladby trvajú poväčšine niekoľko minút. Vieme, že hudobníci často menia techniku hry na nástroji, ktorá je prípadne prerušovaná pauzami. Ak by sme aplikovali DFT na celý audio signál skladby, hoci by sme získali zastúpenie aktivity rôznych frekvenčných rozsahov v skladbe (niečo podobné ako na obrázku 6, no samozrejme s aktivitou viacerých frekvenčných rozsahov) unikala by nám jedna dôležitá informácia, ktorou je zmena tejto aktivity v čase. To v princípe znamená, že by sme nemali informáciu o tom, ako sa vyvíjala hra na jednotlivé hudobné nástroje naprieč skladbou. Navyše tu je ešte ten problém, že v prípade dlhej skladby je otázne, či by sa na bežnom počítači vôbec vykonala DFT nad celou skladbou vzhľadom na jej zvyšujúcu sa pamäťovú a časovú náročnosť pri dlhšom signále.

Limitom DFT je teda to, že nedisponuje informáciou vývoja frekvenčných rozsahov v čase. Na vyriešenie tohto problému sa používa tzv. *Krátkodobá Fourierova transformácia* (angl. *Short-time Fourier transform*, ďalej STFT), čo je spôsob aplikácie DFT na signál. Pri STFT sa aplikuje vždy len na výňatok signálu, no toto okienko výňatku sa posúva s určitým

prekryvom naprieč časom celou skladbou, čim sa vypočíta viacero DFT vždy pre iný, krátky časový úsek. Výsledkom je tzv. STFT spektrogram pričom jeden segment, resp. jedno fourierove spektrum v čase sa nazýva rámik.

Veľkosť prekryvu okienka dvoch za sebou nasledujúcich úsekov, z ktorých sa robí DFT, sa nazýva skok (angl. *hop*) a okrem veľkosti posuvného okienka je jedným z parametrov, ktorý značne ovplyvňuje výkonnosť STFT [10]. Tak ako veľkosť posuvného okienka ovplyvňuje to, aké detailné rozlíšenie bude mať frekvenčné spektrum, tak isto veľkosť skoku ovplyvňuje rozlíšenie v časovej doméne. Aby sme si lepšie vedeli predstaviť spektrogram, tak ak by sme aplikovali STFT na nejaký audio signál s polyfonicky znejúcou hrou bicích nástrojov (hra na bicie pričom v každom čase môže znieť viacero bicích nástrojov naraz), vizualizácia spektrogramu by bola nasledovná



Obrázok 7. Spektrogram nahrávky, v ktorej znejú len bicie nástroje

Na vyššie uvedenom obrázku (Obrázok 7) môžeme vidieť STFT spektrogram nahrávky bubenického sóla. Vertikálna os reprezentuje frekvenčné spektrum v *hertzoch*, horizontálna časovú doménu v *sekundách* a farby v spektrograme znázorňuju magnitúdy jednotlivých frekvenčných rozsahov v čase podľa farebnej lišty, ktorá je vpravo. Jej miera je znázornená v *decibeloch*, čo je merná jednotka hlasitosti.

2.4. Zhrnutie

V tejto kapitole sme si vysvetlili postup digitalizácie audio signálu. Priblížili sme význam kľúčového mechanizmu vzorkovania v analógovo-digitálnom prevodníku na zvukovej karte, ktorý je zodpovedný za aproximáciu zvukovej vlny, ktorá je v analógovom svete spojitá, no v digitálnom musí jej diskrétny priebeh zodpovedať čo najlepšie spojitému. S takto pretransformovaným signálom môžu potom počítače ďalej pracovať. Taktiež sme opísali

postup, ako z diskrétneho audio signálu vyextrahovať užitočné informácie o frekvenciách pomocou DFT a jej variantu STFT, ktorého výsledkom je STFT spektrogram. Existuje viacero typov spektrogramov, no pre účely tejto práce nie je potrebné ich rozoberať.

3. Hlboké neurónové siete

Hlboké neurónové siete sa v posledných rokoch stali veľmi populárnou metódou strojového učenia. Ich využitie môžeme vidieť všade tam, kde sa pracuje s veľkým množstvom informácii a na ktoré sa dajú aplikovať rôzne prístupy dolovania dát. Nimi sú napríklad regresia, klasifikácia alebo zhlukovanie. Neurónové siete zožali úspech aj preto, lebo vysoký výpočtový výkon, ktorý začali dosahovať grafické karty, už viac nepredstavoval problém pre zložité maticové operácie, ktoré sú často využívané práve pri neurónových sieťach.

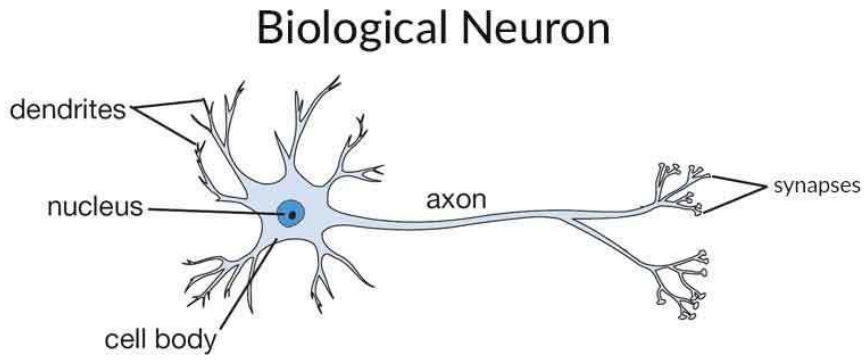
V zásade neurónové siete robia to isté, čo ostatné algoritmy strojového učenia. Hľadajú parametre matematickej funkcie, ktorou sa snažia čo najlepšie opísať všetky pozorovania. Ako príklad uvediem jednoduchú lineárnu regresiu [12].

$$y = a.x + b \tag{4}$$

x vyjadruje vstupné hodnoty, ***a*** definuje sklon krivky a ***b*** vyjadruje posun krivky na osi x. Algoritmy strojového učenia, a teda aj neurónové siete, sa snažia nájsť najoptimálnejšie hodnoty parametrov *a,b* [12]. Odlišujú sa spôsobom, akým tieto parametre hľadajú. To, čím boli inšpirované neurónové siete, na akom princípe fungujú, čo všetko sa dá spraviť pre ich optimalizáciu a aké sú najpopulárnejšie architektúry neurónových sietí v oblasti ADT si rozoberieme v nasledujúcich častiach.

3.1. História

O neurónových sieťach sme prvýkrát počuli už v polovici 20. storočia, kedy neurofyziológ Warren McCulloch a mladý matematik Walter Pitts publikovali vedeckú prácu o tom, ako fungujú neuróny v ľudskom mozgu [13].



Obrázok 8. štruktúra typického biologického neurónu [14]

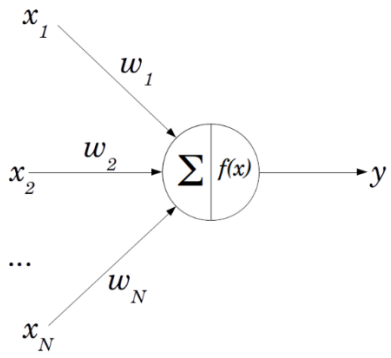
Neurón je základnou stavebnou jednotkou ľudského mozgu. Človek ich má v hlave 100 miliárd. Na obrázku 8 môžeme vidieť stavbu jedného biologického neurónu. Najdôležitejšími

komponentami, ktorými sa inšpirovali aj umelé neurónové siete, sú telo (angl. *cell body*), dendrity (angl. *dendrites*) axon a synapsy. Dendrity sú prepojené s ďalšími neurónmi, od ktorých prijímajú signály. Tie sú cez dendrity posielané do tela, v ktorom sa hromadí potenciál neurónu. V určitom bode nahromadený potenciál spôsobí excitáciu neurónu, vďaka čomu vypudí zo seba signál. Ten pošle cez axon do ďalších neurónov prostredníctvom synáps [15].

Fungovanie neurónov inšpirovalo vedu natoľko, že spolu s príchodom počítačov sa začali robiť prvé pokusy zostrojenia neurónových sietí. Tie spočiatku nedopadli dobre. No prvým konceptom, s ktorým prišiel neurobiológ Frank Rosenblatt a ktorý posunul vedu v oblasti neurónových sieti vpred, bol perceptrón (viac rozoberiem v sekcii 3.2). Tento objav spôsobil veľký ošiaľ. No vzápätí prišlo aj veľké sklamanie, pretože všetky sľuby, očakávania a nádeje, ktoré sa do neho vkladali, brzdila nedostatočná výpočtová sila. Ich praktické využitie prišlo na scénu až začiatkom 21. storočia, kedy sa výpočtový výkon procesorov a grafických kariet výrazne posunul vpred [13].

3.2. Perceptrón

Perceptrón je základným stavebným kameňom komplexnejších architektúr neurónových sietí.

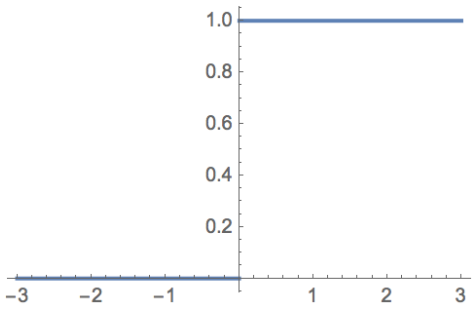


Obrázok 9. Jednoduchý perceptrón [16]

Na obrázku 9 môžeme vidieť jeho grafické znázornenie. Pozostáva zo vstupných parametrov, funkcie a výstupu. ***x*** je vektor vstupných hodnôt, resp. hodnoty pozorovaní v dátach, ***w*** je vektor váh, ktoré sa so vstupnými hodnotami lineárne skombinujú operáciou:

$$h = \mathrm{w}_1.x_1 + w_2.x_2 \ldots w_N.x_N + w_{N+1}.x_{N+1} = \sum_{i=1}^{N+1} w_i.x_i; \quad x_{N+1} = 1\ (bias) \quad (5)$$

Medzivýstup *h* sa pošle do aktivačnej funkcie *f(h)* (Obrázok 10), ktorá rozhodne o tom, aká bude výsledná hodnota.



$$f(h) = \begin{cases} 0, & h < 0 \\ 1, & h \geq 0 \end{cases}$$

Obrázok 10. Aktivačná funkcia [17]

Peceptrón je dobrý na lineárne separovateľné problémy, t.j. klasifikáciu do 2 tried. To znamená, že v konečnom čase sa dá nájsť hraničná priamka, ktorá rozdelí dáta do 2 skupín podľa toho, do ktorých patria [15]. Na to, aby bolo niečo takéto vôbec možné, preceptrón sa potrebuje najprv natrénovať na dátach, na základe ktorých nájde optimálnu priamku (viac v podkapitole 3.3). Na komplexnejšie problémy slúžia zložitejšie architektúry ako napríklad viacvrstvový perceptrón, rekurentné neurónové siete, konvolučné neurónové siete a podobne. Tie vedia modelovať aj zložitejšie prípady. O nich sa dozvieme čosi viac až v neskorších častiach tejto kapitoly.

3.3. Trénovanie

Trénovanie je proces, počas ktorého sa snaží algoritmus nájsť pre trénovaciu dátovú množinu optimálne parametre matematickej funkcie tak, aby ich čo najlepšie opisovala. V prípade neurónových sietí pozostáva z 2 krokov: *dopredného šírenia* a *spätného šírenia chyby*.

Dopredné šírenie je proces, pri ktorom sú vstupné hodnoty lineárne kombinované s váhami (vzorec 5) a spracované aktivačnou funkciou na jednej (perceptrón) alebo postupne na viacerých vrstvách (viacvrstvový perceptrón). Výsledkom tohto procesu je hodnota, ktorá by mala byť čo najbližšia k očakávanému výsledku.

Spätné šírenie chyby je proces, ktorý sa snaží na základe veľkosti chyby medzi vypočítanou hodnotou a očakávanou hodnotou upraviť všetky váhy tak, aby v budúcnosti bola veľkosť chyby menšia. Funkciu, ktorá počíta veľkosť chyby a ktorú chceme minimalizovať, sa volá *chybová funkcia*.

$$E = \frac{1}{2}(y' - y)^2 \tag{6}$$

y' je vypočítaná hodnota a *y* je očakávaná hodnota [16]. Vzorec 6 počíta chybu len pre jedno pozorovanie. Pre viac pozorovaní by musel byť celý vzorec obalený v sume. Výpočet chyby len pre jedno pozorovanie uvádzam kvôli jednoduchosti vysvetlenia. To, akým pomerom jednotlivé váhy prispievajú k veľkosti chyby zistíme tak, že sa zderivuje chybová funkcia vzhľadom na konkrétne váhy.

$$\frac{dE}{dw_i} = \frac{1}{2}\frac{d}{dw_i}(y' - y)^2 \text{ [16]} \tag{7}$$

Deriváciou zistime to, že ak sa spraví malá zmena konkrétnej váhy, ako veľmi ovplyvní hodnotu, teda chybu na výstupe [16]. Ak postupne zderivujeme chybovú funkciu vzhľadom na všetky váhy, t.j. pomocou parciálnych derivácií, zistíme gradient chybovej funkcie [12]. Gradient je vektor, ktorý ukazuje v určitom bode smer najväčšieho nárastu funkcie [18]. Keďže my chceme minimalizovať chybovú funkciu, váhy sa budú upravovať v protismere gradientu. Tento prístup sa v zahraničnej literatúre nazýva *gradient descent* [12]. Váhy sa tak budú upravovať podľa vzorca:

$$w_i' = w_i - \alpha.\frac{dE}{dw_i} = w_i + \alpha(y' - y)x_i \text{ [16]} \tag{8}$$

$\boldsymbol{w_i}$*'* je upravená váha, α je rýchlosť učenia, ktorá určuje veľkosť úpravy váh (viac v podkapitole 3.4), $\boldsymbol{x}_i$ je vstupná hodnota *i* pre váhu *i*. Keď sa tento proces bude opakovať veľa krát (v zložitých prípadoch sa pohybujeme rádovo v miliónoch, preto trénovanie trvá dlho), hodnota chybovej funkcie by mala skonvergovať k minimu. Problémom je, že to nemusí byť globálne minimum ale lokálne. Na všeobecné zlepšenie, resp. optimalizáciu modelu, ktorá nám pomôže zlepšiť viaceré aspekty neurónových sieti, existuje niekoľko techník, ktoré si rozoberieme v ďalšej časti.

3.4. Optimalizačné techniky

V predošlej časti sme hovorili, že cieľom trénovacej fázy je nájsť optimálne hodnoty parametrov matematickej funkcie opisujúcej vzorky dát v dôsledku minimalizovania chybovej funkcie, resp. veľkosti chyby modelu. Nájdenie najlepšieho nastavenia parametrov je optimalizačným problémom, ku ktorému sa pristupuje viacerými spôsobmi. Teraz spomeniem len zopár z tých, ktoré sú charakteristické pre hlboké neurónové siete.

3.4.1. Varianty zostupu gradientu

Pri úprave váh v smere zostupujúceho gradientu má na skonvergovanie chybovej funkcie k minimu veľký vplyv veľkosť vzorky dát, na základe ktorej sa ráta celková chyba. Výpočet

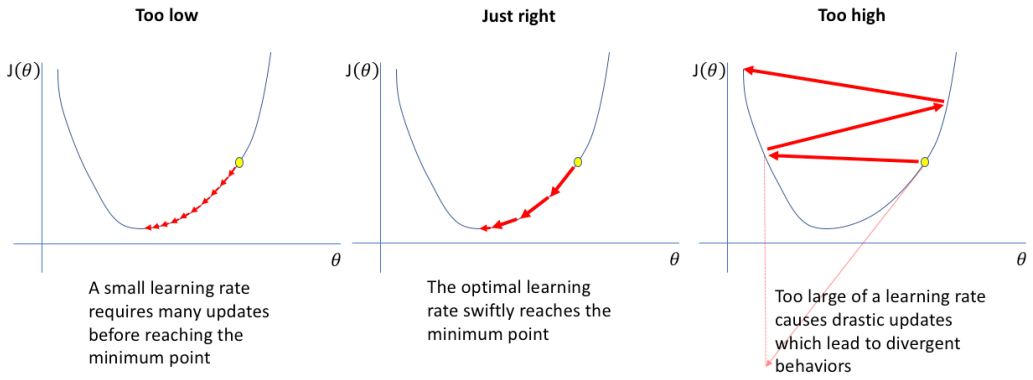
chyby a jej následné prešírenie, pri ktorom sa rátajú veľkosti gradientu na jednotlivých vrstvách, sa v jednej trénovacej iterácií môže robiť po každej vzorke zvlášť (angl. *stochastic gradient descent*), po menších skupinách vzoriek (angl. *mini-batch gradient descent*) alebo až po predikcii hodnôt všetkých vzoriek z trénovacej množiny (angl. *batch gradient descent*). Každá ma svoje výhody a nevýhody.

V prípade, že sa zoberú v jednom trénovacom cykle všetky vzorky dát, veľkosť a smer gradientu budú presné, no na druhej strane takýto výpočet môže trvať pri veľkom počte dát (rádovo na úrovni miliónov vzoriek) veľmi dlho. Navyše v prípade, že chybová funkcia, ktorej minimum sa snaží algoritmus nájsť, nie je konvexná, hrozí, že algoritmus rýchlo skonverguje k lokálnemu minimu. Ak sa v jednom trénovacom cykle zoberie len jedna vzorka, schopnosť algoritmu nájsť optimálne minimum aj v prípade nekonvexnej chybovej funkcie je vyššia oproti predošlému variantu, keďže gradienty sú zašumené a konvergencia k minimu nie je tak priamočiara. Na druhej strane v prípade príliš zložitého problému algoritmus postačujúce optimálne minimum tiež nemusí nájsť a keďže hľadanie minima je kvôli zašumeným gradientom pomalšie, musí prejsť celú trénovaciu množinu viackrát, čo zvyšuje časovú náročnosť vo fáze učenia. Preto existuje ešte tretia varianta, výber menšej vzorky dát (*batch gradient descent*), ktorá je zlatou strednou cestou. To znamená, že v záujme efektívneho odhadu gradientu a nájdenia postačujúceho optimálneho minima chybovej funkcie je dôležité vhodne nastaviť veľkosť vzorky v jednom trénovacom cykle. Kvôli výpočtovej efektivite na GPU sa zvyčajne volí hodnota, ktorá je mocninou dvojky a zároveň je medzi hodnotami 32 – 256 [12].

3.4.2. Rýchlosť učenia

Rýchlosť učenia je najzakladanejší hyperparameter, ktorý sa zvykne označovať ako α. Tento parameter sa vyskytuje napríklad pri spätnom šírení chyby vo vzorci 8. V zahraničnej odbornej literatúre ho nájdeme pod názvom *learning rate*. Určuje nám mieru úpravy váh.

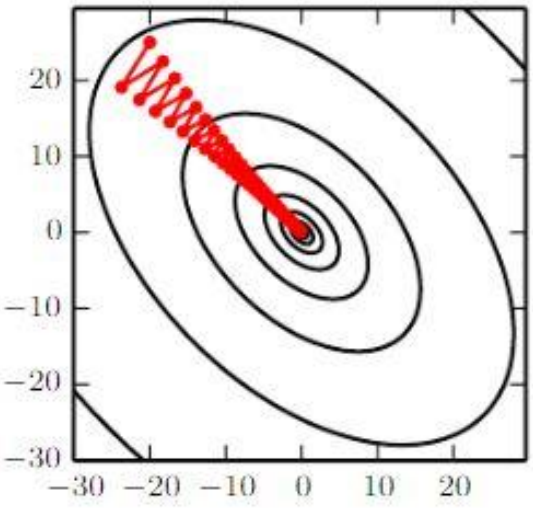
Keď je moc veľká, veľkosť chybovej funkcie môže oscilovať okolo minima (lokálneho alebo globálneho), ale nikdy k nemu neskonverguje (Obrázok 11 – pravá časť). Na druhej strane ak bude malá, veľmi dlho bude trvať, kým skonverguje k minimu (Obrázok 11 – ľavá časť). Preto sa odporúča na začiatku nastaviť väčšiu rýchlosť učenia a iteráciami ju postupne znižovať, aby sa na začiatku pomerne rýchlo našiel priestor s minimom a následne aj k nemu skonvergoval (Obrázok 11 – stredná časť) [12].



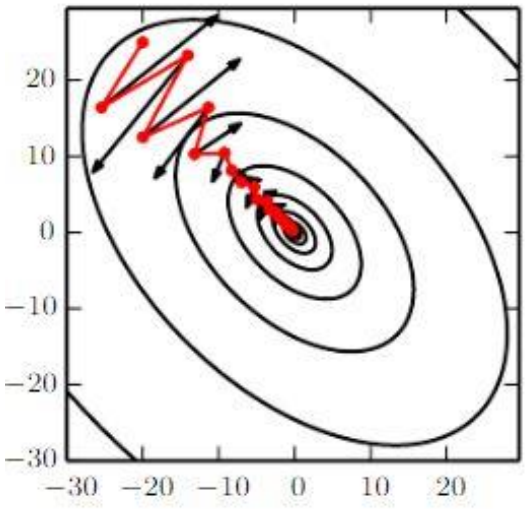
Obrázok 11. Rôzne rýchlosti učenia [19]

3.4.3. Momentum

Ďalšou z techník, pomocou ktorej vieme urýchliť učenie, je momentum. Dobre funguje vtedy, keď má chybová funkcia vysoké zakrivenie, malé ale konzistentné gradienty alebo zašumené gradienty.



Obrázok 132. Gradient Descent [12]



Obrázok 123. Gradient Descent s Momenum [12]

Na obrázkoch 12 a 13 vidíme chybovú funkciu vizualizovanú pomocou vrstevníc. Červená čiara znázorňuje to, ako model hľadal jej minimum. Bez využitia optimalizačnej techniky momentum by musel model spraviť viac krokov na to, aby našiel minimum. Aj keď momentum akumuluje predošlé gradienty a preto urýchľuje pohyb pozdĺž prudko klesajúceho gradientu chybovej funkcie, predsa sa môže stať, že v nejakom bode nemusí ukazovať a teda redukovať chybu správnym smerom. Na to tu je tzv. *Nesterov momentum*. Pri ňom sa

nepočíta gradient pri úprave váh od súčasnej pozície, ale od novej pozície po aplikovaní momentum (tzv. *Nesterov accelerated gradient*) [12, 20].

3.4.4. Batch normalizácia

Pri trénovaní hlbokých neurónových sietí je veľmi dôležité, aká je distribúcia dát na vstupe. V prípade, že distribúcia celého datasetu má vysokú varianciu a trénovanie je stochastické, veľmi jednoducho sa stane, že každý trénovací cyklus bude obsahovať dáta s rôznou distribúciou a teda sieť sa veľmi ťažko naučí funkciu, ktorá bude robustne a genericky aproximovať dáta. To primárne platí pre skryté vrstvy neurónov hlbokej neurónovej siete, ktorých výstupné hodnoty do istej miery závisia od hodnôt na vstupe siete. Ak bude distribúcia dát každý trénovací cyklus odlišná, vo väčšej miere sa bude meniť aj distribúcia hodnôt neurónov v skrytých vrstvách. Aby sa predišlo tomuto problému a sieť mohla robiť približne s rovnakou distribúciou dát v každom trénovacom cykle, používa sa na to batch normalizácia, ktorá znormalizuje distribúciu dát podľa priemeru a variancie danej vzorky (tzv. *batch*) [21, 12].

$$H' = \frac{H - \mu}{\sigma} \text{ [12]} \tag{9}$$

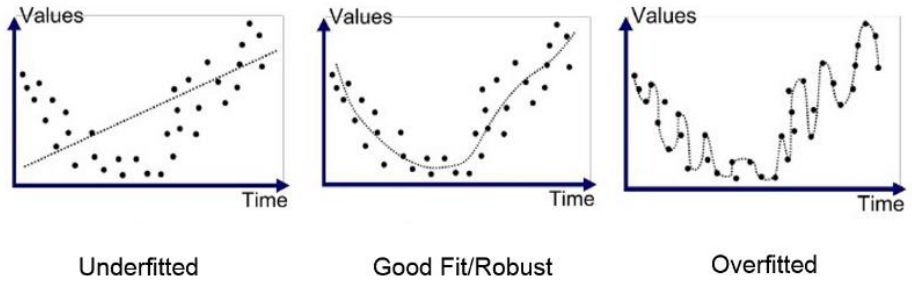
V prípade, že sa aplikuje batch normalizácia (vzorec 9) na vstup a na každú skrytú vrstvu, pri úprave váh sa sieť rýchlejšie naučí minimalizovať chybovú funkciu, keďže distribúcia vstupných dát a na vrstvách neurónov bude naprieč trénovacimi cyklami približne rovnaká [21].

3.4.5. Iné

Existuje ešte veľa ďalších prístupov, ako vylepšiť učenie modelu. Napr. definovať kritérium, ktoré keď model dosiahne, tak sa skončí trénovanie (angl. *early stopping*). Taktiež vieme, že pred samotným trénovaním sa váhy modelu inicializujú na nejakú počiatočnú hodnotu. Spôsob náhodnej inicializácie môže tiež výrazne ovplyvniť rýchlosť nájdenia optimálneho minima. Poznáme aj rôzne aktivačné funkcie, vďaka ktorým môžeme podporiť model v správaní, ktoré od neho očakávame. Potom sú aj rôzne variácie optimalizácie založenej na metóde *gradient descent*. Tie sa rôznymi spôsobmi snažia prispôsobovať veľkosti rýchlosti učenia pre každú váhu zvlášť. Nimi sú napríklad *Adagrad, Adelta, RMSProp, Adam* [22].

3.5. Regularizačné techniky

Regularizácia je mechanizmus na predídenie problému preučenia.



Obrázok 14. Rôzna komplexita modelov [23]

Vo všeobecnosti sa regularizačné techniky snažia riešiť tzv. *Bias-Variance Tradeo-ff* . Bias je veľkosť chyby modelu, resp. celkový rozdiel medzi očakávanými a predikovanými hodnotami. Model, ktorý ma vysoký bias, príliš zjednodušuje problém, čo ma za dôsledok veľkú chybovosť jednak na trénovacej ale aj testovacej množine (Obrázok 14 – ľavá vizualizácia). Variancia modelu definuje jeho variabilitu a schopnosť brať pri učení do úvahy aj prítomný šum v dátach. Model s vysokou varianciou kladie príliš veľký dôraz na dáta v trénovacej množine vrátane šumu, čo ma za dôsledok vysokú úspešnosť predikcii modelu na trénovacej množine, no nízku na testovacej množine (Obrázok 14 – ľavá vizualizácia) [24]. Vďaka aplikovaniu správneho typu regularizácie sa predíde príliš komplexnému modelu, ktorý má vysokú varianciu a malý bias (angl. *overfitting*, Obrázok 14 – pravá vizualizácia) alebo jednoduchému modelu, ktorý má nízku varianciu a veľký bias (tzv. *underfitting*, Obrázok 14 – ľavá vizualizácia). Tým sa zabezpečí dobrý pomer medzi kvalitou modelu a jeho robustnosťou (Obrázok 14 – stredná vizualizácia) [12]. Poznáme napríklad L1, L2 [25, 21], ktoré si v nasledujúcich sekciách detailnejšie vysvetlíme.

3.5.1. L1 regularizácia

Úlohou L1 regularizácie je priradiť nulovú hodnotu bezvýznamným parametrom a nenulovú hodnotu užitočným parametrom. Inými slovami, robí výber dôležitých čŕt (angl. *feature selection*). Jeho vzorec vyzerá nasledovne [25]:

$$E = \sum_{i=1}^{n}\left(y_i - h(x_i)\right)^2 + \lambda \sum_{i=1}^{n} |\theta_i| \text{ [25]} \qquad (10)$$

$\boldsymbol{h}$ je funkcia modelu, ktorá robí predikciu, $\boldsymbol{y_i}$ je očakávaná hodnota a $\boldsymbol{\lambda}$ je parameter, ktorý určuje, ako veľmi penalizuje váhy $\boldsymbol{\theta_i}$. Ak je tento parameter nulový, výsledkom je obyčajná chybová funkcia (v princípe vzorec 6). V opačnom prípade podnecuje „riedkosť“ (angl. *sparsity*) matice čo znamená, že sa hľadajú riešenia s preferenciou v takých priestoroch parametrov, kde regularizované jadrá sú podľa možností riedke matice (veľa nulových hodnôt parametrov) a ak v tomto priestore sú dobré riešenia, tak skôr dôjdeme k tým [25].

3.5.2. L2 regularizácia

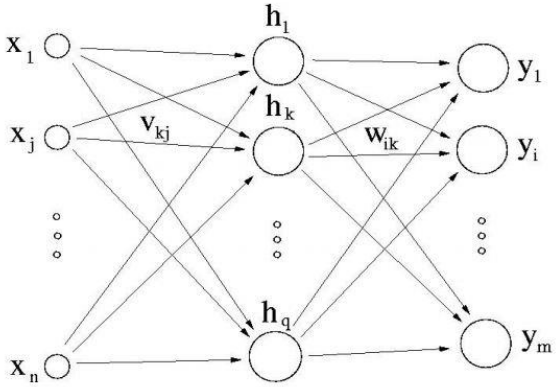
Rola L2 regularizácie je o čosi iná. Potláča hodnoty váh k nule ale neprirađuje im nulové hodnoty ako to robí L1. Zabraňuje tak individuálnym črtám, aby mali neúmerne vyššiu dôležitosť ako ostatné. Vzorec je nasledovný [25]:

$$E = \sum_{i=1}^{n}\left(y_i - h(x_i)\right)^2 + \lambda \sum_{i=1}^{n} \theta_i^2 \text{ [25]} \qquad (11)$$

L2 je matematickým zápisom veľmi podobná L1. Rozdiel je v regularizačnej časti vzorca. Pri L1 sa robí súčet absolútnych hodnôt všetkých váh a v L2 sa robí súčet štvorcov všetkých váh. Po aplikovaní L2 bude výsledkom taký model, ktorý má síce veľa parametrov ale žiadny nie je až tak extrémne dôležitý (všetky váhy sú malé) [25].

3.6. Jednoduchá architektúra hlbokej neurónovej siete – viacvrstvový perceptrón

Ako sme už spomínali, stavebným kameňom hlbokých neurónových sietí je perceptrón. Ten však vieme replikovať, čím vytvoríme vrstvu perceptrónov. Počet vrstiev vieme tiež zväčšovať, až dostaneme niečo takéto:



Obrázok 15. Viacvrstvový perceptrón [26]

Na obrázku 15 vidíme architektúru viacvrstvového perceptrónu. Konkrétne má 3 vrstvy – vstupnú, skrytú a výstupnú vrstvu. Takúto architektúru neurónovej siete nazývame plne prepojenú (angl. *fully-connected*), pretože každý perceptrón (ďalej len neurón) je spojený so všetkými neurónmi na predošlej vrstve a každé spojenie má vlastnú váhu. Toto platí pre všetky vrstvy okrem vstupnej. Výpočet matematickej funkcie, ktorou bude takýto model opisovať dáta, bude nasledovný:

$$h_k = f(net_k); net_k = \sum_{j=1}^{n+1} v_{kj} x_j \text{ [26]} \qquad (12)$$

$$y_i = f(net_i); net_i = \sum_{k=1}^{q+1} w_{ik} h_k \text{ [26]} \quad (13)$$

$$x_{n+1} = h_{q+1} = -1 \text{ [26]} \quad (14)$$

Najprv sa vypočítajú lineárne kombinácie medzi vstupom a váhami na prvej vrstve na ktoré sa aplikujú aktivačné funkcie (vzorec 12). Výsledky aktivačný funkcií na skrytej vrstve budú slúžiť ako vstup do výstupnej vrstvy, kde sa udeje to isté (vzorec 13). x_{n+1} a h_{q+1} majú fixnú hodnotu, pretože reprezentujú vstupný bias. Pri hlbších architektúrach neurónových sietí musíme dbať na to, že fáza trénovania bude trvať dlhšie, lebo veľkosť chyby sa musí prešíriť spätne cez všetky vrstvy. V takomto prípade sa bude spätné šírenie chyby rátať takto:

$$w_{ik}(t+1) = w_{ik}(t) + \alpha \delta_i h_k;\ \delta_i = (d_i - y_i) f'(net_i) \quad (15)$$

$$v_{kj}(t+1) = v_{kj}(t) + \alpha \delta_k x_j;\ \delta_k = (\sum_i w_{ik} \delta_i) f'(net_k) \quad (16)$$

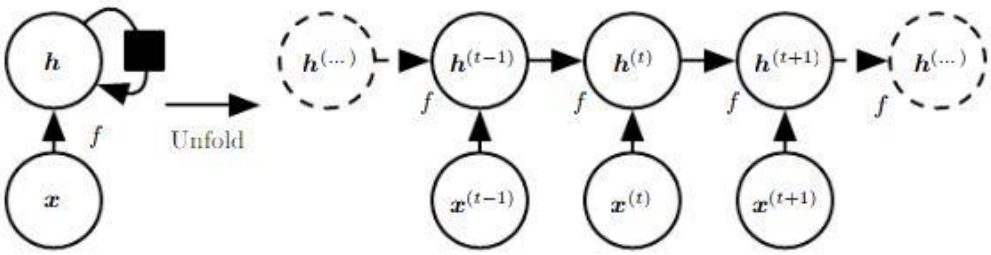
Architektúra viacvrstvového perceptrónu bola odrazovým mostíkom pre zložitejšie architektúry. Tieto architektúry sa vedia vysporiadať s komplexnejšími problémami. Hoci sme doteraz hovorili o hlbokých neurónových sieťach, v skutočnosti až pri viacvrstvovom perceptróne má tento pojem zmysel. A to kvôli variabilnému počtu vrstiev, ktoré sú súčasťou architektúry. Takáto neurónová sieť na rôznych úrovniach modeluje iné charakteristiky dát, ktoré potom používa pri výpočte. Keď sme si už vysvetlili základnú architektúru, môžeme sa pustiť do zložitejších, s ktorými sa stretneme pri riešení rôznych problémov ADT.

3.7. Najpoužívanejšie architektúry hlbokých neurónových sietí pri riešení problémov ADT

V tejto časti sa zameriame na architektúry hlbokých neurónových sietí, ktoré sa veľmi často objavujú pri riešení úloh ADT. Každá z nich je určená na riešenie určitých typov problémov. V nasledujúcich sekciách sa dozvieme ich základné koncepty a intuíciu, z čoho vždy vyvodíme zopár pozitívnych a aj negatívnych dôsledkov.

3.7.1. Jednoduchá rekurentná neurónová sieť (RNN)

Rekurentné neurónové siete (ďalej už len RNN) sú špeciálne určené na spracovanie dát vyvíjajúcich sa v čase, ktoré na seba nadväzujú. Dobrými príkladmi sú napríklad predpoveď počasia, sledovanie vývoja určitých ekonomických ukazovateľov, spracovanie prirodzeného jazyka alebo hudby. Táto schopnosť vyplýva z ich architektúry, ako sú navrhnuté.



Obrázok 16. Koncept základnej architektúry RNN [12]

Konceptuálne môžeme vnímať RNN tak, ako je zobrazená na ľavej časti obrázku. Majú v sebe cyklus, ktorý reprezentuje vplyv vstupu $\boldsymbol{x}$ a hodnoty premennej $\boldsymbol{h}$ na jej hodnotu v budúcnosti. Pre lepšie pochopenie sa môžeme zamerať na pravú časť obrázka, na ktorej je RNN rozvinutá (angl. *unfold*) v čase. Premenná $\boldsymbol{h}$ reprezentuje stav vo forme váh, ktoré sú upravované a zároveň zdieľané naprieč časom. To znamená, že tento stav vie modelovať vnútorné závislosti sekvencie dát do času $\boldsymbol{t-1}$, ktoré majú potom ovplyvniť model sekvencie dát v čase $\boldsymbol{t}$ [12].

$$h^{(t)} = f(h^{(t-1)}, x^{(t)}; \theta) \text{ [12]} \tag{17}$$

Na základe matematickej formuly to môžeme vyjadriť aj tak, že stav v čase $\boldsymbol{t-1}$ modelujúci závislosti zo všetkých predošlých vstupov a aktuálny vstup v čase $\boldsymbol{t}$ ovplyvnia to, ako bude vyzerať stav v čase $\boldsymbol{t}$. Aj keď si predstavíme rekurentnú sieť ako sekvenciu vrstiev, čo sa potom môže javiť ako viacvrstvový perceptrón, architektúry budú stále rozdielne hlavne v dvoch ohľadoch. Prvým je, že pri viacvrstvovom perceptróne sú váhy osobitné v každej vrstve (teda nie sú zdieľané ako pri RNN) a druhým je, že sekvenčné dáta nevstupujú „*postupne do každej vrstvy zvlášť*", ale prešíria sa všetkými vrstvami [12].

Pre lepšiu predstavu si ale môžeme predstaviť RNN ako rozvinutú neurónovú sieť s počtom vrstiev, ktorý je rovný veľkosti vstupu. Keďže neurónová sieť sa musí trénovať, nemôže sa spraviť to, že sa spracuje celý vstup a nakoniec sa upravia váhy pomocou spätného šírenia chyby. Obzvlášť nie pri enormne veľkom množstve dát a systémoch, ktoré tieto dáta spracovávajú v reálnom čase. Preto sa vstup poväčšine spracováva po dávkach (angl. *minibatches*). Po spracovaní dávky sa vždy vykoná tzv. spätné šírenie chyby naprieč časom (angl. *Backpropagation through time*). Je to presne to isté, čo klasické spätné šírenie chyby, ale s drobnými úpravami [12].

Koncept zdieľania váh a rozvinutie grafu v čase sú dobrým predpokladom na to, aby sme pochopili rôzne návrhové vzory pri tvorbe architektúr RNN. Nimi sú napríklad:

- RNN, ktoré v každom časovom okamihu produkujú výstup a majú rekurentné prepojenia medzi skrytými vrstvami (nasledujúci stav ovplyvňuje predošlý stav spolu s aktuálnym vstupom) [12]
- RNN, ktoré v každom časovom okamihu produkujú výstup a majú rekurentné prepojenia medzi výstupmi a skrytými vrstvami (nasledujúci stav ovplyvňuje predošlý výstup a aktuálny vstup) [12]
- RNN, ktoré majú rekurentné prepojenia medzi skrytými vrstvami, prečítajú celý vstup a na záver vyprodukujú jeden výstup [12]

Vzhľadom na to, že hudba, a teda aj bicie, sa vyvíjú v čase, zahrané sekvencie tónov závisia na sebe a preto má zmysel používať RNN v ADT. Avšak jednou z ich hlavných slabín je to, že príliš sa sústredia na závislosť vzťahov medzi bubnami, čo pri nedostatočnej vzorke dát degraduje robustnosť modelu [3].

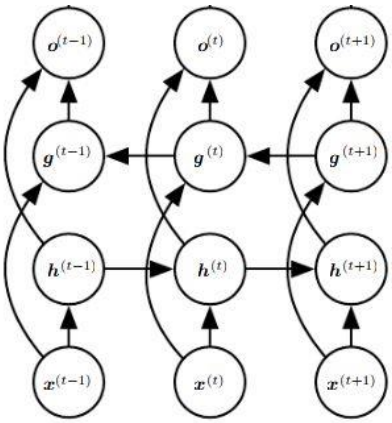
Jednoduché RNN podnietili vedu k tomu, aby vymyslela zložitejšie variácie RNN. Nimi sú napríklad obojsmerné RNN, LSTM RNN alebo GRU RNN. O nich si povieme viac v nasledujúcich častiach. Tieto typy RNN sú súčasťou najmodernejších prístupov v oblasti ADT.

3.7.2. Obojsmerná rekurentná neurónová siet (BDRNN)

Obojsmerná RNN (ďalej už len BDRNN) je typ RNN, ktorej výstup v aktuálnom čase je ovplyvnený informáciami nielen z minulosti, ale aj z budúcnosti. Sú to v podstate 2 nezávislé RNN, ktorých výstupy sa v každom čase kombinujú [12]. Architektúra takejto neurónovej siete je znázornená nižšie (viď Obrázok 17).

$\boldsymbol{x}^{(t)}$ charakterizuje vstup v čase $\boldsymbol{t}$, $\boldsymbol{h}^{(t)}$ je stav doprednej RNN v čase $\boldsymbol{t}$ a $\boldsymbol{g}^{(t)}$ je stav spätnej RNN v čase $\boldsymbol{t}$. Dopredná RNN spracováva vstup od začiatku, spätná RNN spracováva vstup od konca. Stavy oboch RNN v čase $\boldsymbol{t}$ spoločne ovplyvňujú výstup v čase $\boldsymbol{t}$. BDRNN predpokladá prítomnosť celej vstupnej sekvencie. To znamená, že nemôžu byť súčasťou systémov, ktoré pracujú s dátami v reálnom čase. Môžu však slúžiť na spracovanie, ktoré sa vykonáva dodatočne [12].

BDRNN je jedna z architektúr, ktorá sa radí medzi jedny z najmodernejších prístupov riešiacich ADT problémy [3, 27, 28]. Je to primárne kvôli tomu, že úder na konkrétny bubon



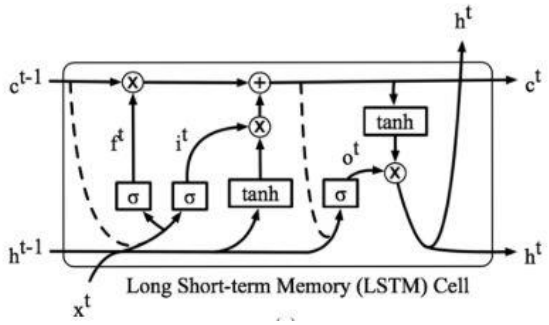
Obrázok 17. Architektúra obojsmernej RNN [12]

v čase $\boldsymbol{t}$, ktorý špecifickým spôsobom dotvára rytmický sled udalostí, môže kvôli plynulej rytmickej artikulácií závisieť od pár predošlých a pár nasledujúcich úderov na bubny. Podobne to funguje aj pri rozpoznávaní reči, kde správna interpretácia aktuálneho zvuku fonémy môže závisieť od niekoľko predošlých a zopár nasledujúcich foném [12]. Opäť však hrozí podobný problém ako pri RNN, že v prípade nedostatočného množstva dát sa robustnosť modelu degraduje.

3.7.3. Rekurentné neurónové siete s dlhodobejšou pamäťou (LSTM, GRU)

Ako sme už viackrát spomenuli, RNN slúžia na modelovanie vnútorných štruktúr v sekvenčných dátach. Môžeme ich nazvať aj ako siete s pamäťou. Klasické RNN sú schopné zapamätať si informácie, ktoré prišli pomerne nedávno na vstupe a z nich vyvodiť aktuálny výstup. No majú problém zapamätať si informácie v širšom kontexte. Preto existujú typy RNN, ktorých architektúra je vyslovene zameraná na modelovanie dlhodobejších vzťahov v sekvenčných dátach [12]. V súčasnosti jedny z najpopulárnejších sú LSTM (angl. *Long Short Term Memory*) RNN a potom GRU (angl. *Gated recurrent unit*) RNN, ktorá je zjednodušená verzia LSTM. Tie teraz podrobnejšie rozpíšem.

LSTM sú navrhnuté na dlhodobé pamätanie informácií. Tým, že LSTM je typ rekurentnej siete, tak aj ju si môžeme predstaviť ako rozbalenú RNN v čase, ktorá vyzerá takto:



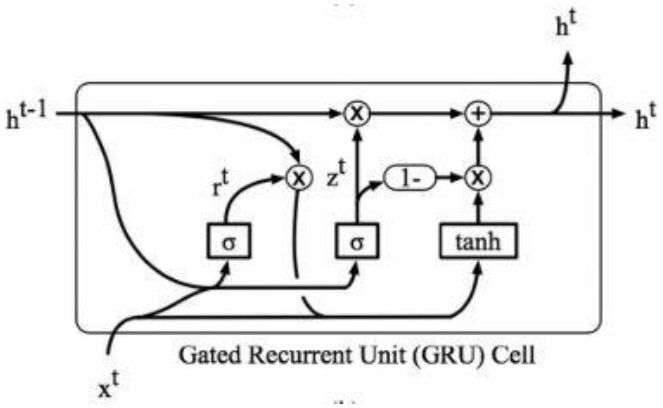
Obrázok 18. LSTM RNN [3]

Na obrázku 18 vidíme podrobnejšiu architektúru LSTM pamäťovej bunky. Jej fungovanie vysvetlím najprv matematicky po krokoch a potom uvediem slovný opis.

1. $f_t = \sigma(W_f . [h_{t-1}, x_t] + b_f)$ [29] (18)
2. $i_t = \sigma(W_i . [h_{t-1}, x_t] + b_i)$ [29] (19)
3. $C'_t = \tanh(W_C . [h_{t-1}, x_t] + b_C)$ [29] (20)
4. $C_t = f_t * C_{t-1} + i_t * C'_t$ [29] (21)
5. $o_t = \sigma(W_o . [h_{t-1}, x_t] + b_o)$ [29] (22)
6. $h_t = o_t * tanh(C_t)$ [29] (23)

LSTM bunka obsahuje stav (pamäť $\boldsymbol{C_t}$), ktorý je zdieľaný naprieč časom. Jeho zmeny sú vyjadrené na obrázku 18 na hornej, horizontálnej čiare. Približne v tej časti začína prvý krok. V ňom LSTM rozhodne, ktoré informácie vyhodí zo svojej pamäte (1 krok). Toto sa udeje pomocou vrstvy s aktivačnou funkciou sigmoid $\boldsymbol{\sigma}$, ktorá sa nazýva *forget gate layer* a budeme ju označovať $\boldsymbol{f_t}$. V ďalšom kroku sa rozhodne, ktoré nové informácie sa uložia do pamäte. Tento krok pozostáva z dvoch častí. V prvej sa iná vrstva, ale tiež s aktivačnou funkciou sigmoid s názvom *input gate layer* (označujeme $\boldsymbol{i_t}$), rozhodne, ktoré hodnoty aktualizuje (2 krok). V druhej časti vrstva s aktivačnou funkciou tangens vytvorí vektor C'_t potenciálnych hodnôt, ktoré by mohli byť pridané do pamäte (3 krok). V ďalšom kroku sa aktualizuje pamäť $\boldsymbol{C_t}$ (4 krok). A to tak, že sa vynásobí predošlý stav pamäte $\boldsymbol{C_{t-1}}$ s $\boldsymbol{f_t}$, čím sa zmažú z pamäte konkrétne hodnoty. Následne sa k nej pripočíta vektor potenciálnych hodnôt $\boldsymbol{C'_t}$ vynásobený s vektorom $\boldsymbol{i_t}$, ktorý určuje, ktoré hodnoty sa majú do nej pridať. V poslednom kroku sa rozhodne, čo bude na výstupe. A to tak, že najprv sa vrstva , tiež s aktivačnou funkciou sigmoid, rozhodne, ktoré časti z pamäte budú na výstupe (5 krok). Označujeme ju $\boldsymbol{o_t}$. Tá sa na záver skombinuje s už aktualizovaným stavom pamäte $\boldsymbol{C_t}$, vďaka čomu dostaneme len tie hodnoty z aktualizovanej pamäte, ktoré chceme (6 krok) [29].

LSTM RNN má aj niekoľko variant. Jednou z nich je spomínaná GRU RNN, ktorej architektúra je jednoduchšia a efekt ten istý – dokážu modelovať dlhodobejšie závislosti v sekvenčných dátach. GRU kombinuje *input gate* a *forget gate* do tzv. *update gate* a zlučuje stav bunky (pamäť $\boldsymbol{C_t}$) so skrytým stavom $\boldsymbol{h_t}$ [29]**.** Jej architektúra je zobrazená na obrázku 19.



Obrázok 19. GRU RNN [3]

Matematické vyjadrenie správania GRU RNN je zredukované len na pár krokov.

1. $z_t = \sigma(W_z.[h_{t-1}, x_t])$ [29] (24)
2. $r_t = \sigma(W_r.[h_{t-1}, x_t])$ [29] (25)
3. $h'_t = tanh(W.[r_t * h_{t-1}, x_t])$ [29] (26)
4. $h_t = (1 - z_t) * h_{t-1} + z_t * h'_t$ [29] (27)

GRU je modifikácia LSTM. Má jednoznačne menej parametrov, čo je ovplyvnené počtom brán (angl. *gate*) a zlúčením pamäte so skrytým stavom. Preto GRU v porovnaní s LSTM trvá menej času, kým optimalizuje všetky svoje parametre v trénovacej fáze. Na druhej strane zas treba viac buniek GRU nato, aby sa vyrovnala tej istej modelovacej kapacite LSTM bunky [30].

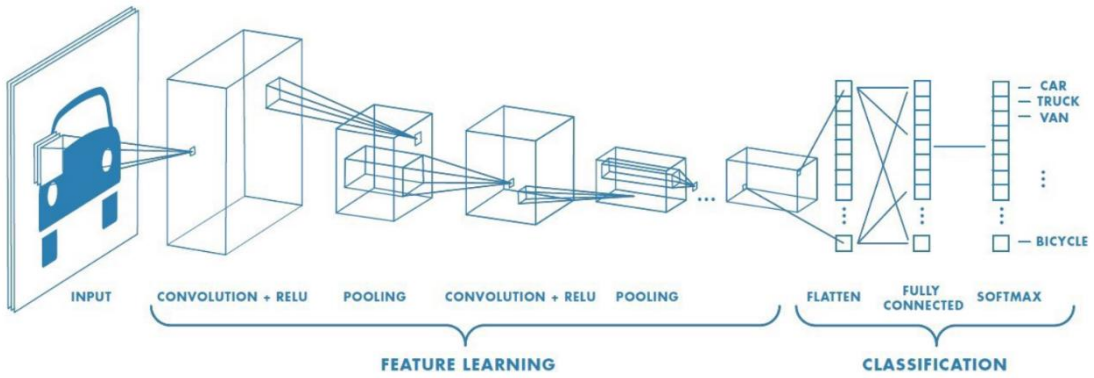
Výhodou použitia takéhoto typu pamäte pri ADT úlohách je taký, že vie dobre modelovať vnútornej reprezentácie vzťahov medzi udalosťami bicích. Nevýhodou je zas to, že sa až príliš sústreďuje na vzťahy medzi údermi bicích a robustnosť tak ostáva v úzadí. Taktiež je veľmi ťažko interpretovateľné to, čo sa naučila. To je však problém pri väčšine architektúr neurónových sietí [3].

3.7.4. Konvolučné neurónové siete (CNN)

Tento typ architektúry neurónovej siete bol inšpirovaný štruktúrou a spôsobom fungovania vizuálnej kôry, ktorá má v mozgu na zodpovednosť spracovávanie vizuálnych informácií. Hoci táto znalosť bola známa už v 80-tych rokoch 20. storočia, až približne o 30 rokov

neskôr, kedy grafické karty počítačov dosahovali zaujímavé časy pri rátaní náročných matematických operácií, bolo možné túto vedomosť prakticky aplikovať do návrhu samotnej architektúry neurónovej siete. Zásadný zlom nastal presne v roku 2012, kedy skupina vedcov vytvorila taký návrh architektúry neurónovej siete, ktorý dosahoval lepšie výsledky pri klasifikácií obrázkov z databázy ImageNet ako vtedajšie najlepšie modely strojového učenia [31].

Konvolučné neurónové siete sa zvyčajne aplikujú na problémy, kde sa pracuje s vizuálnymi informáciami - analýza obrazu alebo videa, rozpoznávanie objektov, segmentácia obrazu či spracovávanie digitálneho signálu [32].



Obrázok 20. Vizualizácia architektúry konvolučnej neurónovej siete [32]

Ako môžeme vidieť na obrázku vyššie, CNN pozostáva z 2 hlavných častí: sady konvolučných blokov, ktoré sa učia rozpoznávať jednotlivé črty na obrázku a zároveň redukujú dimenziu obrázka do vlastnej reprezentácie (časť obrázka s názvom *feature learning*), klasifikačná časť ktorá pracuje s komplexnejšími vizuálnymi informáciami a jej úlohou je zaradiť obrázok do príslušnej triedy (časť obrázka s názvom *classification*). Viac si priblížime prvú časť.

Jeden konvolučný blok primárne pozostáva z 3 častí. Prvou je konvolúcia, ktorá sa aplikuje na vstup a jej výsledkom sú lineárne aktivácie, keďže konvolúcia je špecifický typ lineárnej operácie. V druhej časti sa na tieto aktivácie aplikuje jedna z nelineárnych aktivačných funkcií (napr. sigmoid, tanh, relu), vďaka ktorej sa model môže učiť nelineárne vzťahy. Tretia časť je tzv. združovanie (angl. *pooling*, ďalej budeme používať anglický preklad), ktorý aplikuje jednu zo súhrnných štatistík na výstupy, ktoré sú blízko pri sebe a vďaka tomu sa

model stáva o čosi robustnejším [12]. V skratke si opíšeme každú jednu z nich a uvedieme, aký majú význam.

Konvolúcia je matematická operácia, konkrétne špecifický typ lineárnej operácie, do ktorej vstupujú dve funkcie a výsledkom je tretia. Tá definuje, ako sa tvar jednej funkcie zmenil aplikovaním druhej. Matematický zápis konvolúcie je nasledovný [12, 33].

$$s(t) = \int x(a)w(t-a) \tag{28}$$

x je jedna funkcia, *w* je druhá funkcia a samotná konvolúcia je integrál kombinácií týchto dvoch funkcií. Tento vzorec zodpovedá konvolúcií, ktorá pracuje so spojitými funkciami. V našom kontexte pracujeme s diskrétnymi hodnotami funkcií a preto vzorec konvolúcie neobsahuje integrál ale sumu [12]. Matematický zápis je potom nasledovný.

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a) \tag{29}$$

V neurónovej sietí sa konvolúcia používa v konvolučných jadrách (angl. *kernels*). Sú to malé výseky (oveľa menšie ako vstup), ktoré sa v závislosti od typu vstupu (1D, 2D, 3D, atď) posúvajú postupne naprieč jeho dimenziou a hľadajú rôzne vzory - hrany, čiary, rohy sa zvyčajne hľadajú v jadrách vrstiev bližším k vstupu a vo vyšších komplexnejšie štruktúry [12]. Tieto vzory sa vopred nedefinujú konvolučným jadrám. Sieť sa musí sama naučiť, aké vzory v nich hľadať.

Už len vďaka tomu, ako sú navrhnuté konvolučné jadrá, sa môže tento typ siete lepšie učiť vzťahy v dátach v porovnaní s klasickými prístupmi neurónových sietí. A to hlave vďaka trom vlastnostiam. Tými sú riedke interakcie (angl. *sparse interactions)*, zdieľanie parametrov (angl. *parameter sharing*) a ekvivariančná reprezentácia vzhľadom na posun (angl. *equivariance to translation*). O riedkosti interakcií hovoríme preto, lebo jadrá sú oveľa menšie ako vstup a posúvajú sa naprieč vstupom s určitým skokom (angl. *stride*). Vďaka tomu vnorené vrstvy siete, ktoré pozostávajú z výstupov konvolúcií, nie sú závislé od všetkých vstupov ako pri bežnej plne prepojenej hlbokej neurónovej sieti, ale len od niektorých. To zabezpečuje menší počet prepojení medzi vstupmi a výstupmi, čo ovplyvňuje menší počet parametrov a teda znižuje sa celková pamäťová a aj výpočtová náročnosť algoritmu v porovnaní s klasickou neurónovou sieťou. Zdieľanie parametrov konvolučných jadier znamená, že namiesto toho, aby sa sieť učila samostatné sady váh pre každú jednu pozíciu (tak to funguje pri obyčajných plne prepojených hlobkých neurónových sieťach) konvolučných

jadier v priestore, učí sa len jednu sadu. Jedno konvolučné jadro sa teda učí rozpoznávať len jednu charakteristickú črtu na vstupe. Vlastnosť zdieľania parametrov zabezpečuje tretiu, už spomínanú ekvivariančnú reprezentáciu konvolučných kernelov vzhľadom na posun. Ak povieme o nejakej funkcií, že je ekvivariantná na posun, znamená to, že ak sa zmení vstup, presne o takú istú mieru sa zmení výstup. Potom je teda jedno, či sa najprv aplikuje transformácia (napr. posun obrázku o *n* pixelov) na vstup a až potom ekvivariantná funkcia, alebo opačne. Dôležité je, že výsledok bude presne ten istý. Ak si to prevedieme do nášho kontextu, tak tým, že konvolúcia je ekvivariančná funkcia, konvolučné kernely dokážu nájsť črty, ktoré hľadajú, kdekoľvek na vstupe za predpokladu, že sa tám nachádzajú. Ak sa napríklad nejaká črta vyskytuje na dvoch rôznych pozíciách, výstupom z konvolučného kernelu by boli presne tie isté dva výstupy s určitým posunom [12].

Nelineárne aktivačné funkcie sa aplikujú na výstupy konvolúcií kvôli tomu, aby sa pri spätnom šírení chyby sieť mohla učiť nelineárne vzťahy. Na optimalizáciu učenia sa pri CNN zvyčajne používajú rôzne regularizačné techniky ako napríklad BatchNorm, ktorý sme spomínali v časti 3.4.4. Tie sa aplikujú ešte pred použitím nelineárnej funkcie aby sa v prípade použitia aktivačných funkcii tanh, sigmoid predišlo problému miznúceho gradientu [34] (angl. *vanishing gradient problem*), kvôli ktorému sa sieť prestáva učiť.

Treťou časťou je **pooling**. Pomáha sieti, aby vnútornú reprezentáciu dát, ktorú si sieť učením vytvára, boli *nemenné na malé posuny* na vstupe (angl. *invariant to small translations*). To znamená, že v prípade, že je nejaký konkrétny objekt o trochu otočený alebo posunutý, sieť ho vie stále rozpoznať. Robí to tak, že aplikuje jednu zo súhrnných štatistík na výstupy, v našom kontexte výstupy z nelineárnych aktivačných funkcií, ktoré sú blízko pri sebe. Poznáme viacero typov *pooling* mechanizmov, napr. *max pooling* (zoberie maximálnu hodnotu z blízkeho okolia), *average pooling* (zoberie priemernú hodnotu z blízkeho okolia) a mnoho iných. Tým, že sa aplikuje na konci každého konvolučného bloku, sieť sa vie naučiť, ktoré črty majú byť nemenné na malé zmeny a ktoré nie [12].

Konvolučný blok, ktorý disponuje týmito tromi časťami, sa môže pri návrhu architektúry opakovať s rôznymi parametrami a zvyšovať tak modelovaciu kapacitu siete. Treba však dávať pozor na to, aby komplexita modelu približne zodpovedala komplexite dát. Toto sa však dá zistiť jedine empiricky pomocou experimentov.

Konvolučná neurónová sieť má praktické využitie pri riešení ADT problémov preto, lebo transkripcia bicích nástrojov sa môže robiť napríklad cez analýzu STFT spektrogramu, ktorý

má podobne ako obraz 2-dimenzionálnu štruktúru. Tento typ architektúry vie byť v ADT o čosi robustnejší oproti RNN, keďže modeluje lokálne črty úderov v krátkom časovom úseku a nie ich časovú závislosť, čo vie byť náchylné na preučenie.

3.8. Zhrnutie

Môžeme povedať, že hlboké neurónové siete sú veľmi zložitým algoritmom, ťažko interpretovateľným, no zato schopným riešiť náročne úlohy na porovnateľnej úrovni ako človek, ak nie ešte lepšie. Najmodernejšie prístupy v ADT používajú rôzne architektúry neurónových sietí. Najpopulárnejšími sú rôzne druhy RNN, BDRNN a CNN. Medzi RNN patria architektúry ako LSTM RNN alebo GRU RNN. BDRNN varianty môžu tiež využívať dlhodobejšie pamäť a preto poznáme napríklad BDLSTM, BDGRU . Všetky architektúry sú však závisle od množiny dát. To znamená, že ak pri testovaní narazia na typ dát, ktorý predtým nevideli (v našom prípade rozdielny zvuk bicích nástrojov ako v trénovacej množine), je možné, že úspešnosť nebude dostatočne vysoká kvôli pretrénovaniu modelu. Preto nájsť vhodnú architektúru alebo ich kombináciu s použitím vhodných regularizačných techník, ktoré vyvážia robustnosť a úspešnosť modelu je cieľ, ale zároveň aj veľký výskumný problém.

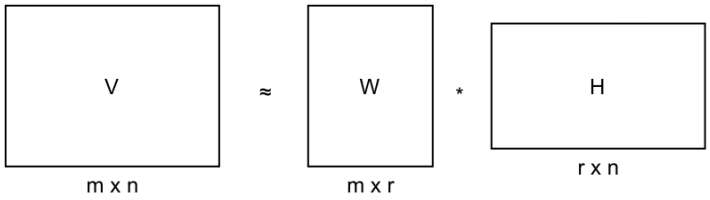
4. Nezáporná maticová faktorizácia (NMF)

Nezáporná maticová faktorizácia (NMF) je jeden z algoritmov strojového učenia založený na prístupe učenie bez učiteľa (angl. *unsupervised learning*) [35]. Už dávnejšie sa ukázalo, že je vhodný pre dekompozíciu viacrozmerných dát [36]. To je aj jeden z dôvodov, prečo ho nájdeme pomerne často vo výskumných prácach zaoberajúcich sa transkripciou hudby a teda aj v podprobléme ADT.

Pre ujasnenie pojmov, slovo *nezáporná* znamená to, že najnižšia možná hodnota, ktorá sa môže vyskytnúť v dekomponovaných maticiach počas procesu faktorizácie, je 0. Táto vlastnosť algoritmu je veľmi dôležitá, pretože vďaka nej sa môže použiť na riešenie problémov, ktoré keď vyjadríme matematicky, budú disponovať nezápornými hodnotami. Základná idea NMF je založená na matematickej formule

$$V \approx WH \text{ [36]} \tag{30}$$

ktorú môžeme vizualizovať nasledovným spôsobom



Obrázok 21. Vizualizácia algoritmu NMF

V je matica o rozmeroch *m x n*, ktorá je dekomponovaná na dve menšie matice, *W* s rozmermi *m x r*, *H* s rozmermi *r x n*. Tie lineárnou kombináciou aproximujú pôvodnú maticu *V*. Rozmer *r* je zvyčajne menší ako rozmery *m* alebo *n*, aby dekomponované matice boli menšie a tak sa docielila redukovaná verzia pôvodnej matice [36]. *V* disponuje nezápornými hodnotami a môže reprezentovať čokoľvek. Pri riešení problémov ADT však reprezentuje magnitúdový spektrogram, ktorého rozmer *m* definuje počet frekvenčných rozsahov (angl. *frequency bins*) a rozmer *n* počet časových rámcov (angl. *frames*) v čase. Od toho sa odvíja aj význam matíc *W* a *H*. *W* je vo všeobecnosti matica atribútov, ktoré sa nachádzajú v dátach, t.j. v matici *V*. Vo výskumnej oblasti ADT je definovaná ako matica komponentov (angl. *dictionary matrix*), ktorá reprezentuje zoznam zvukových signálov, v našom prípade bicích a harmonických nástrojov, ktoré sú súčasťou audio nahrávky reprezentovanej magnitúdovým spektrogramom. Nasvedčujú tomu aj jej rozmery, pretože *m* definuje počet frekvenčných rozsahov frekvenčného spektra a *r* počet nástrojov. Z toho mimochodom

vyplýva, že jeden komponent je reprezentovaný vektorom. H je vo všeobecnosti váhovou maticou a pri riešení problémov ADT je definovaná ako aktivačná matica (angl. *activation matrix*), ktorá reprezentuje to, v akom čase bol ktorý nástroj aktívny [37]. Jej rozmery túto intuíciu tiež potvrdzujú, pretože *r* je počet komponentov, kde každý riadok (vektor) prislúcha práve jednému komponentu v matici *W* a rozmer *n* je počet časových rámcov.

To, ako samotná faktorizácia funguje a aké sú najpoužívanejšie typy tohto algoritmu pri riešení problémov ADT, si rozoberieme v nasledujúcich častiach.

4.1. Proces faktorizácie

NMF je iteratívny algoritmus. V každej iterácií dochádza k faktorizácií (úprave) matíc *W* a *H* na základe zvolenej chybovej funkcie (jej význam som opísal v sekcii 3.3), veľkosti chyby, ktorá sa vyráta medzi pôvodnou maticou a jej aproximáciou a spôsobe úpravy matíc. Ako už vieme zo sekcie 3.3, úlohou algoritmu strojového učenia je minimalizovať veľkosť chybovej funkcie optimalizovaním jeho parametrov. Pri použití NMF na to slúžia 2 techniky – metóda projektovaného zostupujúceho gradientu (angl. *projected gradient descent*) a metóda multiplikatívnych aktualizácií (angl. *multiplicative update rules*). Vzhľadom na to, že pri riešení ADT problémov prostredníctvom NMF sa používajú 2 hlavné chybové funkcie – *Euklidovská vzdialenosť* a *KL-divergencia* – tie v nasledujúcich častiach rozoberiem spolu s optimalizačnými technikami a ich odvodenými pravidlami na úpravu matíc [38].

4.1.1. Metóda projektovaného zostupujúceho gradientu (PG)

Vypočítanie gradientu je také isté ako pri metóde zostupujúceho gradientu (vzorec 7, pričom existujú viaceré chybové funkcie) len s tým rozdielom, že po úprave sa hodnoty váh projektujú na zvolenú množinu oboru hodnôt [39]. V našom prípade sa všetky záporné hodnoty aktualizujú na 0 (chceme sa vyhnúť záporným hodnotám). Ako som už spomenul v predošlej sekcii, faktorizácia pozostáva z výpočtu chyby vhodne zvolenou chybovou funkciou a príslušnou úpravou matíc, ktorá je odvodená z derivácie chybovej funkcie. Pri použití *Euklidovskej vzdialenosti*, ktorej vzdialenosť medzi dvoma maticami sa vypočíta pomocou Frobeniovej normy [38]

$$D_{EUC}(X,Y) = \left|\left|X-Y\right|\right|_F = \sqrt{\sum_{i=1}^{M}\sum_{j=1}^{N}\left(x_{ij}-y_{ij}\right)^2}\ [38] \qquad (31)$$

po dosadení našich premenných bude vyzerať nasledovne

$$D_{EUC}(X,WH) = ||X-WH||_F^2 = \sum_m \sum_n (X_{mn} - WH|_{mn})^2 \qquad (32)$$

a úprava matíc *W* a *H* po zderivovaní pri použití PG bude vyzerať takto

$$H \leftarrow H + \eta_H \circ (W^{\mathrm{T}}X - W^{\mathrm{T}}WH) \tag{33}$$

$$W \leftarrow W + \eta_W \circ (XH^{\mathrm{T}} - WHH^{\mathrm{T}}) \tag{34}$$

Symbol ∘ reprezentuje *Hadamardov súčin*, $(\dots)^{\mathrm{T}}$ definuje transponovanú maticu a η je rýchlosť učenia nastavená pre maticu komponentov a maticu aktivácií zvlášť [38].

Pri použití *KL-divergencie*, ktorej všeobecný zápis vyzerá nasledovne

$$D_{KL}(p,q) = \left[p(i)\ log\frac{p(i)}{q(i)} - p(i) + q(i)\right] \tag{35}$$

bude v maticovej podobe pri NMF kalkulácií vyzerať takto

$$D_{KL}(X,WH) = \sum_m \sum_n \left[X_{mn}\ log\frac{X_{mn}}{WH|_{mn}} - X_{mn} + WH|_{mn})\right] \tag{36}$$

z čoho vypočítame gradient v maticiach nasledovne

$$\nabla_w D_{KL}(X,WH) = -\frac{X}{WH}H^{\mathrm{T}} + 1H^{\mathrm{T}} \tag{37}$$

$$\nabla_H D_{KL}(X,WH) = -W^{\mathrm{T}}\frac{X}{WH} + W^{\mathrm{T}}1 \tag{38}$$

$\nabla_{(\dots)}$ definuje výpočet gradientu v danom bode z určitej funkcie [38]. V našom prípade sa gradient vypočíta z *KL-divergencie* v závislosti od hodnôt, ktoré sa nachádzajú v maticiach.

4.1.2. Metóda multiplikatívnych aktualizácií (MU)

Môžeme si všimnúť, že pravidlá úprav matíc pri metóde PG obsahujú matematickú operáciu *odčítanie*. To znamená, že počas procesu faktorizácie sa v maticiach môžu vyskytnúť záporné hodnoty, ktoré sa však projektovaním upravia. Autori Lee a Seung však prišli s nápadom na špeciálnu úpravu matíc pomocou tzv. multiplikatívnych aktualizácií (angl. *multiplicative update rules*), pri ktorých už zo samotnej matematickej formulácie vyplýva, že pri faktorizácií budú matice obsahovať nezáporné hodnoty [36]. Najhoršie čo môže nastať je to, že sa v maticiach vyskytnú nulové hodnoty, ktoré sa však dodatočne projektujú na veľmi malú hodnotu blízku 0. Tým sa predíde nekonečným hodnotám, ktoré by vznikli pri úprave matíc alebo výpočte chybových funkcií, kvôli čomu by algoritmus zdivergoval.

Pri použití *Euklidovskej vzdialenosti*, ktorej vzorce na úpravy matíc pomocou PG už poznáme (vzorce 33, 34), po úprave na aktualizáciu matíc pomocou MU budú vyzerať nasledovne [38]

$$H \leftarrow H \circ \frac{W^{\mathrm{T}}X}{W^{\mathrm{T}}WH} \text{ [38]} \tag{39}$$

$$W \leftarrow W \circ \frac{XH^{\mathrm{T}}}{WHH^{\mathrm{T}}} \text{ [38]} \tag{40}$$

Pri použití *KL-divergencie,* ktorej vzorce na výpočet gradientov v jednotlivých bodoch matíc tiež poznáme (vzorce 37, 38), po úprave na aktualizáciu matíc pomocou metódy MU budú vyzerať takto [38]

$$H \leftarrow H \circ \frac{W^{\mathrm{T}}\frac{X}{WH}}{W^{\mathrm{T}}1} \text{ [38]} \tag{41}$$

$$W \leftarrow W \circ \frac{\frac{X}{WH}H^{\mathrm{T}}}{1H^{\mathrm{T}}} \text{ [38]} \tag{42}$$

1 je jednotková matica s rovnakými rozmermi ako matica, ktorá sa bude upravovať. Môžeme si všimnúť, že vzorce na úpravy matíc pomocou tejto metódy neobsahujú matematickú operáciu odčítania. Je to preto, lebo tento typ aktualizácie matíc bol odvodený z PG vzorcov. Autori to premysleli tak, že úpravy matíc sa vykonávajú automaticky v smere zostupujúceho gradientu a konvergujú k lokálnemu optimu [36, 38].

4.2. Optimalizácia a regularizácia

V závislosti od toho, akým spôsobom sa upravujú matice, je možné aplikovať rôzne optimalizačné alebo regularizačné techniky. V prípade, že sa matice upravujú pomocou metódy PG, môžu sa použiť tie isté vylepšenia, aké som spomenul už pri neurónových sieťach.

Z optimalizačných techník to môže byť napríklad dynamicky znižujúca sa rýchlosť učenia podľa určitých pravidiel (exponenciálne znižovanie, znižovanie na základe počtu iterácií, atď. [40]), využitie metód adaptívnej rýchlosti učenia (*Adam, RMSProp, Adagrad, Adelta*) alebo momentum.

Z regularizačných techník to môže byť napríklad L1 regularizácia, ktorá sa vo výskumnej oblasti ADT s použitím NMF prístupu aplikuje na aktivačnú maticu *H*. Aplikovanie L1 v kontexte NMF nájdeme vo výskumných článkoch pod názvom *sparsity constraint* [41]. Zo sekcie 3.5.1 vieme, že tento typ regularizácie priraďuje významným parametrom nenulové hodnoty, pričom ostatné zníži na nulu. V prípade, že sa aplikuje na aktivačnú maticu, výsledným efektom bude potlačenie šumu a vyfiltrovanie udalostí, v ktorých boli sledované nástroje aktívne.

4.3. Najpoužívanejšie typy maticových faktorizácií pri riešení problémov ADT

Výskum v oblasti ADT ponúka viacero variácií maticových faktorizácií. Všetky sú postavené na tom istom základe, t.j. dekompozícií hlavnej matice na 2 menšie. Odlišnými prvkami sú napríklad úpravy matice, ako často sa upravujú a či sa vôbec upravujú. V nasledujúcich sekciách si rozoberieme základné koncepty a intuíciu rôznych architektúr, z čoho vždy vyvodíme zopár pozitívnych a aj negatívnych dôsledkov.

4.3.1. Obyčajná NMF

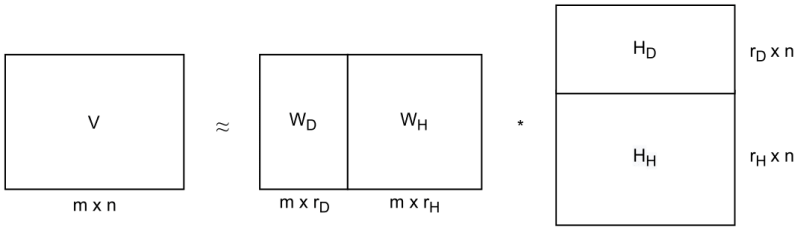
Ide o základný NMF algoritmus, z ktorého sú odvodené všetky jeho vylepšenia a variácie. Funkcia aproximácie je ekvivalentná vzorcu 30 ($V \approx WH$). Aktualizácia matíc sa bude vykonávať podľa toho, aký sa zvolí typ chybovej funkcie a optimalizačnej techniky. To znamená, že úprava matice W bude ekvivalentná jednému zo vzorcov 34, 37, 40 alebo 42 a úprava matice H bude identická jednému zo vzorcov 33, 38, 39 alebo 41 [3]. Komponenty v matici W sú reprezentované vektorom.

Treba si uvedomiť, že použitie základného algoritmu NMF má svoje limity pri riešení komplexnejších problémov ADT. Je to hlavne z toho dôvodu, akými fázami si algoritmus prejde. Spočiatku sa matice inicializujú náhodne s hodnotami veľmi blízkym k 0, aby veľkosť chyby nedosahovala príliš vysoké čísla a jednotlivé parametre rýchlejšie skonvergovali k hodnotám, ktoré v kontexte celého modelu budú považované za optimálne. Teda nikde neurčujeme, aký komponent má modelovať ktorý nástroj. Následne proces faktorizácie prebieha bez akýchkoľvek obmedzení a pravidiel, podľa ktorých by sa dalo určiť, aké komponenty v matici komponentov majú zodpovedať ktorým nástrojom. To znamená, že ak by sme chceli spraviť transkripciu polyfonickej hry bicích za prítomnosti harmonických nástrojov s použitím tohto algoritmu, je vysoko pravdepodobné, že žiaden z komponentov by neobsahoval zvuk len jedného bicieho nástroja, alebo zvuk jedného harmonického nástroja, ale skupinu viacerých.

Ak je problém o niečo jednoduchší, ako napríklad transkripcia monofonických úderov na bicie alebo polyfonickej hry na bicích bez prítomnosti harmonických zložiek a so známym počtom bicích nástrojov, obyčajnú NMF si môžeme dovoliť použiť, pretože poznáme počet nástrojov, t.j. komponentov. Ďalšou z pozitívnych vlastností NMF je jej jednoduchosť a ľahká interpretovateľnosť. To znamená, že po skončení algoritmu vieme celkom jasne zhodnotiť, aké komponenty sa snažila NMF modelovať a v akých časoch boli aktívne [3].

4.3.2. NMF s čiastočne fixnými bázami (PFNMF)

Jedna z variácií NMF vopred inicializuje časť matice *W*, ktorá reprezentuje frekvenčné spektrá bicích nástrojov a zostane fixná alebo sa môže za určitých okolností upravovať počas procesu faktorizácie. Inicializáciu možno spraviť napríklad spriemerovaním spektier z nahrávok izolovaných úderov na bicie nástroje [3]. V zásade úpravy matíc pri rôznych chybových funkciách majú podobný matematický zápis ako pri obyčajnej NMF len s tým rozdielom, že matice *W* a *H* sú rozdelené na 2 časti – v našom prípade pre bicie a pre harmonické nástroje. Pre lepšiu predstavu uvediem vizualizáciu.



Obrázok 22. Vizualizácia algoritmu PFNMF

Potom sa pri použití *Euklidovskej vzdialenosti* budú matice upravovať nasledovným spôsobom

$$H_D \leftarrow H_D \circ \frac{W_D^{\mathrm{T}} X}{W_D^{\mathrm{T}}(W_D H_D + W_H H_H)} \qquad W_D \leftarrow W_D \circ \frac{X H_D^{\mathrm{T}}}{(W_D H_D + W_H H_H) H_D^{\mathrm{T}}} \tag{43}$$

$$H_H \leftarrow H_H \circ \frac{W_H^{\mathrm{T}} X}{W_H^{\mathrm{T}}(W_D H_D + W_H H_H)} \qquad W_H \leftarrow W_H \circ \frac{X H_H^{\mathrm{T}}}{(W_D H_D + W_H H_H) H_H^{\mathrm{T}}} \tag{44}$$

a pri použití *KL-divergencie* takto

$$H_D \leftarrow H_D \circ \frac{W_D^{\mathrm{T}} \frac{X}{W_D H_D + W_H H_H}}{W_D^{\mathrm{T}} 1} \qquad W_D \leftarrow W_D \circ \frac{\frac{X}{W_D H_D + W_H H_H} H_D^{\mathrm{T}}}{1 H_D^{\mathrm{T}}} \tag{45}$$

$$W_H \leftarrow W_H \circ \frac{\frac{X}{W_D H_D + W_H H_H} H_H^{\mathrm{T}}}{1 H_H^{\mathrm{T}}} \qquad H_H \leftarrow H_H \circ \frac{W_H^{\mathrm{T}} \frac{X}{W_D H_D + W_H H_H}}{W_H^{\mathrm{T}} 1} \tag{46}$$

Treba podotknúť, že $(\ldots)_{\mathrm{D}}$ definuje maticu pre bicie nástroje a $(\ldots)_{\mathrm{H}}$ pre harmonické nástroje. Úprava matice W_D (vo vzorcoch 43 a 45) sa môže použiť [37] ale aj nemusí [42] .

Výhodou tohto typu NMF je to, že sa môže na rozdiel od obyčajnej NMF aplikovať na problém transkripcie bicích nástrojov z polyfonickej hudby za prítomnosti harmonickej zložky (ďalej už len DTM). A to preto, lebo sa vopred inicializujú komponenty bicích nástrojov z monofónnych úderov, ktoré sa následne hľadajú v komplexnej audio nahrávke. Vzhľadom

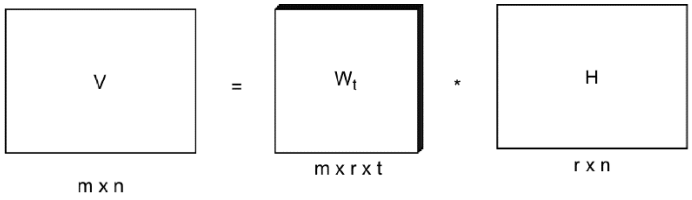
na to, že zložitosť takéhoto problému je oveľa väčšia ako pri transkripcií bicích nástrojov bez prítomnosti harmonických nástrojov (DTD), domnievame sa, že pri použití MU akákoľvek zvolená chybová funkcia má veľa lokálnych miním a hrozí, že algoritmus jednu z nich nájde a v nej aj zostane. Pri použití metódy PG poznáme techniky, ktoré vedia algoritmus vymaniť z lokálneho minima. Preto to v tomto prípade až tak nehrozí. Napriek tomu si myslíme, že aj tak treba opatrne narábať s hyperparametrami rôznych optimalizačných alebo regularizačných techník.

4.3.3. Dekonvolutívna NMF (NMFD)

Predošlé NMF typy hoci fungujú pomerne dobre, časový priebeh vývoju spektra pre daný komponent je ponechaný na aktivačnú maticu a nie maticu komponentov. Čo znamená, že aproximovaná matica len približne vyjadruje to, ako vyzeralo spektrum v čase od začiatku úderu na na nejaký bicí nástroj až po jeho koniec. Preto vznikla variácia NMF, tzv. dekonovolutívna NMF, ktorá sa snaží riešiť tento problém práve tým, že modeluje spektrum komponentu v matici komponentov. Čiže jeden komponent je namiesto vektora reprezentovaný maticou [43]. Funkcia aproximácie potom vyzerá nasledovne

$$V \approx \sum_{t=0}^{T-1} W_t \,.\, H^{t\rightarrow} \tag{47}$$

A vizualizácia takto



Obrázok 23. Vizualizácia algoritmu NMFD

Parameter t definuje počet spektrálnych rámcov, pomocou ktorých zachytáva spektrálno-časový priebeh nástrojov v matici komponentov. Operátor $(.)^{i\rightarrow}$ posúva stĺpce danej matice vpravo s tým, že novo vzniknuté stĺpce majú nulové hodnoty pričom dimenzia matice zostáva zachovaná [43]. Pre lepšiu predstavu si môžeme uviesť príklad, ktorý je podobný tomu z [43]

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} A^{0\rightarrow} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} A^{1\rightarrow} = \begin{bmatrix} 0 & 1 & 2 \\ 0 & 4 & 5 \end{bmatrix} A^{2\rightarrow} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 4 \end{bmatrix} \tag{48}$$

To isté platí pri operátore $(.)^{\leftarrow i}$ s tým rozdielom, že všetko prebieha opačne [43]. Potom pri použití optimalizačnej techniky MU a *Euklidovskej vzdialenosti* budú vyzerať úpravy matíc nasledovne

$$H \leftarrow H \circ \frac{W_t^{\mathrm{T}}(X)^{\leftarrow t}}{W_t^{\mathrm{T}}(WH)^{\leftarrow t}} \quad \text{a} \quad W_t \leftarrow W_t \circ \frac{X(H^{t\rightarrow})^{\mathrm{T}}}{WH(H^{t\rightarrow})^{\mathrm{T}}}, \forall t \in [0 \ldots T-1] \qquad (49)$$

a pri použití *KL-divergencie* takto

$$H \leftarrow H \circ \frac{W_t^{\mathrm{T}}\left[\frac{X}{WH}\right]^{\leftarrow t}}{W_t^{\mathrm{T}}1} \quad \text{a} \quad W_t \leftarrow W_t \circ \frac{\frac{X}{WH}(H^{t\rightarrow})^{\mathrm{T}}}{1(H^{t\rightarrow})^{\mathrm{T}}}, \forall t \in [0 \ldots T-1] \qquad (50)$$

W_t reprezentuje maticu všetkých komponentov v čase *t* [43]. Netreba zabudnúť aj na to, že po posunutí aproximovanej matice (*WH*), ktorá sa v matematických formulách nachádza v menovateli, sa musia nastaviť novovzniknuté stĺpce na veľmi malé hodnoty blízke nule, aby nenastalo delenie nulou.

Dekonvolutívna NMF disponuje takými istými výhodami a nevýhodami, akými obyčajná NMF navyše s jednou výhodou. Tou je, že modelovanie časového vývoja spektra komponentov je ponechané na maticu *W* a zároveň bude vedieť zachytiť viacero informácií, ktoré sa v jednom vektore zachytiť nedajú. To môže pomôcť zmierniť rozdiel medzi aproximovanou a reálnou maticou a tak zlepšiť samotnú rekonštrukciu nahrávok.

4.4. Zhrnutie

Vo všeobecnosti môžeme povedať, že NMF algoritmus a jeho rôzne varianty sú jednoduchým no pomerne schopným algoritmom na transkripciu bicích nástrojov z polyfonickej hudby. Sú ľahko interpretovateľné pretože ich výstupom je matica *W*, z ktorej sa dá jednoducho vyčítať, čo všetko pri modelovaní brala do úvahy a matica *H*, ktorá hovorí len o tom, kedy bol ktorý nástroj aktívny. Tak ako jednoduchosť je výhodou, môže byť aj jeho limitujúcim faktorom, kvôli ktorému sa môže problematickejšie vysporiadať s komplexnejšími nahrávkami. Ďalej sme sa dozvedeli že NMF a NMFD slúžia na riešenie jednoduchších problémov transkripcie (detekcia alebo klasifikácia monofonických úderov na bicie, transkripcia bicích bez harmonických nástrojov) pričom PFNMF slúži na riešenie komplexnejších problémov (transkripcia bicích z polyfonickej hudby za prítomnosti harmonickej zložky).

5. Aktuálny stav automatickej transkripcie bicích nástrojov

V samotných začiatkoch výskumu pri transkripcii bicích nástrojov sa používalo niekoľko metód, ktoré sa zameriavali buď na rozoznávanie vzorov alebo separáciu komponentov. Časom sa prišlo na pokročilejšie metódy a tie sa rozdelili do takzvaných návrhových vzorov. Metódy, ktoré spadajú do týchto kategórii nie sú špecifický určené na riešenie problémov transkripcie bicích nástrojov. Sú veľmi všeobecné a inšpirované z výskumných prác rozpoznávania hlasu, jazyka alebo spracovávania multimediálneho obsahu. Najmodernejšie prístupy riešiace ADT problémy v polyfonickej hudbe rozdeľujeme na 2 skupiny: Prístupy založené na neurónových sieťach a na nezápornej maticovej faktorizácií [3].

5.1. Neurónové siete

V tejto časti opíšem najmodernejšie prístupy neurónových sietí, ktoré riešia problém ADT.

5.1.1. BDRNN 1

Jeden z najmodernejších prístupov v DTM využíva BDRNN. Na vstupe má predspracovaný audio signál so vzorkovacou frekvenciou 44.1 kHz, ktorý je transformovaný do STFT spektrogramu s 1024 frekvenčnými rozsahmi. STFT je vyrátaný pomocou *Hanningovho* okna s veľkosťou 2048 vzoriek pričom veľkosť skoku (angl. *hop size*) je 512 vzoriek. Pri samotnej transkripcii sú použité 3 neurónové siete, kde každá je určená pre jeden nástroj (kopák, rytmičák, hi-hat činela). Každá neurónová sieť pozostáva z troch úplne prepojených vrstiev (angl. *fully-connected*). Každá vrstva disponuje 50 neurónmi. Pri trénovaní je rýchlosť učenia veľká 0.05. Trénovanie sa zastaví, ak sa v priebehu 100 iterácií model nezlepší. Úspešnosť modelu sa rátala pomocou chybovej funkcie *cross-entropy*, ktorá sa rátala z výsledkov aktivačnej funkcie *softmax* na výstupnej vrstve [28].

Pri detekcii aktivácii jednotlivých bubnov použili jednoduchý spôsob na zistenie, ktorý výstup z neurónových sietí v podobe numerických hodnôt je ozajstnou aktiváciou.

$$T = mean(\theta) * \lambda \text{ [28]} \tag{51}$$

T je prahová hodnota, ktorá sa vypočíta z priemeru všetkých rámcov a konštanty λ. Ak aktuálny rámec je považovaný za vrchol a zároveň presahuje prahovú hodnotu, je považovaný za aktiváciu [28].

Pri vyhodnocovaní transkripcie BDRNN z polyfonickej hudby zistili, že celková veľkosť metriky *F-measure* je pomerne vysoká (ale nie najväčšia) oproti modelom, s ktorými sa porovnávala. Najväčšiu spomedzi všetkých má pri rytmičáku. Je dobré to mať na vedomí,

pretože model sa testoval na nahrávkach, v ktorých bubeníci hrali na rytmičák rôznymi technikami. Okrem toho je pre rytmičák typické, že jeho zvuk má širokú frekvenčnú bázu. To znamená, že BDRNN je schopná pomerne dobre klasifikovať komplikované frekvenčné spektrá a hracie techniky. Taktiež BDRNN má najvyššiu hodnotu pri metrike *Recall* spomedzi všetkých modelov, no *Precision* je malá. To znamená, že BDRNN má potenciál, ale aktivačné funkcie by sa museli prečistiť [28] (metriky vysvetlím v kapitole 8).

5.1.2. BDRNN 2

Jedným z ďalších prístupov, ktoré sa snažia o vylepšenie transkripcie v DTM je použitie mechanizmov jemnej pozornosti (angl. *soft attention mechanisms*) a špeciálnej konfigurácie RNN siete s dodatočnými periférnymi prepojeniami. Mechanizmy jemnej pozornosti vytvárajú reprezentáciu, pomocou ktorej sa v dátach dajú sledovať komplexnejšie vzťahy. V tomto prípade boli využité na zachytenie latentných dynamických čŕt v hudbe. Na realizáciu tejto idei použili dvojvrstvovú BDRNN. Nevýhodou tohto riešenia je veľkosť skrytej vrstvy, ktorá limituje množstvo informácií, ktoré môžu byť poslané do výstupnej vrstvy. Pretože na základe množstva informácií vyextrahovaných z latentných čŕt určuje výstupná vrstva konečný výsledok. Tento problém sa im podarilo vyriešiť špeciálnou konfiguráciou RNN s dodatočnými prepojeniami, ktoré smerujú priamo do výstupnej vrstvy. Tým jej zabezpečili prísun väčšieho množstva informácií, ktoré môže brať do úvahy pri výstupe [27].

Obidve architektúry mali na vstupe spektrogram, ktorý bol vytvorený za použitia STFT s veľkosťou *Hanning* okienka $n = 2048$ vzoriek a veľkosťou skoku $\frac{n}{4}$. Samotná architektúra BDRNN pozostávala z už spomínaných 2 vrstiev, pričom obe obsahovali 100 LSTM buniek. Pri trénovaní sa minimalizovala chybová funkcia *cross-entropy* za použitia optimalizačnej techniky *Adam* s rýchlosťou učenia 0.003. Trénovanie sa zastavilo, ak ubehlo aspoň 10 epoch a úspešnosť modelu na validačnej vzorke sa medzi epochami nezlepšila. Oba prístupy využívali dve stratégie pri detekcii aktivácií nástrojov. Prvou je zlepšená technika, ktorá už bola použitá pri obyčajnej BDRNN v [28].

$$T^t = mean(y^{t-\theta}, y^{t+\theta}) \text{ [27]} \tag{52}$$

$$O^t = \begin{cases} 1, & y^t == \max(y^{t-\Omega} : y^{t+\Omega}) \;\&\; y^t > T^t \\ 0, & inak \end{cases} \text{ [27]} \tag{53}$$

Vylepšenie spočíva v tom, že prahová hodnota sa ráta pre každý rámec zvlášť z jeho okolia θ a nie zo všetkých rámcov. Aktuálny rámec O^t je potom považovaný za aktiváciu vtedy, ak jeho hodnota aktivácie y^t prekračuje prahovú hodnotu T^t a zároveň je maximom

spomedzi hodnôt okolitých rámcov. Druhou stratégiou je použitie BDRNN na detekciu aktivácií. Na vstupe dostane výstup z BDRNN, ktorú som už opísal v predošlej sekcii a výstupom druhej BDRNN budú aktivácie jednotlivých nástrojov [27].

Pri vyhodnocovaní modelov skúmali okrem úspešnosti aj ich robustnosť. Pri experimentoch, ktoré sa zameriavali na úspešnosť modelov sa ukázalo, že takto navrhnuté modely sú minimálne porovnateľné s aktuálnymi prístupmi ak nie lepšie. Taktiež tieto modely dobre obstáli aj pri experimentoch, ktoré testovali robustnosť modelov. Pri vyhodnotení, v ktorom boli modely natrénované na polyfonickej hre na bicích a otestované na polyfonickej hre na bicích za prítomnosti harmonických nástrojov, najväčšiu úspešnosť mali 2 modely – konvolučná neurónová sieť a model, ktorý využíval mechanizmy jemnej pozornosti [27]. Ale zas v iných prípadoch boli tieto modely porovnateľné s modernými prístupmi, no nie najlepšie. Taktiež pri porovnaní úspešnosti modelov skúmali obidve stratégie na detekciu aktivácií bicích nástrojov. Ukázalo sa, že pri RNN je prvá stratégia vo všeobecnosti lepšia. V niektorých prípadoch – najmä pri polyfonickej hudbe - použitie druhej stratégie bolo nápomocné, no takých prípadov nebolo veľa. Obzvlášť modely, ktoré boli pri vyhodnocovaní najúspešnejšie, nepoužívali druhú stratégiu na detekciu. To znamená, že spôsob detekcie aktivácií pomocou BDRNN nie je v oblasti ADT veľmi použiteľný. Poprednú pozíciu zastávala CNN, ktorá bola tiež súčasťou výskumu tejto práce, ale tú opíšem v samostatnej sekcii. V konečnom dôsledku sa teda ukázalo, že nové architektúry dosahujú väčšiu úspešnosť oproti modelom, s ktorými ich porovnávali, ale robustnosť modelov o čosi zaostáva aj keď nie je najhoršia [27].

5.1.3. RNN s posunutím anotácií (tsRNN)

Pri riešení jednotlivých pod úloh ADT sa ukázalo, že BDRNN sú vo všeobecnosti lepšie ako obyčajné RNN. Avšak v jednej práci sa ukázalo, že v podstate obyčajná RNN vie dostať rovnaké výsledky ako BDRNN, a to posunutím anotácií aktivácií jednotlivých bicích nástrojov o niekoľko milisekúnd skôr. Toto umožnilo RNN pristúpiť k informáciám pred a po začiatku skutočného nástupu bicích nástrojov. Takúto RNN nazývame *time-shifted* RNN (ďalej už len tsRNN). Ako vstup poslúžili mono audio záznamy so vzorkovacou frekvenciu 44.1 kHz. Prvých 0.25 sekundy v každej skladbe bolo ticho, aby sa predišlo chybám, ktoré by mohli vyplynúť z náhleho začatia hry na viacerých nástrojov hneď na začiatku. Logaritmický spektrogram sa vypočítal z *Hanningovho* okienka o veľkosti 2048 vzoriek. Os reprezentujúca frekvenciu bola transformovaná do logaritmickej stupnice od 20 do 20000 hertzov. Výstupom bol potom spektrogram, ktorý mal 84 frekvenčných rozsahov. Samotná tsRNN

bola dvojvrstvová úplne prepojená neurónová sieť s 50 GRU bunkami. Minimalizovala hodnotu chybovej funkcie *cross-entropy* pomocou optimalizačného algoritmu *RMSProp*. Trénovanie sa zastavilo v prípade, že hodnota chybovej funkcie neklesla počas 10 epóch. Regularizačná technika *dropout* s hodnotou $r_d = 0.3$ bola použitá medzi rekurentnou a výstupnou vrstvou. Výstupná vrstva pozostávala z 3 uzlov s aktivačnou funkciou *sigmoid*, pričom každý uzol charakterizoval jeden bicí nástroj.

Vo fáze detekcie aktivácii jednotlivých bubnov (angl. *peak picking*) použili inú metódu ako predošle na to, aby určili, kedy bod n vo funkcii $F(n)$ je považovaný za vrchol, resp. aktiváciu daného nástroja. Metóda pozostáva z 3 pravidiel [44]:

$$1.\quad F(n) = \max\big(F(n-m), \dots, F(n)\big), \tag{54}$$

$$2.\quad F(n) \geq mean(F(n-a), \dots, F(n)) + \delta \tag{55}$$

$$3.\quad n - n_{lp} > w, \tag{56}$$

kde δ je prahová hodnota, ktorá musí byť prekročená aby mohol byť nejaký vrchol považovaný za aktiváciu. Vrchol je považovaný za aktiváciu vtedy, ak má maximálnu hodnotu v rámci okienka, ktoré je široké $m + 1$ rámcov. Zároveň musí disponovať hodnotou, ktorá je väčšia ako priemerná hodnota plus prahová hodnota v rámci okienka s veľkosťou $a + 1$ rámcov. Okrem toho takýto vrchol musí mať minimálne vzdialenosť $w + 1$ od poslednej aktivácie [28].

Vzhľadom na to, že vo všeobecnosti je ťažšie robiť transkripciu rytmičáku a hi-hat činely ako kopáku, chybová funkcia penalizovala tieto nástroje vo väčšej miere. Aby ich model bol robustnejší, okrem použitých regularizačných techník rozšírili dáta o nové pozorovania (angl. *data augmentation*). Ukázalo sa, že výsledky tsRNN sú porovnateľne s BDRNN v DTD ale aj DTM kontexte, pretože vedia dobre odfiltrovať bicie nástroje od harmonických. Rozšírenie dátovej množiny a použitie GRU tiež výrazne ovplyvnilo úspešnosť modelu [44]. Táto metóda umožňuje použiť jednoduchšiu architektúru RNN úspešnosťou porovnateľnú so zložitejšou BDRNN v systémoch, ktoré spracovávajú dáta v reálnom čase [3].

5.1.4. CNN 1

Keďže cieľom CNN je naučiť sa hľadať v dátach lokálne črty a nie ich časovú následnosť, v jednej z ďalších prác sa ukázalo, že vedia byť flexibilnejšie ako RNN. Vďaka redukovaniu dimenzie a učeniu sa dôležitých čŕt výpočtová náročnosť pri vysoko dimenzionálnych dátach je pre CNN oveľa menším problémom v porovnaní s RNN. CNN sa v tejto práci skladá

z dvoch konvolučných vrstiev využívajúce *max-pooling*, ktoré používajú regularizačné techniky *dropout* a *batch* normalizáciu. Za takouto sekvenciou vrstiev nasleduje 100-neurónová, plne prepojená vrstva, ktorá je napojená na výstupnú vrstvu s 2 neurónmi, na ktorej je použitá aktivačná funkcia *softmax* [27]. Keďže skúmanie využitia CNN v ADT bolo súčasťou práce, ktorá sa zaoberala aj RNN s využitím mechanizmov jemných pozorností a periférnych prepojení (sekcia 5.1.2), na detekciu aktivácií jednotlivých bicích nástrojov, samotné trénovanie siete a na vstupe použili tie isté dáta, parametre a metódy, aké som už uviedol v sekcii BDRNN 2.

Ukázalo sa že, CNN sú flexibilnejšie. Pri porovnávaní CNN s dobre predikujúcimi modelmi ich vedela v niektorých prípadoch prekonať. Tak ako pri BDRNN 2, aj pri vyhodnocovaní CNN sa porovnávali dve stratégie na detekciu aktivácií bicích nástrojov. Použitie druhej, zložitejšej stratégie vždy zhoršilo úspešnosť modelu v porovnaní s prvou stratégiou. To znamená, že využitie druhej stratégie nie je vhodné skombinovať s CNN pri riešení problémov v oblasti ADT [27].

5.1.5. CNN 2

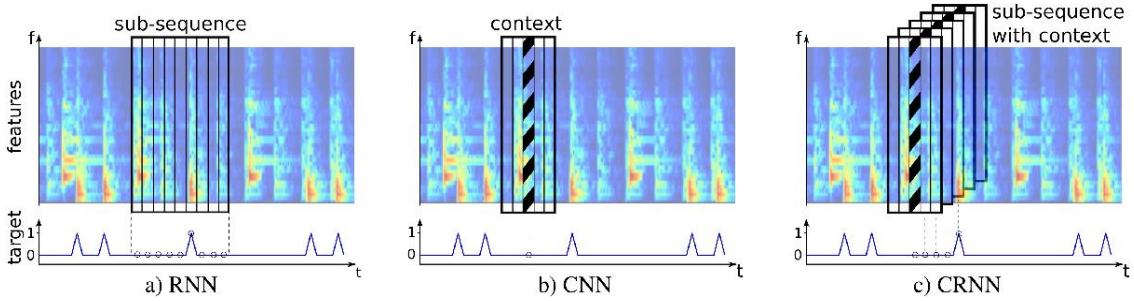
Jedna z ďalších prác skúmala CNN v dvoch ohľadoch. Prvý skúmal jej schopnosť detegovať oblasti aktivácii bicích nástrojov zo spektrogramu. Výsledkom tak bola matica aktivácií, ktorá sa následne poslúžila ako vstup do NMFD. Jej úlohou potom bolo spraviť samotnú transkripciu. Keďže algoritmus bol testovaný na polyfonickej hudbe, úlohou NMFD bolo vyfiltrovať len tie aktivácie, ktoré prislúchali bicím nástrojom. V druhom rade sa skúmala úspešnosť transkripcie, za ktorú zodpovedala CNN. Nepoužila sa len jedna CNN ale rovno tri. A to preto, lebo každá CNN sa zameriavala na iný bicí nástroj. Súčasťou výskumu tejto práce bol aj výber vhodnej chybovej funkcie pri trénovaní a vhodnej vstupnej reprezentácie. Z chybových funkcií porovnávali binárnu krížovú entropiu (angl. *binary cross-entropy*) a priemernú kvadratickú chybu (angl. *mean squared error*). Pri výbere vhodnej vstupnej reprezentácie vyskúšali 4 rôzne veľkosti mel pásiem (angl. *mel-bands*) s 2 rôznymi veľkosťami STFT okienok. Do porovnania zakomponovali aj MCMS. Dokopy tak bolo 9 rôznych vstupných reprezentácií. [45].

Pri vyhodnotení došli k niekoľkým záverom. Pri porovnaní modelu CNN+NMFD s ostatnými prístupmi sa preukázalo, že výsledky sú slabé na to, aby sa aspoň čiastočne vyrovnali súčasne najlepším modelom. Napríklad pri porovnaní s modelom, ktorý je uvedený v sekcií BDRNN 2 sa preukázalo, že mu vôbec nie je schopný konkurovať. Na druhej strane druhý

model, ktorý používal tri konvolučné siete, vykazoval minimálne porovnateľné výsledky s ostatnými modelmi ak nie lepšie (prekonal model uvedený v sekcii BDRNN 2). Ale vzhľadom na to, že každá konvolučná sieť sa sústredí len na jeden nástroj, hrozí jej potenciálne preučenie. To sa dá mať do istej miery pod kontrolou. Z chybových funkcií sa binárna krížová entropia javila ako vhodnejšia voľba. Z rôznych vstupných reprezentácií zas MCMS jednoznačne zaujala popredné miesto [45].

5.1.6. BDRNN3, CNN3, CBDRNN

Plnohodnotná transkripcia bicích by mala brať do úvahy širšie spektrum informácií ako len samotné noty. Od tohto predpokladu sa odrazili autori jednej z ďalších prác, ktorí posunuli výskum v oblasti ADT vpred tak, že do procesu transkripcie zakomponovali užitočné informácie, akými sú doby (angl. *beats*) a prízvučné doby (angl. *downbeats*). Na zrealizovanie experimentu manuálne označkovali dáta týmito informáciami. Tieto dáta potom poslúžili ako vstup do 3 rôznych modelov NN – BDRNN, CNN a CBDRNN (kombinácia konvolučnej a obojsmernej rekurentnej neurónovej siete). Každý typ NN mal 2 rôzne nastavenia – jeden bol jednoduchší a druhý komplexnejší. Teda dokopy ich bolo 6 [30]. Na to, aby sme lepšie pochopili nasledujúce vysvetlenia, uvediem pomocný obrázok:



Obrázok 24. Proces trénovania jednotlivých modelov neurónových sietí [30]

Do BDRNN (Obrázok 24 – ľavá časť) zakomponovali GRU bunky. Jednoduchšia BDRNN sa skladala z 2 vrstiev, pričom na jednej bolo 50 neurónov. Táto sieť sa trénovala po 100-rámcových sekvenciách. Zložitejšia verzia tejto siete sa skladala z 3 vrstiev po 30 neurónov a trénovala sa na 400-rámcových sekvenciách. Pre lepšiu predstavu mriežka s názvom *sub-sequence* znázorňuje dĺžku sekvencie rámcov, na ktorých sa BDRNN trénovala. CNN architektúra (Obrázok 24 – stredná časť) bola o čosi zložitejšia. Skladala sa z 2 blokov – bloku A a B. Blok A disponoval 2 vrstvami s 32 konvolučnými jadrami o veľkosti 3x3 a blok B 2 vrstvami s 64 konvolučnými jadrami tiež o veľkosti 3x3. V oboch blokoch bola použitá batch normalizácia (angl. *batch normalization*) za ktorou nasledoval *max pooling* o veľkosti 3x3 a *dropout* s hodnotou parametra $\lambda = 0.3$. Za blokmi nasledovali 2 plne prepojené vrstvy

s 256 neurónmi. Jediným rozdielom medzi jednoduchou a komplexnou CNN bola veľkosť vstupného kontextu, ktorý sa použije na klasifikáciu jedného rámca. Do jednoduchej CNN išlo na vstupe 9 rámcov a do komplexnej 25 rámcov. Klasifikácia jedného rámca v širšom kontexte je znázornená v strede na obrázku 24. Pri CBDRNN (Obrázok 24 – pravá časť) slúži CNN na predspracovanie spektrogramu. Jej výstup je vstupom pre BDRNN. V jednoduchšej verzii tejto kombinácie architektúr CNN a BDRNN disponovali presne tými istými parametrami, aké som už uviedol pri ich jednoduchších verziách. Pri zložitejšej verzii bol ten rozdiel, že CNN nemala na vstupe 25 rámcov ale len 13 a BDRNN mala na každej vrstve nie 30 neurónov ale 60. Treba podotknúť, že proces trénovania tejto siete bol zložitejší ako predošlé. Vzhľadom na to, že RNN sa trénuje po niekoľko-rámcových sekvenciách, CNN ich najprv musela klasifikovať v rámci širšieho kontextu (Obrázok 24 – pravá časť) [30].

Tak ako v predošlých prácach, aj tu je vstupom spektrogram, ktorý je vytvorený pomocou *Hanningovho* okienka s veľkosťou 2048 vzoriek zo signálu so vzorkovacou frekvenciou 44.1kHz, ktorý je následne zlogaritmovaný. Vstup je pri všetkých modeloch rovnaký. BDRNN majú na výstupnej vrstve 3 alebo 5 neurónov, ktoré majú aktivačnú funkciu *sigmoid*. Čísla 3 alebo 5 závisia od typu experimentu, ktorý vykonávali – buď boli na výstupe len aktivácie bicích (3 neuróny pre rytmičák, kopák a hi-hat činel), alebo spolu s nimi aj dodatočné informácie o dobách a prízvučných dobách (5 neurónov = 3 pre aktivácie bicích + 1 pre doby + 1 pre prízvučné doby). Na trénovanie a validáciu použili 3-stupňovú krížovú validáciu. Tú robili nad dvoma tretinami dát a poslednú si nechali na testovanie. Na optimalizáciu modelov používali *RMSprop* so stochastickou metódou zostupu gradientu s názvom *mini-batch* o veľkosti 8. Rýchlosť učenia sa počas trénovania nemenila a samotné trénovanie sa zastavilo vtedy, keď sa úspešnosť modelu na validačnej vzorke nezlepšila počas 10 epóch [30]. Pri určovaní aktivácií jednotlivých bubnov použili presne tie isté podmienky, o ktorých som sa zmienil v podkapitole 5.1.3 (vzorce 54 - 56).

Pri vyhodnocovaní modelov sa robili 3 druhy experimentov. Prvý sa zameriaval na klasickú transkripciu bicích zo vstupného STFT spektrogramu. Druhý mal na vstupe okrem spektrogramu aj dodatočné informácie o dobách a prízvučných dobách. Chceli zistiť, či dopomôžu ku transkripcii. V treťom experimente mali na vstupe opäť len spektrogram a dodatočné informácie použili na výstupe spolu s anotovanými bicími. Chceli sledovať, či sa dokáže sieť učiť viacero vecí naraz (aktivácie bicích, doby, prízvučné doby). Tento experiment autori odobrili tým, že učenie viacero vecí naraz sa konkrétne v tomto kontexte môže hodiť, keďže údery bicích sú v skladbách prirodzene prepojené s dobami a prízvučnými dobami. 2 a 3

experiment však mohli robiť len na datasete, ktorý disponuje dodatočnými informáciami a preto výsledky modelov z tohto experimentu nemohli použiť na porovnanie s inými modelmi z iných prác. Pri prvom experimente zistili, že komplexnejšia CNN a obidva CBRNN modely prekonávajú model s dovtedy najlepšími výsledkami. Výsledky týchto 3 modelov však nie sú až tak odlišné čo znamená, že obyčajná CNN má schopnosť konkurovať komplexnejšej architektúre NN. Pri 2 a 3 experimente zistili, že len BDRNN vie z dodatočných informácií značne profitovať a je schopná učiť sa viaceré veci naraz. Pri porovnaní výsledkov všetkých 3 experimentoch sa tento model najviac zlepšil v transkripcii spomedzi ostatných. Pri experimente 2 a 3 sa o čosi zlepšila aj jednoduchšia verzia CBDRNN, no nejedná sa až o taký veľký rozdiel ako pri BDRNN. Ostatné modely však nie sú zanedbateľné, pretože napr. dodatočné informácie pri experimente 2 vedeli zlepšiť transkripciu aj jednoduchej CNN a komplexnej CBDRNN. Vo všeobecnosti sú informácie o dobách a prízvučných dobách pri transkripcii nápomocné [30].

5.1.7. CNN 3 (Prototypická sieť)

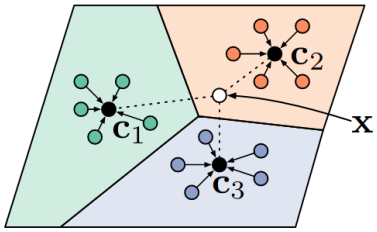
V práci, ktorá sa zaoberala riešením problému ADT pomerne nedávno, bola snaha použiť CNN s trochu odlišným prístupom učenia a aplikácie ako predošlé architektúry neurónových sietí. Na aplikačnej úrovni oproti predošlým prácam je rozdiel v tom, že autori sa zamerali na transkripciu techník hry na bicie nástroje. Spôsob učenia ich neurónovej siete bol na rozdiel od predošlých prístupov založený na báze tzv. *metric learning*.

Je to spôsob učenia, pri ktorom sieť kóduje vstupy do latentnej reprezentácie (angl. *embeddings*) tak, aby sémanticky príbuzné reprezentácie boli v priestore čo najbližšie pri sebe (variancia v rámci jedného zhluku je malá) a zároveň odlišné v dostatočnej vzdialenosti (variancia medzi zhlukmi je veľká). Názov *metric learning* je odvodený z toho, že namiesto toho, aby sa sieti vopred definovalo, akú vzdialenostnú metriku má použiť pri rátaní rozdielov medzi rôznymi reprezentáciami vstupov (napr. euklidovská vzdialenosť, kosinusová vzdialenosť) čo samozrejme vplýva na jej učenie, túto metriku sa sieť naučí sama na základe špecifickosti problému vyplývajúceho z trénovacích dát.

Výhodou tohto prístupu je to, že počas učenia sa sieť učí diskriminatívny priestor, ktorý je natoľko zakrivený, aby jednak vedel dostatočne dobre rozpoznávať odlišnosti medzi skupinami dát, ale aby bol zároveň od týchto tried nezávislý (angl. *class-agnostic*). To znamená, že v prípade výskytu takej skupiny v testovacej vzorke dát, ktorá sa nevyskytovala v trénovacej vzorke, sieť by mal byť schopná rozpoznať jej odlišnosť od ostatných tried a priradiť

jej takú latentnú reprezentáciu, ktorá v priestore bude ďaleko od ostatných, "*už poznaných*" tried, ale jej prvky budú blízko seba [46]. Má to však jeden malý háčik. Na to, aby sieť mohla odlíšiť nepoznanú triedu, potrebuje jej zopár vzoriek na to, aby si vedela vytvoriť jej reprezentáciu, resp. prototyp. V súčasnosti sa experimentuje práve s počtom vzoriek, ktorý je potrebný na vytvorenie prototypu triedy, napríklad *Few-Shot learning* (stačí 2-5 vzoriek) [47], *One-shot learning* (stačí 1 vzorka) [48], alebo *Less than One-shot learning* [49].

V najmodernejšom prístupe ADT, v ktorom robili automatickú transkripciu hracích techník bicích nástrojov sa zaoberali prvým typom – *Few-shot learning*. Neurónovej sieti, ktorá je použitá v tomto prístupe, sa hovorí aj prototypová sieť (angl. *Prototypical network*), keďže jej úlohou je učiť sa odlišovať jednotlivé prototypy tried, resp. hracích techník.



Obrázok 25. Prototypy sú rátané ako priemery zopár latentných reprezentácií danej triedy [50]

Taktiež jej spôsobu učenia sa hovorí epizodické trénovanie (angl. *episodic training*), pretože učenie prebieha po epizódach. Jedna epizóda rieši tzv. *C-way K-shot* klasifikačný problém. To znamená, že sa náhodne vyberie *C* tried (hracích techník), pre každú triedu sa náhodne vyberie *K* vzoriek (vzorka = výsek v skladbe v ktorom je úder danou hracou technikou), tie sa pomocou CNN zakódujú do diskriminatívneho priestoru a spriemerovaním takýchto latentných reprezentácií sa vytvoria prototypy jednotlivých hracích techník (body c_1, c_2, c_3 na obrázku 25). Následne sa pre každú hraciu techniku vyberie ďalších *N* vzoriek, z ktorých sa tiež vytvoria latentné reprezentácie a následne sa vypočítajú vzdialenosti medzi nimi a všetkými prototypmi hracích techník (bod *x* na obrázku 25). Podľa toho sa potom ráta chybová funkcia [47].

V tomto prístupe použili *10-way 5-shot* epizodické trénovanie. Pri predikcii tried používali aktivačnú funkciu *softmax* na vypočítané vzdialenosti. Tieto vzdialenosti však boli v záporných hodnotách, aby *softmax* funkcia priraďovala väčšiu pravdepodobnosť tým vzdialenostiam, ktoré sú menšie. Úlohou chybovej funkcie potom bolo minimalizovať zápornú hodnotu logaritmu pravdepodobností, ktoré boli vypočítané zo vzdialenosti k očakávaným triedam. Na vstupe používali logaritmizovaný Mel spektrogram so 128 mel bankami vytvorenými

z *Hanningovho* okienka o veľkosti 2048 vzoriek a skoku o veľkosti 441 vzoriek so vzorkovaciou frekvenciou 44.1 kHz. Ich CNN pozostávala zo 4 konvolučných blokov, pričom každý bol zložený zo 128 konvolučných jadier, batch normalizácie, ReLu aktivácie a *max-pooling* operácie. Na konci použili ešte jeden *max-pooling* naprieč časovou dimenziou aby vo výsledku dostali jeden vektor s konkrétnou veľkosťou (1024). Tento vektor reprezentoval latentnú reprezentáciu určitého výseku skladby. Sieť trénovali pomocou optimalizačného algoritmu *Adam* s rýchlosťou učenia 0,001. V inferenčnej fáze (vyhodnocovanie na testovacej množine dát) používali na detekciu aktivácií pravidlá, ktoré sme už uvádzali (vzorce 54, 55 a 56 v sekcii 5.1.3).

Pri vyhodnocovaní prišli autori na niekoľko záverov. Prvým je ten, že pri takomto prístupe sieť dokázala pomerne dobre klasifikovať triedy techník nástrojov, ktoré nevidela v trénovacej fáze. Tuto mieru úspešnosti chceli kvantifikovať a preto otestovali, ako dobre model klasifikoval triedy s rôznou početnosťou v dátach v inferenčnej fáze, ktoré sa nenachádzali v trénovacej množine. Ukázalo sa, že je jedno, ako veľmi sú zastúpené triedy v testovacej množine, v prípade, že sa sieť naučí dostatočne pestrý diskriminatívny priestor, úspešnosť modelu by to pri výskyte akejkoľvek triedy nemalo ovplyvniť. Taktiež výkonnosť modelu bola pomerne stabilná naprieč testovacími množinami dát, ktoré pochádzali z rôznych databáz, v porovnaní s najmodernejšími prístupmi neurónových sietí.

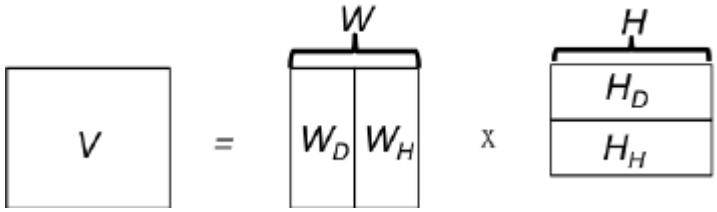
5.2. Nezáporná maticová faktorizácia

Vzhľadom na to, že sa v ADT stretneme s riešeniami, ktoré používajú rôzne varianty NMF [3], v nasledujúcej časti opíšem najpopulárnejšie prístupy.

5.2.1. PFNMF 1

Jedna z prác sa zaoberala algoritmom PFNMF. Z časti 4.3.2 vieme, že počas faktorizácie sa aktualizujú len niektoré komponenty matíc, pričom zvyšok sa nemení. Aby sme si to vedeli lepšie predstaviť v kontexte tejto práce, nižšie uvádzam znázornený konceptuálny návrh algoritmu [42]:

Cieľom algoritmu je aproximovať maticu V, ktorá reprezentuje spektrogram audio signálu, pomocou matíc W a H. Matica W charakterizuje spektrálne črty jednotlivých hudobných nástrojov, pričom matica H sa zase zameriava na ich udalosti, kedy a ako boli aktívne. Na obrázku 26 vidíme, že matice sú zložené z dvoch častí. Matica W_D znázorňuje spektrálne čr-



Obrázok 26. Konceptuálny návrh algoritmu PFNMF 1 [42]

ty bicích nástrojov, W_H harmonických nástrojov. Matica H_D reprezentuje aktivácie jednotlivých bicích nástrojov, H_H harmonických nástrojov. Aby sme v nasledujúcich častiach rozumeli niektorým zápisom a vysvetleniam, najprv si objasnime dimenzie matíc. W_D má rozmery $m\ x\ r_D$, pričom m reprezentuje frekvenčné spektrum nástrojov a r_D počet bicích nástrojov. W_H má rozmery $m\ x\ r_H$. m už poznáme a r_H označuje počet harmonických nástrojov. H_D má rozmery $r_D\ x\ n$, kde r_D už poznáme a n je počet časových rámcov. H_H disponuje rozmermi $r_H\ x\ n$. Keď už rozumieme dimenziám matíc, teraz uvedieme postup algoritmu [42]:

1. Najprv sa vytvorí matica W_D
2. S vhodne určeným r_H sa vytvoria matice W_H, H_H a následne H_D
3. Matice W_D a W_H sa normalizujú
4. Matice W_H, H_H a H_D sa aktualizujú podľa chybovej funkcie *KL-divergence*, ktorá sa minimalizuje pomocou príslušnej opt. metódy
5. Vypočíta sa hodnota chybovej funkcie
6. Kroky 3 - 5 sa budú opakovať dovtedy, pokým výsledná matica neskonverguje

Treba poznamenať, že vytvorenie matice W_D, ktorá disponuje spektrálnymi vlastnosťami jednotlivých bicích nástrojov, v kroku 1 nie je hocijaké. Táto matica sa vytvorí z nahrávok monofonickej hry, v ktorej sa hrá na každom bubne separátne. A to tak, že každý spektrálny komponent sa vypočíta ako medián zo všetkých úderov pre daný bicí nástroj z týchto nahrávok. Táto matica sa potom počas procesu faktorizácie na polyfonických nahrávkach nemení.

Matica V, ktorú sa snažia matice W a H lineárnou kombináciou aproximovať, je vytvorená pomocou STFT, ktorá sa aplikovala na audio signál s použitím *Hanningovho* okienka o veľkosti 2048 a veľkosťou skoku 512. Po aplikovaní faktorizácie je pre nás najdôležitejšia

matica H_D – tá obsahuje potenciálne aktivácie jednotlivých bicích nástrojov. Na to, aby sa zistilo, ktoré z nich sú skutočnými aktiváciami, aplikuje sa na ňu tzv. *filter mediánov* (angl. *median filter*) [42].

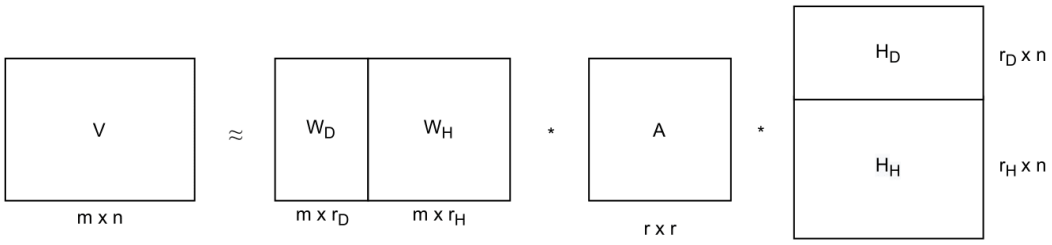
$$G(t) = \lambda + Q(t) \text{ [42]} \tag{57}$$

G je prahová hodnota, ktorá sa mení v závislosti od času. Q je funkcia, ktorá vypočíta hodnotu mediánu z predošlej, vymedzenej časti signálu. V tejto práci dĺžka vymedzenej časti signálu bola 0.1 sekundy. λ je konštantná hodnota, ktorá sa vypočíta ako 0.12-násobok maximálnej hodnoty signálu aktivačnej matici pre daný nástroj [42].

Pred vyhodnocovaním tohto prístupu najprv museli zistiť, akú hodnotu majú zvoliť pre premennú r_H. Totiž ak nie je dané, koľko harmonických nástrojov sa nachádza v nejakej skladbe, táto hodnota sa musí odhadnúť, keďže vďaka nej má matica W_H čo najlepšie modelovať harmonickú zložku hudby. Jej hodnota však nemôže byť hocijaká, lebo príliš veľká alebo príliš malá hodnota znižujú schopnosť modelovať harmonické nástroje a tak aj celkovú úspešnosť modelu. Postupným zväčšovaním tejto hodnoty našli jej optimálnu hodnotu $r_H = 10$. Pri vyhodnotení sa ukázalo, že úspešnosť tohto prístupu je síce o niečo horšia, ale stále porovnateľná s najmodernejšími prístupmi. Výhodou je, že vďaka „predtrénovaniu“ matice spektrálnych komponentov pre bicie nástroje sú výsledky ľahko interpretovateľné. Taktiež správnym nastavením parametra r_H algoritmus dokáže modelovať ľubovoľnú polyfonickú hudbu. Hoci je tento algoritmus oveľa menej výpočtovo náročný ako najmodernejšie prístupy, napriek tomu sú jeho výsledky s nimi porovnateľné. V tomto kontexte môžeme teda hovoriť o robustnosti algoritmu, pretože je použiteľný v reálnych scenároch, kde je k dispozícií len obmedzené množstvo trénovacích dát, ktoré sú odlišné od testovacích a aj napriek tomu dokáže preukazovať pomerne dobré výsledky [42].

5.2.2. PFNMF 2

Autori, ktorí sa podieľali na algoritme PFNMF v [42], hneď vzápätí spravili jeho vylepšenú verziu. Aby sme mu lepšie porozumeli, uvediem pomocný obrázok.



Obrázok 27. Konceptuálny návrh algoritmu PFNMF 2 [37]

Môžeme si všimnúť, že obrázok 27 je takmer identický s obrázkom 26. No tento má navyše maticu A. Jej úlohou je vyvážiť bicie a harmonické nástroje tak, aby bicie mali o niečo väčšiu váhu ako harmonické nástroje. Okrem tejto matice sa autori pokúsili vylepšiť algoritmus tým, že maticu W_D aktualizovali. To však robili dvoma spôsobmi [37].

V prvom sa odrazili od predpokladu, že aj keď W_H modeluje frekvenčné spektrá harmonických nástrojov, niektoré komponenty môžu obsahovať aj niečo z bicích. Teda disponovali by komplementárnymi informáciami k bicím. Preto sa môže stať, že niektoré udalosti bicích budú mať aktiváciu v H_H a H_D zároveň. Tomu predídu tak, že budú hľadať potenciálne korelácie medzi aktiváciami H_H a aktiváciami každého bicieho nástroja v H_D. Ak potenciálna korelácia presiahne určitú prahovú hodnotu, táto udalosť je označená za významnú koreláciu. Na základe počtu a veľkosti jednotlivých korelácií sa potom upraví matica W_D. Tá potom bude lepšie modelovať bicie nástroje [37].

Druhý spôsob bol o niečo jednoduchší. Najprv prebehla maticová faktorizácia presne podľa postupu, ktorý je uvedený v sekcii 5.2.1. Teda W_D sa počas procesu nemenila. Počas tohto procesu sa matica H_D nastavila tak, aby čo najpresnejšie určovala aktivácie bicích. Po ňom sa spustil druhý proces faktorizácie. Ten nechal maticu H_D zachovanú a menili sa W_H, W_D a H_H. Vďaka nemu matica W_D mohla ešte lepšie modelovať jednotlivé bicie nástroje [37].

Pri vyhodnotení modelov autori dospeli k niekoľkým záverom. Prvým bola významnosť váhovej matice A, ktorá ovplyvňuje robustnosť modelu. Ako sme v predošlej sekcii 5.2.1 uviedli, že od hodnoty parametra r_H závisí robustnosť a aj celková úspešnosť modelu, matica A zmierňuje váhu tohto parametra. Teda aj pri väčšej hodnote, ktorá nemusí byť optimálna, je úspešnosť modelu približne rovnaká alebo dokonca aj lepšia. Druhým zistením je to, že pri použití metriky *F-mesaure* dodatočná adaptácia matíc prekonáva takmer identický prístup opísaný v sekcii 5.2.1, ktorý adaptáciu nepoužíva. Hoci sa nejedná o veľký rozdiel, stojí to za zmienku. Detailnejší prieskum výsledkov však ukázal, že adaptácia zlepšuje metriku *Precision*, pričom zhoršuje *Recall*. Odôvodňujú to tým, že algoritmus sa snaží nájsť čo najlepšiu reprezentáciu zvuku bicích na trénovacích dátach, ktorý sa však môže značne líšiť od zvuku v testovacích dátach [37].

5.2.3. SANMF

Jeden z ďalších prístupov NMF sa snažil dynamicky ovplyvňovať mieru, v rámci ktorej sa mohla upravovať matica spektrálnych komponentov bicích počas procesu faktorizácie. Tak ako pri algoritmoch PFNMF 1 a PFNMF 2, tak aj v tomto prípade predchádzala procesu

faktorizácie trénovacia fáza, počas ktorej sa inicializovali vektory jednotlivých bicích nástrojov v matici spektrálnych komponentov z nahrávok s izolovanými údermi [51]. Pre lepšie pochopenie celého algoritmu najprv uvediem dva pomocné vzorce, ktoré mi neskôr pomôžu bližšie objasniť to, čo sa autori snažili docieliť.

$$B = \alpha \,.\, B_p + (1 - \alpha\,) \,.\, B \;\; [51] \tag{58}$$

$$\alpha = \left(1 - \frac{k}{K}\right)^{\beta} \; [51] \tag{59}$$

B je matica spektrálnych komponentov bicích aktuálnej iterácie pri faktorizácií. B_p je inicializovaná matica spektrálnych komponentov bicích. K je celkový počet iterácií procesu faktorizácie, ktorý sa má vykonať, k je aktuálny počet iterácií, β je mocnina, ktorej hodnota je nastavená manuálne a zostáva konštantná počas behu algoritmu, α je meniaca sa premenná, ktorej hodnota výrazne závisí od pomeru aktuálneho k celkovému počtu iterácií procesu faktorizácie (vzorec 59). Vzorec 58 vyjadruje spôsob, akým sa upravuje matica komponentov v každej iterácií faktorizácie. Z neho môžeme vyčítať, že matica spektrálnych komponentov bicích (B) sa bude odlišovať od jej počiatočného stavu (B_p) až v neskorších iteráciach algoritmu kvôli α. Tá totižto spôsobuje to, že úprava spektier nástrojov, t.j. ich nové hodnoty sa počas faktorizácie pri prvých iteráciách nebudú veľmi líšiť od počiatočného stavu. A to preto, lebo hodnota α je spočiatku veľmi blízka číslu 1. Čím sa však bude počet iterácií faktorizácie zvyšovať, veľkosť α sa bude blížiť k hodnote 0, spektrá nástrojov budú môcť byť upravované o hodnoty s väčšou odchýlkou od počiatočného stavu a matica spektrálnych komponentov sa na konci algoritmu bude môcť viac líšiť od jej počiatočného stavu [51].

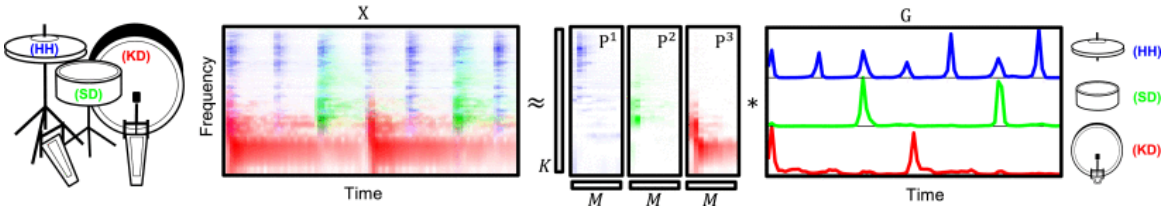
Na vstupe sa spracovával STFT spektrogram vytvorený s veľkosťou skoku 512 vzoriek z audio signálu so vzorkovacou frekvenciou 44.1 kHz. Tak ako vždy, po vykonaní algoritmu jedna z matíc obsahuje potenciálne aktivácie jednotlivých bicích nástrojov. Na to, aby sa zistilo, ktoré z nich sú skutočnými aktiváciami, autori práce použili špecificky prístup detekcie aktivácií s adaptívnou prahovou hodnotou, aby sa čo najviac predišlo detekcií chybných aktivácií.

Algoritmus bol vyhodnotený na 3 súboroch dát. Prvý pozostával z polyfonickej nahrávky akustických bicích a zvyšné 2 boli vytvorené pomocou 2 počítačových programov. Pri porovnaní úspešnosti transkripcie na všetkých dátach sa preukázalo, že algoritmus dosahoval vysoké hodnoty metriky *F-measure* pri dátach s akustickými bicími a jednom súbore dát s umelo vytvorenými bicími. Takýto výsledok autori neočakávali, keďže si mysleli, že práve

reálne prostredie bude predstavovať pre algoritmus najväčší problém. Paradoxne s ním sa najlepšie popasoval. Dôvodom je práve tento špecifický prístup k algoritmu NMF, ktorý priniesol neočakávaný, ale pritom pozitívny výsledok. Pri evaluácií sa porovnával tento nový prístup NMF s klasickou NMF a NMF s fixnou maticou spektrálnych komponentov bicích nástrojov počas celého procesu faktorizácie (tzv. *FNMF* [3]). Ukázalo sa, že výsledky nového prístupu k NMF, t.j. SANMF a FNMF sa veľmi neodlišujú. Autori tejto práce to dávajú za vinu nedostatočnej variabilite dát. Predpokladajú, že ak by bola väčšia, výsledky FNMF by boli výrazne horšie. SANMF je jeden z ďalších prístupov k NMF, ktorý nepotreboval veľké množstvo trénovacích dát na to, aby vedel konkurovať súčasným moderným prístupom. Výhodou tohto prístupu je jeho aplikovateľnosť v systémoch, ktoré pracujú v reálnom čase.

5.2.4. NMFD

Ďalší výskum v NMF ukázal, že modelovanie vývoja signálu v čase pre bicie nástroje nemusí byť zahrnuté v aktivačnej matici. Túto vlastnosť sa v [52] pokúsili zakomponovať do matice, ktorá disponuje spektrálnymi črtami. A to tak, že každý nástroj je modelovaný nie vektorom, ale maticou, resp. spektrogramom. Ten zachytával spektrálno-časový vývin nástroja a aktivačná matica bola veľmi riedka. Jej úlohou bolo zachytávať len impulzy aktivácií, nie ich dynamiku [3]. Pre lepšiu predstavu uvediem obrázok .



Obrázok 28. Koncept algoritmu NMFD [3]

Na obrázku 28 vidíme, že matica X reprezentuje výsledný spektrogram, G obsahuje aktivácie a P disponuje spektrálnymi črtami nástrojov. Treba podotknúť, že matica P nie je dvojdimenzionálna ale trojdimenzionálna, $P \in \mathbb{R}_{\geq 0}^{KxRxM}$. K je frekvenčné spektrum nástrojov, R je počet sledovaných nástrojov a M je veľkosť spektrálnych rámcov (čím väčšie M, tým sa spektrum snaží modelovať dlhší časový úsek pre daný nástroj). Táto zmena dimenzie ovplyvnila aj vzorec lineárnej kombinácie matíc. Jej matematický zápis vyzeral nasledovne [3].

$$X \approx \sum_{m=0}^{M-1} P_m \,.\, G^{m\rightarrow} \text{ [3]} \qquad (60)$$

Teda nejedná sa o obyčajnú lineárnu kombináciu ale o zložitejšiu, kde sa kombinujú nie vektory, ale matice (spektrogramy) spolu s vektormi aktivácií. Autori tohto výskumu skúšali pre tento algoritmus dve rôzne varianty vstupov. Na vstupe mali audio signál so vzorkovacou frekvenciou 44.1 kHz, ktorý transformovali na STFT spektorgram a Mel spektrogram, pričom veľkosť okienka bola 2048 vzoriek. Pri faktorizácií matíc sa snažili minimalizovať rozdiel medzi maticou X a jej aproximáciou, ktorú dostali lineárnou kombináciou matíc P (matica komponentov) a G (matica aktivácií) použitím chybovej funkcie *KL-dievergencia* [52].

Pri vyhodnotení algoritmu došli k niekoľkým záverom. Ak bola veľkosť spektrálnych rámcov pre jednotlivé nástroje príliš veľká a počas faktorizácie sa upravovala aj matica P, jednotlivé spektrálne rámce nástrojov v matici P zachytávali viac ako len jeden úder [3, 52]. Algoritmus testovali len na audio nahrávkach, ktoré obsahujú bicie bez harmonických nástrojov. Tieto audio nahrávky boli rozdelené do 3 skupín. V prvej boli nahrávky, v ktorých zneli polyfónne rytmičák, hi-hat a kopák. V druhej boli v nahrávkach zakomponované viaceré nástroje z bicej súpravy. Tretia skupina obsahovala nahrávky, kde opäť zneli viaceré nástroje bicích pričom na rytmičáku a hi-hat činele sa hralo rôznymi technikami. Autori zistili, že algoritmus funguje dobre na jednoduchších nahrávkach ale komplexnejšie (viacero nástrojov, rôzne techniky) spôsobujú problémy. Pri porovnaní výkonnosti algoritmu sa zistilo, že ak bol na vstupe Mel spektrogram, rýchlosť NMFD bola 12-krát väčšia ako pri použití STFT spektrogramu [52].

5.3. Porovnanie

V tejto časti uvediem porovnanie úspešnosti jednotlivých prístupov, na ktorom si dali prácu autori z prehľadového článku a ktorých väčšinu som spomenul v analýze [3]. Taktiež sa zmienim o tom, aké evaluačné stratégie a aké datasety používali pri vyhodnocovaní úspešnosti modelov.

V tabuľke 1 môžeme vidieť porovnanie úspešnosti jednotlivých modelov podľa metriky *F-measure*. Vyhodnocovanie sa robilo na dátovej množine *ENST-Drums*, na ktorej sa zvyčajne vyhodnocujú modely, ktoré robia transkripciu bicích v DTM [53]. Aby sme tabuľke porozumeli, treba si najprv ujasniť neznáme pojmy ako *Random, Subset, Cross(DTD), Cross(DTP)*. Pomôžeme si tabuľkou z [3], ktorú upravíme na vyhodnocovanie modelov v DTM.

Tabuľka 1. Porovnanie úspešnosti modelov pri rôznych evaluačných stratégiach z článku [3]

	Vyhodnotenie (Priemerná F-measure)				**Zaujímavé štatistiky**			
Architek-túra	Ran-dom	Sub-set	Cross(DTD)	Cross(DTP)	Max-Min	Avg Max-Min	Std	Avg Std
RNN	0,752	0,673	0,648	0,662	0,104	0,128	0,047	0,055
tanhB	0,778	0,710	0,655	0,730	0,123		0,051	
RELUts	0,784	0,695	0,675	0,700	0,109		0,048	
lstmpB	0,777	**0,715**	**0,690**	**0,741**	0,087		0,037	
GRUts	**0,820**	0,687	0,605	0,678	0,215		0,090	
SANMF	0,691	0,664	0,640	0,674	0,051	0,070	0,021	0,032
NMFD	0,675	0,660	0,553	0,658	0,122		0,056	
PFNMF	0,716	0,682	0,640	0,707	0,076		0,034	
AM1	0,733	0,682	**0,690**	0,717	0,051		0,024	
AM2	0,697	0,707	0,655	0,704	0,052		0,024	

Pri evaluačnej stratégií *Random* sa náhodne rozdelila množina nahrávok (DTM) na 70%, na ktorej sa modely natrénovali, 15% na ktorej sa zvalidovali a zvyšných 15%, na ktorých sa otestovali. Pri evaluačnej stratégií *Subset* zas rozdelili dátovú množinu na 3 skupiny. Na 2 skupinách sa modely natrénovali, na jednej polovici z tretej skupiny sa zvalidovali a na druhej

Tabuľka 2 Vysvetlenie evaluačných stratégií uvedených v článku [3]

Evaluečná stratégia	**Trénovanie**	**Validácia**	**Testovanie**
Random	70% D-DTM	15% D-DTM	15% D-DTM
Subset	D-DTM-1, D-DTM-2	50% D-DTM-3	50% D-DTM-3
	D-DTM-1, D-DTM-3	50% D-DTM-2	50% D-DTM-2
	D-DTM-2, D-DTM-3	50% D-DTM-1	50% D-DTM-1
Cross(DTD)	70% D-DTD	15% D-DTD	100% D-DTM
Cross(DTP)	70% D-DTP	15% D-DTP	100% D-DTM

polovici otestovali. Tam je viacero variant, ktoré môžu nastať. Všetky sú spísané v tabuľke 2. Posledná evaluačná stratégia *Cross(...)* modely trénovala, validovala a testovala na dvoch odlišných množinách nahrávok. Pri tomto type vyhodnotenia sa spomedzi ostatných najviac testovala flexibilita modelov. *Cross(DTD)* používa ako trénovaciu množinu nahrávky, kde znie polyfonická hra na bicích bez prítomnosti akýchkoľvek dodatočných perkusijných alebo harmonických nástrojov. *Cross(DTP)* zas používa ako trénovaciu množinu nahrávky, kde sa hrá polyfonicky na bicích s dodatočnými perkusijnými nástrojmi [3].

Z vyhodnotenia teda usudzujeme, že hoci sú modely neurónových sietí úspešnejšie, sú veľmi citlivé na trénovaciu množinu. Ak sa čo najviac podobá testovacej množine, výsledky sú dobré. V opačnom prípade ich úspešnosť razantne klesá. NMF algoritmy sú o niečo stabilnejšie. V tabuľke 1 si môžeme všimnúť, že odchýlka úspešnosti NMF modelov (stĺpce *std* a *Avg std*) v závislosti od trénovacej množiny je v priemere o niečo menšia. Taktiež jednotlivé hodnoty ale aj samotný priemer rozdielu medzi maximálnou a minimálnou úspešnosťou (stĺpce *Max-Min* a *Avg Max-Min*) pre NMF prístupy je zásadne nižší. Tomu zodpovedá vlastnosť robustnosti, ktorou disponujú určite viac ako neurónové siete. Na druhej strane ich úspešnosti sa nie vždy vyrovnajú neurónovým sieťam ale za to sú určite porovnateľné.

5.4. Zhrnutie

Z analýzy jednotlivých prístupov použitých v ADT môžeme vyvodiť niekoľko záverov. Modely neurónových sietí vo všeobecnosti dosahujú vyššie hodnoty metrík *F-measure* a *Recall* ako NMF algoritmy, no za to majú malé hodnoty pri metrike *Precision*. To znamená, že po transkripcii by trebalo dodatočne ešte prečistiť aktivácie. Taktiež sme si mohli všimnúť z vyhodnotenia, že neurónové siete sú menej flexibilné ako NMF algoritmy. Totižto sú veľmi citlivé na to, aký je rozdiel medzi trénovacou a testovacou množinou z hľadiska kontextu. Jednou zo zaujímavosti bolo, že napriek jednoduchosti, architektúra tsRNN vedela konkurovať pri transkripcii zložitejšiemu modelu BDRNN. Taktiež sa ukázalo, že CNN sú flexibilnejšie ako RNN modely. Pri použití iných vstupných reprezentácií ako napr. MCMS, CNN dokázali dosahovať lepšie výsledky ako pri použití STFT.

NMF algoritmy sú ľahko interpretovateľné. Jednou z ich výhod je, že aj keď sú výpočtovo menej náročné ako NN, výsledky sú s nimi porovnateľné. Nie sú závisle od trénovacej množiny a ako sme už spomenuli, sú oveľa viac flexibilné. Na druhej strane ich jednoduchosť je zároveň limitujúcim faktorom. Zložitejšie algoritmy ako NMFD majú o niečo väčší problém ako NN architektúry pri riešení komplexnejších problémov v ADT. Taktiež je tento prístup menej využívaný v systémoch, ktoré potrebujú robiť real-time transkripciu, pretože všetky NMF varianty pracujú iteratívne v inferenčnej fáze (v čase predikcie), pričom NN prístupy v inferenčnej fáze robia len dopredné šírenie.

5.5. Výskumné problémy

Jedným z našich prínosov do výskumu v oblasti ADT by mohlo byť vyskúšanie ďalších konfigurácií metód PFNMF2 a NMFD, pretože autori nikde neuvádzajú, že by skúšali metódu PG na úpravu matíc alebo použitie obmedzenia riedkosti (angl. *sparsity constraint*). Pri

NMFD ju síce použili ale pri PFNMF2 nie. Taktiež by bolo zaujímavé skombinovať metódy NMFD a PFNMF2 s použitím rôznych techník na úpravu matíc komponentov bicích nástrojov ako v [37] a zakomponovanie ďalších parametrov.

Taktiež by sme mohli skúsiť použiť opísanú metódu v [28] a predspracovanie dát, ktoré je uvedené v [37]. Zlepšenie aktuálnych metód transkripcie môžeme docieliť aj vyskúšaním rôznych vstupných reprezentácií, ako napr. MFCC alebo MCMS, ktoré preukazujú v spojení s neurónovými sieťami dobré výsledky [28].

Taktiež by bolo zaujímavé skombinovať neurónové siete využívajúce mechanizmy jemnej pozornosti s konvolučnými neurónovými sieťami. A to preto, lebo CNN sú flexibilnejšie, t.j. lepšie generalizujú model pričom mechanizmy jemnej pozornosti zlepšujú úspešnosť RNN [27].

Väčšina systémov sa pri použití akýchkoľvek prístupov zameriava na transkripciu základných úderov na bicie a techniku hrania berú len máloktoré do úvahy (posledný *Few-shot learning* prístup ich bral do úvahy [47]). Zakomponovanie hracej techniky do transkripcie bicích nástrojov z polyfonickej hudby s vhodným predspracovaním a prípadnou úpravou modelu, ktorý berie do úvahy techniku hrania, by mohol byť tiež jeden z našich prínosov, kde by sme mohli zistiť, či zvýšenie rozlišovacej schopnosti algoritmu pomôže zvýšiť úspešnosť transkripcie.

Jeden z ďalších možných smerov výskumu by sa mohol zaoberať zlepšením detekcie potenciálnych aktivácií bicích nástrojov – detekciu potenciálnych aktivácií rieši CNN v práci [45] v spojení NMFD, ktorý sa stará o samotnú transkripciu. Predpokladáme, že zlepšenie detekcie by malo za následok zlepšenie algoritmu samotnej transkripcie, ktorým by nemuselo byť len NMFD.

V práci [30] boli zlepšenia modelov za použitia dodatočných informácií, ktorými sú doby a prízvučné doby, dostatočné veľké na to, aby sa mohlo skonštatovať, že tieto informácie napomáhajú pri transkripcii bicích nástrojov. Hodnoty metriky *F-measure* sa však pohybujú v rozmedzí 0.66 – 0.68. Preto autori spravili experiment, v ktorom otestovali to, ako dobre vedia modely detegovať doby a prízvučné doby. Výsledky ukázali, že nie veľmi dobre. Preto jedna z ďalších možných ciest výskumu by mohlo byť zameranie sa na zlepšenie detekcie dôb a prízvučných dôb a ako veľmi to pomôže zvýšiť celkovú úspešnosť transkripcie [30].

Jednou z ďalších potenciálnych smerov zlepšenia transkripcie ADT by mohlo byť pri použití algoritmu PFNMF odhadnutie hodnoty parametra r_H na základe pravdepodobnostnej funkcie. Vďaka tomu by sa mohlo zlepšiť nielen modelovanie harmonickej zložky hudby ale aj celkovú transkripciu. Alebo by sme sa mohli pozrieť na rôzne regularizačné mechanizmy ako *sparsity, temporal continuity,* ktoré autori odporúčajú vyskúšať [42].

V tejto práci ako návrh na zlepšenie rozpracujeme viacero metód. Prvou bude skombinovanie NMFD a algoritmu PFNMF2 s rôznymi konfiguráciami. Taktiež skúsime matice algoritmu PFNMF2 upravovať pomocou metódy zostupujúceho gradientu a aplikujeme na aktivačnú maticu obmedzenie riedkosti. Druhou bude navrhnutie vlastného systému, ktorý bude prepájať hlboké neurónové siete a maticovú nezápornú faktorizáciu, pričom systém bude rátať aj s hracími technikami bicích. Ako posledný skúsime rozpracovať najmodernejší prístup v oblasti ADT, ktorým bolo použitie Prototypickej siete v kontexte *Few-shot learning*. Autori ho aplikovali na problém transkripcie hracích techník na bicích nástrojoch za prítomnosti harmonickej hudby. My sa pozrieme na to, ako tento model bude úspešný bez prítomnosti harmonických nástrojov v pozadí a skúsime aplikovať niekoľko vylepšení, ktoré by mohli zvýšiť jeho úspešnosť.

6. Návrh

Ako som uviedol v závere analýzy, v návrhu rozpracujeme 3 rôzne metódy, ktoré následne aj vyhodnotíme. Prvá metóda je zameraná na vylepšenie algoritmu NMFD vytvorením novej varianty, tzv. PFNMFD. Okrem toho sa pokúsime vylepšiť niekoľkými modifikáciami PFNMF variantu. Druhá metóda pozostáva z 2 komponentov – CNN a NMFD, pričom úlohou CNN je detegovať aktivácie a NMFD tieto miesta aktivácií lineárne aproximovuje. Dali sme jej pracovný názov NMFDQ. Tretia metóda je replikáciou najmodernejšieho *metric-learning* prístupu v oblasti ADT [47], ktorý sme spomínali v sekcii 5.1.7. Aplikujeme ho na transkripciu bicích nástrojov bez prítomnosti harmonickej zložky a pomocou niekoľkých vylepšení skúsime úspešnosť algoritmu zvýšiť.

Všetky 3 metódy majú spoločné to, že robia transkripciu polyfonickej hry bicích nástrojov. Prvý a druhý prístup využívajú rôzne NMF varianty, pretože chceme skúsiť vyťažiť z tohto prístupu čo najviac, keďže je robustný a jeho výsledky sú ľahko interpretovateľné. Druhý a tretí prístup zas primárne spája to, že robia transkripciu hracích techník bicích nástrojov na rozdiel od predošlých prístupov, ktoré poväčšine robili transkripciu len základných úderov na bicie, resp. nerobili rozdiely medzi rôznymi typmi úderov na ten istý nástroj. Vďaka tejto informácií chceme obidvom metódam pridať rozlišovaciu schopnosť, dôsledkom čoho budú jednak dodatočné informácie o hracích technikách počas transkripcie, čo predošlým metódam zvyčajne chýba a taktiež chceme sledovať, aký vplyv to bude mať na celkovú úspešnosť transkripcie. Tu chceme zdôrazniť, že akékoľvek skratky bicích nástrojov a ich techník, ktoré sa budú vyskytovať či vo vizualizáciách alebo v komentároch, sú bližšie vysvetlené na začiatku práce v slovníku pojmov a skratiek.

6.1. PFNMFD (kombinácia PFNMF2 a NMFD)

V tejto sekcii viac priblížim motiváciu, architektúru systému a všetky dôležite detaily tejto metódy.

6.1.1. Motivácia

Ako sme uviedli na začiatku sekcie 6, táto metóda je zameraná na vylepšenie algoritmu NMFD. A to takým spôsobom, že sme skombinovali metódy PFNMF2 a NMFD, výsledkom čoho je PFNMFD (náš vlastný pracovný názov). Dôvod kombinácie práve týchto 2 NMF variant teraz objasníme.

Keďže chceme aplikovať NMF metódu na transkripciu bicích nástrojov za prítomnosti harmonických nástrojov, potrebujeme využiť variant PFNMF, kde časť matice komponentov

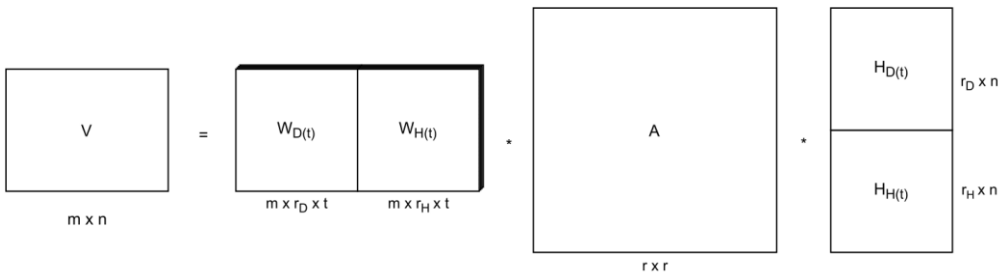
určená pre bicie sa vopred inicializuje a počas transkripcie bude zafixovaná. Vďaka tomu sa potom môže algoritmus sústrediť jednak na detekciu aktivácií bicích nástrojov ale aj na čo najlepšiu aproximáciu harmonickej zložky v hudbe, ktorú pri spätnej rekonštrukcii odstránime. Variant NMFD pouužijeme preto, lebo chceme ponechať modelovaciu kapacitu úderov bicích na maticu komponentov bicích nástrojov, čo sa v predošlých prístupoch, ktoré využívali NMFD, osvedčilo. Taktiež doterajšie prístupy poväčšine používali NMFD len pri transkripcií monofonickej alebo polyfonickej hry bicích bez prítomnosti harmonických nástrojov, kde algoritmus preukazoval pomerne dobrú úspešnosť [3, 52].

Našim hlavným prínosom je kombinácia algoritmov PFNMF2 a NMFD, ktorá sa v rámci výskumnej oblasti ADT explicitne nikde nevyskytuje. Vyskúšame ho s rôznymi konfiguráciami, ktoré následne vyhodnotíme. Taktiež skúsime rôzne nastavenia hyperparametrov a spôsob optimalizácie pri algoritme PFNMF2, ktorý nikto v článkoch doposiaľ neuvádzal. Pod vyjadrením *rôzne spôsoby optimalizácie, ktoré nikto neuvádzal* máme na mysli metódu zostupujúceho gradientu, na ktorú sme počas analýzy najmodernejších metód nikde nenarazili. Pod vyjadreniami *rôzne konfigurácie,* alebo *rôzne nastavenia hyperparametrov* máme na mysli:

- pridanie obmedzenia riedkosti (angl. *sparsity constraint*)
- dodatočné úpravy matice komponentov bicích nástrojov (podľa [37])
- nastavenie parametra r_h, ktorý definuje počet komponentov v matici harmonických nástrojov

6.1.2. Návrh PFNMFD metódy

Náš vlastný návrh je vytvorený kombináciou variant PFNMF2 a NMFD. Metóda funguje nasledovným spôsobom:



Obrázok 29. Architektúra algoritmu PFNMFD

Dimenzia *t* pri maticiach W_D a W_H definuje to, ako vyzerá spektrum komponentov v čase t. Môžeme si všimnúť, že architektúra je podobná algoritmu PFNMF2 s tým rozdielom, že naša matica komponentov je trojrozmerná. Funkcia aproximácie je rovnaká ako pri NMFD (vzorec 47). Pri použití MU a *Euklidovskej vzdialenosti* sa matice upravujú nasledovným spôsobom

$$H_D \leftarrow H_D \circ \frac{{W_D}_t^{\mathrm{T}}(X)^{\leftarrow t}}{{W_D}_t^{\mathrm{T}}(\alpha W_D H_D + \beta W_H H_H)^{\leftarrow t}} \quad \text{a } {W_D}_t \leftarrow {W_D}_t \circ \frac{X({H_D}^{t\rightarrow})^{\mathrm{T}}}{(\alpha W_D H_D + \beta W_H H_H)({H_D}^{t\rightarrow})^{\mathrm{T}}} \tag{61}$$

$$H_H \leftarrow H_H \circ \frac{{W_H}_t^{\mathrm{T}}[X]^{\leftarrow t}}{{W_H}_t^{\mathrm{T}}(\alpha W_D H_D + \beta W_H H_H)^{\leftarrow t}} \quad \text{a } {W_H}_t \leftarrow {W_H}_t \circ \frac{X({H_H}^{t\rightarrow})^{\mathrm{T}}}{(\alpha W_D H_D + \beta W_H H_H)({H_H}^{t\rightarrow})^{\mathrm{T}}}, \tag{62}$$

$$\forall t \in [0 \ldots T - 1]$$

a pri použití *KL-divergencie* takto

$$H_D \leftarrow H_D \circ \frac{{W_D}_t^{\mathrm{T}}\left[\frac{X}{\alpha W_D H_D + \beta W_H H_H}\right]^{\leftarrow t}}{{W_D}_t^{\mathrm{T}}1} \quad \text{a} \quad {W_D}_t \leftarrow {W_D}_t \circ \frac{\frac{X}{\alpha W_D H_D + \beta W_H H_H}({H_D}^{t\rightarrow})^{\mathrm{T}}}{1({H_D}^{t\rightarrow})^{\mathrm{T}}} \tag{63}$$

$$H_H \leftarrow H_H \circ \frac{{W_H}_t^{\mathrm{T}}\left[\frac{X}{\alpha W_D H_D + \beta W_H H_H}\right]^{\leftarrow t}}{{W_H}_t^{\mathrm{T}}1} \quad \text{a} \quad {W_H}_t \leftarrow {W_H}_t \circ \frac{\frac{X}{\alpha W_D H_D + \beta W_H H_H}({H_H}^{t\rightarrow})^{\mathrm{T}}}{1({H_H}^{t\rightarrow})^{\mathrm{T}}}, \tag{64}$$

$$\forall t \in [0 \ldots T - 1]$$

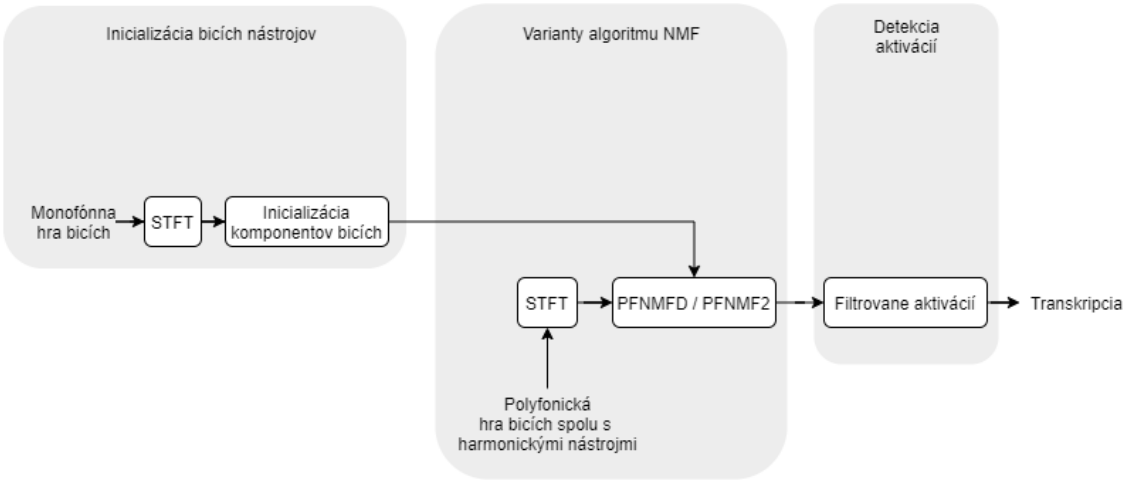
Pozitíva a negatíva tohto prístupu sú tie isté ako pri PFNMF s tým vylepšením, že tento typ NMF modeluje spektrálno-časový priebeh nástrojov v matici komponentov. To sa môže odzrkadliť pri menšej veľkosti chyby a zlepšením rekonštrukcie spektrogramu.

6.1.3. Architektúra ADT systému

Architektúra nášho ADT systému, ktorý sme implementovali a na ktorom budeme vyhodnocovať naše modely, je zobrazená na nasledujúcom obrázku (viď Obrázok 30). V nasledovných sekciách si rozoberieme niektoré časti architektúry.

Inicializácia bicích nástrojov

Na inicializáciu bicích nástrojov je potrebné mať k dispozícií dataset s nahrávkami, v ktorých údery znejú monofonicky. Ideálne by bolo, ak by jedna nahrávka disponovala viacerými, za sebou nasledujúcimi monofonickými údermi s dlhšími pauzami.



Obrázok 30. Architektúra ADT systému

Varianty algoritmu NMF (PFNMFD, PFNMF2)

V tejto časti opíšeme jednotlivé prístupy, ktoré zimplementujeme a vyhodnotíme.

- PFNMF2 s použitím metódy PG - Algoritmus optimalizuje matice W_H, H_D a H_H pomocou spomenutej metódy. Optimalizácia sa robí na základe zvolenej chybovej funkcie.
- PFNMF2 s použitím obmedzenia riedkosti – algoritmus optimalizuje matice W_H, H_D a H_H buď podľa metódy PG alebo MU. Dôležitým prvkom je aplikovanie obmedzenia riedkosti na aktivačnú maticu. Nepoužijeme ju na celú ale iba na časť pre bicie nástroje. Chceme aby udalosti bicích boli riedke, pretože ako sme uviedli v časti 1.4, bicie sú rytmickou zložkou hudby a ich úlohou je udávať presný rytmus a tempo. Preto musia znieť krátko, ale za to presne. Aby sme si ukázali, ako ovplyvní obmedzenie riedkosti spôsob úpravu matice aktivácií pre bicie nástroje, tak napríklad v prípade použitia metódy MU a Euklidovskej vzdialenosti to bude vyzerať nasledovne

$$H_D \leftarrow H_D \circ \frac{W_D^T X}{W_D^T(\alpha W_D H_D + \beta W_H H_H) + \lambda} \tag{65}$$

  Vidíme, že vzorec sa veľmi nezmenil oproti vzorcu 61. Na koniec menovateľa sa pridal len znak λ. Ide o parameter optimalizácie, ktorý je zodpovedný za mieru obmedzenia riedkosti. Čím väčšie číslo, tým je obmedzenie silnejšie.
- PFNMF2 s dodatočným aktualizovaním matice W_D – tak ako je uvedené v [37], najprv je matica W_D fixná a ostatné sa upravujú. Po dokončení úprav sa spustí druhý proces faktorizácie, pri ktorom je fixná H_D a W_D, H_D, H_H sa budú upravovať. V princípe ide o to, že najprv sa nájdu aktivácie bicích a potom sa lepšie domodelujú ich spektrálne komponenty.

- PFNMFD s použitím metódy PG – Tak ako pri jednom z predchádzajúcich PFNMF2 algoritmov aj tento optimalizuje matice W_H, H_D a H_H pomocou spomenutej metódy len s tým rozdielom, že tu je každý nástroj reprezentovaný maticou a nie vektorom. Matice sa optimalizujú v závislosti od zvolenej chybovej funkcie.
- PFNMFD s použitím obmedzenia riedkosti – platí to isté čo pri PFNMF2 s obmedzením riedkosti len s tým rozdielom, že nástroje sú v matici komponentov reprezentované maticami. Ak by sme napríklad upravovali matice pomocou MU s použitím KL-divergencie, úprava aktivačnej matice pre bicie nástroje bude vyzerať nasledovne

$$H_H \leftarrow H_H \circ \frac{W_{H_t}^{\mathrm{T}}\left[\frac{X}{\alpha W_D H_D + \beta W_H H_H}\right]^{\leftarrow t}}{W_{H_t}^{\mathrm{T}} 1 + \lambda} \tag{66}$$

  Môžeme si všimnúť, že oproti úprave vo vzorci 64 sa tento vzorec zmenil len v tom, že na konci menovateľa pribudol znak λ, ktorý ovplyvňuje mieru riedkosti.

- PFNMFD s dodatočným aktualizovaním matice W_D – platí to isté čo pri PFNMF2 s dodatočným aktualizovaním matice W_D len s tým rozdielom, že úpravy matíc budú odlišné vzhľadom na to, že každý hudobný nástroj je modelovaný maticou a nie vektorom.

Vstupom do týchto variant algoritmu je STFT polyfonickej hudby, ktorý v závislosti od testovacieho scenára bude obsahovať hru na bicích nástrojoch bez harmonických nástrojov alebo aj s nimi.

6.2. NMFDQ (CNN + NMFD)

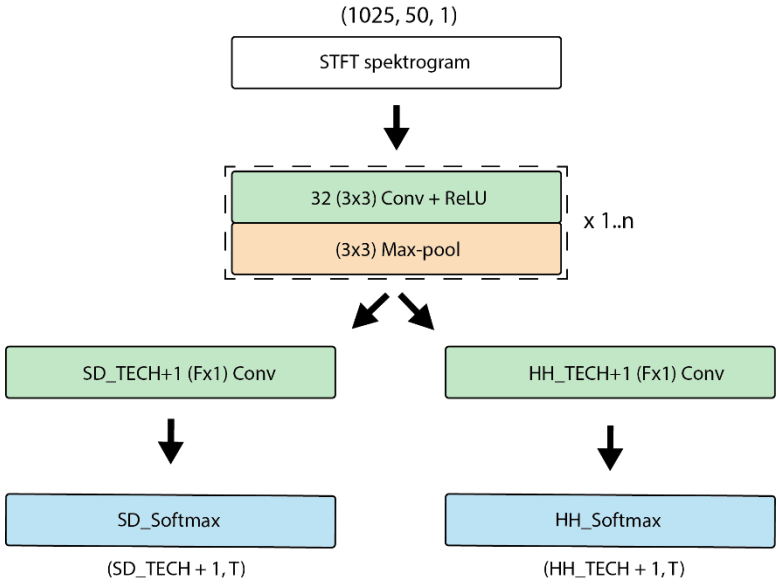
V tejto sekcií si viac priblížime nami navrhnutú metódu NMFDQ.

6.2.1. Motivácia

Na začiatku sekcie 6 sme uviedli, že táto metóda bude pri transkripcii bicích nástrojov uvádzať aj informáciu o hracích technikách. Tento typ informácie chceme zakomponovať do návrhu našej metódy primárne z 2 dôvodov. Prvým je jej reálne využitie v praxi, pretože sa vo výučbových materiáloch pre bubeníkov zvyčajne nejakým spôsobom nachádza. Druhým je snaha pridať metóde rozlišovaciu schopnosť, dôsledkom čoho budú už spomínané dodatočné informácie o hracích technikách počas transkripcie, čo predošlým metódam chýba, ale chceme pozorovať aj to, či takto nami navrhnutá metóda má potenciál konkurovať súčasným najmodernejším prístupom v ADT. Vzhľadom na komplexnosť metódy sme sa ju rozhodli aplikovať na jednoduchší problém - polyfonickú hru bicích nástrojov bez prítomnosti harmonickej zložky v pozadí.

6.2.2. Návrh NMFDQ metódy

Nami navrhnutý algoritmus pozostáva primárne z 2 hlavných komponentov. Prvým je konvolučná neurónová sieť, ktorá deteguje aktivitu hracích techník na bicích nástrojoch. Jej výstupom je aktivačná matica Q, ktorá svojimi informáciami obohacuje druhý komponent – metódu NMFDQ. Najprv rozoberiem prvý komponent, ktorý ozrejmime samostatne a neskôr aj v kontexte celej NMFDQ metódy.



Obrázok 31. Architektúra konvolučnej neurónovej siete

Na obrázku 31 môžeme vidieť architektúru CNN. Jej úlohou je rozoznávať aktiváciu hracích techník rôznych bicích nástrojov v čase. Z architektúry môžeme usúdiť, že naša CNN vie riešiť tzv. *multi-output multi-class* klasifikačný problém.

Slovné spojenie *multi-output* v kontexte strojového učenia pomocou hlbokých neurónových sietí znamená, že neurónová sieť má viacero výstupných vrstiev, pričom každá rieši separátny klasifikačný problém. V našom prípade má takéto niečo zmysel práve preto, lebo v čase môže znieť viacero bicích nástrojov naraz. To znamená, že každá výstupná vrstva rieši samostatný klasifikačný problém pre jeden nástroj [54].

Z obrázka môžeme ďalej vyčítať, že naša neurónová sieť vie rozoznávať aktivitu hracích techník len pre 2 nástroje - rytmičák a hi-hat činel. Je to tak preto, lebo našu metódu chceme otestovať najprv len na menšom počte nástrojov – konkrétne rytmičák, kopák, hi-hat činel – z dôvodu obmedzení dostupných datasetov vhodných na takéto testovanie. Detekcia hracích techník pre kopák tam chýba. Nie je to chyba. Je to zámerné, pretože hracie techniky pre kopák nie sú jednoznačne definované a existuje len jedna jediná – obyčajný úder.

Slovné spojenie *multi-class* v kontexte strojového učenia zas znamená, že algoritmus priradí vstupným dátam vždy len jednu triedu. Spojením týchto dvoch typov klasifikácií vznikne *multi-output multi-class* klasifikácia, čo znamená, že každá výstupná vrstva neurónovej siete bude môcť predikovať len jednu triedu v čase [54]. V našom kontexte takéto niečo má význam, lebo naraz v každom časovom okamihu môže hrať bubeník len jednou hracou technikou.

Keď si detailnejšie prejdeme architektúru CNN (viď Obrázok 31) zhora nadol, môžeme si všimnúť, že CNN očakáva na vstupe STFT spektrogram s 1025 frekvenčnými rozsahmi a dĺžkou 50 rámcov. Potom nasleduje sada konvolučných blokov. Jeden pozostáva z 32 konvolučných jadier o rozmeroch 3x3, ktoré robia 2D konvolúciu, aktivačnej funkcie *ReLU* a *max pooling* operácie o rozmeroch 3x3. Presný počet konvolučných blokov nedefinujeme, pretože ten je dynamicky závislý od veľkosti konvolučného kontextu, ktorý definuje veľkosť výsekov zo vstupu, ktoré bude CNN klasifikovať. Pri experimentoch bude platiť pravidlo, že čím väčší konvolučný kontext zvolíme, tým bude sieť hlbšia, pretože bude mať väčší počet konvolučných blokov.

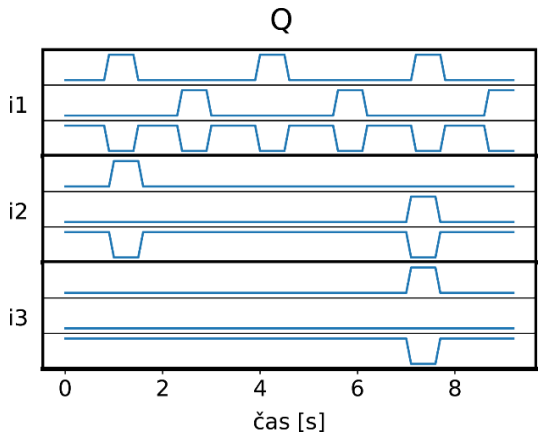
Po sade konvolučných blokov dochádza k vetveniu. Každá vetva má na starosti klasifikáciu hracích techník pre jeden nástroj. Obidve vetvy najprv disponujú vrstvou, ktorá robí 2D konvolúciu s počtom konvolučných jadier, ktorý je rovný počtu hracích techník pre daný nástroj + 1 (názvy SD_TECH a HH_TECH reprezentujú počty hracích techník pre rytmičák a hi-hat činel). To, že tam je o jedno jadro naviac si objasníme neskôr pri opise aktivačnej funkcie *softmax*. No inými slovami povedané, každé konvolučné jadro v tejto vrstve sa pri trénovaní bude učiť čo najlepšie rozoznať špecifickú hraciu techniku. Tieto jadrá budú robiť konvolúciu naprieč celou frekvenčnou dimenziou, ktorej veľkosť však nepoznáme, keďže je tiež dynamicky závislá od veľkosti konvolučného kontextu (písmeno F pri rozmeroch konvolučných blokov). Treba ešte zvýrazniť druhý rozmer týchto konvolučných jadier, ktorý je staticky nastavený na hodnotu 1. Je to tak preto, aby CNN mohla mať na vstupe rôzne veľkosti konvolučných kontextov a počet predikcii na výstupe sa vedel automaticky zarovnať so vstupom (napr. ak bude konvolučný kontext veľký 7 rámcov, zo vstupu o veľkosti 50 rámcov CNN spraví 44 predikcii, ak bude konvolučný kontext veľký 5 rámcov, CNN spraví 46 predikcií). V oboch vetvách sa v poslednej vrstve aplikuje funkcia *softmax*, ktorá robí samotnú klasifikáciu. Jej matematické vyjadrenie je nasledovné

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{67}$$

V princípe ide o to, že softmax priradí rôzne pravdepodobnostné hodnoty všetkým triedam (v našom kontexte hracím technikám pre daný nástroj), nad ktorými sa robí klasifikácia. Trieda s najväčšou pravdepodobnosťou je zvyčajne označovaná za tú, ktorú algoritmus predikoval pre určený vstup. Platí, že ak sčítame všetky pravdepodobnosti, výsledkom bude hodnota 1.

Tu treba zdôrazniť, že v oboch vetvách softmax robí klasifikáciu nad počtom hracích techník pre daný nástroj + 1 (Obrázok 31 - v poslednej vrstve v oboch vetvách si môžeme všimnúť HH_TECH + 1, SD_TECH + 1). Táto trieda navyše slúži na to, aby ju CNN klasifikovala v prípade, že nezdetegovala aktivitu daného bicieho nástroja. V podstate prítomnosť tejto triedy je v našom kontexte nevyhnutná, keďže akýkoľvek bicí nástroj je v skladbe zvyčajne viac neaktívny ako aktívny a preto musíme CNN učiť rozoznávať aj túto charakteristiku. Navyše aktivita tejto nazvime ju "neutrálnej" triedy pre každý nástroj pridáva celej metóde NMFDQ vlastnosť *fault-tolerance*, ktorú však rozoberieme neskôr.

Definitívny výstup z oboch vetiev, ktorý sa nakoniec zapíše do aktivačnej matice Q, je vždy binárny, čo znamená, že len tej hracej technike, ktorá bude aktívna (bude mať najväčšiu pravdepodobnosť) pre daný nástroj sa priradí hodnota 1 a ostatným technikám nástroja priradí 0. Aby sme zdôraznili ešte zopár dôležitých detailov výstupnej matice Q, uvedieme si jej vizualizáciu na nami zvolenom príklade

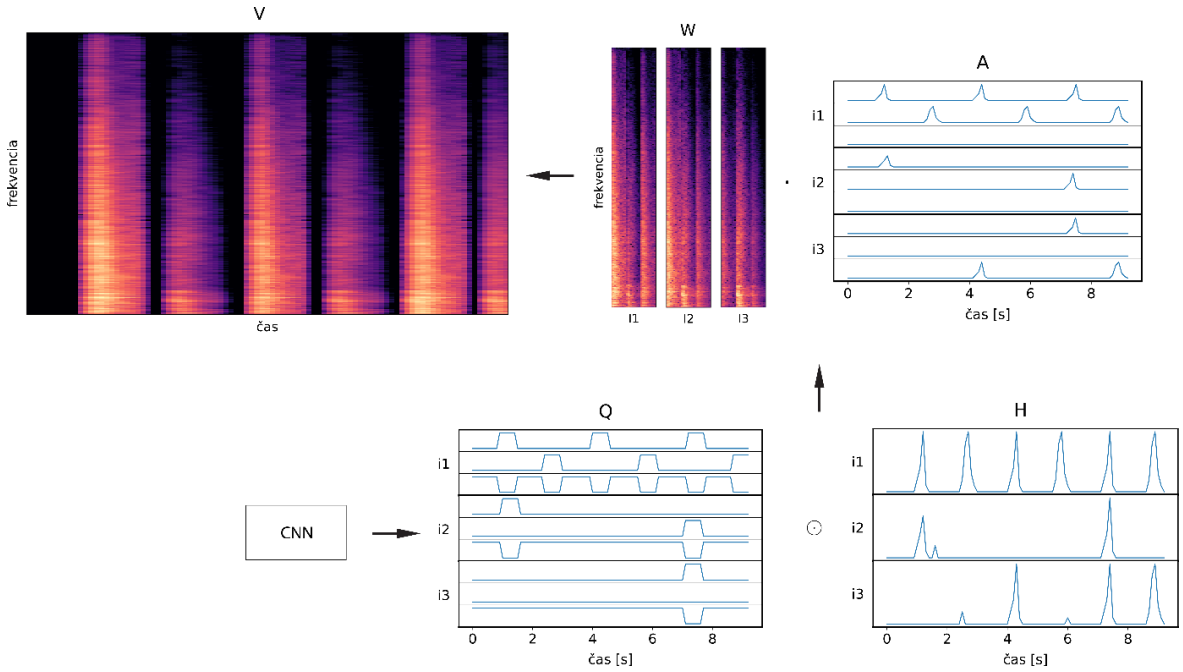


Obrázok 32. Vizualizácia Q matice, ktorá je výstupom CNN

Na obrázku 32 môžeme vidieť Q maticu, ktorej x-ová os reprezentuje čas a y-ová jednotlivé hracie techniky hudobných nástrojov. Tieto hracie techniky sú priradené konkrétnym nástrojom, ktoré majú generické názvy *i1, i2, i3* (sú to skratky anglických slov *instrument1, instrument2, instrument3*). Môžeme si všimnúť, že jednotlivé hracie techniky sú v čase rôzne aktívne ale nikdy sa nestane, že 2 techniky toho istého nástroja (či už je to reálna hracia

technika alebo trieda reprezentujúca neaktivitu hudobného nástroja) sú v tom istom čase naraz aktívne. Taktiež si môžeme všimnúť, že vždy posledná hracia technika reprezentuje neaktivitu hudobného nástroja, pretože je aktívna len vtedy, keď nie je aktívna žiadna z ostatných hracích techník. Na záver by som ešte zdôraznil to, že keď budeme trénovať CNN, budeme ju učiť rozoznávať aktivitu hracích techník nie len v jednom konkrétnom čase (resp. v jednom STFT rámci) ale aj v jeho blízkom okolí. Toto si môžeme všimnúť na vyššie uvedenom obrázku kde aktivita hracích techník nie je vizualizovaná špicatými skokmi, ale hranatými. Aký to bude mať význam pre celkovú NMFDQ metódu, to si objasníme neskôr.

Teraz by som rád priblížil, ako je do našej metódy zakomponovaná NMFDQ a ako spolupracuje s CNN. Na obrázku 33 môžeme vidieť architektúru našej metódy. Veľké písmena nad zobrazenými maticami označujú ich mená. Označenia *i1*, *i2*, *i3* sú generické názvy pre ľubovoľné bicie nástroje tak ako na obrázku 32. Vo vizualizácií určujú, ktorá časť v matici komponentov (matica W) alebo v maticiach aktivácií (matice H, Q alebo A) prislúcha jednému nástroju. Časová dimenzia je pre matice Q, H, A a V rovnaká. Taktiež frekvenčná dimenzia je pre matice V a W rovnaká. Môžeme si všimnúť, že v matici W každý nástroj disponuje viacerými krátkymi spektrogramami, ktoré reprezentujú priebeh hracej techniky v čase. Tie sú vždy vopred inicializované. Ich počet je rovný počtu hracích techník pre daný nástroj + 1. Význam spektrogramu navyše si vysvetlíme neskôr. Treba však podotknúť, že reprezentuje neutrálnu triedu, ktorú si môžeme definovať rôznymi spôsobmi – napríklad mô-



Obrázok 33. Architektúra metódy NMFDQ

že byť vypočítaný ako priemer všetkých hracích techník pre daný nástroj, alebo sa vyberie najčastejšie hraná technika nástroja, alebo iné.

Komponent NMFDQ, ktorý je súčasťou našej NMFDQ metódy a ktorý teraz bližšie ozrejmime, funguje veľmi podobne ako klasická NMFD.

$$V \approx \sum_{t=0}^{T-1} W_t \,.\, A^{t\rightarrow} \tag{68}$$

Funkcia aproximácie je v zásade identická vzorcu 47 len s tým rozdielom, že matica komponentov sa lineárne kombinuje s maticou A, ktorá je odlišná od obyčajnej aktivačnej matice H.

$$\mathrm{A} = \begin{bmatrix} h_{i1} \circ Q_{i1} \\ h_{i2} \circ Q_{i2} \\ \cdots \\ h_{in} \circ Q_{in} \end{bmatrix} \tag{69}$$

Totižto matica A vzniká *Hadamardovým súčinom* medzi vektormi aktivácií v matici H a maticami aktivácií v matici Q, ktoré patria tomu istému nástroju (viď vzorec 69). O maticiach aktivácií a nie vektoroch v matici Q hovoríme preto, lebo ako sme už vyššie spomínali, jeden nástroj môže disponovať viacerými hracími technikami a konvolučná sieť sa ich snaží na vhodných miestach detegovať. Výsledkom tohto súčinu je potom matica A, ktorá disponuje aktivitou hracích techník jednotlivých bicích nástrojov.

Keďže NMFDQ komponent našej metódy je príbuzný klasickej NMFD, tiež je iteratívny a upravuje sa identickým spôsobom (vzorce 49 alebo 50). Rozdiel oproti klasickej NMFD je však v tom, že neupravuje sa matica, ktorá sa lineárne kombinuje s maticou komponentov (v našom prípade A), ale matica H. Matica Q s potenciálnymi aktivitami nástrojov, ktorú vygeneruje konvolučná sieť, je počas iteratívneho procesu zafixovaná a matica H sa tak snaží hľadať vhodné miesta aktivít jednotlivých nástrojov, ktoré sa po Hadamardovom súčine s maticou Q premietnu do aktivácií konkrétnych hracích techník.

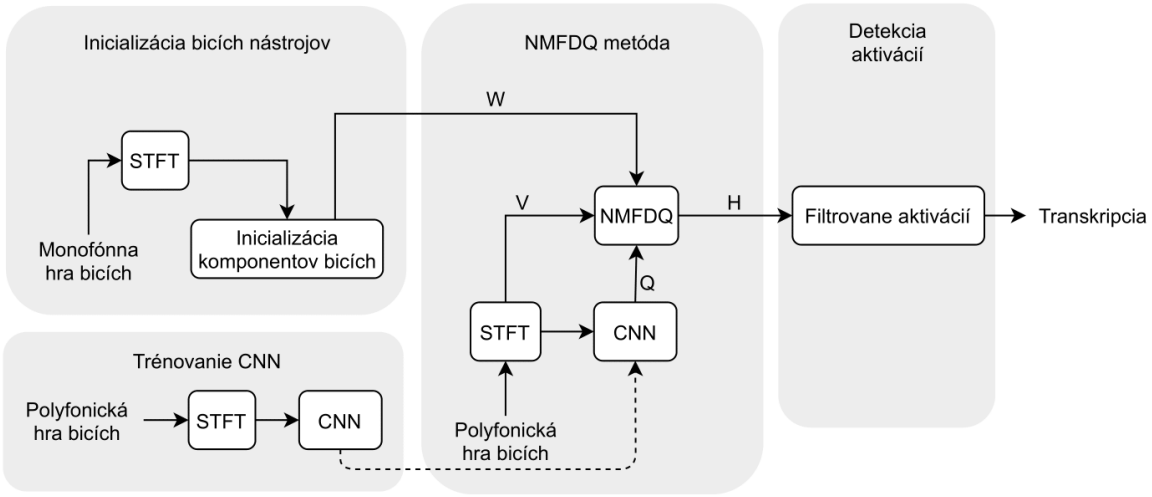
Teraz by som rád ozrejmil 2 vlastnosti, ktorými by naša metóda mohla disponovať vďaka neutrálnym triedam pre každý nástroj v Q matici a vďaka oblastiam aktivít hracích techník, ktoré deteguje CNN. Vďaka neutrálnym triedam v maticiach W a Q by mohla byť čiastočne *fault-tolerant* voči chybne detegovanej neaktivite nástroja v matici Q. A to preto, lebo v prípade, že CNN nedeteguje aktivitu nástroja na mieste, na ktorom by mala, v Q matici sa bude na takomto mieste vyskytovať aktivita neutrálnej triedy a NMFD môže "vysvetliť" tento úder aspoň aktivitou neutrálnej triedy nástroja (napr. priemerom hracích techník daného

nástroja), čo by mohlo zlepšiť celkovú úspešnosť transkripcie na úrovni nástrojov. Toto experimentálne dokážeme.

Oblasti aktivít hracích techník, ktoré sú detegované CNN majú zas taký význam, že pri iteratívnych aktualizáciách hodnôt v jednotlivých maticiach počas procesu faktorizácie si NMFD môže vybrať konkrétne miesto z vybranej oblasti, v ktorom nastane aktivita danej hracej techniky tak, aby celková chybová funkcia bola čo najmenšia.

6.2.3. Architektúra ADT systému

Architektúra systému, ktorého súčasťou je naša metóda NMFDQ, vyzerá nasledovne.



Obrázok 34. Architektúra ADT systému

Krabičky s bielym pozadím a textom reprezentujú proces. Šípky s plnou čiarou reprezentujú to, že nesú so sebou nejaký artefakt (informáciu v podobe aktivačnej matice, matice predspracovaného signálu, atď.). Prerušované šípky reprezentujú to, že koncová entita je kópiou zdrojovej (napr. CNN sa natrénuje a jej inštancia sa použije v metóde NMFDQ). V nasledujúcej sekcií si bližšie priblížime inicializáciu hracích techník, kde by sme zdôraznili zopár dôležitých detailov, ktoré musíme zohľadniť pri samotných experimentoch.

Inicializácia bicích nástrojov

Na to, aby sme mohli robiť inicializáciu hracích techník jednotlivých nástrojov, potrebujeme si najprv nájsť dataset, ktorý týmito hracími technikami disponuje a ktorý použijeme aj pri vyhodnotení. Totižto musíme vedieť, ktoré techniky máme vopred inicializovať v matici W a ktoré sa budú pri transkripcii bicích nástrojov "hľadať" v skladbách (v matici V). Ideálne by bolo, aby dataset obsahoval samostatné nahrávky monofonických úderov hracích techník,

z ktorých ľahko tieto techniky inicializujeme . Na experimentálne účely bude úplne stačiť, ak v nahrávkach monofonických úderov budú len 2-3 údery každou hracou technikou.

6.3. Prototypická CNN

V tejto časti si viac priblížime metódu, ktorá na rozdiel od predošlých 2 riešení používa len hlbokú neurónovú sieť. Metódu reimplementujeme z [47] a modifikujeme o nami navrhnuté vylepšenia.

6.3.1. Motivácia

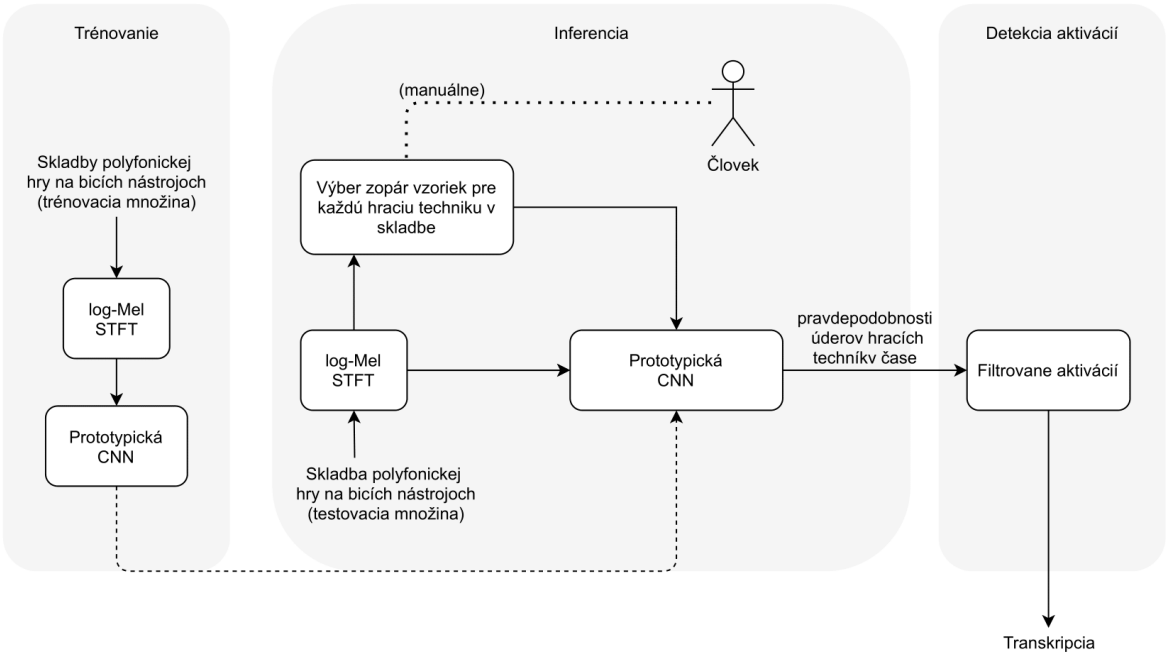
Ako sme sa už na začiatku sekcie 6 zmienili, aj táto metóda robí transkripciu hracích techník bicích nástrojov. V tomto prípade sme sa však rozhodli riešiť špecifickejší problém. Tým je schopnosť robiť transkripciu akustických bicích nástrojov (samozrejme aj s hracími technikami) vo veľmi obmedzených podmienkach, v ktorých má bubeník k dispozícii len jeden mikrofón, ktorý sníma celú biciu sadu. Predtým ako bubeník začne hrať na biciu sadu, chceme mu umožniť "nakalibrovať" metódu na zvuk jeho akustickej bicej sady (podobný problém bol v minulosti motiváciou aj pre inú skupinu autorov, ktorí sa ho rozhodli riešiť pomocou metódy NMFD [52]).

To, čo bolo pre nás rozhodujúcim faktorom na to, aby práve táto metóda riešila spomínaný problém, vyplýva hlavne z 2 dôvodov. Prvým bolo nedávne (koniec roka 2020) zverejnenie článku, v ktorom bol tento pomerne nový typ hlbokého učenia aplikovaný na podobný problém – automatická transkripcia hracích techník bicích nástrojov za prítomnosti harmonickej hudby v pozadí [47]. Keďže úspešnosť najlepšej konfigurácie modelu dosahovala pomerne nízku hodnotu metriky *F-measure* na validačných množinách (medzi 0.5 – 0.6), chceli sme aplikovať tento istý prístup na náš problém (transkripcia hracích techník bez prítomnosti harmonickej zložky) a vylepšiť ho, čím by sme mohli dosiahnuť vyššiu úspešnosť. Druhým dôvodom bola vlastnosť kalibrácie, ktorou táto metóda disponuje už len tým, ako je navrhnutá. Totižto v sekcii 5.1.7 sme spomínali, že hoci je metóda schopná predikovať neznáme triedy dát (také triedy, ktoré sa vyskytujú v testovacej množine ale nevyskytujú sa v trénovacej množine), na to potrebuje zopár vzoriek vopred, aby si vedela vytvoriť prototyp danej triedy a aby neskôr vedela do tejto triedy priraďovať ďalšie vzorky. Práve spomínané *ukázanie zopár vzoriek neznámej triedy vopred* robí metódu kalibrovateľnou . Dôležité však je, aby sa metóda predtým naučila pestrý diskriminatívny priestor, v ktorom budú latentné reprezentácie nových, neznámych tried jasne oddelené od ostatných.

V nasledujúcich sekciách si najprv priblížime to, ako vyzerá architektúra ADT systému s prototypickou CNN, akým spôsobom sa učí a ako prebieha inferenčná fáza, v ktorej sa robí samotná transkripcia. Potom navrhneme zopár zmien a vylepšení, ktoré sú vhodné vzhľadom na špecifickosť nášho problému a zároveň bypo mohli zlepšiť úspešnosť metódy.

6.3.2. Architektúra ADT systému s prototypickou CNN

Celková architektúra ADT systému vyzerá nasledovne

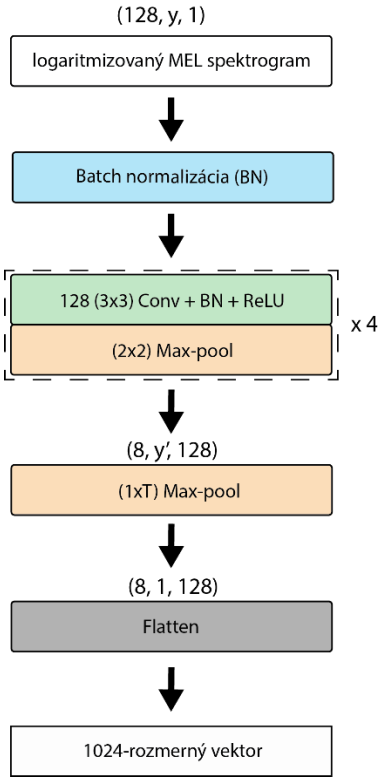


Obrázok 35. Vizualizácia architektúry ADT systému s prototypickou CNN

Na obrázku 35 môžeme vidieť vizualizáciu celého ADT systému. Význam jednotlivých komponentov a šípok bol už spísaný pri vizualizácií ADT architektúry v sekcii 6.2.3. Na vyššie uvedenom obrázku je novým komponentom prerušovaná čiara (nie šipka), ktorá znázorňuje to, že prepojené komponenty spolu úzko súvisia. Konkrétne v našom kontexte reprezentuje to, že človek manuálne vyberá vzorky pre každú hraciu techniku zo skladby a tie sú spolu s predspracovanou skladbou vstupom do prototypickej CNN. Celý proces priblížime viac v nasledujúcich sekciách.

Prototypická CNN

Keďže CNN je stavebným kameňom celej metódy, najprv si uvedieme jej architektúru, ktorá je identická s tou, ktorá bola použitá v už viackrát spomínanom článku [47].



Obrázok 36. Architektúra CNN [47]

Na obrázku 36 môžeme vidieť architektúru CNN. Na vstupe očakáva logaritmizovaný Mel spektrogram so 128 Mel bankami. Počet rámcov nie je konštantne definovaný (parameter označený *y*), pretože časová dimenzia vstupu bude experimentálnou záležitosťou a chceme ju vedieť dynamicky meniť. Jadrom architektúry je sada 4 konvolučných blokov. Jeden bude pozostávať zo 128 konvolučných jadier o rozmeroch 3x3, ktoré robia 2D konvolúciu, *batch* normalizácie, aktivačnej funkcie *ReLU* a *max pooling* operácie o rozmeroch 2x2. Po sade konvolučných blokov nasleduje jedna vrstva, ktorá prijíma na vstupe redukovanú reprezentáciu vstupu (frekvenčná doména sa zredukovala zo 128 na 8 a časová doména z *y* na *y'*) a robí *max pooling* operáciu naprieč časovou dimenziou (tzv. *Global Max Pooling*). Tento komponent slúži na to, aby model vedel ľubovoľne dlhý vstup zakódovať do výsledného 1024-rozmerného vektora (latentná reprezentácia vstupu) s ktorým sa ďalej pracuje.

Trénovanie

Ako sme už spomenuli v sekcii 5.1.7, táto CNN je preto prototypická, lebo jej úlohou je naučiť sa nelineárnu funkciu dostatočne členitého diskriminatívneho priestoru, v ktorom sú latentné reprezentácie jednotlivých prototypov tried, v našom prípade prototypov hracích techník bicích nástrojov, v dostatočnej vzdialenosti od seba, a zároveň prototypy spolu s ich prislúchajúcimi vzorkami sa vedia organizovať v spoločných zhlukoch.

Na to, aby sme takéto niečo docielili, potrebujeme sieť učiť riešiť *C-way K-shot* klasifikačný problém pomocou epizodického trénovania (angl. *episodic training*). V prenesenom význame to znamená, že v jednej epizóde, resp. jednej iterácií počas trénovania sa udejú nasledovné kroky. Najprv sa náhodne zvolí *C* hracích techník (*C* = celé číslo väčšinou od 1 do 10), ktoré sa vyskytujú v trénovacej množine. Potom sa pre každú hraciu techniku náhodne vyberie *K* vzoriek (K = celé číslo väčšinou od 1 do 5) z trénovacej množiny. Každá vzorka je reprezentovaná malým Mel spektrogramom, ktorý obsahuje úder na bicí nástroj vybranou hracou technikou. Tento úder je vždy časovo zarovnaný na stred spektrogramu. Tieto vzorky pomocou CNN zakódujeme a vytvoríme množinu latentných vektorov *S* o veľkosti $C \times K$. Následne vytvoríme množinu $M = \{\mu_1, \dots, \mu_c\}$, ktorá bude obsahovať prototypy týchto hracích techník. Prototyp pre jednu techniku bude vypočítaný ako priemer jej latentných reprezentácií z množiny *S*:

$$\mu_c = \frac{1}{K} \sum_{(x,y) \in S_c} f_\theta(x) \tag{70}$$

kde S_c označuje skupinu latentných reprezentácií pre jednu techniku a f_θ reprezentuje nelineárnu funkciu, pomocou ktorej CNN kóduje vstupné vzorky techník do latentného priestoru. Potom vytvoríme disjunktnú množinu Q o veľkosti $C \times q$, ktorá bude obsahovať q vzoriek pre každú hraciu techniku. Každú vzorku x_q z množiny Q pomocou CNN opäť premietneme do latentného priestoru a vďaka funkcií *softmax* aplikovanej na vzdialenosti k jednotlivým prototypom dostaneme pravdepodobnostnú distribúciu vybraných hracích techník:

$$p_\theta(y = c \mid x_q) = \frac{\exp\left(-d(f_\theta(x_q), \mu_c)\right)}{\sum_{c'} \exp\left(-d(f_\theta(x_q), \mu_{c'})\right)} \tag{71}$$

kde *d* je funkcia, ktorá ráta Euklidovskú vzdialenosť. Vo vzorci 71 si môžeme všimnúť, že funkcia *softmax* ráta pravdepodobnostnú distribúciu z **negatívnych** Euklidovských vzdialeností. Je to tak preto, aby funkcia priraďovala väčšiu pravdepodobnosť tým vzdialenostiam, ktoré sú menšie, resp. bližšie k prototypu danej hracej techniky. Na konci epizódy sa bude minimalizovať negatívna zlogaritmizovaná pravdepodobnosť očakávanej hracej techniky *c* pre každú latentnú reprezentáciu vzorky x_q v epizóde:

$$L(\theta) = -log\ p_\theta(y = c \mid x_q) \tag{72}$$

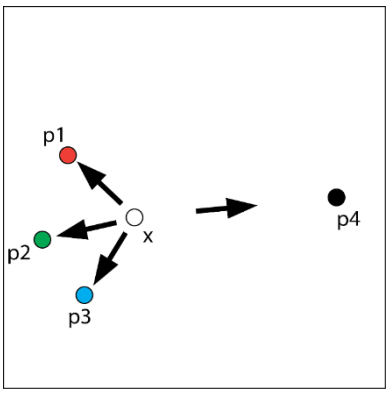
Takto natrénovaná CNN potom umožní akékoľvek hracie techniky bicích nástrojov premietnuť do diskrimantívneho priestoru, v ktorom sa budú latentné reprezantácie vzoriek tej istej

techniky združovať blízko pri sebe v jednom zhluku pričom rôzne zhluky budú v dostatočnej vzdialenosti od seba.

Inferencia

Inferenčná fáza, v ktorej sa robí automatická transkripcia hracích techník bicích nástrojov z rôznych skladieb, vyzerá nasledovne. V prvom rade treba opäť pripomenúť, že táto metóda pri procese automatickej transkripcie berie do úvahy ľudský faktor (viď Obrázok 35). Je to tak preto, lebo na to, aby táto metóda vôbec vedela, aké hracie techniky sa budú v neznámych skladbách z testovacej množiny nachádzať a ktoré bude mať CNN klasifikovať, musí z nich najprv vytvoriť prototypy. Tieto vzorky musí manuálne vybrať človek zo skladby určenej na transkripciu, keďže v reálnom prípade pri neznámej skladbe túto informáciu nebudeme mať automaticky k dispozícií. Potom ako človek zopár vzoriek pre každú hraciu techniku z danej skladby vyberie, sa zásah človekom do metódy končí.

Následne sa skladba, z ktorej sa robí transkripcia, rozseká na menšie časti – krátke Mel spektrogramy – a z nich sa vytvoria latentné reprezentácie pomocou CNN. Najprv sa všetky tieto vektory latentných reprezentácií spriemerujú, čím vznikne prototyp celej skladby. Potom sa opäť postupne prejdú všetky latentné reprezentácie a každá sa bude klasifikovať do žiadnej, jednej alebo viacerých hracích techník. To, či sa úder danou technikou v latentnej reprezentácií vybraného úseku skladby nachádza, sa určí binárnou klasifikáciou. Binárna klasifikácia sa bude robiť pre každú vybranú hraciu techniku pomocou funkcie softmax, ktorá sa aplikuje na vzdialenosti latentnej reprezentácie daného výseku skladby k prototypu celej skladby a k prototypu danej hracej techniky. To čiastočne môže ale aj nemusí byť problém.



Obrázok 37. Latentné reprezentácie prototypov hracích techník p1, p2, p3, prototypu skladby p4 a výseku skladby x

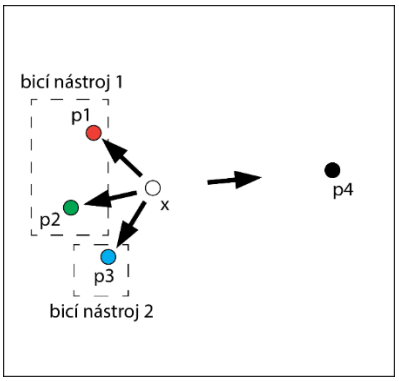
Binárna klasifikácia pre každú hraciu techniku nemusí byť problémom v prípade, ak prototypy hracích techník p1, p2, p3 z obrázka 37 patria rôznym nástrojom. To znamená, že v čase môže hrať bubeník na viacerých nástrojoch naraz, čo je v poriadku. Binárna klasifikácia však

môže byť problémová vtedy, ak prototypy hracích techník p1, p2, p3 patria tomu istému nástroju. To by znamenalo, že v jednom čase môže hrať bubeník na jeden nástroj viacerými technikami naraz, čo nechceme, lebo také niečo v skutočnosti nenastane. Ak by aj nastalo, tak je to empiricky nerozpoznateľné. Tu vidíme priestor na vylepšenie, ktoré uvedieme neskôr.

V jednom z predošlých odsekov sme spomínali, že táto metóda pri automatickej transkripcii bicích zo skladby ráta so zásahom človeka. V našom probléme, ktorý chceme touto metódou adresovať to dáva význam, keďže predtým, ako by bubeník začal hrať na bicie a nechal si robiť transkripicu toho, čo zahral, si vopred nakalibruje metódu vytvorením prototypov jednotlivých hracích techník z monofonických úderov na jednotlivé bicie nástroje jeho akustickej bicej sady.

6.3.3. Nami navrhnuté zmeny a vylepšenia

Našim hlavným vylepšením tejto existujúcej metódy z [47] je to, že v inferenčnej fáze metóda nerobí binárnu klasifikáciu každej hracej techniky ale n-árnu na základe toho, akým nástrojom prislúchajú jednotlivé prototypy hracích techník. Bližšie si to ilustrujeme na príklade uvedenom na obrázku 38.



Obrázok 38. Latentné reprezentácie protytpov hracích p1, p2 patria biciemu nástroju 1, prototyp hracej techniky p3 patrí biciemu nástroju 2. p4 je latentná reprezentácia prototypu skladby a x výseku skladby.

V prípade, že 2 prototypy hracích techník budú prislúchať jednému nástroju a tretí prototyp inému, pre danú latentnú reprezentáciu výseku skladby x sa nespraví binárna klasifikácia 3-krát, ale raz sa spraví binárna klasifikácia medzi p3 a p4 a raz sa spraví ternárna klasifikácia medzi p1, p2 a p4. Týmto zabezpečíme to, že vo výseku skladby bude klasifikátor schopný nájsť viacero zahraných techník naraz, no zamedzíme, aby tieto techniky prislúchali tomu istému biciemu nástroju.

Ďalšími menšími obmenami vzhľadom na náš špecifický typ problému je to, že tento model budeme trénovať na tom istom datasete (viac v sekcii 8.3.1), ktorý bol použitý v [47] len s tým rozdielom, že v skladbách nebudú prítomné harmonické nástroje. Ale chceme vyskúšať aj to, že ak tento model natrénujeme na skladbách s harmonickými nástrojmi a otestujeme ho na skladbách bez harmonickej zložky, aký vplyv to bude mať na diskriminatívny priestor CNN a celkovú úspešnosť metódy.

6.4. Zhrnutie

V tejto kapitole sme podrobne opísali 3 metódy, pričom 2 sú odvodené od už existujúcich metód (PFNMFD a Prototypická CNN) a jednu sme samy navrhli (NMFDQ). Tieto metódy implementujeme a spravíme potrebné experimenty na potvrdenie alebo vyvrátenie našich hypotéz o vlastnostiach metód a vplyve na ich úspešnosť.

7. Implementácia

Všetky algoritmy, ktoré sú uvedené v návrhu, som implementoval v jazyku Python vo verzii 3.7.5. Na samotnú implementáciu som využil školský linuxový server Ti1, ktorý disponuje výkonnou grafickou kartou *GeForce RTX 3090*. Pre efektívnu prácu, manažovanie a komentovanie kódu som použil *Jupyter lab*[3] - nástroj, ktorý poskytuje webové rozhranie pre manažment Jupyter notebook-ov. Všetky dôležité knižnice, ktoré boli použité pri implementácií, sú nasledovné:

- tensorflow (2.0.0)[4] – efektívny vývoj algoritmov strojového učenia
- torch (1.8.0)[5] – efektívny vývoj algoritmov strojového učenia
- librosa[6] – spracovanie audio signálu
- matplotlib[7] – vizualizácie
- numpy[8] – efektívna práca s poliami a maticami
- xml.etree.ElementTree[9] – parsovanie XML súboru
- tensorboard[10] – interaktívne webového rozhranie, ktoré nám pomôže pri monitorovaní správania jednotlivých algoritmov

[3] https://jupyterlab.readthedocs.io/en/stable/
[4] https://www.tensorflow.org/guide/effective_tf2
[5] https://pypi.org/project/torch/
[6] https://librosa.github.io/librosa/
[7] https://matplotlib.org/
[8] https://numpy.org/
[9] https://docs.python.org/2/library/xml.etree.elementtree.html
[10] https://www.tensorflow.org/tensorboard

8. Vyhodnotenie

V tejto kapitole vyhodnotíme všetky naše metódy, ktoré na záver zhrnieme a vyvodíme niekoľko záverov. Vo všetkých vizualizáciach a komentároch uvádzame skratky hracích techník bicích nástrojov, ktoré sú vysvetlené na začiatku práce v slovníku pojmov a skratiek.

8.1. PFNMFD

Pri vyhodnotení nás zaujímala transkripcia bicích nástrojov z polyfonickej hudby za prítomnosti harmonickej zložky (DTM). A to preto, lebo existujúce prístupy majú stále medzery. Z analýzy sme zistili, že jednotlivé modely nedosahovali vysoké hodnoty pri metrike *F-measure* (0.6 – 0.7) z čoho usudzujeme, že stále je čo zlepšovať Keďže sme si chceli overiť, či vôbec a ako dobre fungujú rôzne konfigurácie našich algoritmov, vyhodnotili sme ich aj na menej komplexnej úlohe – transkripcii bicích z polyfonickej hudby bez prítomnosti harmonickej zložky (DTD).

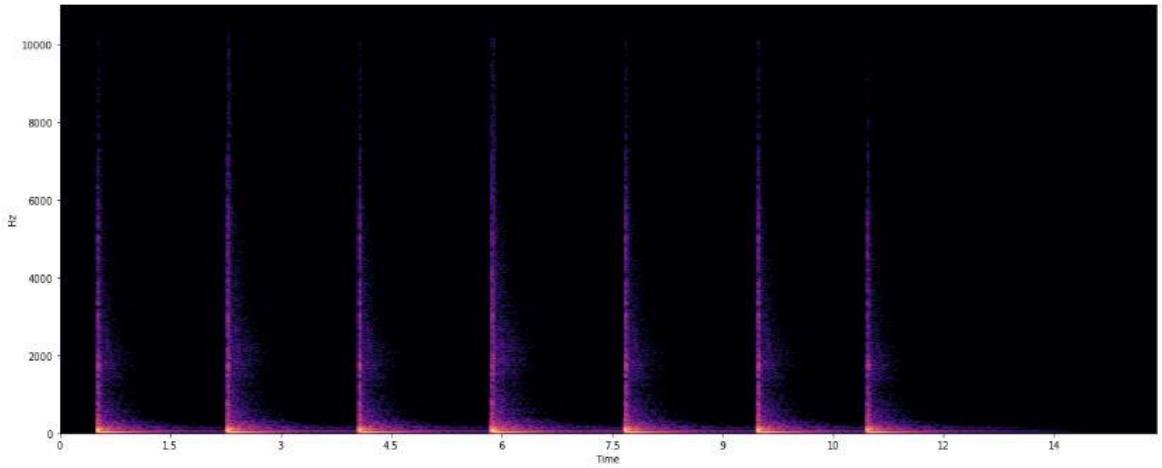
V tejto časti si najprv uvedieme, s akými datasetmi a koľkými nahrávkami sme pracovali, potom vysvetlíme postupnosť inicializácie bicích nástrojov a aké hodnoty parametrov sme využili, ako sme detegovali aktivácie a transformovali audio nahrávku pomocou STFT do spektrogramu a na záver uvedieme, ako sme rátali jednotlivé metriky. V poslednej časti uvediem výsledky modelov k DTD a DTM s patričnými komentármi.

8.1.1. Datasety

Pri riešení problému DTD sme vyhodnocovali metódy na podmnožine *RealDrum* datasetu *IDMT-SMT-Drums*. Tá obsahuje 20 skladieb. V týchto nahrávkach hrá bubeník polyfonicky na akustickej súprave bicích. Hrá len na rytmičáku, kopáku a hajtke. Na určenie rozdielu medzi nami predikovanými aktiváciami a skutočnými sme pri vyhodnotení použili XML súbor, ktorý obsahoval časy, v ktorých boli jednotlivé bicie nástroje aktívne. Pri DTM sme testovali metódy na podmnožine *Drummer1* datasetu *ENST-Drums* [53]. Tá obsahuje 21 skladieb. Na nahrávkach je okrem bicích počuť gitaru, basu, klavír a iné harmonické nástroje. Na určenie presností rozdielu medzi nami predikovanými aktiváciami a skutočnými sme použili pri vyhodnotení textový súbor, ktorý obsahoval časy, v ktorých boli zaznamenané aktivity bicích nástrojov. Oba datasety používame preto, aby sme sa vedeli potom čiastočne porovnať s najmodernejšími prístupmi v ADT.

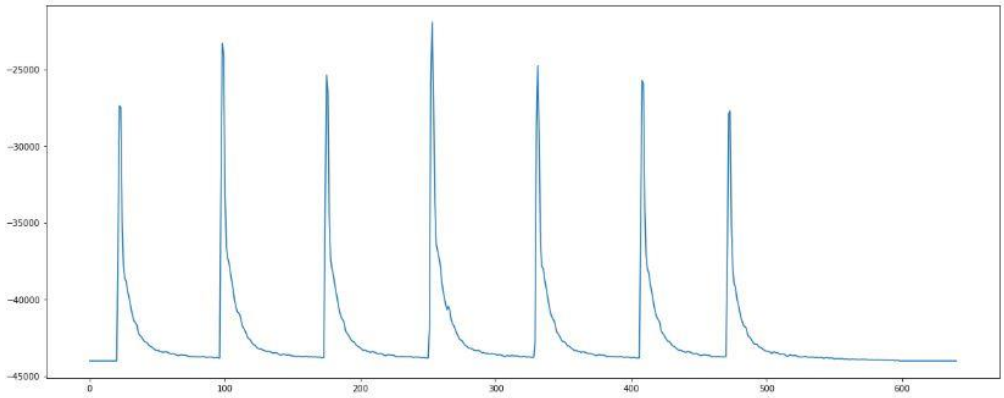
8.1.2. Inicializácia bicích nástrojov

Na inicializáciu sme použili nahrávky z datasetu *IDMT-SMT-Drums* [51], kde pre každý sledovaný bicí nástroj (kopák, hajtka, rytmičák) je 95 nahrávok s monofonickými údermi. Jedna nahrávka obsahuje poväčšine viacero úderov na daný nástroj pričom medzi za sebou nasledujúcimi údermi je pauza. Treba podotknúť, že spôsob inicializácie je odlišný pre algoritmus PFNMF2 a náš PFNMFD. A to preto, lebo pri PFNMF2 je komponent hudobného nástroja reprezentovaný vektorom pričom v nami navrhnutom algoritme je reprezentovaný maticou. Oba spôsoby pozostávajú z troch krokov – 1. zistenie aktivácií v nahrávke, 2. výpočet priemerného spektra nástroja z danej nahrávky, 3. výpočet priemerného spektra nástroja z danej množiny nahrávok. Prvý krok tejto fázy je spoločný. Uvedieme si ho na príklade, ktorým je spektrogram nahrávky, v ktorej sa hralo monofonicky na kopáku.



Obrázok 39 Spektrogram z nahrávky, v ktorej sa hralo monofonicky na kopák

Ak sčítame magnitúdy všetkých frekvencií pre každý rámec (vertikálne) z obrázku 39, dostaneme celkovú energiu pre každý rámec. Výsledkom je vektor, ktorý keď si vykreslíme, bude vyzerať nasledovne



Obrázok 40. Aktivačné funkcie zo spektrogramu nahrávky, v ktorej sa hralo monofonicky na kopák

Ako vidíme na obrázku 40, týmto spôsobom si vieme zistiť ako veľmi a kedy je nástroj aktívny. Pomocou algoritmu na detekciu aktivácií, ktorý pozostáva zo vzorcov 54, 55, 56 s menšou obmenou, sme si potom vieme zistiť ich presné pozície a takýto vektor pozícií môže poslúžiť ako vstup do druhého kroku.

Druhý krok je pre oba algoritmy odlišný. Pri použití PFNMF2, kde je nástroj reprezentovaný vektorom, priemerné spektrum nástroja sa vypočíta tak, že sa vyberú zo spektrogramu nahrávky tie rámce, v ktorých nastali aktivácie. Následne sa z nich spraví priemer. Pri našom PFNMFD to je o čosi zložitejšie. Podľa toho, koľko rámcov v matici komponentov má modelovať hudobný nástroj, toľko rámcov sa zoberie zo spektrogramu pri každej aktivácií. Ak náhodou nastane situácia, že nástroj je aktívny na konci nahrávky a s počtom rámcov to nevychádza, vývoj spektrogramu danej aktivácie sa umelo doaproximuje. Pridá sa toľko rámcov, aby sedel s rozmermi ostatných matíc rámcov a aby sa vedel na konci s ostatnými spriemerovať.

Tretí krok je opäť spoločný. Podľa toho, koľko nahrávok je k dispozícií (my sme mali 95 pre každý bicí nástroj), toľko krát sa kroky 1 a 2 zopakujú a na záver sa všetky priemerné spektrá spriemerujú, čím dostaneme priemerné spektrá nástrojov z danej množiny nahrávok

Takto inicializované komponenty jednotlivých bicích nástrojov poslúžili ako vstup do rôznych variant algoritmu NMF – PFNMF2 alebo PFNMFD.

8.1.3. Predspracovanie signálu

Ako vo väčšine najmodernejších prístupov riešiace problém ADT, tak aj v našom prípade sme použili na spracovanie audio nahrávok magnitudový STFT spektrogram. Pri analýze výskumnej oblasti ADT sme si však všimli, že niektorí autori skúšali aj iné predspracovania signálu, čo znamená, že algoritmy strojového učenia narábali s inou vstupnou reprezentáciou – napr. MCMS [45]. Keďže sme sa chceli porovnať s väčšinou najmodernejších prístupov, preto sme používali STFT. Do budúcna sme však otvorení iným modifikáciám vstupu.

Pri aplikovaní STFT na audio nahrávku sme museli uviesť typ okienka, veľkosť okienka (angl. *window size*) a veľkosť skoku (angl. *hop length*). Aby sme sa mohli neskôr porovnávať s najmodernejšími prístupmi, potrebovali sme spracovať vstup rovnakým spôsobom. Preto sme použili Hammingové okienko o veľkosti 2048 s veľkosťou skoku = ¼ veľkosti okienka, čiže 512.

8.1.4. Detekcia aktivácií

Ešte z návrhu metódy vieme, že po skončení faktorizácie matica aktivácií bicích slúži ako vstup do algoritmu detekcie aktivácií. Ten pozostáva z pravidiel, ktoré sú vyjadrené vzorcami 54, 55, 56. Úlohou algoritmu je vyfiltrovať reálne aktivácie. Aby algoritmus mohol určiť, či nejaké pozorovanie z aktivačnej funkcie daného nástroja, t.j. $H_{D(i)}$ pričom $i \in [0,1,2]$ (3 indexy, lebo máme 3 bicie nástroje) je naozaj aktivácia, museli sme určiť

- **pre_avg** a **post_avg** – či je jeho hodnota väčšia ako priemer z niekoľkých pozorovaní pred a po ňom – parametre sme nastavili na hodnotu 5
- **pre_max** a **post_max** – či je jeho hodnota väčšia ako maximum z niekoľkých pozorovaní pred a po ňom – parametre sme nastavali na hodnotu 4
- **wait** – aký minimálny počet pozorovaní musí byť medzi dvoma za sebou nasledujúcimi aktiváciami – parameter sme nastavili na hodnotu 4
- **threshold** – akou minimálnou hodnotou musí disponovať pozorovanie, ktoré je potenciálnou aktiváciou – parameter sme nastavili na $0.1*[\max(H_{D(i)})]$, t.j. hodnota musí byť aspoň 0.1 násobok najsilnejšieho úderu zo všetkých aktivácií pre daný nástroj

Parametre sme nastavili na základe empirického odpozorovania pri ktorom sa nám zdalo, že v danej konfigurácií môže detekcia fungovať správne. Toto nie je najšťastnejšie riešenie a do budúcna si vieme predstaviť použiť nejaký automatizovaný spôsob na nájdenie vhodných hodnôt parametrov. Pre lepšiu prehľadnosť uvádzam malú tabuľku, v ktorej je nastavenie parametrov prehľadne spísané.

Tabuľka 3. Nastavenie parametrov algoritmu detekcie aktivácií pre metódy PFNMFD a PFNMF2

Parametre algoritmu detekcie aktivácií					
pre_max	post_max	pre_avg	post_avg	threshold	wait
4	4	5	5	0.1*max(HD(i))	4

Treba zdôrazniť, že detekcia aktivácií sa použila pri inicializácií bicích komponentov a taktiež pri samotnej transkripcií. Hodnoty parametrov, ktoré sú vyššie uvedené, boli v oboch prípadoch rovnaké.

8.1.5. Metriky a tolerancia

Na vyhodnocovanie úspešnosti modelov sme používali metriky *F-measure, Precision, Recall*. Pomocou nich vyhodnocovali úspešnosť modelov aj ďalší autori, ktorých rozbor článkov sme uviedli v analýze najmodernejších prístupov ADT. Na to, aby sme mohli zapísať vzorce metrík, potrebujeme najprv uviesť tabuľku, z ktorej sa odvádza výpočet metrík

Tabuľka 4. Tabuľka, ktorá slúži ako podklad pre vysvetlenie rátania metrík

		Predicted	
		Positive	Negative
Actual	Positive	TP	FN
	Negative	FP	TN

Tak ako je uvedené v [55], metriky sa rátajú nasledovným spôsobom

$$Precision = \frac{TP}{TP+FP}, \quad Recall = \frac{TP}{TP+FN}, \quad F1 = 2\,x\,\frac{Precision * Recall}{Precision+Recall} \qquad (73)$$

Pri vyhodnocovaní nebudeme uvádzať všetky hodnoty metrík pre každú skladbu ale ich priemerné hodnoty spomedzi všetkých skladieb. Taktiež budeme uvádzať priemerné hodnoty metrík pre každý bicí nástroj.

Treba podotknúť, že pri určení, či je nami predikovaná aktivácia skutočnou aktiváciou (tie máme dostupné zo súborov, ktoré uvádzam v sekcii 8.1.1) tolerujeme to, ak je čas našej predikovanej aktivácie vychýlený maximálne o 50 milisekúnd pred alebo po nastatí skutočnej aktivácie. Toleranciu 50ms uvádzajú aj rôzne ďalšie prístupy, ktoré sme spomenuli v analýze.

8.1.6. Výsledky algoritmov

V tejto časti uvediem vyhodnotenia rôznych konfigurácií PFNMF2 a PFNMFD algoritmov. Tie budem uvádzať v prehľadných tabuľkách. Na to, aby sme tabuľkám porozumeli, treba najprv vysvetliť význam skrátených pojmov, ktoré sa v nich nachádzajú

- **opt.fun** – optimalizačná funkcia. Tá môže nadobudať hodnoty *mu* (multiplikatívne aktualizácie) alebo *pg* (projected gradient descent = projektovaný zostupujúci gradient)
- **rýchl.uč.** – rýchlosť učenia. Používa sa iba pri optimalizácií matíc pomocou *pg*
- **chyb.fun.** – chybová funkcia. Tá môže nadobúdať hodnoty *ed* (Euklidovská vzdialenosť) alebo *kl* (KL-divergencia)
- **riedkosť** – obmedzenie riedkosti. Tá môže nadobúdať ľubovoľné reálne hodnoty. Budeme uvádzať hodnoty z intervalu <0;1>. Čím vyššia hodnota, tým je matica H redšia.
- **uprav W_D** – či sa má dodatočne upraviť matica W_D. Tá bude obsahovať hodnoty Áno alebo Nie.

Teraz uvedieme výsledky algoritmov, ktoré riešili problém DTD a potom DTM

1. Experiment - DTD

V tejto časti uvediem vyhodnotenia algoritmov, ktoré riešili problém transkripcie bicích nástrojov z polyfonickej hudby bez harmonickej zložky.

V nasledujúcej tabuľke uvádzame rôzne konfigurácie algoritmu PFNMF2 spolu s ich výsledkami. Počet iterácií bol rovný 1000.

Tabuľka 5. Výsledky rôznych konfigurácií algoritmu PFNMF2 na podmnožine RealDrum datasetu IDMT-SMT-Drums

	Konfigurácia algoritmu					KD			SD			HH			Všeobecne		
ID	opt.fun.	rýchl. uč.	chyb.fun.	riedkosť	uprav Wd	P	R	F	P	R	F	P	R	F	P	R	F
1	mu	-	ed	0	Nie	1.0	0.99	0.99	0.85	1.0	0.90	0.85	0.84	0.84	0.90	0.94	0.91
2	mu	-	ed	0.5	Nie	1.0	0.99	0.99	0.86	1.0	0.91	0.85	0.69	0.74	0.90	0.89	0.88
3	mu	-	ed	0	Áno	1.0	0.99	0.99	0.86	1.0	0.91	0.84	0.84	0.84	0.90	0.94	0.91
4	mu	-	ed	0.5	Áno	1.0	0.99	0.99	0.86	1.0	0.91	0.86	0.69	0.75	0.90	0.89	0.88
5	mu	-	kd	0	Nie	1.0	0.99	0.99	0.86	1.0	0.91	0.86	0.69	0.75	0.90	0.89	0.88
6	mu	-	kd	0.5	Nie	0.76	0.99	0.85	0.56	0.99	0.70	0.86	0.87	0.87	0.73	0.95	0.81
7	mu	-	kd	0	Áno	0.99	0.99	0.99	0.83	1.0	0.89	0.84	0.84	0.84	0.89	0.94	0.91
8	mu	-	kd	0.5	Áno	0.68	0.99	0.80	0.50	0.99	0.65	0.88	0.88	0.88	0.68	0.95	0.77
9	pg	0.15	ed	0	Nie	1.0	0.99	0.99	0.86	1.0	0.91	0.84	0.84	0.84	0.90	0.94	0.91
10	pg	0.15	ed	0.5	Nie	1.0	0.99	0.99	0.81	1.0	0.88	0.85	0.46	0.55	0.89	0.82	0.81
11	pg	0.15	ed	0	Áno	1.0	0.99	0.99	0.85	1.0	0.91	0.84	0.84	0.83	0.90	0.94	0.91
12	pg	0.15	ed	0.5	Áno	1.0	0.99	0.99	0.81	1.0	0.88	0.84	0.49	0.56	0.88	0.83	0.81

V ďalšej tabuľke uvádzame rôzne konfigurácie algoritmu PFNMFD spolu s ich výsledkami. Počet iterácií každého algoritmu bol 500 a počet rámcov, ktoré modelovali spektrálno-časový vývoj bicích nástrojov v matici komponentov, bol nastavený na hodnotu 10.

Tabuľka 6. Výsledky rôznych konfigurácií algoritmu PFNMFD na podmnožine RealDrum datasetu IDMT-SMT-Drums

	Konfigurácia algoritmu					KD			SD			HH			Všeobecne		
ID	opt.fun.	rýchl. uč.	chyb.fun.	riedkosť	uprav Wd	P	R	F	P	R	F	P	R	F	P	R	F
1	mu	-	ed	0	Nie	0.88	0.98	0.91	0.77	1.0	0.86	0.96	0.88	0.91	0.87	0.95	0.90
2	mu	-	ed	0.5	Nie	0.88	0.97	0.91	0.76	0.99	0.85	0.96	0.88	0.92	0.87	0.95	0.89
3	mu	-	ed	0	Áno	0.87	0.97	0.91	0.76	1.0	0.85	0.96	0.89	0.92	0.87	0.95	0.89
4	mu	-	ed	0.5	Áno	0.88	0.97	0.91	0.74	1.0	0.84	0.96	0.89	0.92	0.86	0.95	0.89
5	mu	-	kd	0	Nie	0.88	0.97	0.91	0.68	0.98	0.78	0.95	0.93	0.94	0.84	0.96	0.88
6	mu	-	kd	0.5	Nie	0.89	0.98	0.92	0.68	0.98	0.78	0.95	0.93	0.94	0.84	0.96	0.88
7	mu	-	kd	0	Áno	0.89	0.97	0.92	0.68	0.98	0.78	0.95	0.93	0.94	0.84	0.96	0.88
8	mu	-	kd	0.5	Áno	0.89	0.98	0.92	0.68	0.98	0.78	0.95	0.93	0.94	0.84	0.96	0.88
9	pg	0.15	ed	0	Nie	0.96	0.99	0.97	0.80	1.0	0.87	0.96	0.89	0.93	0.91	0.96	0.92
10	pg	0.15	ed	0.5	Nie	0.97	0.99	0.98	0.70	1.0	0.80	0.93	0.41	0.52	0.86	0.80	0.77
11	pg	0.15	ed	0	Áno	0.96	0.99	0.97	0.80	1.0	0.87	0.95	0.89	0.92	0.90	0.96	0.92
12	pg	0.15	ed	0.5	Áno	0.97	0.99	0.98	0.69	1.0	0.80	0.94	0.36	0.48	0.87	0.79	0.75

Zvýraznené riadky hovoria o tom, ktoré algoritmy boli najlepšie spomedzi ostatných na základe metriky celkovej *F-measure* . V tabuľke 5 vidíme, že viacero modelov malo rovnako dobrý výsledok – 0.91. Môžeme si všimnúť, že ani jeden z nich nepoužíva obmedzenie riedkosti (hodnota 0 v stĺpci riedkosť). Môže to znamenať, že hodnota 0.5, ktorú sme používali bola príliš vysoká. V súvislosti s aplikovaním riedkosti si tiež môžeme všimnúť, že metrika *Recall* pri hi-hat činely výrazne klesla. Dávame to za príčinu jej vysoko frekventovanému výskytu v skladbách. A ak parameter riedkosti bol príliš veľký, s veľkou

pravdepodobnosťou mohol spôsobiť zmazanie niektorých jej úderov. Ďalej vidíme, že modely využívajúce MU, ktoré používajú Euklidovskú vzdialenosť, majú vo všeobecnosti o trochu lepšie výsledky ako tie, čo používajú KL-divergenciu. Taktiež si môžeme všimnúť, že pri použití multiplikatívnych aktualizácií a KL-divergencie sa úspešnosť modelu zlepšila vďaka dodatočnej úprave matice komponentov W_D. Dôvodom sú zlepšené hodnoty metrík hi-hat činely. Ďalej vidíme, že prestrelená hodnota parametra riedkosti oveľa viac uškodila algoritmom s PG ako s MU. To znamená, že metóda PG je na neho veľmi citlivá. Vo všeobecnosti si môžeme všimnúť, že algoritmy majú najväčší problém s modelovaním hi-hat. Dáva nám to zmysel, lebo jej zvuk v skladbách zvykne byť veľmi rôznorodý. Taktiež si pri porovnaní všetkých algoritmov môžeme všimnúť, že to, či sa matice upravujú pomocou MU alebo PG je jedno. Rozdiely medzi nimi nejaké sú, ale nie veľké. Pri použití metódy PG však treba dať pozor na vhodne zvolenú rýchlosť učenia.

V tabuľke 6 jednoznačne dominujú PFNMFD algoritmy, ktoré používajú na úpravu matíc metódu PG. Navyše majú o niečo lepšie výsledky ako najlepšie algoritmy pri použití PFNMF2. Ďalej si môžeme všimnúť, že vo všeobecnosti algoritmy oveľa lepšie modelujú hi-hat. Dôkazom sú lepšie metriky oproti PFNMF2. Aj tu platí, že modely MU, ktoré používajú Euklidovskú vzdialenosť, majú vo všeobecnosti lepšie výsledky ako tie, čo používajú KL-divergenciu. Ďalej môžeme spozorovať, že algoritmy MU sú rezistentnejšie oproti prepálenej hodnote parametra riedkosti. Neplatí to však pre metódu PG, ktorá je aj v tomto prípade na neho veľmi citlivá. Ďalej vidíme, že dodatočná úprava matice W_D nijako nepomáha.

2. Experiment - DTM

V tejto časti uvedieme vyhodnotenia algoritmov, ktoré riešili problém transkripcie bicích nástrojov z polyfonickej hudby za prítomnosti harmonickej zložky.

V nasledujúcej tabuľke uvádzame rôzne konfigurácie algoritmu PFNMF2 spolu s ich výsledkami. Počet iterácií bol rovný 1000 a počet harmonických nástrojov (parameter r_h) v matici komponentov pre harmonické nástroje bol nastavený na hodnotu 50.

Tabuľka 7. Výsledky rôznych konfigurácií algoritmu PFNMF2 na pdmnožine Drummer1 datasetu ENST-Drums, pričom r_h je rovné hodnote 50

	Konfigurácia algoritmu					KD			SD			HH			Všeobecne		
ID	opt.fun.	rýchl. uč.	chyb.fun.	riedkosť	uprav WD	P	R	F	P	R	F	P	R	F	P	R	F
1	mu	-	ed	0	Nie	0.67	0.82	0.72	0.71	0.62	0.62	0.56	0.56	0.52	0.65	0.67	0.62
2	mu	-	ed	0.5	Nie	0.70	0.77	0.71	0.82	0.42	0.51	0.56	0.16	0.18	0.69	0.45	0.47
3	mu	-	ed	0	Áno	0.64	0.82	0.70	0.70	0.63	0.62	0.57	0.56	0.51	0.64	0.67	0.61
4	mu	-	ed	0.5	Áno	0.72	0.75	0.71	0.82	0.44	0.51	0.56	0.13	0.15	0.70	0.44	0.46
5	mu	-	kd	0	Nie	0.64	0.84	0.71	0.79	0.63	0.67	0.63	0.66	0.60	0.69	0.71	0.66
6	mu	-	kd	0.5	Nie	0.54	0.33	0.37	0.60	0.13	0.18	0.51	0.15	0.22	0.55	0.21	0.26
7	mu	-	kd	0	Áno	0.62	0.84	0.70	0.79	0.63	0.65	0.66	0.68	0.61	0.69	0.72	0.65
8	mu	-	kd	0.5	Áno	0.56	0.34	0.40	0.61	0.09	0.15	0.54	0.16	0.23	0.57	0.20	0.26
9	pg	0.15	ed	0	Nie	0.64	0.59	0.59	0.59	0.60	0.56	0.60	0.52	0.52	0.61	0.57	0.55
10	pg	0.15	ed	0.5	Nie	0.64	0.60	0.59	0.60	0.61	0.57	0.6	0.43	0.47	0.61	0.55	0.55
11	pg	0.15	ed	0	Áno	0.65	0.60	0.60	0.60	0.61	0.57	0.60	0.51	0.51	0.62	0.57	0.56
12	pg	0.15	ed	0.5	Áno	0.63	0.58	0.58	0.59	0.59	0.55	0.61	0.45	0.48	0.61	0.54	0.54

V ďalšej tabuľke uvádzame tie isté konfigurácie PFNMF2 len s tou zmenou, že algoritmy sa budú snažiť modelovať 100 harmonických komponentov (r_h = 100)

Tabuľka 8. Výsledky rôznych konfigurácií algoritmu PFNMF2 na pdmnožine Drummer1 datasetu ENST-Drums, pričom r_h je rovné hodnote 100

	Konfigurácia algoritmu					KD			SD			HH			Všeobecne		
ID	opt.fun.	rýchl. uč.	chyb.fun.	riedkosť	uprav WD	P	R	F	P	R	F	P	R	F	P	R	F
1	mu	-	ed	0	Nie	0.65	0.83	0.71	0.78	0.62	0.65	0.57	0.53	0.50	0.67	0.66	0.62
2	mu	-	ed	0.5	Nie	0.69	0.78	0.71	0.84	0.44	0.53	0.61	0.08	0.14	0.71	0.44	0.46
3	mu	-	ed	0	Áno	0.65	0.84	0.72	0.78	0.62	0.65	0.56	0.52	0.50	0.67	0.66	0.62
4	mu	-	ed	0.5	Áno	0.68	0.78	0.71	0.86	0.42	0.52	0.65	0.13	0.19	0.73	0.44	0.47
5	mu	-	kd	0	Nie	0.65	0.85	0.72	0.80	0.64	0.67	0.65	0.69	0.62	0.70	0.72	0.67
6	mu	-	kd	0.5	Nie	0.59	0.38	0.42	0.67	0.14	0.21	0.58	0.33	0.40	0.60	0.28	0.34
7	mu	-	kd	0	Áno	0.63	0.85	0.71	0.84	0.63	0.68	0.65	0.67	0.61	0.71	0.72	0.67
8	mu	-	kd	0.5	Áno	0.60	0.37	0.41	0.66	0.15	0.23	0.58	0.30	0.36	0.61	0.28	0.33
9	pg	0.0005	ed	0	Nie	0.67	0.72	0.68	0.49	0.72	0.56	0.62	0.54	0.50	0.59	0.66	0.58
10	pg	0.0005	ed	0.5	Nie	0.67	0.72	0.68	0.50	0.72	0.56	0.63	0.57	0.52	0.60	0.67	0.59
11	pg	0.0005	ed	0	Áno	0.65	0.71	0.67	0.48	0.72	0.55	0.61	0.53	0.50	0.58	0.65	0.57
12	pg	0.0005	ed	0.5	Áno	0.67	0.73	0.68	0.49	0.71	0.56	0.63	0.55	0.51	0.59	0.67	0.58

V nadchádzajúcej tabuľke uvádzame rôzne konfigurácie algoritmu PFNMFD spolu s ich výsledkami. Počet iterácií algoritmov bol 500 a počet rámcov, ktoré modelovali spektrálno-časový vývoj bicích nástrojov v matici komponentov, bol nastavený na hodnotu 10. Počet harmonických nástrojov, ktoré algoritmy modelovali v matici komponentov pre harmonické nástroje, bolo 50 (r_h = 50).

Table 9. Výsledky rôznych konfigurácií algoritmu PFNMFD na pdmnožine Drummer1 datasetu ENST-Drums, pričom r_h = 50

	Konfigurácia algoritmu					KD			SD			HH			Všeobecne		
ID	opt.fun.	rýchl. uč.	chyb.fun.	riedkosť	uprav WD	P	R	F	P	R	F	P	R	F	P	R	F
1	mu	-	ed	0	Nie	0.42	0.05	0.08	0.25	0.03	0.05	0.24	0.00	0.01	0.30	0.03	0.05
2	mu	-	ed	0.5	Nie	0.32	0.50	0.37	0.24	0.46	0.30	0.27	0.42	0.30	0.28	0.46	0.33
3	mu	-	ed	0	Áno	0.44	0.06	0.08	0.23	0.02	0.04	0.26	0.01	0.02	0.31	0.03	0.05
4	mu	-	ed	0.5	Áno	0.31	0.53	0.38	0.25	0.48	0.31	0.26	0.40	0.29	0.27	0.47	0.33
5	mu	-	kd	0	Nie	0.28	0.49	0.34	0.25	0.48	0.32	0.29	0.39	0.30	0.27	0.45	0.32
6	mu	-	kd	0.5	Nie	0.28	0.52	0.35	0.24	0.47	0.30	0.28	0.41	0.31	0.27	0.46	0.32
7	mu	-	kd	0	Áno	0.28	0.52	0.35	0.27	0.49	0.33	0.28	0.39	0.29	0.27	0.47	0.32
8	mu	-	kd	0.5	Áno	0.29	0.53	0.36	0.25	0.48	0.31	0.28	0.40	0.30	0.28	0.47	0.32
9	pg	0.0001	ed	0	Nie	0.20	0.50	0.28	0.19	0.47	0.25	0.22	0.46	0.28	0.20	0.48	0.27
10	pg	0.0001	ed	0.5	Nie	0.20	0.52	0.28	0.20	0.52	0.27	0.22	0.50	0.29	0.21	0.51	0.28
11	pg	0.0001	ed	0	Áno	0.20	0.49	0.27	0.20	0.52	0.27	0.23	0.5	0.30	0.21	0.51	0.28
12	pg	0.0001	ed	0.5	Áno	0.19	0.47	0.26	0.19	0.50	0.26	0.22	0.46	0.28	0.20	0.48	0.27

V ďalšej tabuľke uvádzame tie isté konfigurácie PFNMFD len s tou zmenou, že algoritmy sa snažili modelovať 100 harmonických komponentov (r_h = 100)

Table 10. Výsledky konfigurácií algoritmu PFNMFD na pdmnožine Drummer1 datasetu ENST-Drums, pričom r_h = 100

	Konfigurácia algoritmu					KD			SD			HH			Všeobecne		
ID	opt.fun.	rýchl. uč.	chyb.fun.	riedkosť	uprav Wd	P	R	F	P	R	F	P	R	F	P	R	F
1	mu	-	ed	0	Nie	0.41	0.25	0.28	0.68	0.14	0.22	0.68	0.15	0.23	0.59	0.18	0.24
2	mu	-	ed	0.5	Nie	0.41	0.23	0.27	0.65	0.13	0.20	0.63	0.14	0.21	0.56	0.17	0.23
3	mu	-	ed	0	Áno	0.38	0.22	0.25	0.70	0.14	0.22	0.63	0.15	0.21	0.57	0.17	0.22
4	mu	-	ed	0.5	Áno	0.41	0.22	0.26	0.71	0.14	0.22	0.63	0.13	0.20	0.58	0.17	0.23
5	mu	-	kd	0	Nie	0.44	0.33	0.35	0.88	0.23	0.35	0.61	0.24	0.31	0.64	0.27	0.34
6	mu	-	kd	0.5	Nie	0.46	0.32	0.35	0.90	0.20	0.31	0.68	0.26	0.35	0.68	0.26	0.34
7	mu	-	kd	0	Áno	0.43	0.32	0.35	0.80	0.24	0.35	0.69	0.26	0.35	0.64	0.27	0.35
8	mu	-	kd	0.5	Áno	0.43	0.32	0.33	0.84	0.23	0.34	0.70	0.27	0.36	0.65	0.27	0.39
9	pg	0.00001	ed	0	Nie	0.77	0.82	0.78	0.78	0.67	0.67	0.53	0.15	0.19	0.69	0.55	0.55
10	pg	0.00001	ed	0.5	Nie	0.77	0.82	0.78	0.79	0.66	0.68	0.56	0.11	0.14	0.70	0.53	0.53
11	pg	0.00001	ed	0	Áno	0.77	0.82	0.78	0.78	0.67	0.8	0.48	0.12	0.15	0.67	0.54	0.53
12	pg	0.00001	ed	0.5	Áno	0.77	0.82	0.78	0.79	0.67	0.68	0.55	0.11	0.13	0.70	0.53	0.53

Môžeme si všimnúť, že všeobecne metriky klesli oproti PFNMF2 a PFNMFD pri riešení problému DTD. Je to logické, pretože transkripcia bicích z polyfonickej hudby za prítomnosti harmonickej zložky je náročnejší problém. Môžeme si všimnúť, že najlepšie modely pri oboch verziách PFNMF2 dosahujú metriku *F-measure* okolo 0.66. Pri použití PFNMF2 s r_h = 50 a r_h = 100 vidíme, že zväčšenie hodnoty tohto parametra vo všeobecnosti nezlepšilo ani nezhoršilo výsledky. Dáva to zmysel, pretože v [37] uvádzajú, že použitie váhovej matice A zmierňuje vplyv parametra r_h na robustnosť algoritmu a teda aj transkripciu. Ďalej si môžeme všimnúť, že parameter riedkosti znižuje úspešnosť všetkých modelov. Najmä pri použití MU a KL-divergencie. Paradoxne pri metóde PG to teraz neplatí. Dávame to za dôsledok tomu, že pri komplexnejšej nahrávke takouto pomerne ustrelenou hodnotou riedkosti sa zredukoval len šum (harmonické nástroje), pretože metriky to nezlepšilo. Ďalej si treba všimnúť, že rýchlosť učenia sme pri použití metódy PG oveľa zmenšili, keď bola hodnota parametra r_h = 100. Predpokladáme, že to je hlavne kvôli pridaniu veľkého množstva premenných, ktoré vplývajú na chybovú funkciu. Preto veľkosti krokov, ktoré určuje rýchlosť učenia, musia byť menšie. Ďalej vidíme, že nemožno jasne určiť, ktorý nástroj je najproblematickejšie modelovať.

Pri PFNMFD je to o čosi horšie. Ak bol parameter r_h = 50, najlepší model dosahoval *F-measure* 0.33. Pri r_h = 100 to bolo 0.55. Vidíme, že zmena tohto parametra veľmi neovplyvnila úspešnosti algoritmov s MU ale iba s tými, ktoré používali metódu zostupujúceho gradientu. Jednak najlepším modelom z PFNMFD pri r_h = 100 sú tie čo používajú metódu zostupujúceho gradientu ale aj vo všeobecnosti majú lepšie hodnoty vo väčšinách metrík. Ďalej vidíme, že obmedzenie riedkosti zlepšilo úspešnosť niektorých modelov. Môže to byť preto,

lebo aktivácie nástrojov odstránili šum a algoritmus na detekciu aktivácií tak mohol lepšie určiť, kedy naozaj nastali. Môžeme si všimnúť, že zväčšením parametra r_h sme museli opäť o niečo znížiť rýchlosť učenia. Vo všeobecnosti vidíme, že algoritmy majú najväčší problém s modelovaním hi-hat činely.

8.1.7. Zhrnutie

Z vyhodnotenia môžeme usúdiť niekoľko záverov pre DTD. Treba opatrnejšie narábať s parametrom riedkosti pri PFNMF algoritmoch. Obzvlášť vtedy, ak používame metódu PG. Algoritmy majú vo všeobecnosti najväčší problém s transkripciou hi-hat činely. PFNMFD lepšie modelujú zvuk hi-hat, čo sa odzrkadlilo aj na zlepšených metrikách. Je otázne, či uvedený spôsob upravovania matice W_D naozaj pomáha. Rozdiel medzi použitím NMF algoritmov s MU alebo PG nie je až taký razantný.

Môžeme usúdiť aj niekoľko záverov pre DTM. Vo všeobecnosti úspešnosti algoritmov klesli oproti tým, ktoré dosiahli pri DTD. Zmena hodnoty parametra r_h pri PFNMF pravdepodobne nemá vplyv na úspešnosti modelov. Použitie obmedzenia riedkosti malo pozitívny vplyv pri PFNMFD algoritmoch. Taktiež PFNMFD s použitím metódy PG preukázal snubné výsledky oproti ostatným konfiguráciám. Treba však myslieť na to, že je veľmi citlivý na rýchlosť učenia. No výsledky sú stále o niečo menšie ako v PFNMF2.

8.2. NMFDQ

Metódu NMFDQ sme vyhodnocovali na skladbách s polyfonicky znejúcich bicích nástrojov bez prítomnosti harmonickej zložky. Táto podsekcia bude mať podobnú štruktúru ako vyhodnotenie predošlej metódy. To znamená, že najprv si povieme niečo v skratke o datasetoch, ktoré sme použili na trénovanie, inicializáciu a vyhodnotenie, potom opíšeme ako sme inicializovali bicie nástroje, ako sme trénovali CNN, opíšeme rôzne varianty predspracovania, ktoré sme používali, definujeme hodnoty parametrov algoritmu na detekciu aktivácií a na záver si uvedieme vizualizácie tých najzaujímavejších experimentov, ktoré patrične okomentujeme.

8.2.1. Datasety

Na trénovanie CNN, inicializáciu bicích nástrojov a vyhodnotenie našej metódy sme použili 2 datasety – *MDB Drums* a *SampleRadar*.

MDB Drums vznikol pomerne nedávno (2017). Stal sa zároveň referenčným datasetom pri porovnávaní najmodernejších prístupov v ADT doméne. Obsahuje 23 nahrávok, ktoré majú

dokopy 7994 úderov na bicie. Každá nahrávka sa vyskytuje 2-krát, pričom v jednej znejú len bicie nástroje, v druhej znejú bicie aj spolu s harmonickými nástrojmi v pozadí. Nás však zaujímali len tie nahrávky, v ktorých zneli len bicie. [56]. Tým, že sa orientujeme na 3 bicie nástroje, transkripciu sme robili z nasledovných hracích techník

- **Hi-hat činel:** CHH, PHH, OHH
- **Kopák:** KD
- **Rytmičák:** SD, SDB, SDF, SDG, SDNS, SST

Hoci tento dataset disponuje rôznymi hracími technikami, ich zastúpenie nie je rovnomerné. Napríklad klasický úder na rytmičák sa vyskytuje v celom datasete 1510-krát pričom *flam* úder len 11-krát. Výraznejší rozdiel je pri počtoch aktivít neutrálnych tried nástrojov (zo sekcie 6.2.2 vieme, že neutrálna trieda reprezentuje neaktivitu daného nástroja) voči ich hracím technikám. Vzhľadom na to, že chceme na tomto datasete trénovať CNN a vyhodnocovať našu metódu, museli sme rozdeliť údery hracích techník do trénovacej a testovacej sady. Robili sme to manuálne na úrovni celých skladieb tak, aby bola každá technika v oboch množinách prítomná. Pomer zastúpenia každej techniky v trénovacej voči testovacej sme si zvolili 7:3, čo znamená, že približne 1/3 z úderov každej techniky sa nachádzala v testovacej množine a zvyšok bol v trénovacej. V prenesenom význame to znamenalo, že 2/3 skladieb bolo v trénovacej množine a zvyšok v testovacej. Mali sme však problém s technikou SDNS, ktorá sa nachádzala len v jednej skladbe. Túto skladbu sme museli manuálne rozdeliť na 2 časti tak, aby pomer úderov v množinách tiež sedel.

Ďalší dataset, ktorý sme použili v súvislosti s touto metódou bol *SampleRadar*[11]. Obsahuje 1000 rôznych akustických a elektronických nahrávok bicích, ktoré znejú monofonicky. Nás však zaujímali nahrávky len s akustickým zvukom. Tento dataset sme využili iba na inicializáciu zopár hracích techník (konkrétny priebeh inicializácie je opísaný v sekcii 8.2.2)

8.2.2. Incializácia bicích nástrojov

Všetky techniky, ktoré sme uviedli v predošlej sekcii, sme si museli vopred inicializovať. Z datasetu *SampleRadar* sme inicializovali všetky techniky hi-hat činely, ostatné sme inicializovali z *MDB Drums*. Zo *SampleRadar* bolo pre nás užitočných len 55 nahrávok – 20 pre zavretú hi-hat, 20 pre otvorenú hi-hat a 15 pre hi-hat pedal. Každá nahrávka obsahovala len 1 monofonický úder, čo znamená, že inicializácia sa robila jednoducho. Jej postup bol

[11] https://www.musicradar.com/news/drums/1000-free-drum-samples

identický s postupom, ktorý je opísaný v sekcií 8.1.2 pre algoritmus PFNMFD len s tým rozdielom, že prvý krok zistil len jednu aktiváciu a v druhom kroku sa spravil priemer len z jedného úderu. Tretí krok inicializácie bol identický.

Inicializácia kopáku pozostávala presne z tých istých krokov, ktoré sú opísané v sekcií 8.1.2 pre algoritmus PFNMFD len s tým rozdielom, že nahrávky monofonických úderov pochádzali z *MDB Drums* datasetu.

Pri hracích technikách rytmičáku to bolo o čosi zložitejšie, keďže sme pre ne nenašli samostatné nahrávky s monofonickými údermi. Inicializáciu sme museli robiť manuálne. A to tak, že v *MDB Drums* skladbách sme našli pre každú techniku takú pasáž, v ktorej znel len jej úder a žiadny iný. Z takéhoto miesta sme potom spravili spektrogram, ktorý priamo slúžil ako inicializovaný komponent danej techniky.

Tu treba ešte podotknúť, že neutrálnu triedu každého nástroja sme inicializovali, resp. vypočítali spriemerovaním už inicializovaných spektrogramov jeho hracích techník.

8.2.3. Trénovanie CNN

Už z návrhu vieme, že sieť sme museli učiť rozlišovať okrem hracích techník aj neaktivitu nástroja. No bicie nástroje sú v skladbách viac neaktívne ako aktívne. To znamená, že keď sme učili našu sieť rozoznávať neaktivitu nástroja priradením aktivity do neutrálnej triedy, jej zastúpenie bolo v porovnaní s ostatnými hracími technikami rádovo vyššie, čo môže spôsobiť preučenie. Toto ale je riešiteľný problém aplikovaním rôznych techník na rozšírenie dát (angl. *data augmentation*). Ten sme však neaplikovali, pretože by to bolo zbytočné vzhľadom na limity našej metódy (viac v experimente 3 v sekcii 8.2.7).

Ako sme už spomínali pri opise CNN v návrhu NMFDQ metódy, sieť sme učili detegovať nie konkrétne rámce, v ktorých nastala aktivita, ale väčšie okolie. Môžeme si to predstaviť tak, že sieť určuje aktivitu hracích techník niekoľko rámcov pred a niekoľko po samotnej aktivite. Nastavenie veľkosti okolia bolo parametrizované a konkrétne hodnoty uvedieme až pri jednotlivých experimentoch.

8.2.4. Predspracovanie signálu

Už z vizualizácie návrhu našej metódy (viď Obrázok 34) vieme, že signál bol predspracovaný pomocou STFT. V tejto fáze sme sa však rozhodli trochu experimentovať. A to tak, že výsledný STFT spektrogram sme preškálovali na hodnoty z intervalu <0;1> a premietli ho na logaritmickú škálu. Robili sme to hlavne preto, lebo v tejto metóde používame CNN a vo

všeobecnosti neurónové siete lepšie pracujú s dátami, ktoré sú normalizované [57]. Chceli sme preto zistiť, či dodatočné predspracovanie celkovo pomôže našej metóde a ak áno, tak o koľko. Súčasťou našich experimentov boli 2 varianty predspracovania. Buď sme použili iba STFT, alebo sme dodatočne tento spektrogram znormalizovali a premietli tieto hodnoty do logaritmickej škály podľa prirodzeného logaritmu. Pri oboch variantách STFT sme použili *Hanningové* okienko o veľkosti 2048 a veľkosťou skoku = ¼ z veľkosti okienka, čo je 512.

8.2.5. Detekcia aktivácií

Algoritmus detekcie aktivácií je presne ten istý, aký sme použili pri predošlej metóde (viac o algoritme v sekcii 8.1.4). Rozdiel je len v tom, že pri použití NMFDQ metódy sme nastavili prahovú hodnotu zvlášť pre každý nástroj. Konkrétne hodnoty uvádzame v tabuľke nižšie.

Tabuľka 11. Nastavenie parametrov algoritmu detekcie aktivácií pre metódu NMFDQ

Nástroj	Parametre algoritmu detekcie aktivácií					
-	pre_max	post_max	pre_avg	post_avg	threshold	wait
Hi-hat činel	4	4	5	5	$0.5*\max(H_{hh})$	4
Kopák	4	4	5	5	$0.3*\max(H_{kd})$	4
Rytmičák	4	4	5	5	$0.2*\max(H_{sd})$	4

K takto nastaveným prahovým hodnotám sme dospeli empiricky. Vďaka tomu sme zistili, že zvlášť nastavené prahové hodnoty prospievajú celkovej úspešnosti algoritmu.

8.2.6. Metriky a tolerancia

Na vyhodnotenie rôznych experimentov sme taktiež používali metriky *Precision*, *Recall* a *F-measure* ako pri predošlej metóde.

Tak ako pri predošlej metóde, aj v tomto prípade tolerujeme určité vychýlenie predikovaného času aktivity hracej techniky od jej skutočnej udalosti. Pri predošlej metóde sme pracovali s toleranciou 50 milisekúnd, no v tomto prípade sme ju o trochu sprísnili. Pracovali sme s hodnotou 40 milisekúnd.

8.2.7. Výsledky experimentov

V tejto časti predstavíme 3 experimenty.

1. Experiment - Nájdenie najlepšieho CNN modelu a porovnanie s baseline CNN

Na to, aby sme mohli očakávať od našej NMFDQ metódy, že sa bude správať aspoň trochu rozumne, v prvom rade sme museli natrénovať CNN. A to tak, že sme spravili tzv. *hyperparamtuning*, čo znamená, že sme hľadali najlepšie nastavenie takých parametrov, ktoré značným spôsobom ovplyvňujú úspešnosť predikcií CNN. Parametre boli nasledovné

- **konvolučný kontext** – skúšali sme hodnoty 5, 7, 9
- **veľkosť mini-batch** – skúšali sme hodnoty 8, 16, 32
- **optimalizačný algoritmus** – skúšali sme *RMSProp, Adam, SGD, Adagrad*
- **rýchlosť učenia** – skúšali sme hodnoty 0.01, 0.02, 0.001, 0.005, 0.0001 s tzv. *early-stopping rate* a pravidlom, že ak chybová funkcia neklesla po 10-tich za sebou nasledujúcich epochách, učenie sme zastavili
- **počet rámcov pred aktivitou –** skúšali sme hodnoty 1, 2, 3
- **počet rámcov po aktivite** – skúšali sme hodnoty 2, 3, 4

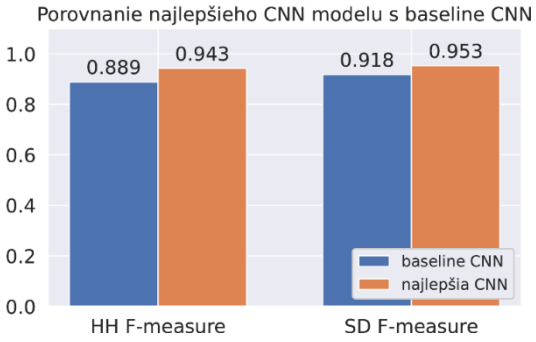
Prvé 4 parametre sú nám asi jasnejšie, no posledné 2 bližšie objasním. V sekcií 8.2.3 sme hovorili o tom, že CNN sme učili klasifikovať okolie, v ktorom nastal úder. No a na to je potrebné určiť, koľko rámcov pred a koľko rámcov po reálnej aktivite má CNN klasifikovať, že v nich pretrváva úder.

Pri návrhu NMFDQ metódy sme spomínali, že na vstupe CNN bude očakávať STFT spektrogram s veľkosťou 1025 frekvenčných rozsahov, 50 rámcov a počtom kanálov rovný 1. S týmito parametrami sme nehýbali. Z vyššie uvedených možností nastavenia CNN mal najlepší model nasledovné hodnoty parametrov

- **konvolučný kontext** = 7
- **veľkosť mini-batch** = 8
- **optimalizačný algoritmus** = *RMSProp*
- **rýchlosť učenia** = 0.001
- **počet rámcov pred aktivitou =** 1
- **počet rámcov po aktivite** = 3

Keď sme našli najlepšie nastavenie parametrov pre CNN, chceli sme zistiť, či sa vôbec naučila rozlišovať hracie techniky, keďže vieme, že zastúpenie neutrálnej triedy pre každý nástroj je veľmi veľké a nevyvážené oproti ostatným triedam a takáto skutočnosť môže ovplyvňovať výsledné metriky. Túto kontrolu sme spravili tak, že sme vytvorili *baseline*

CNN, ktorá všade predikovala neutrálne triedy a ktorá slúžila ako odrazový mostík pri porovnávaní toho, či sa natrénovaná CNN vôbec niečo naučila. Keď sme si dali najlepšiu CNN a *baseline* do porovnania na úrovni nástrojov, výsledok bol nasledovný

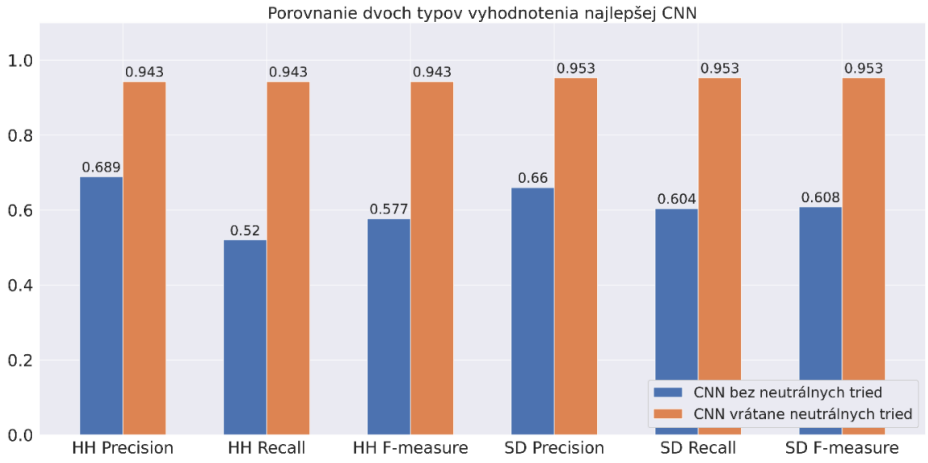


Obrázok 41. Porovnanie najlepšieho CNN modelu s baseline CNN

Z obrázka môžeme vyčítať že nami nájdená najlepšia CNN sa predsa naučila rozlišovať rôzne hracie techniky. No to, ktoré sú pre ňu menej a ktoré viac problémové, si ukážeme v ďalšom experimente.

2. Experiment – Kvantifikovanie rozlišovacej schopnosti CNN

Keď sme po predošlom experimente zistili, že naša CNN sa niečo naučila, chceli sme túto mieru kvantifikovať. Na to sme pri vyhodnotení museli zámerne zanedbať predikcie neutrálnych tried, aby sme vo výsledných metrikách zohľadňovali korektnosť klasifikácií len reálnych hracích techník. Výsledky tohto vyhodnotenia sme zobrazili spolu s výsledkami z predošlého experimentu najlepšej CNN do jednej vizualizácie.

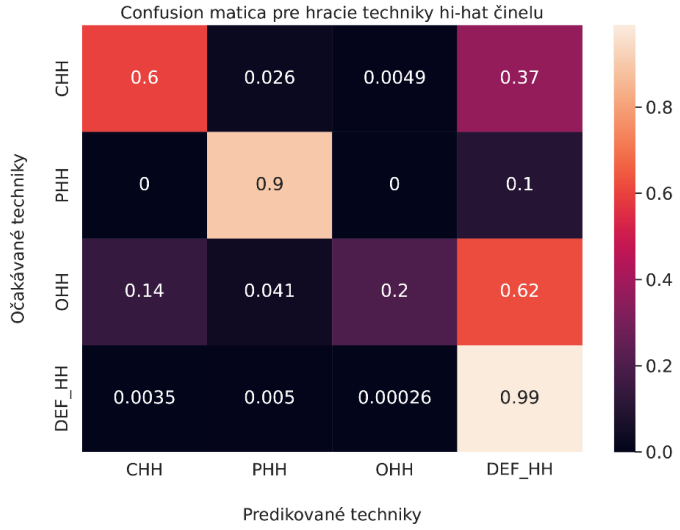


Obrázok 42. Porovnanie dvoch typov vyhodnotení

Z obrázka 42 môžeme vyčítať, že reálna rozlišovacia schopnosť hracích techník nie je až taká vysoká. Úspešnosť do veľkej miery ovplyvňujú neutrálne triedy, ktoré sa

pravdepodobne CNN naučila rozlišovať dobre, keďže ich zastúpenie spomedzi ostatných tried je najväčšie.

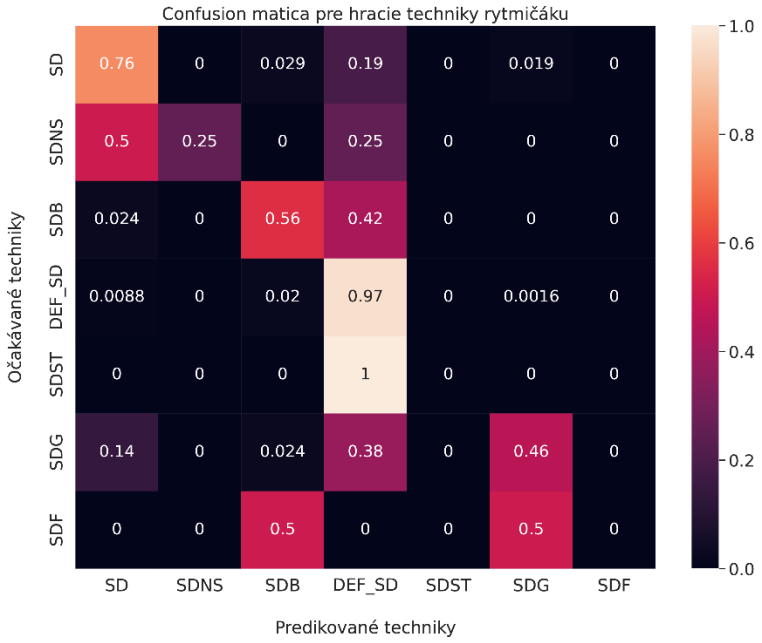
Toto vyhodnotenie kvantifikovalo rozlišovaciu schopnosť CNN na úrovni nástrojov. My sme chceli ísť ďalej a pozreli sme sa na to, ako dobre CNN predikovala jednotlivé hracie techniky a tentokrát aj vrátane neutrálnych tried. Na to sme použili tzv. *confusion matrix*, ktorá všetky potrebné informácie zaradí do mriežky a pri vizualizácií sú v nej informácie čitateľné a veľmi prehľadné. My sme vytvorili 2 matice, jednu pre techniky hi-hat činely a druhú pre rytmičák.



Obrázok 43. Confusion matica pre hi-hat činel

Na obrázka 43 môžeme vidieť, že obe osi matice nesú informáciu o hracích technikách. Keď sa pozeráme na maticu po riadkoch, vtedy môžeme vyčítať, aké hracie techniky CNN predikovala pre očakávané techniky a ako veľmi boli zastúpené. To je vyjadrené v relatívnej miere. Z matice môžeme tiež vyčítať, že najmenší problém mal model s neutrálnou triedou hi-hat činely čo je viacmennej samozrejmé (DEF_HH). Sieť nemala až taký problém s detekciu hracej techniky PHH. Môže to byť aj vďaka tomu, že jej zastúpenie je vzhľadom na veľkosť dátovej množiny pomerne veľké (523 úderov). Sieť si ju jedine mýlila s neutrálnou triedou. Čo je však zvláštne, tak s technikou CHH, ktorá je vo všeobecnosti najviac vyskytujúcou sa technikou v dátach (1847 úderov), mala CNN väčší problém ako s PHH. Opäť si ju mýlila poväčšine s neutrálnou triedou. Predpokladáme, že dôvod nižšej klasifikačnej úspešnosti pri tejto hracej technike je ten, že táto technika sa zvykne vyskytovať veľmi často v rôznych rytmoch a za prítomnosti variabilného počtu ďalších bicích nástrojov, ktoré túto techniku prehlušia. Vzhľadom na to, že sa môže vyskytovať v rozličnom kontexte, vstupy

do neurónovej siete mohli mať na miestach úderu veľkú varianciu a kvôli nie až tak dostatočne veľkému množstvu dát sa nevedela dobre naučiť jej charakteristiku. Tiež z matice môžeme vyčítať, že najväčší problém mala sieť s technikou OHH, ktorú si viac mýlila s neutrálnou triedou ako klasifikovala správne. Môže to byť kvôli jej menšiemu výskytu v dátach (269). No je zaujímavé, že úspešnosť klasifikácie je nižšia oproti CHH, lebo tento typ techniky je veľmi hlučný. Zároveň by sme očakávali, že by si ju algoritmus nemal mýliť s neutrálnou triedou až v tak veľkej miere. No vzhľadom na nerovnomernosť tried v dátach je takéto niečo možné.



Obrázok 44. Confusion matica pre rytmičák

Na obrázku 44 môžeme vidieť *confusion* maticu pre hracie techniky rytmičáku. Opäť vidíme, že algoritmus mal najmenší problém s neutrálnou triedou, čo je pochopiteľné (DEF_SD). Na druhej strane ak sa algoritmus pomýlil, zvyčajne si mýlil ľubovoľnú hraciu techniku s neutrálnou triedou. Opäť to dávame za vinu nerovnomernému zastúpeniu tried v dátach. Ďalej si môžeme všimnúť, že z hracích techník bol obyčajný úder na rytmičák (SD) pomerne veľa krát správne predikovaný. CNN sa ani raz nepodarilo úspešne zaklasifikovať techniky SDST a SDF. SDST vždy priradil neutrálnej triede a SDF si mýlil medzi SDB a SDG. Tento problém môže byť opäť kvôli tomu, že početnosť techník v dátach je veľmi malá (SDST=38, SDF=11).

Hoci nie sme úplne spokojný s tým, ako dobre sa CNN naučila rozlišovať jednotlivé hracie techniky, v kontexte NMFDQ to až tak nevadí. A to práve preto, lebo ako sme spomínali v sekcii 6.2.2, v prípade, že CNN nezdeteguje aktivitu nástroja (označenie neutrálnou

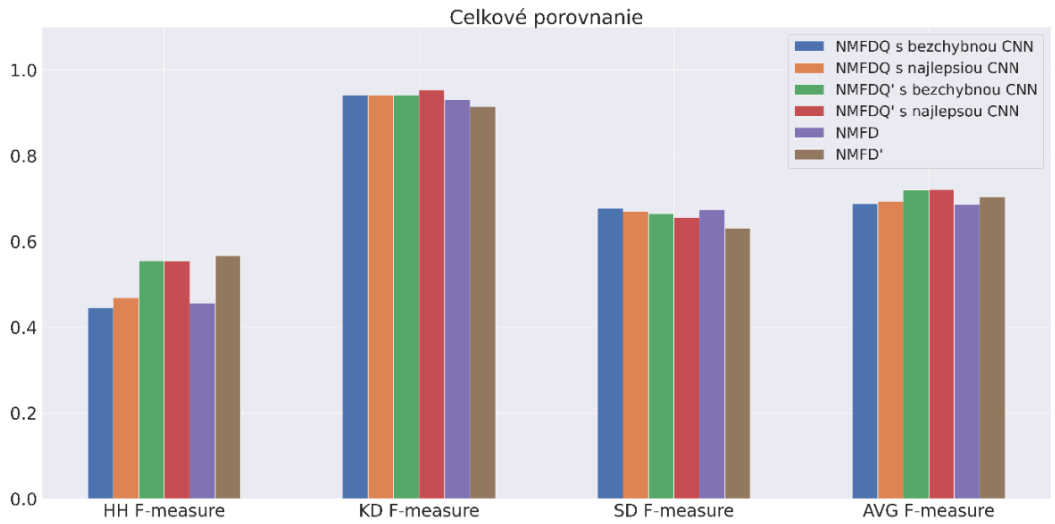
triedou), NMFD komponent môže tieto miesta "zachrániť" tým, že vysvetlí ich aktivitou neutrálnej triedy, čo môže mať pozitívny vplyv na úspešnosť transkripcie na úrovni nástrojov. V ďalšom experimente overíme, či NMFDQ metóda touto vlastnosťou disponuje.

3. Experiment – Overenie *fault-tolerance* vlastnosti metódy NMFDQ

V tomto experimente okrem primárneho cieľa, ktorým bolo overiť vlastnosť *fault-tolerance* sme chceli zistiť, či metóde NMFDQ pomôže, ak vstup znormalizujeme a logaritmujeme a či celková úspešnosť transkripcie bude signifikantne vyššia oproti klasickej NMFD metóde. Na overenie týchto výskumných podotázok bolo potrebné spraviť niekoľko krokov.

Najprv sme implementovali obyčajnú metódu NMFD a našli také nastavenie parametrov, ktoré vykazovalo najlepšie výsledky na obyčajnom vstupe a dodatočne upravenom vstupe (dodatočne upravený znamená normalizovaný a zlogaritmizovaný, ďalej budeme operovať so slovným spojením dodatočne upravený vstup). Toto sme spravili preto, aby sme vedeli našu metódu NMFDQ porovnať s najmodernejšou NMF variantou. Potom sme implementovali dve varianty našej metódy NMFDQ. V prvej sme použili najlepšie natrénovanú CNN, najlepšiu konfiguráciu NMFDQ, ktorá bola identická najlepšej NMFD a na vstupe bol buď obyčajný vstup alebo dodatočne upravený. V druhej variante sme nepoužívali CNN. Pracovali sme len s jej výstupmi, teda Q maticami. A to tak, že pri každej skladbe sme si automaticky vygenerovali Q maticu, ktorej distribúcia aktivít hracích techník naprieč časom zodpovedala 100% úspešnej, bezchybnej CNN. Všetky konfigurácie sú prehľadne zobrazené v nasledovnej vizualizácií. Chcem len podotknúť, že názov metódy s apostrofom v legende vždy znamená, že vstup do metódy bol dodatočne upravený.

Z nižšie uvedenej vizualizácie vyhodnotenia (viď Obrázok 45) môžeme vyvodiť niekoľko záverov. Prvým je ten, že dodatočná úprava vstupu výrazne zlepšila metriky pre transkripciu hi-hat činely. Taktiež tieto metódy majú vo všeobecnosti najlepšiu úspešnosť (metriku *AVG F-measure* majú najvyššiu spomedzi ostatních a zároveň približne rovnakú medzi sebou). Druhým záverom je to, že NMFDQ je vo všeobecnosti o niečo lepšia ako NMFD, no rozdiel nie je až taký výrazný – na úrovni zopár desatinných miest. Tretím záverom je to, že sme potvrdili vlastnosť *fault-tolerance* našej metódy. Z vizualizácie sme to vyčítali tak, že všetky metriky NMFDQ metódy s najlepšou CNN a bezchybnou CNN sú takmer rovnaké. Inými



Obrázok 45. Celkové porovnanie variant NMFDQ a NMFD

slovami, je jedno či naša CNN bude klasifikovať hracie techniky so 100% úspešnosťou alebo nie, na konci dňa to finálnu transkripciu na úrovni nástrojov neovplyvní. Táto vlastnosť môže do istej miery pridávať robustnosť algoritmu, pretože nie je úplne závislá od predikčnej schopnosti CNN. Na druhej strane je táto vlastnosť aj jej limitom, pretože vyššia výkonnosť CNN neovplyvní celkovú transkripciu NMFDQ metódy.

8.2.8. Zhrnutie

Pri metóde NMFDQ sme spravili 3 experimenty, ktorými sme chceli dostať odpoveď primárne na 2 otázky. Prvou bola tá, že aká bude celková úspešnosť transkripcie, ak pridáme metóde rozlišovaciu schopnosť na úrovni hracích techník. Nepodarilo sa nám ju zhoršiť a ani výrazne zlepšiť, čo sme zistili pri porovnaní už s existujúcou, najlepšie fungujúcou NMF variantou na tento problém, ktorou je NMFD. Druhou otázkou bolo, či naša metóda má nami navrhnutú vlastnosť *fault-tolerance*. Našimi experimentami sme ju potvrdili. Ako sme spomenli v závere tretieho experimentu, má to svoje pozitívum ale aj negatívum. Ak by sme sa rozhodli ďalej rozvíjať túto metódu, viac by sme sa sústredili na zlepšovanie NMFD komponentu, keďže úspešnosť CNN nevplýva na úspešnosť celkovej transkripcie NMFDQ metódy.

8.3. Prototypická CNN

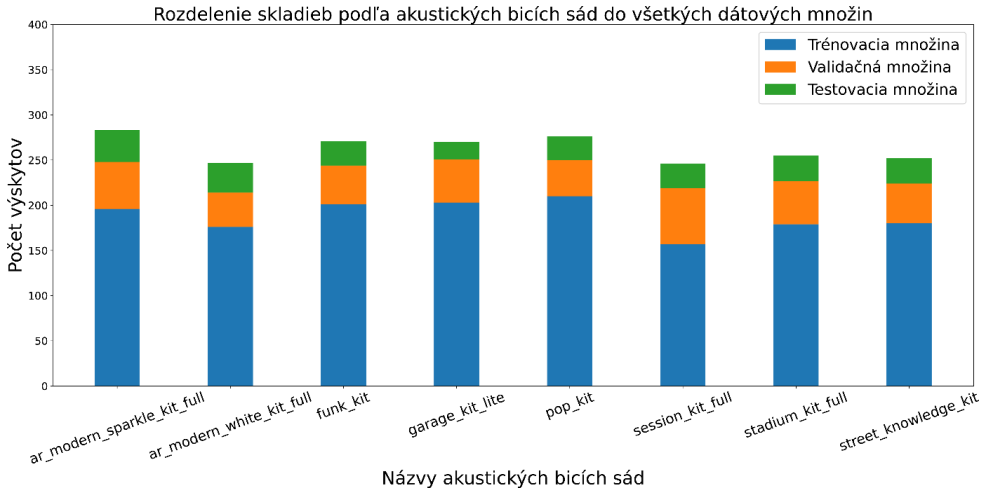
Prototypickú CNN sme vyhodnocovali na skladbách polyfonicky znejúcich bicích nástrojov bez prítomnosti harmonickej zložky. Táto časť bude mať opäť rovnakú štruktúru ako vyhodnotenia predošlých metód. Najprv si povieme niečo o datasetoch, ktoré sme použili na trénovanie a vyhodnotenie. Potom uvedieme ako sme robili predspracovanie audio signálu

a detekciu aktivácií. Následne si povieme niečo bližšie o metrikách, ktoré boli trochu odlišné od tých, ktoré sme použili pri predošlých vyhodnoteniach. Na záver uvedieme zopár experimentov, ktoré sme spravili a vyvodíme niekoľko záverov.

8.3.1. Datasety

Na trénovanie našej prototypickej CNN sme použili pomerne nedávno vytvorený (2019) dataset *Slakh2100* [58]. Obsahuje 2100 syntetizovaných skladieb s polyfonicky znejúcimi bicími nástrojmi s možnosťou pridania harmonických komponentov do pozadia, pričom všetky aktivity hudobných nástrojov v skladbách sú zaznamenané v príslušných MIDI súboroch. Každá skladba trvá približne 4 minúty čo je dokopy 145 hodín. Tento dataset bol vygenerovaný syntetizátorom, ktorý mal k dispozícii zvuky 8 rôznych akustických bicích sád. Každá bicia sada je zastúpená približne v 250 skladbách. Každá z nich obsahuje približne 25-45 rôznych tried, ktoré reprezentujú hracie techniky konkrétnych bicích nástrojov. Tieto hracie techniky boli naprieč všetkými akustickými bicími sadami poväčšine rovnaké, preto sme strávili netriviálne množstvo času nad tým, aby sme týmto triedam priradili rovnaké názvy a predišli duplicitným názvom tried v dátach.

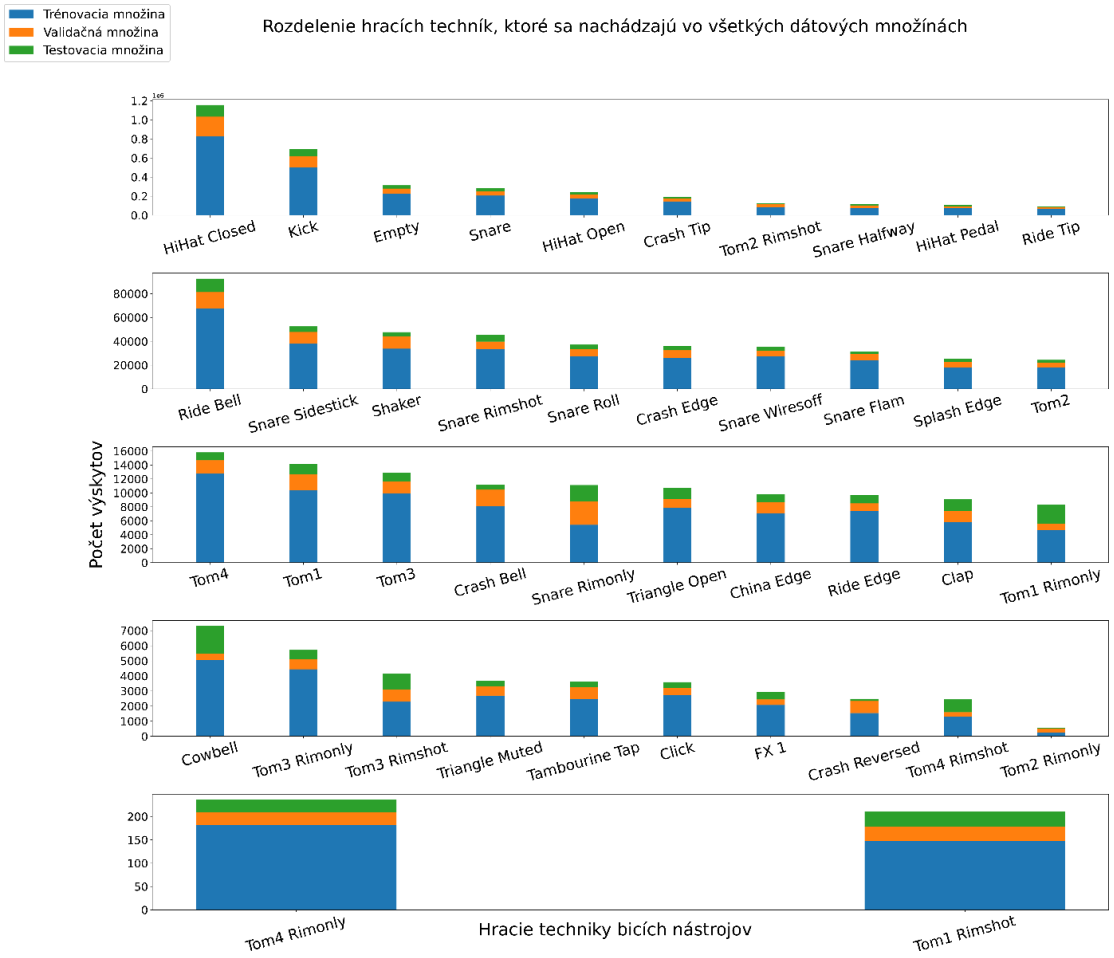
Keďže tento dataset je rádovo väčší ako v predošlých experimentoch, rozhodli sme sa spraviť detailnú dátovú analýzu. Po jeho stiahnutí z internetu bolo všetkých 2100 skladieb automaticky rozdelených do trénovacej, validačnej a testovacej množiny. Na úvod sme si chceli zistiť, ako sú skladby s jednotlivými akustickými bicími sadami distribuované naprieč všetkými množinami.



Obrázok 46. Vizualizácia analýzy rozdelenia skladieb podľa akustických sád naprieč všetkými dátovými množinami

Analýzou obrázka sme zistili, že všetky akustické bicie sady sa nachádzajú vo všetkých dátových množinách. Sú rozdelené približne v pomere 7:2:1. Ďalej sme si chceli zistiť v akom

pomere a s akou početnosťou sú zastúpené jednotlivé hracie techniky naprieč všetkými dátovými množinami. Najprv sme si zvizualizovali tie, ktoré sú zastúpené vo všetkých množinách.



Obrázok 47. Vizualizácia analýzy výskytov hracích techník naprieč dátovými množinami

Analýzou obrázka sme zistili, že veľa techník je zastúpených vo všetkých množinách tiež približne v pomere 7:2:1. No ich početnosť sa mení. Je to logické, pretože vo všeobecnosti bubeníci hrajú napríklad na kopák, rytmičák alebo zavretý hi-hat činel oveľa častejšie ako na prechody (Tom1, Tom2, Tom3), ktoré sú zväčša použité na konci slôh alebo refrénov skladieb (samozrejme závisí to od žánru hudby, ale napríklad v popovej či rockovej hudbe to platí). Ďalej sme chceli zistiť, ktoré hracie techniky sa nenachádzajú vo všetkých dátových množinách.

Analýzou obrázka sme zistili, že 4 hracie techniky s dosť nízkou početnosťou sa nachádzajú len v trénovacej množine. Kvôli tomu, aby ich model pri trénovaní mohol použiť a bol čo najrobustnejší, sme skladby s týmito technikami v tejto množine nechali.

Vzhľadom na rozumné rozdelenie tried sme sa rozhodli tento dataset kompletne použiť v prvých experimentoch (tie tu spomenieme ale nebudeme ich detailnejšie rozoberať). Na to,

Obrázok 48. Vizualizácia analýzy rozdelenia hracích techník, ktoré sa nenachádzajú vo všetkých dátových množinách

aby sme vedeli porovnať úspešnosť našich modelov aspoň čiastočne s modelom v článku [47], na vyhodnocovanie sme použili *MDB Drums* dataset, ktorý sme bližšie opísali v sekcii 8.2.1.

8.3.2. Predspracovanie signálu

Ako je zrejmé z vizualizácie na obrázku 35, skladby boli transformované do Mel spektrogramu, ktorý sme preškálovali na hodnoty z intervalu <0;1> a premietli ho na logaritmickú škálu. Robili sme to preto, lebo tak ako pri predošlej metóde NMFDQ, aj tu používame CNN a vo všeobecnosti neurónové siete lepšie pracujú s dátami, ktoré sú normalizované [57]. Pri spracovaní signálu do Mel spektrogramu sme používali *Hanningové* okienko o veľkosti 2048 a veľkosťou skoku 441, aby sme mali identické predspracovanie ako v [47].

Tu treba ešte podotknúť, že v trénovacej aj inferenčnej fáze sme potrebovali mať dáta rozdelené na malé Mel spektrogramy, ktoré boli následne vstupom do Prototypickej CNN v trénovacej aj inferenčnej fáze. Pri opise architektúry CNN v sekcii 6.3.2 sme spomínali, že časová dimenzia Mel spektrogramu (počet rámcov) bola experimentálnou záležitosťou. Experimentom, v ktorom sme skúšali rôzne veľkosti časovej dimenzie sme empiricky zistili, že ideálne bude nastaviť počet rámcov na hodnotu 25, tak ako to bolo v [47]. V tomto experimente boli použité skladby z trénovacej a validačnej množiny datasetu *Slakh2100*.

8.3.3. Detekcia aktivácií

Tak ako pri predošlých metódach, aj v tejto sme použili algoritmus na detekciu aktivácií. Tu sme spravili experiment (na vstupe boli tiež skladby zo *Slakh2100* datasetu), ktorým sme vyskúšali viacero nastavení hodnôt parametrov *pre_avg*, *post_avg*, *pre_max*, *post_max*, *wait*

a *threshold* (viac o týchto parametroch v sekcií 8.1.4). Empirickým spôsobom sme dospeli k záveru, že najlepšou kombináciou hodnôt parametrov je táto

Tabuľka 12. Nastavenie parametrov algoritmu detekcie aktivácií pre Prototypickú CNN

Parametre algoritmu detekcie aktivácií					
pre_max	post_max	pre_avg	post_avg	threshold	wait
5	5	5	5	0.12	5

Nastavenia detekcie aktivácií pri predošlých metódach (PFNMFD, NMFDQ) prahovú hodnotu (angl. *threshold*) vždy násobili s maximálnou hodnotou aktivácie pre daný nástroj. Môžeme si všimnúť, že tu to nerobíme (viď Tabuľka 12), pretože v tomto prípade hodnoty všetkých aktivácií hracích techník, ktoré sú kvôli binárnej alebo n-árnej klasifikácií Prototypickej CNN vo forme pravdepodobností, pochádzajú z rovnakej škály hodnôt - intervalu <0;1>.

8.3.4. Metriky a tolerancia

Aj pri týchto experimentoch sme primárne využívali metriky *Precision, Recall* a *F-measure*. Pri vyhodnoteniach experimentov sme však operovali iba s metrikou *F-measure*. Aby sme mali detailnejší náhľad na samotné výsledky a na konci sa vedeli porovnať s výsledkami v článku [47], rátali sme *F-measure* pre každú hraciu techniku a taktiež celkovú *macro F-measure* a *micro F-measure*. *Micro F-measure* sa rátala tak, že sa sčítali všetky TP (True Positive), FP (False Positive), FN (False Negative) všetkých hracích techník naprieč všetkými skladbami, ktoré boli vo validačnej množine a z nich sa následne vypočítala celková *F-measure*. *Macro F-measure* sa vyrátala tak, že najprv sa vypočítali *F-measure* hodnoty pre každú skladbu spriemerovaním všetkých *F-measure* hodnôt hracích techník a potom sa spravil priemer *F-measure* hodnôt zo všetkých skladieb.

Doteraz sme tento rozdiel neuvádzali, pretože sme rátali metriku *F-measure* len jedným spôsobom. Pre úplnosť, vo všetkých predošlých vyhodnoteniach, v ktorých sme uvádzali celkovú hodnotu metriky *F-measure*, sa jednalo o *macro F-measure*.

8.3.5. Výsledky experimentov

V tejto časti predstavíme 2 experimenty. Ku každému uvedieme vizualizáciu, komentáre a nakoniec vyvodíme niekoľko záverov.

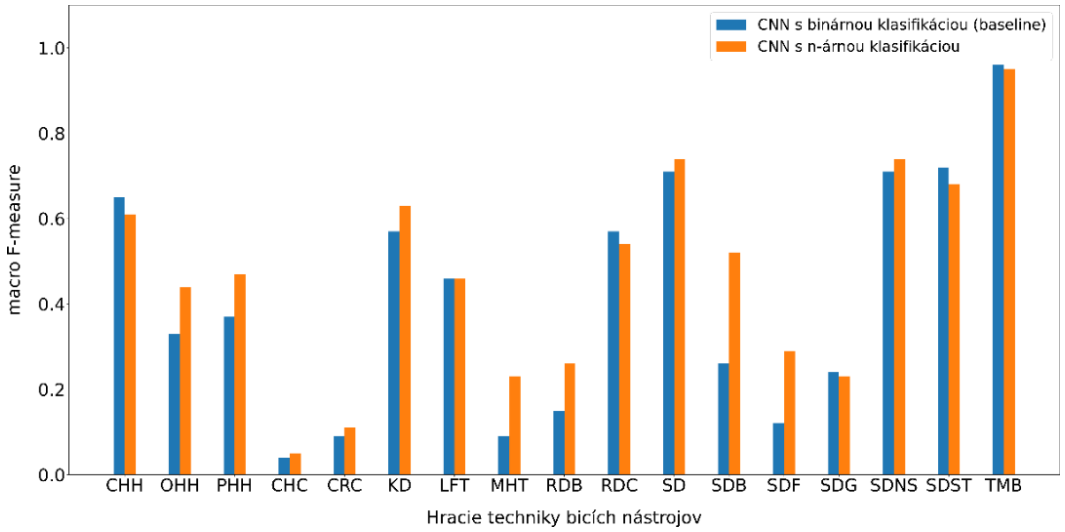
1. Experiment – Vplyv n-árnej klasifikácie na úspešnosť transkripcie bicích nástrojov

Cieľom tohto experimentu bolo zistiť, či nami navrhnuté zlepšenie opísané v sekcií 6.3.3 pomôže zvýšiť úspešnosť modelu. Predtým sme však potrebovali nájsť najlepšiu konfiguráciu CNN, ktorú sme označili za *baseline*. Najvhodnejšiu kombináciu hodnôt rôznych parametrov ovplyvňujúcich úspešnosť Prototypickej CNN sme opäť hľadali pomocou tzv. *hyperparatemer tuning* (význam tohto pojmu je opísaný v 1. experimente v časti 8.2.7). Osvedčilo sa nastavenie použité v [47]. To znamená, že sme použili *Adam* ako optimalizačnú funkciu pri spätnom šírení chyby, rýchlosť učenia bola 0.001, v každej trénovacej epizóde CNN robila 10-way 5-shot klasifikáciu (C=10, K=5) kde počet vzoriek, ktoré sa v každej epizóde dodatočne premietli do latentného priestoru a z ich vzdialeností sa rátala celková chyba, bol rovný 16 (q = 16, viac v sekcii 6.3.2 o Trénovaní). Tu treba zdôrazniť, že v každej epizóde sa 10 techník náhodne vybralo vždy z náhodne vybranej akustickej bicej sady (v *Slakh2100* datasete je 8 bicích sád). Počet epizód bol nastavený na hodnotu 20000 (toto je jediný rozdiel oproti nastaveniu parametrov modelu v [47] kde počet epizód = 100000) s *early stopping* pravidlom. Pravidlo v našom kontexte bolo nasledovné: Počas učenia siete sa každých 1000 epizód vykonal validačný test nad fixným zoznamom skladieb z validačnej množiny. V prípade, že *macro F-measure* 2-krát za sebou klesla (po 2000 trénovacích epizódach) na validačnej množine, učenie sme zastavili.

V tomto experimente našou validačnou množinou bol *MDB Drums* dataset a trénovacou *Slakh2100*. *MDB Drums* sme na vyhodnotenie použili preto, aby sme zistili, aká bude úspešnosť modelu s identickou konfiguráciou v [47] len v odlišnom kontexte (transkripcia bicích bez prítomnosti harmonických nástrojov), ako veľmi pomôže naše vylepšenie modelu a taktiež chceme porovnať naše výsledky s výsledkami v [47] hoci ich model bol aplikovaný na komplexnejší problém (transkripcia bicích za prítomnosti harmonickej zložky).

Na nižšie uvedenom obrázku (viď Obrázok 49) môžeme vidieť, že nami navrhnuté vylepšenie metódy z [47] pomohli zlepšiť metriku *macro F-measure* na úrovni techník. Konkrétne u 11-tich hracích techník sa táto metrika zvýšila. U 5-tich klesla, no rozdiel je minimálny. Jednu sa nám nepodarilo vôbec zlepšiť.

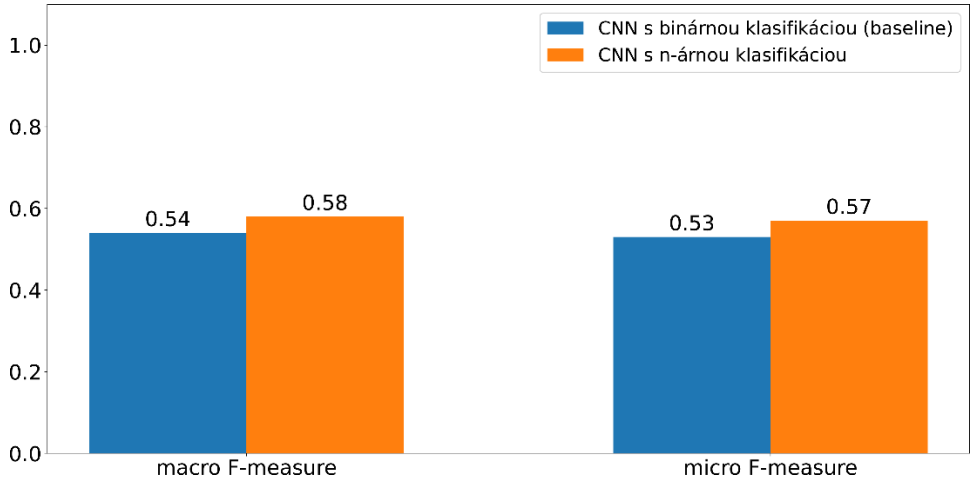
Hracie techniky SDB a SDF sa nám podarilo zlepšiť o viac ako 100%. Tu treba podotknúť, že hracia technika SDB sa vôbec nenachádzala v trénovacích dátach a nášmu vylepšeniu sa podarilo s ňou pomerne dobre vysporiadať. Na druhej strane tento model mal celkovo



Obrázok 49. Porovnanie úspešnosti prototypickej CNN bez nášho vylepšenia a s našim vylepšením (n-árnou klasifikáciou) na úrovni hracích techník

problém klasifikovať viaceré hracie techniky - CHC, CRC, MHT, RDB, SDF a SDG - hoci ich zastúpenie v trénovacej množine je pomerne veľké (viď Obrázok 47).

Ďalej sme sa pozreli na to, ako naše vylepšenie ovplyvnilo celkovú úspešnosť modelu v porovnaní s CNN, ktorá robí binárnu klasifikáciu.

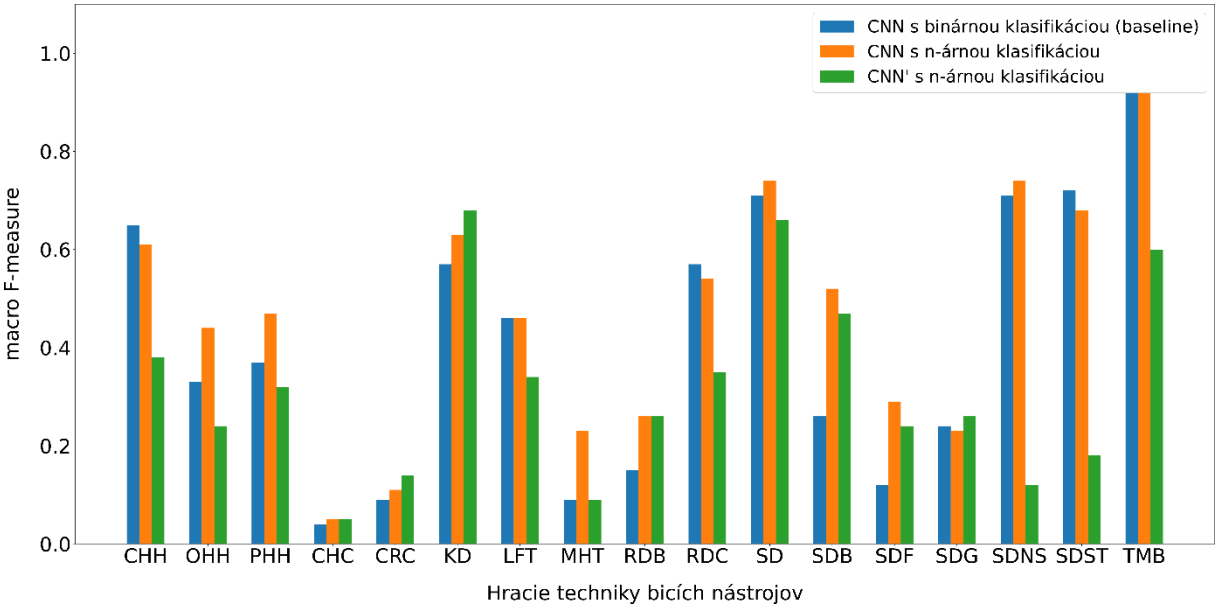


Obrázok 50. Porovnanie celkovej úspešnosti prototypickej CNN bez nášho vylepšenia a s našim vylepšením (n-árnou klasifikáciou)

Keďže sme videli zlepšenie na úrovni hracích techník, dalo sa očakávať, že sa zlepšia metriky celkovej úspešnosti modelu. No rozdiel nie je až taký signifikantný (o 0.04, viď Obrázok 50), keďže výrazné zlepšenia nastali len pri zopár hracích technikách

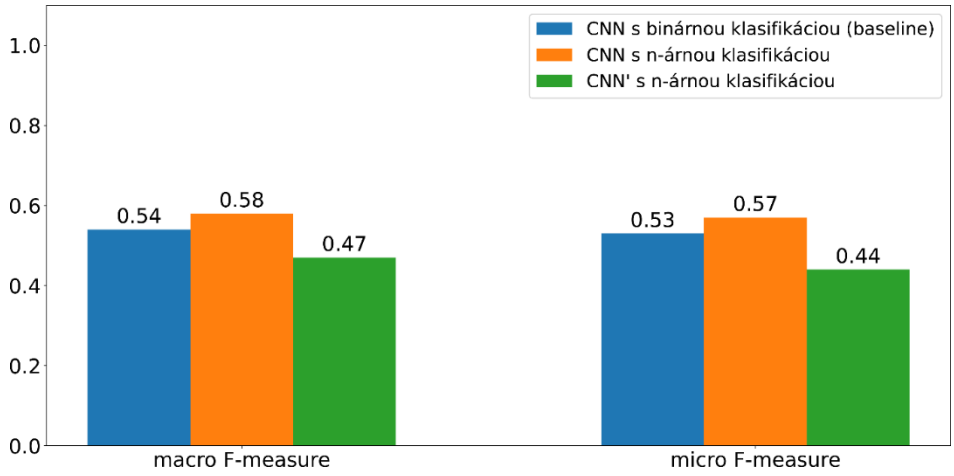
2. Experiment – Vplyv harmonických nástrojov v skladbách na diskriminatívny priestor a úspešnosť transkripcie bicích nástrojov

Cieľom tohto experimentu je zistiť, či trénovacie dáta s harmonickou zložkou v pozadí zvýšia diskriminatívnosť priestoru prototypickej CNN do takej miery, aby pri transkripcií bicích nástrojov bez harmonických nástrojov v pozadí mala sieť menší problém správne klasifikovať hracie techniky, čo by napokon malo mať vplyv na úspešnosť modelu. Prototypickú CNN sme trénovali na *Slakh2100* datasete, pričom v skladbách zneli aj harmonické nástroje. Testovanie prebiehalo opäť na *MDB Drums* datasete, kde harmonické komponenty neboli prítomné. Na tento experiment sme použili presne tie isté nastavenia parametrov CNN ako pri prvom experimente aj spolu s našim vylepšením.



Obrázok 51. Úspešnosť rozpoznávania hracích techník - porovnanie modelu natrénovaného na skladbách, v ktorých sú prítomne harmonické nástroje s modelmi natrénovaných na skladbách bez harmonickej zložky (modely z predošlého experimentu)

Na obrázku 51 môžeme vidieť tú istú vizualizáciu porovnania úspešnosti modelov na úrovni hracích techník ako na obrázku 49 len s tým rozdielom, že v tejto sú výsledky 3 modelov. Tretí, ktorý má v názve apostrof (zelená farba) reprezentuje model natrénovaný na skladbách s harmonickými nástrojmi v pozadí. Môžeme vidieť, že úspešnosť rozpoznávania hracích techník vo väčšine prípadov klesla oproti pôvodným modelom z experimentu 1. Pozreli sme sa aj na to, ako veľmi sa znížila celková úspešnosť modelu.



Obrázok 52. Celková úspešnosť - porovnanie modelu natrénovaného na skladbách, v ktorých sú prítomne harmonické nástroje s modelmi natrénovanými na skladbách bez harmonickej zložky (modely z predošlého experimentu)

Na obrázku 52 môžeme vidieť, že celková úspešnosť modelu natrénovaného na skladbách, v ktorých okrem bicích nástrojov znejú aj harmonické nástroje, klesla oproti najlepšiemu modelu z predošlého experimentu o viac ako jednu desatinu pri oboch metrikách. Z toho usudzujeme, že prítomnosť harmonických nástrojov v skladbách trénovacej množiny neovplyvnia diskriminatívny priestor modelu takým spôsobom, aby mu umožnil ľahšie rozoznávať hracie techniky bicích nástrojov v skladbách, v ktorých znejú len bicie.

Keď sme naše výsledky všetkých 3 konfigurácií porovnali s vyhodnotením najlepšieho modelu v [47], zistili sme, že sme neprekonali ani jednu z *F-measure* metrík na datasete *MDB Drums* ale minimálne sme dosiahli rovnakú úroveň presnosti aj s našou úpravou. Ich najlepší model mal m*acro F-measure = 0.6* a *micro F-measure = 0.58*.

8.3.6. Zhrnutie

Pri vyhodnocovaní prototypickej CNN sme spravili 2 experimenty. Prvým sme chceli zistiť, ako ovplyvní úspešnosť modelu použitie n-árnej klasifikácia namiesto binárnej. Ukázalo sa, že trochu zlepší výkonnosť modelu. Okrem toho si myslíme, že použitie prototypickej CNN by malo v kontexte transkripcie bicích nástrojov spolu s hracími technikami používať n-árnu klasifikáciu vždy, lebo ako sme už spomínali, v realite bubeníci nehrajú naraz viacerými technikami na ten istý nástroj. V druhom experimente sme zistili, že natrénovaním CNN na skladbách, v ktorých znejú okrem bicích aj harmonické nástroje sme možno spestrili jej diskriminatívny priestor no nie takým spôsobom, aby sme jej uľahčili klasifikáciu hracích techník v skladbách, kde znejú len bicie. Preto úspešnosť takéhoto modelu výrazne klesla. Na konci sme zistili, že hoci náš problém, ktorý sme riešili týmto prístupom, je podobný a v

zásade o niečo jednoduchší ako v [47] (my neberieme do úvahy harmonické nástroje v pozadí), nepodarilo sa nám prekonať úspešnosť ich modelu.

9. Záver

V tejto práci sme zanalyzovali problematiku automatickej transkripcie bicích z polyfonickej hudby. Navrhli sme 3 metódy – PFNMFD, NMFDQ a vylepšenie prototypickej CNN. Vyskúšali sme ich rôzne konfigurácie, ktoré sme otestovali a vyhodnotili. Z vyhodnotenia sme nadobudli niekoľko poznatkov, ktoré môžeme zohľadniť a zapracovať pri ďalšej práci.

Pri PFNMFD môžeme skúsiť nastaviť menšie hodnoty parametra riedkosti – obzvlášť pri použití PG optimalizácie. Taktiež by sme mohli skúsiť nastaviť parameter riedkosti zvlášť pre hi-hat činel, keďže na ňu mal špeciálne negatívny vplyv. Vo všeobecnosti vidíme zmysel v použití metódy PG na optimalizáciu parametrov matíc a mohli by sme ju skúsiť skombinovať s vylepšeniami ako adaptívna rýchlosť učenia či momentum. Aplikovanie dekonvolutívnej NMF má podľa nás vo všeobecnosti zmysel, pretože ako sa ukázalo, lepšie modeluje hudobné nástroje (obzvlášt hi-hat činel) ako NMF varianty, ktoré prenechávajú modelovanie úderu bicích nástrojov na aktivačné matice. Tiež vidíme zmysel vyskúšať rôzne nastavenia parametra *t* v algoritme PFNMFD, ktorý určuje dĺžku spektrálno-časového vývoja komponentov.

Pri vyhodnotení NMFDQ metódy sme si uvedomili, že *fault-tolerance* vlastnosť je jej výhodou a zároveň limitom. Výhodou je to, že miera úspešnosti klasifikácie hracích techník pomocou CNN nemá vplyv na celkovú úspešnosť metódy, pretože komponent NMFDQ vie chybné miesta “vysvetliť“ pomocou neutrálnych tried. Priestor na zlepšenie preto nevidíme v komponente CNN ale NMFDQ. A to napríklad tak, že počas faktorizácie by sme maticu komponentov nenechali zafixovanú, ale upravovala by sa polo-adaptívnym spôsobom (využíva sa v SANMF, viac v sekcií 5.2.3).

Priestor na zlepšenie metódy prototypickej CNN vidíme napríklad v spôsobe, akým sa v inferenčnej fáze počíta prototyp celej skladby. Momentálne sa ráta spriemerovaním latentných reprezentácií všetkých jej výsekov. To však môže mať negatívny vplyv na samotnú transkripciu, keďže je veľmi veľa miest (veľa preto, lebo úspešnosť modelu je vo všeobecnosti nízka), v ktorých mali byť aktívne hracie techniky, no model ich zaklasifikoval ako neaktívne, resp. latentná reprezentácia daného výseku skladby bola bližšie k prototypu celej skladby ako k prototypu hracej techniky.

Literatúra

[1] E. Benetos, S. Dixon, Z. Duan a S. Ewert, „Automatic Music Transcription: An Overview,“ IEEE Signal Processing Magazine, zv. 36, %1. vyd.1, pp. 20 - 30, 2018.

[2] A. Droppová, Elementárna hudobná teória, Prešov, 1998

[3] C.-W. Wu, C. Dittmar, C. Southall, R. Vogl, G. Widmer, J. Hockman, M. Müller a A. Lerch, „A Review of Automatic Drum Transcription,“ *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, zv. 26, %1. vyd.9, pp. 1457 - 1483, 2018.

[4] Hudobné nástroje :: Hudobná náuka,“ [Online]. Available: https://zus-nauka.webnode.sk/hudobne-nastroje/. [Cit. 14 Máj 2019].

[5] „Parts of a Drum Kit - Beginner Drums,“ [Online]. Available: https://beginnerdrums.uk/how-to-play-drums/parts-of-a-drum-kit/. [Cit. 14 Máj 2019].

[6] „Bicia súprava - BUBENÍK,“ 27 December 2007. [Online]. Available: https://bubenik.napady.net/rubriky/bicia-suprava. [Cit. 14 Máj 2019].

[7] „Mechanické vlnenie,“ [Online]. Available: http://web.tuke.sk/feikf/sk/files/F1_Vlny.pdf. [Cit. 3 March 2021].

[8] „How the Ear Works,“ [Online]. Available: https://www.hopkinsmedicine.org/health/conditions-and-diseases/how-the-ear-works#:~:text=It%20collects%20sound%20waves%20and,cause%20the%20eardrum%20to%20vibrate.. [Cit. 3 March 2021]

[9] „Sampling rate (audio) - Glossary,“ Federal Agencies Digital Guidelines Initiative, [Online]. Available: http://www.digitizationguidelines.gov/term.php?term=samplingrateaudio. [Cit. 18 March 2021].

[10] A. Klapuri a M. Davy, rev. *Signal Processing Methods for Music Transcription*, Springer US, 2006, pp. 21 – 25

[11] J. D. Cook, „Gibbs phenomenon,“ ScienceDirect, 10 June 2020. [Online]. Available: https://www.johndcook.com/blog/2020/06/10/gibbs-phenomenon/. [Cit. 20 March 2021].

[12] I. Goodfellow, Y. Bengio a A. Courville, Deep Learning, MIT Press, 2016.

[13] „History of Neural Networks,“ [Online]. Available: http://www.psych.utoronto.ca/users/reingold/courses/ai/cache/neural4.html. [Cit. 15 Máj 2019].

[14] N. S. Gill, „Artificial Neural Networks and Neural Networks Applications - Xenonstack,“ Xenonstack, 1 Marec 2019. [Online]. Available: https://www.xenonstack.com/blog/artificial-neural-network-applications/. [Cit. 15 Máj 2019].

[15] J. Bryden, „Biologically Inspired Computing: The Neural Network,“ Leeds.

[16] P. Lucidarme, „Simplest perceptron update rules demonstration • Robotics, Teaching & Learning,“ 6 December 2017. [Online]. Available:

https://www.lucidarme.me/simplest-perceptron-update-rules-demonstration/. [Cit. 15 Máj 2019].

[17] „Common Neural Network Activation Functions | Rubik's Code," 20 November 2017. [Online]. Available: https://rubikscode.net/2017/11/20/common-neural-network-activation-functions/. [Cit. 15 Máj 2019].

[18] „The Gradient and Directional Derivative," [Online]. Available: https://math.oregonstate.edu/home/programs/undergrad/CalculusQuestStudyGuides/vcalc/grad/grad.html. [Cit. 15 Máj 2019].

[19] J. Jordan, „Setting the learning rate of your neural network.," 1 Marec 2018. [Online]. Available: https://www.jeremyjordan.me/nn-learning-rate/. [Cit. 16 Máj 2019].

[20] D. Schmidt, „Understanding Nesterov Momentum (NAG)," 23 November 2018. [Online]. Available: https://dominikschmidt.xyz/nesterov-momentum/. [Cit. 16 Máj 2019].

[21] J. Brownlee, „A Gentle Introduction to Batch Normalization for Deep Neural Networks," Machine Learning Mastery, 4 December 2019. [Online]. Available: https://machinelearningmastery.com/batch-normalization-for-training-of-deep-neural-networks/. [Cit. 25 March 2021].

[22] S. Ruder, „An overview of gradient descent optimization algorithms," 19 Január 2016. [Online]. Available: http://ruder.io/optimizing-gradient-descent/. [Cit. 16 Máj 2019].

[23] A. Bhande, „What is underfitting and overfitting in machine learning and how to deal with it.," 11 Marec 2018. [Online]. Available: https://medium.com/greyatom/what-is-underfitting-and-overfitting-in-machine-learning-and-how-to-deal-with-it-6803a989c76. [Cit. 16 Máj 2019].

[24] S. Singh, „Understanding the Bias-Variance Tradeoff," Towards Data science, 21 May 2018. [Online]. Available: https://towardsdatascience.com/understanding-the-bias-variance-tradeoff-165e6942b229. [Cit. 29 April 2021].

[25] R. Khandelwal, „L1 L2 Regularization – Data Driven Investor – Medium," 4 November 2018. [Online]. Available: https://medium.com/datadriveninvestor/l1-l2-regularization-7f1b4fe948f2. [Cit. 16 Máj 2019].

[26] I. Farkaš, „Neural Networks," Knižničné a edičné centrum FMFI UK, Bratislava, 2011.

[27] C. Southall, R. Stables a J. Hockman, „Automatic Drum Transcription for Polyphonic Recordings Using Soft Attention Mechanisms and Convolutional Neural Networks," rev. *18th International Society for Music Information Retrieval Conference,* Suzhou, 2017.

[28] C. Southall, R. Stables a J. Hockman, „Automatic Drum Transcription Using Bi-Directional Recurrent Neural Networks," rev. *17th International Society for Music Information Retrieval Conference,* New York City, 2016.

[29] „Understanding LSTM Networks -- colah's blog," 27 August 2015. [Online]. Available: https://colah.github.io/posts/2015-08-Understanding-LSTMs/. [Cit. 17 Máj 2019].

[30] R. Vogl, M. Dorfer, G. Widmer a P. Knees, „Drum Transcription via Joint Beat and Drum Modeling Using Convolutional Recurrent Neural Networks," rev. *18th International Society for Music Information Retrieval Conference*, Suzhou, 2017.

[31] J. Zafra, „AlexNet the revolution ImageNet challenge 2012," Medium, 13 June 2020. [Online]. Available: https://medium.com/@947_34258/alexnet-the-revolution-imagenet-challenge-2012-48a9a4a6b3ef. [Cit. 31 March 2021].

[32] S. Saha, „A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way," Towards Data Science, 15 December 2018. [Online]. Available: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53. [Cit. 31 March 2021].

[33] S. W. Smith, The Scientis and Engineer's Guide to Digital Signal Processing, California: California Technical Publishing, 1999.

[34] S. Hochreiter, „The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions," *International Journal of Uncertainty Fuzziness and Knowledge-Based Systems,* pp. 107-116, 1998.

[35] Z. ZY., „ Nonnegative Matrix Factorization: Models, Algorithms and Applications," rev. *Data Mining: Foundations and Intelligent Paradigms. Intelligent Systems Reference Library, vol 24. ,* Berlin, Heidelberg, Springer, 2012, pp. 99-134.

[36] D. D. Lee and H. S. Seung, „Algorithms for non-negative matrix fac- torization," rev. *Neural Inf. Process. Syst.*, Denver, CO, USA, 2000.

[37] C.-W. Wu a A. Lerch, „Drum Transcription using Partially Fixed Non-Negative Matrix Factorization with Template Adaptation," rev. *16th International Society for Music Information Retrieval Conference*, Malaga, 2015.

[38] J. J. Burred., „Detailed derivation of multiplicative update," March 2014. [Online]. Available: https://www.jjburred.com/research/pdf/jjburred_nmf_updates.pdf. [Cit. 5 December 2019].

[39] N. He, „Lecture 10: Lower bounds & Projected Gradient Descent," 22 September 2016. [Online]. Available: http://niaohe.ise.illinois.edu/IE598_2016/pdf/IE598-lecture10-projected%20gradient%20descent.pdf. [Cit. 10 December 2019].

[40] S. Lau, „Learning Rate Schedules and Adaptive Learning Rate Methods for Deep Learning," 29 July 2017. [Online]. Available: https://towardsdatascience.com/learning-rate-schedules-and-adaptive-learning-rate-methods-for-deep-learning-2c8f433990d1. [Cit. 6 December 2019].

[41] P. O. Hoyer, „Non-negative matrix factorization with sparseness constraints.," *Journal of Machine Learning Research,* %1. vyd.5, p. 1457–1469, 2004.

[42] C.-W. Wu a A. Lerch, „Drum transcription using partially fixed non-negative matrix factorization," rev. *23rd European Signal Processing Conference,* Nice, 2015.

[43] P. Smaragdis, „Non negative matrix factor deconvolution, extraction of multiple sound sources from monophonic inputs," *International Congress on Independent Component Analysis and Blind Signal Separation,* pp. 494-499, 2004.

[44] R. Vogl, M. Dorfer a P. Knees, „Drum transcription from polyphonic music with recurrent neural networks," rev. *IEEE International Conference on Acoustics, Speech and Signal Processing*, New Orleans, 2017.

[45] C. Jacques a A. Roebel, „Automatic drum transcription with convolutional neural networks," rev. *21th International Conference on Digital Audio Effects*, Aveiro, 2018.

[46] J. Patel, „Learning To Differentiate using Deep Metric Learning," Towards Data Science, 4 October 2020. [Online]. Available: https://towardsdatascience.com/deep-metric-learning-76fa0a5a415f. [Cit. 2 April 2021].

[47] W. Yu, S. Justin, C. Mark, J. B. Nicholas a P. B. Juan, „Few-shot drum transcription in polyphonic music," rev. *21st International Society for Music Information Retrieval Conference*, Montréal, 2020.

[48] X. Wanqi a W. Wei, *One-Shot Image Classification by Learning to Restore Prototypes*, 2020.

[49] S. Ilia a S. Matthias, *Less Than One-Shot Learning: Learning N Classes From M<N Samples*, 2020.

[50] J. Snell, K. Swersky a R. S. Zemel, „Prototypical Networks for Few-shot Learning," rev. *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, Long Beach, 2017.

[51] C. Dittmar a D. Gärtner, „Real-time transcription and separation of drum recordings based on NMF decomposition," rev. *17th Internation Conference on Digital Audio Effects*, Erlangen, 2014.

[52] H. Lindsay-Smith a S. McDonald, „Drumkit transcription via convolutive NMF," rev. *Proc. of the 15th Int. Conference on Digital Audio Effects (DAFx-12)*, York, 2012.

[53] O. Gillet a G. Richard, „ENST-Drums: an extensive audio-visual database for drum signals processing," rev. *ISMIR'06*, Victoria, Canada,, 2006.

[54] A. Rosebrock, „Keras: Multiple outputs and multiple losses," Pyimagesearch, 4 June 2018. [Online]. Available: https://www.pyimagesearch.com/2018/06/04/keras-multiple-outputs-and-multiple-losses/. [Cit. 13 April 2021].

[55] K. P. Shung, „Accuracy, Precision, Recall or F1?," Towards Data Science, 15 March 2018. [Online]. Available: https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9. [Cit. 9 December 2019].

[56] C. Southall, C.-W. Wu, A. Lerch a J. Hockman, „MDB Drums - An Annotated Subset of MedleyDB for Automatic Drum Transcription," rev. *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR)*, 2017.

[57] T. Stottner, „Why Data should be Normalized before Training a Neural Network," Towards Data Science, 16 May 2019. [Online]. Available: https://towardsdatascience.com/why-data-should-be-normalized-before-training-a-neural-network-c626b7f66c7d. [Cit. 15 April 2021].

[58] E. Manilow, G. Wichern, P. Seetharaman a J. L. Roux, „Cutting Music Source Separation Some Slakh: A Dataset to Study the Impact of Training Data Quality and

Quantity,“ *IEEE Workshop on Appli- cations of Signal Processing to Audio and Acoustics (WASPAA)*, pp. 45-49, 2019.