VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

SEPARACE NÁSTROJŮ A ZPĚVU Z HUDEBNÍ NAHRÁVKY

MUSIC SOURCE SEPARATION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **VILIAM HOLÍK**
AUTHOR

VEDOUCÍ PRÁCE **Ing. LADISLAV MOŠNER**
SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání bakalářské práce

147883

Ústav:	Ústav počítačové grafiky a multimédií (UPGM)
Student:	**Holík Viliam**
Program:	Informační technologie
Specializace:	Informační technologie
Název:	**Separace nástrojů a zpěvu z hudební nahrávky**
Kategorie:	Zpracování signálů
Akademický rok:	2022/23

Zadání:

1. Seznamte se s úlohou separace zdrojů ze zvukové nahrávky a se současnými modely pro separaci založenými na strojovém učení (zejména Demucs, Conv-TasNet).
2. Seznamte se s některou z knihoven pro trénování neuronových sítí (např. PyTorch, Tensorflow).
3. Natrénujte model Conv-TasNet na datové sadě MUSDB18 s využitím existující implementace a proveďte objektivní vyhodnocení.
4. Navrhněte a implementujte modifikace neuronové sítě a/nebo trénování za účelem lepšího porozumění modelu a potenciálního zlepšení.
5. Experimentujte s navrženými modifikacemi, proveďte vyhodnocení experimentálních výsledků a porovnejte je s výsledky v relevantní literatuře.

Literatura:

- DÉFOSSEZ, Alexandre, et al. Music source separation in the waveform domain. *arXiv preprint arXiv:1911.13254*, 2019.
- LUO, Yi; MESGARANI, Nima. Conv-tasnet: Surpassing ideal time-frequency magnitude masking for speech separation. *IEEE/ACM transactions on audio, speech, and language processing*, 2019, 27.8: 1256-1266.

Při obhajobě semestrální části projektu je požadováno:
Body 1, 2, rozpracovaný bod 3

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce:	**Mošner Ladislav, Ing.**
Vedoucí ústavu:	Černocký Jan, prof. Dr. Ing.
Datum zadání:	1.11.2022
Termín pro odevzdání:	10.5.2023
Datum schválení:	31.10.2022

Abstrakt

Na separáciu zdrojov z hudobných nahrávok sa používajú neurónové siete. Jednou z takýchto sietí je Conv-TasNet. Cieľom práce je experimentovať s už existujúcou implementáciou tejto siete za účelom potenciálneho zlepšenia. Trénovanie modelov prebiehalo na dátovej sade MUSDB18. Postupne sa experimentovalo so zmenou štruktúry siete, transformáciou signálov z časovej domény do frekvenčnej pre účely počítania objektívnej funkcie, zámenou rôznych objektívnych funkcií za pôvodnú, hľadaním optimálneho koeficientu rýchlosti učenia pre každú objektívnu funkciu a jeho postupným zmenšovaním v priebehu učenia. Ako najlepšie experimenty podla metriky SDR vyšli trénovania s objektívnymi funkciami L1 a logaritmickou L2 v časovej doméne pri vyššom počiatočnom koeficiente rýchlosti učenia s jeho postupným zmenšovaním v priebehu učenia. V relatívnom porovnaní najlepších modelov oproti východziemu ide o viac ako 2,5% zlepšenie.

Abstract

Neural networks are used for the problem of music source separation from recordings. One such network is Conv-TasNet. The aim of the work is to experiment with the already existing implementation of this network for the purpose of potential improvement. The models were trained on the MUSDB18 dataset. It was successively experimented with the change of the network structure, transforming signals from the time domain to the frequency domain for the purpose of calculating the loss function, replacing different loss functions with the original one, finding the optimal learning rate for each loss function and gradually decreasing the learning rate during the learning process. The best experiments according to the SDR metric were training with loss functions L1 and logarithmic L2 in the time domain with a higher initial learning rate with its gradual decrease during the learning process. In a relative comparison of the best models to the baseline, it is more than 2.5% improvement.

Kľúčové slová

separácia zdrojov hudby, neurónové siete, objektívne funkcie, L1, L2, logaritmická L1, logaritmická L2, SI-SDR, SDR, SIR, SAR, STFT, rýchlosť učenia, frekvenčná doména, časová doména, Conv-TasNet, MUSDB18

Keywords

music source separation, neural networks, loss functions, L1, L2, logarithmic L1, logarithmic L2, SI-SDR, SDR, SIR, SAR, STFT, learning rate, frequency domain, time domain, Conv-TasNet, MUSDB18

Citácia

HOLÍK, Viliam. *Separace nástrojů a zpěvu z hudební nahrávky.* Brno, 2023. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Ladislav Mošner

Separace nástrojů a zpěvu z hudební nahrávky

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána Ing. Ladislava Mošnera. Uviedol som všetky literárne pramene, publikácie a ďalšie zdroje, z ktorých som čerpal.

.....
Viliam Holík
9. mája 2023

Poďakovanie

Chcel by som sa poďakovať môjmu vedúcemu Ing. Ladislavovi Mošnerovi za jeho odbornú pomoc naprieč celou tvorbou tejto práce. Výpočtové zdroje zabezpečil projekt e-INFRA CZ (ID:90140), podporený Ministerstvom školstva, mládeže a telovýchovy Českej republiky.

Obsah

Kapitola 1

Úvod

Separácia zdrojov hudby sa vzťahuje na úlohu rozdeľovania zmiešaného zvukového signálu na jeho jednotlivé časti, ako sú spev, bicie, basa a iné nástroje. Táto úloha má množstvo aplikácií v hudobnej produkcii, remixovaní, prepise textu a analýze. Techniky hlbokého učenia, ako sú neurónové siete, ukázali významný potenciál na separáciu hudobných zdrojov. Tieto techniky sú schopné naučiť sa komplexné reprezentácie vstupných údajov a manipulovať s nimi, vďaka čomu sú pre túto úlohu veľmi vhodné. Jednou z takýchto neurónových sietí je Conv-TasNet. Conv-TasNet je neurónová sieť založená na prevažne konvolučných vrstvách, používaná na oddelenie zvuku v časovej doméne.

Pre pochopenie tejto práce je potrebné poznať úlohu separácie zdrojov zo zvukovej nahrávky. Cieľom práce je zoznámiť sa s architektúrou Conv-TasNet založenou na strojovom učení, natrénovať tento model na dátovej sade MUSDB18, navrhnúť modifikácie neurónovej siete za účelom lepšieho porozumenia a potenciálneho zlepšenia, implementovať ich a experimentovať s nimi.

Modifikácie, ktoré stoja za vyskúšanie sú zmena štruktúry siete, transformácia časovej domény do frekvenčnej pre účely počítania objektívnej funkcie, zámena objektívnej funkcie za iné, nastavenie vhodného koeficientu rýchlosti učenia a postupné zmenšovanie tohto koeficientu. Presnosť a kvalita modelu sa vyhodnocuje metrikami určenými na hodnotenie zvukových signálov. Tieto modifikácie môžu smerovať k modelu s lepším celkovým skóre. Jednotlivé experimenty budú vyhodnotené a porovnané voči sebe.

Práca je rozdelená na niekoľko kapitol, ktoré postupne pokrývajú jednotlivé logické celky. Kapitola 2 popisuje úlohu separácie zdrojov. V kapitole 3 sú popísané jednotlivé typy reprezentácie zvuku. Typy neurónových sietí, siete určené na separáciu zdrojov hudby a architektúra siete Conv-TasNet sú rozobrané v kapitole 4. Kapitola 5 sa venuje trénovaniu neurónových sietí a spôsobe ich hodnotenia. V kapitole 6 sú popísané dátové sady na trénovanie sietí pre úlohu separácie zdrojov hudby. Kapitola 7 sa zaoberá návrhom modifikácií. Implementácia navrhnutých modifikácií a ich výsledky sú v kapitole 8 a objektívne vyhodnotenie spolu s diskusiou v kapitole 9.

Kapitola 2

Separácia zdrojov

Separácia zdrojov alebo aj slepá separácia zdrojov („blind source separation“) je oddelenie súboru zdrojových signálov od súboru zmiešaných signálov bez pomoci informácií (alebo s veľmi malým množstvom informácií) o zdrojových signáloch alebo procese miešania. Najčastejšie sa používa pri digitálnom spracovaní signálov a zahŕňa analýzu zmesí signálov. Jej cieľom je obnoviť pôvodné signály komponentov zo zmiešaného signálu. Klasickým príkladom problému oddeľovania zdroja je problém tzv. kokteilovej párty, kde v miestnosti súčasne hovorí niekoľko ľudí a posluchár sa snaží sledovať jeden z monológov [48]. Ľudský mozog dokáže zvládnuť tento druh problému oddeľovania sluchového zdroja, ale pre digitálny svet je to zložitý problém pri spracovaní signálu.

Separácia zdrojov je dôležitou úlohou v oblasti strojového učenia a v posledných rokoch bola široko študovaná. Všeobecný problém separácie zdrojov možno formulovať takto: vzhľadom na zmes (mix) signálov z rôznych zdrojov je cieľom oddeliť tieto jednotlivé zdroje, ktoré prispievajú k zmesi [50]. Separácia zdrojov je dôležitým problémom so širokou škálou aplikácií vrátane spracovania reči [29], spracovania zvuku [35], spracovania obrazu [55] a spracovania biomedicínskych signálov [13]. Na vyriešenie tohto problému bolo navrhnutých niekoľko prístupov, ale vývoj stále napreduje. Niektoré z úspešnejších prístupov sú analýza hlavných komponentov a analýza nezávislých komponentov, ktoré fungujú dobre, keď nie sú prítomné žiadne oneskorenia alebo ozveny. To znamená, že problém je značne zjednodušený. Ako ďalší prístup na separáciu zdrojov sa používajú rôzne algoritmy strojového učenia.

2.1 Separácia zdrojov zvuku

Separácia zdrojov zvuku je proces oddeľovania zmesi zvukových signálov do zdrojov, z ktorých pozostáva. Cieľom je extrahovať jednotlivé zdroje, ktoré sa podieľajú na zmesi, aby bolo možné každý zdroj spracovať alebo analyzovať samostatne [28]. Do kategórie separácie zdrojov zvuku patrí separácia reči a separácia zdrojov hudby. Separácia reči je špeciálny scenár problému separácie zdrojov, kde sa pozornosť sústreďuje iba na prekrývajúce sa zdroje rečového signálu a iné interferencie, ako sú hudba alebo iné šumové signály, nie sú hlavným záujmom štúdie [1].

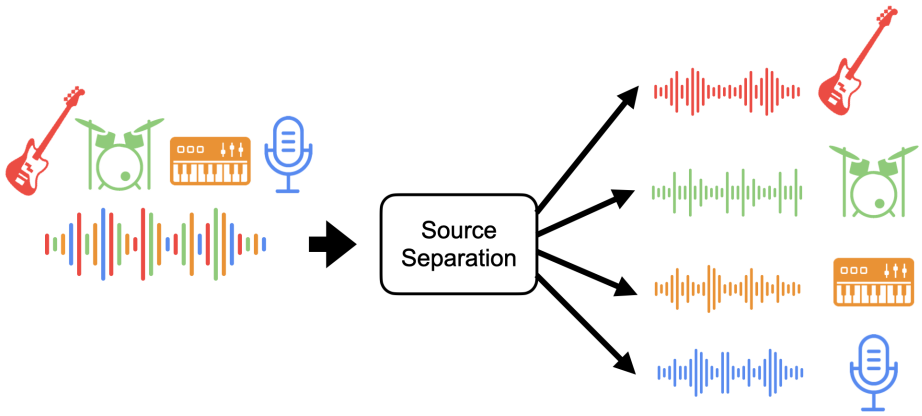
2.2 Separácia zdrojov hudby

Úlohu, pri ktorej sú separovanými zdrojmi hudobné nástroje (vrátane spevu), nazývame separácia zdrojov hudby („music source separation“ – MSS) [5]. Každý tento hudobný nástroj prípadne spev nachádzajúci sa v zmesi nazývame zdrojom. Princípom separácie hudobných zdrojov je teda dej, pri ktorom sa z celej nahrávky viacerých zdrojov oddelia tieto zdroje a tak vzniknú nahrávky len separovaných zdrojov, ako je aj zobrazené na Obrázku 2.1. Napríklad by sme mohli chcieť izolovať speváka od hudby na pozadí, aby sme vytvorili karaoke verziu piesne, alebo izolovať basgitaru od zvyšku kapely, aby sa hudobník mohol naučiť svoju časť. Inak povedané, ak je zmes viacerých zdrojov, môžu sa oddeliť len tie, ktoré nás zaujímajú.

Matematicky vyjadrené, predpokladáme že zmes signálov x_t pozostáva z K zdrojov, kde je každý zdroj $y_{t,k}$ pre $k = 1..K$:

$$x_t = \sum_{K=1}^{K} y_{t,k}.$$

Daná je len zmes x_t a cieľom systému separácie zdrojov je obnoviť jeden alebo viac zdrojov $y_{t,k}$.



Obr. 2.1: Princíp separácie zdrojov hudby [30].

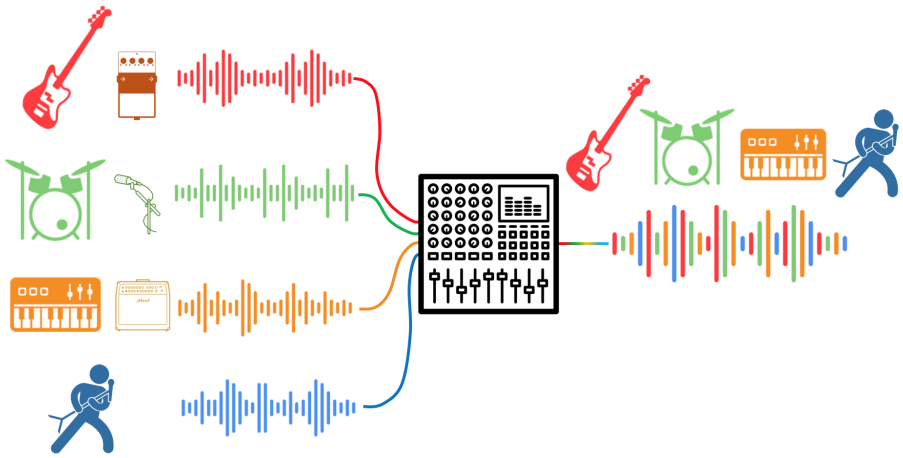
Separácia hudby je vnímaná ako odlišný problém od iných typov separácie zdrojov, pretože existuje veľa faktorov, ktoré ju robia jedinečnou [5]. Zdroje v hudbe sú vysoko korelované, čo znamená, že všetky zdroje sa zvyčajne menia spoločne v rovnakom čase a tak tvoria harmóniu. Napríklad v rockovej kapele, ak basgitara zmení tón na začiatku nového taktu, je pravdepodobné, že sa zmenia aj ostatné nástroje.

Hudba sa mixuje a spracováva spôsobmi, ktoré nie sú lineárne. Súčasné postupy zaznamenávania a nahrávania sú také, že sa daný zdroj v zmesi nemusí prirodzene vyskytovať v reálnom svete. Reverb, filtrovanie a nelineárne techniky spracovania signálu sťažujú separáciu hudby, a napriek tomu sú to nástroje, ktoré hudobníci bežne používajú na vytváranie hudby. Proces miešania hudby znázorňuje Obrázok 2.2. Miešanie hudby je problém pri následnej spätnej separácii, pretože len zriedka vieme, aké spracovanie bolo aplikované na ktorúkoľvek skupinu zdrojov alebo či už celú zmes dohromady [30].

V oblasti získavania informácií o hudbe („music information retrieval“ – MIR) existuje mnoho použití na oddelenie hudobných zdrojov. Má mnoho aplikácií, napríklad pri mixovaní alebo remixovaní nahrávok, ktorých jednotlivé zdrojové signály nie sú prístupné. Slúži tiež

na vytváranie hracích stôp pre študentov hudobných nástrojov. MSS je dôležitým krokom predspracovania pre niekoľko úloh získavania hudobných informácií, napríklad:

- automatický prepis textu [10, 39],
- detekcia hudobných nástrojov [18],
- rozpoznávanie textov [33],
- zarovnanie textu piesne podľa hudby [12],
- automatická identifikácia speváka [46].



Obr. 2.2: Miešanie hudobných signálov je zložitý a nelineárny proces. To sťažuje separáciu hudobných zdrojov [30].

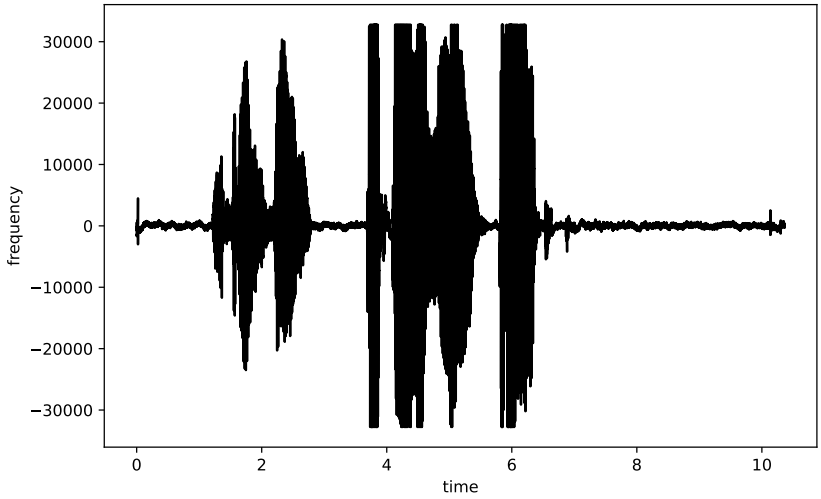
Kapitola 3

Reprezentácia zvuku

Táto kapitola bola inšpirovaná internetovou knihou *Open Source Tools & Data for Music Source Separation* [30].

Jednou z hlavných vecí, ktorými sa prístupy k separácii zvuku rozlišujú, je reprezentácia zvuku na systémovom vstupe a výstupe. Zvuk je zväčša vo svojej najneopracovanejšej forme uložený ako priebeh signálu („waveform“). Niektoré prístupy k separácii zdrojov fungujú priamo na priebehu signálu, hoci mnohé z nich vyžadujú určité predbežné spracovanie pred separáciou zdrojov. V tejto časti sa rozoberú rôzne typy vstupných a výstupných reprezentácií, ktoré sa bežne používajú v prístupoch separácie zdrojov.

3.1 Priebeh signálu

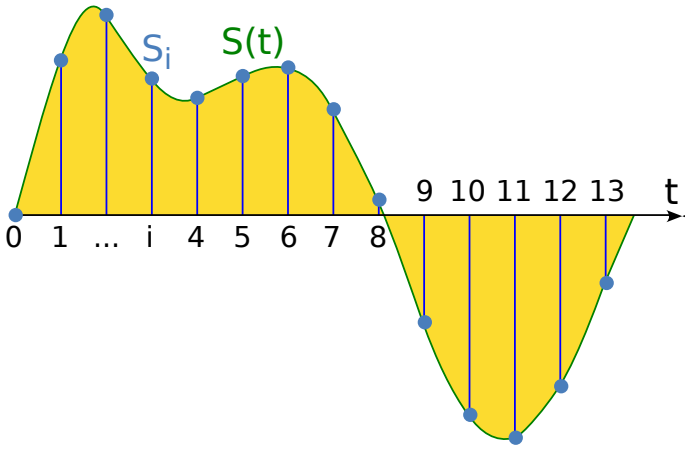


Obr. 3.1: Priebeh signálu v čase.

Priebeh signálu je okamžitá hodnota frekvencie signálu v čase (Obrázok 3.1). Okamžitá hodnota signálu je zobrazená na osi y (zvislá) a čas na osi x (vodorovná) [53]. Digitalizovaný priebeh signálu („waveform“), ďalej už len priebeh signálu, je digitalizovaný zvukový signál, ktorý sa najviac podobá reálnemu zvuku. Dôležité je vedieť, že signál so spojitým časom je

diskretizovaný (prevedený zo spojitej veličiny do diskrétnej) v čase aj amplitúde, pretože signál je pravidelne vzorkovaný a prevedený do digitálnej reprezentácie.

Dôležitým aspektom priebehu signálu je vzorkovacia frekvencia, ktorá popisuje, koľko meraní alebo vzoriek sa uskutoční za sekundu a meria sa v Hertzoch (skratka Hz). Vzorkovanie signálu možno vidieť na Obrázku 3.2. *Shannonov vzorkovací teorém* hovorí, že ideálna rekonštrukcia signálu je možná iba vtedy, keď je vzorkovacia frekvencia väčšia ako dvojnásobok maximálnej frekvencie vzorkovaného signálu. Rozsah ľudského sluchu je približne 20 Hz až 20 kHz [43]. Minimálna vzorkovacia frekvencia, ktorá spĺňa vzorkovací teorém pre túto šírku pásma je 40 kHz. Z tohto dôvodu je pre základné zaznamenávanie zvuku zvolená vzorkovacia frekvencia o trochu vyššia, a to 44,1 kHz. Táto hodnota je používaná kvôli formátu kompaktného disku (CD), ktorý sa datuje od roku 1979[1], keď ho začala používať spoločnosť *Sony*.



Obr. 3.2: Vzorkovanie signálu, diskretizácia[2].

3.2 Vlastnosti reprezentácie zvuku

Dá sa tvrdiť, že prístup oddeľovania zdroja je len taký dobrý, ako je jeho schopnosť reprezentovať zvuk oddeliteľným spôsobom. Z toho dôvodu je dôležité, ako je samotný zvuk reprezentovaný na účely oddelenia zdroja. Pre prístupy k separácii zdrojov je základný postup nasledovný:

1. Prevedenie zvuku na reprezentáciu, v ktorej sa oddeľuje.
2. Separácia zvuku v tejto reprezentácii.
3. Prevedenie zvuku späť z manipulovanej reprezentácie za účelom získania izolovaných zdrojov.

Tieto kroky sú všeobecnou postupnosťou pre prístupy k separácii zdrojov zvuku. Jednotlivé kroky môžu pozostávať z viacerých samostatných podkrokov.

[1] história CD - https://www.discwizards.com/history-of-the-cd.htm
[2] zdroj https://medium.com/analytics-vidhya/audio-data-processing-feature-extraction-science-concepts-behind-them-be97fbd587d8

Dôležitým aspektom zvukovej reprezentácie je aj schopnosť konvertovania signálu, ktorý bol nejakým spôsobom reprezentovaný, späť na priebeh signálu s čo možno najmenšou chybou. Tieto chyby predstavujú potom artefakty, ktoré je počuť v separovaných nahrávkach. V nasledujúcich sekciách sú popísané niektoré metódy, ktoré vytvárajú reprezentácie zvuku.

3.3 Časovo-frekvenčné reprezentácie

Časovo-frekvenčné reprezentácie sú najbežnejšími typmi reprezentácií používaných v prístupoch separácie zdrojov. Časovo-frekvenčná („time-frequency“ – TF) reprezentácia [49] je 2-rozmerná matica, ktorá predstavuje frekvenčný obsah zvukového signálu v priebehu času. Konkrétny záznam v tejto matici nazývame TF element („bin“). Reprezentáciu TF môžeme vizualizovať pomocou tepelnej mapy, ktorej os x je čas a os y frekvencia. Každý TF element v tepelnej mape predstavuje zastúpenie danej frekvencie v daný čas.

3.3.1 STFT

Mnohé z časovo-frekvenčných reprezentácií začínajú krátkodobou Fourierovou transformáciou („Short Time Fourier Transform“ – STFT). V praxi je postup výpočtu STFT taký, že sa dlhší časový signál rozdelí na kratšie prekrývajúce sa segmenty rovnakej dĺžky a potom sa diskrétna Fourierova transformácia (DFT) vypočíta samostatne pre každý kratší segment. Absolútna hodnota elementu TF $|X(t, f)|$ v čase t a frekvencia f určuje množstvo energie počutej z frekvencie f v čase t.

Dôležité je, že každý element v STFT je komplexný, čo znamená, že každý záznam obsahuje zložku magnitúdy aj fázy. Elementy STFT sú invertovateľné, čo znamená, že komplexnú maticu STFT možno previesť späť na priebeh signálu. Spätnou transformáciou je inverzná krátkodobá Fourierova transformácia ISTFT. Medzi základné parametre pri STFT patrí: oknová funkcia („window“), dĺžka okna a dĺžka skoku („hop“).

Oknová funkcia určuje tvar krátkodobého okna, ktorým sa jednotlivé segmenty vynásobia pred použitím diskrétnej Fourierovej tranformácie. Tvar tohto okna ovplyvňuje, ktoré hodnoty vzoriek budú v DFT zdôraznené alebo zoslabené. Existuje mnoho typov okien, ale medzi najpoužívanejšie patria obdĺžnikové, Hammingove, Kaiserove alebo Hannove okná. Dĺžka okna určuje, koľko vzoriek je zahrnutých v každom okne. Dĺžka skoku určuje vzdialenosť vo vzorkách medzi akýmikoľvek dvoma susednými oknami.

3.3.2 Spektogramy

Niektoré prístupy k separácii zdrojov zvuku fungujú iba na nejakom variante spektrogramu, ktorý explicitne nepredstavuje fázu v každom TF elemente. Medzi tieto spektogramy patrí spektogam magnitúd, spektogram výkonu, logaritmický spektogram a logaritmický spektogram výkonu.

Spektrogram magnitúd sa vypočíta tak, že sa zoberie absolútna hodnota každého prvku v STFT. Spektrogram výkonu sa vypočíta tak, že sa na každý prvok v STFT aplikuje druhá mocnina. Logaritmický spektrogram sa vypočíta ako logaritmus absolútnej hodnoty každého prvku v STFT. Logaritmický spektrogram výkonu sa vypočíta s použitím logaritmu druhej mocniny každého prvku v STFT.

Kapitola 4

Neurónové siete

Neurónová sieť je výpočtový model využívaný v oblasti strojového učenia, zostavený na základe abstrakcie vlastností biologických nervových sústav. Neurónové siete poskytujú celý rad výkonných techník na riešenie problémov pri rozpoznávaní vzorov, analýze dát a kontrole [16]. Základnou vlastnosťou neurónových sietí je schopnosť abstrakcie pravidiel medzi vstupnými a výstupnými hodnotami prezentovanými vo vhodnej forme. Táto abstrakcia pravidiel sa potom aplikuje na akékoľvek vstupné hodnoty.

Neurónové siete sú založené na súbore spojených jednotiek alebo uzlov nazývaných umelé neuróny, ktoré voľne modelujú neuróny v biologickom mozgu. Každé spojenie, podobne ako synapsie v biologickom mozgu, môže prenášať signál do iných neurónov. Umelý neurón prijíma signály, potom ich spracuje a následne ich môže poslať ďalej neurónom, s ktorými je spojený. Spojenia neurónov sa nazývajú hrany.

Proces učenia je hľadanie vhodných váh neurónových spojení, aby neurónová sieť vedela vhodnou abstrakciou pravidiel riešiť daný problém v čo-možno najlepšej forme a môže prebiehať viacerými spôsobmi bližšie uvedenými v sekcii 5.1. Počas tohto procesu sa aktualizujú hodnoty váh spojení neurónov. Po ukončení učenia (natrénovaní siete) sa už hodnoty váh nemenia a sieť produkuje výstupy podľa naučenej funkcie aplikovanej na vstupné hodnoty.

Typicky sú neuróny agregované do vrstiev. Vrstva je základným stavebným blokom najvyššej úrovne neurónovej siete. Vrstva predstavuje tzv. kontajner, ktorý zvyčajne prijíma vážený vstup, transformuje ho množinou zväčša nelineárnych funkcií a potom tieto hodnoty odovzdáva ako svoj výstup ďalšej vrstve. Rôzne vrstvy môžu na svojich vstupoch vykonávať rôzne transformácie. Signály prechádzajú postupne z prvej vrstvy (vstupná vrstva) až do poslednej vrstvy (výstupná vrstva) a medzi nimi sa často nachádzajú aj skryté vrstvy.

4.1 Typy neurónových sietí

Existuje niekoľko typov neurónových sietí, ktoré sú k dispozícii. Môžu byť klasifikované v závislosti na ich štruktúre, dátovom toku, použitých neurónov a ich hustoty, vrstiev atď. Každý typ má svoje výhody a nevýhody. Ďalej budú popísané niektoré typy sietí.

Perceptrón

Model perceptrónu navrhnutý Frankom Rosenblattom je jedným z najjednoduchších a najstarších modelov nerónu [44]. Je to najmenšia jednotka neurónovej siete, ktorá vykonáva určité výpočty na detekciu vzťahov vo vstupných dátach. Perceptrón je algoritmom na ria-

dené učenie binárnych klasifikátorov. Binárny klasifikátor je funkcia, ktorá môže rozhodnúť, či vstup reprezentovaný vektorom čísel patrí do nejakej špecifickej triedy alebo nie [11].

Perceptróny môžu implementovať logické hradlá ako AND, OR alebo NAND. Perceptróny sa však môžu naučiť riešiť iba lineárne separovateľné problémy, ale pre nelineárne problémy napr. booleovský problém XOR to nefunguje.

Dopredná neurónová sieť

Dopredná („feed forward“) neurónová sieť je známa svojou jednoduchosťou štruktúry. Dopredná neurónová sieť má vstupnú vrstvu, skryté vrstvy a výstupnú vrstvu. Informácie vždy prechádzajú jedným smerom – od vstupnej vrstvy po výstupnú vrstvu – a nikdy sa nevracajú dozadu. Aplikáciou dopredných neurónových sietí je jednoduchá klasifikácia, rozpoznávanie tváre (jednoduché priame spracovanie obrazu) a rozpoznávanie reči. Medzi výhody týchto sietí patrí že sú menej zložité, jednoduché na návrh a ich rýchlosť (jednosmerné šírenie).

Viacvrstvový perceptrón

Viacvrstvový perceptrón je vstupným bodom ku komplexným neurónovým sieťam, kde vstupné dáta prechádzajú rôznymi vrstvami neurónov. Každý jeden uzol je spojený so všetkými neurónmi v ďalšej vrstve, čo z neho robí plne prepojenú neurónovú sieť. V tejto sieti sú prítomné vstupná vrstva, skryté vrstvy a výstupná vrstva, t. j. celkovo aspoň tri alebo viac vrstiev.

Výhodou je, že sa môžu používať na hlboké strojové učenie a to vďaka prítomnosti hustých plne prepojených vrstiev a spätného šírenia. Ako nevýhodou týchto sietí je pomerne zložitý proces návrhu.

Rekurentná neurónová sieť

Rekurentná neurónová sieť (RNN) je rozšírenie doprednej neurónovej siete, kde spojenia medzi uzlami môžu vytvárať cyklus, umožňujúci výstupom z niektorých uzlov ovplyvniť následný vstup do rovnakých uzlov [32]. To jej umožňuje prejavovať dočasné dynamické správanie. RNN môže využiť svoj vnútorný stav (pamäť) na spracovanie sekvencií vstupov s premenlivou dĺžkou.

Použiteľné sú na úlohy, ako je rozpoznávanie rukopisu, kontrolu gramatiky alebo rozpoznávanie reči. Jednou z výhod použitia je modelovanie sekvenčných údajov, kde sa dá predpokladať, že každá vzorka závisí od predošlých (historických) údajov. Ako nevýhoda sú problémy s miznúcim („vanishing“) a explodujúcim gradientom [42].

LSTM neuronová sieť

Siete LSTM („Long Short-Term Memory“) sú typom RNN, ktorý okrem štandardných jednotiek používa aj špeciálne jednotky. Jednotky LSTM obsahujú „pamäťovú bunku“, ktorá dokáže uchovávať informácie v pamäti po dlhú dobu. Sada brán sa používa na kontrolu, kedy informácie vstupujú do pamäte, kedy sú výstupom, a kedy sú zabudnuté. Existujú tri typy brán, a to vstupná brána, výstupná brána a zabudnutá („forget“) brána. Vstupná brána rozhoduje o tom, ktoré časti nových informácií sa uložia v aktuálnom stave. Výstupné brány riadia, ktoré informácie v aktuálnom stave majú byť na výstupe, berúc do úvahy predchádzajúci a aktuálny stav. Zabudnuté brány rozhodujú o tom, aké informácie

sa majú z predchádzajúceho stavu vyradiť, teda riadia rýchlosť trhania („tearing rate“) uloženej pamäte. Táto architektúra im umožňuje naučiť sa dlhodobejšie závislosti.

4.1.1 Konvolučné neurónové siete

Konvolučná neurónová sieť („convolutional neural network“ – CNN) je typom neurónovej siete, ktorá sa najčastejšie používa na analýzu vizuálnych snímok. CNN používa matematickú operáciu nazývanú konvolúcia namiesto všeobecného násobenia matice aspoň v jednej zo svojich vrstiev [14]. CNN je špeciálne navrhnutá predovšetkým na spracovanie pixelových údajov a používa sa pri rozpoznávaní a spracovaní obrazu. Má aplikácie v rozpoznávaní obrázkov a videa, klasifikácii obrázkov, segmentácii obrázkov, spracovaní prirodzeného jazyka [8] atď.

Namiesto toho, aby sa CNN pokúšala spracovať celý vstup naraz, rozdeľuje ho na menšie, jednoduchšie prvky pomocou filtrov a spracúva ich postupne. Tieto filtre (jadrá) sa aplikujú na rôzne (resp. postupne všetky) časti vstupu na extrahovanie relevantných informácií. Znamená to, že CNN spracováva dáta hierarchicky. Tento hierarchický prístup umožňuje CNN efektívne sa učiť zložité komplexné vzory zo vstupných dát. Typická štruktúra CNN obsahuje alebo môže obsahovať tieto špecifické vrstvy:

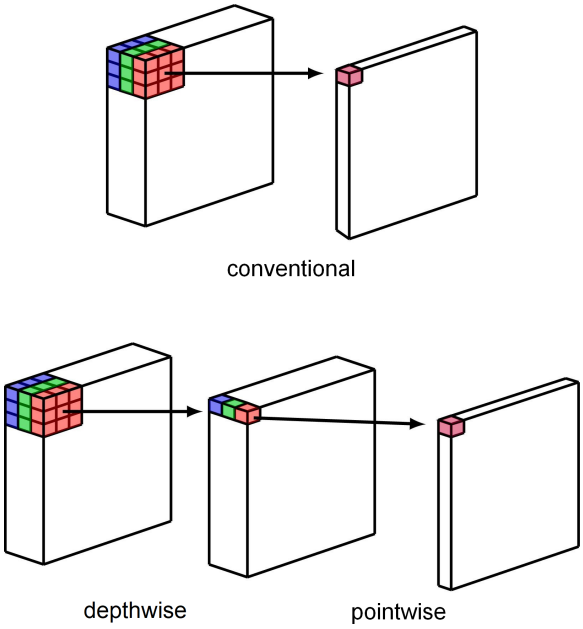
- Konvolučná
- Združovacia
- Plne prepojená

Konvolučná vrstva

Táto vrstva je prvou vrstvou, ktorá sa používa na extrahovanie rôznych vzorov zo vstupov. V tejto vrstve sa vykonáva matematická operácia konvolúcie medzi časťami vstupu a filtrom (jadrom). Hodnoty filtra sú trénovateľné parametre. Konvolučná vrstva vytvára bodový súčin konvolučného filtra s časťami vstupu posúvaním filtra po vstupnej matici.

Konvolučné vrstvy konvolujú vstup a odovzdajú jeho výsledok ďalšej vrstve. Každý neurón v konvolučnej vrstve spracováva údaje len pre svoje receptívne pole. Výstup je tzv. mapa prvkov („feature map“), ktorá poskytuje abstrakciu informácií o vstupe a zachytáva zložitejšie vzory vstupov. Následne sa táto mapa prvkov dodáva do ďalších vrstiev, aby sa naučili niekoľko ďalších funkcií vstupu.

Na urýchlenie spracovania môžu byť štandardné konvolučné vrstvy nahradené hĺbkovo oddeliteľnými („depthwise“) konvolučnými vrstvami [7], ktoré sú založené na hĺbkovej konvolúcii, po ktorých nasleduje bodová („pointwise“) konvolúcia ako vidieť na Obrázku 4.1. Hĺbková konvolúcia je priestorová konvolúcia aplikovaná nezávisle na každom kanáli vstupného tenzora, zatiaľ čo bodová konvolúcia je štandardná konvolúcia obmedzená na použitie 1×1 jadra.



Obr. 4.1: Štandardnú konvolučnú vrstvu môžu nahradiť po sebe idúce depthwise a pointwise konvolučné vrstvy[2].

Združovacia vrstva

Vo väčšine prípadov po konvolučnej vrstve nasleduje združovacia („pooling“) vrstva, tiež známa ako „downsampling“. Primárnym cieľom tejto vrstvy je zmenšiť veľkosť mapy prvkov (po prevedení konvolúcie), aby sa znížili výpočtové náklady. Toto sa vykonáva agregáciou hodnôt v istom okolí a nezávisle funguje na každej mape prvkov. Táto vrstva v podstate sumarizuje vlastnosti generované konvolučnou vrstvou.

Existuje niekoľko typov operácií združovania, ale v bežnom používaní sú dva typy združovania, a to maximálne a priemerné. Maximálne združovanie používa maximálnu hodnotu prvkov každého lokálneho zhluku na mape prvkov, zatiaľ čo priemerné združovanie vypočíta priemernú hodnotu prvkov v zhluku.

Plne prepojená vrstva

Plne prepojená vrstva („fully connected“ – FC) pozostáva z váh a „bias“-ov a používa sa na prepojenie neurónov medzi dvoma rôznymi vrstvami. Táto vrstva je zvyčajne umiestnená pred výstupnou vrstvou a tvorí niekoľko posledných vrstiev architektúry CNN.

Táto vrstva plní úlohu klasifikácie na základe vlastností extrahovaných cez predchádzajúce vrstvy a ich rôznych filtrov. Zatiaľ čo konvolučné a združovacie vrstvy majú tendenciu používať aktivačné funkcie ReLU, vrstvy FC zvyčajne využívajú funkciu softmax na vhodnú klasifikáciu vstupov, čím sa vytvára pravdepodobnosť od 0 do 1. V tejto vrstve teda zvyčajne prebieha konečný klasifikačný proces.

[2]Inšpirované https://www.sciencedirect.com/science/article/pii/S0168169918318696

4.2 Neurónové siete na separáciu zdrojov hudby

Pre úlohu separácie zdrojov hudby existuje viacero modelov. V Tabuľke 4.1 sú uvedené niektoré z nich spolu s ich SDR celkovým skóre, kde bola evaluácia vykonaná na dátovej sade MUSDB18. Vyhodnocovacia metrika SDR je bližšie popísaná v sekcii 5.5. Vyššie skóre SDR predstavuje lepší model. Všetky tieto modely boli trénované na dátovej sade MUSDB18 (viac v sekcii 6.2) bez ďalších extra dát.

Tabuľka 4.1: Prehľad modelov neurónových sietí na separáciu zdrojov hudby zoradených od najnižšieho SDR skóre.

Názov	**Rok**	**Celkové SDR [dB]**
Wave-U-Net	2018	3,23
Open-Unmix	2019	5,33
Sams-Net	2019	5,65
Conv-TasNet	2018	5,73
LaSAFT	2020	5,88
D3Net	2021	6,01
Demucs	2019	6,28
Hybrid Demucs	2021	7,68
Band-Split RNN	2022	8,24

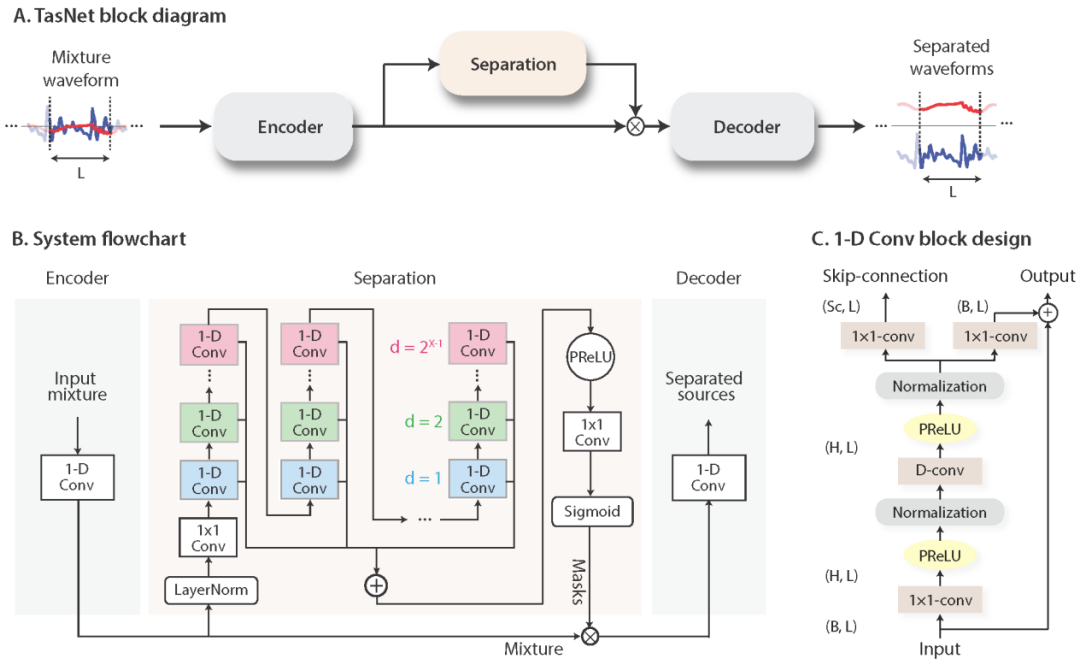
Všetky tieto modely sú prístupné z webovej stránky *Papers with Code*[3], kde sú ku každému modelu podrobné výsledky a odkaz na článok, ktorý ich popisuje.

4.3 Conv-TasNet

Pre túto prácu bola využitá neurónová sieť Conv-TasNet. Pôvodná sieť bola navrhnutá na separáciu reči, ale následne bola adaptovaná na problém separácie zdrojov hudby, a to konkrétne podporou viacerých zvukových kanálov, čo v hudbe môže predstavovať jej zdroje (jednotlivé nástroje a spev).

Plne konvolučná sieť na oddelenie zvuku v časovej doméne (Conv-TasNet) pozostáva z troch fáz spracovania, ako je znázornené aj na Obrázku 4.2(A): kodér, separácia a dekodér. Najprv sa modul kodéra použije na transformáciu krátkych segmentov priebehu signálu zo zmesi pomocou báz, ktoré sa upravujú spoločne s ostatnými váhami v priebehu trénovania. Táto 2D reprezentácia sa potom použije na odhad multiplikatívnej funkcie (masky) pre každý zdroj pri každom časovom kroku. Zdrojové priebehy signálu sa potom zrekonštruujú transformáciou maskovaných reprezentácií kodéra pomocou modulu dekodéra. Popis tejto siete vychádza z článku [27].

[3] https://paperswithcode.com/sota/music-source-separation-on-musdb18



Obr. 4.2: (A): Bloková schéma systému Conv-TasNet. (B): Vývojový diagram systému. (C): Štruktúra 1-D konvolučného bloku. [27]

Konvolučný kodér-dekodér

Vstupnú zvukovú zmes možno rozdeliť na prekrývajúce sa segmenty dĺžky L, reprezentované $\boldsymbol{x}_k \in \mathbb{R}^{1\times L}$, kde $k = 1, ..., \hat{T}$ označuje index segmentu a $\hat{T}$ celkový počet segmentov na vstupe. $\boldsymbol{x}_k$ sa transformuje na N-rozmernú reprezentáciu, $\boldsymbol{w}_k \in \mathbb{R}^{1\times N}$ pomocou operácie 1-D konvolúcia, ktorá je preformulovaná ako maticové násobenie (index k sa odteraz vypustí):

$$\boldsymbol{w} = \mathcal{H}(\boldsymbol{x}\boldsymbol{U}),$$

kde $\boldsymbol{U} \in \mathbb{R}^{L\times N}$ obsahuje N vektorov (bázových funkcií kodéra) s dĺžkou L a $\mathcal{H}(\cdot)$ je voliteľná nelineárna funkcia. Dekodér rekonštruuje priebeh z tejto reprezentácie pomocou 1-D transponovanej konvolučnej operácie, ktorá môže byť preformulovaná ako ďalšie násobenie matice:

$$\hat{\boldsymbol{x}} = \boldsymbol{w}\boldsymbol{V},$$

kde $\hat{\boldsymbol{x}} \in \mathbb{R}^{1\times L}$ je rekonštrukcia $\boldsymbol{x}$ a riadky v $\boldsymbol{V} \in \mathbb{R}^{N\times L}$ sú bázové funkcie dekodéra, každý s dĺžkou L. Prekrývajúce sa rekonštruované segmenty sú spolu sčítané na vytvorenie konečného priebehu signálu.

Aj keď preformulujeme operácie kodéra/dekodéra ako maticové násobenie, termín „konvolučný autokóder“ sa používa preto, že pri skutočnej implementácii modelu môžu konvolučné a konvolučné transponované vrstvy ľahšie zvládnuť presah medzi segmentami, a tým dokážu urýchliť trénovanie a dosiahnuť lepšiu konvergenciu.

Odhad separačných masiek

Separácia pre každý rámec sa vykonáva odhadom C vektorov (masiek) $\boldsymbol{m}_i \in \mathbb{R}^{1\times N}, i = 1, ..., C$, kde C je počet zdrojov v zmesi, ktoré sa vynásobia s výstupom kodéra $\boldsymbol{w}$. Reprezentácia každého zdroja $\boldsymbol{d}_i \in \mathbb{R}^{1\times N}$ sa potom vypočíta použitím zodpovedajúcej masky $\boldsymbol{m}_i$ na reprezentáciu zmesi $\boldsymbol{w}$:

$$\boldsymbol{d}_i = \boldsymbol{w} \odot \boldsymbol{m}_i,$$

kde $\odot$ označuje násobenie po jednotlivých prvkoch. Priebeh signálu každého zdroja $\hat{\boldsymbol{s}}_i, i = 1, ..., C$ potom rekonštruuje dekodér:

$$\hat{\boldsymbol{s}}_{\boldsymbol{i}} = \boldsymbol{d}_i \boldsymbol{V}.$$

Konvolučný separačný modul

Motivovaný časovou konvolučnou sieťou (TCN) [25], plne konvolučný separačný modul pozostáva z na seba naskladaných 1-D dilatovaných konvolučných blokov, ako je znázornené na Obrázku 4.2(B). TCN bola navrhnutá ako náhrada za RNN (rekurentné neurónové siete) v rôznych úlohách modelovania sekvencií. Každá vrstva v TCN pozostáva z 1-D konvolučných blokov s rastúcimi dilatačnými faktormi. Dilatačné faktory rastú exponenciálne, aby sa zabezpečilo dostatočne veľké časové kontextové okno na využitie dlhodobých závislosti zvukového signálu, ako je označené rôznymi farbami na Obrázku 4.2(B). V sieti Conv-TasNet sa M konvolučné bloky s dilatačnými faktormi $1, 2, 4, ..., 2^{M-1}$ opakujú R-krát. Vstup pre každý jeden blok je zodpovedajúcim spôsobom vyplnený nulou, aby bola výstupná dĺžka rovnaká ako vstupná. Výstup TCN je odovzdaný do konvolučného bloku s veľkosťou jadra 1 (1×1-*conv* blok, tiež známy ako bodová („pointwise“) konvolúcia) na odhad masky. 1×1-*conv* blok spolu s nelineárnou aktivačnou funkciou odhaduje vektory C masky pre C cieľové zdroje.

Obrázok 4.2(C) zobrazuje štruktúru každého 1-D konvolučného bloku. Návrh 1-D konvolučných blokov pochádza z článku [36], kde je aplikovaný reziduálny spoj a preskakujúci spoj („skip-connection“). Reziduálny spoj bloku slúži ako vstup do ďalšieho bloku a preskakujúce spoje sú pre všetky bloky zhrnuté dokopy a použité ako výstup TCN. Na ďalšie zníženie počtu parametrov sa používa hĺbkovo oddeliteľná („depthwise separable“) konvolúcia S-*conv*$(\cdot)$ na nahradenie štandardnej konvolúcie v každom konvolučnom bloku. Hĺbkovo oddeliteľná konvolúcia (označovaná aj iba ako oddeliteľná konvolúcia) sa ukázala ako účinná v úlohách spracovania obrazu [19] a neurónového strojového prekladu [23]. Operátor hĺbkovo oddeliteľnej konvolúcie rozdeľuje štandardnú konvolučnú operáciu na dve po sebe idúce operácie, hĺbková konvolúcia (D-*conv*$(\cdot)$) nasledovaná bodovou konvolúciou (1×1-*conv*$(\cdot)$):

$$D\text{-}conv(\boldsymbol{Y}, \boldsymbol{K}) = concat(\boldsymbol{y}_j \circledast \boldsymbol{k}_j), j = 1, ..., N,$$
$$S\text{-}conv(\boldsymbol{Y}, \boldsymbol{K}, \boldsymbol{L}) = D\text{-}conv(\boldsymbol{Y}, \boldsymbol{K}) \circledast \boldsymbol{L},$$

kde $\boldsymbol{Y} \in \mathbb{R}^{G\times M}$ je vstup pre S-*conv*$(\cdot)$, $\boldsymbol{K} \in \mathbb{R}^{G\times P}$ je konvolučné jadro s veľkosťou P, $\boldsymbol{y}_j \in \mathbb{R}^{1\times M}$ a $\boldsymbol{k}_j \in \mathbb{R}^{1\times P}$ sú riadky matíc $\boldsymbol{Y}$ a $\boldsymbol{K}$ v tomto poradí. $\boldsymbol{L} \in \mathbb{R}^{G\times H\times 1}$ je konvolučné jadro s veľkosťou 1 a $\circledast$ označuje operáciu konvolúcie. Inými slovami, operácia D-*conv*$(\cdot)$ konvoluje každý riadok vstupu $\boldsymbol{Y}$ so zodpovedajúcim riadkom matice $\boldsymbol{K}$ a 1×1-*conv* blok lineárne transformuje priestor prvkov. V porovnaní so štandardnou konvolúciou s veľkosťou jadra $\hat{\boldsymbol{K}} \in \mathbb{R}^{G\times H\times P}$, hĺbkovo oddeliteľná konvolúcia obsahuje iba $G\times P+G\times H$ parametrov, čo znižuje veľkosť modelu o faktor $\frac{H\times P}{H+P} \approx P$, keď $H \gg P$.

Po každom prvom 1×1-*conv* a *D-conv* bloku je pridaná nelineárna aktivačná funkcia a za ňou hneď normalizačná operácia v uvedenom poradí. Nelineárnou aktivačnou funkciou je parametricky rektifikovaná lineárna jednotka (PReLU):

$$\mathrm{PReLU}(x) = \begin{cases} x, & \text{ak } x \geq 0 \\ \alpha x, & \text{inak} \end{cases},$$

kde $\alpha \in \mathbb{R}$ je trénovateľný skalár ovládajúci sklon pre záporné x usmerňovača. Výber typu metódy normalizácie v sieti závisí od požiadavky kauzality. Pre nekauzálne konfigurácie sa zistilo, že globálna normalizácia vrstvy (gLN) prekonáva všetky ostatné normalizačné metódy [27].

Na začiatku separačného modulu je lineárny 1×1-*conv* blok pridaný ako „bottleneck" vrstva. Tento blok určuje počet kanálov na vstupe a na reziduálnom spoji následných konvolučných blokov. Napríklad, ak má lineárna „bottleneck" vrstva B kanálov, potom pre 1-D konvolučný blok s H kanálmi a veľkosťou jadra P by mala byť veľkosť jadra v prvom 1×1-*conv* bloku a prvom *D-conv* bloku $\boldsymbol{L} \in \mathbb{R}^{B \times H \times 1}$ a $\boldsymbol{K} \in \mathbb{R}^{H \times P}$ v uvedenom poradí. Veľkosť jadra v reziduálnych spojoch by mala byť $\boldsymbol{L}_{Rs} \in \mathbb{R}^{H \times B \times 1}$. Počet výstupných kanálov v preskakovacích spojoch môže byť iná ako B a veľkosť jadier v tomto spoji sa označuje ako $\boldsymbol{L}_{Sc} \in \mathbb{R}^{V \times Sc \times 1}$.

Kapitola 5

Trénovanie neurónových sietí

5.1 Tréningové procesy a vlastnosti učenia

Jednou z najdôležitejších vlastností umelých neurónových sietí je ich schopnosť učiť sa z prezentácie vzoriek (vzorov), čo vyjadruje správanie systému. Keď sa teda sieť naučí vzťah medzi vstupmi a výstupmi, môže zovšeobecniť riešenia, čo znamená, že sieť môže produkovať výstup, ktorý je blízky očakávanému (alebo požadovanému) výstupu akýchkoľvek daných vstupných hodnôt [47].

Tréningový proces neurónovej siete preto pozostáva z aplikácie požadovaných ordinovaných krokov na vyladenie synaptických váh jej neurónov, aby sa zovšeobecnili riešenia produkované jej výstupmi. Treba však dávať pozor na príliš veľké zovšeobecnenie, pretože aplikácia týchto krokov môže v konečnom dôsledku viesť k pretrénovaniu.

Súbor zoradených krokov používaných na trénovanie siete sa nazýva algoritmus učenia. Počas svojej realizácie je tak sieť schopná extrahovať diskriminačné vlastnosti o mapovanom systéme zo vzoriek získaných zo systému.

Obvykle je dátová sada obsahujúca všetky dostupné vzory správania systému rozdelená do dvoch podmnožín, ktoré sa nazývajú tréningová podmnožina a testovacia podmnožina. Tréningová podmnožina sa používa v procese učenia. Na druhej strane, testovacia podmnožina sa použije na overenie, či je sieť schopná zovšeobecniť riešenie v rámci prijateľných úrovní, čím sa umožní validácia danej topológie. Pri dimenzovaní týchto podmnožín sa však musia brať do úvahy aj štatistické vlastnosti dátovej sady.

Počas tréningového procesu umelých neurónových sietí sa každý prechod algoritmu cez celý súbor tréningovej množiny, vykonávajúci úpravy synaptických váh neurónov, nazýva tréningová epocha. Je to hyper-parameter, ktorý určuje proces trénovania modelu strojového učenia.

5.1.1 Učenie s učiteľom

Stratégia učenia s učiteľom („supervised learning“) pozostáva z dostupnosti požadovaných výstupov pre daný súbor vstupných signálov. Inými slovami, každá tréningová vzorka sa skladá zo vstupných signálov a ich zodpovedajúcich výstupov. Vyžaduje to množinu dvojíc so vstupnými a výstupnými údajmi, ktorá predstavuje proces a jeho správanie. Práve z týchto informácií neurónové štruktúry formulujú „správanie systému“, z ktorého sa neurónová sieť učí. V tomto prípade aplikácia riadeného učenia závisí iba od dostupnosti týchto dvojíc a správa sa, ako keby „tréner“ učil sieť, aká je správna odpoveď pre každú vzorku vstupujúcu do siete.

Synaptické váhy neurónov siete sú neustále upravované aplikáciou komparatívnych akcií, vykonávaných samotným učiacim algoritmom, ktorý dohliada na nezrovnalosti medzi produkovanými výstupmi vzhľadom na požadované výstupy, využívajúc tento rozdiel v nastavovaní synaptických váh neurónov. Sieť sa považuje za „natrénovanú“, keď je tento nesúlad v prijateľnom rozsahu hodnôt, berúc do úvahy účely zovšeobecňovania riešení.

5.1.2 Učenie bez učiteľa

Učenie bez učiteľa („unsupervised learning“) sa používa pri dátach, ktoré neboli vopred klasifikované. Chýbajú tu teda požadované (správne) výstupy, čo znamená že nemožno porovnávať produkované výstupy s požadovanými. Tento prístup sa používa vtedy, keď nie je možné odpozorovať správne výstupy alebo ich jednoznačne určiť. Algoritmy učenia bez učiteľa sa potom v týchto neklasifikovaných dátach snažia objaviť, modelovať a popísať vzory, s cieľom dozvedieť sa o týchto dátach niečo viac.

Najčastejšími problémami, ktoré rieši učenie bez učiteľa, sú zhlukovanie a asociácia. Zhlukovanie je metóda zoskupovania objektov do zhlukov tak, že objekty s najväčšou podobnosťou zostávajú v jednej skupine a majú menšiu alebo žiadnu podobnosť s objektami inej skupiny. Asociácia hľadá spoločné znaky medzi dátovými objektami a kategorizuje ich podľa prítomnosti a neprítomnosti týchto spoločných znakov.

5.1.3 Učenie posilňovaním

Osobitnou kategóriou algoritmov strojového učenia je učenie posilňovaním („reinforcement learning“) resp. učenie formou odmeňovania. V tomto prípade sa už na trénovanie modelu nepoužijú žiadne označené, či neoznačené trénovacie príklady. Učenie tu prebieha tak, že sa vytvorí tzv. systémový agent, ktorý sa nasadí do prostredia a nechá sa učiť prostredníctvom interakcie s prostredím [22].

Jediné čo sa agentovi musí určiť, sú pravidlá ako sa môže v danom prostredí správať a tzv. odmeňovacia funkcia, ktorou odmeňuje alebo penalizuje vnútorné parametre neurónov. Tento proces učenia siete sa zvyčajne vykonáva metódou pokus-omyl, pretože jedinou dostupnou odpoveďou pre daný výstup je vo výsledku to, či bol uspokojivý alebo neuspokojivý. Ak je výstup uspokojivý, synaptické váhy sa postupne zvyšujú, aby sa posilnil (odmenil) tento behaviorálny stav spojený so systémom a naopak ak je výstup neuspokojivý tak sa váhy postupne znižujú.

5.2 Objektívna funkcia

V kontexte optimalizačného algoritmu sa objektívna funkcia („loss function“) používa na vyhodnotenie kandidátskeho riešenia (t. j. na množiny váh). V princípe ide o minimalizovanie objektívnej funkcie, čo znamená, že sa hľadá kandidátske riešenie, ktoré má najnižšie skóre (najmenšiu chybu). Objektívna funkcia sa často označuje aj ako stratová, nákladová alebo chybová funkcia a hodnota vypočítaná touto funkciou sa označuje ako celková chyba prípadne len strata („loss“) [14].

Objektívna funkcia znižuje všetky rôzne dobré a zlé aspekty potenciálne zložitého systému na jediné číslo, skalárnu hodnotu, ktorá umožňuje zoradiť a porovnať kandidátske riešenia. Zlepšenie (zmenšenie) v tomto čísle by malo byť znakom lepšieho modelu.

Pri výpočte chyby modelu počas procesu optimalizácie je potrebné zvoliť vhodnú objektívnu funkciu. Môže to byť náročný problém, pretože funkcia musí zachytávať vlastnosti

problému a musí byť motivovaná obavami, ktoré sú dôležité pre danú úlohu a zainteresované strany. Je preto dôležité, aby funkcia verne reprezentovala navrhnuté ciele. Ak sa zvolí zlá objektívna funkciu a získajú sa tak neuspokojivé výsledky, chyba je v zlej špecifikácii cieľa vyhľadávania [41].

5.3 Gradientný zostup

Gradientný zostup („gradient descent“) je v súčasnosti najpopulárnejšou optimalizačnou stratégiou na nájdenie lokálneho minima objektívnej funkcie používanou v strojovom učení a hlbokom učení. Myšlienkou metódy je posúvať sa z východiskového bodu po krokoch vždy v opačnom smere gradientu objektívnej funkcie v danom bode, pretože to je smer najstrmšieho klesania jej hodnoty, a tým sa iteratívne upravujú parametre trénovaného modelu.

Gradient je zovšeobecnenie sklonu (strmosti) funkcie pre viacero premenných. Je to vektor parciálnych derivácií podľa jednotlivých premenných skalárnej funkcie resp. zovšeobecnenie takéhoto vektora pre prípady, kde je namiesto skalárnej funkcie tenzor vyššieho rádu. Smer gradientu je smerom najväčšej zmeny danej funkcie. Zostup je činnosť smerovania nadol. Algoritmus gradientného zostupu preto definuje pohyb smerom nadol na základe dvoch jednoduchých definícií týchto fráz.

Objektívna funkcia $J(\theta)$ je parametrizovaná parametrami modelu $\theta \in \mathbb{R}^d$. Aktualizácia parametrov prebieha v opačnom smere ako je gradient objektívnej funkcie $\nabla_\theta J(\theta)$ vzhľadom na parametre:

$$\theta_{i+1} = \theta_i - \alpha \cdot \nabla_\theta J(\theta),$$

kde koeficient rýchlosti učenia α určuje veľkosť krokov, ktoré sa robia, aby sa dosiahlo (lokálne) minimum objektívnej funkcie. Inými slovami, pri gradientnom zostupe sa sleduje smer strmosti „svahu“ povrchu, ktorý je vytvorený objektívnou funkciou, ktorým sa posúva postupne smerom nadol až do „údolia“ [45].

Existujú tri varianty gradientného zostupu, ktoré sa líšia v tom, koľko údajov sa použije na výpočet gradientu objektívnej funkcie. Medzi tieto varianty patrí dávkový, stochastický a mini-dávkový gradientný zostup [24]. V závislosti od množstva údajov sa robí kompromis medzi presnosťou aktualizácie parametrov a časom potrebným na vykonanie aktualizácie.

5.3.1 Dávkový gradientný zostup

Dávkový gradientný zostup („batch gradient descent“) počíta gradienty objektívnej funkcie pre celú tréningovú množinu dátovej sady naraz. Keďže je potreba vypočítať gradienty pre celú množinu, aby sa vykonala len jedna aktualizácia, tento spôsob môže byť veľmi pomalý a nedá sa zvládnuť pre veľkú dátovú sadu, ktorá sa nezmestí do pamäte. Dávkový gradientný zostup tiež neumožňuje aktualizovať model online t. j. s použitím nových príkladov za chodu.

5.3.2 Stochastický gradientný zostup

Stochastický gradientný zostup („stochastic gradient descent“ – SGD) naopak vykonáva aktualizáciu parametrov pre každú jednotlivú tréningovú vzorku [24]. Dávkový gradientný zostup vykonáva redundantné výpočty pre veľké dátové sady, pretože prepočítava gradienty pre podobné príklady pred každou aktualizáciou parametrov. SGD odstraňuje túto nadbytočnosť tým, že sa vykonáva len jedna aktualizácia naraz. Je teda väčšinou oveľa rýchlejší

a dá sa použiť aj na online učenie. SGD vykonáva časté aktualizácie s vysokým rozptylom, ktoré spôsobujú silné kolísanie (fluktuáciu) hodnoty objektívnej funkcie.

V stochastickom gradientnom zostupe je skutočný gradient $J(\theta)$ aproximovaný gradientom na jednej tréningovej vzorke $x^{(j)}$ a prislúchajúcemu označeniu $y^{(j)}$:

$$\theta_{i+1} = \theta_i - \alpha \cdot \nabla_\theta J(\theta; x^{(j)}, y^{(j)}).$$

Keď algoritmus prechádza tréningovou množinou, vykoná aktualizáciu parametrov pre každú tréningovú vzorku. Cez tréningovú množinu je možné vykonať niekoľko prechodov, kým algoritmus neskonverguje. Ak sa tak stane, sada sa môže pre každý prechod premiešať, aby sa predišlo cyklom. Typické implementácie môžu využívať adaptívny koeficient rýchlosti učenia, takže algoritmus konverguje [34].

Pomocou pseudo-kódu, SGD môže byť reprezentovaný ako:

```
Vyber počiatočný vektor parametrov θ a koeficient rýchlosti učenia α.
Opakuj, kým model neskonverguje:
    Náhodne zamiešaj vzorky v tréningovej sade.
    Pre každé j = 1, 2, ..., n sprav:
        θ_{i+1} = θ_i − α · ∇_θ J(θ; x^(j), y^(j)).
```

5.3.3 Mini-dávkový gradientný zostup

Mini-dávkový gradientný zostup („mini-batch gradient descent“) nakoniec používa to najlepšie z oboch predošlých variant a vykonáva aktualizáciu parametrov pre určitú mini-dávku m tréningových vzoriek:

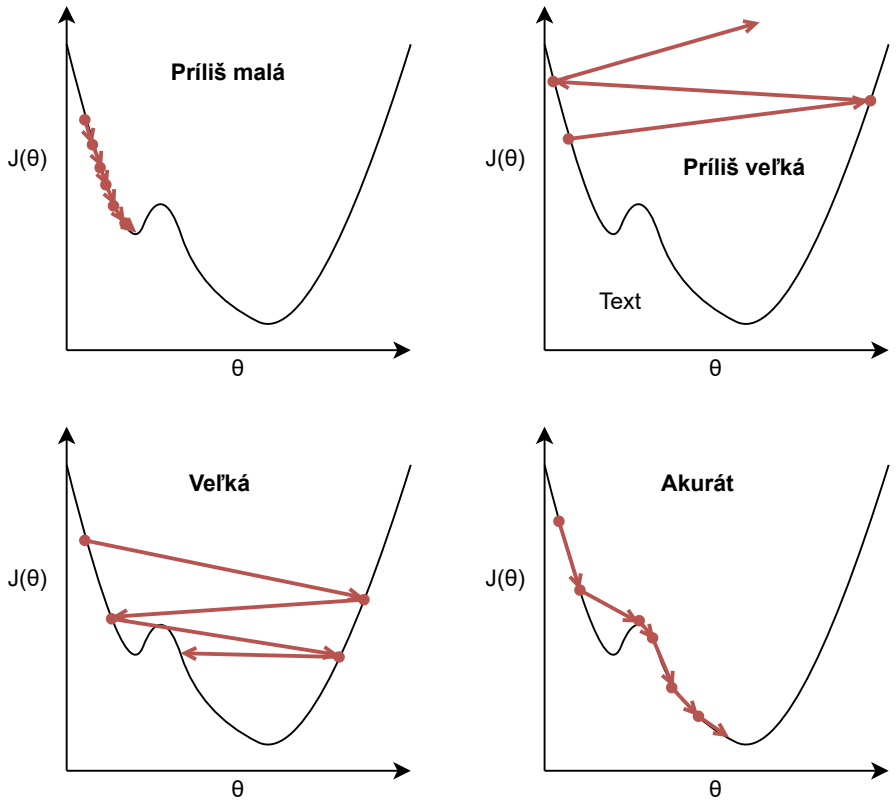
$$\theta_{i+1} = \theta_i - \alpha \cdot \nabla_\theta J(\theta; x^{(j:j+m)}, y^{(j:j+m)}).$$

Týmto spôsobom sa znižuje rozptyl aktualizácií parametrov, čo môže viesť k stabilnejšej konvergencii. Táto varianta môže využívať výhodné maticové optimalizácie bežné pre knižnice hlbokého učenia, vďaka ktorým je výpočet gradientu na základe mini-dávky veľmi efektívny. Bežné veľkosti mini-dávok sa pohybujú v rozmedzí 32 a 256, ale môžu sa líšiť pre rôzne aplikácie. Mini-dávkový gradientný zostup je zvyčajne prvou voľbou pri trénovaní neurónovej siete a termín SGD sa zvyčajne používa aj vtedy, keď sa používajú mini-dávky.

5.4 Koeficient rýchlosti učenia

Koeficient rýchlosti učenia („learning rate“) je veľmi dôležitým hyper-parametrom v optimalizačnom algoritme, ktorý určuje veľkosť kroku pri každej iterácii, pri ktorom sa gradientným zostupom pohybuje smerom k minimu objektívnej funkcie [34]. Keďže ovplyvňuje, do akej miery sú novozískané informácie dosiahnuté postupným trénovaním lepšie ako staré informáciami, metaforicky predstavuje rýchlosť, akou sa model strojového učenia „učí“. V literatúre o adaptívnom riadení sa koeficient rýchlosti učenia bežne označuje ako zisk („gain“) [9].

Pri nastavovaní koeficientu rýchlosti učenia existuje kompromis medzi mierou konvergencie a prekročením („overshooting“). Zatiaľ čo sa smer zostupu zvyčajne určuje pomocou gradientu objektívnej funkcie, koeficient rýchlosti učenia určuje, aký veľký krok sa v tomto smere urobí. Príliš vysoká hodnota koeficientu spôsobí, že učenie preskočí minimá, ale pri použití príliš nízkej hodnoty bude buď trvať príliš dlho, kým sa k nim priblíži, alebo sa zasekne na nežiadúcom lokálnom minime [4]. Hľadanie a nastavenie vhodného koeficientu rýchlosti učenia je vidieť na Obrázku 5.1.



Obr. 5.1: Vplyv hodnoty koeficientu rýchlosti učenia na konvergenciu[1].

Aby sa dosiahla rýchlejšia konvergencia, zabránilo sa osciláciám a uviaznutiu v nežiaducich lokálnych minimách, koeficient rýchlosti učenia sa počas tréningu často mení buď v súlade s rozvrhom koeficientu rýchlosti učenia („learning rate schedule“), alebo pomocou adaptívneho koeficientu rýchlosti učenia [38].

5.5 Vyhodnocovacie metriky

Meranie výsledkov konkrétnych netrénovaných modelov na separáciu zdrojov a ich porovnávanie je náročný problém. Vo všeobecnosti existujú dve hlavné kategórie na hodnotenie výstupov modelu separácie zdrojov, a to objektívne a subjektívne. Objektívne hodnotenie meria kvalitu separácie na základe vykonania súboru výpočtov, ktoré porovnávajú výstupné signály separačného modelu s izolovanými zdrojmi („ground truth“). Subjektívne merania zahŕňajú, že rôzni hodnotitelia, teda ľudia na to určení (vybratí), dávajú skóre jednotlivým výstupom modelu separácie zdrojov.

Objektívne aj subjektívne hodnotenia majú výhody aj nevýhody. Objektívne hodnotenia niekedy zaostávajú, pretože existuje veľa aspektov ľudského vnímania, ktoré je mimoriadne ťažké zachytiť iba výpočtovými prostriedkami. V porovnaní so subjektívnymi hodnoteniami sú však oveľa rýchlejšie a lacnejšie. Na druhej strane, subjektívne merania sú drahé, časovo náročné a podliehajú variabilite ľudských hodnotiteľov, ale môžu byť spoľahlivejšie ako objektívne merania, pretože do procesu hodnotenia sú zapojení skutoční ľudskí poslucháči.

[1] Inšpirované http://www.bdhammel.com/learning-rates/

5.5.1 SDR, SIR a SAR

Pomer zdroja k skresleniu („Source-to-Distortion Ratio“, SDR), pomer zdroja k interferencii („Source-to-Interference Ratio“, SIR) a pomer zdroja k artefaktom („Source-to-Artifact Ratio“, SAR) patria v dnešnej dobe ku najpoužívanejším metódam na objektívne vyhodnotenie výstupu modelu separácie zdrojov [30].

Pri odhade zdroja $\hat{\boldsymbol{s}}_i$ sa predpokladá, že sa v skutočnosti skladá zo štyroch samostatných komponentov:

$$\hat{\boldsymbol{s}}_i = s_{target} + e_{interf} + e_{noise} + e_{artif},$$

kde s_{target} je skutočným zdrojom a e_{interf}, e_{noise}, e_{artif} sú chybové výrazy pre rušenie („interference“), šum („noise“) a pridané artefakty („artifacts“) v tomto poradí. Skutočné výpočty týchto pojmov sú pomerne zložité, preto ich presný výpočet možno nájsť v článku [52].

Pomocou týchto štyroch pojmov môžno definovať vyhodnocovacie metriky. Všetky hodnoty metrík sú vyjadrené v decibeloch (dB), pričom vyššie hodnoty sú lepšie. Na ich výpočet je potrebný prístup k izolovaným zdrojom a zvyčajne sa počítajú na základe signálu, ktorý bol rozdelený do krátkych okien s dĺžkou niekoľkých sekúnd.

SAR

$$\text{SAR} = 10\log_{10}\left(\frac{||s_{target} + e_{interf} + e_{noise}||^2}{||e_{artif}||^2}\right)$$

Zvyčajne sa SAR interpretuje ako množstvo nežiaducich artefaktov, ktoré má odhad zdroja vo vzťahu k skutočnému zdroju.

SIR

$$\text{SIR} = 10\log_{10}\left(\frac{||s_{target}||^2}{||e_{interf}||^2}\right)$$

Táto metrika sa interpretuje ako množstvo iných zdrojov, ktoré možno počuť v odhade zdroja. Používa sa na to aj pojem krvácanie („bleed“) alebo únik („leakage“).

SDR

$$\text{SDR} = 10\log_{10}\left(\frac{||s_{target}||^2}{||e_{interf} + e_{noise} + e_{artif}||^2}\right)$$

SDR sa zvyčajne považuje za celkovú mieru toho, ako dobre znie zdroj. Ak sa niekde pri modeloch uvádza iba jedno číslo pre odhadovanú kvalitu, pravdepodobne ide o SDR.

SDR sa mnohokrát uvádza vo výskumných prácach. Toto číslo zvyčajne predstavuje priemer distribúcie SDR vypočítaný na dátovej sade. Preto nie je vždy garantované, že model s vyšším SDR je aj lepší, hlavne ak sa líši len o malé hodnoty.

Kapitola 6

Dáta

Dáta sú jednou zo základných zložiek pri vývoji a vyhodnocovaní algoritmov na separáciu hudobných zdrojov. Toto platí pri prístupe učenia s učiteľom. Pre úlohu separácie zdrojov hudby sa zvyčajne používa tento prístup. Cieľom separácie hudobných zdrojov, ako už bolo spomenuté, je oddeliť jednotlivé zdroje od zmesi zdrojov, ako je napríklad hudobná nahrávka. Aby sa mohli trénovať a vyhodnocovať algoritmy separácie zdrojov, sú potrebné údaje, ktoré pozostávajú zo zmesí a ich zodpovedajúcich zdrojových signálov.

Existuje niekoľko variant formy rozdelenia údajov, ktoré možno použiť na separáciu zdrojov hudby. Jedným z bežných variant sú viackanálové nahrávky, ktoré obsahujú samostatné zvukové kanály pre každý zdroj (nástroj alebo spev) v nahrávke. Tieto nahrávky je možné použiť na generovanie zmesí zdrojov sčítaním jednotlivých kanálov. Viackanálové nahrávky možno použiť aj na vyhodnotenie kvality oddelených zdrojov ich porovnaním s pôvodnými zdrojmi jednotlivých kanálov.

Ďalším variantom formy rozdelenia údajov používaných na separáciu zdrojov hudby sú dvojkanálové zmesi. V tomto prípade existujú iba dva zvukové kanály, ktoré možno rozdeliť na dva zdroje (napr. vokály a sprievod). V tomto prípade zdroj neznamená automaticky len jeden druh nástroja. Dvojkanálové zmesi sa často používajú na vyhodnotenie algoritmov, ktoré sú špeciálne navrhnuté na oddelenie spevu od ostatných hudobných nástrojov.

Pre separáciu hudobných zdrojov je okrem typu dát dôležitá aj kvalita dát. Vysokokvalitné údaje sú potrebné na zabezpečenie toho, aby boli zdroje dobre definované, a aby zmesi presne odzrkadľovali skutočné scenáre. Okrem toho je veľká a rôznorodá dátová sada („dataset“) dôležitá pre trénovacie algoritmy, ktoré sú robustné a dobre sa zovšeobecňujú na nové nahrávky.

Kvôli autorským právam je ťažké získať a zdieľať hudobné nahrávky na účely strojového učenia. Ešte ťažšie je však získať viackanálové nahrávky, ktoré obsahujú izolované zdroje jednotlivých kanálov, pretože ich umelci sprístupňujú len zriedka. Aj napriek tomu bola výskumná komunita schopná vytvárať a zdieľať viackanálové dátové sady. Veľkosť týchto sád je zvyčajne veľmi malá v porovnaní s inými dátovými sadami strojového učenia. Našťastie existujú nástroje na generovanie viacerých rôznych zmesí z rovnakej množiny stôp, čo pomáha maximalizovať to, čo sa model môže naučiť z danej množiny stôp [30].

6.1 Dátové sady

Stručný náhľad na niektoré používané dátové sady možno vidieť v Tabuľke 6.1.

Tabuľka 6.1: Prehľad dátových sád používaných na separáciu hudby.

Názov	Rok	Počet nahrávok	Celková dĺžka	Stereo
MIR-1K	2010	1000	133 min	Nie
MedleyDB	2014	122	7 h	Áno
DSD100	2015	100	7 h	Áno
iKala	2015	252	2,1 h	Nie
MedleyDB 2.0	2016	196	–	Áno
MUSDB18	2017	150	10 h	Áno
Slakh2100	2019	2100	145 h	Nie
MuseScore	2019	344,166	–	Nie
CocoChorales	2022	240,000	1400 h	Nie

MIR-1K je dátová sada z laboratória *Multimedia Information Retrieval* (MIR) obsahujúca 1000 nahrávok a určená na separáciu spevu. Každá nahrávka trvá 4 až 13 sekúnd. Tieto nahrávky sú extrahované zo 110 karaoke skladieb vybraných z čínskych popových skladieb a spievajú ich výskumníci z laboratória MIR [20]. Väčšina speváčov je amatérska.

MedleyDB je dátová sada viackanálových nahrávok navrhnutý na výskum separácie hudobných zdrojov a súvisiacich úloh. Obsahuje celkovo 122 skladieb, pokrývajúcich rôzne žánre a štýly. Každá skladba v sade je poskytovaná ako zbierka viackanálových nahrávok, kde každý kanál zodpovedá inému nástroju alebo zdroju zvuku v mixe [2]. *MedleyDB 2.0* priniesla 74 nových viackanálových nahrávok do pôvodnej dátovej sady MedleyDB. Nové skladby pochádzajú z rôznych žánrov, pričom najviac dominuje klasika a jazz [3].

DSD100 je dátová sada 100 kompletných hudobných skladieb rôznych štýlov spolu s ich izolovanými bubnami, basou, spevom a ostatnými zdrojmi. Dátová sada je pozoruhodná vysokou kvalitou svojich nahrávok, ktoré boli všetky profesionálne vyrobené. Okrem toho poskytuje rôzne typy metadát a anotácií vrátane informácií o inštrumentácii každej skladby, časovo zarovnaných textov a anotácií rytmu a tempa [26].

Dátová sada *iKala* je určená na separáciu spevu, ktorá obsahuje 252 pol-minútových častí skladieb pochádzajúcich z 206 čínskych pop-skladieb. Táto dátová sada obsahuje 2-kanálové nahrávky, kde jeden kanál predstavuje spev a druhý ostatný sprievod. Okrem skladieb obsahuje dátová sada aj časovo zoradené texty, ktoré možno použiť na úlohy ako je automatický prepis textu [6].

Synthesized Lakh (*Slakh*) Dataset je dátová sada, ktorá je syntetizovaná z Lakh MIDI Dataset v0.1 pomocou virtuálnych nástrojov na profesionálnej úrovni. Toto prvé vydanie Slakh, nazvané Slakh2100, obsahuje 2100 automaticky zmiešaných skladieb a sprievodných MIDI súborov syntetizovaných pomocou vzorkovacieho enginu [31].

Dátová sada *MuseScore* je zbierka 344 166 zvukových a MIDI párov stiahnutých z webovej stránky MuseScore. Kvôli problémom s autorskými právami nie je dátová sada verejne dostupná. Zvuk je zvyčajne syntetizovaný za pomoci MuseScore syntetizátora. Zvukové nahrávky majú rôzne hudobné žánre a sú v priemere dlhé asi dve minúty [21].

CocoChorales je dátová sada pozostávajúca z viac ako 1 400 hodín zvukových zmesí obsahujúcich štvordielne nahrávky v podaní 13 nástrojov, všetky syntetizované za pomoci realisticky znejúcich modelov. CocoChorales obsahuje mixy, zdroje a MIDI dáta, ako aj anotácie na vyjadrenie noty (napr. hlasitosť noty a vibrato) a parametre syntézy (napr. multi-f_0) [54].

6.2 MUSDB18

Ako už bolo skôr spomenuté, pri tejto neurónovej sieti je použitá na trénovanie dátová sada MUSDB18 [40]. MUSDB18 pozostáva zo 150 celých hudobných nahrávok, čo predstavuje cca 10 hodín hudby. Tieto nahrávky sú rôznych žánrov. Sada obsahuje stereo mixy a originálne zdroje, ktoré sú rozdelené na tréningovú a testovaciu sadu. Tréningová sada obsahuje 100 nahrávok a testovacia zvyšných 50.

MUSDB18 pozostáva zo 100 skladieb prevzatých z dátovej sady DSD100, ktoré sú prevzaté z bezplatnej knižnice *The Mixing Secrets*[1], 46 skladieb je prevzatých z dátovej sady MedleyDB, 2 skladby poskytla spoločnosť *Native Instruments*[2] a 2 skladby sú od kanadskej rockovej kapely *The Easton Ellises*.

Všetky nahrávky sú komprimované na AAC 256kbps a zabalené do *STEMS* mp4 formátu. Každá nahrávka obsahuje 5 stereo kanálov a to konkrétne:

- 0 - mix
- 1 - bubny
- 2 - basy
- 3 - ostatné sprievodné nástroje
- 4 - vokály

Táto dátová sada bola použitá vrámci *Signal Separation Evaluation Campaign 2018*[3] (SiSEC 2018) [51].

[1] https://www.cambridge-mt.com/ms/mtk/
[2] https://www.native-instruments.com/en/specials/stems-for-all/free-stems-tracks/
[3] https://sisec18.unmix.app

Kapitola 7

Návrh modifikácií

Táto kapitola popisuje návrh modifikácií pre už existujúcu implementáciu neurónovej siete Conv-TasNet. Kód spoločne so skriptami potrebnými na trénovanie tejto siete sa nachádza na *GitHub* repozitári od výskumnej skupiny *Meta Research* s názvom Demucs[1]. Konkrétnejšie bola použitá vetva („branch“) *v2*. Postupne sa prejde všetkými navrhnutými modifikáciami. V sekcií 7.1 sa popisuje nastavenie štruktúry siete Conv-TasNet, zmena počtu vzoriek na vstupe a zmena počtu epoch. Sekcia 7.2 ukazuje postup pri transformácii signálov z časovej domény na frekvenčnú. Ďalej sa sekcia 7.3 zaoberá zmenou objektívnej funkcie na iné. Tu sú popísané vzorce navrhnutých objektívnych funkcií. Posledná sekcia 7.4 hovorí o zmene koeficientu rýchlosti učenia a nájdenie vhodného koeficientu pre každú navrhnutú objektívnu funkciu z predošlej sekcie.

7.1 Konfigurácie siete

Ako jeden z prvých experimentov stojí za vyskúšanie natrénovať sieť s predvolenými hodnotami na rôznych počtoch grafikách GPU. Predvolené hodnoty hyper-parametrov siete a konfigurácie trénovania sú v Prílohe A. Podľa výsledkov a času na to potrebného sa ďalej bude experimentovať s hyper-parametrami štruktúry siete za účelom zmenšenia/zväčšenia. Hyper-parametre siete všeobecne definujú jej štruktúru ako je napr. počet skrytých vrstiev a premenné, ktoré určujú, ako bude sieť trénovaná (napr. koeficient rýchlosti učenia). Hyper-parametre štruktúry tejto siete možno vidieť v Tabuľke 7.1.

Ďalším experimentom bude zmena počtu vzoriek na vstupe z pôvodných 441 000 na 80 000 a 44 100. Počet vzoriek 80 000 bolo použitých na referenčné trénovanie, ktoré uvádzajú vo svojom *README.md* súbore pre Conv-TasNet ako východzie. Počet vzoriek 44 100 je rovnaký ako vzorkovacia frekvencia, a preto stojí za vyskúšanie aj tento experiment. Menší počet vzoriek na vstupe môže viesť k modelu s horším výsledkom. Ako ďalší bude experiment s rôznym počtom epoch.

7.2 Transformácia do frekvenčnej domény

Časová doména a frekvenčná doména sú dva režimy používané na analýzu údajov. Keďže sieť pracuje so signálmi v časovej doméne, jednou z modifikácií bude transformácia signálov do frekvenčnej domény pre účel hodnotenia objektívnych funkcií. Na túto transformáciu sa využije krátkodobá Fourierova transformácia STFT („Short Time Fourier Transform“).

[1] https://github.com/facebookresearch/demucs/tree/v2

Tabuľka 7.1: Hyper-parametre štruktúry siete Conv-TasNet.

Symbol	Popis	Predvolená hodnota
X	Počet konvolučných blokov v každom opakovaní	8
N	Počet filtrov v kodéri	256
L	Dĺžka jadier (vo vzorkách)	20
B	Počet kanálov v „bottleneck“-u a vo zvyškových spojoch 1×1-*conv* blokov	256
H	Počet kanálov v konvolučných blokoch	512
P	Veľkosť jadra v konvolučných blokoch	3
R	Počet opakovaní	4

Ako oknová funkcia („window function“) bude použité Hannove okno. Dĺžka okien bude postupne 512, 1024 a 2048. Ako dĺžka skoku („hop“) budú použité hodnoty 256 a 512.

7.3 Objektívne funkcie

Keďže objektívne funkcie majú vplyv na trénovanie siete, ďalšou modifikáciou bude experimentovanie s rôznymi objektívnymi funkciami ako v časovej doméne tak aj vo frekvenčnej. Rovnice výpočtov sú inšpirované článkami [15, 17].

7.3.1 Časová doména

Vzhľadom na audio x_t s $\mathcal{T}$ vzorkami, extrahujeme dávku („batch“) o veľkosti B z K rôznych zdrojov z C rôznych kanálov (v tomto prípade stereo) $\bar{y}_{t,c,k,b}$.

L1 a L2

Objektívna funkcia L1 je základnou a veľmi často používanou objektívnou funkciou. Vypočíta sa ako stredná absolútna odchýlka („Mean Absolute Error“ – MAE) medzi predpoveďou a skutočnou hodnotou pri každom prvku zo vstupu. Oobjektívna funkcia L2 patrí tiež medzi často používané. Výpočet je podobný ako u L1, ale rozdiel hodnôt sa ešte umocní na druhú. Označuje sa aj ako stredná kvadratická odchýlka („Mean Squared Error“ – MSE).

$$\mathrm{L1}_{\mathrm{time}} = \frac{1}{\mathcal{T}CKB} \sum_{t,c,k,b} \left| \bar{y}_{t,c,k,b} - y_{t,c,k,b} \right|$$

$$\mathrm{L2}_{\mathrm{time}} = \frac{1}{\mathcal{T}CKB} \sum_{t,c,k,b} \left| \bar{y}_{t,c,k,b} - y_{t,c,k,b} \right|^2$$

Logaritmická L1 a L2

Objektívne funkcie logaritmická L1 a L2 sú modifikáciou obyčajných L1 a L2 tým, že je tam vložený logaritmus.

$$\text{LogL1}_{\text{time}} = \frac{10}{CKB} \sum_{c,k,b} \log_{10} \sum_{t} \left| \bar{y}_{t,c,k,b} - y_{t,c,k,b} \right|$$

$$\text{LogL2}_{\text{time}} = \frac{10}{CKB} \sum_{c,k,b} \log_{10} \sum_{t} \left| \bar{y}_{t,c,k,b} - y_{t,c,k,b} \right|^2$$

SI-SDR

Pomer signálu k skresleniu invariantný ku škálovaniu SI-SDR („Scale-Invariant Signal-to-Distortion Ratio“) sa používa na trénovanie aj hodnotenie.

$$\text{SISDR}_{\text{time}} = \frac{-10}{CKB} \sum_{c,k,b} \log_{10} \frac{\sum_t \left| \alpha \cdot y_{t,c,k,b} \right|^2}{\sum_t \left| \alpha \cdot y_{t,c,k,b} - \bar{y}_{t,c,k,b} \right|^2},$$

$$\text{kde } \alpha = \frac{\sum_t \left| \bar{y}_{t,c,k,b} \cdot y_{t,c,k,b} \right|}{\sum_t \left| y_{t,c,k,b} \right|^2}$$

7.3.2 Frekvenčná doména

Krátkodobú Fourierovu transformáciu definujeme ako $STFT(x_t) = X_{n,\omega}$, kde n a ω predstavujú indexy rámcov a frekvenčných elementov („bin“) z celkového počtu N rámcov a Ω frekvenčných elementov. Objektívne funkcie sa tak upravia nasledovne:

L1 a L2

$$\text{L1}_{\text{freq}} = \frac{1}{N\Omega CKB} \sum_{n,\omega,c,k,b} \left| |\bar{Y}_{n,\omega,c,k,b}| - |Y_{n,\omega,c,k,b}| \right|$$

$$\text{L2}_{\text{freq}} = \frac{1}{N\Omega CKB} \sum_{n,\omega,c,k,b} \left| |\bar{Y}_{n,\omega,c,k,b}| - |Y_{n,\omega,c,k,b}| \right|^2$$

Logaritmická L1 a L2

$$\text{LogL1}_{\text{freq}} = \frac{10}{CKB} \sum_{c,k,b} \log_{10} \sum_{n,\omega} \left| |\bar{Y}_{n,\omega,c,k,b}| - |Y_{n,\omega,c,k,b}| \right|$$

$$\text{LogL1}_{\text{freq}} = \frac{10}{CKB} \sum_{c,k,b} \log_{10} \sum_{n,\omega} \left| |\bar{Y}_{n,\omega,c,k,b}| - |Y_{n,\omega,c,k,b}| \right|^2$$

SI-SDR

$$\text{SISDR}_{\text{freq}} = \frac{-10}{CKB} \sum_{c,k,b} \log_{10} \frac{\sum_{n,\omega} \left| \alpha \cdot Y_{n,\omega,c,k,b} \right|^2}{\sum_{n,\omega} \left| \alpha \cdot Y_{n,\omega,c,k,b} - \bar{Y}_{n,\omega,c,k,b} \right|^2},$$

$$\text{kde } \alpha = \frac{\sum_{n,\omega} \left| \bar{Y}_{n,\omega,c,k,b} \cdot Y_{n,\omega,c,k,b} \right|}{\sum_{n,\omega} \left| Y_{n,\omega,c,k,b} \right|^2}$$

7.4 Koeficient rýchlosti učenia

Koeficient rýchlosti učenia je dôležitý hyper-parameter, ktorý nám určuje ako rýchlo bude sieť konvergovať k minimu objektívnej funkcie. Preto ďalší experiment bude pozostávať z nájdenia najväčšieho optimálneho koeficientu rýchlosti učenia pre každú navrhnutú objektívnu funkciu z predošlej sekcie. Tento pokus bude prebiehať na zmenšenej vzorke dát s menšími hyper-parametrami siete za účelom pomerne rýchleho zistenia najväčšieho koeficientu rýchlosti učenia, pri ktorej bude model konvergovať. Predvolený koeficient rýchlosti učenia, ako už bolo spomenuté je 3×10^{-4}. Postupne na každej objektívnej funkcii, budú testované tieto koeficienty rýchlosti učenia:

- 3×10^{-4}
- 1×10^{-3}
- 3×10^{-3}
- 1×10^{-2}

7.4.1 Zmenšovanie koeficientu rýchlosti učenia

Poslednou modifikáciou bude postupné zmenšovanie koeficientu rýchlosti učenia vždy o polovicu, keď sa za istý počet epoch nezmenší celková chyba („loss“). Počiatočný koeficient bude prevzatý z predošlého experimentu nájdenia optimálneho koeficientu rýchlosti učenia. To, o koľko presne epoch bude treba zmeniť koeficient rýchlosti učenia sa zistí z vyhodnotenia predošlých experimentov, keď sa po dlhšiu dobu nebude meniť najlepšia celková chyba.

7.5 Sumarizácia

Výslednú sumarizáciu navrhnutých modifikácií možno vidieť v Tabuľke 7.2.

Tabuľka 7.2: Sumarizácia navrhnutých modifikácií.

Číslo	**Popis modifikácie**
1	Zmena konfigurácie štruktúry siete Conv-TasNet
2	Zmena počtu vzoriek na vstupe
3	Zmena celkového počtu epoch
4	Transformácia signálov do frekvenčnej domény
5	Experiment s rôznymi objektívnymi funkciami
6	Nájdenie optimálneho koeficientu rýchlosti učenia pre každú objektívnu funkciu
7	Zmenšovanie koeficientu rýchlosti učenia

Kapitola 8

Implementácia a výsledky

Táto kapitola popisuje výslednú implementáciu modifikácií a ich výsledky.

8.1 Príprava prostredia

Na experimentovanie so sieťou Conv-TasNet bolo použité výpočtové centrum *MetaCentrum*[1]. *MetaCentrum* je výpočtové centrum, ktoré distribuuje viacero výpočtových infraštruktúr a sprístupňuje výpočtové zdroje na riešenie náročných výpočtových úloh, medzi ktoré patrí aj strojové učenie. Ďalej na to, aby sa mohli vykonať experimenty, je potrebné stiahnuť kód spoločne so skriptami potrebnými na trénovanie tejto siete (popísané na začiatku kapitoly 7). Na účely trénovania bola stiahnutá dátová sada MUSDB18[2]. Všetky tieto dáta a kódy boli uložené do gridovej infraštruktúry *MetaCentrum*.

Ako nasledujúca, veľmi dôležitá vec je príprava prostredia, v ktorom sa bude sieť trénovať. Na vytvorenie prostredia je potrebné mať nainštalovanú distribúciu *Anaconda*. Je to bezplatná platforma s otvoreným zdrojovým kódom, ktorá umožňuje vytvárať a spravovať virtuálne prostredia s programovacím jazykom Python. Za pomoci *condy* je potrebné naištalovať tieto závislosti v daných verziách:

- **python** verzia >=3.7, <3.9
- **ffmpeg** verzia >=4.2
- **pytorch** verzia >=1.8.1
- **cudatoolkit** verzia >=10
- **torchaudio** verzia >=0.8
- **tqdm** verzia >=4.36
- **pip**

Ďalšie závislosti sú nainštalované pomocou inštalátora balíkov už nainštalovaného pip-u a to konkrétne:

- **diffq** verzia <0.2.0
- **julius** verzia >=0.2.3

[1] https://metavo.metacentrum.cz
[2] Dátová sada MUSDB18 z https://sigsep.github.io/datasets/musdb.html

- **lameenc** verzia >=1.2
- **musdb** verzia >=0.4.0
- **museval** verzia >=0.4.0
- **soundfile**
- **treetable** verzia >=0.2.3

Posledným potrebným programom je *SoundStretch/SoundTouch*[3], ktorý je potreba taktiež nainštalovať. Tento program sa používa na zvýšenie výšky tónu/tempa. V tomto momente je prostredie plne nachystané a môže sa tak začať experimentovať s modifikáciami.

8.2 Trénovanie na rôznych počtoch GPU

Referenčné trénovanie siete Conv-TasNet, teda to, ktoré uvádzajú vo svojom *GitHub* repozitári, prebiehalo na 8 *V100* GPU s 32GB RAM, pri ktorom bolo nastavenie hyperparametrov siete a trénovania zmenené oproti predvolenému (predvolené nastavenie možno vidieť v Kapitole A). Zmenené parametre boli tieto 3:

- počet vzoriek na vstupe 441 000 → 80 000
- veľkosť dávky 64 → 32
- počet konvolučných blokov v každom opakovaní (X) 8 → 10

V MetaCentre je viacero GPU klastrov, ale len 3 také na ktorých GPU majú aspoň 32GB RAM. Tieto klastre majú viacero uzlov. Jeden z týchto klastrov má len 1 GPU na uzol, preto sa tento kluster nebude používať na trénovanie, a tak ostali len 2 klastre (*Galdor, Zia*) na ktorých možno trénovať sieť. Oba tieto klastre majú 4 GPU na jeden uzol. Klaster *Galdor* je obstaraný *A40* GPU s 48GB RAM a klaster *Zia* má *A100* GPU s 42GB RAM. Z dôvodu príliš dlhého čakania na pridelenie viacerých uzlov naraz, sa trénovalo vždy len na 1 uzle a teda maximálne na 4 GPU naraz. Kvôli tomuto obmedzeniu bolo nutné zmenšiť veľkosť dávky oproti referenčnému trénovaniu, aby sa celá dávka zmestila do GPU RAM pamäte, a preto sa z prepočtu referenčného trénovania, kedy mal beh veľkosť dávky rovný 32 a bol trénovaný na 8 GPU vyráta, že na 1 GPU pripadá veľkosť dávky rovná 4 (32 / 8 = 4). Výsledné veľkosti dávky preto sú:

- trénovanie na 1 GPU – veľkosť dávky 4
- trénovanie na 2 GPU – veľkosť dávky 8
- trénovanie na 4 GPU – veľkosť dávky 16

Tieto 3 konfigurácie trénovania boli prevedené a výsledky je vidieť v Tabuľke 8.1, kde je pridaný, ako prvý, výsledok referenčného trénovania. V tabuľke sú uvedené výsledky postupne celkové SDR, SDR pre jednotlivé zdroje, SIR, SAR (všetky hodnoty v dB). Všetky tieto trénovania trvali cca 8 dní čistého času t. j. času potrebného na trénovanie bez času spotrebovaného na čakanie na pridelenie zdrojov. Čakacia doba na pridelenie zdrojov pre 1 GPU sa pohybovala približne v rozmedzí 0-5h, pre 2 GPU v rozmedzí 1-24h a pre 4 GPU to bolo 3-7dní. Z tohto dôvodu sa ďalej trénovalo už len na 2 GPU, keďže je všeobecne lepšie trénovať na viac GPU a zároveň čakacia doba na pridelenie zdrojov je pomerne akceptovateľná vzhľadom na ostatné trénovania.

[3]Dostupný z https://www.surina.net/soundtouch/soundstretch.html

Tabuľka 8.1: Výsledky trénovania siete Conv-TasNet na rôznych počtoch GPU spolu s referenčným výsledkom.

Konfigurácia trénovania	SDR	Bubny	Basy	Ostatné	Vokály	SIR	SAR
Referencia, 8 GPU, veľkosť dávky 32	5,73	6,02	6,20	4,27	6,43	11,75	6,06
1 GPU, veľkosť dávky 4	5,98	6,17	6,51	4,48	6,77	12,23	6,25
2 GPU, veľkosť dávky 8	5,98	6,32	6,31	4,40	6,89	12,01	6,39
4 GPU, veľkosť dávky 16	6,08	6,29	6,56	4,62	6,84	11,79	6,29

8.3 Zmenšenie štruktúry siete

Z predošlého experimentu sa zistilo, že celé trénovanie siete trvá približne 8 dní čistého času, čo je pomerne veľa ak sa má so sieťou viac experimentovať. Z tohto dôvodu sa na ďalšie experimenty štruktúra siete zmenšila a to spôsobom, že sa upravili tieto hyper-parametre štruktúry siete z Tabuľky 7.1:

- **L** – $20 \rightarrow 80$
- **B** – $256 \rightarrow 128$
- **H** – $512 \rightarrow 256$
- **R** – $4 \rightarrow 3$
- **X** – zostane na pôvodnej hodnote 8

Toto zmenšenie siete skrátilo čistý čas behu trénovania z pôvodných 8 dní na zhruba 5 dní, s čim už je ľahšie pracovať. Samozrejme ako následok zmenšenia štruktúry siete je výsledok horší. Porovnanie pôvodného trénovania a trénovania so zmenšenou štruktúrou siete možno vidieť v Tabuľke 8.2. V tabuľke sú výsledky postupne celkové SDR, SDR pre jednotlivé zdroje, SIR, SAR (SDR, SIR a SAR výsledky sú uvádzané v dB).

Tabuľka 8.2: Výsledky trénovania siete Conv-TasNet s pôvodnou štruktúrou siete a zmenšenou. Výsledky sú uvedené v dB.

Štruktúra siete	SDR	Bubny	Basy	Ostatné	Vokály	SIR	SAR
Pôvodná	5,98	6,32	6,31	4,40	6,89	12,01	6,39
Zmenšená	5,28	5,89	5,85	3,69	5,67	10,29	5,63

8.4 Transformácia do frekvenčnej domény

Transformácia do frekvenčnej domény bola implementovaná pomocou knižnice *PyTorch*. Transformáciu vykonáva funkcia `make_freq`. Táto funkcia má ako vstupné parametre 2 tenzory a to predpoveď („predictions“) a cieľ („target“), ktoré sú v časovej doméne. Ďalšie parametre sú voliteľné. Výstupom funkcie sú 2 tenzory tiež predpoveď a cieľ, ale už vo frekvenčnej doméne.

Parametre funkcie `make_freq`:

- `predictions` (Tenzor) – vstupný tenzor, predpoveď
- `target` (Tenzor) – vstupný tenzor, cieľ
- `length` (int, voliteľný) – veľkosť rámca okna pre Fourierovu transformáciu a STFT filtra (predvolené: 2048)
- `hop` (int, voliteľný) – vzdialenosť medzi susednými rámcami okien (predvolené: 512)
- `pad` (str, voliteľný) – ovláda metódu vypchávky – „padding“ (predvolené: ”`constant`”)

```
import torch

def make_freq(predictions, target,
    length = 2048,
    hop = 512,
    pad = "constant"):

    win = torch.hann_window(length, device="cuda")

    # STFT input must be either 1D or 2D, flatten 3D/4D tensor to acceptable shape
    if target.dim() == 4:   # TRAIN cycle [batch, sources, stereo, samples]
        pr = torch.flatten(predictions, start_dim=0, end_dim=2)
        tr = torch.flatten(target, start_dim=0, end_dim=2)
        tensor_dims = [target.size(dim=0), target.size(dim=1), target.size(dim=2)]

    else:                   # VALID cycle [sources, stereo, samples]
        pr = torch.flatten(predictions, start_dim=0, end_dim=1)
        tr = torch.flatten(target, start_dim=0, end_dim=1)
        tensor_dims = [target.size(dim=0), target.size(dim=1)]

    # STFT
    pred = torch.stft(pr, length, hop_length=hop, window=win, pad_mode=pad, return_complex=
    True)
    tar = torch.stft(tr, length, hop_length=hop, window=win, pad_mode=pad, return_complex=
    True)

    # reshape tensor back
    freq = list(tar.size())
    size = [*tensor_dims, freq[-2], freq[-1]]
    pred = torch.reshape(pred, size)
    tar = torch.reshape(tar, size)

    return pred, tar
```

Výpis 8.1: Funkcia na transformáciu tenzorov do frekvenčnej domeny.

Kód pre funkciu `make_freq` je možno vidieť vo Výpise 8.1. Vstupné tenzory majú 3 alebo 4 dimenzie v závislosti na konkrétnom cykle iterácie, pričom pod pojmom iterácie

chápeme epochu a vrámci nej sú 2 typy cyklov, a to cyklus trénovania a cyklus validácie. Keď sa iterácia nachádza v cykle trénovania tak má 4 dimenzie a to [dávka, zdroje, stereo, vzorky]. Počas cyklu validácie má tenzor 3 dimenzie [zdroje, stereo, vzorky], dimenzia dávky tu chýba pretože tento cyklus nerozdeľuje vstupy na dávky, ale používa celý vstup naraz. Za pomoci `torch.stft` sú tenzory transformované do frekvenčnej domény. Nakoľko ale `torch.stft` očakáva na vstupe len 1D alebo 2D tenzor, tak je potrebné tieto vstupné 3D/4D tenzory vektorizovať („flatten“). Tenzory sa vektorizujú tak, aby posledná dimenzia (vzorky) zostala sama ako druhá a ostatné dimenzie sú zjednotené dokopy, t. j. 2D tenzor [(dávka+)zdroje+stereo, vzorky]. Prevedením `torch.stft` dostávame tenzor [(dávka+)zdroje+stereo, frekvenčný bin, rámce], kde frekvenčný bin predstavuje počet frekvencií na ktorých je aplikovaná STFT. Po STFT je potrebné ešte pretvarovať tenzor spať, a to dimenzie ktoré boli vektorizované dokopy sa teraz vrátia do pôvodného tvaru a teda výstupné tenzory sú 4D/5D [(dávka,) zdroje, stereo, frekvenčný bin, rámce] v závislosti v ktorom cykle iterácie sa trénovanie siete nachádza (4D pre validáciu a 5D pre trénovanie).

8.5 Objektívne funkcie

Na implementáciu objektívnych funkcií bola využitá knižnica *PyTorch*. Kód pre všetky objektívne funkcie je uložený v súbore `loss.py` na priloženom pamäťovom médiu. Objektívne funkcie L1 a L2 (MSE) už boli súčasťou skriptov pre Conv-TasNet, takže ich nebolo potrebné znovu implementovať. Pri trénovaní s logaritmickými objektívnymi funkciami podľa vzorcov zo sekcie 7.3 modely nekonvergovali. Aj pri opätovnom skúšaní zmeny koeficientu rýchlosti učenia tieto modely nekonvergovali. Zistil som, že modely konvergujú vtedy ak vnútorná suma nie je len cez vzorky (vo frekvenčnej doméne cez frekvenčný bin a rámce), ale aj cez kanály a zdroje. S takto upravenými rovnicami pre výpočty logaritmických objektívnych funkcií modely konvergovali. Rovnice som upravil nasledovným spôsobom:

$$\mathrm{LogL1}_{\mathrm{time}} = \frac{10}{CKB} \sum_{b} \log_{10} \sum_{t,c,k} \left| \bar{y}_{t,c,k,b} - y_{t,c,k,b} \right|$$

$$\mathrm{LogL1}_{\mathrm{freq}} = \frac{10}{CKB} \sum_{b} \log_{10} \sum_{n,\omega,c,k} \left| |\bar{Y}_{n,\omega,c,k,b}| - |Y_{n,\omega,c,k,b}| \right|$$

Rovnakým spôsobom som upravil aj funkcie logaritmická L2, SISDR a to ako aj v časovej doméne tak aj frekvenčnej.

Pri predvolenom nastavení je sieť trénovaná s objektívnou funkciou L1 v časovej doméne. Pre ostatné objektívne funkcie boli pridané nasledovné prepínače ako vstupné argumenty pri spustení trénovania:

`--mse`	pre $\mathrm{L2}_{\mathrm{time}}$ (MSE)
`--logL1`	pre $\mathrm{LogL1}_{\mathrm{time}}$
`--logL2`	pre $\mathrm{LogL2}_{\mathrm{time}}$
`--SISDR`	pre $\mathrm{SISDR}_{\mathrm{time}}$
`--freqL1`	pre $\mathrm{L1}_{\mathrm{freq}}$
`--freqMSE`	pre $\mathrm{L2}_{\mathrm{freq}}$ (MSE)
`--freqLogL1`	pre $\mathrm{LogL1}_{\mathrm{freq}}$
`--freqLogL2`	pre $\mathrm{LogL2}_{\mathrm{freq}}$
`--freqSISDR`	pre $\mathrm{SISDR}_{\mathrm{freq}}$

V Tabuľke 8.3 je možno vidieť výsledky pre jednotlivé objektívne funkcie. Tento experiment prebehol na zmenšenej sieti spolu s nastavením veľkosti dávky na hodnotu 8. Pre objektívne funkcie pracujúce vo frekvenčnej doméne bola použitá veľkosť rámu okna Fourierovej transformácie 512 a vzdialenosť medzi oknami 256. Ako je vidieť v tabuľke, objektívne funkcie pracujúce v časovej doméne dosahujú približne rovnaké výsledky. Ako najlepšia v tomto experimente vyšla funkcia logaritmická L2 a hneď za ňou obyčajná L1. Výsledky pre objektívne funkcie vo frekvenčnej doméne dosahovali záporné hodnoty SDR. SDR je logaritmus pomeru signálu ku skresleniu. Takže SDR s hodnotou 0 znamená, že sa signál rovná skresleniu. Hodnota SDR menšia ako 0 znamená, že existuje väčšie skreslenie ako signál. V separovaných nahrávkach s modelmi kde bola použitá transformácia do frekvenčnej domény je počuť tzv. prašťanie, čo môže byť ako jeden z dôsledkov záporného SDR. Ďalším problémom je, že model by sa mal naučiť predpovedať výstupy zarovnané so vzorkami, po prevedení transformácie za pomoci STFT sa však fáza nepredpovedá presne a aj to môže byť dôvod záporného SDR. Môže to byť spôsobené aj zlým nastavením veľkosti okna Fourierovej transformácie, ktoré môže byť príliš malé. Model s objektívnou funkciou SISDR v časovej doméne dosiahol veľmi nízku hodnotu SDR, čo môže byť signalizácia zlej implementácie, a preto nebol vykonaný ani experiment s touto funkciou vo frekvenčnej doméne, keďže sa tu používa rovnaká časť výpočtu.

Tabuľka 8.3: Výsledky trénovania siete Conv-TasNet s rôznymi objektívnymi funkciami. Výsledky sú uvedené v dB.

Obj. funkcia	SDR	Bubny	Basy	Ostatné	Vokály	SIR	SAR
$L1_{time}$	5,28	5,89	5,85	3,69	**5,67**	10,29	5,63
$L2_{time}$	5,18	**6,09**	5,66	**3,71**	5,26	9,49	5,82
$LogL1_{time}$	5,14	5,75	5,79	3,59	5,42	**10,38**	5,65
$LogL2_{time}$	**5,33**	6,07	**6,08**	3,62	5,57	9,55	**5,91**
$SISDR_{time}$	-91,05	-92,39	-93,52	-86,62	-91,67	4,10	5,25
$L1_{freq}$	-5,14	-5,10	-5,50	-4,74	-5,21	8,46	2,96
$L2_{freq}$	-0,97	4,14	-2,98	-2,36	-2,67	8,04	5,33
$LogL1_{freq}$	-2,76	-5,29	4,71	-5,10	-5,36	8,85	4,88
$LogL2_{freq}$	-2,42	3,32	-2,68	-5,04	-5,27	7,85	5,23

8.6 Trénovanie s menším počtom vzoriek a väčšou dávkou

Ako ďalší experiment bolo vykonané trénovanie s menším počtom vzoriek a to konkrétne s hodnotou 44 100 a veľkosťou dávky 32. Pre objektívne funkcie pracujúce vo frekvenčnej doméne bola použitá veľkosť okna Fourierovej transformácie 1024 a vzdialenosť medzi oknami 512, a to kvôli záporným výsledkom SDR predchádzajúceho experimentu, kedy sa použilo menšie okno. Výsledky trénovania sú v Tabuľke 8.4. Ako je vidieť, dosiahnuté výsledky sú horšie ako v predošlom pokuse, a preto experiment nepokračoval na všetkých objektívnych funkciách. Z uvedených výsledkov aj tak možno pozorovať, že funkcia logaritmická L2 má najvyššie SDR skóre a tesne za ňou opätovne L1, obe v časovej doméne. Funkcie pracujúce vo frekvenčnej doméne znova dosahujú záporné SDR, ale je vidieť, že s použitím väčšej veľkosti okna pre Fourierovu transformáciu sú výsledky SIR v relatívnom porovnaní oproti

funkciám v časovej doméne lepšie ako v predošlom pokuse, čo je pozitívne zistenie a vedie k tomu, že pôvodná veľkosť okna bola príliš malá.

Tabuľka 8.4: Výsledky trénovania siete Conv-TasNet s menším počtom vzoriek. Výsledky sú uvedené v dB.

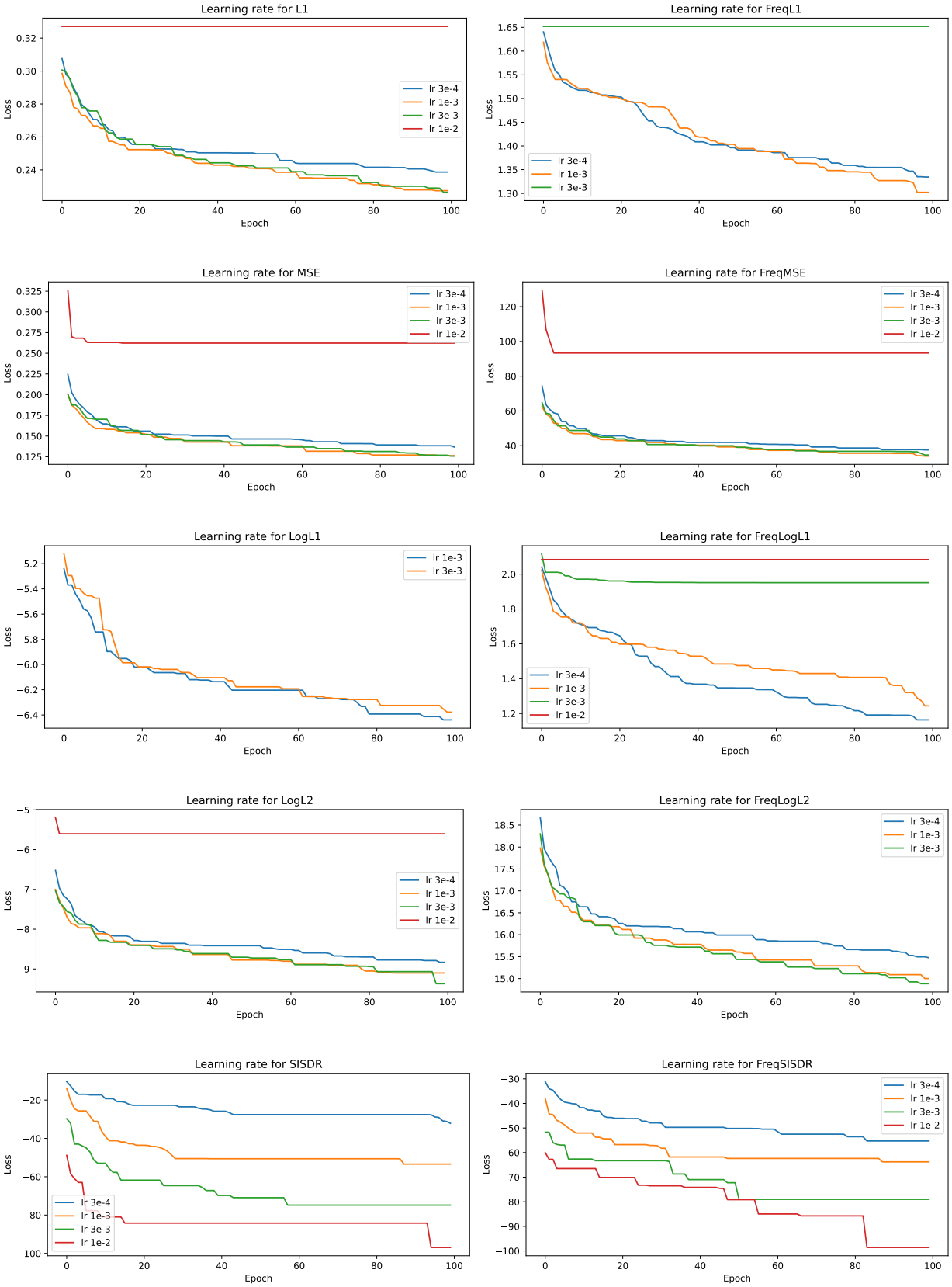
Obj. funkcia	SDR	Bubny	Basy	Ostatné	Vokály	SIR	SAR
$L1_{time}$	4,87	**5,78**	5,11	**3,40**	5,18	**9,66**	5,47
$LogL1_{time}$	4,72	5,48	5,37	3,19	4,84	9,18	5,25
$LogL2_{time}$	**4,92**	5,69	**5,43**	3,33	**5,22**	8,50	**5,76**
$L1_{freq}$	-4,51	-5,19	-2,73	-4,95	-5,17	8,53	5,54
$L2_{freq}$	-4,60	2,79	-5,46	-4,81	-5,34	7,78	4,93

8.7 Koeficient rýchlosti učenia

Na nájdenie najväčšieho koeficientu rýchlosti učenia sa použila zmenšená sieť popísaná v sekcii 8.3. Pri tomto experimente bol zmenšený aj počet epoch a to z hodnoty 180 na 100. Ďalej bol pre tento experiment upravený súbor `train.py` a to tak, aby sa pri trénovaní nepremiešavali dáta a cyklus trénovania bežal len na prvých 10 dávkach. Zaručuje to, že sa každý cyklus trénovania vykonáva na vždy rovnakej malej časti dát, a preto je možné vyhodnotiť rýchlosť konvergencie pre jednoduchšiu úlohu plynúcu z obmedzeného počtu príkladov. Keby sa dáta premiešavali, nemuselo by to byť smerodajné. Zmeny pre tento experiment boli vykonané za účelom rýchleho natrénovanie siete, aby sa dali vzájomne porovnať rôzne koeficienty rýchlosti učenia. Zmeny, ktoré tu boli aplikované sú v súbore `train.py` označené ako ladenie („tuning“).

Na Obrázku 8.1 je možno vidieť grafy ukazujúce konvergenciu hodnôt jednotlivých objektívnych funkcií k minimu v priebehu epoch pre jednotlivé koeficienty rýchlosti učenia. Na grafoch s objektívnou funkciou SISDR v oboch doménach možno vidieť, že sa celková chyba na pomerne dlhých intervaloch epoch nemení, čo môže byť potenciálna súvislosť so zlými výsledkami SDR pre modely s touto funkciou. Výsledné vybrané koeficienty rýchlosti učenia sú nasledovné:

- 3×10^{-3} pre L1, MSE, LogL1, LogL2, SISDR, FreqMSE, FreqLogL2, FreqSISDR
- 1×10^{-3} pre FreqL1, FreqLogL1

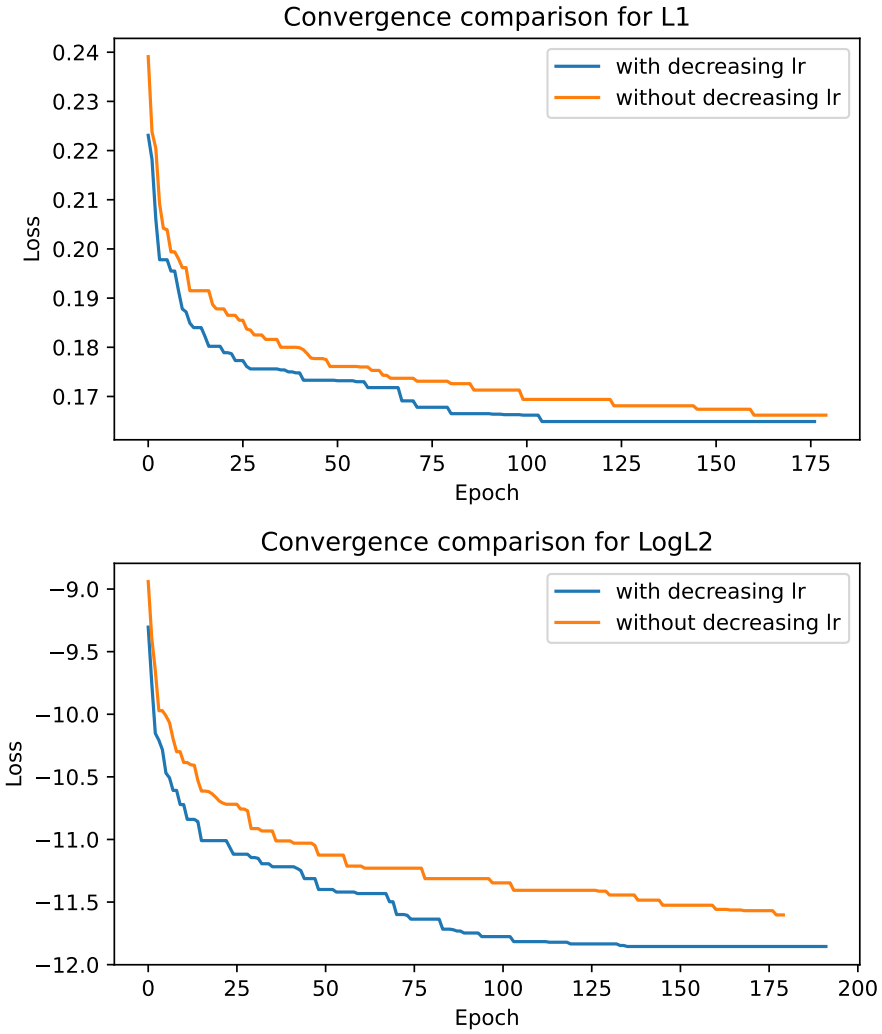


Obr. 8.1: Hľadanie najväčšieho koeficientu rýchlosti učenia pri ktorej celková chyba konverguje v priebehu epoch pre každú objektívnu funkciu.

8.7.1 Zmenšovanie koeficientu rýchlosti učenia

Dôsledkom postupného zmenšovania koeficientu rýchlosti by mala byť rýchlejšia konvergencia k minimu hodnoty objektívnej funkcie a umožnenie poklesu hodnoty do „užšieho údolia“. Z predošlých trénovaní siete som zistil, že ako vhodná konštanta určujúca počet epoch, po ktorých by sa mal koeficient rýchlosti učenia zmenšiť o polovicu, ak sa nezmení najlepšia celková chyba, je hodnota 8. Ďalej aby sa predišlo zbytočnému zmenšovaniu tohto koeficientu na príliš malé hodnoty, kód som upravil tak, aby keď sa koeficient rýchlosti učenia dostane pod hodnotu menšiu ako 1×10^{-6}, tak sa skončí trénovanie v prípade, ak ešte stále nebol dosiahnutý prednastavený počet epoch.

Na Obrázku 8.2 sú porovnané priebehy trénovania (najlepšia celková chyba v priebehu epoch) bez zmenšovania koeficientu rýchlosti učenia a so zmenšovaním. Ako je vidieť, trénovanie s postupným zmenšovaním tohto koeficientu skonvergovalo k minimu rýchlejšie a ešte aj dosiahlo nižšiu hodnotu celkovej chyby ako bez zmenšovania.



Obr. 8.2: Porovnanie konvergencie najlepšej celkovej chyby bez aplikácie zmenšovania koeficientu rýchlosti učenia a s jeho aplikáciou v priebehu epoch.

Ako posledný veľký experiment boli aplikované predošlé zistenia a sieť trénovaná s týmito nastaveniami:

- počet vzoriek zostane predvolený - 441 000
- veľkosť dávky - 16
- počet epoch - 250
- zmenšená štruktúra siete
- počiatočný koeficient rýchlosti učenia nastavený pre každú objektívnu funkciu podľa predošlého zistenia
- postupné zmenšovanie koeficientu rýchlosti učenia

Ďalej pre objektívne funkcie pracujúce vo frekvenčnej doméne bola použitá veľkosť rámu okna Fourierovej transformácie 2048 a vzdialenosť medzi oknami 512. Objektívna funkcia SISDR sa implementovala nanovo, keďže dosahovala až príliš nízke hodnoty SDR. Výsledky experimentu sa nachádzajú v Tabuľke 8.5.

Tabuľka 8.5: Výsledky trénovania siete Conv-TasNet s počiatočným nastavením vyššieho koeficientu rýchlosti učenia s jeho postupným zmenšovaním.

Obj. funkcia	SDR	Bubny	Basy	Ostatné	Vokály	SIR	SAR
L1$_{\text{time}}$	**5,42**	**5,84**	6,08	**3,94**	**5,81**	**11,25**	5,80
L2$_{\text{time}}$	5,33	**5,84**	6,07	3,78	5,61	9,93	6,03
LogL1$_{\text{time}}$	5,30	5,72	**6,46**	3,64	5,38	10,84	5,42
LogL2$_{\text{time}}$	5,40	5,83	6,21	3,78	5,77	9,84	**6,05**
SISDR$_{\text{time}}$	-148,64	-146,34	-146,94	-149,96	-151,33	-5,59	-3,31
L1$_{\text{freq}}$	-5,07	-5,24	-5,45	-4,62	-4,96	10,48	3,09
L2$_{\text{freq}}$	-4,66	-2,80	-5,59	-4,91	-5,35	8,83	4,99
LogL1$_{\text{freq}}$	-3,03	-2,98	-2,83	-2,93	-3,36	9,32	2,73
LogL2$_{\text{freq}}$	-3,04	3,65	-5,57	-4,92	-5,33	8,83	4,38
SISDR$_{\text{freq}}$	-150,09	-148,42	-151,61	-149,45	-150,86	-5,62	-7,53

Ako je vidieť z výsledkov, znova objektívne funkcie L1 a logaritmická L2 v časovej doméne dosahujú najvyššie SDR, tentokrát funkcia L1 dosiahla najlepší výsledok. Funkcie pracujúce vo frekvenčnej doméne opäť dosahujú záporné hodnoty SDR, no pri prehratí separovaných nahrávok neznejú zle. Pri relatívnom porovnaní hodnôt SIR objektívnych funkcií vo frekvenčnej doméne oproti funkciám v časovej doméne je vidieť, že zväčšením okna pre STFT sú výsledky opäť o málo lepšie oproti predošlým experimentom. Objektívna funkcia SISDR v oboch doménach, ktorá bola reimplentovaná nanovo opäť dosahuje príliš nízke hodnoty SDR ako v predošlom behu, avšak tentokrát sú záporné hodnoty aj pre SIR a SAR, čo už je signalizácia zlej implementácie. V separovaných nahrávkach pre túto funkciu nepočuť takmer nič, len silné šumenie.

8.8 Reimplementácia SISDR

Keďže v predošlom experimente dosiahli objektívne funkcie SISDR (v oboch doménach) záporné hodnoty pre všetky vyhodnocovacie metriky, bolo potrebné znovu reimplementovať túto funkciu. Tentokrát bol kód pre SISDR prevzatý resp. inšpirovaný a upravený z *Asteroid* knižnice [37]. Je to knižnica na separáciu zdrojov zvuku založená na *PyTorch*. Dodáva sa so zdrojovým kódom, ktorý podporuje veľké množstvo architektúr sietí, a súborom „receptov“ na reprodukciu niektorých dôležitých experimentov z výskumných prác. V tejto knižnici sa nachádza aj implementácia objektívnej funkcie SISDR.

Prevzatá implementácia je metódou `forward`() triedy `SISDR` v súbore `loss.py`, predošlá implementácia je uložená v metóde `forward_old`(). S touto zmenou bol vykonaný posledný experiment s rovnakou konfiguráciou ako v predošlom experimente na tejto funkcii v časovej doméne. Výsledok tohto experimentu je v Tabuľke 8.6. Ako je vidieť, výsledok dosiahol znova veľmi nízke hodnoty SDR, ale hodnoty SIR a SAR sú kladné. V separovaných nahrávkach sú ale počuť jednotlivé zdroje, aj keď je prítomné pomerne silné prašťanie a aj šumenie. Z dôvodu časovej tiesni sa experiment pre túto funkciu vo frekvenčnej doméne nestihol vykonať, ale je pravdepodobné, že by bol výsledok podobný ako v časovej doméne.

Tabuľka 8.6: Výsledok trénovania siete Conv-TasNet po reimplementácii objektívnej funkcie SISDR v časovej doméne.

SDR	**Bubny**	**Basy**	**Ostatné**	**Vokály**	**SIR**	**SAR**
-149,96	-151,18	-150,77	-148,02	-149,88	7,97	5,15

Kapitola 9

Vyhodnotenie a diskusia

9.1 Objektívne vyhodnotenie

Jednou z úloh tejto bakalárskej práce je porovnanie výsledkov v relevantnej literatúre. *On loss functions and evaluation metrics for music source separation* [15] je kľúčový článok, z ktorého sa vychádzalo a s ktorým sme chceli porovnať trendy. V tomto článku sú vyhodnotené rôzne objektívne funkcie, ale model na ktorom boli prevádzané experimenty je *Open-Unmix* [50]. Myšlienka bola taká, že aj keď ide o iný model, mohli by sa očakávať rovnaké trendy výsledkov pre jednotlivé objektívne funkcie.

Zo spomínaného článku vyšli najlepšie výsledky pre objektívne funkcie (len tie s ktorými sa pracovalo aj v tejto práci) $L2_{freq}, SISDR_{freq}, LogL1_{time}$ a $LogL1_{freq}$. Väčšina bola vo frekvenčnej doméne. Použitá vyhodnocovacia metrika bola SDR. Výsledky z článku možno vidieť v Tabuľke 9.1. V tejto práci však najlepšie výsledky dosahovali modely s objektívnymi funkciami $L1_{time}, LogL2_{time}$, čo je odlišné oproti článku.

Tabuľka 9.1: Výsledky experimentov jednotlivých objektívnych funkcií pre model Open-Unmix [15].

Obj. funkcia	SDR	Bubny	Basy	Ostatné	Vokály
L1$_{\text{time}}$	3,99	4,80	3,53	3,01	4,63
L2$_{\text{time}}$	3,70	4,25	3,06	2,92	4,55
LogL1$_{\text{time}}$	5,34	5,82	5,23	4,35	5,95
LogL2$_{\text{time}}$	5,21	5,81	5,04	4,10	5,88
SISDR$_{\text{time}}$	5,35	5,76	5,06	4,37	6,24
L1$_{\text{freq}}$	4,92	5,58	4,24	3,90	5,95
L2$_{\text{freq}}$	5,53	5,86	5,28	4,57	6,40
LogL1$_{\text{freq}}$	5,53	5,90	5,49	4,44	6,28
LogL2$_{\text{freq}}$	5,42	5,79	5,38	4,36	6,15
SISDR$_{\text{freq}}$	5,61	6,09	5,55	4,54	6,26

Porovnanie výsledkov dopadlo teda tak, že sa nepotvrdili trendy pre jednotlivé objektívne funkcie prezentované v literatúre. To môže mať rôzne príčiny, ako napríklad zmena rovníc výpočtov objektívnych funkcií oproti článku [15] a ďalšia analýza (hlavne objektívnych funkcií vo frekvenčnej doméne) by mohla byť predmetom ďalších experimentov.

9.2 Diskusia

Práca bola založená na experimentovaní s už existujúcou implementáciou neurónovej siete Conv-TasNet. Navrhnuté modifikácie mali za cieľ lepšie porozumieť model resp. aj všeobecne trénovanie neurónových sietí a potencionálne zlepšiť model. Experimentovalo sa s konfiguráciou štruktúry siete, zmenou počtu vzoriek na vstupe, veľkosti dávky, počtu epoch, transformáciou signálu do frekvenčnej domény pre účely počítania objektívnej funkcie, rôznymi objektívnymi funkciami, hľadaním optimálneho koeficientu rýchlosti učenia a postupným zmenšovaním tohto koeficientu. Trénovanie prebiehalo na dátovej sade MUSDB18.

Trénovanie s podobnou konfiguráciou siete (originálnou štruktúrou siete, nie zmenšenou) a nastavením dosiahlo lepšie výsledky (SDR 5,98 a 6,08 dB) ako uvádzané referenčné trénovanie (SDR 5,73 dB). Referenčné trénovanie prebehlo na 8 GPU, zatiaľ čo v tomto prípade bolo trénovanie len na maximálne 4 GPU. Kvôli tomuto „obmedzeniu“ bola zmenšená veľkosť dávky a to spôsobom prepočítania dávky na 1 GPU. Očakávaný výsledok tak mal byť približne rovnaký, ale dosiahnutá hodnota SDR bola lepšia o približne 0,30 dB. Tento dej je pravdepodobne spôsobený tým, že model je z roku 2018, ktorý bol však aktualizovaný, ale výsledky modelu nie.

Zmenšenie štruktúry siete bolo vykonané za účelom skrátenia času potrebného na trénovanie siete. Trénovanie s pôvodnou štruktúrou trvalo približne 8 dní čistého času, čo bolo pomerne veľa na to aby sa stihlo vykonať viacero experimentov. Z tohoto dôvodu bola štruktúra siete zmenšená (presne popísané v sekcii 8.3). Po úprave štruktúry sa trvanie trénovania skrátilo na približne 5 dní čistého času. Toto zmenšenie malo za následok horší výsledok (SDR 5,28 dB), ako aj bolo očakávané.

Transformácia do frekvenčnej domény je implementovaná funkciou `make_freq`, ktorej kód je možný vidieť vo Výpise 8.1. Transformácia je vykonaná len pre účel hodnotenia objektívnych funkcií. Tenzory sa transformujú za pomoci krátkodobej Fourierovej transformácie STFT. Objektívne funkcie vo frekvenčnej doméne dosahovali záporné výsledky SDR, ale výsledky SIR sa postupne zlepšovali počas experimentov v relatívnom porovnaní voči funkciám v časovej doméne, a to bol ako dôsledok spôsobený tým, že sa 2-krát zväčšovalo okno pre Fourierovu transformáciu STFT (pôvodne 512, potom 1024 a 2048).

Ako ďalší experiment bola zmena objektívnej funkcie. Experimentoval som s L1, L2, logaritmickou L1 a L2, a SISDR objektívnymi funkciami v časovej aj frekvenčnej doméne. Pri experimentoch som zistil, že model nekonverguje pri logaritmických funkciách s navrhnutými rovnicami ani pri zmene koeficientu rýchlosti učenia. Po zmenení rovnice už modely konvergovali. Zmena spočívala v tom že vnútorná suma pred logaritmom mala byť pôvodne iba cez vzorky v časovej doméne alebo cez frekvenčný zásobník a rámce vo frekvenčnej doméne, ale bolo nutné túto sumu spraviť aj cez kanály a zdroje, aby model konvergoval (rozpísané v sekcii 8.5). Ako najlepšie dopadli funkcie logaritmická L2 a obyčajná L1 v časovej doméne, ktoré mali porovnateľné hodnoty. Funkcie L2 a logaritmická L1 taktiež v časovej doméne len o trochu zaostávali. Funkcie pracujúce vo frekvenčnej doméne dosahovali záporné výsledky SDR (od cca -5 dB po -0,5 dB), aj keď separované nahrávky nezneli tak zle. Problémom je, že model by sa mal naučiť predpovedať výstupy zarovnané so vzorkami. Po prevedení transformácie za pomoci STFT sa však fáza nepredpovedá presne, a preto tu vzniká problém. Aj napriek tomu funkcia SISDR v oboch doménach dosahovala až príliš nízke hodnoty SDR (-90 dB -150 dB). V separovaných nahrávkach je síce počuť tzv. prašťanie a šumenie, ale nie natoľko aby bola hodnota tak nízka. Presnú príčinu sa nepodarilo zistiť.

Pri trénovaní s menším počtom vzoriek ale väčšou dávkou som zistil, že počet vzoriek na vstupe je dôležitejší parameter trénovania ako veľkosť dávky, pretože po zmenšení dávky takmer 2-násobne a zväčšení dávky 4-násobne boli výsledky trénovania SDR horšie zhruba o 0,40 dB. Z toho možno usúdiť, že veľkosť dávky až tak neovplyvňuje výsledok trénovania ako počet vzoriek na vstupe.

Koeficient rýchlosti učenia ovplyvňuje ako rýchlo daný model konverguje k minimu objektívnej funkcii a či vôbec. Experimentom som zisťoval najväčší optimálny koeficient rýchlosti učenia pre každú implementovanú objektívnu funkciu, pri ktorej model konverguje. Pre každú objektívnu funkciu môže byť tento koeficient iný, ako aj bolo zistené. Pri použití len najväčšieho koeficientu rýchlosti učenia model zo začiatku rýchlejšie konverguje, ale potom sa nedokáže dostať až tak blízko lokálnych miním ako pri menšom koeficiente rýchlosti učenia. Z toho dôvodu bolo zavedené postupné zmenšovanie koeficientu rýchlosti učenia. Pri nastavení najväčšieho optimálneho koeficientu rýchlosti učenia a postupnom zmenšovaní model jednak rýchlejšie konverguje, ale dokáže sa dostať aj bližšie k lokálnym minimám a tak dosahuje celkovo lepší výsledok ako bez tohto použitia. Konkrétne v tomto prípade dosahovali modely trénované s počiatočným nastavením vyššieho koeficientu rýchlosti učenia a postupným zmenšovaním výsledky SDR o približne 0,15 dB lepšie ako bez tohto použitia.

Je nutné spomenúť aj fakt, že uvedené čisté trvania trénovania sú oproti celkovým trvaniam trénovania o dosť menšie. Prvou príčinou bolo niekedy pomerne dlhé čakanie v rade na spustenie výpočtu vo výpočtovom centre. To síce niekedy trvalo menej ako hodinu, ale niekedy aj 8 dní. Priemerné čakanie v rade trvalo 4-12 hodín. Ďalším a hlavným problémom pri trénovaní bolo padanie programu na príkaze `ffmpeg`. Aj po inštalácií iných verzií `ffmpeg` tento problém nezmizol. Toto padanie bolo nedeterministické, raz sa nevyskytlo za celý beh trénovania (málokedy) a inokedy sa vyskytlo hneď krátko po spustení. Z toho dôvodu bolo nutné vždy po páde znovu posielať úlohu do rady na výpočet, a to značne naťahovalo celkový čas trénovania. Tak isto ak bežali viaceré rôzne úlohy (experimenty vo výpočtovom centre) naraz, bolo takmer isté, že aspoň jedna z nich padne na tejto chybe. Z tohto dôvodu väčšinou bežala naraz iba 1 úloha na minimalizáciu týchto padaní. Ďalšou vecou, čo pomohla pri riešení tohto problému, bolo odchytenie tejto chyby v skripte na trénovanie, po ktorom sa beh opätovne spustil. Toto riešenie síce nedokázalo úplne vyriešiť problém (niekedy nekonečne padajúci cyklus), ale v značnej miere ho eliminovalo.

Snahou tejto práce bolo aj zlepšenie výsledkov oproti východziemu modelu. Za východzí model sa dá v tejto práci považovať model so zmenšenou štruktúrou siete a s objektívnou funkciou L1 v časovej doméne (z Tabuľky 8.3). Tento model dosiahol výsledok SDR 5,28 dB. Ako najlepší model so zmenšenou štruktúrou siete z experimentov vyšiel model taktiež s objektívnou funkciou L1 v časovej doméne, ktorý bol ale trénovaný s vyššou počiatočnou hodnotou koeficientu rýchlosti učenia, ktorá sa postupne zmenšovala v priebehu učenia (model je v Tabuľke 8.5). Najlepší model dosiahol výsledok SDR 5,42 dB, čo tvorí 2,65% zlepšenie oproti východziemu modelu. Podobné zlepšenie by sa dalo očakávať aj s pôvodnou štruktúrou siete, čo môže byť predmetom ďalšieho pokračovania výskumu.

Kapitola 10

Záver

Cieľom tejto práce bol návrh a implementácia modifikácií neurónovej siete Conv-TasNet a trénovania za účelom porozumenia modelu a potenciálneho zlepšenia pre úlohu separácie zdrojov hudby. Na túto úlohu bolo nutné použiť výpočtové centrum, kvôli veľkým nárokom na výpočet, konkrétne počítanie na GPU s veľkou RAM pamäťou. Použité výpočtové centrum bolo *MetaCentrum*. Modifikácie boli implementované pomocou knižnice *PyTorch*.

Trénovanie siete s referenčným nastavením siete a pôvodnou štruktúrou dopadlo pozitívne, nakoľko modely dosiahli lepší výsledok ako referenčný. Trénovanie však trvalo pomerne dlho, a preto bola štruktúra siete ďalej zmenšená.

Experimenty ukázali, že modely pre objektívne funkcie L1 a logaritmická L2 v časovej doméne dosahovali najlepšie výsledky spomedzi všetkých objektívnych funkcií. Modely pre funkcie L2 a logaritmická L1 dosiahli o trochu málo horšie výsledky. Výsledky trénovania pre objektívne funkcie vo frekvenčnej doméne dosiahli záporné hodnoty SDR, čo je ako následok transformácie tenzorov do frekvenčnej domény použitím STFT. Trénovanie s objektívnou funkciou SISDR v oboch doménach dosahovalo až príliš nízke hodnoty SDR (-90 dB až -150 dB). V separovaných nahrávkach je síce počuť tzv. prašťanie a šumenie, ale nie natoľko aby bola hodnota tak nízka. Presnú príčinu sa nepodarilo zistiť. Experimentovaním sa tiež zistilo, že modely pri počiatočnom nastavení koeficientu rýchlosti učenia na vyššiu hodnotu a následnom postupnom zmenšovaní dosahovali lepšie výsledky. Spôsobilo to, že modely zo začiatku trénovania rýchlejšie konvergovali a nakoniec sa dostali bližšie k lokálnemu minimu objektívnej funkcie. Tento jav bol očakávaný a tak aj potvrdený. Pre všetky natrénované modely sú dostupné vyseparované nahrávky na priloženom pamäťovom médiu.

Týmito experimentami sa nie len zlepšil výsledok siete, ale aj vedomosti o strojovom učení všeobecne ako aj porozumenie modelu Conv-TasNet. Cieľ tejto práce je tak splnený. Pre budúci vývoj práce by bolo vhodné použiť vyhodnocovaciu metriku SI-SDR, ďalej analyzovať dosiahnuté výsledky (hlavne objektívne funkcie vo frekvenčnej doméne a funkciu SISDR), pri transformácii do frekvenčnej domény spraviť samostatné experimenty pre rôzne veľkosti okien v STFT a skúsiť použiť optimalizátor SGD.

Literatúra

[1] Bahmaninezhad, F., Zhang, S.-X., Xu, Y., Yu, M., Hansen, J. H. L. et al. *A Unified Framework for Speech Separation.* arXiv, 2019. DOI: 10.48550/arXiv.1912.07814.

[2] Bittner, R., Salamon, J., Tierney, M., Mauch, M., Cannam, C. et al. MedleyDB: A Multitrack Dataset for Annotation-Intensive MIR Research. In: *15th International Society for Music Information Retrieval Conference.* Taipei, Taiwan: [b.n.], October 2014.

[3] Bittner, R., Wilkins, J., Yip, H. a Bello, J. MedleyDB 2.0: New Data and a System for Sustainable Data Collection. In: *International Conference on Music Information Retrieval (ISMIR-16).* New York, NY, USA: [b.n.], 2016.

[4] Buduma, N. a Locascio, N. *Fundamentals of Deep Learning: Designing Next-Generation Machine Intelligence Algorithms.* O'Reilly Media, 2017. ISBN 978-1-4919-2558-4.

[5] Cano, E., FitzGerald, D., Liutkus, A., Plumbley, M. D. a Stöter, F.-R. Musical Source Separation: An Introduction. *IEEE Signal Processing Magazine.* 1. vyd. IEEE. 2018, zv. 36, č. 1, s. 31–40.

[6] Chan, T.-S., Yeh, T.-C., Fan, Z.-C., Chen, H.-W., Su, L. et al. Vocal Activity Informed Singing Voice Separation with The IKala Dataset. In: *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* 2015, s. 718–722. DOI: 10.1109/ICASSP.2015.7178063.

[7] Chollet, F. *Xception: Deep Learning with Depthwise Separable Convolutions.* arXiv, 2017. DOI: 10.48550/arXiv.1610.02357.

[8] Collobert, R. a Weston, J. A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning. In: ACM. *Proceedings of the 25th international conference on Machine learning.* New York, NY, USA: ACM, 2008, s. 160–167. ICML '08. DOI: 10.1145/1390156.1390177. ISBN 978-1-60558-205-4.

[9] Delyon, B. Stochastic Approximation with Decreasing Gain: Convergence and Asymptotic Theory. *Unpublished lecture notes, Université de Rennes.* Citeseer. 2000, zv. 26.

[10] Demirel, E., Ahlback, S. a Dixon, S. *Automatic Lyrics Transcription using Dilated Convolutional Neural Networks with Self-Attention.* arXiv, 2020. DOI: 10.48550/arXiv.2007.06486.

[11] Freund, Y. a Schapire, R. E. Large Margin Classification Using The Perceptron Algorithm. *Machine learning*. Springer. 1999, zv. 37, č. 3, s. 277–296.

[12] Fujihara, H., Goto, M., Ogata, J., Komatani, K., Ogata, T. et al. Automatic Synchronization between Lyrics and Music CD Recordings Based on Viterbi Alignment of Segregated Vocal Signals. In: *Eighth IEEE International Symposium on Multimedia (ISM'06)*. 2006, s. 257–264. DOI: 10.1109/ISM.2006.38.

[13] Ghazdali, A., Hakim, A., Laghrib, A., Mamouni, N. a Raghay, S. A New Method for the Extraction of Fetal ECG from the Dependent Abdominal Signals Using Blind Source Separation and Adaptive Noise Cancellation Techniques. *Theoretical Biology and Medical Modelling*. BioMed Central. 2015, zv. 12, č. 1, s. 1–20.

[14] Goodfellow, I., Bengio, Y. a Courville, A. *Deep Learning*. Illustrated. The MIT Press, november 2016. 800 s. ISBN 0262035618.

[15] Gusó, E., Pons, J., Pascual, S. a Serrà, J. *On Loss Functions and Evaluation Metrics for Music Source Separation*. arXiv, 2022. DOI: 10.48550/arXiv.2202.07968.

[16] Haykin, S. *Neural Networks and Learning Machines*. 3. vyd. Pearson, 2009. ISBN 978-0-13-147139-9.

[17] Heitkaemper, J., Jakobeit, D., Boeddeker, C., Drude, L. a Haeb Umbach, R. *Demystifying TasNet: A Dissecting Approach*. arXiv, 2020. DOI: 10.48550/arXiv.1911.08895.

[18] Heittola, T., Klapuri, A. a Virtanen, T. Musical Instrument Recognition in Polyphonic Audio Using Source-Filter Model for Sound Separation. In:. Kobe, Japan: [b.n.], Január 2009, s. 327–332.

[19] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W. et al. Mobilenets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *ArXiv*. 2017. DOI: 10.48550/arXiv.1704.04861.

[20] Hsu, C.-L. a Jang, J.-S. R. On the Improvement of Singing Voice Separation for Monaural Recordings Using the MIR-1K Dataset. *IEEE Transactions on Audio, Speech, and Language Processing*. 1. vyd. 2010, zv. 18, č. 2, s. 310–319. DOI: 10.1109/TASL.2009.2026503.

[21] Hung, Y.-N., Chen, Y.-A. a Yang, Y.-H. *Multitask Learning for Frame-level Instrument Recognition*. arXiv, 2018. DOI: 10.48550/arXiv.1811.01143.

[22] Kaelbling, L. P., Littman, M. L. a Moore, A. W. Reinforcement Learning: A Survey. *Journal of artificial intelligence research*. 1996, zv. 4, s. 237–285.

[23] Kaiser, Ł., Gomez, A. N. a Chollet, F. Depthwise Separable Convolutions for Neural Machine Translation. *ArXiv*. 2017. DOI: 10.48550/arXiv.1706.03059.

[24] Khasanov, D., Tojiyev, M. a Primqulov, O. Gradient descent in machine learning. In: *2021 International Conference on Information Science and Communications Technologies (ICISCT)*. 2021, s. 1–3. DOI: 10.1109/ICISCT52966.2021.9670169.

[25] Lea, C., Vidal, R., Reiter, A. a Hager, G. D. Temporal Convolutional Networks: A Unified Approach to Action Segmentation. In: Springer. *European Conference on Computer Vision*. 2016, s. 47–54.

[26] Liutkus, A., Stöter, F.-R., Rafii, Z., Kitamura, D., Rivet, B. et al. The 2016 Signal Separation Evaluation Campaign. In: Tichavský, P., Babaie Zadeh, M., Michel, O. J. a Thirion Moreau, N., ed. *Latent Variable Analysis and Signal Separation - 12th International Conference, LVA/ICA 2015, Liberec, Czech Republic, August 25-28, 2015, Proceedings*. Cham: Springer International Publishing, 2017, s. 323–332.

[27] Luo, Y. a Mesgarani, N. Conv-TasNet: Surpassing Ideal Time–Frequency Magnitude Masking for Speech Separation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*. 1. vyd. Institute of Electrical and Electronics Engineers (IEEE). aug 2019, zv. 27, č. 8, s. 1256–1266. DOI: 10.1109/taslp.2019.2915167.

[28] Makino, S. *Audio Source Separation*. 1. vyd. Springer Cham, March 2018. ISBN 978-3-319-73031-8.

[29] Makino, S., Sawada, H. a Lee, T.-W. Blind Speech Separation. In:. 1. vyd. Springer Dordrecht, September 2007, s. 432. DOI: 10.1007/978-1-4020-6479-1. ISBN 978-1-4020-6479-1.

[30] Manilow, E., Seetharman, P. a Salamon, J. *Open Source Tools & Data for Music Source Separation*. 1. vyd. October 2020. Dostupné z: https://source-separation.github.io/tutorial.

[31] Manilow, E., Wichern, G., Seetharaman, P. a Le Roux, J. Cutting Music Source Separation Some Slakh: A Dataset to Study the Impact of Training Data Quality and Quantity. In: IEEE. *Proc. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. 2019.

[32] Medsker, L. R. a Jain, L. Recurrent Neural Networks. *Design and Applications*. 2001, zv. 5, s. 64–67.

[33] Mesaros, A. a Virtanen, T. Automatic Recognition of Lyrics in Singing. *EURASIP Journal on Audio, Speech, and Music Processing*. 1. vyd. 2010, zv. 2010, č. 1, s. 546047.

[34] Murphy, K. P. *Machine Learning: A Probabilistic Perspective*. Cambridge: MIT Press, 2012. 247 s. ISBN 978-0-262-01802-9.

[35] Nugraha, A. A., Liutkus, A. a Vincent, E. Multichannel Audio Source Separation with Deep Neural Networks. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*. IEEE. 2016, zv. 24, č. 9, s. 1652–1664.

[36] Oord, A. van den, Dieleman, S., Zen, H., Simonyan, K., Vinyals, O. et al. *WaveNet: A Generative Model for Raw Audio*. arXiv, 2016. DOI: 10.48550/arXiv.1609.03499.

[37] Pariente, M., Cornell, S., Cosentino, J., Sivasankaran, S., Tzinis, E. et al. Asteroid: The PyTorch-Based Audio Source Separation Toolkit for Researchers. In:. Október 2020, s. 2637–2641. DOI: 10.21437/Interspeech.2020-1673.

[38] Patterson, J. a Gibson, A. Understanding Learning Rates. In: *Deep Learning: A Practitioner's Approach*. O'Reilly Media, 2017, s. 258–263. ISBN 978-1-4919-1425-0.

[39] Plumbley, M. D., Abdallah, S. A., Bello, J. P., Davies, M. E., Monti, G. et al. Automatic Music Transcription and Audio Source Separation. *Cybernetics & Systems*. 1. vyd. 2002, zv. 33, č. 6, s. 603–627.

[40] Rafii, Z., Liutkus, A., Stöter, F.-R., Mimilakis, S. I. a Bittner, R. *The MUSDB18 Corpus for Music Separation*. December 2017 [cit. 2023-2-2]. DOI: 10.5281/zenodo.1117372. Dostupné z: https://doi.org/10.5281/zenodo.1117372.

[41] Reed, R. a II, R. J. M. *Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks*. Illustrated. Bradford Books, február 1999. 358 s. ISBN 0262527014.

[42] Ribeiro, A. H., Tiels, K., Aguirre, L. A. a Schön, T. Beyond exploding and vanishing gradients: analysing RNN training using attractors and smoothness. In: Chiappa, S. a Calandra, R., ed. *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*. PMLR, 26–28 Aug 2020, sv. 108, s. 2370–2380. Proceedings of Machine Learning Research.

[43] Rosen, S. a Howell, P. *Signals and Systems for Speech and Hearing: Second Edition*. Leiden, The Netherlands: Brill, 1990. ISBN 978-18-48-55226-5.

[44] Rosenblatt, F. *The Perceptron–a Perceiving and Recognizing Automaton*. Cornell Aeronautical Laboratory, 1957.

[45] Ruder, S. *An Overview of Gradient Descent Optimization Algorithms*. arXiv, 2017. DOI: 10.48550/arXiv.1609.04747.

[46] Sharma, B., Das, R. K. a Li, H. On The Importance of Audio-source Separation for Singer Identification in Polyphonic Music. In:. 2019, s. 2020–2024. DOI: 10.21437/Interspeech.2019-1925.

[47] Silva, I. N. da, Hernane Spatti, D., Andrade Flauzino, R., Liboni, L. H. B. a Reis Alves, S. F. dos. Artificial Neural Network Architectures and Training Processes. In: *Artificial Neural Networks : A Practical Course*. Cham: Springer International Publishing, 2017, s. 21–28. DOI: 10.1007/978-3-319-43162-8. ISBN 978-3-319-43162-8. Dostupné z: https://doi.org/10.1007/978-3-319-43162-8.

[48] Simpson, A. J. R. *Deep Transform: Cocktail Party Source Separation via Complex Convolution in a Deep Neural Network*. arXiv, 2015. DOI: 10.48550/arXiv.1809.07454.

[49] Smith, J. O. Spectral Audio Signal Processing. In:. W3K Publishing, 2011. ISBN 978-0-9745607-3-1. Dostupné z: https://ccrma.stanford.edu/~jos/sasp/.

[50] Stöter, F.-R., Liutkus, A., Ito, N. a Nakamura, Y. Open-Unmix - A Reference Implementation for Music Source Separation. *Journal of Open Source Software*. The Open Journal. 2019, zv. 4, č. 41, s. 1667. DOI: 10.21105/joss.01667.

[51] Stöter, F.-R., Liutkus, A. a Ito, N. *The 2018 Signal Separation Evaluation Campaign*. arXiv, 2018. DOI: 10.48550/arXiv.1804.06267.

[52] Vincent, E., Gribonval, R. a Fevotte, C. Performance Measurement in Blind Audio Source Separation. *IEEE Transactions on Audio, Speech, and Language Processing*. 2006, zv. 14, č. 4, s. 1462–1469. DOI: 10.1109/TSA.2005.858005.

[53] Wei, Y. a Zhang, Q. Common Waveform Analysis: A New And Practical Generalization of Fourier Analysis. In:. Springer US, 2000. The International Series on Asian Studies in Computer and Information Science. ISBN 9780792379058.

[54] Wu, Y., Gardner, J., Manilow, E., Simon, I., Hawthorne, C. et al. The Chamber Ensemble Generator: Limitless High-Quality MIR Data via Generative Modeling. *ArXiv*. 2022. DOI: 10.48550/arXiv.2209.14458.

[55] Xian chuan, Y., Jin dong, X., Dan, H. a Hai hua, X. A New Blind Image Source Separation Algorithm Based on Feedback Sparse Component Analysis. *Signal Processing*. 2013, zv. 93, č. 1, s. 288–296. DOI: 10.1016/j.sigpro.2012.08.010. ISSN 0165-1684.

Príloha A

Konfigurácie siete

Predvolené hodnoty hyper-parametrov siete Conv-TasNet a konfigurácie trénovania sú nasledovné:

- 180 – počet epoch
- 2 – opakovanie tréningového cyklu vrámci 1 epochy
- 441 000 – počet vzoriek na vstupe
- 44 100 – vzorkovacia frekvencia
- 2 (stereo) – audio kanály
- 42 – „seed“
- 64 – velkosť dávky („batch“)
- 3×10^{-4} – koeficient rýchlosti učenia
- 0,2 – pravdepodobnosť zmeny tempa/výšky tónu
- 12 – maximálna zmena relatívneho tempa v percentách

Tieto hodnoty sú uvedené v súbore `parser.py`.

Príloha B

Obsah priloženého úložiska

Priložené pamäťové médium obsahuje:

```
/
├── README.md ... súbor s informáciami o tejto práci,
│                návod na inštaláciu a spustenie
├── __main__.py ... súbor potrebný na nahradenie pôvodného
│                   po stiahnutí siete
├── train.py ... súbor potrebný na nahradenie pôvodného
│                po stiahnutí siete
├── parser.py ... súbor potrebný na nahradenie pôvodného
│                po stiahnutí siete
├── loss.py ... súbor potrebný na vloženie
│              po stiahnutí siete
├── separated.zip ... archív, ktorý obsahuje
│                    vyseparované nahrávky
│                    modelmi z experimentov
├── text_source.zip ... archív zdrojových súborov
│                      pre text tejto práce
└── BP_Music_source_separation.pdf ... výsledná (táto) práca
```