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 MLUVČÍCH V ČASOVÉ DOMÉNĚ

TIME DOMAIN AUDIO SEPARATION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **JIŘÍ PEŠKA**
AUTHOR

VEDOUCÍ PRÁCE **ing. KATEŘINA ŽMOLÍKOVÁ,**
SUPERVISOR

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2019/2020

Zadání bakalářské práce

Student: **Peška Jiří**
Program: Informační technologie
Název: **Separace mluvčích v časové doméně pomocí neuronové sítě**
Time-Domain Neural Network Based Speaker Separation
Kategorie: Zpracování řeči a přirozeného jazyka

Zadání:

1. Seznamte se s problémem separace mluvčích pomocí neuronových sítí.
2. Seznamte se s metodou TasNet pro jednokanálovou separaci signálu v časové doméně.
3. Implementujte danou metodu s využitím vhodného toolkitu (např. PyTorch, Keras).
4. Otestujte systém na vhodném datasetu. Zaměřte se na vyhodnocení vlivu velikosti sítě na přesnost.
5. Navrhněte a diskutujte možné zlepšení použité metody.

Literatura:

- Luo, Yi, and Nima Mesgarani. "Conv-TasNet: Surpassing Ideal Time-Frequency Magnitude Masking for Speech Separation." *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 27.8 (2019): 1256-1266.
- dle doporučení vedoucího

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

Vedoucí práce: **Žmolíková Kateřina, Ing.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 31. července 2020
Datum schválení: 5. listopadu 2019

Abstrakt

Práce se zabývá využitím konvolučních neuronových sítí pro automatickou separaci mluvčích v akustickém prostředí. Cílem je implementovat neuronovou síť podle architektury TasNet za použití frameworku PyTorch, natrénovat síť s různými hodnotami hyperparametrů a porovnat kvalitu separací vzhledem k velikosti sítě.

Architektura oproti dosavadním metodám, které převáděly vstupní směs do časově–frekvenční reprezentace, používá konvoluční autoenkodér, který vstupní směs převádí do nezáporné reprezentace, která je optimalizovaná pro extrakci jednotlivých mluvčích. Samotné separace je docíleno aplikací masek, které jsou odhadnuty v separačním modulu. Modul tvoří opakující se posloupnost konvolučních bloků se zvyšující se dilatací, která napomáhá k modelování časových závislostí ve zpracovávané směsi.

K vyhodnocení přesnosti byly použity metriky signal to distortion ratio (SDR), která určuje poměr zastoupení rušení a cílového zvuku v nahrávce, dále perceptual evaluation of speech quality (PESQ) a short–time objective intelligibility (STOI). Trénování a vyhodnocování proběhlo na množině směsí dvou mluvčích, která byla vygenerována náhodnou kombinací mluvčích z Wall Street Journal datasetu (WSJ0). Natrénováním několika modelů s různými hodnotami hyperparametrů bylo možno pozorovat závislost mezi velikostí sítě a hodnotou SDR. Zatímco menší síť dosahovala, po 60 epochách trénování, přesnosti 10.8 dB, větší síť dosahovala až 12.71 dB.

Abstract

Thesis is about the usage of a convolutional neural networks for automatic speech separation in acoustic environment. The goal is to implement the neural network by following a TasNet architecture in PyTorch framework, train this network with various values of a hyper parameters and to compare the quality of separations based on the size of network.

In contrast to older architectures that transformed an input mixture into a time–frequency representation, this architecture uses a convolutional autoencoder, which transforms input mixture into a non–negative representation optimized for a speaker extraction. Separation is achieved by applying the masks, which are estimated in the separation module. This module consists of a stacked convolutional blocks with increasing dilation, which helps with modelling of the long–term time dependencies in processed speech.

Evaluation of the precision of the network is measured by a signal to distortion (SDR) metric, which defines ratio of distortion and target sound in the audio, by perceptual evaluation of speech quality (PESQ) and the short–time objective intelligibility (STOI). The set of mixtures of two speakers, that was generated by combining various speakers from Wall Street Journal dataset (WSJ0) has been used for training and evaluation. Trained models with various values of hyper parameters enabled to observe the dependency between the size of the network and SDR value. While smaller network after 60 epochs of training reached 10.8 dB of accuracy, bigger network reached 12.71 dB.

Klíčová slova

neuronové sítě, zpracování řeči, konvoluční neuronová síť, autoenkodér, separace mluvčích, strojové učení, tasnet, feed forward, hluboké učení

Keywords

artificial neural networks, speech processing, convolutional neural networks, autoencoder, speech separation, machine learning, tasnet, feed forward, deep learning

Citace

PEŠKA, Jiří. *Separace mluvčích v časové doméně*. Brno, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce ing. Kateřina Žmolíková,

Separace mluvčích v časové doméně

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Kateřiny Žmolíkové. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Jiří Peška
27. července 2020

Poděkování

Chtěl bych velmi poděkovat Ing. Kateřině Žmolíkové za její trpělivost, obrovskou ochotu a nedocenitelné rady, které mi poskytla při tvorbě této bakalářské práce.

Obsah

Kapitola 1

Úvod

Zpracování řeči hraje v dnešní době důležitou roli v mnoha rozličných oborech. Mezi jeden z hlavních úkolů bezesporu patří separace zdrojů v zaznamenaném signálu, který může být složen ze signálů N mluvčích, ale i nechtěného hluku okolí. Vyřešení problému je předpoklad k dalším úkonům jako identifikace konkrétního mluvčího nebo třeba přepis konverzace na text. Se stále se zrychlujícím vývojem počítačů a s jejich zvyšujícím se výkonem se do popředí dostávají metody zpracování řeči založené na neuronových sítích, které v mnoha ohledech předčily ostatní algoritmy strojového učení.

Separace mluvčích v časové doméně dosahuje mimořádných výsledků v porovnání s dosavadními metodami založenými na převodu signálu z časové domény do frekvenční domény pomocí algoritmu STFT (Short–Time Fourier Transform). Taková reprezentace signálu není optimalizovaná pro separaci řeči a nemusí pro tento úkol být optimální. V architektuře, která je navržena v referenční studii s názvem **TasNet: Surpassing Ideal Time-Frequency Masking for Speech Separation** [24], je vstupní signál převeden do nezáporné reprezentace, která je optimální pro extrakci jednotlivých mluvčích. Silnou stránkou systému je hluboká architektura sítě, která lépe modeluje dlouhodobé závislosti v signálu. Zároveň se ale musí vypořádat s problémy, které hluboké neuronové sítě mohou přinášet.

Úkolem této práce je nastudovat si problematiku neuronových sítí a jejich základní principy, seznámit se problémem separace mluvčích pomocí neuronových sítí a následně implementovat síť podle architektury TasNet pro separaci mluvčích v časové doméně, která byla navržena a popsána ve studii [24]. Poté tuto neuronovou síť natrénovat s různými kombinacemi hodnot hyperparametrů, které ovlivňují velikost sítě a její vlastnosti, a nakonec porovnat přesnost a kvalitu separace mezi jednotlivými, různě velkými sítěmi a s výsledky, kterých bylo dosaženo ve studii. Přesnost a kvalita separace bude vypočítána pomocí metrik určených k hodnocení kvality řečového signálu. Sítě budou testovány a vyhodnocovány na testovací množině jednokanálových směsí dvou mluvčích.

Text je rozdělen na několik kapitol, které postupně pokrývají různé logické celky. V kapitole 2 je popsán problém separace mluvčích a dosavadní přístupy k jeho řešení. Kapitola 3 se zabývá základními vlastnostmi a principy neuronových sítí. Kapitola 4 je věnována architektuře TasNet, kterou se moje práce zabývá. Dále kapitola 5 popisuje implementaci sítě a pomocných nástrojů a nakonec v kapitole 6 jsou experimenty a vyhodnocování přesnosti sítě.

Kapitola 2

Separace mluvčích

Živočichové mají vrozenou schopnost zaměřit se na jeden konkrétní zvuk, zatímco všechny ostatní dokážou potlačit (efekt selektivní pozornosti) [9]. Naučit ale této dovednosti počítač se ukázalo jako obtížný úkol, který je překážkou automatic speech recognition (ASR) systémům v rozsáhlejším nasazení do běžného života. Tento problém se nazývá separace mluvčích. Vyřešení tohoto problému, tedy separování zdrojů, umožňuje širokou aplikaci v oblasti zpracování řeči, jako přepis konverzací na text, ovládání počítače hlasem, používání hlasových asistentů či identifikace konkrétního mluvčího v prostředí s mnoha současně mluvícími lidmi.

2.1 Problém separace mluvčích

V běžném prostředí se přes sebe prolínají různé zvuky, jako je hudba, směs mnoha mluvčích či další akustické jevy. Problém separace mluvčích, neboli koktejl párty problém, si dává za úkol separovat tuto směs zvukových signálů. To umožní se soustředit na jednotlivé signály bez přítomnosti ostatních a využít je tak na další analýzu, jako třeba identifikace řečníka a vlastností jeho hlasu či přepis promluvy na text. Ostatní signály jsou na separované nahrávce potlačeny [29].

Problém separace mluvčích na jednokanálové nahrávce lze definovat jako odhad C zdrojů mluvčích $s_1(t), \ldots, s_c(t) \in \mathbb{R}^{1\times T}$ na diskrétním signálu směsi $x(t) \in \mathbb{R}^{1\times T}$, kde T je délka nahrávky a kde

$$x(t) = \sum_{i=1}^{C} s_i(t) \tag{2.1}$$

Cílem je odhadnout $s_i(t), i = 1, \ldots, C$ ze signálu směsi $x(t)$.

Pro vyřešení problému je potřeba se zaměřit na dvě oblasti. Na separaci mluvčího ze signálu směsi mluvčích, což je suma všech jednotlivých mluvčích a zvuků a na zaměření a udržení pozornosti na konkrétního mluvčího, a možnost přepínat mezi jednotlivými zdroji pozornost. Úspěšnost v jedné oblasti zvyšuje úspěšnost oblasti druhé [29]. Algoritmy pro separaci mluvčích berou v potaz obě oblasti s cílem dosáhnout co nejlepších výsledků. Kvalita separace se hodnotí pomocí metrik, které jsou popsány později.

2.2 Metody pro separaci jednokanálových nahrávek

Pro separaci zdrojových signálů z nahrávek směsí existuje mnoho algoritmů založených na zpracování signálů, z nichž asi nejznámější jsou computational auditory scene analysis (CASA) [7] a non–negative matrix factorization (NMF) [23], [38], používané pro separaci jednokanálových nahrávek. V posledních letech se do popředí dostaly přístupy založené na hlubokých neuronových sítích, které předčily svou přesností a výkonností dosavadní algoritmy. Pro separaci vícekanálových nahrávek se používají metody beamforming [15] nebo state–of–the–art multi–channel blind source separation [26], které ale jsou nad rámec této práce. Tato kapitola shrnuje některé metody pro řešení separace, které jsou zmíněny v článku [29].

2.2.1 Computational auditory scene analysis

Metoda computational auditory scene analysis (CASA) je založena na procesu v lidském mozku, který separuje zdroje ze směsi mluvčích a simuluje vysokoúrovňové chování lidského sluchu. Pro separaci jsou většinou manuálně navrhnuta segmentační pravidla pro operování nad nízkoúrovňovými příznaky k odhadu časově frekvenční (T–F) masky, která izoluje komponenty signálu jednotlivých mluvčích a následně je použita pro rekonstrukci všech zdrojů.

Přestože tato metoda byla i nadále rozvíjena, tak má mnoho nevýhod jako špatnou generalizaci v důsledku manuálního vytváření pravidel, nemožnost automatického učení z dat nebo nemožnost použití na separaci jiných zdrojů než je lidská řeč, které značně omezují její použití v mnoha reálných případech.

2.2.2 Non–negative matrix factorization

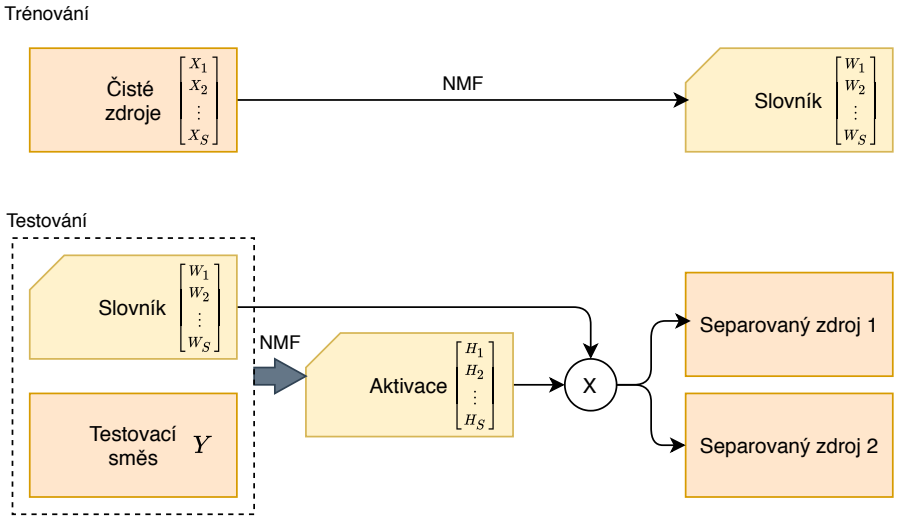
Non-negative matrix factorization (NMF) patří mezi daty řízené metody a je založena na předpokladu, že struktura spektrogramu může být reprezentována malým počtem bází.

Pro NMF platí

$$Y = \sum s\mathbf{W}_s\mathbf{H}_s \tag{2.2}$$

kde každý zdroj s je modelován aproximací nízkého řádu nezápornými maticemi $\mathbf{W}_s$, která reprezentuje slovník a $\mathbf{H}_s$, která reprezentuje aktivační funkci. Ty jsou sečteny a dohromady tvoří výslednou směs Y.

Metoda prochází trénovací a testovací fází. V trénovací fázi je každý zdroj dekomponován a mapován na množinu bází a aktivací a tím je zformován slovník $\mathbf{W}$ pro tento zdroj. V testovací fázi jsou naučené slovníky jednotlivých zdrojů s spojeny do jednoho, který dále není upravován. Následuje optimalizace aktivační matice $\mathbf{H}$ pro každý zdroj. Každý zdroj je následně ze směsi rekonstruován s použitím výsledných bází a aktivací.



Obrázek 2.1: Znázornění trénovací a testovací fáze. Obrázek byl inspirován [29].

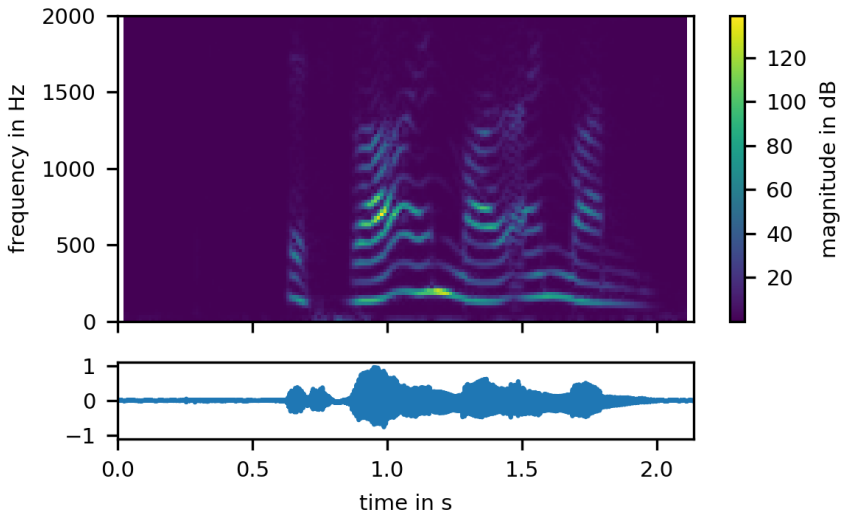
Stejně jako metoda CASA, i tato má mnoho nevýhod, které omezují její použití pro řešení reálných problémů separace.

2.2.3 Hluboké neuronové sítě

Nejlépe si při řešení problému separace mluvčích vedou postupy založené na hlubokém učení a neuronových sítích. Takové modely jsou nejefektivnější, jsou–li definovány jako problémy učení s učitelem. Podle definice 2.1, je cílem separace rekonstruovat původní zdroj signálu.

Metody většinou převáděly směs do časově–frekvenční (T–F) reprezentace znázorněné na obrázku 2.2 pomocí short–time Fourier transform (STFT). Tato reprezentace $Y(t, f)$ sloužila pro rekonstrukci každého zdroje $X_s(t, f)$ v T–F doméně pro každý časový bod t a frekvenci f. Vzhledem k nekonečnému počtu kombinací zdrojových signálů, ze kterých by se dala poskládat původní směs, je nutné naučit model vlastnosti signálů z trénovacího datasetu a odstranit tak nemožné kombinace [29].

Takový problém lze formulovat jako mnoha–třídový regresní problém. Máme–li vstupní příznak $Y(t, f)$ směsi mluvčích, model se bude učit predikovat individuální tok příznaků $X_s(t, f)$ pomocí vhodného trénovacího datasetu. Později se ukázalo, že se lepších výsledků dosáhne, bude–li model odhadovat množinu masek $M_s(t, f)$ pro každý ze zdrojů, místo přímého odhadu spektra $X_s(t, f)$ a ty pak použít k rekonstrukci spektra $X_s(t, f)$ jako $X_s = M_s \circ Y$, kde $\circ$ je násobení po prvcích. Masky jsou invariantní vůči proměnným vlastnostem vstupních dat a jejich správný odhad je důležitým faktorem pro kvalitu separace [29].



Obrázek 2.2: Ukázka nahrávky řeči a korespondujícího spektrogramu, na němž jsou zobrazeny frekvence (osa y) z nahrávky řeči v čase (osa x). Zvuk se na spektrogram (T–F reprezentaci) převádí pomocí operace short–time Fourier transform (STFT). Obrázek převzat z [3].

Vzniklo mnoho dalších přístupů založených na hlubokém učení. Jeden z nich ukázal, že pro separaci mluvčích není T–F doména optimální. Architektura TasNet [24], na kterou se tato práce zaměřuje a je popsána v kapitole 4, převádí vstupní směs na nezápornou reprezentaci a separaci provádí v časové doméně s velmi dobrými výsledky.

2.3 Vyhodnocovací metriky

Pro vyhodnocení kvality separace se používají různé metriky. Jejich volba závisí na řešeném problému. Jejich výpočet probíhá na testovacím datasetu. Nejčastěji používané metriky pro měření kvality separace jsou metriky pro vyhodnocování slepé separace zdrojů (BSS), dále short-time objective intelligibility (STOI) a procentuální vyhodnocení kvality řeči (PESQ). Metriky pro BSS jsou source–to–distortion ratio (SDR), signal–to–inference ratio (SIR) a signal–to–artifacts ratio (SAR).

2.3.1 Source to distortion ratio

SDR [36] je jedna z nejznámějších metrik pro měření kvality separace mluvčích. Hodnota je udávána v decibelech (dB). Na výstupu separačního algoritmu se může nacházet rušení, které lze kategorizovat jako inference a artefakty.

Inference odkazuje na přetrvávající podíl druhého zdroje na separované nahrávce prvního zdroje. SIR je tedy metrika zachycující schopnost algoritmu odstranit ostatní zdroje a zachovat pouze ten požadovaný.

Během separace může algoritmus do separovaných nahrávek zanést artefakty, které se mohou projevovat jako rušení či zvuky, které na původní nahrávce nebyly. SAR je metrika zachycující schopnost algoritmu separovat nahrávky bez toho, aby do nich zanesl nechtěné artefakty [35].

Metriky SIR a SAR lze zkombinovat do jediné metriky SDR, která udává celkovou kvalitu separace.

Hodnota SDR se vypočítá jako

$$SDR = 10 * log_{10} \frac{\|s_{target}\|^2}{\|e_{interf} + e_{noise} + e_{artif}\|^2} \tag{2.3}$$

Hodnoty e_{interf}, e_{noise} a e_{artif} udávají chybu inference, hluku a artefaktů. $s_{target} = f(s_j)$ je verze původního zdroje s_j, který je obohacen o dovolenou hladinou rušení $f \in \mathcal{F}$. Metrika je počítána pro každý odhadnutý zdroj $\hat{s_j}$ tak, že ten je porovnán s referenčním zdrojem s_j. V prvním kroku je dekomponován zdroj $\hat{s_j}$ jako součet $s_{target} + e_{interf} + e_{noise} + e_{artif}$. Ve druhém kroku jsou vypočítány poměry energie, které jsou potřeba při vyhodnocení relativního zastoupení těchto čtyř složek. Je–li při výpočtu porovnáván $\hat{s_j}$ se všemi ostatními zdroji s'_j, tak je jako finální vybrána taková hodnota, která během porovnávání dávala nejlepší výsledek. Porovnávat lze buď celý signál, nebo pouze jeho segmenty [36]. V této práci je pro výpočet SDR používána knihovna `mir_eval` [30].

2.3.2 Perceptual evaluation of speech quality

Procentuální vyhodnocení kvality řeči (PESQ) [31] je standardizovaná objektivní metoda pro vyhodnocování kvality řeči na end–to–end síti. PESQ byla vyvinuta pro předpověď subjektivního skóre *mean opinion score* (MOS), kde posluchač hodnotil kvalitu připojení (řeči) jednou z pěti možností („bad“, „poor“, „fair“, „good“, „excellent“). Těmto možnostem odpovídaly číselné hodnoty 1 až 5, které pak byly zprůměrovány, což nakonec byla finální hodnota MOS.

Hodnota PESQ se pohybuje od -0.5 do 4.5 s tím, že vyšší hodnota znamená lepší procentuální kvalitu [19]. Hodnota je určena na základě porovnání originálního signálu a přeneseného signálu skrze komunikační systém. Lze ho ale aplikovat i pro vyhodnocení separace mluvčích, kde referenční nahrávka je porovnána s nahrávkou, která byla vyprodukována neuronovou sítí.

2.3.3 Short–time objective intelligibility

Short–time objective intelligibility (STOI) [33] je metrika založená na hodnotě korelačního koeficientu mezi časově–frekvenčním spektrem referenčního a odhadnutého signálu řeči v krátkých překrývajících se segmentech [34]. Spektrum čisté a separované nahrávky je získáno pomocí short–time Fourier transform (STFT).

Oba signály jsou dekomponovány pomocí $\frac{1}{3}$–oktávových filtrů pomocí STFT, následně segmentovány na krátké úseky s 50% překrytím, normalizovány, klipnuty a nakonec porovnány průměry hodnot korelačních koeficientů [33]. Hodnota STOI se pohybuje v rozmezí 0 a 1, korespondující s procentuální úspěšností, což znamená, že čím vyšší hodnota, tím lepší výsledek.

Kapitola 3

Neuronové sítě

V dnešní době zažívají neuronové sítě díky výkonnosti počítačů velký rozmach. Jejich využití prostupuje skrze mnohé vědní obory a dokáží řešit celou řadu problémů, ve kterých dosahují výborných výsledků, které zdaleka předčily dosavadní postupy.

Neuronové sítě (*artificial neural networks*) jsou výpočetní model, který je inspirovaný strukturou lidského mozku, ve kterém je obrovské množství propojených a komunikujících neuronů. Ty se skládají ze vstupních dendridů, výstupních axonů a samotného těla neuronu. Na základě vnitřního potenciálu a vstupních hodnot je po přesažení prahové hodnoty neuron vybuzen a je vyslán signál na výstupní axon. Signál je nakonec předán dalším neuronům skrze jejich vstupní dendridy [18].

Účelem neuronové sítě je naučit se plnit zadanou úlohu. Rozdíl oproti běžným algoritmům je ale ten, že způsob, jakým síť má problém řešit, není explicitně naprogramován, ale je postupně naučen. Základní způsoby učení jsou s učitelem (*supervised*) a bez učitele (*unsupervised*).

Učení s učitelem, pod které spadá i tato práce, spočívá v mapování vstupních dat na data výstupní na základě vzorových příkladů dvojic vstup–výstup. Množině takových dvojic se říká trénovací dataset.

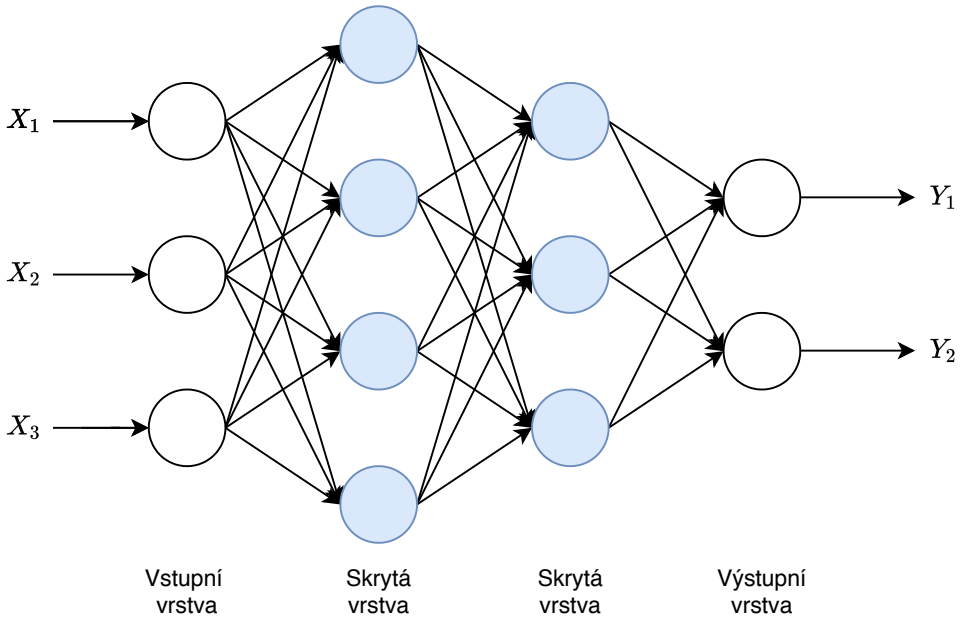
Mezi problémy, které se dají řešit neuronovými sítěmi patří klasifikační a regresní problémy. Konkrétní příklad z oblasti klasifikace může být rozpoznávání objektů na obraze, psaného písma nebo detekce obličejů na videu, ale i mnohé aplikace ve zpracování řeči. Problém separace mluvčích, který spadá do zpracování řeči, se dá klasifikovat jako mnoho–třídový regresní problém.

Upravovat samotnou strukturu a chování neuronové sítě lze pomocí jejích hyperparametrů, což jsou parametry určující nastavení neuronové sítě a trénovacího algoritmu. Tyto parametry musí být určeny před začátkem trénování a většinu nelze později měnit. Mohou určovat kapacitu modelu, velikost záběrného pole, velikost filtrů, ale i regulovat samotný proces učení nastavením počtu epoch, po které se model má učit, nebo také počet dat předaný v jedné dávce (tzv. *minibatches*) během učení sítě.

3.1 Organizace dopředných sítí

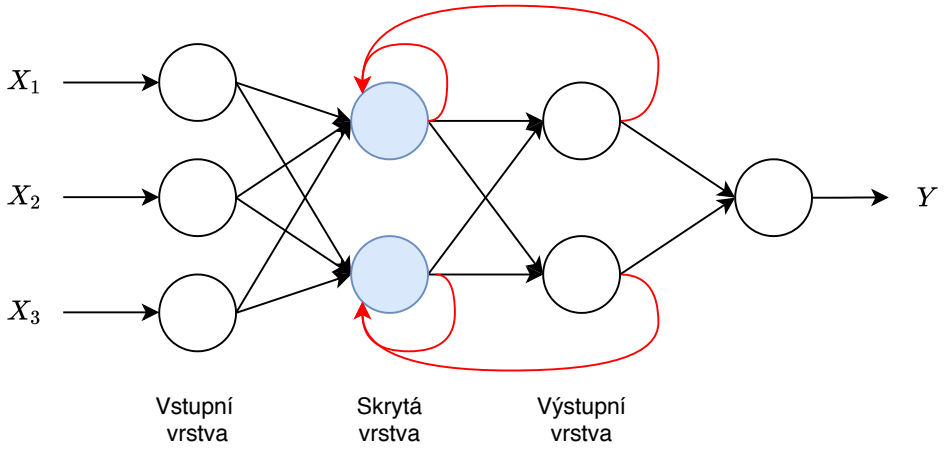
Dopředná neuronová síť (feed forward network, či multi layer perceptron – MLP), zobrazená na obrázku 3.1, je typ umělých neuronových sítí, kde se nevyskytují cykly ve výpočetním grafu, tedy zpětná propojení vrstev. Informace se pohybuje pouze jedním směrem,

od vstupní vrstvy přes skryté vrstvy až po vrstvu výstupní. Sítě, které obsahují cykly, se nazývají rekurentní a pro představu je taková síť znázorněna na obrázku 3.2.



Obrázek 3.1: Schéma dopředné neuronové sítě, která má 2 skryté vrstvy. Šipky směřují pouze jedním směrem.

Struktura neuronové sítě je organizována do vrstev, které se skládají z neuronů. Dopředná síť je tvořena třemi typy vrstev (viz obrázky 3.2 a 3.1). Vstupní vrstva slouží k předání hodnot do sítě, ale nijak tyto hodnoty nemodifikuje. Nezměněné jsou zkopírovány do první skryté vrstvy. Následují skryté vrstvy, z nichž poslední je napojena na výstupní vrstvu. Hodnoty na jejím výstupu mohou představovat třídy, do kterých má být klasifikován vstup v případě klasifikátorů, nebo predikce hodnot na základě vstupních dat v případě regrese. S počtem jednotlivých vrstev souvisí pojem hloubka sítě, která je rovna počtu všech vrstev neuronové sítě od vstupní až po výstupní vrstvu. Pojem „hluboká neuronová síť“ označuje takovou síť, která má dvě nebo více skrytých vrstev. Existuje mnoho typů vrstev, například plně propojené, pooling, s přeskočením nebo vrstvy konvoluční.

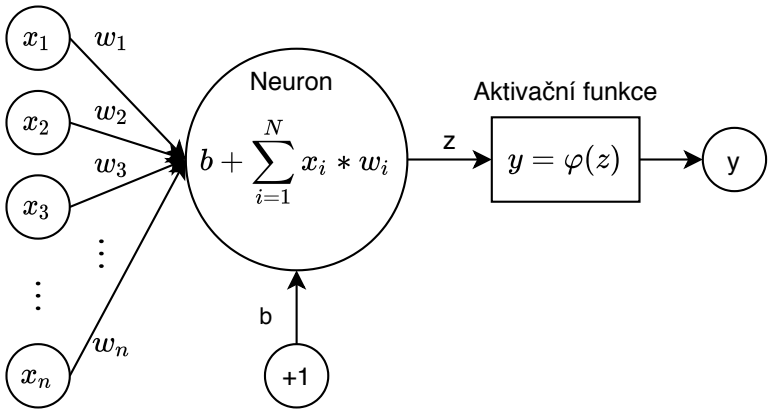


Obrázek 3.2: Schéma rekurentní neuronové sítě. Lze si všimnout orientace šipek, které směřují i zpět k předešlým uzlům grafu [28].

3.2 Umělý neuron

Základní stavební jednotka neuronových sítí je umělý neuron (*artificial neuron*) (viz obrázek 3.3). Tento model je založen na principu reálných neuronů, které se nacházejí v organizmech. Umělý neuron obsahuje libovolně mnoho vstupních propojení, přes které se mu předávají data v podobě vstupního vektoru $\vec{\boldsymbol{x}} \in \mathbb{R}^N$, kde $\vec{\boldsymbol{x}} = [x_1, x_2, \ldots, x_n], n \in N$. Sám neuron obsahuje hodnotu bias $b \in \mathbb{R}$ a vektor vah $\vec{\boldsymbol{w}} \in \mathbb{R}^N$, kde $\vec{\boldsymbol{w}} = [w_1, w_2, \ldots, w_n], n \in N$, jenž je upravován během trénování neuronu.

Výstupní hodnota závisí na vstupních datech, aktuálním vnitřním stavu (hodnoty vah a biasu) a na zvolené aktivační funkci. Vstupní hodnoty jsou váhovány, což znamená, že každá vstupní hodnota je vynásobena s váhou na daném vstupním spojení. S použitím definovaných vektorů lze napsat, že vstupní vektor je vynásoben s vektorem vah.



Obrázek 3.3: Schéma umělého neuronu.

Hodnota bias b, která je přičtena k sumě násobků vah a vstupních hodnot, je prahová hodnota modifikující dobu, kdy se aktivuje neuron a změní svůj výstup. Matematicky to znamená, že s grafem aktivační funkce horizontálně pohybuje doleva nebo doprava v závislosti na tom, je-li hodnota biasu pozitivní nebo negativní. V závislosti na řešeném problému

může být žádoucí, aby i hodnota bias byla modifikována během učení společně s ostatními váhami.

Výstup neuronu se vypočítá jako

$$y = f\left(b + \sum_{k=1}^{N} w_k x_k\right) \tag{3.1}$$

kde f je aktivační funkce, $x_k \in \mathbb{R}$ je vstupní hodnota, $w_k \in \mathbb{R}$ je váha, kterou se vstupní hodnota vynásobí a $b \in \mathbb{R}$ je hodnota bias.

3.3 Aktivační funkce

Aktivační, neboli prahová funkce, určuje výstupní hodnotu neuronu. Funkce se vybírá na základě problému, který se má neuronová síť naučit řešit. Správná volba aktivační funkce vede k lepší konvergenci učení sítě. Naopak špatná volba může vést ke stále větší odchylce od správného řešení – může divergovat. Povaha problému může vyžadovat specifické vlastnosti aktivační funkce – lineární nebo nelineární. Pro nestandardní problémy je obvykle potřeba experimentálně zjistit, která funkce bude nejlépe vyhovovat danému problému.

Pokud by veškeré aktivační funkce v modelu byly lineární, tak celkové mapování sítě by bylo omezeno pouze na lineární mapování vstupu na výstup. Reálné problémy ale lineární obvykle nejsou a v případě pokusu modelovat takovým modelem nelineární vztahy by vedlo k velice nepřesným výsledkům, který by byl zapříčiněn podučením (*underfitting*), což znamená, že model, který se učí zakódovat nějaký vzor v datasetu, je příliš jednoduchý. Proto je potřeba zavést do modelu i nelineární aktivační funkce, které tento problém řeší [18].

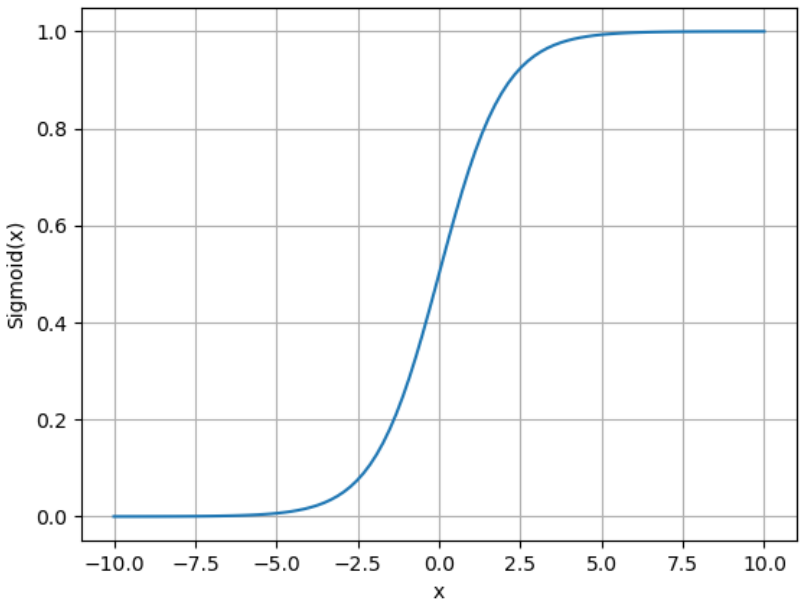
Z pohledu učení je také důležité, aby aktivační funkce byla diferencovatelná. To umožňuje použití trénovacích metod založených na výpočtu gradientu, jako je algoritmus backpropagation.

Logická sigmoida

Logická sigmoida je nelineární aktivační funkce používaná v hlubokém učení, nejčastěji v dopředných neuronových sítích. Graf funkce má hladký průběh, jak lze vidět na obrázku 3.4. Je to omezená diferencovatelná funkce, definovaná jako

$$f(x) = \frac{1}{1 + \exp(-x)} \tag{3.2}$$

Používá se na výstupních vrstvách pro predikci výstupu na základě pravděpodobnosti, čehož se využívá při řešení logistických regresních problémů, u binárních klasifikátorů a dalších oblastí neuronových sítí. Podle [25] je nejčastěji používána v sítích s menším počtem vrstev (tzv. shallow networks). Mezi hlavní nevýhody patří ostrý gradient během zpětné propagace, saturace gradientu a pomalá konvergence [27].

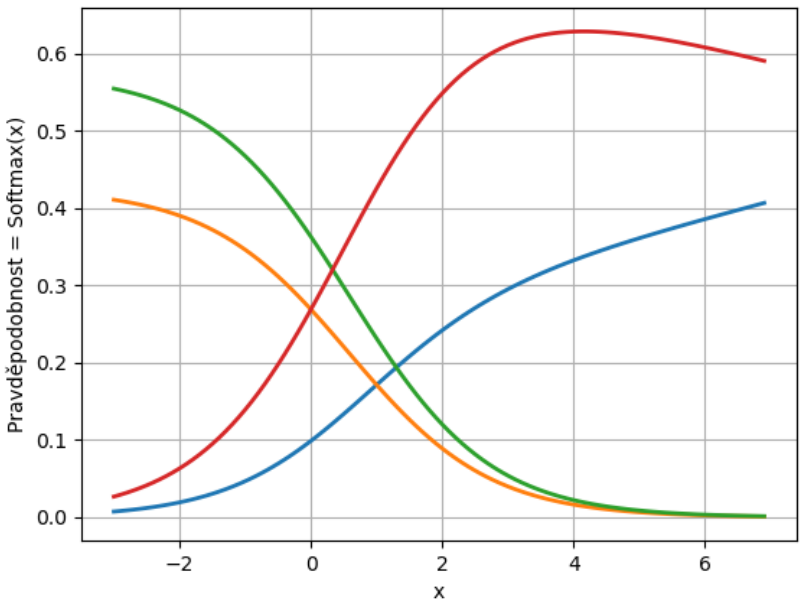


Obrázek 3.4: Graf aktivační funkce sigmoid.

Softmax

Aktivační funkce softmax [11], neboli normalizovaná exponenciální funkce, se často umisťuje na výstupní vrstvy a používá se pro výpočet rozdělení pravděpodobnosti z vektoru reálných čísel. Výstupem je hodnota v rozmezí 0 až 1, jak zobrazuje graf 3.5, na kterém lze vidět, že každá z křivek reprezentuje pravděpodobnost nějaké třídy a jejich součtem v jakémkoli bodě x získáme hodnotu 1. V klasifikačních modelech výstup reprezentuje pravděpodobnosti jednotlivých tříd s tím, že cílová třída má nejvyšší hodnotu pravděpodobnosti [27]. Vypočítá se jako

$$f(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)} \tag{3.3}$$



Obrázek 3.5: Graf aktivační funkce softmax. Každá křivka reprezentuje pravděpodobnost příslušnosti do nějaké třídy. Graf zobrazuje pravděpodobnostní rozdělení celkem 4 tříd. Součtem hodnot y v libovolném bodě x získáme hodnotu 1.

Rectified Linear Unit

Rectified Linear Unit (ReLU) je nejčastěji používaná aktivační funkce a stojí za nedávnými úspěchy v oblasti neuronových sítí [20]. Vyžaduje-li síť nějakou nelinearitu, je ReLU pro většinu případů ideální. Její použití zrychluje konvergenci trénovacího procesu a vede k lepším výsledkům, než u jednotek používající konvenční sigmoidní aktivační funkce [40]. Pro každou zápornou hodnotu x vrací 0 a pro kladnou hodnotu x vrací tutéž hodnotu x, jak jde vidět na grafu 3.6. ReLU je definována jako

$$f(x) = max(0, x) \tag{3.4}$$

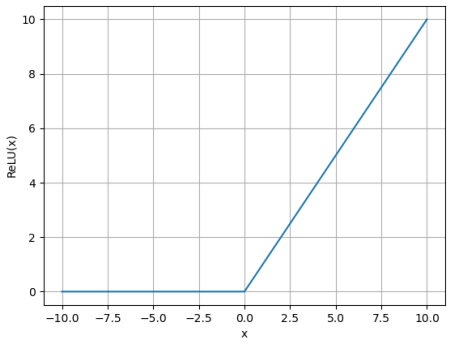
Parametric Rectified Linear Unit

Parametrizovaná ReLU (PReLU) [12], jejíž graf je na obrázku 3.7, je nelineární aktivační funkce a je to generičtější varianta funkce ReLU. Používá se v případě, že chceme produkovat na výstup malý nenulový gradient i v případě záporné vstupní hodnoty x, kdy je vynásobena parametrem α. Parametr α reguluje klesání záporné části grafu a adaptivně se učí společně s ostatními váhami během trénovacího procesu.

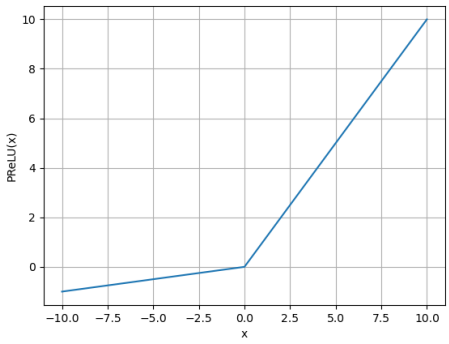
Předpis funkce PReLU je definován jako

$$f(x) = \begin{cases} x & \text{if } x \geq 0 \\ \alpha x & \text{if } x < 0 \end{cases} \tag{3.5}$$

V případě, že je hodnota parametru $\alpha = 0$, pak se jedná o ReLU. Je–li parametr $\alpha = c$, kde $c \in \mathbb{R}$ je malá konstanta (například $c = 0.03$), jedná se o Leaky ReLU (LReLU) a motivací pro její zavedení bylo vyhnout se nulovému gradientu.



Obrázek 3.6: Graf aktivační funkce ReLU.



Obrázek 3.7: Graf aktivační funkce PReLU.

3.4 Konvoluční vrstvy

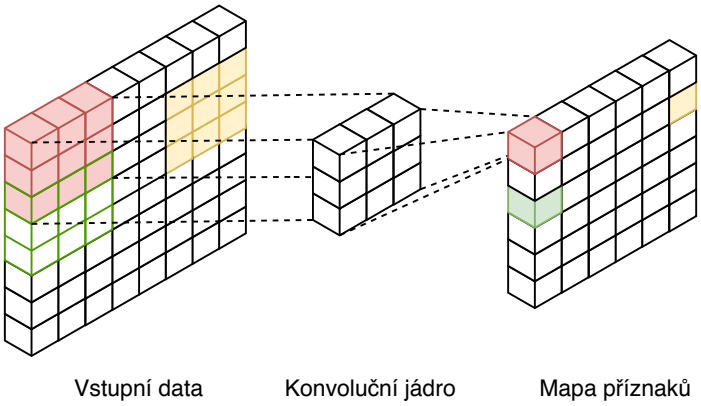
Konvoluční neuronová síť (CNN) [22] je typ dopředných neuronových sítí, která obsahuje jednu či více konvolučních vrstev. CNN jsou vhodné pro zpracování dat v mřížkovitém uspořádání. To může být ve 2–D obrázek ve formě pole pixelů či spektrum nahrávky, nebo vzorky zvukové nahrávky v čase v 1–D. Konvoluční vrstvy používají místo běžného násobení matic speciální typ lineární operace – konvoluci.

Diskrétní konvoluce je definována jako

$$(f \star g)_k = \sum_{i=-\infty}^{\infty} f_i g_{k-i} = \sum_{i=-\infty}^{\infty} f_{k-i} g_i \tag{3.6}$$

kde $\star$ je konvoluční operátor, f je funkce signálu, funkce g je konvoluční jádro a f_i a g_i jsou hodnoty funkce na indexu i.

Účelem CNN je v prvních vrstvách sítě extrahovat lokální příznaky ze vstupu. V pozdějších vrstvách neurony kombinují extrahované lokální příznaky do komplexnějších příznaků (oko, nos), až dokud nezformují v konečných vrstvách například celý obličej, znak, kočka či jiný objekt, který se síť učí detekovat. V některých modelech zpracování řeči je zvuk transformován na 2–D spektrogram (viz obrázek 2.2) a lze s ním pracovat stejně, jako při zpracování obrazu.



Obrázek 3.8: Průběh konvoluování jádra se vstupními daty a mapování lokálních příznaků na prostor příznaků (feature space), který je výstupem konvoluce. Jádro je postupně aplikováno skrze celý vstup. Je zde také dobře vidět, jak velikost jádra ovlivňuje rozměr výstupních dat. Jádro o rozměru 3×3 zmenšilo rozměr vstupních dat o rozměru 8×8 na výstupní rozměr 6×6. Toto chování se dá ovlivnit například zvýšením hodnoty *padding*.

Konvoluování jádra přes nějaký vstup (například obrázek či zvuk) je ekvivalentní s detektorem, který detekuje nějaký lokální příznak. Při aplikaci takového detektoru přes nějaký vstup zaznamenává všechny pozice, kde se příznak nacházel [18]. Jak vypadá konvoluování a mapování příznaků ilustruje obrázek 3.8.

Během trénování CNN jsou učeny váhy sítě (filtry) k detekci různých úrovní komplexnosti příznaků. Aby síť dokázala detekovat příznak nezávisle na jeho transformaci (pootočení, převrácení, posuvu), měla by být invariantní. Toho lze docílit tak, že neurony budou sdílet některé váhy (filtry).

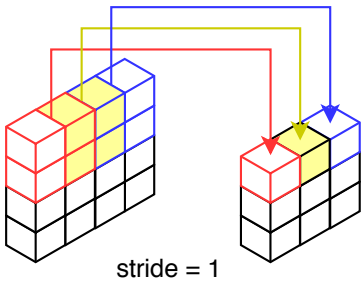
Konvoluční operace je definována jako

$$s[n] = (x \star w)[n] \tag{3.7}$$

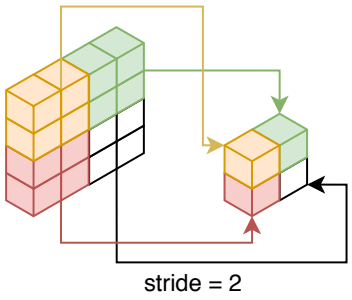
kde x jsou vstupní data obvykle ve formě multidimenzionílního pole (tenzoru) a funkce w je konvoluční jádro. Výstupem operace je mapa příznaků. Konvoluční operaci lze definovat i pro vyšší dimenze. Máme–li například 2–D vstupní obrázek, pak i konvoluční jádro bude dvourozměrné. Přestože v mnoha knihovnách lze nalézt implementovanou operaci konvoluce, často se jedná o cross–korelaci.

Průběh a chování konvoluce lze řídit hyperparametry *stride*, *padding* a *dilation*. Hodnota stride modifikuje velikost kroku konvolučního jádra. Chování při různých hodnotách lze vidět na obrázku 3.9 a 3.10.

Hodnota padding určuje, o kolik se mají rozšířit vstupní data. V případě malého rozměru konvolučního jádra může dojít k nechtěné změně rozměrů vstupních dat a proto se tato data po stranách rozšíří na takový rozměr, aby po konvoluci byl výsledný tvar v požadovaném rozměru. Vyplnění ukazuje obrázek 3.11.

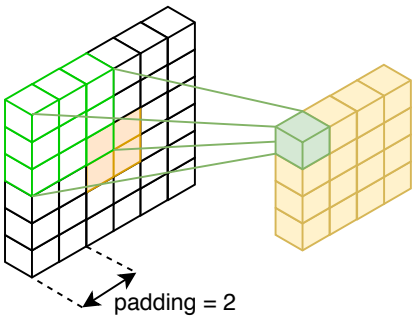


Obrázek 3.9: Hodnota stride ovlivňuje konvoluční krok jádra nad zpracovávanými daty. Čím větší hodnota stride, tím menší je výstupní rozměr dat.

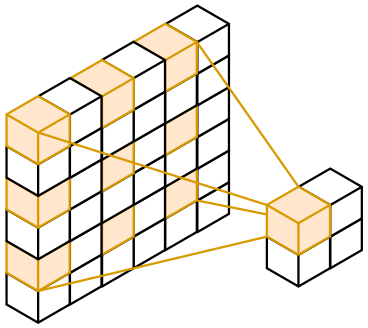


Obrázek 3.10: Hodnota kroku ovlivňuje výstupní rozměr dat. Zde je vidět, že při kroku o velikosti 2, rozměru jádra 2×2 a rozměru vstupních dat 4×4 je výstupní rozměr pouze 2×2.

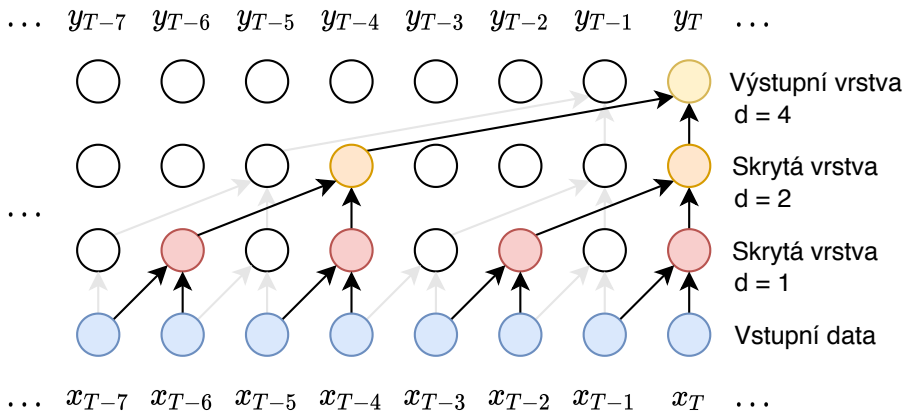
Hodnota dilation podporuje rozšíření pole působnosti bez ztráty rozlišení nebo pokrytí dat konvolučním jádrem během konvoluční operace (viz obrázek 3.12 a 3.13). Zatímco pole působnosti roste exponenciálně, počet parametrů roste pouze lineárně. Této vlastnosti se využívá například v temporal convolutional networks (TCN), kde dilatační faktor nabývá hodnot $1, 2, 4, 8, \ldots$.



Obrázek 3.11: Nastavením hodnoty padding se zpracovávaná data na okrajích rozšíří o danou hodnotu a toto rozšíření bude vyplněno nulami či jakoukoli jinou hodnotou.

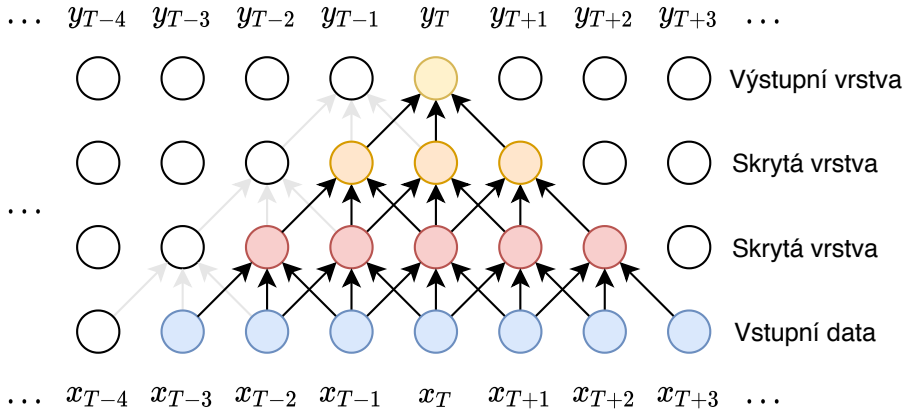


Obrázek 3.12: Vizualizace hodnoty dilatace upravující pole působnosti.



Obrázek 3.13: Vizualizace časové dilatace na vstupním signálu. Zde je vidět kauzální konfigurace, při které je hodnota y_T počítána pouze z hodnot v čase $t \leq T$.

Dalším faktorem, který ovlivňuje výpočet, je kauzalita sítě. Tato vlastnost určuje, která data mohou být použita pro výpočet výstupní hodnoty. Kauzální nastavení sítě ilustruje obrázek 3.13, zatímco obrázek 3.14 ukazuje nekauzální konfiguraci, kde je hodnota y_T počítána i z hodnot v čase $t > T$.



Obrázek 3.14: Nekauzální konfigurace umožňující výpočet hodnoty i z budoucích dat. Zde není použita časová dilatace (ale mohla by být).

3.5 Učení neuronových sítí

Cílem trénování je naučit neuronovou síť vykonávat nějaký úkol. Trénování založené na metodě učení s učitelem vyžaduje dostatečně velký dataset obsahující dvojice vstup–výstup a obnáší volbu objektivní funkce, kterou se během trénování snažíme minimalizovat.

Formálně je cílem učení aproximovat nějakou funkci f^*. Síti je předána vstupní hodnota x, pro kterou síť definuje mapování na výstupní hodnotu jako $y = f(x; \theta)$, kde θ je parametr, který se síť učí tak, aby dosáhla nejlepší aproximace funkce [11].

3.5.1 Objektivní funkce

Objektivní funkce (loss funkce, cost funkce) je funkce, která je během učení minimalizována nebo maximalizována, v závislosti na konkrétním úkolu a kontextu použití. Tato funkce mapuje událost či hodnoty na reálné číslo reprezentující „cenu“, která je asociována s touto událostí či hodnotami. Je–li tato funkce použita pro optimalizační problém, pak je cílem hodnotu minimalizovat a funkci se pak říká loss funkce. Výstupní hodnota loss funkce nám udává velikost chyby, která se počítá na základě rozdílu mezi výstupem sítě a odpovídajícími trénovacími daty z datasetu. Pak platí, že čím menší chyba, tím lépe síť provádí svůj úkol, na který je trénována. Použije–li se hodnota objektivní funkce s opačným znaménkem, tak lze funkci použít jako hodnotící metriku.

V rámci učení sítě je žádoucí, aby gradient objektivní funkce byl dostatečně velký (prudký) a předvídatelný. V takovém případě bude dobře sloužit pro účely trénování. V případě malého gradientu by funkce saturovala (byla by příliš plochá) a to by mohlo negativně ovlivnit trénování sítě [11].

Nejčastěji se pro regresní problémy používají objektivní funkce *Mean Squared Error Loss* a *Mean Absolute Error Loss*.

Mean squared error loss

Mean squared error loss (MSE) je nejčastěji používaná objektivní funkce při řešení regresních problémů. Vypočítá se jako

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y_i})^2 \tag{3.8}$$

kde N je počet trénovacích dat v datasetu, y_i je předpovězená hodnota a $\hat{y_i}$ je odhadnutá hodnota. Vzhledem k druhé mocnině je výsledek vždy pozitivní nezávisle na znaménku hodnot y_i a $\hat{y_i}$. Druhá mocnina také zajišťuje, že čím větší je rozdíl mezi předpovídanou a aktuální odhadnutou hodnotou, tím více se chyba projeví.

Mean absolute error loss

Mean absolute error loss (MAE) je velmi podobná objektivní funkce jako MSE, ale s téměř opačnými vlastnosti. Stejně jako MSE, ani tato funkce nenabývá negativní hodnoty, ale na rozdíl od MSE, která má tuto vlastnost díky druhé mocnině rozdílu, MAE toho dosahuje tak, že rozdíl předpovídané hodnoty y_j a odhadnuté hodnoty $\hat{y_j}$ je uzavřen v absolutní hodnotě. Vypočítá se jako

$$MAE = \frac{1}{N}\sum_{j=1}^{N}|y_j - \hat{y_j}| \tag{3.9}$$

Výhodnou MAE je její lineární průběh, takže se, vzhledem k lineární závislosti, chyba projeví úměrně, oproti MSE, kde je závislost kvadratická a i menší chyba se projeví více.

Nevýhodou je, že kvůli absolutní povaze není diferencovatelná v hodnotě $x = 0$, což může mít negativní následky pro výpočet gradientu.

3.5.2 Algoritmus zpětné propagace

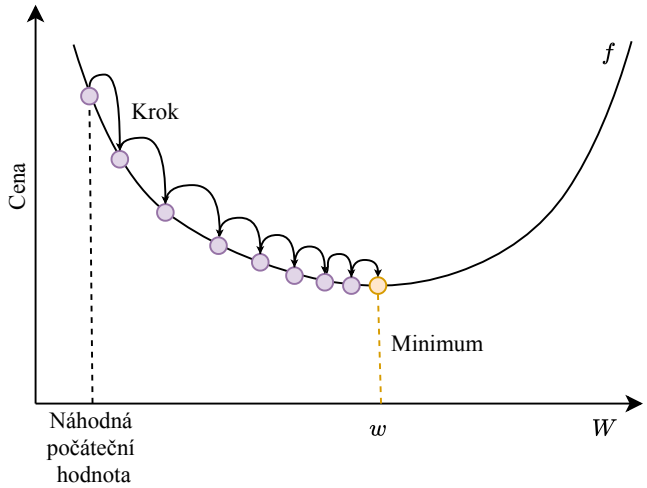
Algoritmus zpětné propagace (backpropagation algorithm) je adaptační algoritmus pro učení neuronových sítí metodou učení s učitelem. Algoritmus pozpátku prochází neuronovou sítí a na základě hodnoty loss počítá gradient objektivní funkce $J(\theta)$ vzhledem ke

všem váhám v síti. Skalární hodnota loss je výstupem objektivní funkce $J(\theta)$ a gradient této funkce vyjadřuje, jak moc každá váha přispívá k celkové chybě. Cílem je minimalizovat hodnotu loss. Algoritmus je nejčastěji používán během trénování neuronových sítí, ale obecně ho lze použít pro výpočet jakékoli derivace [18], [11].

Proces začíná na výstupní vrstvě sítě. Algoritmus zde pro každý neuron spočítá chybový gradient δ reprezentující citlivost chyby sítě na změnu váhovaného součtu daného neuronu. Tyto gradienty jsou iterativně zpětně propagovány do předchozích vrstev, na základě nichž jsou počítány gradienty neuronů v předchozích vrstvách, až po vrstvu vstupní. Pro úpravu vah na základě vypočítaných chybových gradientů se obvykle používá algoritmus gradient descent [18].

Gradient descent

Gradient descent je iterativní optimalizační algoritmus, definující pravidla pro samotnou úpravu vah v síti. Je určený pro hledání lokálního minima diferencovatelné objektivní funkce $J(\theta)$, kde $\theta \in \mathbb{R}^d$ reprezentuje učené parametry modelu. Hledání lokálního minima probíhá úpravou parametrů modelu v opačném směru, než je hodnota gradientu. Výpočet je prováděn postupným pohybem ve směru největšího klesání, které je určeno zápornou hodnotou gradientu. Rychlost pohybu závisí na velikosti kroku, což udává *learning rate*. Správná volba *learning rate* ovlivňuje rychlost, jakou je nalezeno minimum funkce. Při nízké hodnotě bude výsledek přesnější, ale nalezení minima bude výpočetně náročnější, protože v každém kroku se počítá nová hodnota gradientu. Při větší hodnotě je riziko, že minimum bude přeskočeno [32]. Obrázek 3.15 zobrazuje možný průběh algoritmu.



Obrázek 3.15: Ukázka algoritmu gradient descent při hledání minima dané funkce f postupným přibližováním dle nastavené velikosti kroku. Inspirováno podle [5].

3.5.3 Underfitting a generalizace

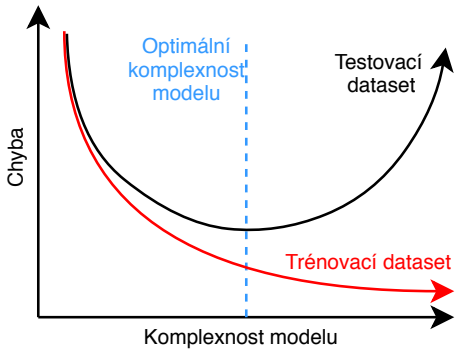
Generalizace je vlastnost modelu udávající, jak dobře model pracuje s dříve neviděnými daty, tedy s daty, které nebyly použity pro jeho učení. Měří se pomocí testovací chyby. Během učení vzniká učící chyba, která je trénováním postupně minimalizována, ale kvalita modelu je pak měřena na testovacích datech.

Testovací chyba by ideálně měla být co nejblíže chybě na trénovací sadě. Z rozdílu mezi těmito chybovými hodnotami se dají diagnostikovat problémy s naučením sítě – podučení (*underfitting*) a přeučení(*overfitting*).

Underfitting nastane v případě, že model již nedokáže v důsledku jeho kapacity zmenšit dostatečně jeho chybovou hodnotu na trénovacím datasetu.

Overfitting nastane, když rozdíl mezi trénovací chybou a testovací chybou je příliš velký.

Kapacita je vlastnost modelu určující, kolik se toho dokáže model naučit. V případě nízké kapacity se model nedokáže naučit všechny příznaky z trénovacích dat. V případě vysoké kapacity se mohou přeučit zapamatováním si jejich příznaků a tím sice dosáhnou nízké trénovací chyby, ale špatných výsledků během testování [11]. Vliv kapacity na výsledky trénování a testování lze vidět na 3.16.

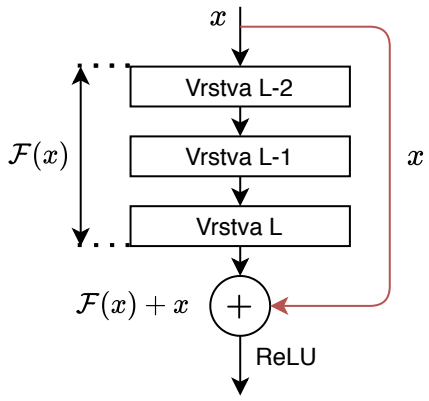


Obrázek 3.16: Vliv kapacity modelu na trénovací a testovací chybu. Obrázek inspirován podle [6].

Regularizace je souhrn postupů, které říkají, jak modifikovat učící algoritmus tak, aby se zredukovala chyba generalizace, zatímco testovací chyba zůstane stejná [11].

3.5.4 Reziduální spojení

Velmi hluboké neuronové sítě přinesly v mnoha aplikacích velmi dobré výsledky. S přibývající hloubkou se ale stává její trénování složitější. Jeden z problémů, který se při učení projevoval, byl explodující a mizející gradient [4]. Tento problém byl adresován zavedením normalizačních vrstev [17] a počáteční optimalizované inicializace, což dovolilo sítím konvergovat s využitím *stochastic gradient descent* se zpětnou propagací (*back–propagation*) [22]. Poté se objevil problém degradace, který nebyl zapříčiněn přetrénováním, a způsobil, že s přibývající hloubkou sítě její přesnost náhle prudce klesla, což indikovalo, že systémy nelze optimalizovat stejným způsobem a přidání dalších vrstev pouze zvýšilo trénovací chybu. Problém degradace lze řešit zakomponováním reziduálních spojení, které je zobrazeno na obrázku 3.17 [13].



Obrázek 3.17: Reziduální spojení mezi vstupem a výstupem stavebního bloku sítě [13].

Reziduální spojení, neboli identitní mapování, je zkratka mezi jednou či více vrstvami. Problém mizejícího gradientu řeší tím, že používá aktivace z předchozí vrstvy, dokud váhy následující vrstvy nejsou naučeny. Spojení lze formulovat jako $\mathcal{F}(x)+x$, kde $\mathcal{F}(x)$ je výsledná hodnota transformace jednotlivými vrstvami a x je původní vstup, který vrstvy přeskočí a následně je sečten s výstupem $\mathcal{F}(x)$. Výhodou je, že jeho použitím se nezvyšuje počet parametrů ani výpočetní náročnost [13], [39].

3.5.5 Význam validační množiny v trénování

Většina algoritmů strojového učení má nějakou sadu hyperparametrů, kterou je upravováno chování algoritmu. Hodnoty hyperparametrů obvykle bývají nastavovány ručně ještě před spuštěním procesu učení a hodnota se v průběhu nemění. Některá nastavení se nicméně mohou stát hyperparametrem a být upravována během trénování, ale není vhodné je měnit na základě výsledku učení na trénovací sadě, protože by mohlo dojít k přetrénování (*overfitting*). Pro tento případ se použije validační sada, která je odlišná od trénovací sady a síť z ní ještě neviděla nahrávky. Validační sada je zpracována po skončení zpracování trénovací sady a je použita pro monitorování, jak dobře síť generalizuje dosud neviděná data. Na základě výsledků lze následně rozhodnout o úpravě některých hyper–parametrů [11].

Kapitola 4

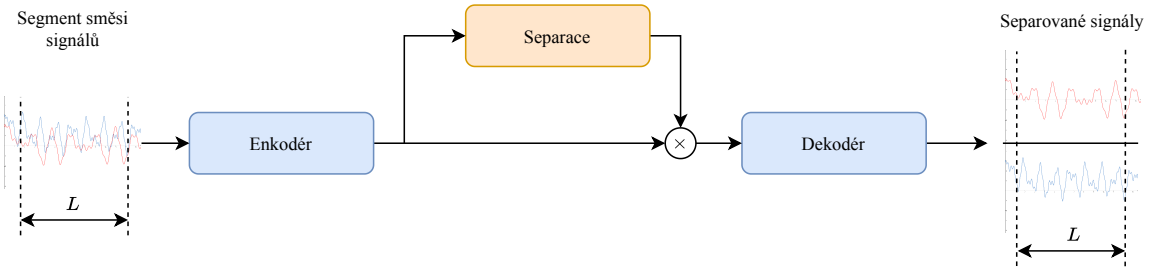
Time–Domain Audio Separation Network

Tato kapitola vychází z referenční studie **TasNet: Surpassing Ideal Time-Frequency Masking for Speech Separation** [24] a popisuje její navrhnuté řešení pro separaci mluvčích v časové doméně.

Přestože metody pro zpracování řeči v takovém akustickém prostředí, ve kterém se současně prolíná mnoho řečových signálů, v poslední době zaznamenaly velké zlepšení, stále trpí mnoha nedostatky. Přesnost systémů, odezva a nároky na výpočetní výkon jsou u těchto metod natolik zásadní, že znemožňují nebo velmi omezují jejich nasazení mimo výzkumné prostředí, například v aplikacích, které by mohly zpracovávat řeč v reálném čase.

Většina dosavadních postupů byla založena na převodu směsi řečových signálů do časově–frekvenční (T–F) reprezentace (spektrogramu) pomocí short–time Fourier transform (STFT), ale tato reprezentace není optimální pro separaci mluvčích [37].

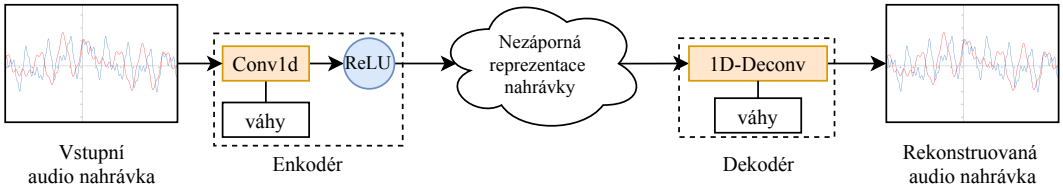
Pro překonání nedostatků předešlých metod byla navržena architektura fully convolutional time–domain audio separation network (Conv–TasNet), založena na hlubokém učení a konvolučních neuronových sítích. Model prvně použije konvoluční enkodér k převodu krátkých segmentů směsi mluvčích na odpovídající nezápornou reprezentaci, která je optimalizovaná pro extrakci jednotlivých mluvčích. Samotné separace je docíleno aplikací masek na danou reprezentaci. Masky pro každého mluvčího pro každý segment v každém časovém kroku jsou odhadnuty v temporal convolutional network (TCN) [21], [2]. TCN je tvořena opakující se posloupností konvolučních bloků se zvyšující se časovou dilatací. Po aplikaci masek jsou separovaní mluvčí rekonstruováni lineárním dekodérem. Tuto posloupnost operací zobrazuje zjednodušený obrázek 4.1. Dále v kapitole budou jednotlivé části popsány detailněji.



Obrázek 4.1: Schéma architektury TasNet.

4.1 Konvoluční auto–enkodér

Konvoluční auto–enkodér převádí vstupní segmenty nahrávky na nezápornou reprezentaci a následně zase zpět z reprezentace na původní nahrávku, jak ukazuje obrázek 4.2.



Obrázek 4.2: Schéma konvolučního autoenkodéru.

4.1.1 Enkódování směsi

Každý segment směsi o délce L, $\boldsymbol{x_k} \in \mathbb{R}^{1\times L}$, kde $k = 1, \ldots, \frac{T}{L}$, je transformován na nezápornou reprezentaci $\boldsymbol{w} \in \mathbb{R}^{1\times N}$ pomocí 1–D konvoluční operace jako

$$\boldsymbol{w} = ReLU(\boldsymbol{x} \circledast \boldsymbol{U}) \tag{4.1}$$

kde $\boldsymbol{U} \in \mathbb{R}^{N\times L}$ obsahuje N vektorů, každý délky L, které reprezentují bázové funkce enkodéru. Operace $\circledast$ značí konvoluční operaci. ReLU je nelineární aktivační funkce, která byla blíže popsána v podkapitole 3.3.

4.1.2 Dekódování extrahovaných mluvčích

Pro převod z reprezentace zpět do podoby audio nahrávky slouží lineární dekodér. Pomocí 1–D dekonvoluční operace rekonstruuje původní signál $\boldsymbol{x}$ jako $\boldsymbol{x} \in \mathbb{R}^{1\times L}$. Tuto operaci lze definovat jako

$$\hat{\boldsymbol{x}} = \boldsymbol{w}\boldsymbol{V} \tag{4.2}$$

kde každý řádek v matici $\boldsymbol{V} \in \mathbb{R}^{N\times L}$ představuje jednu bázovou funkci dekodéru s délkou L.

4.2 Separační modul

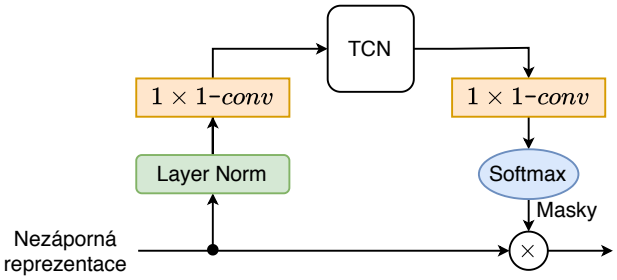
Cílem separačního modulu je najít váhovanou funkci, neboli masku, pro každého zdrojového mluvčího pro každý výstup enkodéru v každém časovém kroku. Formálně lze zapsat, že cílem je odhadnout C masek $\boldsymbol{m}_i \in \mathbb{R}^{1\times N}, i = 1, \ldots, C$, kde C představuje počet mluvčích ve směsi. Vektory masek $\boldsymbol{m}_i$ mají takové omezení, že $\sum_{i=0}^{C} \boldsymbol{m}_i = 1$, kde 1 je jednotkový vektor v $\mathbb{R}^{1\times N}$. Toto omezení garantuje, že rekonstruované zdroje po sečtení zformují původní směs $\hat{\boldsymbol{x}} = \sum_{i=1}^{C} \hat{\boldsymbol{s}}_i$.

Separace je pro každý zdroj provedena vynásobením odpovídající masky $\mathbf{m}_i$ s nezápornou reprezentací (výstupem enkodéru) w, jako

$$\boldsymbol{d_i} = \boldsymbol{w} \odot \boldsymbol{m}_i \tag{4.3}$$

kde $\boldsymbol{d_i} \in \mathbb{R}^{1\times N}$ je reprezentace každého ze zdrojů a operace $\odot$ je součin po složkách.

Nakonec jsou reprezentace $\boldsymbol{d_i}$ rekonstruovány zpět na korespondující nahrávky zdrojů $\hat{\boldsymbol{s}}_i, i = 1, \ldots, C$ pomocí dekodéru, jak je znázorněno v 4.2.



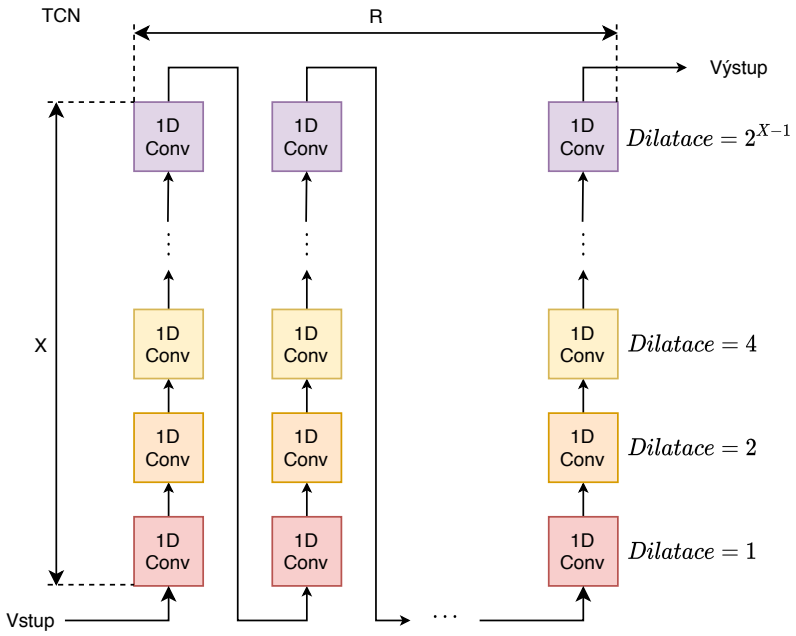
Obrázek 4.3: Schéma separačního modulu, jehož vstupem je nezáporná reprezentace směsi mluvčích.

Na začátku separačního modulu je přidána lineární operace 1×1–conv jako bottleneck, která určuje počet kanálů na vstupu a výstupu následující sekvence konvolučních bloků. Jak zapadá separační modul do schématu je zobrazeno na obrázku 4.3.

4.2.1 Temporal Convolutional Network

Jádrem separační části je *temporal convolutional network* skládající se z naskládaných 1–D konvolučních bloků s časovou dilatací jak lze vidět na obrázku 4.4. Každá vrstva v TCN obsahuje X konvolučních bloků se zvyšující se časovou dilatací, která se zvyšuje exponenciálně podle počtu bloků a nabývá hodnot $1, 2, 4, \ldots, 2^{X-1}$. Taková sekvence bloků je opakována R–krát. Exponenciální růst dilatačního faktoru zajišťuje dostatečně velké okno pro využití výhod dlouhých časových závislostí v signálu řeči. Obrázek 3.13 zobrazuje rostoucí časovou dilataci na vstupních vzorcích signálu.

Výstup posledního bloku posledního opakování v TCN je předán 1×1 konvoluční vrstvě, která má $N \times C$ filtrů a nakonec aktivační funkci softmax k odhadu C vektorů masek pro každého z C cílových mluvčích.



Obrázek 4.4: Jádro separačního modulu – naskládané konvoluční bloky s časovou dilatací odhadující masky na základě nezáporné reprezentace.

4.2.2 Konvoluční bloky

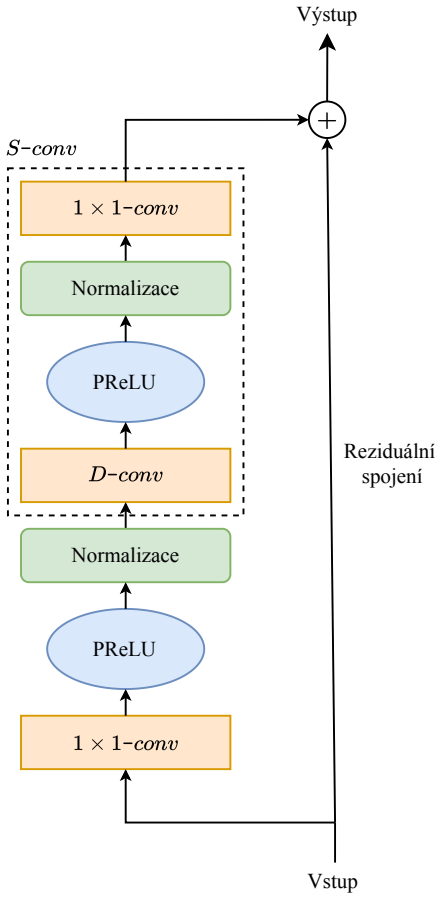
Navržená architektura dále nahradila standardní konvoluci uvnitř 1×1 konvolučních bloků za *depthwise separable convolution* (*S-conv*), která pomáhá snížit počet parametrů a ukázala se jako efektivní ve zpracováni obrazu [8], [16]. Konvoluční bloky jsou ve schématu 4.4 znázorněny jako „1D Conv“.

Depthwise separable convolution se skládá ze dvou, po sobě jdoucích, operací – depthwise convolution (*D-conv*) a standardní konvolucí 1×1-*conv* s velikostí konvolučního jádra 1:

$$D\text{-}conv(\boldsymbol{Y}, \boldsymbol{K}) = concat(\boldsymbol{y}_j \circledast \boldsymbol{k}_j), j = 1, \ldots, N \tag{4.4}$$

$$S\text{-}conv(\boldsymbol{Y}, \boldsymbol{K}, \boldsymbol{L}) = D\text{-}conv(\boldsymbol{Y}, \boldsymbol{K}) \circledast \boldsymbol{L} \tag{4.5}$$

kde $\boldsymbol{Y} \in \mathbb{R}^{G \times M}$ je vstup do *S-conv*, $\boldsymbol{K} \in \mathbb{R}^{G \times P}$ je konvoluční jádro o velikosti P, dále $\boldsymbol{y}_j \in \mathbb{R}^{1 \times M}$ a $\boldsymbol{k}_j \in \mathbb{R}^{1 \times P}$ jsou řádky matic $\boldsymbol{Y}$ a $\boldsymbol{K}$. $\boldsymbol{L} \in \mathbb{R}^{G \times H \times 1}$ je konvoluční jádro o velikosti 1. Operace 1×1-*conv* se chová jako plně propojená vrstva a transformuje příznaky do potřebných rozměrů.



Obrázek 4.5: Jeden konvoluční blok obsahující reziduální spojení mezi vstupem a výstupem. Je zde vidět sekvence operací D-$conv$ a 1×1-$conv$, které dohromady tvoří operaci S-$conv$.

V každém konvolučním bloku jsou operace 1×1-$conv$ a D-$conv$ následovány nelineární aktivační funkcí PReLU [12], která byla popsána v kapitole 3, a normalizační vrstvou. Konvoluční bloky dále obsahují reziduální spojení [13] mezi vstupem a výstupem, které výrazně zjednodušuje trénování velmi hlubokých neuronových sítí díky ponechané referenci na vstupní netransformovaná data.

4.3 Normalizace

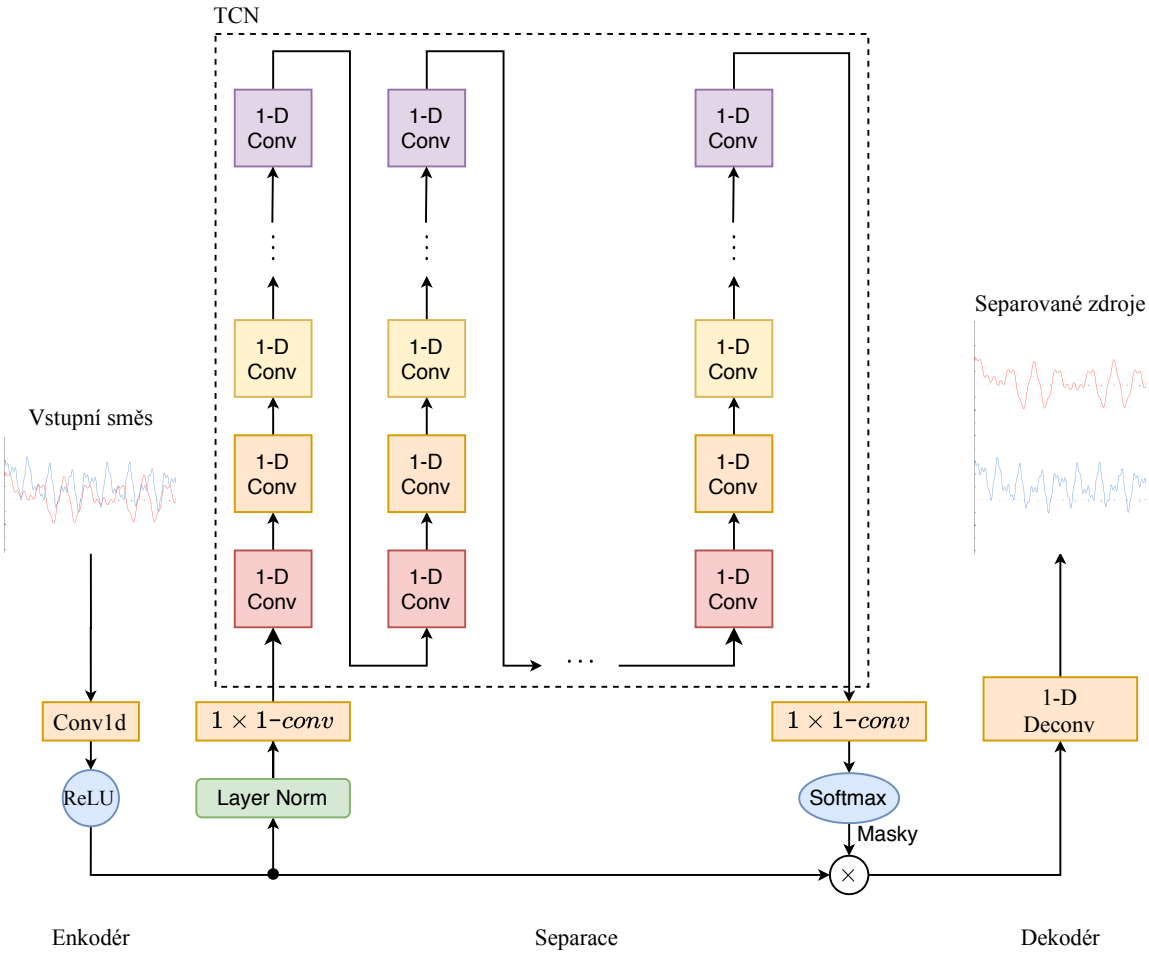
Při trénování neuronových sítí může docházet k fenoménu „vnitřní kovariantní posuv“ (*internal covariate shift*), kvůli kterému je nutné pečlivě inicializovat parametry a volit menší hodnotu learning rate, což zpomaluje trénování sítě. K adresování problému slouží zakomponování normalizace do architektury neuronové sítě. Normalizace je aplikována na každý mini–batch trénovacích dat [17], [1].

Konvoluční bloky obsahují normalizační vrstvy, které můžou významně ovlivnit výkon sítě. Ve studii bylo experimentováno s channel–wise layer normalization (cLN), global layer normalization (gLN) a batch normalization (BN) [17], kterou jsem se rozhodl použít v mé implementaci.

Metoda cLN je aplikována na vstup separačního modulu pro zajištění invariance při změně měřítka vstupních dat. Tato metoda je vhodná pro použití v kauzální i nekauzální konfiguraci, je–li použita v konvolučních blocích. Normalizace cLN je aplikována na každý segment y_k.

Metoda gLN je aplikována globálně na každý příznak na rozměry kanálu i času. Tuto normalizaci lze použít pouze při nekauzální konfiguraci, protože výpočet probíhá na základě celého vstupu. Při kauzální konfiguraci může být použita také metoda BN.

Jak vypadá kompletní schéma architektury po sestavení a propojení jednotlivých modulů lze vidět na obrázku 4.6.



Obrázek 4.6: Schéma kompletní architektury TasNet.

Kapitola 5

Implementace sítě

Pro implementaci neuronových sítí vzniklo mnoho frameworků jako PyTorch, Tensorflow, Keras a další, které umožňují síť poskládat z již předdefinovaných modulů. Jejich vlastnosti lze modifikovat dle potřeby pomocí argumentů při jejich instanciaci nebo při jejich použití. Kromě těchto bloků a mnoha dalších možností frameworky obsahují i metody pro práci s datasety, funkce pro vykreslování grafů a pro matematické operace.

Mým úkolem bylo implementovat neuronovou síť podle architektury TasNet [24] pro separaci mluvčích v časové doméně. V rámci zadání jsem si zvolil použít framework s názvem PyTorch[1], který má kvalitní dokumentaci a aktivní uživatelskou základnu. PyTorch funguje nad jazykem Python[2] a v celé práci používám jeho aktuálně nejnovější verzi (Python 3.8.2). Mimo jiné podporuje práci s datasety, včetně implementace vlastního nahrávání a manipulace s daty. Základní jednotkou, se kterou síť pracuje, je tensor. Ve frameworku PyTorch je `torch.Tensor` třída reprezentující multidimenzionální matici obsahující data jednoho typu. Na tensory jsou převáděna data během načítání datasetu a následně předávána síti ke zpracování. Samotná implementace probíhala na systému Kali linux (4.19.0–kali5–amd64), který je založený na systému Debian. Aplikace byla vyvíjena v textovém editoru Vim[3] a byla během implementace spouštěna a testována v příkazovém procesoru Bash[4] (Bourne Again SHell) verze 5.0.16. Bash jsem zvolil i pro implementaci některých pomocných skriptů kvůli jednoduchému spouštění z terminálu a možnostem automatizace některých opakujících se příkazů. Pro správu verzí jsem používal verzovací systém Git[5].

Trénování sítě probíhalo z počátku na CPU *intel i5* mého osobního notebooku *Lenovo Y50–70*, což se ale ukázalo jako naprosto nevhodné vzhledem k náročnosti výpočtu kvůli nedostatečnému výkonu. Začal jsem tedy používat službu *Google colab*, která poskytuje na omezenou dobu (12 hodin) stroje, které navíc obsahují technologii *cuda*, což mi umožnilo výpočet provádět na GPU, čímž se trénování zrychlilo několikanásobně. Ještě více pomohlo předplacení služby, čímž se mi prodloužila doba, po kterou jsem mohl síť trénovat, na 24 hodin a zvýšila se šance na získání silnějších strojů. Modely byly trénovány na GPU *T80* a *T100* v závislosti na tom, která z nich mi zrovna byla přidělena pro dané sezení (*session*).

[1] https://pytorch.org/
[2] https://www.python.org/
[3] https://www.vim.org/
[4] https://www.gnu.org/software/bash/
[5] https://git-scm.com/

Pro vyhodnocování výkonu sítě byly použity metriky SDR, PESQ a STOI. Pro výpočet SDR jsem použil zdrojový kód nacházející se na githubu [6]. Metriky PESQ [7] a STOI [8] byly nainstalovány pomocí balíkového nástroje pip [9].

Algoritmus 1: Zjednodušený algoritmus běhu programu pro trénování sítě.

```
parsování argumentů programu;
instanciace modelu TasNet;
if args.checkpoint! = NULL then
    načtení checkpointu;
end
instanciace třídy SegmentDataset a dataloaderu pro trénovací data;
instanciace třídy SegmentDataset a dataloaderu pro validační data;
for epocha v args.Epochs do
    /* trénování */
    loss = 0;
    for směs_mluvčích z trénovacího datasetu do
        /* cyklus načítá z dataloaderu segmenty nahrávek, které jsou
           generovány v třídě SegmentDataset */
        odhadovan_separace = TasNet(sms_mluvch);
        loss = sisnr(optimln_separace, odhadovan_separace);
        propagace chyby a úprava vah;
    end
    /* validace */
    vypnout gradient;
    for směs_mluvčích z validačního datasetu do
        odhadovan_separace = TasNet(sms_mluvch);
        validacni_loss = sisnr(optimln_separace, odhadovan_separace);
        if hodnota validační loss neklesla potřetí v řadě then
            learning_rate = learning_rate/2;
        end
    end
    na konci epochy uložit checkpoint;
end
```

5.1 Implementace tříd a modelu

Model neuronové sítě byl logicky rozdělen na několik dílčích částí, z nichž každá je reprezentována jednou třídou. Jednotlivé třídy obsahují metody pro nahrávání a transformaci dat, segmentaci nahrávek a další jsou přímo stavebním blokem neuronové sítě. Některé parametry předané třídám při jejich instanciaci jsou hyperparametry, jejichž hodnoty byly určeny parametry z příkazové řádky při volání programu. Každá z tříd bude popsána dále v textu.

[6] https://github.com/craffel/mir_eval
[7] https://github.com/ludlows/python-pesq
[8] https://github.com/mpariente/pystoi
[9] https://pypi.org/project/pip/

Třídy SegmentDataset a AudioDataset

Třídy `SegmentDataset` a `AudioDataset` slouží pro načítání korespondujících trojic nahrávek směs–zdroj1–zdroj2 z adresáře, kde je uložen dataset. Nahrávky jsou ve formátu wav a vzorkovány na 8KHz. Třídy tato data dále transformují na tenzory a normalizují na jednotkovou varianci a zero–mean.

Tyto třídy používají některé pomocné funkce definované v souboru `tools.py` pro dorovnání nedostatečně velkých nahrávek ve zpracovávaném mini–batchi nulami na velikost nejdelší z nich, protože data obsažená v mini–batchi musejí mít stejnou velikost.

Při spuštění programu jsou nahrány názvy wav souborů do polí, které jsou poté buď náhodně či sekvenčně procházeny a postupně dochází k načítání trojic nahrávek na aktuálním indexu.

Obě třídy jsou periodicky volány instancí třídy `Dataloader`[10], která v cyklu posílá požadavky na data ke zpracování. Rozdíl mezi třídami je ten, že třída `AudioDataset` předává trojice nahrávek v původním stavu a délka epochy tak trvá tolik iterací, kolik je nahrávek ve zpracovávaném datasetu a je kvůli tomu používána během testování sítě, kde chceme vyhodnocovat celé nahrávky. Třída `SegmentDataset` je použita pro trénovací a validační dataset, kde jsou nahrávky rozdělovány na segmenty o parametrem dané délce a až poté jsou poskytnuty ven. Segmentace je detailně popsána v podkapitole 5.2.

Po vyčerpání datasetu obě třídy vyvolají vyjímku `StopIteration`, která je detekována `Dataloaderem` a ten ukončí cyklus, ve kterém se dataset zpracovával. Tím v případě trénování končí učení sítě a začíná validace, v případě validace končí jedna epocha a v případě testování končí vyhodnocování sítě.

Třída Tasnet

Třída `Tasnet.py` reprezentuje model neuronové sítě. Atributy třídy představují jednotlivé vrstvy neuronové sítě. Síť obsahuje konvoluční vrstvy, enkodér, dekodér a *temporal convolutionan network*, která je tvořena sekvencí konvolučních bloků, které jsou rozebrány později. Konvolučním vrstvám jsou nastaveny hodnoty *padding* a *stride*, které ovlivňují chování konvolučních vrstev. V této třídě jsou také inicializovány váhy (filtry) konvolučních vrstev algoritmem Xavier [10]. Ve funkci `forward()`, která je volána, když jsou instanci sítě předána data ke zpracování, jsou sekvenčně volány jednotlivé vrstvy. Během zpracování předaných dat je v této sekvenci vypočítána maska a aplikována na zpracovávanou směs. Tím vzniknou dvě separované nahrávky, které jsou předány na výstup v podobě tensoru.

Třída ResBlock

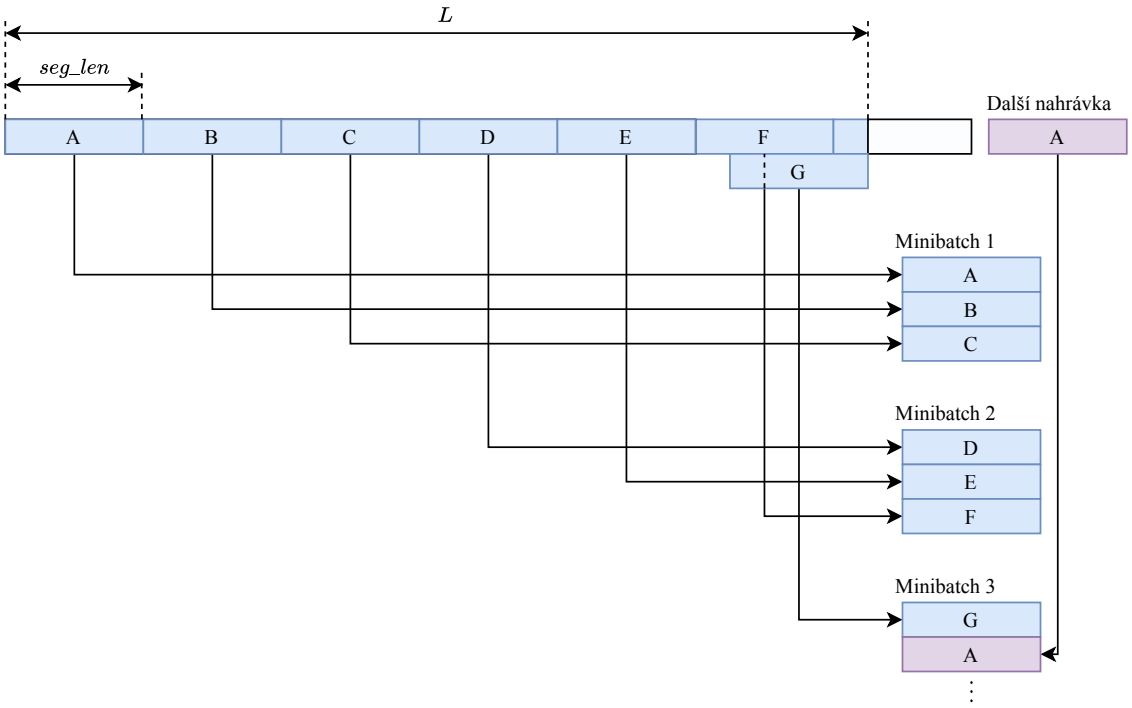
Třída `ResBlock` reprezentuje jeden konvoluční blok, který je opakován se zvyšující se dilatací v *temporal convolutional network* v separační části určené k odhadu masek. Podobně jako třída `Tasnet`, i tato třída obsahuje inicializaci a zřetězení vrstev. V inicializaci jsou nastaveny hodnoty pro konvoluční operace jako počet konvolučních jader, časová dilatace a počet kanálů. Tyto operace jsou ve funkci `forward()` zřetězeny do sekvence konvolučních operací a aktivačních funkcí, podle vzorové architektury TasNet. Třída `ResBlock` je specifická svým residuálním spojením, kde dochází k rozdělení dat při zpracování, kdy jedna kopie je transformována operacemi v bloku a na konci sečtena s daty, které transformacemi neprošly a tento součet je výstupem bloku. Residuální spojení jsou podrobněji popsána v kapitole 3.

[10] https://pytorch.org/docs/stable/data.html#torch.utils.data.DataLoader

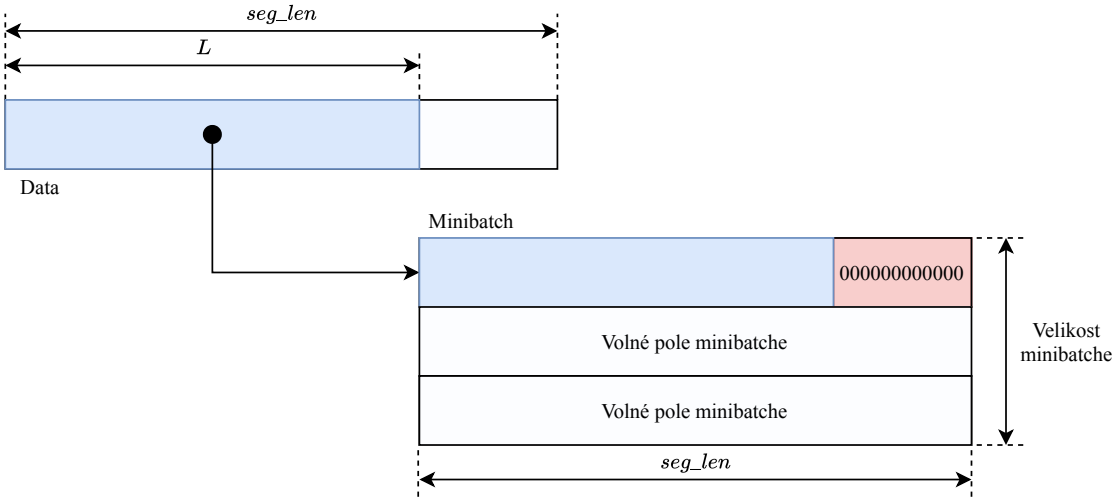
5.2 Segmentace nahrávek

Pro účely trénování sítě byly vstupní nahrávky rozdělovány na segmenty o délce 4, případně 2 sekund. Délku segmentů L_s lze nastavit přes argument `--segment_length` při volání skriptu `train.py`. Tato hodnota reprezentuje jeden z hyper–parametrů sítě. Výchozí hodnota je 4 sekundy. Při nižších hodnotách se prodlužuje délka trénování, protože se z jedné vstupní nahrávky o délce L vygeneruje až L/L_s segmentů a tím narůstá počet dat, který síť musí zpracovat.

Poslední segment by byl kratší, pokud by délka nahrávky byla kratší než délka segmentu. V takovém případě je segment doplněn nulami do délky ostatních segmentů jak lze vidět na obrázku 5.2. Toto doplnění je nutné kvůli dávkovému zpracování (*mini–batches*), které neumožňuje rozdílnou délku elementů v dávce. Pokud je délka nahrávky delší než délka segmentu, tak se poslední segment vezme od konce bez nutnosti ho pak nulami doplňovat. Skládání segmentů do minibatche zobrazuje obrázek 5.1. Při vzorkovací frekvenci $f_s = 8000Hz$, obsahují segmenty $f_s * L_s$ vzorků vstupní nahrávky.



Obrázek 5.1: Dělení nahrávky na segmenty o délce `segment_len` a jejich následné vkládání do minibatche. Lze si všimnout, že pokud poslední segment není dostatečně dlouhý, tak se vezme od konce nahrávky a jeho počáteční vzorky budou duplicitní s posledními vzorky předešlého segmentu.

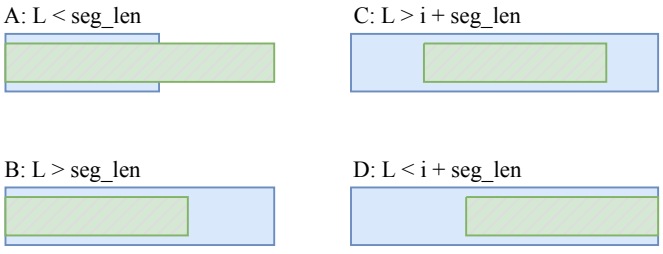


Obrázek 5.2: Je–li zpracovávána nahrávka, jejíž délka je kratší než je délka segmentu, je tento segment doplněn z prava nulami do požadované délky.

Algoritmus 2 a obrázek 5.3 zobrazují proces segmentace. Funkce `segmentGenerator()`, která slouží jako generátor, pracuje současně se třemi nahrávkami – se směsí dvou mluvčích, s nahrávkou prvního mluvčího a s nahrávkou druhého mluvčího, které jsou do atributů `current_mixture`, `current_source1` a `current_source2` třídy `SegmentDataset` nahrávány funkcí `loadNextAudio()` poté, co z předchozí trojice nahrávek již nelze vygenerovat další segmenty.

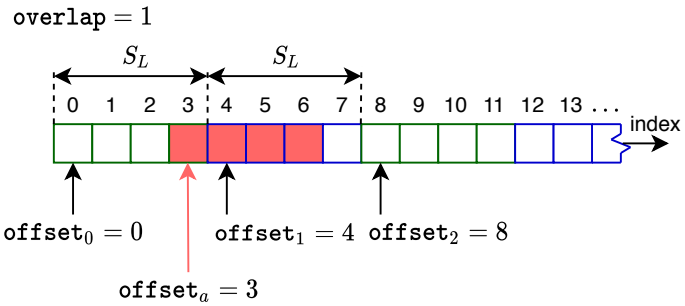
Algoritmus 2: Algoritmus segmenace nahrávek používá výhody Python generátoru, který, na rozdíl od běžných funkcí, po vrácení hodnoty příkazem `yield` neztrácí svůj vnitřní stav a při jeho dalším zavolání pokračuje tam, kde skončil. V uvedeném příkladě je znázorněna pouze segmentace nahrávky jednoho mluvčího, ale analogicky se segmentují i nahrávky směsi mluvčích a nahrávky druhého mluvčího. Vizuálně lze sledovat varianty segmentace podle značek A, B, C a D v komentářích algoritmu na obrázku 5.3.

```
1  Function segmentGenerator() : int[] is
2      s1_segment = [];
3      segptr = 0;
4      while je dostupná další trojice nahrávek do
5          if current_mixture_len < SEGMENT_LENGTH then
               /* A) aktuální nahrávka je kratší než délka segmentu */
6              s1_segment = current_source1[:];
7              yield s1_segment;
8          else
               /* B) aktuální nahrávka je delší než délka segmentu */
9              if segptr + SEGMENT_LENGTH < current_mixture_len then
                   /* C) z aktuální nahrávky vzít segment */
10                 s1_segment = current_source1[(segptr+SEGMENT_LENGTH)];
11                 segptr+ = SEGMENT_LENGTH;
12                 yield s1_segment;
13             else
                   /* D) nelze již načíst celý segment, takže se vezme od
                      konce */
14                 start_index = (current_mixture_len - SEGMENT_LENGTH);
15                 s1_segment =
                    current_source1[start_index:current_mixture_len];
16                 yield s1_segment;
17             end
18         end
19     end
20 end
```



Obrázek 5.3: Způsob segmentace nahrávek. Obrázek slouží jako doplnění algoritmu 2. Význam schémat A, B, C a D je označen na odpovídajících pozicích v komentářích algoritmu. Šrafovaná pole reprezentují vytvářený segment a modrá pole reprezentují data zpracovávaného signálu.

Kód segmentace také umožňuje určit délku, o kterou se segmenty budou překrývat. Této funkcionality je docíleno zavedením proměnné `overlap`, která obsahuje délku překrytí a při segmentaci je od proměnné `offset`, který určuje začátek následujícího segmentu, tato hodnota odečtena, jak ukazuje obrázek 5.4. Při trénování je překrytí nulové, protože by to zvýšilo počet trénovacích dat jejich částečnou duplikací, což by negativně ovlivnilo délku trénování.



Obrázek 5.4: Segmentace nahrávek s nenulovým překrytím. Zde je vidět překrytí (`overlap`) o délce 1. Výsledný `offset` n–tého segmentu se vypočítá jako $\texttt{offset}_n = n * (seg\_len - overlap)$, kde $n = 0, 1, \ldots, N$.

5.3 Pomocné skripty

Během implementace se vyskytla potřeba zjednodušit si některé repetetivní úkony a jelikož upřednostňuji práci z terminálu, tak jsem si pro ně naprogramoval několik skriptů za použití jazyků Python3 a Bash.

Skript ***gupload.sh*** pro odesílání zdrojových souborů na *Google Drive*[11], ze kterého *Google Colab*[12] načítá data a kód, který lze následně po částech spouštět z grafického rozhraní. K interakci s Google Drive jsem použil nástroj *rclone*[13], který nabízí mnoho operací pro vzdálenou práci s Google Drive z terminálu. S pomocí rclone lze stahovat či nahrávat data, vzdáleně měnit adresářovou strukturu, upravovat soubory a podobně.

Skript ***gdownload.sh*** je určený pro snadné stahování dat z Google Drive pomocí nástroje rsync. Při zavolání stáhne adresář obsahující data z posledního trénování, nebo adresář, jehož název je předán jako argument skriptu v příkazové řádce. Stažený adresář obsahuje checkpointy, soubory obsahující výstup trénování, soubor obsahující dvojice hodnot `[zpracováno_nahrávek, loss]`, které lze jednoduše vykreslit do grafu, a nakonec soubor s výsledky testování obsahující hodnoty metrik pro testovaný checkpoint.

Dále jsem vytvořil několik jednoduchých skriptů pro rychlé spuštění trénování a testování modelu s různými přednastavenými parametry. Bylo tak možno mít mezi skripty rozdílné cesty k adresářům pro účely trénování a ladění chyb nebo třeba nastavení času, po kterém se proces má sám ukončit. Pro spuštění trénování modelu tak lze zavolat skript ***nntrain.sh***, pro testování skript ***nntest.sh*** a pro inferenci (separaci mluvčích na předané nahrávce) skript ***nninference.sh***. Skript ***nninference.sh*** přijímá na svém vstupu směs mluvčích a vyžaduje checkpoint s naučenou sítí. Jeho výstupem jsou dvě separované nahrávky.

[11] https://www.google.com/drive/
[12] https://colab.research.google.com/notebooks/intro.ipynb
[13] https://rclone.org/

Kapitola 6

Experimenty a vyhodnocení

Neuronové sítě představují mocný nástroj mnoha využití, který ale musí být naučen, jak danou činnost provádět. Trénování sítí je výpočetně náročný úkol, jehož náročnost se, krom dalších parametrů, může lišit v závislosti na velikosti datasetu a sítě. Obecně čím větší dataset a rozměr sítě, tím déle trénování trvá. Proto je podstatný výkon stroje, na jakém je síť učena. Je–li k dispozici výkonný stroj, dá se doba trénování výrazně snížit.

Po dostatečném natrénování probíhá vyhodnocení sítě. Výsledky trénování jsou měřeny metrikami, které se volí na základě problému, který se síť učila řešit. Výsledky testování sítě jsou následně porovnávány v závislosti na velikosti daného modelu. Velikost modelu je ovlivněna hyper–parametry, které určují například počet vrstev sítě, velikost segmentace a další vlastnosti.

Cílem experimentů je zjistit, které z hyper–parametrů mají největší vliv na kvalitu separace a tedy na výsledky vyhodnocení, a jak moc lze model zmenšit, aby stále dával dostatečně dobré výsledky separace. Tato kvalita je měřena metrikami STOI, SDR a PESQ, které byly popsány v kapitole 2.

[[zkratit pod to]]

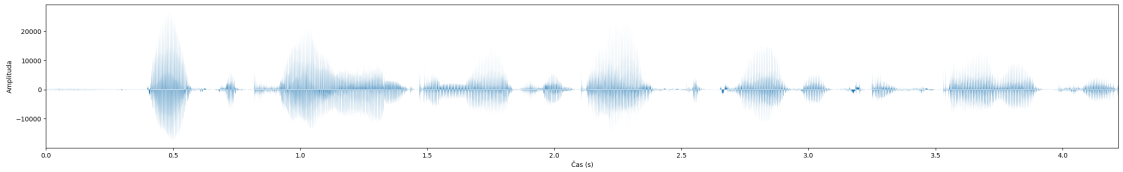
6.1 Dataset

Dataset je množina dat, na kterých je síť učena a testována. Dataset se dá rozdělit na tři podmnožiny, které jsou navzájem exkluzivní. Trénovací dataset je určen k trénování sítě a bývá největší. Validační dataset se prochází po zpracování trénovacího datasetu v průběhu trénování a slouží pro ověření, že hodnota loss na dosud neviděných datech stále klesá a nedochází tak k přetrénování. Při zpracování tohoto datasetu je síti zabráněno v učení. Poslední je testovací dataset, na kterém probíhá vyhodnocení sítě, tedy výpočet hodnot jednotlivých metrik.

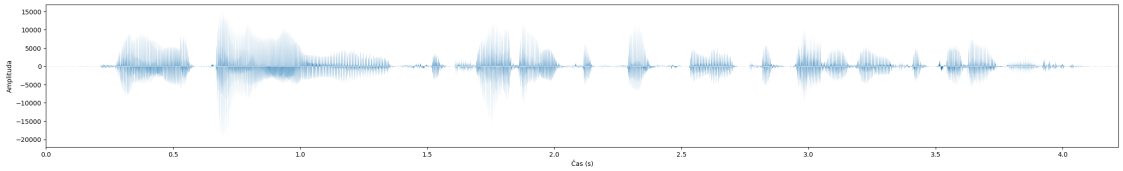
[[zkratit pod to]]

Trénování a vyhodnocení modelu proběhlo na množině jednokanálových nahrávek směsí dvou mluvčích. Množina byla vygenerována náhodným výběrem různých mluvčích z Wall Street Journal (WSJ0) [14] a vytvořením směsi mluvčích z těchto dvou náhodně vybraných nahrávek. Celková délka trénovacích dat je přes 10 hodin a přes 6 hodin validačních dat. Testovací data jsou vygenerována z 16ti, doposud neviděných, mluvčích. Nahrávky jsou převzorkovány na 8kHz a během trénování zarovnány na *zero means* a jednotkovou varianci. Skript pro generování datasetu lze nalézt v [1].

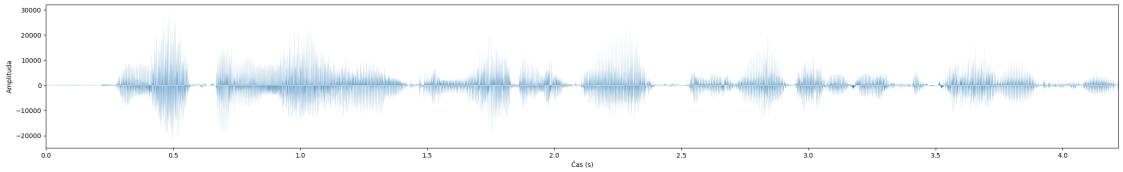
[1] https://www.merl.com/demos/deep-clustering



Obrázek 6.1: První mluvčí ze směsi



Obrázek 6.2: Druhý mluvčí ze směsi

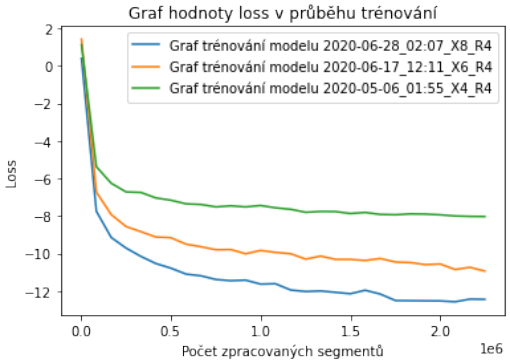


Obrázek 6.3: Ukázka směsi dvou mluvčích

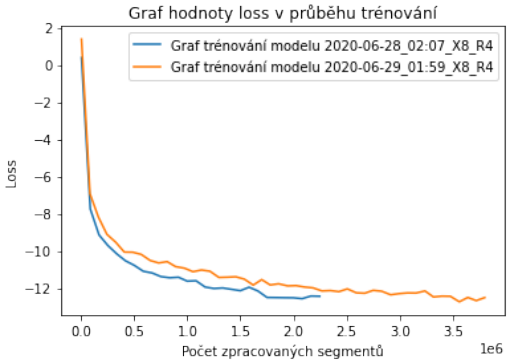
Lze si všimnout, že sečtením signálů jednotlivých mluvčích na obrázku 6.1 a 6.2 dostaneme signál původní směsi 6.3.

6.2 Průběh trénování

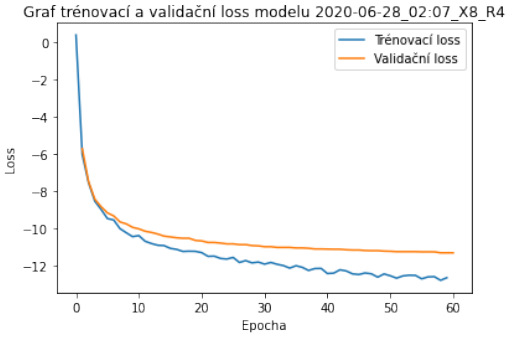
Bylo natrénováno 7 modelů s různými hodnotami zkoumaných hyper–parametrů, jak ukazují tabulky 6.3, 6.2 a 6.1. Hodnota X představuje počet konvolučních bloků, hodnota R představuje počet opakování sekvence X bloků a hodnota „segment length“ představuje počet vzorků v jednom segmentu. Trénování proběhlo na datasetu Wall Street Journal (WSJ0), který obsahuje 20000 trénovacích nahrávek následováno průchodem validačního datasetu, který obsahuje 10000 nahrávek, na konci každé epochy. K tomu byl použit stroj na Google Colab – Tesla P100-PCIE, který poskytuje rovněž výkonné GPU. Doba trénování jednotlivých modelů se pohybovala kolem 20 hodin. Všechny modely byly trénovány po 60 epoch s hodnotou `learning rate` = 0.001 a velikostí mini–batchů fixovanou na 50 segmentů v jednom mini–batchi. Hodnotu loss během trénování zobrazují, pro vybrané modely, grafy 6.4 a 6.5. Dále grafy 6.6 a 6.7 zobrazují trénovací i validační hodnoty loss na konci epoch v průběhu trénování vybraných modelů.



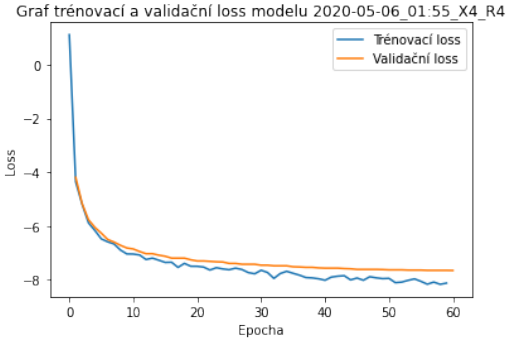
Obrázek 6.4: Hodnota loss při trénování modelů s různým počtem konvolučních bloků. Z grafu je patrné, že se zmenšujícím se počtem bloků se zhoršuje natrénovaný model.



Obrázek 6.5: Graf trénování modelů s rozdílnou délkou segmentů při trénování. Vzhledem k tomu, že je graf vykreslený v závislosti na počtu segmentů, tak model, který měl délku segmentů kratší (oranžový graf), tak jich zpracoval více, proto je delší než modrý.



Obrázek 6.6: Graf trénovací a validační loss pro každou epochu trénování. Model má rozměry $X = 8$, $R = 4$.



Obrázek 6.7: Graf trénovací a validační loss pro každou epochu trénování. Model má rozměry $X = 4$, $R = 4$.

6.3 Vliv parametrů X a R

[[mozna nejake fajnove grafy porovnani vysledky mezi jednotlivymi X, obrazky, grafy pro sdr, pesq, stoi]]

Rozměry modelů mají značný vliv na přesnost. V této části se zaměřím na vliv dvou parametrů – X a R. Parametr X ovlivňuje počet konvolučních bloků, což přímo ovlivňuje velikost celkové časové dilatace. Parametr R udává, kolikrát se sekvence X konvolučních bloků opakuje. Tabulky 6.1, 6.2 a 6.3 přehledně ukazují, jaké modely a o jakých kombinacích parametrů byly natrénovány.

Modely o kombinaci parametrů z tabulky 6.1 dosahují nejlepších výsledků. Výrazného zlepšení oproti menším modelům s parametrem dosahují u metriky SDR hodnoty až 12.71 dB. Modely si vedou oproti menším poměrně lépe i v metrice STOI, kde dosahují až hodnoty 2.71. Podle této tabulky se zdá, že parametr R nemá moc velký vliv na výslednou

Tabulka 6.1: Tabulka zobrazující vliv počtu opakování pro $X = 8$ konvolučních bloků na výsledky testování.

X	R	Délka segmentů [vzorky]	SDR [dB]	STOI	PESQ
8	2	32000	12.71	0.94	2.71
8	4	32000	12.45	0.93	2.69

přesnost sítě. Model s nižším počtem opakování dokonce dosahuje lehce lepších výsledků než model s počtem opakování $R = 4$. Důvodem by mohlo být to, že v závislosti na velikosti trénovaného modelu by mohla lépe vyhovovat menší či větší hodnota `learning rate`, než ta, která byla použita při trénování (0.001).

Tabulka 6.2: Tabulka zobrazující vliv počtu opakování pro $X = 6$ konvolučních bloků na výsledky testování.

X	R	Délka segmentů [vzorky]	SDR [dB]	STOI	PESQ
6	2	32000	10.80	0.92	2.52
6	3	32000	10.78	0.92	2.52
6	4	32000	10.69	0.92	2.52

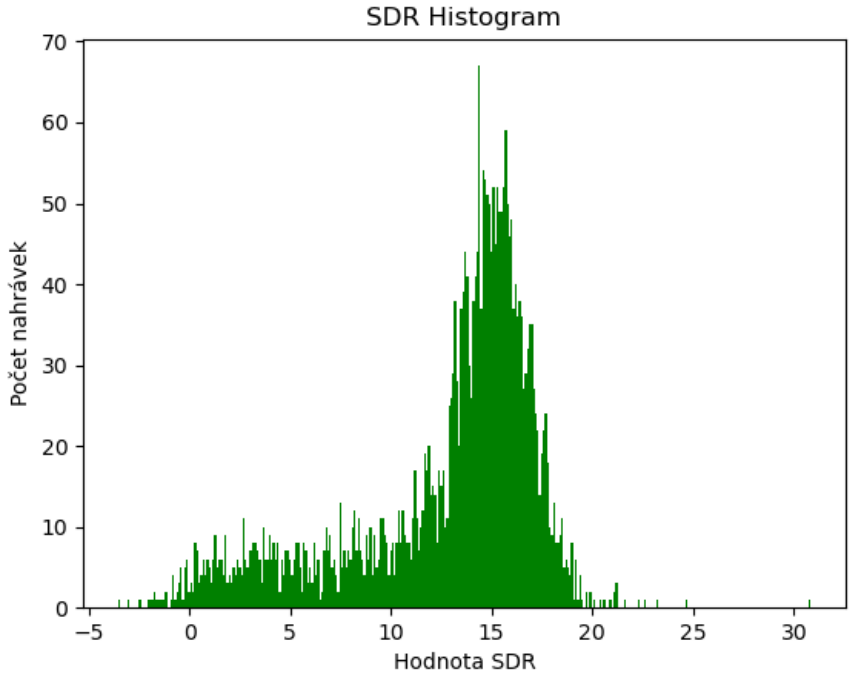
Tabulka 6.3: Tabulka zobrazující vliv počtu opakování pro $X = 4$ konvolučních bloků na výsledky testování.

X	R	Délka segmentů [vzorky]	SDR [dB]	STOI	PESQ
4	4	32000	9.23	0.89	2.23

Při zmenšování počtu konvolučních bloků X lze vidět znatelný pokles přesnosti ve sledovaných metrikách. Tabulky 6.2 a 6.3 obsahují rozměry těchto modelů a jejich dosaženou přesnost. V první tabulce pro model s velikostí $X = 6$ téměř nedochází k nějakému znatelnému zvýšení přesnosti při modifikaci parametru R. Znovu se tedy ukazuje, že parametr R nemá velký vliv na celkový výkon sítě. Druhá tabulka ukazuje hodnoty, zmenšíme–li model ještě více. Při parametru $X = 4$ má model nejnižší přesnost z mých natrénovaných modelů ve všech měřených metrikách, z nichž největšího rozdílu doznaly metriky SDR a PESQ.

Zdá se, že příčinou, proč má změna parametru X takový vliv na výkon, přímo souvisí s dilatačním faktorem, který se zvyšuje právě se zvyšujícím se počtem konvolučních bloků.

Přestože víme, že nejlépe si vede model o rozměrech $X = 8$ a $R = 2$ s jeho výslednou hodnotou SDR 12.71 dB ale tato hodnota nám neříká nic o výsledcích přesnosti jednotlivých směsí, protože je to průměrná hodnota ze všech nahrávek. Histogram 6.8 zobrazuje rozdělení hodnoty SDR v rámci zpracovaných nahrávek.



Obrázek 6.8: Histogram hodnot SDR pro jednotlivé separace.

Na první pohled je vidět největší koncentrace nahrávek kolem hodnoty SDR 15 dB. Některé nahrávky dokonce dosáhly hodnoty větší než 19 dB, což by také příjemně zvyšovalo celkovou přesnost modelu, ale vzhledem k levé části od největší koncentrace je vidět, že pro mnoho nahrávek dosahovala síť pouze přesnosti od 0 dB po 10 dB.

6.4 Vliv délky segmentů

Pro zjištění, zda–li má různá délka segmentů při trénování vliv na přesnost sítě, byly natrénovány dva modely o stejných rozměrech, jak ukazuje tabulka 6.4. Jak je vidět, jeden model při trénování segmentoval nahrávky na délku 16000 vzorků, zatímco druhý na 32000 vzorků.

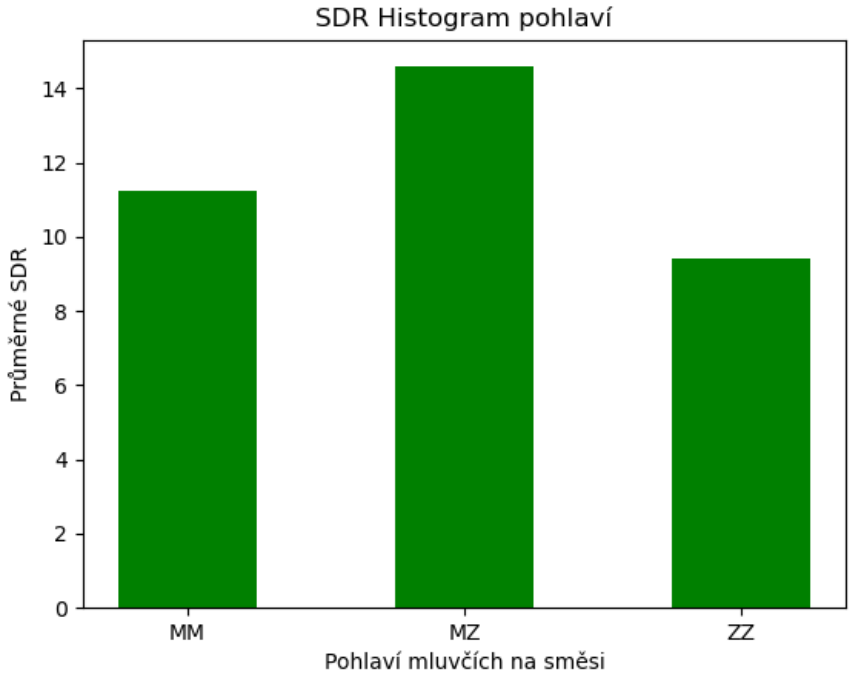
Tabulka 6.4: Tabulka zobrazující vliv délky segmentů na výsledky testování.

X	**R**	**Délka segmentů [vzorky]**	**SDR [dB]**	**STOI**	**PESQ**
8	4	32000	12.45	0.93	2.69
8	4	16000	12.49	0.93	2.7

Testování ukázalo, že model, který používal kratší segmenty, dosahoval zanedbatelně lepších výsledků u hodnoty SDR (12.49 dB), než model, který používal segmenty delší a dosahoval hodnoty SDR 12.45 dB. Hodnoty ostatních metrik dosahují téměř stejných hodnot v případě obou modelů.

6.5 Úspěšnost s ohledem na pohlaví mluvčích

Testovací nahrávky jsou náhodné kombinace mužských a ženských mluvčích. Pro separaci dvou mluvčích tedy mohou vzniknout až 3 typy nahrávek na základě pohlaví mluvčího – muž–muž, muž–žena a žena–žena. Jednotlivé kombinace mluvčích mohou být pro síť rozdílně náročné pro separaci. Na mém nejlepším modelu o rozměru $X = 8$ a $R = 2$ jsem zkoumal, jaké výsledky mají jednotlivé kombinace.



Obrázek 6.9: Histogram průměrné hodnoty SDR napříč různou kombinací pohlaví.

Z grafu 6.9 lze vypozorovat, jaký vliv má kombinace pohlaví na přesnost sítě. Nejlepších výsledků síť dosahuje v případě, že na nahrávce jsou zastoupena obě pohlaví. V takovém případě model dosahuje hodnoty SDR 14.6 dB. Oproti tomu, v případě stejných pohlaví si síť vede poměrně hůř. V případě kombinace dvou žen síť dosahuje hodnoty SDR pouze 9.4 dB a v případě kombinace dvou mužů jen o něco lépe 11.2 dB. Výsledná hodnota je průměrem hodnot v dané kombinaci pohlaví.

Výsledkem pozorování je, že síť od sebe separuje mluvčí mnohem lépe v případě, že jsou zastoupena na směsi rozdílná pohlaví.

6.6 Výsledky referenčního modelu TasNet

V referenční studii [24] byly modely trénovány 100 epoch s různými hyper–parametry. Já se zaměřím na takové, se kterými experimentuji v rámci mých modelů. Modely jsou v nekauzální konfiguraci a používají batch normalizaci (BN).

Nejlepší model ze studie TasNet s hodnotami parametrů $X = 8$ a $R = 4$ dosahuje hodnoty $SDR = 14.7$ dB při délce filtrů $L = 20$ vzorků. Hodnoty $SDR = 13.0$ dB dosahuje při hodnotě parametru $X = 7$ a délce filtrů $L = 40$ vzorků. Po zvětšení velikosti filtrů

a zmenšení počtu konvolučních bloků (což zmenšuje také hodnotu dilatace) je patrný pokles hodnoty SDR. Při dalším zmenšování modelu dále klesá hodnota SDR, jak zobrazuje tabulka 6.5.

Tabulka 6.5: Tabulka ukazující výsledky testovaní referenčních modelů TasNet v závislosti na konfiguraci.

L [vzorky]	**X**	**R**	**Normalizace**	**SDR [dB]**	**Receptive field [s]**
20	8	4	BN	14.7	2.55
40	7	4	BN	13.0	2.55
40	8	2	BN	12.6	2.56
40	6	4	BN	12.1	1.27
40	7	2	BN	11.7	1.28

Můj nejlepší model v porovnání s modely ze studie dosahuje hodnoty SDR 12.71 dB, jak ukazuje tabulka 6.1. Důvodem horší přesnosti může být nedostatečné natrénování modelu (60 epoch oproti jejich 100 epochám), použitá hodnota learning rate při trénování či implementační rozdíly. Přesto tato hodnota ukazuje, že je model již schopen obstojně separovat nahrávky.

6.7 Shrnutí výsledků

[[shrnuti a porovnani jednotlivych zmen a vysledku mezi sebou komplexne. Pripadne porovnani vysledku s ref studii, Okomentovat, co je tedy dulezity parametr, a ktery jde zmensit a neni moc dulezity z pohledu vysledku evaluace. okomentovat take pohlavi vysledky, prumery sdr, mean, a pod.]]

Během testování a experimentů jsem sledoval vliv parametrů X, R a délka segmentů na kvalitu sítě, která je vyhodnocována metrikami SDR, STOI a PESQ. Nejlepších hodnot dosahuje model rozměrově téměř největší (X = 8, R=2), a to SDR 12.71 dB, STOI 0.94 a PESQ 2.71. Při zmenšování hodnoty X se ale přesnost razantně snižuje. Parametr X má tedy dle výsledků největší vliv na přesnost sítě. Oproti tomu změna parametru R přesnost téměř neovlivnila, stejně jako délka segmentů při trénování sítě. Vzhledem k dosaženým přesnostem bych vybral, jako kandidátní, modely o velikosti $R = 2$ a $X = 8$ nebo $X = 6$. Tyto modely dosahují dostatečně dobré kvality separovaných nahrávek.

Graf rozdělení SDR ukázal, že přestože se mnoho hodnot pohybuje lehce pod průměrnou hodnotou SDR daného modelu, tak mnoho separací dosáhlo poměrně větší přesnosti a mnoho separací poměrně menší přesnosti, než je průměrná.

Zajímavý poznatek vyplynul ze sledování hodnoty SDR v rámci ruzných kombinací pohlaví na separovaných nahrávkách. Nejvyšší hodnotu (14.6 dB) měly směsi, kde se nacházela dvojice rozdílného pohlaví. Homogenní dvojice dosahovaly mnohem menší přesnosti.

6.8 Možná rozšíření a navrhnutá vylepšení

[[k poslednimu bodu zadani, ze diskutujte – tak lze jen napsat, ze by mohlo toto toto, proste ukazat, ze jsem se nad tim minimalne zamyslel. mrknout na Tasnet studii, kde to resili jak by to slo vyresit, pripadne jestli bych nasel nejake studie ktere diskutuji o zlepseni. Kdyz bude cas, tak to pak i vyzkouset a porovnat.]]
[[Procist studii jaky oni tam navrhuji vylepseni]]

Práce by mohla být rozšířena o variabilnější dataset pro trénování modelů. Zásadní problém při použití mých naučených modelů v reálném scénáři je ten, že trénovací dataset není variabilní, když nepočítám různé kombinace mluvčích. V případě, že by nahrávky řečníků byly pořízeny ve variabilnějším prostředí s rozdílnou technikou, mikrofony a různým ruchem v pozadí (další hlasy, hudba či tleskot), model by tak byl lépe využitelný. Trénování by také mohlo být rozšířeno o nahrávky s větším počtem mluvčích, což by ale vyžadovalo rozsáhlejší implementační úpravy.

Přímo na model by se dalo navázat další neuronovou sítí, která by se učila identifikovat řečníka či analyzovat vlastnosti jeho hlasu a na základě toho provádět další operace. Další možností je přepis separované řeči mluvčích na text.

Kapitola 7

Závěr

Cílem práce bylo implementovat síť podle architektury TasNet pro separaci mluvčích v časové doméně a porovnat vliv velikosti sítě na kvalitu separace. Síť byla implementována za pomoci frameworku PyTorch a jazyku Python a natrénována na datasetu obsahujícím jednokanálové směsi dvou mluvčích. Bylo natrénováno 7 modelů, které se od sebe lišily počtem opakujících se konvolučních bloků, velikostí časové dilatace a délkou vstupních segmentů směsí. Pro účel vyhodnocení modelů byly použity metriky SDR, STOI a PESQ, které se používají pro měření kvality separace.

Experimenty ukázaly, že během testování nejlépe dopadla síť, která měla 8 konvolučních bloků po 2 opakováních, učena s délkou vstupních segmentů $L = 4$ sekundy. Tento model dosáhl po 60 epochách trénování hodnoty až 12.71 dB a tím se stal nejúspěšnějším modelem. Při fyzickém poslechu separovaných nahrávek není slyšet téměř žádný náznak druhého mluvčího. Oproti tomu, nejméně přesný model měl pouze 4 konvoluční bloky, 4 opakování a při délce segmentů $L = 4$ sekundy dosahoval hodnoty SDR pouze 9.23 dB. Přestože podmínky pro trénování nebyly ideální, tak bylo natrénováno několik úspěšných modelů.

Zkoušel jsem separovat také nahrávky, které se v datasetu vůbec nenacházely, ale výsledek se nedá hodnotit jako úspěšný, jelikož hraje velkou roli prostředí, mikrofon, šum v pozadí a další vlivy, na které byla neuronová síť naučena. Tento problém by se dal překonat rozšířením trénovacího datasetu o větší škálu nahrávek mluvčích, které by byly pořízeny z různých zařízení a v různě rušném prostředí.

[[co jak dopadlo, vysledky a vyhodnoceni velikosti modelu a jaky byl nejlepsi]] [[Doplnit ještě neco eh]]

Literatura

[1] Ba, J. L., Kiros, J. R. a Hinton, G. E. Layer normalization. *ArXiv preprint arXiv:1607.06450*. červenec 2016. Dostupné z: https://arxiv.org/abs/1607.06450.

[2] Bai, S., Kolter, J. Z. a Koltun, V. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. *ArXiv preprint arXiv:1803.01271*. 2018, abs/1803.01271. Dostupné z: http://arxiv.org/abs/1803.01271.

[3] Bechtold, B. Analyzing Speech Signals in Time and Frequency. *Basti's Scratchpad on the Internet* [online], 20. září 2019 [cit. 2020-07-25]. https://bastibe.de/2019-09-20-analyzing-speech-signals-in-time-and-frequency.html. Dostupné z: https://bastibe.de/. Path: Projects; Other posts; Analyzing Speech Signals in Time and Frequency.

[4] Bengio, Y., Simard, P. a Frasconi, P. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*. březen 1994, sv. 5, č. 2, s. 157–166. DOI: 10.1109/72.279181. ISSN 1941-0093.

[5] Bhattarai, S. What is Gradient Descent in machine learning? *A TECH BLOG* [online], 22. červen 2018 [cit. 2020-07-26]. https://saugatbhattarai.com.np/what-is-gradient-descent-in-machine-learning/. Dostupné z: https://saugatbhattarai.com.np/. Path: search; What is Gradient Descent.

[6] Chen, S. a Cheng, M.-J. Building an Adaptive Evaluation System: A Design Education Application. *Computer-Aided Design & Applications*. leden 2006, sv. 3, s. 49–58. DOI: 10.1080/16864360.2006.10738441.

[7] Choi, S., Cichocki, A., Park, H.-M. a Lee, S.-Y. Blind source separation and independent component analysis: A review. *Neural Information Processing-Letters and Reviews*. leden 2005, sv. 6, č. 1, s. 1–57.

[8] Chollet, F. Xception: Deep Learning With Depthwise Separable Convolutions. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, červenec 2017, s. 1800–1807. DOI: 10.1109/CVPR.2017.195. ISSN 1063-6919.

[9] Getzmann, S., Jasny, J. a Falkenstein, M. Switching of auditory attention in "cocktail-party" listening: ERP evidence of cueing effects in younger and older adults. *Brain and Cognition*. Leibniz Research Centre for Working Environment and Human Factors, D-44139 Dortmund, Germany: [b.n.]. 2017, sv. 111, s. 1–12. DOI:

https://doi.org/10.1016/j.bandc.2016.09.006. ISSN 0278-2626. Dostupné z: http://www.sciencedirect.com/science/article/pii/S0278262616302408.

[10] Glorot, X. a Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In: Teh, Y. W. a Titterington, M., ed. *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. Chia Laguna Resort, Sardinia, Italy: PMLR, 13–15 May 2010, sv. 9, s. 249–256. Proceedings of Machine Learning Research. Dostupné z: http://proceedings.mlr.press/v9/glorot10a.html.

[11] Goodfellow, I., Bengio, Y. a Courville, A. *Deep Learning*. The MIT Press, 2016. ISBN 9780262035613. http://www.deeplearningbook.org.

[12] He, K., Zhang, X., Ren, S. a Sun, J. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In: *2015 IEEE International Conference on Computer Vision (ICCV)*. Prosinec 2015, s. 1026–1034. DOI: 10.1109/ICCV.2015.123. ISSN 2380-7504. Dostupné z: https://arxiv.org/abs/1502.01852.

[13] He, K., Zhang, X., Ren, S. a Sun, J. Deep Residual Learning for Image Recognition. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. červen 2016, s. 770–778. DOI: 10.1109/CVPR.2016.90. ISSN 1063-6919. Dostupné z: https://arxiv.org/abs/1512.03385.

[14] Hershey, J. R., Chen, Z., Le Roux, J. a Watanabe, S. Deep clustering: Discriminative embeddings for segmentation and separation. In: IEEE. *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2016, s. 31–35. Dostupné z: http://arxiv.org/abs/1508.04306.

[15] Hidri, A., Meddeb, S., Alaqeeli, A. a Amiri, H. Beamforming Techniques for Multichannel audio Signal Separation. *CoRR*. 2012, abs/1212.6080. Dostupné z: http://arxiv.org/abs/1212.6080.

[16] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W. et al. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *CoRR*. 2017, abs/1704.04861. Dostupné z: http://arxiv.org/abs/1704.04861.

[17] Ioffe, S. a Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *ArXiv preprint arXiv:1502.03167*. 2015, abs/1502.03167. Dostupné z: http://arxiv.org/abs/1502.03167.

[18] Kelleher, J. *Deep learning / John D. Kelleher*. Cambridge, Massachusetts, London, England: The MIT Press, srpen 2019. The MIT Press Essential Knowledge series. ISBN 9780262537551.

[19] Kim, J., El-Khamy, M. a Lee, J. End-to-End Multi-Task Denoising for joint SDR and PESQ Optimization. *CoRR*. leden 2019, abs/1901.09146. Dostupné z: http://arxiv.org/abs/1901.09146.

[20] Krizhevsky, A., Sutskever, I. a Hinton, G. E. ImageNet Classification with Deep Convolutional Neural Networks. In: Pereira, F., Burges, C. J. C., Bottou, L. a Weinberger, K. Q., ed. *Advances in Neural Information Processing Systems*

25. Curran Associates, Inc., 2012, s. 1097–1105. Dostupné z: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[21] Lea, C., Vidal, R., Reiter, A. a Hager, G. D. Temporal Convolutional Networks: A Unified Approach to Action Segmentation. In: Hua, G. a Jégou, H., ed. *Computer Vision – ECCV 2016 Workshops*. Cham: Springer International Publishing, 2016, s. 47–54. ISBN 978-3-319-49409-8. Dostupné z: https://arxiv.org/abs/1608.08242.

[22] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E. et al. Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation*. MITP. prosinec 1989, sv. 1, č. 4, s. 541–551. DOI: 10.1162/neco.1989.1.4.541. ISSN 0899-7667.

[23] Lee, D. D. a Seung, H. S. Algorithms for Non-negative Matrix Factorization. In: Leen, T. K., Dietterich, T. G. a Tresp, V., ed. *Advances in Neural Information Processing Systems 13*. MIT Press, 2001, s. 556–562. Dostupné z: http://papers.nips.cc/paper/1861-algorithms-for-non-negative-matrix-factorization.pdf.

[24] Luo, Y. a Mesgarani, N. TasNet: Surpassing Ideal Time-Frequency Masking for Speech Separation. *CoRR*. září 2018, abs/1809.07454. v2. Dostupné z: http://arxiv.org/abs/1809.07454.

[25] Neal, R. M. Connectionist learning of belief networks. *Artificial Intelligence*. 1992, sv. 56, č. 1, s. 71–113. DOI: https://doi.org/10.1016/0004-3702(92)90065-6. ISSN 0004-3702. Dostupné z: http://www.sciencedirect.com/science/article/pii/0004370292900656.

[26] Negro, F., Muceli, S., Castronovo, A. M., Holobar, A. a Farina, D. Multi-channel intramuscular and surface EMG decomposition by convolutive blind source separation. *Journal of Neural Engineering*. IOP Publishing. únor 2016, sv. 13, č. 2, s. 026027. DOI: 10.1088/1741-2560/13/2/026027. Dostupné z: http://dx.doi.org/10.1088/1741-2560/13/2/026027.

[27] Nwankpa, C., Ijomah, W., Gachagan, A. a Marshall, S. Activation Functions: Comparison of trends in Practice and Research for Deep Learning. *CoRR*. 2018, abs/1811.03378. Dostupné z: http://arxiv.org/abs/1811.03378.

[28] Pekel, E. a Soner Kara, S. A comprehensive review for artificial neural network application to public transportation. *Sigma: Journal of Engineering & Natural Sciences/Mühendislik ve Fen Bilimleri Dergisi*. březen 2017, sv. 35, č. 1, s. 157–179.

[29] Qian, W. C. C. X. e. a. Past review, current progress, and challenges ahead on the cocktail party problem. *Frontiers Inf Technol Electronic Eng 19*. Springer Nature. leden 2018, sv. 19, č. 1, s. 40–63. DOI: 10.1631/FITEE.1700814. ISSN 2095-9230. Dostupné z: https://doi.org/10.1631/FITEE.1700814.

[30] Raffel, C., McFee, B., Humphrey, E. J., Salamon, J., Nieto, O. et al. Mir_eval: a transparent implementation of common MIR metrics. In: *In Proceedings of the 15th International Society for Music Information Retrieval Conference, ISMIR*. 2014. Dostupné z: https://colinraffel.com/publications/ismir2014mir_eval.pdf.

[31] Rix, A., Beerends, J., Hollier, M. a Hekstra, A. Perceptual Evaluation of Speech Quality (PESQ): A New Method for Speech Quality Assessment of Telephone Networks and Codecs. In:. únor 2001, sv. 2, s. 749–752 vol.2. DOI: 10.1109/ICASSP.2001.941023. ISBN 0-7803-7041-4.

[32] Ruder, S. An overview of gradient descent optimization algorithms. *ArXiv*. 2016, abs/1609.04747. Dostupné z: http://arxiv.org/abs/1609.04747.

[33] Taal, C. H., Hendriks, R. C., Heusdens, R. a Jensen, J. A short-time objective intelligibility measure for time-frequency weighted noisy speech. In: *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*. Březen 2010, s. 4214–4217. DOI: 10.1109/ICASSP.2010.5495701. ISSN 2379-190X.

[34] Taal, C. H., Hendriks, R. C., Heusdens, R. a Jensen, J. An Algorithm for Intelligibility Prediction of Time–Frequency Weighted Noisy Speech. *IEEE Transactions on Audio, Speech, and Language Processing*. září 2011, sv. 19, č. 7, s. 2125–2136. DOI: 10.1109/TASL.2011.2114881. ISSN 1558-7924.

[35] Venkataramani, S., Higa, R. a Smaragdis, P. Performance based cost functions for end-to-end speech separation. In: IEEE. *2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. Listopad 2018, s. 350–355. DOI: 10.23919/APSIPA.2018.8659758. ISSN 2640-0103.

[36] Vincent, E., Gribonval, R. a Févotte, C. Performance measurement in blind audio source separation. *IEEE Transactions on Audio, Speech and Language Processing*. Institute of Electrical and Electronics Engineers. 2006, sv. 14, č. 4, s. 1462–1469. Dostupné z: https://hal.inria.fr/inria-00544230.

[37] Wang, D. a Chen, J. Supervised Speech Separation Based on Deep Learning: An Overview. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.* IEEE Press. říjen 2018, sv. 26, č. 10, s. 1702–1726. DOI: 10.1109/TASLP.2018.2842159. ISSN 2329-9290. Dostupné z: https://doi.org/10.1109/TASLP.2018.2842159.

[38] Yoshii, K., Tomioka, R., Mochihashi, D. a Goto, M. Beyond NMF: Time-Domain Audio Source Separation without Phase Reconstruction. In: *ISMIR*. 2013, s. 369–374. Dostupné z: http://ismir2013.ismir.net/wp-content/uploads/2013/09/32_Paper.pdf.

[39] Zaeemzadeh, A., Rahnavard, N. a Shah, M. Norm-Preservation: Why Residual Networks Can Become Extremely Deep? *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2020, s. 1–1. DOI: 10.1109/TPAMI.2020.2990339. ISSN 1939-3539. Dostupné z: https://arxiv.org/abs/1805.07477v5.

[40] Zeiler, M. D., Ranzato, M., Monga, R., Mao, M., Yang, K. et al. On Rectified Linear Units For Speech Processing. In: IEEE. *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. Vancouver, BC, Canada: [b.n.], 2013, s. 3517–3521. DOI: 10.1109/ICASSP.2013.6638312. ISSN 2379-190X.