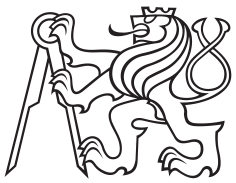
Diplomová práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra teorie obvodů

Detektor řečové aktivity na bázi DNN

Bc. Mojmír Lakosil
Studijní program: Komunikace, multimédia a elektronika
Studijní obor: Komunikační systémy

Červen 2017
Vedoucí práce: doc. Ing. Petr Pollák, CSc.

Poděkování / Prohlášení

Chtěl bych poděkovat vedoucímu diplomové práce panu doc. Ing. Petru Pollákovi, CSc. za odborné vedení, za pomoc a rady při zpracování této práce.

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Abstrakt / Abstract

Tato práce řeší problematiku detekce řečové aktivity s použitím hlubokých neuronových sítí. V práci jsou stručně popsány obecné principy detekce řeči, základní používané algoritmy, hlavní pozornost však je věnována analýze funkčnosti detektorů na bázi neuronových sítí. Je studován přínos použití více vrstev neuronové sítě a různých učících algoritmů a provedeno srovnání s detektory na bázi základních třívrstvých sítí. Detektory jsou realizovány s využitím tří různých frameworků, konkrétně v MATLABu, v neural2D toolkitu a ve speciálním toolkitu pro vývoj rozpoznávačů spojité řeči Kaldi. Zdrojové kódy realizovaných detektorů jsou součástí práce a jsou přiloženy na CD-ROM. Bylo také analyzováno použití různých řečových příznaků na vstupu sítě a jako optimální nastavení se ukázalo zřetězení vektorů 13 mel-frekvenčních kepstrálních koeficientů v kontextu 10 předcházejících resp. 10 následujících krátkodobých segmentů. V experimentální části je ověřena a vyhodnocena úspěšnost detekce na dostupných reprezentativních řečových datech z databáze TIMIT resp. QUT-TIMIT. Detekce byla testována v různých prostředích (různým typem aditivního šumu) a s různým SNR. Byla analyzována chybovost detekce v závislosti na zastoupení signálů stejného či podobného typy v souboru signálů pro učení sítě. U realizovaných detektorů bylo dosaženo přesnosti detekce na nezašuměném signálu těsně pod 2%, při 15 dB SNR přibližně 4% a při 0 dB SNR přibližně 5% chybovosti. Experimentálně byl potvrzen předpoklad, že v zašuměném prostředí detekuje optimálně navržená a natrénovaná hluboká neuronová síť hlas lépe než základní třívrstvá síť. V případě čistého záznamu řečového signálu, nebyl přínos DNN oproti třívrstvé síti tak významný.

This thesis deals with detection of voice activity based on deep neural networks. It briefly describes classical approaches to voice activity detection and basic VAD algorithms. Its primary focus is on analysis of voice activity detection based on neural networks. This thesis discusses advantages of deep neural network and comparison to basic three layer neural networks. Three implementations of voice activity detectors were developed as part of this work, which were based on three frameworks, namely Matlab, neural2D and Kaldi. Source codes of these detectors are on the attached CD-ROM. Usage of various speech features at the input of the ANN was also analyzed and it was proved that 13 MFCC, 10 short-term segments spliced (previous and consecutive ones) as optimal for voice detection. The speech detection was tested and evaluated using TIMIT and QUT-TIMIT speech databases in the experimental part of the work. The detection was tested in various environments (with various additive noise) and various SNR. The dependency of detection error on coverage of signals of various type in training dataset was also analyzed. The detection error rate achieved with the developed detectors was below 2% for signal with minimum noise background, about 4% with 15 dB SNR and about 5% with 0 dB SNR signal. It was shown that the well designed and trained DNN was more successful than basic three layer ANN in the case of performance in noisy environment. In the case of clear signal, without any noise added, the DNN produced similar error rate as basic three layer ANN.

Title translation: DNN-Based Voice Activity Detector

Obsah /

Tabulky / Obrázky

Kapitola 1
Úvod

Pro člověka je vzájemná komunikace prostřednictvím řeči s ostatními lidmi nejjednodušší a nejefektivnější metodou přenosu informací. Při komunikaci člověka s počítačem se jedná o velmi náročnou úlohu, kterou se zatím nepodařilo dokonale zvládnout. V dnešní době komunikace mezi počítači běžně dosahuje rychlostí 1 Gb/s, což odpovídá v závislosti na kódování 125 000 000 znaků za sekundu. Přitom komunikace člověka s počítačem stále probíhá většinou prostřednictvím klávesnice, kde přenosová rychlost u průměrného uživatele je několik desítek slov za minutu. Pro většinu uživatelů by plnohodnotné rozpoznávání řeči počítačem přineslo řádové zrychlení vkládání dat, mnohem vyšší uživatelský komfort a pohotovost (ne všude máme k dispozici klávesnici). Počítačové zpracování hlasu je nutným předpokladem hromadného zpracování (vyhledávání, klasifikace atd.) obrovského množství telefonních hovorů dle potřeb armádních a státních bezpečnostních agentur. Kromě technologických společností a akademické sféry se výzkumu zpracování řeči věnovaly také státní a zejména vojenské agentury (např. v USA Defence Advance Research Project Association). Již v sedmdesátých letech se začaly používat techniky porovnávání a kvantitativního porovnávání promluv, modelování nelineární proměnlivosti tempa řeči s využitím dynamického programování.

V letech 1971 až 1976 byly v rámci projektu DARPA-SUR (Speech Understanding Research) navrženy a experimentálně ověřeny systémy klasifikátoru mluvené řeči a moduly pro porozumění významu posloupnosti slov na principu umělé inteligence. V průběhu osmdesátých let se rozvinula technika klasifikace řeči na statistickém přístupu. Možnost rozpoznávat souvislou řeč se slovníkem až stovek tisíc slov poskytl způsob modelování řeči na principu Markovových modelů, které modelují nikoli celá slova ale pouze fonémy, trifony apod.

Tento systém je využíván i v současnosti, ačkoliv nyní i díky výrazně vyššímu výkonu současných počítačů, dosahují lepších výsledů systémy na principech hlubokých neuronových sítí. Často se ale oba přístupy kombinují. Počítačové zpracování hlasu není jen o rozpoznávání mluvené řeči, ale použití je mnohem širší. Dalšími aplikacemi jsou např. rozpoznávání mluvčího, potlačování šumu a echa, zvýrazňování řeči. S tím jak se technologie vylepšuje, vznikají nové způsoby užití. Poslední dobou se začínají komerčně uplatňovat technologie jako hlasová autentizace, emoční analýza apod. Pro většinu těchto technologií je nutnou součástí kvalitní detektor řečové aktivity. V mnoha případech nepatrným zlepšením parametrů tohoto detektoru dosáhneme výrazného zlepšení aplikace.

Detektory řečové aktivity určují začátek a konec řečové aktivity, resp. odlišují řečový signál od pozadí. Při potlačování šumu pauzy v řeči využíváme pro získávání parametrů šumu pozadí. V komunikačních systémech využíváme detektorů řečové aktivity k přenosu pouze dat za přítomnosti řeči. V době nepřítomnosti řečové aktivity je potom generován umělý šum (comfort noise nebo comfort tone). Tento princip se používá např. v IP telefonii v udp protokolu Real-time Transport Protocol (RTP), který využívá "CN" payload type, při využití kodeků, které přímo nepodporují umělý šum (G.711,G.726, G.727, G.728, G.722). Existují však kodeky s implementovanou podporou detekce ře-

čové aktivity a generující umělý šum na straně přijímače. Detektory řečové aktivity často označujeme zkratkou VAD z anglického Voice Activity Detector.

Cílem této práce je právě návrh takového detektoru řečové aktivity na principu hluboké neuronové sítě. Součástí návrhu je nalezení vhodného frameworku, topologie neuronové sítě, její parametry, způsob trénování, předtrénování, nalezení vhodných řečových parametrů a jejich dimenzí. Důležitou součástí návrhu je také metodika ověření přesnosti detekce a vyhodnocení dosažených parametrů.

Kapitola 2
Řečový signál

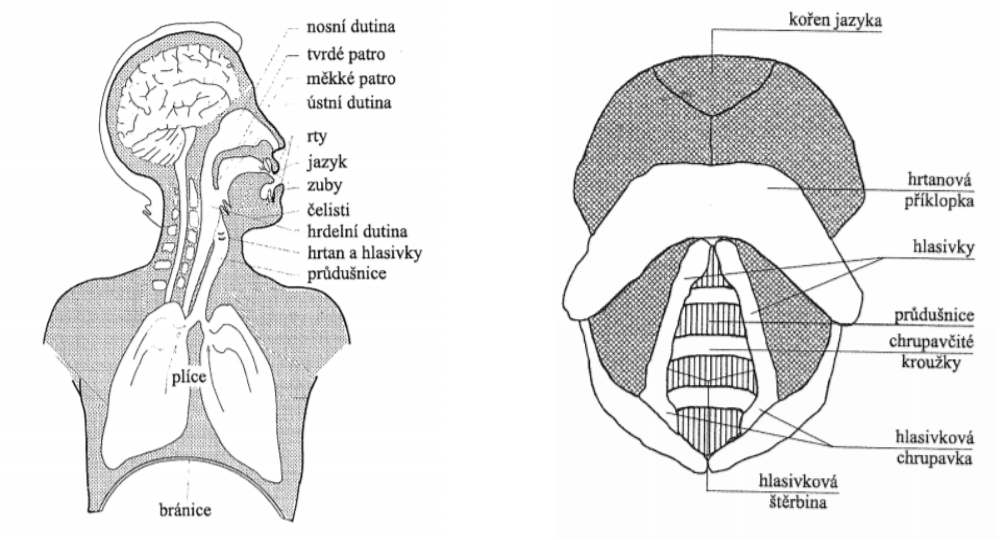
Pro vysvětlení zpracování detekce hlasu v této práci je nejdříve potřebné popsat mechanismus vzniku řeči a její vlastnosti. Mluvená řeč se přenáší komunikačním kanálem akustickými vlnami. Řečový signál je vlnění prostředí v oboru slyšitelných frekvencí. Komunikačním kanálem rozumíme prostředí, kterým je akustický signál přenášen od zdroje, úst až k příjemci, uším. V akustickém signálu řeči je kromě akustické složky nejdůležitější lingvistická informace. Ta je dána fonetickou, morfologickou, syntaktickou a sémantickou strukturou, protože vyjadřuje význam sdělované myšlenky. Akustický signál obsahuje také specifické informace o mluvčím (např. intonaci, rytmus řeči, barvu hlasu atd.) dané charakteristikami hlasového traktu řečníka.

2.1 Vznik řeči a její číslicová reprezentace

Řeč se v lidském těle vytváří prostřednictvím řečových (artikulačních) orgánů. Z hlediska tvorby řeči tyto orgány tvoří hlasový trakt. Hlasový trakt lze rozdělit na tři základní ústrojí: dechové, hlasové a artikulační. Hlasové ústrojí je uloženo v hrtanu, který je plícemi spojen průdušnicí. Nejdůležitější částí hlasového ústrojí jsou hlasivky, které se nacházejí v hrtanové dutině za „ohryzkem“. Hlasivky tvoří dvě ostré slizniční řasy, umístěné v nejužším místě hrtanu. Výdechový proud vzduchu z dechového ústrojí postupuje z plic průdušnicí k hrtanu. Zde se stažené hlasivky pod tlakem vzduchu rozkmitají. Hlasivky se střídavě otevírají a uzavírají. Takto vzniklý periodický proud zvukových pulzů tvoří základ lidského hlasu. Označuje se jako základní nebo hlasivkový tón. Jeho frekvence se označuje F_0 a nazývá se **frekvence základního hlasivkového tónu**. Frekvence kmitání je různá u dětí a dospělých, ale i u žen a mužů. U mužů bývá v rozmezí 80 – 160 Hz, u žen 150 – 300 Hz a u dětí 200 – 600 Hz.

Posledním ústrojím podílejícím se na tvorbě řeči je **artikulační ústrojí**. Skládá se z nadhrtanových dutin (hrdelní, ústní a nosní) a z artikulačních orgánů. Mezi nejvýznamnější artikulátory patří jazyk, rty a měkké patro. Mezi další artikulátory řadíme zuby, tvrdé patro, čelisti a hrtan. Při průchodu základního hlasivkového tónu nadhrtanovými dutinami dochází vlivem rezonance uvnitř dutin ke změně rozložení akustické energie. Ta se soustředí kolem určitých frekvencí, které nazýváme **formantové frekvence**. Oblasti zesílení akustické energie se nazývají formanty a označují se $F_1, F_2, F_3 ..., F_n$.

Artikulátory ovlivňují průchod vzduchu nadhrtanovými dutinami. Svým pohybem mohou průchod vzduchu na různých místech zúžit nebo uzavřít. Výdechový proud vzduchu pak při průchodu přes různé vytvořené překážky vytváří šum různého druhu. Tento šum je potom základem tvoření souhlásek [1].



(a) (b)

Obrázek 2.1. Hlasový trakt (a) a hrtan s hlasivkami (b)

Pro snímání řeči se používá mikrofon, který převádí řeč na elektrický signál. Dále se tento signál digitalizuje a zpracovává jako posloupnost řečových vzorků.

Digitalizace hlasového analogového signálu spočívá v jeho vzorkování a kvantování. Vzorkování je transformace signálu spojitého v čase na posloupnost vzorků. Vzorkování probíhá v časových okamžicích označovaných jako vzorkovací perioda. Základními parametry digitálního řečového signálu jsou potom vzorkovací kmitočet f_s (nebo vzorkovací perioda T_s) a počet bitů na jeden vzorek a typ kódu. Kvantování může být lineární, logaritmické, případně v různém kódu.

Při vzorkování řečového signálu jsou podstatné informace obsaženy v kmitočtovém rozsahu do 8 kHz. Pro kvalitní řečový digitální vzorek tedy vystačíme dle Nyquistova vzorkovacího teorému s f_s 16 kHz. V telefonních aplikacích vystačíme s kmitočtovým rozsahem 300 Hz až 3 400 Hz. Používaná f_s je potom 8 kHz.

Počet kvantovacích kroků určuje dosažitelnou dynamiku řeči. Ta by měla být 60 dB. V případě lineárního kvantování tedy potřebujeme 11 až 12 bitů. Běžně se používá 16 bitového PCM lineárního kvantování poskytující dynamiku 90 dB. Toto kvantování je použito i u wav formátu, se kterým budu v této práci pracovat.

V časové oblasti se řečový signál reprezentuje pomocí časového průběhu vlny, který zobrazuje vývoj amplitudy signálu v čase. Časové průběhy odhalují střídání téměř periodických úseků **(kvaziperiodicita)** s frekvencí F_0 s velkou amplitudou a šumových úseků s poměrně malou amplitudou. Spektrální vlastnosti řeči se zobrazují ve frekvenční oblasti a k jejich výpočtu se používá krátkodobá Fourierova transformace.

Základní řečovou jednotkou z hlediska fonetiky je **hláska (fon)**. Řeč je takto definována jako posloupnost hlásek. Množina všech odlišných hlásek se nazývá fonetický inventář. Fonetickou reprezentaci řeči bývá zvykem zapisovat do hranatých závorek. Fonetika se nezabývá významem řeči.

Klasické VAD algoritmy můžeme rozdělit na deterministické a stochastické.

2.2 Obecné principy detekce řečové aktivity

Deterministické algoritmy jsou založeny na výpočtu základních charakteristik řečového signálu, ze kterých je určen vhodný průběh kriteriální funkce pro následné prahování. Deterministické algoritmy můžeme dále dělit na:

a) Výkonové – kriteriální funkcí je výkon (energie) signálu řečové aktivity. Tyto metody selhávají při příliš vysoké úrovni rušivého pozadí, kdy slabší úseky signálu s řečovou aktivitou jsou maskovány silnějším aditivním šumem pozadí. Segmenty se rozdělí na základě energie na nízkoenergetické a vysokoenergetické. Obvykle toto rozdělení provedeme na základě fixně stanovené energetické úrovně, která segmenty rozdělí do těchto dvou kategorií. Práh je možné stanovit fixně přes celý signál nebo dynamicky na základě průměru nějakého množství segmentů.

b) Kepstrální – kritériem je rozdíl ve spektru aktuálního segmentu a rušivého pozadí. Většinou se používá euklidovská vzdálenost v kepstrálním prostoru, případně váhované kepstrální vzdálenosti, kdy váhovacím koeficientem zvýrazňujeme vyšší kepstrální koeficienty, což zvyšuje citlivost na jemné rozdíly ve spektrálních charakteristikách.

c) Koherenční – kritériem je průměrná koherence při přítomnosti řeči. Použití je pouze v prostředí nekoherentního šumu, především pak při snímání ve více kanálech.

Finální rozhodnutí o přítomnosti řeči je možné učinit po získání kriteriální funkce vhodným prahováním. Pricipiálně můžeme prahovací algoritmy rozdělit na prahování pevné a adaptivní. V případě pevného prahování je jednou z možností nastavení hodnoty prahu empiricky. Příkladem může být kriteriální funkce na bázi koherence, nabývající hodnot (0,1). Pro off-line zpracování je vhodné použít nastavení prahu na základě dynamiky průběhu kriteriální funkce celého signálu. Práh je nastaven na základě rozdílu maximální a minimální hodnoty kriteriální funkce. Avšak v případě určitého trendu v charakteristikách pozadí může být pevné nastavení prahu nevhodné. V případě použití adaptivního prahování se sledují změny v charakteristikách pozadí. Práh avšak nesmí sledovat nárůst kriteriální funkce v důsledku přítomnosti řeči [3], [5].

Stochastické detektory řečové aktivity mají blízko k rozpoznávání řeči. Často se jedná v principu o jednoduché rozpoznávací (klasifikační) metody. Tyto detektory mají jednu společnou vlastnost – vyžadují trénovací fázi pro nastavení parametrů používaných následně pro detekci. Stochastické detektory můžeme dělit na:

a) Detektor na bázi HMM (Hidden Markov Models) – skrytých Markovových modelů.
- Levopravé – pouze s postupnými přechody, zřetězeny na základě rozpoznávací gramatiky.
- Ergodické – přechody možné mezi všemi stavy.

b) Detektor na bázi GMM (Gaussian Mixture Model) – zjednodušený model HMM, kdy model obsahuje pouze jeden stav. Klasifikační kritérium je založeno na prav-

děpodobnosti pro daný vektor parametrů signálu. Nejčastěji se jedná o vektor kepstrálních koeficientů a pravděpodobnosti, že daný vektor odpovídá řeči či nikoliv. Klasifikační kritérium je pak založeno na generované pravděpodobnosti pro daný vektor parametrů segmentu signálu. Nejčastěji se jedná o vektor kepstrálních koeficientů [3].

2.3 Řečové příznaky pro VAD

Vzhledem k tomu, že při vstupu akustické vlny z artikulačního ústrojí do volného prostoru dochází k útlumu intenzity zvuku pro vyšší kmitočtové složky, provádí se před vlastním výpočtem příznaků kompenzace jejich útlumu. Pro některé hlásky je důležitá informace právě ve vyšších kmitočtových složkách spektra. Tato kompenzace se provádí jednoduchou pre-filtrací filtrem 1. řádu zesilujícím vyšší kmitočty. Tento proces se jmenuje **preemfáze**. Filtrace je dána následující rovnicí:

$$s\prime[n] = s[n] - m \cdot s[n-1] \tag{1}$$

Kde m je koeficient preemfáze a volí se z rozsahu 0,9 až 1 (typicky 0,97).

Řečový signál má kvazistacionární charakter, tedy z krátkodobého hlediska jej lze považovat za stacionární. Délka stacionárních signálů je asi 10 až 100 ms a závisí na rychlosti promluvy. Pro krátkodobou analýzu řečového signálu obvykle používáme 256 vzorků při $f_s = 8$ kHz, případně 512 vzorků při $f_s = 16$ kHz. Abychom zamezili prosakování ve spektru při použití diskrétní Fourierovy transformace (DFT), je nutné jednotlivé segmenty (kvazistacionární signál) váhovat vhodným oknem. Nejčastěji se používá Hammingovo okno, které toto prosakování dostatečně potlačuje [3].

2.3.1 Kepstrální koeficienty

Kepstrum označuje inverzní Fourierovu transformaci logaritmu komplexního spektra. Pokud nepožadujeme informaci o fázi, lze použít reálné kepstrum:

$$c_n = IDFT\{\ln|DFT\{x[k]\}|\} = \sum_{k=0}^{N-1} \ln|DFT\{x[k]\}|e^{\frac{jkn2\pi}{N}} \tag{2}$$

Koeficienty c_n nazýváme reálným kepstrem. Při použití malého počtu koeficientů c_n (v této práci pracuji s 13 kepstrálními koeficienty) získáme aproximaci spektra signálu, tzv. spektrální obálku (vyhlazené spektrum signálu) [3].

2.3.2 MFCC

MEL-kepstrální koeficienty obvykle aproximují nelineární vnímání frekvence lidským sluchem. Postup výpočtu MFCC je takový, že na vstup systému jsou přiváděny vzorky řečového signálu s(k), provede se preemfáze a na segmenty o délce 10 až 30 ms je aplikováno obvykle Hammingovo okno. Dále se pomocí FFT vypočte spektrum $|S(f)|$ (případně výkonové spektrum). Následuje provedení melovské filtrace. Ta se provede trojúhelníkovou bankou pásmových filtrů podél frekvenční osy s měřítkem v melovské škále. Počet pásem této banky filtrů je rozumné volit v závislosti na vzorkovací frekvenci a šířce přenášeného pásma. Většinou se volí 22 pásem pro f_s 8 kHz a 30 pásem pro f_s 16 kHz.

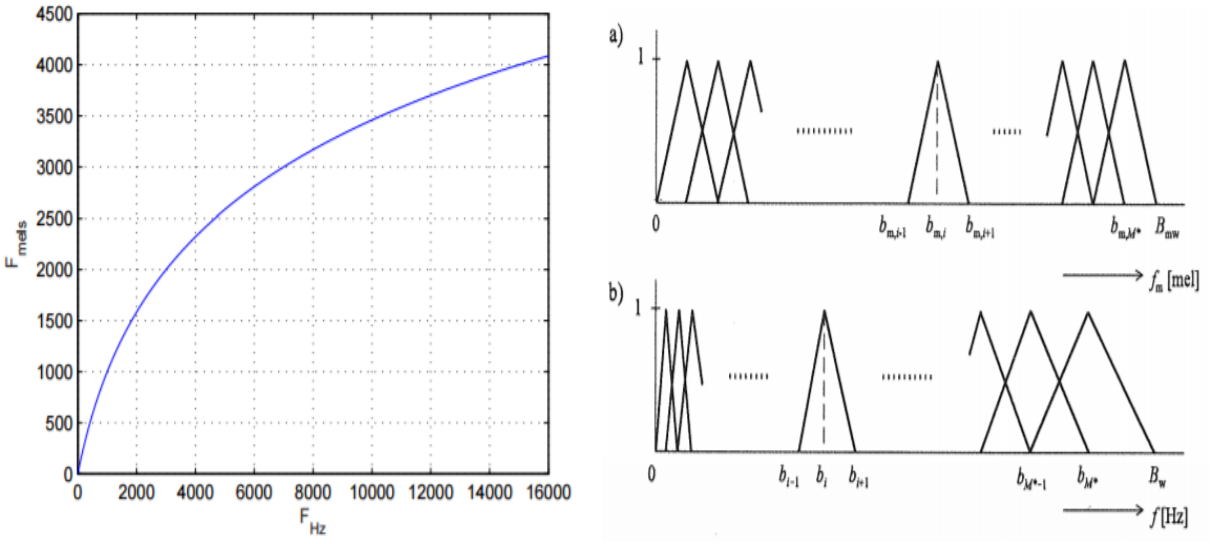
Přepočet mezi lineární frekvenční osou v Hz a melodickou frekvenční osou v melech:

$$f_{mel} = Mel(f) = 2595 \log(1 + \frac{f}{700}) \tag{3}$$

$$f = InvMel(f_{mel}) = 700(10^{\frac{f_{mel}}{2595}} - 1) \tag{4}$$

Kde f [Hz] je frekvence v lineární škále a f_{mel} [mel] je odpovídající frekvence v nelineární melovské škále.

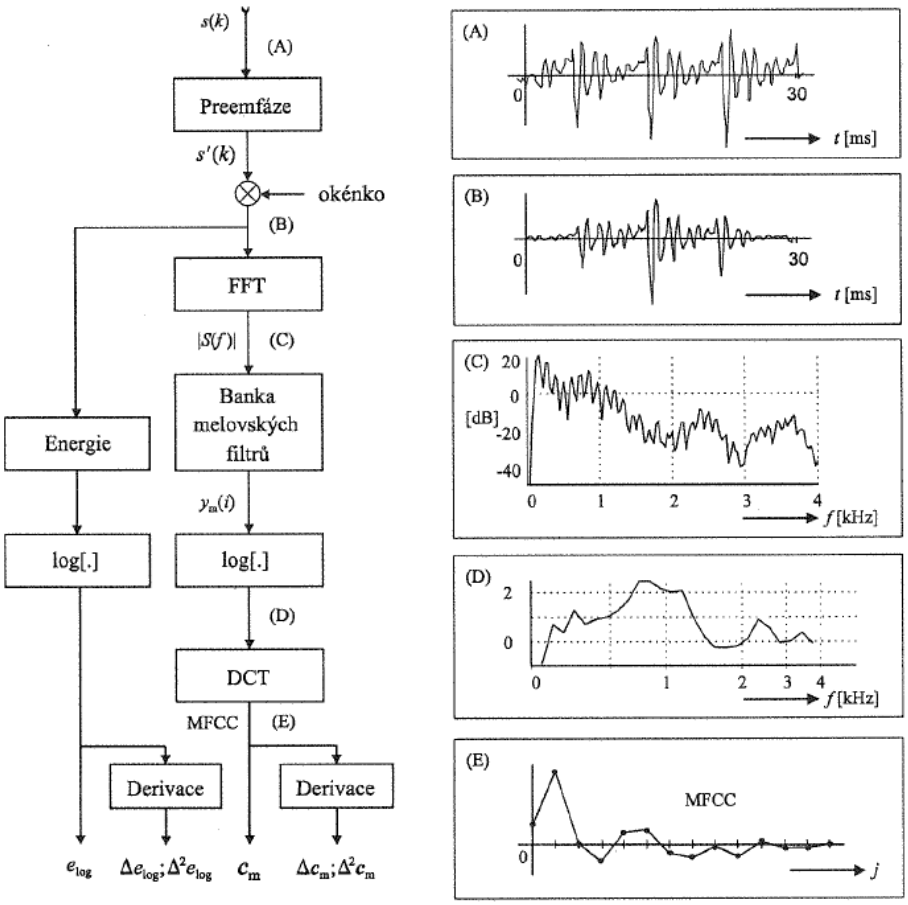
Trojúhelníkové filtry jsou obvykle rozloženy přes celé frekvenční pásmo od nuly až do Nyquistovy frekvence. Každý filtr má trojúhelníkovou pásmovou frekvenční odezvu a vzdálenosti filtrů a rovněž jejich šířka pásma jsou určeny konstantním intervalem v melovské frekvenční škále [1].



(a) (b)

Obrázek 2.2. Vztah mezi F_{mel} a F_{Hz} (a) a banka trojúhelníkových filtrů: v melovské škále a v původní škále (b)

Dalším krokem výpočtu je provedení logaritmů výstupů jednotlivých filtrů. Posledním krokem výpočtu melovských kepstrálních koeficientů je provedení zpětné diskrétní Fourierovy transformace (IDFT). Protože výkonové spektrum je reálné a symetrické, bude se IDFT redukovat na diskrétní kosinovou transformaci (DCT). Blokové schéma výpočtu je na následujícím obrázku:



Obrázek 2.3. Výpočet melovských kepstrálních koeficientů a průběhů v bodech A až E

Výpočet diskrétní kosinové transformace (DCT):

$$c_m(j) = \sum_{i=1}^{M} \log y_m(i) \cos(\frac{\pi j}{M}(i - 0.5)) \; pro \; j = 0, 1...M_c \tag{5}$$

Kde M je počet pásem melovského pásmového filtru a M_c je počet melovských kepstrálních koeficientů. Počet těchto koeficientů se obvykle volí nižší než je počet pásem melovského pásmového filtru. Obvykle se používá prvních 10 až 13 koeficientů. V této práci je použito 13 koeficientů.

Nulový koeficient $c_m(0)$ je úměrný logaritmu energie signálu. Často se počítá výpočtem logaritmu krátkodobé energie přímo z časových vzorků signálu:

$$e_{log} = \log \sum_{k=0}^{N-1} [s\prime(k)w(N - 1 - k)] \tag{6}$$

Kde N značí počet vzorků ve zpracovávaném segmentu řeči.

Právě tyto koeficienty vypočítané pro každý segment hlasového signálu používám

v této práci jako vstupní vektor parametrů neuronové sítě jak pro trénování sítě, tak pro následné testy klasifikace VAD/noVAD.

Tyto příznaky je dále možné zřetězit tak, že kromě příznaků aktuálně vyhodnocovaného segmentu použijeme N okolních segmentů. Získáme tak časový kontext. Význam v případě detekce řeči se dá ukázat na příkladu, že pokud je v předcházejícím i následném segmentu signálu řečová aktivita, je vyšší pravděpodobnost, že aktuálně zpracovávaný segment obsahuje také řeč, než kdyby okolní segmenty byly bez řečové aktivity. Takovéto kritérium má velký význam zejména pokud příznaky aktuálního segmentu jsou hraniční mezi řečovou aktivitou a řečovou pauzou.

Výše popsané příznaky jsou statické. K takto vytvořenému statickému vektoru příznaků c_m se často přidávají jako další příznaky ještě tzv. dynamické koeficienty. Dynamické koeficienty označované jako delta a delta-delta (akcelerační) vyjadřují dynamiku (derivaci) časové změny vektorů příznaků. Tyto příznaky nejsou v této práci použity, proto nejsou více popisovány.

Existují ještě další příznaky a jejich kombinace (např. PLP, LDA+STC +fMLLR), kterými se tato práce vzhledem k zaměření na dektekci řeči nezabývá. Stále se objevují novější a komplexnější příznaky, ale jejich použití je významné zejména při rozpoznávání řeči a v dalších složitějších úlohách zpracování řeči.

Kapitola 3
Detekce řeči na bázi umělé neuronové sítě

Umělé neuronové sítě byly navrženy na základě snahy o matematické modelování jevů v lidském mozku a základní funkční buňky nervového systému – neuronu. První práce na tomto poli vznikly již ve čtyřicátých letech 20. století (neurobiolog W. S. McCulloch, statistik W. Pitts). Až v roce 1958 byl publikován algoritmus pro učení vrstevnaté neuronové sítě s dopředným šířením signálu (Frank Rosenblatt – síť Perceptron). Bohužel větší uplatnění ANN neumožnil tehdejší stav výpočetní techniky a v roce 1969 došlo ke ztrátě zájmu o umělé neuronové sítě na poměrně dlouhou dobu, poté co matematici Minski a Papert odvodili, že pomocí jednovrstvé NN nelze řešit všechny logické operace (XOR). Toto omezení je však možno řešit vícevrstvými NN, avšak v té době nebyly známy algoritmy učení vícevrstvých sítí. Až v roce 1983 v rámci projektu Defence Advance Research Project Association (DARPA) se vědcům D. Rumelhartovi a LeCunovi podařilo odvodit algoritmus zpětného šíření chyby (Error Back-propagation of gradient). Od té doby prochází ANN mohutným rozvojem v oblasti nových algoritmů učení, architektur i aplikací. Rychlý vývoj je podpořen nejen teoretickým výzkumem v této oblasti, ale i rychle rostoucím výkonem výpočetní techniky, který umožňuje praktické aplikace ANN [4].

3.1 Generativní a diskriminativní algoritmy

Algoritmy učení neuronových sítí můžeme rozdělit na generativní a diskriminativní.

Generativní algoritmy
Neuronová síť se navrhuje přímo na rozlišování mezi třídami na základě příznaků a to tak, že se nejdříve naučí modelovat strukturu vstupních dat. Výstup jedné naučené vrstvy feature detektoru pak slouží jako vstupní data pro trénování další vrstvy. Po tomto generativním „pretrainingu“ jsou takto naučené sítě mnohem lepším výchozím bodem pro učení celé sítě něž při počátečním náhodném nastavení vah. Následně je síť „doladěna“ diskriminativním tréninkem algoritmem zpětného šíření chyby napříč celou DNN [6]. Mezi tyto generativní učící algoritmy patří zejména Restricted Boltzmann Machine, který je v této práci využit pro předtrénování hluboké neuronové sítě.

Diskriminativní algoritmy
Diskriminativní (nebo také kondicionální) modely představují třídu modelů strojového učení pro modelování závislosti nějaké třídy (unobserved) na jiné pozorované proměnné (observed variable). Zajímá nás pravděpodobnost

$$P(y|x)$$

To jest pravděpodobnost y na základě algoritmu prezentovaných dat x. Mezi tyto generativní učící algoritmy patří např.:

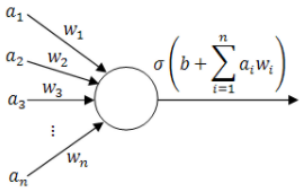
- Logistic regression

- Support vector machines
- Conditional random fields
- Random Forests
- Neural networks

Z těchto algoritmů se dále zaměříme pouze na neuronové stítě.

3.2 Perceptronové neuronové sítě

Perceptron je nejjednodušším modelem dopředné neuronové sítě, který se skládá jen z jednoho neuronu. Jeho užití je velmi omezené, neboť je možné ho použít pouze na množiny, které jsou lineárně separovatelné. Jeho rozšířením je vícevrstevný perceptron, který má již mnohem širší možnosti použití.



Obrázek 3.1. Perceptron

Inicializace spočívá v nastavení vah w_n na náhodné hodnoty. Při procesu učení jsou váhy a prahy měněny na základě vstupu do sítě (vstupního vektoru a_1 až a_n) tak, aby chyba na výstupu byla minimální.

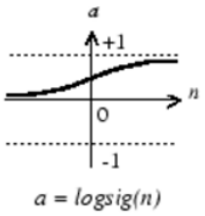
$$W_{t+1} = W_t + \Delta W_t \tag{1}$$

Kde W je matice vah a ΔW_t je odchylka aktuálních vah.
Jako potenciál neuronu je označován součet součinu všech vstupů a_n perceptronu vynásobených příslušnými vahami w_n. K tomuto součtu se přičítá ještě práh neuronu b (bias).

$$z = b + \sum_{n=0}^{n} w_n \cdot a_n \tag{2}$$

Na potenciál je aplikována přenosová funkce σ (transfer function). Nejčastěji se používá sigmoidální aktivační funkce:

$$\sigma(z) = \frac{1}{1+e^{-z}} \tag{3}$$



Obrázek 3.2. Log-Sigmoid transfer function

nebo hyperbolický tangens:

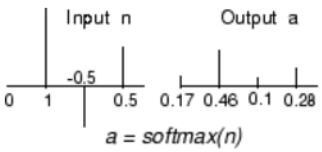
$$tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \tag{4}$$

Výstupní rozsah $\sigma(z)$ je (0,1) a tanh(z) je (-1,1).
Pro svůj jednoduchý gradient je také často používanou funkcí „Rectified linear unit function“ (ReLU):

$$ReLU(z) = max(0, z) \tag{5}$$

U výstupní vrstvy se často z důvodu správného rozdělení pravděpodobnosti používá normalizace aktivační funkce funkcí softmax::

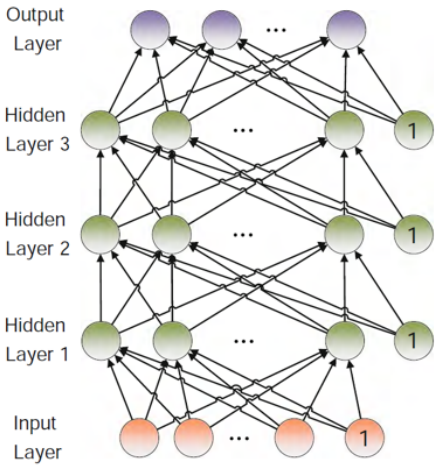
$$P(i|o) = softmax(z^L) \tag{6}$$



Obrázek 3.3. Softmax transfer function

3.3 Hluboké neuronové sítě

Termín hluboká neuronová síť původně označovala vícevrstvý perceptron s mnoha skrytými vrstvami. Nyní však tímto pojmem označujeme i sítě typu „deep belief network“, avšak terminologie není ještě úplně standardizována.



Obrázek 3.4. Hluboká neuronová síť

Vstupní vrstva neuronů se obvykle označuje jako vrstva 0 a výstupní jako vrstva L pro L + 1 vrstvou DNN. Data přivedená na vstupní vrstvu obvykle vyžadují nějakou formu preprocesignu (většinou normalizace vstupního vektoru). Vícevrstvá neuronová síť je většinou trénována algoritmem zpětného šíření chyby. Často bývá síť předtrénována algoritmy hlubokého učení. V této práci bylo použito předtrénování pomocí Restricted Boltzman Machine algoritmu (RBM). Takováto síť většinou dosahuje lepších výsledků než při počátečním náhodném nastavení vah [2], [6].

3.4 Předzpracování dat

Mezi základní používané techniky předzpracování dat (preprocessingu) patří normalizace příznaků per vzorek a globální standardizace příznaků. Pokud průměr příznaků každého vzorku se liší a hodnota tohoto průměru není podstatná pro popis řešeného problému, je vhodné tento průměr odečíst od každé hodnoty v tomto vzorku. Takto je snížena variabilita vzorků na vstupu do DNN. Ve zpracování řeči se průměr kepstrálních koeficientů (cepstral mean normalization – CMN) odečítá per promluva a pomáhá snížit zkreslení vlivem akustického kanálu.
Cílem globální příznakové standardizace je škálování dat v každé dimenzi tak, aby cílové vektory byly ve stejném rozsahu. Často se příznaky standardizují tak, aby měly nulový průměr a jednotkový rozptyl. V tom případě se jedná o Cepstral Mean and Variance Normalization (CMVN).

3.5 Algoritmus zpětného šíření chyby

Algoritmus zpětného šíření chyby (Error backpropagation - BP) je generalizací Widrow-Hoffova algoritmu nejmenších čtverců a byl odvozen pro vícevrstvé neuronové sítě se spojitou nelineární diferencovatelnou aktivační funkcí. Této podmínce dobře vyhovuje sigmoidální funkce nebo hyperbolická tangenta. Pomocí tohoto algoritmu se minimalizuje chybová funkce prostřednictvím adaptace synaptických vah. Chybová funkce je rovna nejčastěji střední kvadratické chybě mezi požadovaným a skutečným výstupem. Suma čtverců chyb bude vypočítána podle vztahu:

$$E = \sum_{k=1}^{K} [e(k)]^2 = \sum_{k=1}^{K} [t(k) - y(k)]^2 \tag{7}$$

Kde y je výstup z neuronu a t je požadovaná hodnota.
Váhy w_i jsou při učení modifikovány dle vztahu:

$$w_i(t+1) = w_i(t) + \alpha \sum_{k=1}^{K} \delta(k) x_i(k) \tag{8}$$

Chyba $\delta(k)$ je rovna chybě na výstupu neuronu násobené derivací aktivační funkce. Konstanta α udává rychlost učení (learning rate).

3.6 Momentum

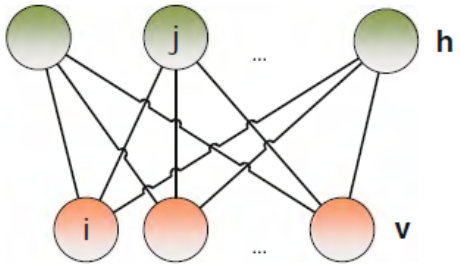
Je známo, že rychlost konvergence se dá zlepšit, je-li je aktualizace vah nejen na základě současných, ale i předchozích gradientů. Tento princip využívá Nesterovův algoritmus zrychleného gradientu pro učení DNN. Parametr momentum se volí v rozsahu 0.9 až 0.99. Momentum vyhlazuje změny parametrů DNN při procesu učení a je v této práci použit v realizaci detektoru v neuro2d.

3.7 Architektura neuronové sítě

Každá neuronová vrstva může být považována za extraktor příznaků předcházející vrstvy. Proto každá vrstva by měla mít dostatek neuronů, aby odlišila důležité příznaky. Zejména je to důležité v nižších vrstvách. Pokud má vrstva příliš mnoho neuronů, hrozí nebezpečí tzv. overfittingu. Taková neuronová síť příliš reaguje na šum ve vstupních datech a s rostoucím počtem neuronů ve vrstvě se výsledky zhoršují. Obecně široké a mělké sítě mívají sklon k nebezpečí overfittingu a hluboké a úzké sítě mají nebezpečí underfittingu. V případě underfittingu síť není schopna rozlišit podstatné příznaky. V literatuře [2] se pro úlohy hlasového rozpoznávání doporučuje 5 – 7 vrstev DNN s 1000 až 3000 neurony v každé vrstvě. Doporučují se spíše široké a hluboké sítě než úzké a mělké, jelikož v hlasových rozpoznávačích je mnoho lokálních optim. Součástí této práce je najít a experimentálně ověřit optimální architekturu voice activity detectoru.

3.7.1 Restricted Boltzmann Machine (RBM)

Restricted Boltzmann Machine (RBM) je stochastická generativní neuronová síť. Jedná se v podstatě o neorientovaný graf skládající se z vrstvy stochastických viditelných neuronů a z vrstvy stochastických skrytých neuronů. Viditelné a skryté neurony vytvářejí biparitní graf, který nemá žádnou vazbu mezi neurony ve viditelné vrstvě a žádnou vazbu mezi neurony ve skryté vrstvě. Díky chybějícím vazbám mezi neurony ve stejné vrstvě je možné snadno vypočítat posteriorní pravděpodobnosti $P(v|h)$ a $P(h|v)$, kde h je vektor skryté vrstvy a v je vektor viditelné vrstvy.



Obrázek 3.5. Restricted Boltzmann Machine

3.7.2 Deep Belief Network Pretraining

Jedná se o hierarchický generativní model sítě pro extrakci příznaků. V případě zpracování řeči vstupují do takové sítě obvykle zřetězené řečové příznaky, z nichž síť tvoří příznaky přes jednotlivé nelineární skryté vrstvy. Po natrénování RBM sítě se stav

skryté vrstvy použije pro trénink další RBM sítě, která se naučí modelovat závislosti mezi neurony skryté vrstvy první RBM sítě. Tento postup se může opakovat až se vytvoří mnohovrstevný nelineární příznakový detektor. Takto navršené RBM sítě jsou poté schopny fungovat jako jedna vícevrstvá generativní Deep Belief Network (DBN). Ačkoli RBM sítě jsou neorientované (undirected model), takto stohované RBM sítě mají 2 horní vrstvy neorientované, ale ostatní vrstvy se potom chovají jako orientované [6].

3.8 Detektor řeči na bázi ANN

ANN – (Artificial Neural Networks) jsou v detektorech využívány zejména ke klasifikaci analyzovaných příznaků signálu. Nejčastěji se opět používají kepstrální koeficienty. Klasifikace se nemusí týkat pouze řečové aktivity (voice activity), ale použití ANN je možné i pro rozpoznávání řeči (voice recognition), rozpoznávání mluvčího (speaker recognition), emoční analýzy a další. Výhodou použití ANN je zejména její odolnost k šumu v klasifikovaných úlohách. Při těchto úlohách se setkáváme nejčastěji s trojvrstvými sítěmi, kde počet neuronů ve vstupní vrstvě odpovídá počtu parametrů (např. počtu analyzovaných kepstrálních koeficientů). Počet neuronů ve druhé, skryté vrstvě, se volí s ohledem na řešenou úlohu (často stovky až tisíce neuronů). Počet výstupních neuronů odpovídá počtu klasifikovaných tříd. Tedy v případě VAD detektoru 2 neuronů, jejichž aktivace odpovídá stavům detekce řečové aktivity či pauzy v řeči. Součet pravděpodobností obou aktivací je tedy 1. V případě detektoru řeči je možné použít také pouze jeden výstupní neuron, jehož aktivace představuje řečovou aktivitu. Pravděpodobnost řečové pauzy je potom doplněk do 1 a je reprezentován neaktivním výstupním neuronem [3].

Výhodou použití ANN nejen pro detektor řeči je zejména [4]:

a) Získávání znalostí učením pomocí množiny vzorů bez nutnosti složitého matematického popisu.
b) Schopnost generalizace a diskriminace - do správných tříd jsou klasifikována vstupní data, na která nebyla síť přímo natrénována.

Ve srovnání s klasickými metodami nemá DNN problém s modelováním více současných příznaků v jednom rámci nebo okně, jelikož může použít jinou skupinu skrytých neuronů pro modelování různých událostí. Zatímco např. GMM předpokládá, že každý „datapoint“ je generován jednou komponentou, takže není možné modelovat více příznaků současně. DNN nemají problém zpracovávat kombinaci několika rámců vstupních koeficientů a využít takovéto okolní příznaky pro zpřesnění popisu chování vstupních charakteristik řeči v čase, což např. při použití GMM není takovým přínosem, jelikož GMM vyžaduje dekorelovaná vstupní data [6].

Kapitola 4
Implementace

4.1 Realizace v Matlabu s Neural Network Toolboxem

Matlab (matrix laboratory) – je programové prostředí společnosti MathWorks. Má svůj matematicky orientovaný jazyk, jehož syntaxe vychází z jazyka Fortran. Jedná se o placený nástroj pro výpočty, počítačovou simulaci, analýzu a prezentaci dat. Tento nástroj je rozšířen zejména mezi vědeckotechnickými a vývojovými pracovníky v technických oborech. I vzhledem k poměrně vysoké ceně tohoto komerčního software vznikla i bezplatná verze pod licencí GNU General Public License (GPL) s názvem GNU Octave, která je do značné míry s Matlabem kompatibilní.

Neural Network Toolbox je modul do Matlabu a je také vyvíjen společností MathWorks. Poskytuje algoritmy a aplikace pro vytváření, trénování, vizualizaci a simulaci neuronových sítí.

Detektor řeči jsem realizoval nejdříve v Matlabu s Neural Network Toolboxem, kde jsem experimentoval s různou topologií sítě (různý počet vrstev a neuronů v jednotlivých vrstvách dopředné neuronové sítě), různými přenosovými funkcemi neuronů a počtem učících epoch.

4.1.1 Trénování a příprava dat

Pro trénování ANN (fázi učení – nastavování vah) musíme kromě vektoru vstupních dat poskytnout síti i očekávané správné výstupní hodnoty. Pro tento účel jsem využil Acoustic - Phonetic Continuous Speech Corpus TIMIT, který vznikl v rámci projektu DARPA. ČVUT má zakoupenu licenci. Výhodou použití tohoto řečového korpusu je, že jeho součástí je kompletní fonetická transkripce, což velmi usnadňuje přípravu dat vhodných pro trénování ANN. Přesto bylo nutné poskytnutá data transformovat do vhodné formy.

Pro tento účel jsem vytvořil perl skript *Phon2SilOrSp.pl*. Výstupní očekávané hodnoty pro trénování jsou vytvořeny na základě fonetické transkripce. Pokud jako výstupní cestu uvedeme cestu k původnímu zdroji (adresář TIMITu), budou se soubory vytvářet nikoliv v rámci nové struktury, ale přímo v adresářové struktuře TIMITu. V konfiguračním souboru tohoto skriptu je nutné definovat, kterou transkripci považujeme za aktivní řeč a kterou za řeč nepovažujeme. Obecně se nemusí jednat pouze o ticho, ale o šum a jiné zvuky, které nemají povahu řeči. Nejdůležitějším souborem pro trénování ANN pro funkci voice activity detektoru je soubor s výchozí příponou `refout`. Tento soubor obsahuje právě výše zmíněné mapování fonetické transkripce na VAD/noVAD. Každý řádek tohoto souboru odpovídá jednomu segmentu odpovídajícího `wav` souboru. 1 v prvním sloupci znamená hlasovou aktivitu - „voice activity detected" (VAD), 0 znamená pauzu v řeči - „no voice activity detected" (noVAD). Druhý sloupec odpovídá prosté negaci sloupce prvního. Význam druhého sloupce je v tom, že Matlab vyžaduje, aby dimenze výstupních dat pro učení odpovídala počtu

neuronů ve výstupní vrstvě ANN.

Další soubory, které tento skript vytváří, jsou soubory typu:
- idet: jako refout, ale nemá sloupec s negací (obsahuje pouze jeden sloupec 0/1: VAD/noVAD)
- ndet: jako idet, ale po jednotlivých vzorcích (ne po segmentech)
- vad: na každém řádku je interval vzorků VAD nebo noVAD
- vas: vad po segmentech

4.1.2 Načtení dat do Matlabu

K načtení trénovacích dat a přípravě dat v Matlabu jsem vytvořil program *myPrepareData.m*, který v Matlabu zajistí načtení všech souborů `refout` a odpovídajících `wav` souborů ve smyčce. Z `wav` souborů jsou postupně pomocí modulu *vmfcc* (autor doc. Ing. Petr Pollák, ČVUT) vypočítány mel - kepstrální koeficienty. Vektory kepstrálních koeficientů jednotlivých segmentů každého `wav` souboru jsou jako řádky ukládány do matice „ceps“. O tuto je při další iteraci rozšířena matice vstupních dat „mSig“ o třináct sloupcích. Matice „m“ o stejném počtu řádků jako matice „mSig“ a 2 sloupcích (2 výstupní neurony) vznikne iteracemi, kdy tato matice je při každém průchodu rozšířena o řádky z příslušného `refout` souboru odpovídajícího `wav` souboru (resp. o kepstální koeficienty).

4.1.3 Vytvoření sítě v Matlabu

Matice „mSig“ a „m“ jsou použity jako parametry příkazu *train(net,mSig',m')*. Příkazem *net = feedforwardnet([x],'traingdx')* v Matlabu jsem vydefinoval neuronovou síť NN toolboxu. [x] definuje počet neuronů ve skryté vrstvě. Případně, pokud zadáme více vrstev, tak v jednotlivých vrstvách [x,x,x]. Dalším parametrem příkazu *feedforwardnet* je způsob učení sítě. Já jsem v této práci použil *traingdx* - Gradient descent backpropagation. Tento typ dopředné NN je možné vydefinovat jako *fitnet* – fitting nebo *patternnet* - pattern recognition sítě, případně *cascadeforwardnet* – síť má spoje přímo ze vstupní vrstvy do hlubších vrstev. S těmito sítěmi jsem však neexperimentoval. Příkazem *net.layers1.transferFcn = 'logsig'* definujeme přenosovou funkci neuronu. Ve skrytých vrstvách jsem použil *logsig* a ve výstupní vrstvě *softmax*.

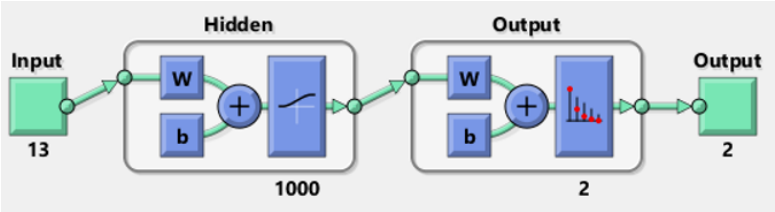
Příkazem *net.trainParam.epochs = 1000* se definuje maximální počet trénovacích epoch. Pokud trénování projde maximálním množstvím učících epoch nebo je dosaženo při trénování minimálního gradientu nastaveného příkazem *net.trainParam.min_grad* nebo maximálního počtu selhání validace trénování nastaveného příkazem *net.trainParam.max_fail*, je trénování ukončeno.

V rámci této práce jsem experimentoval s různým množstvím neuronů ve skryté vrstvě, počtu skrytých vrstev a počtu trénovacích epoch.

Příklad konfigurace třívrstvé neuronové sítě v Matlabu (vstupní vrstva 13 neuronů, skrytá vrstva 1000 neuronů, výstupní vrstva 2 neurony):

```
neu = 1000;
net = feedforwardnet(neu,'traingdx');
net.layers{1}.transferFcn = 'logsig';
net.layers{2}.transferFcn = 'softmax';
```

```
net.inputs{1}.processFcns = {};
net.outputs{2}.processFcns = {};
```



Obrázek 4.1. Příklad topologie NN v Matlabu

4.1.4 Vyhodnocení klasifikace

Natrénované sítě byly otestovány novými, při trénování nepoužitými, daty. Na vstup byly přivedeny stejným způsobem připravené kepstrální koeficienty. Pro získání vektoru klasifikovaných dat VAD/noVAD byla použita funkce *outputs = sim(net,mSig')*, kde *net* je natrénovaná síť a vektor *mSig* jsou vstupní kepstrální koeficienty testovacích dat. Pro zjištění kolik segmentů bylo chybně klasifikováno, jsem provedl porovnání sítí vyhodnocených dat VAD ve vektoru outputs s připraveným vektorem očekávaných dat, jak by je síť měla vyhodnotit při 100% úspěšnosti. Pro porovnání sítí vypočtených pravděpodobností, musíme tyto nejdříve převést do binární podoby, abychom mohli snadno binárně porovnat vektory dat vypočtených sítí s námi očekávanými hodnotami VAD/noVAD.

$$b = (outputs > 0.5); \tag{1}$$

$$e = xor(b', m); \tag{2}$$

Vektor e obsahuje 1 na místě, kde se vektory neshodují, tedy chybně klasifikovaný segment. Nula je ve vektoru e na místech, kde se oba vektory shodují. Jedničky (chyby) spočítáme:

$$sum(e == 1) \tag{3}$$

Výsledek vydělíme celkovým počtem hodnot a po vynásobení 100 získáme chybu v procentech.

4.1.5 Vyhodnocení realizace v Matlabu

Počet epoch nutných pro natrénování sítě z dat zvukového korpusu TIMIT je asi 200. Další trénování pouze způsobuje oscilace ve velikosti gradientu a odchylky výstupu od cílových hodnot a již nepřináší zlepšení schopností cíle správně klasifikovat vstupní data. Pro cílové, produkční, řešení není řešení v Matlabu vhodné, jelikož se jedná o poměrně velký softwarový balík a navíc má různá omezení (veškerá vstupní data musí být součástí jedné matice, značné nároky na paměť…). Pro experimentování je však tento SW velmi vhodný, jelikož umožňuje poměrně snadnou konfiguraci sítě a poskytuje dobré možnosti vyhodnocování učení a ověření správné klasifikace sítě na základě síti předložených řečových příznaků.

4.2 Realizace v neural2d

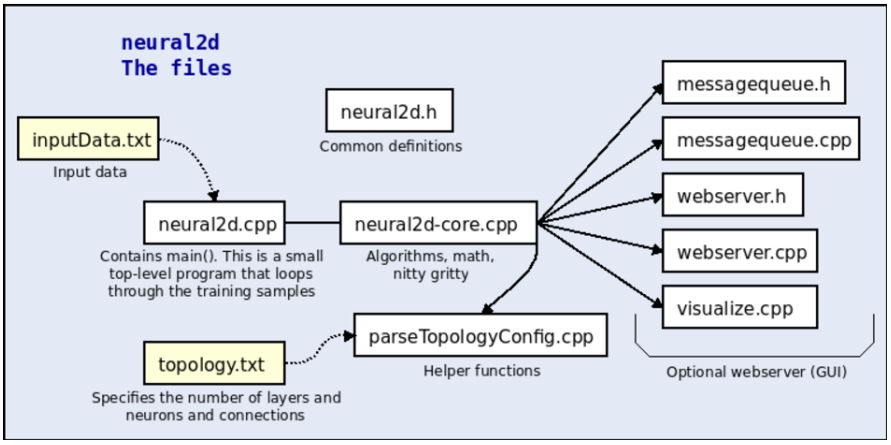
4.2.1 O neural2d

Neural2d je konzolový program s možností HTTP webového rozhraní. Program byl napsán v jazyce C++ pod MIT licencí. Jedná se o neuronovou síť s algoritmem zpětného šíření chyby. Vstupní data mohou být jednodimenzionální nebo dvoudimenzionální (např. obraz). Topologie sítě se specifikuje v souboru `topology.txt`. Vstupní data do neuronové sítě jsou očekávána v textovém souboru `inputData.txt`. Soubor `inputData.txt` může obsahovat přímo vstupní data, může obsahovat seznam `.bmp` souborů nebo `.dat` souborů, které obsahují vstupní data v binární formě.

Pokud není použito grafické uživatelské rozhraní, nemá program žádné další závislosti kromě C++ 11 kompilátoru. V případě kompilace s podporou GUI je potřeba standardní POSIX socket networking knihovna. Program je možné přeložit v Linuxu i ve Windows.

Kompilace programu v Linuxu:

```
cd neural2d
mkdir build
cd build
cmake ..
make
```



Obrázek 4.2. Soubory neural2d

4.2.2 Vytvoření sítě

Pro vytvoření neuronové sítě v *neural2d* je nejdříve nutné specifikovat její topologii v konfiguračním souboru `topology.txt`.

Gramatika tohoto souboru:

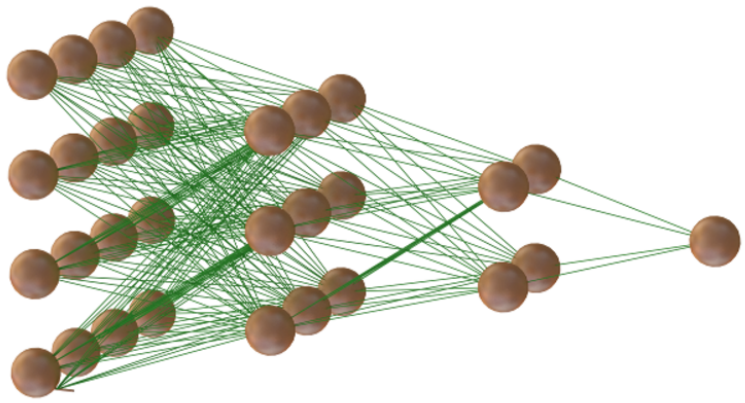
```
layer-name parameters := parameter [ parameters ]
parameters := parameter [ parameters ]
parameter :=
input | output | layername
size dxy-spec
from layer-name
channel channel-spec
radius xy-spec
tf transfer-function-spec
convolve filter-spec
convolve xy-spec
pool  max | avg  xy-spec
dxy-spec := [ integer * ] integer [ x integer ]
xy-spec := integer [ x integer ]
channel-spec := R | G | B | BW
transfer-function-spec := tanh | logistic | linear | ramp | gaussian |
relu
```

Konfigurační soubor musí splňovat následující pravidla:

- Komentáře jsou uvozeny znakem „#”
- Vstupní vrstva se musí jmenovat „input”
- Výstupní vrstva se musí jmenovat „output“
- Jméno každé skryté vrstvy musí začínat „layer“
- Argumentem „from“ musí být definována vrstva
- Parametry barevných kanálů mohou být pouze ve vstupní vrstvě
- Pokud je vynechán parametr „size“, je velikost zkopírována z vrstvy specifikované v parametru „from“
- Parametr „radius“ nesmí být na stejném řádku jako parametr „convolve“ nebo „pool“
- Stejné jméno vrstvy může být opakovaně definováno s různým „from“ parametrem
- V „xy-spec“ a v X,Y části „dxy-spec“ může být specifikována jedna nebo dvě dimenze. Pokud je specifikována jedna dimenze, tak druhá je považována za 1 (př. „8 x 1“ je řada 8 neuronů).

Příklad konfiguračního souboru a neuronová síť s touto architekturou:

```
input size 4x4
layer1 size 3x3 from input
layer2 size 2x2 from layer1
output size 1 from layer2
```



Obrázek 4.3. Příklad topologie NN dle uvedené konfigurace

Příklad konfiguračního souboru, který jsem použil pro realizaci detektoru řeči v neural2d:

```
input size 13
layer1 size 500 from input
layer2 size 100 from layer1
layer3 size 30 from layer2
output size 2 from layer3 tf logistic
```

Síť dle této konfigurace vytvoří 5 vrstvou neuronovou síť s 13 vstupními neurony, 500 neurony v první skryté vrstvě, 100 neurony ve druhé skryté vrstvě, 30 neurony v poslední skryté vrstvě a 2 výstupními neurony reprezentujícími pauzu v řeči (no voice activity) a hlasovou aktivitu (voice activity). Výstupní vrstva má nastavenu logistickou (sigmoidální) přenosovou funkci. Ostatní vrstvy mají tanh přenosovou funkci. V případě zřetězení vstupních příznaků je nutné rozšířit počet vstupních vektorů na násobek 13 dle použitého násobku příznaků (splicing).

4.2.3 Příprava dat a extrakce řečových parametrů

Pro transformaci fonetického popisu TIMITu do formy vhodné pro trénování neuronové sítě jsem vytvořil skript *neural2d_feat.sh*. Tento skript využívá k přípravě dat pro trénování některé soubory z Kaldi toolkitu. Zejména pro výpočet MFCC koeficientů. K funkci tohoto detektoru je však možné použít jiné možnosti výpočtu příznaků a řešení není tedy na těchto souborech závislé. Ve skriptu je třeba nastavit potřebné cesty k TIMITu (řečové databázi) a Kaldi binárním souborům. Tento skript spouští program *compute-mfcc-feats*, který realizuje výpočet příznaků pro připravená data. Jedná se právě o binární spustitelný soubor z Kaldi toolkitu. Výstupní soubor tohoto programu obsahuje všechny promluvy daného souboru převedené na MFCC. Každá promluva je uvedena unikátním identifikátorem následovaným levou hranatou závorkou. Poté následují jednotlivé řádky po 13 koeficientech. Každý řádek odpovídá jednomu segmentu řeči. Konec promluvy označuje pravá hranatá závorka.

Výstupní očekávané hodnoty pro trénování jsou vytvořeny na základě fonetické transkripce. Pro tento účel jsem vytvořil perl skript *Phon2SilOrSpNN2d.pl*. V konfiguračním souboru *config.txt* tohoto skriptu je nutné definovat, kterou transkripci považujeme za aktivní řeč a kterou za řeč nepovažujeme. Obecně se nemusí jednat pouze o ticho, ale o šum a jiné zvuky, které nemají povahu řeči. V tomto řešení jsou jako non-voice mapovány labely ve skriptu takto:

```
@silence = (pau, epi, h#)
```

Abychom mohli převést fonetický popis na binární vektor (VAD/noVAD) odpovídající řečovým segmentům, potřebujeme ve skriptu nastavit délku okna a přesah oken, které provedeme nastavením proměnných `$wlen` a `$wstep` v konfiguračním souboru *config.txt*. Tyto parametry musí přesně odpovídat použitým parametrům při získání řečových příznaků, abychom dostali takovou délku binárního vektoru, která odpovídá počtu řádků příznaků pro jednotlivé promluvy. Výstupem tohoto skriptu je soubor *idets.scp*, který obsahuje list cest na jednotlivé *idet* soubory uvozené jedinečným identifikátorem promluvy. Soubory, na které tento list odkazuje, obsahují právě výše zmíněné mapování fonetické transkripce na `1` - řečová aktivita nebo `0` - žádná řečová aktivita (VAD/noVAD). Každý řádek tohoto souboru odpovídá jednomu segmentu odpovídajícího `wav` souboru.

Struktura výstupního souboru je následující:

```
faem0_si1392 [
38.42281 -31.03193 -10.48713 -9.130266 -11.1834 -1.999403 18.74048
3.368952 6.262515 -5.230333 -4.666165 -14.61002 -12.81378
…]
```

Pro trénování sítě je potřeba připravit vstupní soubor, standardně pojmenovaný `inputNNData.txt`. Tento soubor v případě trénování obsahuje jak vstupní příznaky, které se uvádějí ve složených závorkách, tak i cílové vektory. Každý řádek odpovídá jednomu segmentu signálu. Formát vstupního souboru je tedy následující:

{35.31086 – 29.43537 – 2.165891 – 5.569582 1.829457 2.867076 6.062552 – 1.787973 – 3.104678 – 1.536021 5.721472 1.975193 3.525253 } `0 1`

Tento vstupní soubor pro trénování neuronové sítě vytváří perlový skript *FeaturesPrepNN2d.pl*, který jsem vytvořil. V konfiguračním souboru tohoto skriptu se nastaví proměnná `InputFile`, což je soubor *idets.scp* popsaný výše a *InputFileMFCCtext*, což je soubor s mfcc příznaky. Výstupem je vstupní soubor pro trénování neuronové sítě *inputNNData.txt*.

4.2.4 Trénování DNN a detekce řeči v neural2d

Parametry pro učení neuronové sítě je nutné nastavit přímo ve zdrojovém souboru `neural2d.cpp`. Jedná se zejména o tyto parametry:

```
myNet.eta = 0.3f;
```

```
myNet.dynamicEtaAdjust = false;
myNet.alpha = 0.3f;
myNet.reportEveryNth = 2000;
myNet.repeatInputSamples = true;
myNet.shuffleInputSamples = false;
myNet.doneErrorThreshold = 0.000205f;
```

Po kompilaci je možné spustit učení příkazem:

```
/neural2d topologyNN.txt inputNNData.txt weights.txt
```

Po úspěšném natrénování sítě jsou naučené váhy uloženy v souboru `weights.txt`.

Uložené váhy mohou být nahrány funkcí `loadWeights(filename)` a síť může dekódovat nová data. Vlastní detekce řeči se spouští zavoláním funkce `feedForward()`. Pro vlastní detekci řeči musíme síti poskytnout nový vstupní soubor `inputNNData.txt`, který obsahuje pouze řečové příznaky, nikoliv cílové vektory.

4.2.5 Vyhodnocení realizace v neural2d

Pro vyhodnocení úspěšnosti detekce byl použit program *vadcrit*. Výstupní soubor z neuronové sítě bylo proto nutné převést do požadovaného formátu. Pro porovnání s referenčním detektorem bylo nutné převést do tohoto formátu také cílové vektory *targets.ark*. Pro převod byl použit perlovský skript *AliNNoutTest.pl*, který jsem za tímto účelem vytvořil.

Pro vyhodnocení úspěšnosti detekce nebyly použity promluvy, na kterých byla síť trénována, ale skupina promluv testovacího souboru.

Vyhodnocení úspěšnosti se potom spustí příkazem `vadcrit file1 file2`, kde `file1` je testovaný výstupní vzorek a `file2` je referenční vzorek vzniklý převodem z cílových vektorů.

Detektor navržený v neural2D pro svoji kompaktnost a jednoduchý kód v C++ může být snadno implementovatelný v různých embedded zařízeních, jako jsou různé detektory v komunikační technice, Internetu věcí (IoT) apod. Tento detektor se při experimentech prokázal jako funkční. Framework neural2D nepodporuje RBM předtrénování. O přesnosti detekce je pojednáno v kapitole zabývající se experimenty.

4.3 Realizace v Kaldi

4.3.1 O Kaldi

Kaldi je sada nástrojů pro rozpoznávání řeči. Jsou napsány v jazyce C++ a licencovány pod licencí Apache v2.0. Kaldi je primárně určeno pro výzkumníky v oblasti zpracování řeči. Součástí Kaldi jsou i vzorové skripty, tzv. „recipes“ pro některé řešení rozpoznávačů. Pro účely této práce se ale ze žádného přímo vycházet nedá. Autory projektu jsou Povey, Daniel and Ghoshal, Arnab and Boulianne, Gilles and Burget, Lukas and Glembek, Ondrej and Goel, Nagendra and Hannemann, Mirko and Motlicek, Petr and Qian, Yanmin and Schwarz, Petr and Silovsky, Jan and Stemmer, Georg and Karel Vesely. Projekt obsahuje 3 různé implementace neuronových sítí (nnet1, nnet2 a nnet3).

Já jsem si vybral řešení nnet1, jelikož se mi zdálo lépe zdokumentované a nalezl jsem k této síti více zdrojů.

4.3.2 Instalace Kaldi

Je potřeba nejdříve nainstalovat, pokud v sytému již nejsou, následující programy:

- *atlas* – nutné pro výpočty lineární algebry
- *autoconf* – automatická kompilace software
- *automake* – vytváří portabilní Make soubory
- *git* – umožní pracovat s Git distribuovaným verzovacím úložištěm
- *libtool* – vytváření statických a dynamických knihoven
- *svn* – revision control system (Subversion), nezbytné pro stažení a instalaci
- *wget* – data transfer pomocí HTTP, HTTPS a FTP protokolů
- *zlib* – komprese dat
- *awk* – jazyk pro práci s textovými soubory a datovými proudy
- *bash* – Unixový shell
- *grep* – command-line utilita pro prohledávání textu podporující regulární výrazy
- *make* – sestavuje spustitelné soubory a knihovny
- *perl* – programovací jazyk vhodný zejména pro zpracování textu

Stažení Kaldi z Gitu:

git clone https://github.com/kaldi-asr/kaldi.git kaldi –origin upstream

Případné aktualizace a opravy chyb se instalují:

git pull

4.3.3 Adresářová struktura receptu

steps – obsahuje skripty, které jsou součástí Kaldi
utils – obsahuje skripty, které slouží k modifikaci souborů
local – obsahuje pouze soubory specifické pro daný korpus, na kterém pracujeme
data – standardně se tu ukládají adresáře jako „train“ a „test“
exp – obsahuje aktuální experimenty, modely a logy
conf – obsahuje konfigurační soubory pro různé skripty

Soubor *path.sh* obsahuje cestu k adresáři Kaldi a dalším potřebným souborům. Binární soubory jsou obvykle uloženy v jiných adresářích, než kde provádíme experimenty. Abychom k těmto souborům mohli z našich skriptů (recipe) přistupovat, je nutné nejprve spustit soubor *path.sh*.

4.3.4 Příprava dat a extrakce řečových parametrů v Kaldi

Pro transformaci fonetického popisu TIMITu do formy vhodné pro trénování neuronové sítě jsem vytvořil skript *VADdataPrep.sh*. Ve skriptu je třeba nastavit následující parametry, které je pro běh na ČVUT klastru magi208 možné nastavit např. takto:

feats_nj = 10 - počet výpočetních jader pro výpočet
timit=/workspace/user/timit/timit – cesta k TIMUTu

kaldi=/usr/kaldi-trunk - cesta k instalaci Kaldi
workDir=/scratch/user/workdir - zde se uloží vytvořené soubory s extrahovanými příznaky jednotlivých `wav` souborů a target soubory s vektory 0/1 (speech/non-speech)
sourceDir=/workspace/user/VAD7/data/local/data – adresář, kde se uloží výstup subskriptu *timit_data_prep.sh*, který je součástí Kaldi toolkitu

Skript *timit_data_prep.sh*, který *VADdataPrep.sh* spouští, vytvoří různé soubory popisující nahrávky uložené v souborech `wav` z pohledu mluvčího promluvy, fonetického přepisu, pohlaví mluvčího, délky promluvy atd. Zároveň rozdělí soubory promluv na 3 části „train“, „dev“, „test“. Soubory vzorků „train“ a „dev“ slouží pro učení neuronové sítě. „Test“ soubory jsou určeny pro otestování správné funkce DNN based VAD detektoru na vzorcích, na které nebyla síť trénována. Pro realizaci VAD detektoru použijeme 3 soubory vytvořené skriptem *timit_data_prep.sh: train_wav.scp, dev_wav.scp, test_wav.scp*. Tyto soubory nám poskytnou jedinečný identifikátor promluvy, který je navázán na konkrétní `wav` soubor, ve kterém je promluva uložena, viz. příklad:

```
mzmb0_sx86 /usr/local/kaldi-trunk/tools/sph2pipe_v2.5/sph2pipe -f wav
/data/timit/timit/train/dr2/mzmb0/sx86.wav
```

Pro výpočet příznaků je k dispozici *compute-mfcc-feats*, který realizuje výpočet příznaků pro připravená data. Tento program z Kaldi toolkitu nám pro každý soubor „train“, „dev“ a „test“ vypočítá melovské kepstrální koeficienty a uloží do souboru odkazy na tyto příznaky dle unikátního identifikátoru promluvy s rozdělením do 3 souborů (train, dev, test):

$workDir/raw_mfcc/feats.scp
$workDir/rawDev_mfcc/feats.scp
$workDir/rawTest_mfcc/feats.scp
v následujícím formátu:

```
faem0_sx312 /scratch/lakosil/workdir/raw_mfcc/feats.ark:194451
```

Na začátku řádku je opět unikátní identifikátor promluvy, následuje cesta a název souboru archivu, ve kterém jsou uloženy MFCC náležející příslušnému souboru promluv, a poslední číslo udává polohu začátku MFCC příslušné promluvy (byte offset pro funkci `fseek` C++). Standardně jsou soubory `ark` ukládány komprimovaně. Pokud chceme uložit soubor bez komprimace, předáme v parametru programu za typ souboru `ark` parametr „t“ oddělený čárkou (př. `ark,t,scp:$workDir/raw_mfcc/feats.ark`). Struktura souboru *feats.ark* je následující:

```
faem0_si1392 [
38.42281 -31.03193 -10.48713 -9.130266 -11.1834 -1.999403 18.74048
3.368952 6.262515 -5.230333 -4.666165 -14.61002 -12.81378
…]
```

Soubor *feats.ark* obsahuje všechny promluvy daného souboru převedené na MFCC. Každá promluva je uvedena unikátním identifikátorem, za kterým následuje levá hranatá závorka. Poté následují jednotlivé řádky po 13 koeficientech. Každý řádek odpovídá jednomu segmentu řeči. Konec promluvy označuje pravá hranatá závorka.

Následně se spouští skript, který jsem vytvořil v perlu *Phon2SilOrSp.pl*. Jedná se o mírně modifikovaný skript popsaný v sekci realizace VAD v Matlabu. Modifikace byla zejména v přepsání některých částí závislých na OS pro Linux. Dále byly ze skriptu odstraněny konfigurační soubory a zrušeno kopírování `wav` souborů. Výstupní očekávané hodnoty pro trénování se musí vytvořit na základě fonetické transkripce. Ve skriptu se vydefinuje, která transkripce je považována za aktivní řeč a která za řečovou pauzu. V našem případě jsou jako řečové pauzy mapovány značky ve skriptu takto:

```
@silence = (pau, epi, h#)
```

Aby bylo možné převést fonetický popis na binární vektor (VAD/noVAD) odpovídající řečovým segmentům, je potřeba ve skriptu nastavit délku okna a přesah oken, které se provede nastavením proměnných `$wlen` a `$wstep`. Tyto parametry musí přesně odpovídat použitým parametrům při získání řečových příznaků, aby se docílilo takové délky binárního vektoru, která odpovídá počtu řádků příznaků pro jednotlivé promluvy. Tato shoda je v dalším skriptu kontrolována. Pokud by nějaký vektor počtem prvků neodpovídal počtu extrahovaných MFCC příznaků pro tuto promluvu, tak by se tato promluva nepoužila pro trénování detektoru. Nejdůležitějším výstupem tohoto skriptu jsou právě tyto binární vektory pro každou promluvu. Tyto soubory mají výchozí příponu *refout*. Každý řádek tohoto souboru odpovídá jednomu segmentu odpovídajícího `wav` souboru. "1" v prvním sloupci znamená hlasovou aktivitu, "0" znamená řečovou pauzu (noVAD). Další soubory, které tento skript vytváří, nejsou pro realizaci VAD potřeba a jejich význam byl popsán v kapitole věnované realizaci v Matlabu. Posledním úkolem v rámci přípravy dat pro trénování neuronové sítě detektoru je sloučení jednotlivých `refout` souborů obsahující výše popsané cílové binární vektory do jednoho souboru (targets.ark) ve formátu očekávaném neuronovou sítí v Kaldi. K tomuto účelu jsem napsal perl skripty *AlignFeaForNet.pl, AlignFeaForNetDev.pl a AlignFeaForNetTest.pl*. Formát je uvozen jednoznačným identifikátorem promluvy následovaným "0" – no speech a "1" – speech.

```
faem0_si1392 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1…
```

Identifikátor promluvy slouží jako klíč pro propojení se souborem *feats.ark* obsahujícím koeficienty MFCC. Na základě tohoto propojení se ověří, jestli přesně odpovídá délka cílového vektoru VAD/noVAD na segment s počtem segmentů (řádků) v souboru příznaků MFCC. Pokud by některá promluva neodpovídala přesně v mapování segmentů příznaků na cílový vektor, tak se tato promluva vyřadí ze souboru promluv pro učení neuronové sítě. Skript *AlignFeaForNetDev.pl* se liší pouze v tom, že výstup se sloučí s výstupem skriptu *AlignFeaForNet.pl* takovým způsobem, že vzniknou 2 nezávislé soubory `scp` pro train a dev trénovací soubory (*feats.scp* v různých adresářích dle nastavení proměnných `$outputFileFeats`), ale pouze jeden společný soubor *targets.ark* s cílovými vektory za oba soubory promluv (train i dev). Target vektory nejsou sice potřeba pro testovací soubor promluv pro trénování detektoru, přesto je nutné mít skript i pro tento soubor *AlignFeaForNetTest.pl*. Tyto testovací target vektory budou dále potřeba pro ověření správné funkce detektoru. Výhoda separátních skriptů pro každý soubor promluv je, že pokud se s některým souborem nepracuje (např. pokud je potřeba pouze otestovat již natrénovanou síť), ostatní se zakomentuje a nemusí se zbytečně počítat.

4.3.5 Trénování DNN v Kaldi

Za účelem natrénování neuronové sítě jsem vytvořil dva skripty *VADrunDNN.sh a VADrunDNN2.sh*. Tyto skripty se liší od sebe tím, že *VADrunDNN2.sh* provádí před vlastním trénováním algoritmem zpětného šíření chyby předtrénování prostřednictvím Restricted Boltzmann Machine. *VADrunDNN.sh* předtrénování neprovádí. Jedná se tedy o síť typu MLP.

Před spuštěním samotného trénování je potřeba nastavit několik proměnných, které určují jakým způsobem se úlohy spouští na klastru ve frontě. K tomuto účelu se na začátku těchto skriptů spustí shellovský skript *cmd.sh*. Zde se specifikuje, jestli je pro danou úlohu využívána fronta cpu a je možné specifikovat další pametry, jako např. počet jader. Pokud máme k dispozici grafická jádra (gpu), nastavuje se i proměnná `cuda_cmd`. CUDA (Compute Unified Device Architecture) je hardwarová a softwarová architektura, která umožňuje na vybraných gpu spouštět programy napsané v jazycích C/C++, OpenCL, DirectCompute a některých dalších. Výpočet na grafických jádrech probíhá u úloh s vysokou možností paralelizace výrazně rychleji. Při experimentech v této práci byla úloha, jejíž výpočet trval na notebooku s cpu Core i7 celý den, vypočítána na grafické kartě GeForce GTX 650 Ti do 30 minut.

Příklad *cmd.sh* pro běh na klastru magi ČVUT:

```
export train_cmd=queue.pl -q cpu.q
export decode_cmd=queue.pl -q cpu.q
export mkgraph_cmd=queue.pl -q cpu.q
export cuda_cmd=queue.pl -q gpu.q
```

Pokud má skript běžet na lokálním počítači místo na klastru, nastaví se *cmd.sh* takto:

```
export train_cmd=run.pl
export decode_cmd=run.pl
export cuda_cmd=run.pl
export mkgraph_cmd=run.pl
```

Pokud není pro výpočty k dispozici grafická karta s podporou CUDA, je nutné projít subskripty a nastavit `'--skip-cuda-check true'`, jinak skripty končí chybou. Vlastní trénování neuronové sítě v Kaldi se spustí příkazem:

```
train_nnet.sh <data-train> <data-dev> <lang-dir> <ali-train> <ali-dev>
<exp-dir>
```

Důležité parametry příkazu *train_nnet.sh* specifikují:

`hid_layers` - počet skrytých vrstev
`hid_dim` - počet neuronů ve vrstvě
`copy_feats` - zkopíruje příznaky v přeházeném pořadí pro učení
`splice` – počet zřetězených příznaků
`num_tgt` – počet výstupních neuronů

`learn_rate` – výchozí koeficient učení (learning rate)
`labels` – manuální cílové vektory (jinak se očekávají pdf alignment soubory)

Parametrem `labels` přepneme trénování z režimu, kdy jsou jako vstup očekávány pdf alignment soubory, na režim manuálních cílových vektorů – targets. Parametru `labels` je potřeba předat výstup programu *ali-to-post ark*, který převede formát souboru `targets.ark` na vstupní formát potřebný pro trénování v Kaldi nnet1.

Dále je nutné předat jako parametr adresáře `<data-train> <data-dev> <lang-dir> <ali-train> <ali-dev> <exp-dir>`. Adresář `<data-train>` určuje, kde bude hledán soubor `feats.scp` s příznaky extrahovanými z `wav` souborů. Adresář `<data-dev>` určuje, kde bude hledán soubor `feats.scp` s příznaky souboru dev (což by mělo být 10% promluv ze souboru promluv pro trénování). Adresáře `<lang-dir> <ali-train> <ali-dev>` jsou v případě použití manuálního mapování labelů nepoužity. Jako tento parametr je zadán řetězec „dummy-dir“, ze kterého je zřejmé, že se nejedná o reálnou cestu. Parametr je sice ignorován, ale je programem vyžadován.

Samotné trénování neuronové sítě bez RBM předtrénování se spustí tedy tímto příkazem:

```
$cuda_cmd $dir/log/train_nnet.log  local/nnet/train_nnet.sh --num_tgt 2
--hid-layers 6 --hid-dim 500
--copy-feats false --labels
ark:ali-to-post ark:/scratch/user/workdir/raw\_data/targets.ark ark:-|
--feat-type plain --splice 5 --learn-rate 0.008
/scratch/user/workdir/raw_data /scratch/user/workdir/rawDev_data dummy-
dir dummy-dir dummy-dir $dir || exit 1;
```

Předtrénování Deep Believe Network algoritmem RBM se spustí skriptem: *steps/nnet/pretrain_dbn.sh*. Příkaz pro trénování neuronové sítě *train_nnet.sh* se rozšíří o parametr `dbn`, kterým specifikujeme, kde je uložena předtrénovaná Deep Believe Network.

Učení, včetně předtrénování, se tedy spustí:

```
$cuda_cmd $dir/_pretrain_dbn.log
steps/nnet/pretrain_dbn.sh --hid_dim 500
/scratch/user/workdir/raw_data $dir || exit 1

dbn=/scratch/user/workdir/exp/nnet/6.dbn

$cuda_cmd $dir/_train_nnet.log
local/nnet/train_nnet.sh --dbn $dbn --num_tgt 2 --hid-dim 500
--hid-layers 6 --learn-rate 0.008
--labels "ark:ali-to-post ark:/scratch/user/raw_data/targets.ark ark:-|"
/scratch/lakosil/workdir/raw_data /scratch/user/rawDev_data dummy-dir
dummy-dir dummy-dir $dir || exit 1;
```

Natrénovaná neuronová síť je potom uložena do souboru *final.nnet*.

4.3.6 Detekce řeči v Kaldi

Vlastní detekce řečové aktivity pomocí natrénované DNN se spouští skriptem *VADdecodeTest.sh*. V tomto skriptu se kromě nastavení běhového prostředí spouští detekce řeči příkazem `nnet-forward`. Jako parametry se musí uvést cesta k feature-transform souboru, cesta k natrénované síti `final.nnet`, cesta k připraveným příznakům pro dekódování a cesta k souboru, kam se má uložit výstupní soubor detektoru. Soubory *final.feature_transform* a *final.nnet* vznikly v trénovací fázi neuronové sítě a je v nich uložena kompletní sktruktura naučené sítě.

Ve výstupním souboru jsou po jednoznačném identifikátoru promluvy a úvodní hranaté závorce 2 sloupce, kde první sloupec odpovídá pravděpodobnosti aktivace neuronu reprezentující řečovou pauzu, druhý sloupec odpovídá pravděpodobnosti aktivace neuronu reprezentující řečovou aktivitu. Formát výstupního souboru je následující:

```
fdhc0_si1559 [
0.9907688 0.009231115
0.9937748 0.006225155
0.9948644 0.005135663
0.9957378 0.00426215
0.99477 0.005229985
0.9965784 0.003421595
0.9901966 0.009803334
0.9849619 0.01503809
0.6492161 0.3507839
0.8381851 0.1618149
0.8201469 0.1798531
0.7284303 0.2715697
0.6942619 0.3057381
0.3273127 0.6726873
0.002980406 0.9970196
...
```

4.3.7 Vyhodnocení realizace v Kaldi

Pro vyhodnocení úspěšnosti detekce byl použit program *vadcrit*. Výstupní soubor z neuronové sítě bylo proto nutné převést do požadovaného formátu. Pro porovnání s referenčním detektorem bylo nutné převést do tohoto formátu také cílové vektory *targets.ark*. Pro převod byl použit perlovský skript *AliNNoutTest.pl*, který jsem za tímto účelem vytvořil.

Pro vyhodnocení úspěšnosti detekce nebyly použity promluvy, na kterých byla síť trénována, ale skupina promluv testovacího souboru. Vyhodnocení úspěšnosti se potom spustí příkazem `vadcrit file1 file2`, kde `file1` je testovaný výstupní vzorek a `file2` je referenční vzorek vzniklý převodem z cílových vektorů.

Detektor navržený v Kaldi může být součástí složitější implementace (např. hlasového rozpoznávače) realizované v tomto současném state-of-art hlasovém toolkitu. Tento detektor se při experimentech prokázal jako funkční a poskytuje širokou škálu

algoritmů a nástrojů pro zpracování hlasu. O přesnosti detekce je pojednáno v kapitole zabývající se experimenty.

Kapitola 5
Experimentální část

Pro realizaci detektoru řečové aktivity je nutné nejdříve experimentálně najít a ověřit optimální parametry neuronové sítě a vstupní řečové příznaky.

5.1 Evaluační kritéria

Pro posouzení správné funkčnosti VAD detektoru je důležitá metodika vyhodnocení chyb. Při použité metodice vyhodnocení chyb vycházíme z porovnání dvou souborů výsledků detektorů:
- detektoru referenčního, jehož výsledky považujeme za vždy správné - ref[i]. Typicky vzniká manuálním popisem signálu značkami symbolizujícími speech/non-speech. Jelikož manuální popis je velmi náročný, nahrazuje se automatickým popisem založeným na rozpoznávání hranic jednotlivých fonémů
- reálného VAD detektoru - vad[i]

Aby byly soubory porovnatelné, musí mít stejnou délku – N a v obou souborech se vyskytují pouze hodnoty 1 – speech nebo 0 – non-speech. [7]

Pro vyhodnocení parametrů jsem použil program „vadcrit“ – autor Milan Václavík, ČVUT - FEL K331. Tento program vyhodnocuje následující parametry VADu:
- ERR celkový počet chybných rozhodnutí

$$ERR = \sum_{i=0}^{N} |ref[i] - vad[i]| \tag{1}$$

- ERS počet chybných rozhodnutí v řeči “ERror in Speech”

$$ERS = \sum_{i=0}^{N} |ref[i] - vad[i]|.ref[i] \tag{2}$$

- $ERP(ERN)$ počet chybných rozhodnutí v pauze “ERror in Pause” (ERror decission in Noise)

$$ERP = \sum_{i=0}^{N} |ref[i] - vad[i]|.(1 - ref[i]) \tag{3}$$

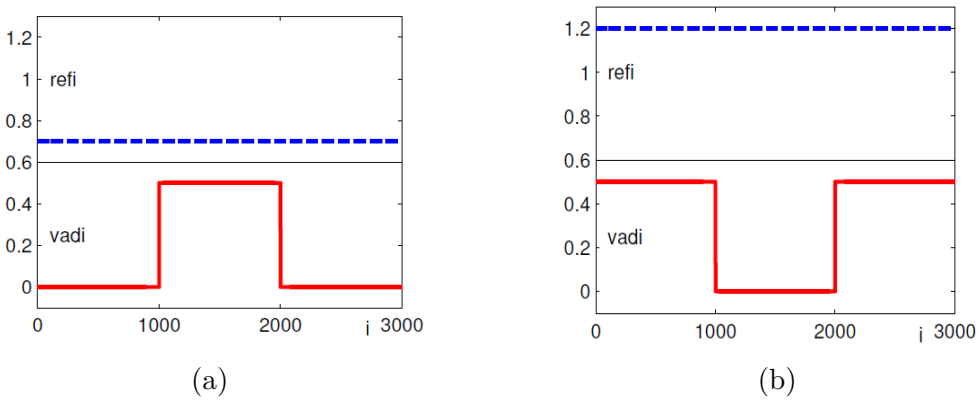
Zřejmě platí:

$$ERR = ERS + ERP \tag{4}$$

Níže jsou popsány základní typy chyb detekce hlasu, které rozlišujeme:

- NDS - “Noise Detected as Speech” - šum je nesprávně detekován jako řeč během non-speech (noise) úseku, je to tedy podmnožinou ERN

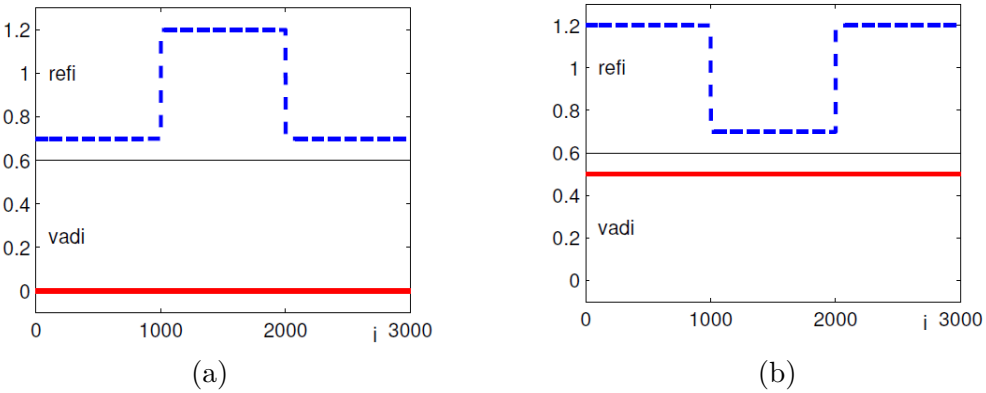
- *SDN* - "Speech Detected as Noise" - řeč je nesprávně detekována jako šum během řečového úseku. Chyba je podmnožinou *ERS*.



Obrázek 5.1. Chyba NDS (a) a chyba SDN (b)

- *MIS* - "Missed Speech" – tato chyba je detekována, pokud úsek řeči není kompletně detekován (začátek i konec). Jedná se o podmnožinu *ERS*.

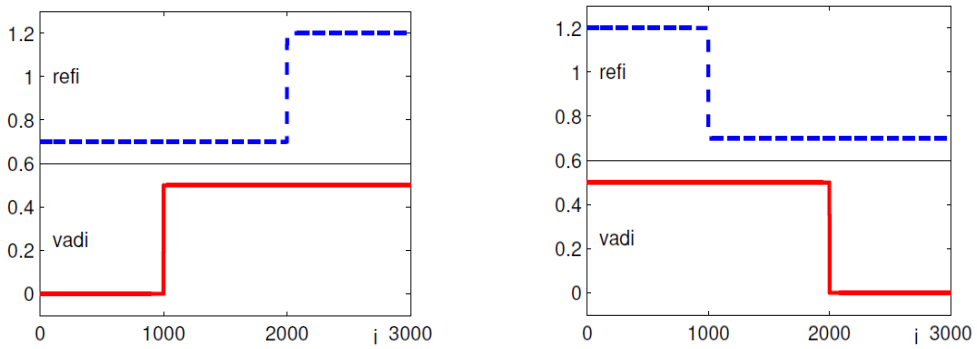
- *MIN* - "Missed Noise" - tato chyba je detekována, pokud celá řečová pauza není detektorem detekována. Typicky se jedná o pauzu mezi slovy. Jedná se o podmnožinu *ERN*.



Obrázek 5.2. Chyba MIS (a) a chyba MIN (b)

- *OVF* - "OVerlap at the Front" - tato chyba je detekována, pokud je řeč detekována dříve než ve skutečnosti má být

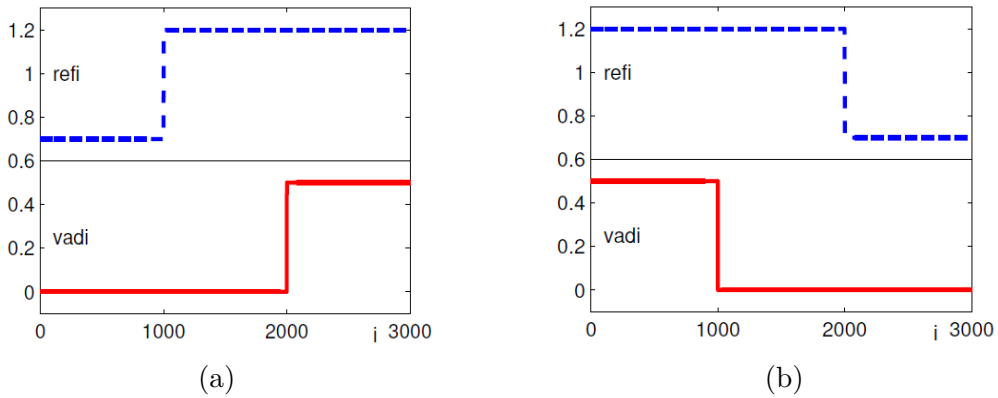
- *OVB* - "OVerlap at the Back" - tato chyba je detekována, pokud je řeč detekována později než ve skutečnosti má být



Obrázek 5.3. Chyba OVF (a) a chyba OVB (b)

- *TRF* - "TRuncation at the Front" - tato chyba je detekována, pokud je řeč detekována s určitým zpožděním

- *TRB* - "TRuncation at the Back" - tato chyba je detekována, pokud je řeč detekována dříve než by ve skutečnosti měla být



Obrázek 5.4. Chyba TRF (a) a chyba TRB (b)

5.2 Použité řečové databáze

Experimenty v této práci jsou založeny na použití dat ze dvou řečových databází - TIMIT a QUT-TIMIT. TIMIT proto, protože tato databáze je velmi dobře foneticky popsána a jedná se o databázi svým způsobem referenční. Jelikož na této databázi bylo již provedeno velmi mnoho experimentů, jsou výsledky snadno porovnatelné s jinými experimenty. Databáze QUT-TIMIT byla použita proto, že jen samotný TIMIT neobsahuje šumové pozadí, což není úplně ideální pro experimenty s detekcí hlasu. Pro detekci hlasu nebo pauzy v řeči v ideálních podmínkách bez přidaného šumu postačí detekce na základě energie signálu. Mnohem složitější úloha je detekovat řeč v signálu, kde vysoká energie signálu nemusí znamenat řeč, ale nějaký hluk, šum, klapot apod.

Protože povaha signálů, se kterými se experimenty v této práci prováděly, je velmi důležitá, bude pojednáno v následujících bodech o obou použitých řečových databázích podrobněji.

5.2.1 Řečová databáze TIMIT

Řečová databáze The DARPA TIMIT Acoustic-Phonetic Continuous Speech Corpus (TIMIT)vznikla na základě iniciativy Defense Advanced Research Projects Agency - Information Science and Technology Office (DARPA-ISTO). Výsledek této práce je poskytován na základě zakoupené licence. Pro experimenty dle této práce je nutné mít zakoupenu tuto licenci. ČVUT tuto licenci má, nicméně na základě licenčních podmínek nejsou data databáze součástí přiloženého SW. Řeč byla namluvena ve společnosti Texas Instruments (TI), transkripce byla pořízena na Massachusetts Institute of Technology (MIT) a databáze je spravována a distribuována na CD-ROM Institutem Standardizace a Technologie (NIST). TIMIT obsahuje celkem 6300 promluv, 10 vět namluvených každým z 630 mluvčích z 8 hlavních dialektů v USA. Tabulka níže ukazuje počet mluvčích 8 dialektů rozdělených podle pohlaví [8].

Dialect Region(dr)	Muži	Ženy	Celkem
1	31 (63%)	18 (27%)	49 (8%)
2	71 (70%)	31 (30%)	102 (16%)
3	79 (67%)	23 (23%)	102 (16%)
4	69 (69%)	31 (31%)	100 (16%)
5	62 (63%)	36 (37%)	98 (16%)
6	30 (65%)	16 (35%)	46 (7%)
7	74 (74%)	26 (26%)	100 (16%)
8	22 (67%)	11 (33%)	33 (5%)

Dialekty v TIMITu:
dr1: New England
dr2: Northern
dr3: North Midland
dr4: South Midland
dr5: Southern
dr6: New York City
dr7: Western
dr8: Army Brat

Promluvy byly rozděleny na trénovací a testovací části takto:

1- cca 25% promluv je určeno pro testování, 75% pro trénování

2- žádný mluvčí není zařazen jak do testovací, tak trénovací části

3- všechny dialekty jsou součástí jak testovací, tak trénovací části (vždy alespoň jeden muž a jedna žena z každého dialektu)

Řečové nahrávky a doprovodná data jsou na CD-ROM organizována dle následujícího schématu:
/CORPUS/USAGE/DIALECT/SEX/SPEAKER_ID/SENTENCE_ID.FILE_TYPE

kde:

CORPUS :== timit
USAGE :== train | test
DIALECT :==$dr1|dr2|dr3|dr4|dr5|dr6|dr7|dr8$ SEX :== m | f
SPEAKER_ID :== INITIALS_DIGIT

kde:

INITIALS :== iniciály mluvčího, 3 písmena
DIGIT :== číslo 0-9 k odlišení mluvčího v případě shodných iniciálů

SENTENCE_ID :== TEXT_TYPE__SENTENCE_NUMBER

kde:

TEXT_TYPE :== $sa|si|sx$

Číslo věty :== 1 ... 2342
FILE_TYPE :== $wav|txt|wrd|phn$

Např.:
/timit/train/dr1/fcjf0/sa1.wav

(TIMIT korpus, trénovací soubor, dialekt region 1, žena, ID-mluvčího „cjf0“, text „sa1“, soubor wav)

/timit/test/df5/mbpm0/sx407.phn

(TIMIT korpus, testovací soubor, dialekt region 5, muž, ID-mluvčího „bpm0“, text „sx407“, soubor fonetické transkripce)

Ke každé promluvě (souboru wav) řečové databáze TIMIT náleží 3 přidružené soubory typu txt, wrd a phn.

.txt – ortografická transkripce
0 61748 She had your dark suit in greasy wash water all year.

.wrd – časově uspořádaná slovní transkripce.
7470 11362 she

11362 16000 had
15420 17503 your
17503 23360 dark
23360 28360 suit
28360 30960 in
30960 36971 greasy
36971 42290 wash
43120 47480 water
49021 52184 all
52184 58840 year

.phn – časově uspořádaná fonetická transkripce

(úvodní a koncová pauza v řeči je označena „h#“, řečová pauza v promluvě je označena „sil“)
0 7470 h#
7470 9840 sh
9840 11362 iy
11362 12908 hv
12908 14760 ae
14760 15420 dcl
15420 16000 jh
16000 17503 axr
17503 18540 dcl
18540 18950 d
18950 21053 aa
21053 22200 r
22200 22740 kcl
22740 23360 k
23360 25315 s
25315 27643 ux
...

5.2.2 Řečová databáze QUT-TIMIT

Velkým problémem při objektivním vyhodnocení funkčnosti hlasových detektorů je nedostatek vhodných řečových databází se šumovým pozadím různých prostředí, s různou úrovní aditivního šumu a přesným fonetickým popisem vhodným pro trénování neuronových sítí a ověření správné klasifikace – detekce hlasu. Tyto požadavky splňuje QUT-NOISE-TIMIT řečová databáze. Skládá se z 600 hodin hlasových signálů navržených přímo pro vyhodnocení přesnosti detekce hlasových detektorů s různým zvukovým pozadím. Databáze vznikla smícháním 10 hodin zvuku pozadí pořízeného v 10 různých prostředích (auto, kavárna, kuchyně…) s hlasovou databází TIMIT. Byly použity různé délky promluv s popisem událostí dle TIMITu a různou úrovní míchání bylo dosaženo různých poměrů hlasového signálu a šumu pozadí (SNR) [9].

Zvuky pozadí, které jsou v této databázi k dispozici:
- CAFE – venkovní kavárna (CAFE-CAFE) a zvuky uvnitř kavárny v nákupním centru

(CAFE-FOODCOURTB). Osahuje řeč pozadí, zvuky z kuchyně a zvuky okolí.
- HOME – zvuky domácí kuchyně (HOME-KITCHEN) a obývacího pokoje (HOME-LIVINGB) při běžných domácích činnostech. Kuchyně obsahuje většinou relativní ticho, přerušované typickými zvuky při vaření a manipulaci s nádobím. Obývací pokoj obsahuje mluvení dětí, televizi a hudbu.
- STREET – zvuky v centru města (STREET-CITY) a zvuky mimo centrum (STREET-KG). Obě pozadí obsahují především zvuky městské dopravy, rušné křižovatky. Centrum města obsahuje také hlasy chodců a ptáků v nedalekém parku.
- CAR – zvuky při jízdě autem s otevřeným oknem (CAR-WINDOWNB) a se zavřeným oknem (CAR-WINUPB). Neobsahuje zvuky rádia nebo řeč na pozadí.
- REVERB – obsahuje zvuk vnitřního bazénu (REVERB-POOL) a uzavřené parkoviště (REVERB-CARPARK). Obě prostředí obsahují hodně ozvěn. Mimoto prostředí vnitřního bazénu obsahuje zvuky šplouchání a tekoucí vody. Parkoviště obsahuje zvuky z nedaleké silnice a občasné zvuky aut.

5.3 Výsledky experimentů

Provedené experimenty jsou z velké části společné pro všechny 3 použité frameworky (Matlab, neural2D, Kaldi). Experimenty vykazovaly pro stejný typ neuronové sítě jen velmi malé odchylky. Proto byla většina experimentů prováděna pouze v prostředí Kaldi. V prostředí Matlabu a neural2D byly prováděny pouze typové testy, aby se ověřilo, že na dané architektuře sítě se úspěšnost výrazně neliší od stejného typu experimentu v Kaldi. Výjimkou jsou experimenty s předtrénováním sítě pomocí RBM. Použitý NN toolbox v Matlabu stejně jako neural2D nepodporuje RBM předtrénování, takže tento typ sítě byl použit pouze v prostředí Kaldi.

Nejdříve byly provedeny experimenty na řečové databázi TIMIT bez přidaného šumu. Experimentováno bylo s různým počtem skrytých vrstev při zachování počtu neuronů ve skrytých vrstvách, předtrénování pomocí DBN (RBM algoritmus) a serializací příznaků (splicing).

5.3.1 Detekce bez aditivního šumu

Experimentálně byl ověřen vliv počtu skrytých vrstev na chybu detekce řečové aktivity. Ostatní parametry sítě byly konstantní: 143 neuronů v jedné vrstvě (N), použito RBM předtrénování sítě (R) a zřetězení příznaků 5 předcházejících na následujících segmentů (S). Stejný experiment byl proveden pro 500 neuronů ve skryté vrstvě. Ostatní parametry byly stejné, aby se ověřilo, jestli vliv počtu skrytých vrstev (L) neovlivňuje počet neuronů ve skryté vrstvě.

S/L/N	ERR	ERS	ERP	S/L/N	ERR	ERS	ERP
5/1/143	2.317	1.051	1.266	5/1/500	2.298	1.016	1.282
5/2/143	2.296	1.014	1.282	5/2/500	2.315	1.037	1.278
5/3/143	2.303	1.006	1.297	5/3/500	2.342	1.047	1.295
5/4/143	2.303	1.008	1.295	5/4/500	2.329	1.037	1.293
5/5/143	2.338	1.037	1.301	5/5/500	2.315	0.993	1.322
5/6/143	2.298	0.987	1.311	5/6/500	2.367	1.016	1.351

Tabulka 5.1. Závislost chyby detekce na počtu vrstev

Výsledky pro 143 a 500 neuronů nepotvrzují, že více než jedna skrytá vrstva měla nezanedbatelný vliv na detekci hlasu za stanovených podmínek.

Dále bylo experimentováno s počtem neuronů ve skrytých vrstvách. Ostatní parametry sítě byly konstantní: 3 skryté vrstvy, použito RBM předtrénování sítě a zřetězení příznaků 5 předcházejících a 5 následujících segmentů (splice 5). Stejný experiment byl proveden pro 6 skrytých vrstev. Ostatní parametry byly stejné.

S/N/L	ERR	ERS	ERP	S/N/L	ERR	ERS	ERP
5/143/3	2.303	1.006	1.297	5/143/6	2.298	0.987	1.311
5/500/3	2.342	1.047	1.295	5/500/6	2.375	1.088	1.287
5/750/3	2.323	1.011	1.312	5/750/6	2.345	1.022	1.323
5/1000/3	2.314	1.003	1.311	5/1000/6	2.336	1.014	1.322
5/2000/3	2.526	1.137	1.389	5/2000/6	2.439	1.049	1.390

Tabulka 5.2. Závislost chyby detekce na počtu neuronů

Vliv počtu neuronů ve skrytých vrstvách byl ověřen i pro neuronové sítě bez předtrénování. Experimentováno bylo s jednou a třemi skrytými vrstvami (síť typu MLP), zřetězení příznaků 5 předcházejících a 5 následujících segmentů (splice 5).

S/N/L	ERR	ERS	ERP	S/N/L	ERR	ERS	ERP
5/143/1	2.431	1.076	1.355	5/143/3	2.634	1.146	1.488
5/500/1	2.505	1.084	1.421	5/500/3	2.644	1.111	1.533
5/750/1	2.313	1.026	1.287	5/750/3	2.454	1.133	1.321
5/1000/1	2.414	1.074	1.340	5/1000/3	2.512	1.124	1.388
5/2000/1	2.422	1.070	1.353	5/2000/3	2.521	1.122	1.399
5/10000/1	2.614	1.140	1.474	5/10000/3	3.186	1.422	1.764

Tabulka 5.3. Závislost chyby detekce na počtu neuronů - síť MLP

Z experimentů vyplývá, že počet neuronů ve skrytých vrstvách nemá velký vliv na

přesnost detekce nezašuměných signálů. Při vyšších počtech v tisících neuronů už přesnost detekce klesá, zřejmě vlivem overfittingu. Mnohem větší vliv na přesnost detekce byla zjištěna při použití RBM předtrénování. Přesnost se významně zvýšila dokonce při jedné skryté vrstvě. Je možné, že v případě jedné skryté vrstvy by se dalo jistého zlepšení dosáhnout více pokusy trénování (nastavení vah je v tomto případě inicializováno na náhodné hodnoty). Dalším experimentováním s learning rate apod. Optimalizace MLP s jednou skrytou vrstvou není ale součástí zadání této práce, kde se věnujeme pouze hlubokým neuronovým sítím a MLP sítě s jednou skrytou vrstou jsou zde použity pouze pro srovnání výsledků.

Následně byl ověřen vliv zřetězení příznaků. Je-li v tabulce parametr „S“ 0 znamená to, že byly použity kepstrání koeficienty pouze aktuálního segmentu. 40 znamená, že byly jako příznaky zřetězeny kepstrální koeficienty 40 předchozích segmentů a 40 následujících. Experimentováno bylo s jednou a třemi skrytými vrstvami (síť typu MLP bez předtrénování), 300 neuronů ve skryté vrstvě.

S/N/L	ERR	ERS	ERP	S/N/L	ERR	ERS	ERP
0/300/1	7.356	4.217	3.139	0/300/3	7.475	4.238	3.237
1/300/1	4.436	2.393	2.042	1/300/3	4.614	2.373	2.241
2/300/1	3.403	1.586	1.817	2/300/3	3.460	1.576	1.884
3/300/1	2.870	1.245	1.625	3/300/3	2.865	1.244	1.621
4/300/1	2.623	1.158	1.464	4/300/3	2.621	1.156	1.465
5/300/1	2.400	1.084	1.315	5/300/3	2.450	1.141	1.309
6/300/1	2.300	1.024	1.276	6/300/3	2.267	1.025	1.242
7/300/1	2.086	0.863	1.223	7/300/3	2.073	0.862	1.211
8/300/1	1.991	0.863	1.128	8/300/3	1.989	0.862	1.127
9/300/1	1.918	0.845	1.074	9/300/3	1.913	0.844	1.069
10/300/1	1.987	0.902	1.084	10/300/3	1.863	0.857	1.006
20/300/1	2.172	0.954	1.218	20/300/3	2.148	0.995	1.152
40/300/1	2.604	1.043	1.561	40/300/3	2.356	0.977	1.379

Tabulka 5.4. Závislost chyby detekce na délce zřetězených příznaků - síť MLP

Z výsledků experimentů vyplývá silná závislost přesnosti detekce hlasu na počtu zřetězených příznaků. Nejlepších hodnot bylo dosaženo při zřetězení příznaků 10 po sobě jdoucích segmentů. V dalších experimentech bylo použito zřetězení 10 příznaků před a 10 po aktuálním segmentu jakožto optimální hodnoty.

5.3.2 Nepřizpůsobený detektor

Další experimenty probíhaly na řečové databázi TIMIT s přidaným aditivním šumem a signálem okolního prostředí (QUT-TIMIT). Detektor natrénovaný na čistém TIMITu byl potom testován signálem z různého prostředí.

R/L/N/Env/SNR	ERR	ERS	ERP
N/3/300/CAR-WINDOWNB-1/00	48.462	1.303	47.159
N/3/300/CAR-WINDOWNB-1/+15	43.396	0.607	42.790
A/3/1100/CAR-WINDOWNB-1/+15	43.178	0.718	42.460
N/3/300/HOME-KITCHEN-1/00	47.010	1.517	45.493
N/3/300/HOME-KITCHEN-1/+15	53.559	6.051	47.507
A/3/1100/HOME-KITCHEN-1/+15	49.878	3.132	46.746
N/3/300/CAFE-CAFE-1/00	49.012	1.117	47.896
N/3/300/CAFE-CAFE-1/+15	49.768	0.974	48.793
A/3/1100/CAFE-CAFE-1/+15	48.983	0.986	47.997
N/3/300//HOME-LIVINGB-1/00	50.789	0.845	49.944
N/3/300//HOME-LIVINGB-1/+15	50.363	0.921	49.442
A/3/1100/HOME-LIVINGB-1/+15	49.766	0.910	48.856
N/3/300/REVERB-POOL-1/00	51.189	0.769	50.420
N/3/300/REVERB-POOL-1/+15	51.572	0.705	50.867

Tabulka 5.5. Závislost chybovosti nepřizpůsobeného detektoru na prostředí

Detektor natrénovaný na čistém TIMITu při detekci signálu s přidaným šumem SNR 15 dB a 0 dB zcela selhává. Chyba se projevuje zejména chybou v pauze (ERP), což znamená, že detektor šum neignoruje a považuje jej za řečový signál. Chyba detekované řeči (ERS) je naopak malá. Detektor tedy považuje jakýkoli signál za řeč. V podstatě funguje jako energetický detektor. Na čistém TIMITu se naučil nejjednodušší možnost detekce, která v prostředí bez šumu plně postačovala.

5.3.3 Přizpůsobený detektor

Další experimenty probíhaly na řečové databázi s přidaným aditivním šumem a signálem okolního prostředí (QUT-TIMIT). Detektor byl natrénovaný na QUT-TIMITu SNR 0 dB a 15 dB a tento byl potom testován signálem ze stejného prostředí, ve kterém byl trénován.

R/L/N/Env/SNR	ERR	ERS	ERP
N/1/300/CAR-WINUPB-1/00	6.439	3.041	3.398
A/6/1100/CAR-WINUPB-1/00	4.626	2.025	2.501
N/1/300/CAR-WINUPB-1/15	5.671	2.847	2.824
A/6/1100/CAR-WINUPB-1/15	4.612	1.801	2.811
A/3/300/CAR-WINDOWNB-1/15	5.670	2.846	2.824
N/1/300/HOME-KITCHEN-1/00	7.595	4.079	3.516
A/6/1100/HOME-KITCHEN-1/00	5.222	2.422	2.800
N/1/300/HOME-KITCHEN-1/15	4.050	2.117	1.933
N/3/300/HOME-KITCHEN-1/15	4.708	2.341	2.368
A/6/1100/HOME-KITCHEN-1/15	4.454	2.243	2.211
N/1/300/REVERB-POOL-1/00	55.186	0.000	55.186
A/6/1100/REVERB-POOL-1/00	11.310	6.772	4.539
N/1/300/REVERB-POOL-1/15	55.705	2.734	2.972
A/6/1100/REVERB-POOL-1/15	4.553	2.276	2.277

Tabulka 5.6. Závislost chybovosti přizpůsobeného detektoru na prostředí

Z výsledků experimentů je zřejmé, že DNN byla schopna výrazně lépe detekovat řeč než klasická třívrstvá neuronová síť s jednou skrytou vrstvou. Např. v prostředí bazénu s vysokým echem nebyl detektor s jednou skrytou vrstvou schopen správně detekovat řeč ani při SNR 15 dB s chybou 55%, zatímco hluboká neuronová síť pracovala i v tomto prostředí s chybovostí 4,5%. Při zvýšení úrovně šumu na SNR 0 dB došlo sice u DNN ke zhoršení detekce na 11,3%, ale třívrstvá síť vykazovala chybu v řeči 0% (celková chyba 55%), což znamená, že ani jeden segment nebyl detekován jako pauza v řeči. Šum tedy v tomto případě úplně překryl řeč. DNN si tedy poradila i v takto zašuměném prostředí s přidaným echem. Výstup chybovosti v tomto prostředí je dále ukázán podrobněji včetně dalších chybových parametrů.

TYPE	%	COUNT
ERR:	11.310%	33916
ERS:	6.772%	20306
ERP:	4.539%	13610
NDS:	3.585%	10751
SDN:	6.384%	19144
MIS:	0.000%	0
MIN:	0.070%	209
OVB:	0.785%	2354
OVF:	0.064%	191
TRB:	0.403%	1208
TRF:	0.005%	15
BEE:	0.000%	0

Tabulka 5.7. Chybovost 6 L/1100 N, bazén, 0 dB - podrobně

TYPE	%	COUNT
ERR:	5.705%	17108
ERS:	2.734%	8197
ERP:	2.972%	8911
NDS:	0.712%	2136
SDN:	2.725%	8172
MIS:	0.000%	0
MIN:	0.200%	601
OVB:	2.008%	6022
OVF:	0.038%	115
TRB:	0.013%	39
TRF:	0.003%	10
BEE:	0.000%	0
RES:	0.004%	13

Tabulka 5.8. Chybovost 1 L/300 N, bazén, 15 dB - podrobně

TYPE	%	COUNT
ERR:	4.929%	14780
ERS:	2.537%	7608
ERP:	2.392%	7172
NDS:	0.230%	691
SDN:	2.509%	7523
MIS:	0.000%	0
MIN:	0.209%	626
OVB:	1.925%	5772
OVF:	0.019%	57
TRB:	0.027%	80
TRF:	0.005%	16
BEE:	0.000%	0
RES:	0.005%	15

Tabulka 5.9. Chybovost 6 L/1100 N, CAR-WINUPB, 0 dB - podrobně

5.3.4 Detektor s rozšířeným trénovacím souborem

Poslední část experimentů testuje vliv trénování neuronové sítě na několik prostředí. Trénovací soubor byl v těchto experimentech rozšířen na několik prostředí (QUT-TIMIT). Testovací soubor byl z jednoho tohoto prostředí, na který byl řečový detektor natrénován.

Detektor natrénovaný na signálu s přidaným šumem SNR 15 dB - auto, kuchyně SNR 15 dB a 0 dB (testovací vzorky, auto - otevřené okno, SNR 15 dB)

L/N	ERR	ERS	ERP
1/300	4.793	2.231	2.562
6/1100	4.179	2.080	2.099

Tabulka 5.10. Chybovost CAR-WINUPB, 0 dB, HOME-KITCHEN, 15 dB a 0 dB

Detektor natrénovaný na signálu s přidaným šumem SNR 15 dB - auto, kuchyně SNR 15 dB a 0 dB (testovací vzorky, kuchyně SNR 15 dB)

L/N	ERR	ERS	ERP
1/300	4.948	2.401	2.547
6/1100	4.050	2.117	1.933

Tabulka 5.11. Chybovost CAR-WINUPB, 15 dB, HOME-KITCHEN, 15 dB a 0 dB

Kapitola 6
Závěr

Cílem této práce bylo navrhnout a implementovat detektor řečové aktivity s využitím hlubokých neuronových sítí (DNN). Součástí zadání bylo nalezení vhodné příznakové sady a struktury sítě a následné otestování detektoru. Práce popisuje principy detekce hlasu s hlavním zaměřením na DNN a následným návrhem detektoru. Samotný detektor byl implementován v prostředí Matlab s Neural Network Toolboxem, neural2D a Kaldi.

Použití detektoru navrženého v prostředí Matlab je pro experimentální, akademické nebo prototypové použití. Detektor navržený v neural2D naopak pro svoji kompaktnost a jednoduchý kód v C++ může být snadno implementovatelný v různých embedded zařízeních jako jsou různé detektory v komunikační technice, Internetu věcí (IoT) apod. Detektor navržený v Kaldi může být součástí složitější implementace (např. hlasového rozpoznávače) realizované v tomto současném state-of-art hlasovém toolkitu.

Pro všechny tři typy implementace byly vypracovány skripty pro přípravu trénovacích a testovacích dat. Jako sada příznaků bylo vybráno použití mel - kepstrálních koeficientů (MFCCs). Použito bylo celkem 13 kepstrálních koeficientů (včetně nultého kepstrálního koeficientu). Experimentálně bylo zjištěno, že nejlepších výsledků bylo dosaženo zřetězením koeficientů z 10 předcházejících a 10 následujících segmentů signálu.

Přínos hluboké neuronové sítě oproti MLP síti s jednou skrytou vrstvou se začíná uplatňovat až u signálů s přidaným šumem pozadí. Důležitější než hloubka sítě se jeví způsob trénování sítě a výběr příznaků řeči. Detektor, který byl trénován na čistém signálu bez přidaného aditivního šumu, nedokázal se zašuměným signálem korektně pracovat. Nejlépe naopak dopadly experimenty, kdy se pracovní podmínky detektoru co nejvíce blížily tomu, jak byla neuronová síť detektoru natrénována. Detektor, který používal pro detekci pouze příznaky aktuálního segmentu, vykazoval výrazně horší výsledky, ačkoli stále ještě omezeně použitelné. Nejlepší výsledky detektoru na nezašuměném signálu se pohybovaly těsně pod 2% chybovostí, při 15 dB SNR potom kolem 4% chybovosti a při 0 dB SNR kolem 5% chybovosti. Experimentálně byl potvrzen předpoklad, že v zašuměném prostředí detekuje optimálně navržená a natrénovaná hluboká neuronová síť hlas lépe než základní třívrstvá síť.

Příloha A
Zadání práce

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Lakosil** Jméno: **Mojmír** Osobní číslo: **457325**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra teorie obvodů**

Studijní program: **Komunikace, multimédia a elektronika**

Studijní obor: **Komunikační systémy**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Detektor řečové aktivity na bázi DNN

Název diplomové práce anglicky:

DNN-Based Voice Activity Detector

Pokyny pro vypracování:

1. Seznamte se s problematikou detekce řečové aktivity s užším zaměřením na řešení využívající umělé neuronové sítě.
2. Navrhněte řešení s využitím hlubokých neuronových sítí (DNN) a vlastní detektor implementujte pomocí volně dostupných nástrojových sad.
3. Nalezněte vhodnou strukturu sítě i příznakovou sadu pro účely detekce řečové aktivity a zvažte použití vhodných trénovacích dat z dostupných řečových databází.
4. Otestujte spolehlivost detektoru na signálech z dostupných databází případně v reálném provozu.

Seznam doporučené literatury:

[1] J. Psutka, L. Müller, J. Matoušek, V. Radová. Mluvíme s počítačem česky. Academia 2006.
[2] X. Huang, A. Acero, H.-W. Hon. Spoken Language Processing. Prentice Hall, 2001.
[3] D. Yu, L. Deng. Automatic Speech Recognition A Deep Learning Approach. Springer-Verlag London. 2015
[4] D. Povey et al, The Kaldi Speech Recognition Toolkit. In Proc. of IEEE 2011 ASRU, Hawai, US, 2011. Note. Project WEB-page http://kaldi.sourceforge.net/.
[5] G. Ferroni, R. Bonfigli, E. Principi, S. Squartini, and F. Piazza. A deep neural network approach for voice activity detection in multi-room domestic scenarios. In Proc. of IJCNN, Killarney, Ireland, Jul. 12-17 2015, pp. 1?8.

Jméno a pracoviště vedoucí(ho) diplomové práce:

doc. Ing. Petr Pollák CSc., katedra teorie obvodů FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **08.02.2017** Termín odevzdání diplomové práce: **26.05.2017**

Platnost zadání diplomové práce: **30.09.2018**

III.

15.2.2017
Datum převzetí zadání

Podpis studenta

Příloha B
Obsah přiloženého CD

Přiložené CD obsahuje následující soubory:

- text diplomové práce ve formátu PDF
- adresář *kaldi* obsahuje soubory detektoru řeči realizovaného ve frameworku Kaldi
- adresář *matlab* obsahuje soubory detektoru řeči realizovaného v Matlabu s NN toolboxem
- adresář *neural2d* obsahuje soubory detektoru řeči realizovaného v neural2D frameworku
- adresář *vadcrit* obsahuje sw pro vyhodnocení detekce hlasu

Literatura

[1] J. Psutka, L. Müller, J. Matoušek, V. Radová. Mluvíme s počítačem česky. Academia 2006.

[2] D. Yu, L. Deng. Automatic Speech Recognition A Deep Learning Approach. Springer-Verlag London. 2015.

[3] Uhlíř J. a kol.: Technologie hlasových komunikací. Nakladatelství ČVUT, Praha, 2007

[4] Tučková J.: Vybrané aplikace umělých neuronových sítí při zpracování signálů. Nakladatelství ČVUT, Praha, 2009

[5] Rajnoha, J. - Pollák, P.: Detektory řečové aktivity na bázi perceptivní kepstrální analýzy. In Technical Computing Prague 2008. Praha: Humusoft, 2008, díl 1, s. 1-9. ISBN 978-80-7080-692-0

[6] G. Hinton, L. Deng, D. Yu, , G. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, B. Kingsbury. A deep neural network for Acoustic Modeling in Speech Recognition. IEEE Signal Processing Magasine, Nov. 2012

[7] Pollak: Criteria for VAD classification. Internal research report R02-1, Dec 2002.

[8] Garofolo, John, et al. TIMIT Acoustic-Phonetic Continuous Speech Corpus LDC93S1. Web Download. Philadelphia: Linguistic Data Consortium, 1993.

[9] Dean, David B. and Sridharan, Sridha and Vogt, Robert J. and Mason, Michael W., září 2010 The QUT-NOISE-TIMIT corpus for the evaluation of voice activity detection algorithms Interspeech 2010, International Convention Complex, Makuhari, Japan.