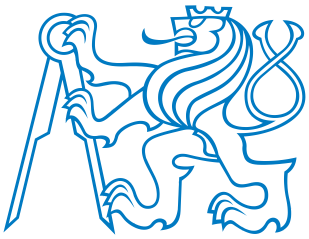
diplomová práce

Analýza robustnosti moderních rozpoznávačů řeči na bázi TANDEM architektury

Bc. Aleš Brich



květen 2016

Doc. Ing. Petr Pollák, CSc.

České vysoké učení technické v Praze
Fakulta elektrotechnická, Katedra kybernetiky

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: Bc. Aleš B r i c h

Studijní program: Kybernetika a robotika

Obor: Robotika

Název tématu: Analýza robustnosti moderních rozpoznávačů řeči na bázi TANDEM architektury

Pokyny pro vypracování:

1. Seznamte se s principy rozpoznávání řeči s užším zaměřením na systémy na bázi TANDEM architektury s příznaky odhadovanými na bázi hlubokých neuronových sítí (DNN).
2. Implementujte TANDEM rozpoznávač s různými variantami příznakových vektorů (aposteriorní příznaky, Bottleneck příznaky, artikulační příznaky). Systém realizujte s nástroji KALDI. Na základě zavedených konvencí vytvořte pro danou úlohu skripty („recepty“) umožňující jejich využití na řešitelském pracovišti resp. odbornou komunitou v oblasti rozpoznávání řeči.
3. Analyzujte robustnost navržených systémů na úlohách rozpoznávání spojité řeči resp. rozpoznávání povelů v nepříznivých akustických podmínkách (jedoucí automobil, veřejné prostory) a srovnejte s výsledky dosahovanými pro řeč snímanou s nižší úrovní rušivého pozadí.

Seznam odborné literatury:

[1] J. Psutka, L. Müller, J. Matoušek, V. Radová. Mluvíme s počítačem česky. Academia, 2006.
[2] J. Uhlíř. a kol. Technologie hlasových komunikací. Nakladatelství ČVUT, Praha, 2007.
[3] X. Huang, A. Acero, H.-W. Hon. Spoken Language Processing. Prentice Hall, 2001.
[4] R. Su et al, "Efficient Use of DNN Bottleneck Features in Generalized Varaible Parameter HMMs for Noise Robust Speech Recognition," In Proc of Interspeech 2015, Dresden, Germany, 201ř.
[5] G. Hinton et al, "Deep Neural Networks for Acoustic Modeling in Spech Recognition“ The Shared Views of Four Research Groups," IEEE Signal Processing Magazine, vol. 29, no. 6, pp. 82-97, 2012.
[6] 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/.

Vedoucí diplomové práce: doc. Ing. Petr Pollák, CSc.

Platnost zadání: do konce letního semestru 2016/2017

L.S.

prof. Dr. Ing. Jan Kybic
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 11. 12. 2015

Poděkování

Rád bych vyjádřil vděk svému vedoucímu Doc. Ing. Petru Pollákovi, CSc. za výborné vedení a cenné rady s danou problematikou. Také bych rád poděkoval Ing. Petru Mizerovi za konstruktivní připomínky a technickou podporu. Dále bych rád poděkoval za výpočetní zdroje poskytnuté CESNET LM2015042 a CERIT Scientific Cloud LM2015085 v rámci programu "Projects of Large Research, Development, and Innovations Infrastructures".

Prohlášení

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.

V Praze dne _____ _____

podpis

Abstrakt

Tato práce se zabývá analýzou robustnosti rozpoznávače řeči na bázi TANDEM architektury. Cílem je zjistit, jaký vliv na úspěšnost rozpoznávání mají různé varianty příznakových vektorů, s užším zaměřením na příznaky odhadovanými vícevrstvými sítěmi. K implementaci je použit široce používaný balíček nástrojů Kaldi. Pro splnění cíle práce byl vytvořen tzv. recept, který využívá zavedených konvencí Kaldi nástrojů k modulárnímu sestavení experimentů. Základním zdrojem řečových signálů je databáze SPEECON, která obsahuje signály nahrávané v různých prostředích čtyřmi mikrofony. Pro každé prostředí jsou tedy dostupná data ze čtyř různě kvalitních kanálů. Robustnost je testována na všech dostupných prostředích databáze SPEECON. Pro většinu prostředí bylo dosaženo uspokojivých výsledků, kde se TANDEM systém ukázal jako robustnější a úspěšnější než standardní řešení a to v průměru o přibližně 5 %.

Klíčová slova

rozpoznávání řeči; TANDEM architektura; umělé neuronové sítě; Kaldi; Metacentrum

Abstract

This paper deals with the analysis of robustness of a speech recognizer based on the TANDEM architecture. The main goal is to find out which types of the TANDEM architecture feature vectors improve the classification accuracy. The influence of the multi-layer artificial neural network feature vectors is observed in more detail. The implementation is based on the free, widely spread tool called Kaldi and based on its conventions, the Kaldi recipe was created. The main source of data is the SPEECON database which contains the signals recorded in the different environments with the four microphone channels of a different quality of recording. The robustness is tested on all the available environments of the SPEECON database. The satisfying results were achieved for the most of the environments where the TANDEM architecture outperformed the standard approach about 5% WER on average.

Keywords

speech recognition; TANDEM features; artificial neural networks; Kaldi; Metacentrum

Obsah

Seznam obrázků

Použité zkratky

MFCC	Mel-freqvency Cepstral Coeficient (Melovské kepstrální koeficienty)
TRAPs	Temporal Patterns (Časové příznaky)
FFT	Fast Fourier Transform (Rychlá Fourierova transformace)
DFT	Discrete Fourier Transform (Diskrétní Fourierova transformace)
IDFT	Inverse Discrete Fourier Transform (Inverzní diskrétní Fourierova transformace)
DCT	Discrete Cosine Transform (Diskrétní Kosinova transformace)
PCA	Principal Component Analysis (Metoda hlavních komponent)
ANN	Artificial Neural Network (Umělá neuronová síť)
MLP	Multilayer Perceptron (Vícevrstvý perceptron)
HMM	Hidden Markov Model (Skrytý Markovův model)
WER	Word Error Rate
SGD	Stochastic gradient descent
CMVN	Cepstral Mean Variance Normalization
LVCSR	Large Vocabulary Common Speech Recognizer
ASR	Automatic Speech Recognition

1 Úvod

Masivní rozšíření technologií v našem okolí vyžaduje hledání nových způsobů jejího ovládání. Řeč jako prostředek k přenosu informace, je mezi lidmi nejběžnější způsob komunikace. S rozvojem techniky, má strojové rozpoznávání řeči stále větší potenciál stát se majoritním rozhraním k ovládání technologií v našem okolí. Může se jednat o hlasové příkazy a tedy úlohu rozpoznávání s malým slovníkem. Další možností je hlasový vstup spojité, přirozené řeči, kdy je nutné použít LVCSR (Large Vocabulary Common Speech Recognizer), tedy rozpoznávač s velkým slovníkem.

Úloha rozpoznávání řeči je výpočetně velmi náročná a proto její rozvoj jde ruku v ruce s rozvojem výpočetního výkonu počítačů. V dnešní době se rychle rozvíjí aplikace umělých neuronových sítí (ANN - Artificial Neural Network) a to ve formě extrakce příznaků, DNN-HMM nebo samostatného rozpoznávače. ANN dokáží z akustické informace, případně předzpracovaných příznakových vektorů, vyseparovat užitečnou informaci k vytvoření příznakových vektorů s vysokou informační hustotou, například tzv. Bottleneck sítě. Také mohou poskytnout odhad pravděpodobnosti při evaluaci skrytých Markovových modelů, pak hovoříme o DNN-HMM rozpoznávačích, jako alternativách pro standardní GMM-HMM. Samozřejmě mohou tvořit samostatný rozpoznávač, kde výstupem jsou pravděpodobnosti příslušnosti vstupního příznakového vektoru k hláskám (Multi-Layer Perceptron, DNN).

Cílem této práce je popsat a implementovat rozpoznávač na principu umělých neuronových sítí a skrytých Markovových řetězců s použitím TANDEM architektury příznaků. Příznakové vektory v případě TANDEM architektury tvoří dva typy příznakových vektorů, transformovaných a spojených do jednoho. Implementace bude realizována hlavně s použitím nástrojů Kaldi na výpočetním clusteru Metacentum, které je součástí Národní Gridové Iniciativy.

Práce je členěna do následujících kapitol. První kapitola diplomové práce se věnuje popisu rozpoznávačů obecně, výpočtu základních typů příznaků, jako jsou Melovské kepstrální koeficienty a perceptivní lineární prediktivní analýza. Také popisuje akustické modelování pomocí GMM-HMM a zvýšení úspěšnosti rozpoznávání pomocí slučování základních slovních jednotek. Dále se dotýká jazykového modelování a dekódování. Další kapitola podrobněji rozebírá umělé neuronové sítě, některé jejich různé typy, použití a metody trénování. Také hovoří o samotné TANDEM architektuře příznakových vektorů.

Popisuje TRAP příznaky, které se dají považovat za předchůdce TANDEM systémů a hovoří o metodě hlavních komponent jako o transformaci na dekorelaci a snížení dimenze příznakových vektorů. V implementační kapitole je představen balíček nástrojů Kaldi, CtuCopy a výpočetní cluster Metacentrum, s poznámkami k instalaci a použití těchto nástrojů. V kapitole popisující experimenty je nejprve představena použitá databáze a následně uspořádání a výsledky jednotlivých experimentů, které shrnuje závěrečná kapitola.

2 Princip rozpoznávání řeči

Standardní a nejpoužívanější typ rozpoznávače je na bázi skrytých Markovových modelů. Slova jsou definována zřetězenými HMM modelujícími subslovní elementy (monofóny, trifóny, ...). Parametry Markovova modelu pro každý subslovní HMM jsou nastaveny během trénování. Neznámá promluva je rozpoznána jako řetězec odpovídajících subslovních HMM, jež generuje nejvyšší aposteriorní pravděpodobnost. Tato metoda v současné době zcela vytlačila porovnávání se vzory díky své univerzálnosti a robustnosti.

Rozpoznávání pomocí HMM v současné době tvoří dva hlavní přístupy. Standardní GMM-HMM, kde jsou pravděpodobnostní funkce HMM modelovány pomocí gaussovských funkcí. Další variantou je DNN-HMM, který místo gaussovských funkcí používá pravděpodobnosti vygenerované neuronovou sítí. Tento přístup se začal používat až s rozvojem výpočetní kapacity počítačů. Alternativní přístup je použití TANDEM příznaků s kombinaci se standardním GMM-HMM rozpoznávačem.

2.1 Základní schéma rozpoznávače

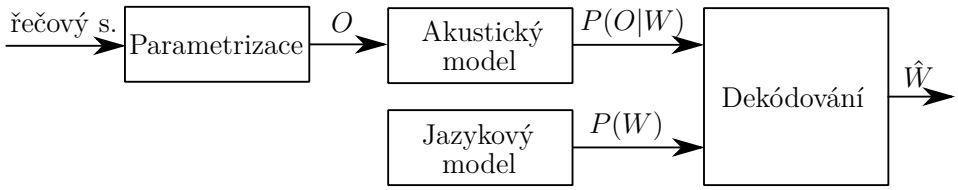
Úloha rozpoznávání řeči je komplexní problém, který se dá rozdělit na několik samostatných úkolů. Parametrizace a výběr příznaků, akustické a jazykové modelování, dekódování. Každá část je kritická pro úspěšné rozpoznání s vysokou úspěšností. Pokud označíme $W = \{w_1,\ ...\ w_N\}$ jako posloupnost slov a $O = \{o_1,\ ...\ o_T\}$ jako posloupnost příznakových vektorů, můžeme říct, že hledáme posloupnost slov $\hat{W}$ takovou, která maximalizuje podmíněnou pravděpodobnost $P(W|O)$ (nejpravděpodobnější posloupnost slov pro danou posloupnost příznakových vektorů). Odpovídající rovnice a její další rozklad podle Bayesova pravidla ukazuje 2.1.1.

$$\hat{W} = \arg\max_{W} P(W|O) = \arg\max_{W} \frac{P(W)P(O|W)}{P(O)} \tag{2.1.1}$$

Při hledání maxima můžeme ignorovat $P(O)$. $P(W)$ reprezentuje pravděpodobnost určité posloupnosti slov. $P(O|W)$ je pravděpodobnost, že určitá posloupnost slov generuje posloupnost příznakových vektorů. Rovnice 2.1.1 se redukuje na maximalizaci sdružené pravděpodobnosti tak, jak ukazuje rovnice 2.1.2.

$$\hat{W} = \arg\max_{W} P(W, O) = \arg\max_{W} P(W)P(O|W) \tag{2.1.2}$$

Rozpoznávače řeči se tedy skládají ze čtyř hlavních funkčních bloků, které ilustruje obrázek 2.1.1. Parametrizace představuje transformaci převádějící řečový signál na posloupnost příznakových vektorů O. Z těch se pomocí akustického modelu generuje pravděpodobnost $P(O|W)$. Ta se spolu s pravděpodobností $P(W)$ získané z jazykového modelu, dekóduje ve výslednou posloupnost $\hat{W}$. Dekódování reprezentuje hledání $\arg\max_W$. Zatímco akustický model modeluje *jak* mluvíme, jazykový model modeluje *co* říkáme.



Obrázek 2.1.1 Blokové schéma rozpoznávače řeči.

2.2 Parametrizace

Řeč je akustický signál a dá se popsat mnoha způsoby, časovým průběhem vlny, spektrogramem, kepstry a dalšími. Každé dvě promluvy se z fyzikálního hlediska snadno liší a to i když mají shodný obsah, pronesený stejnou osobou ve stejném prostředí. Pro rozpoznání obsahu promluvy je vhodné použít takovou reprezentaci řeči, která se pro různé podmínky liší co nejméně, obsahuje co nejvíce užitečné informace a naopak minimum té neužitečné.

Při optimálně zvolené parametrizaci řeči se příznaky pro základní subslovní akustický element příliš neliší napříč mluvčími, ale naopak se dostatečně oddělitelně liší pro různé fóny. Míra úspěšnosti dosažení tohoto předpokladu má markantní dopad na celkovou úspěšnost rozpoznávání. Vstupem pro parametrizaci je akustický signál v časové oblasti. Výstupem je sekvence příznakových vektorů O.

Existuje mnoho typů příznaků. Spektrální, různé typy kepstrálních příznaků, příznaky získané výstupem z jiného rozpoznávače, například neuronové sítě, nebo přímé použití vzorků akustického záznamu v časové oblasti, se kterým se začalo experimentovat až s rozvojem výpočetní síly počítačů a umělých neuronových sítí.

Většina příznaků vychází z amplitudového, nebo výkonového spektra. Při jeho výpočtu se provádí úpravy, které podchycují některé vlastnosti řeči za účelem jejich kompenzace.

Preemfáze koriguje útlum amplitud spektrálních složek ve vyšších frekvencích. Útlum je přibližně -20dB/dek od 100Hz. Výsledkem preemfáze je vyrovnání energetického spektra v celém pásmu.

Řeč je v krátkých časových úsecích kvazistacionární. Právě proto má výpočet krátko-

dobého spektra význam. Signál je však nutné segmentovat tak, aby byly zachyceny tyto kvazistacionární úseky. Typická délka segmentu je typicky 20-30 ms. Tyto segmenty se váhují, většinou Hammingovým oknem, kvůli omezení prosakování ve spektru při DFT.

Takto upravený segment je možno převést z časové oblasti do frekvenční. Diskrétní Fourierova transformace používaná k tomuto účelu, je dána vztahem 2.2.1. Amplitudové spektrum je pak dáno jako $|S(k)|$, výkonové $\frac{1}{N}|S(k)|^2$, kde N je počet vzorků signálu.

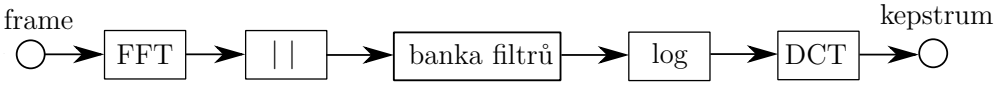
$$S[k] = \sum_{n=0}^{N-1} s[n] \exp\left(-j\frac{2\pi}{N}kn\right), \text{ pro k=0, 1, ..., N-1} \tag{2.2.1}$$

2.2.1 Melovské kepstrální koeficienty

Krátkodobé DFT spektrum je pro rozpoznávání nevhodné, protože obsahuje příliš mnoho nadbytečné informace. Proto se z něj ve většině případech dále počítají jiné typy příznaků. Jedním z osvědčených typů příznaků jsou kepstra. Reálné kepstrum je definováno podle vztahu 2.2.2.

$$C = \text{IDFT}\left\{\ln|\text{DFT}(x)|\right\} \tag{2.2.2}$$

Blokové schéma výpočtu mel-kepstra ukazuje obrázek 2.2.1.

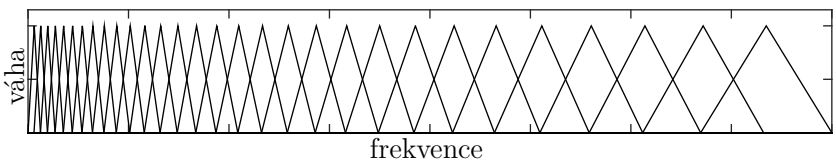


Obrázek 2.2.1 Blokové schéma výpočtu mel-kepstra pro jeden frame.

V případě melovských kepstrálních koeficientů se na spektrogram nejprve aplikuje banka filtrů k získání výkonů v pásmech, mel-spektra. Tato banka filtrů se snaží kompenzovat nelineární vnímání frekvencí lidským sluchem. Proto jsou filtry rovnoměrně rozmístěné v takzvané melovské frekvenční škále (rovnice 2.2.3), která má logaritmické měřítko.

$$f_m = 2595 \log_{10}\left(1 + \frac{f}{700}\right) \tag{2.2.3}$$

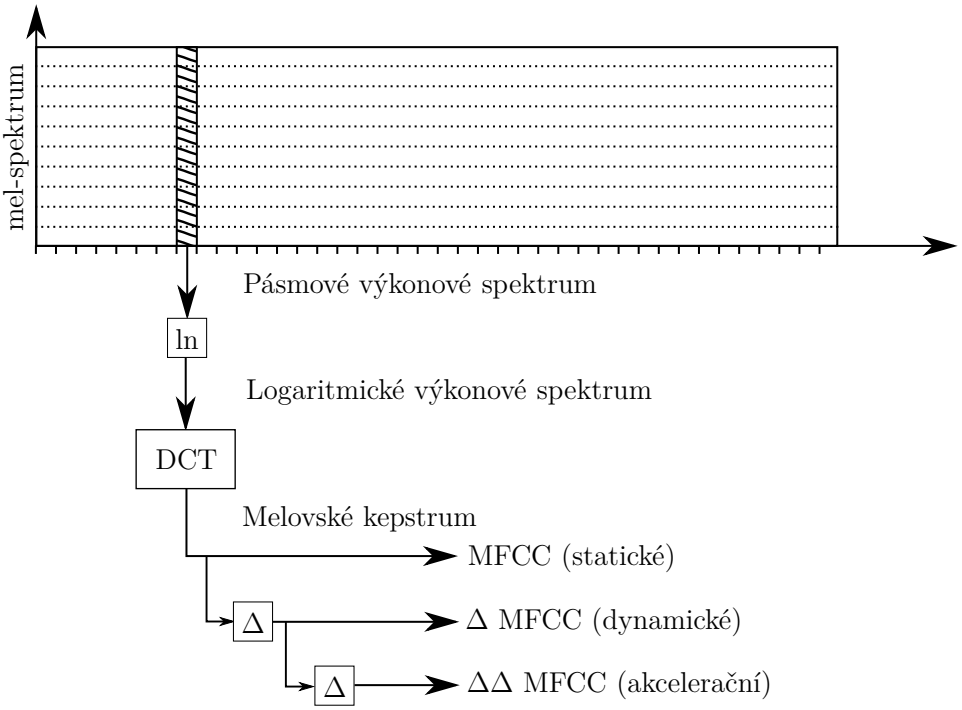
Podobu filtrů pro výpočet mel-spektra po zpětné transformaci do lineárního frekvenčního měřítka ukazuje obrázek 2.2.2.



Obrázek 2.2.2 Banka filtrů pro výpočet mel-spektra.

Tato operace zredukuje nadbytečnou informaci krátkodobého spektra. To obsahuje pro jeden segment typicky stovky příznaků, mel-spektrum je redukuje do řádu desítek, podle počtu pásem. Následně se logaritmuje a provádí DCT, na kterou se díky symetričnosti a reálnosti mel-spektra redukuje IDFT. Schéma výpočtu MFCC s delta a delta-delta koeficienty z mel-spektra ukazuje obrázek 2.2.3.

MFCC bez dynamických koeficientů se v implementacích označují jako MFCC_0, kde 0 značí použití nultého kepstra v příznakovém vektoru. Nulté kepstrum u MFCC odpovídá energii signálu. Pokud kepstrální příznaky obsahují i delta a delta-delta koeficienty, označují se jako MFCC_0_d_a.



Obrázek 2.2.3 Ilustrativní příklad výpočtu MFCC s delta a delta-delta parametry.

2.2.2 Perceptivní lineární prediktivní analýza

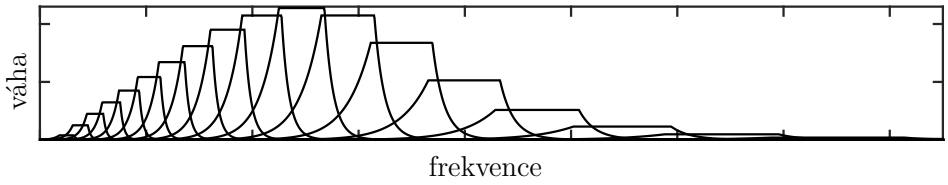
Další často používané příznaky jsou PLP kepstrální koeficienty. Ty rozšiřují metodu lineární prediktivní analýzy, která aproximuje spektrum krátkodobého segmentu řeči. PLP se navíc oproti LPC analýze, stejně jako MFCC snaží podchytit i nelineární vlastnosti lidského sluchu, jako je vnímání intenzity a frekvencí, nebo maskování frekvencí, kdy jeden zvuk zvyšuje práh slyšitelnosti jiného.

PLP vychází stejně jako MFCC ze spektra. Na spektrum se aplikuje banka filtrů, která šířkou jednotlivých filtrů kompenzuje nelineární vnímání frekvencí. Banka filtrů PLP používá nelineární frekvenční měřítko dané rovnicí 2.2.4. Tvar jednoho filtru je pak dán v nelineární frekvenční škále vztahem 2.2.5.

$$\Omega(f) = 6 \ln \left(\frac{f}{600} + \sqrt{\left(\frac{f}{600} \right)^2 + 1} \right) \tag{2.2.4}$$

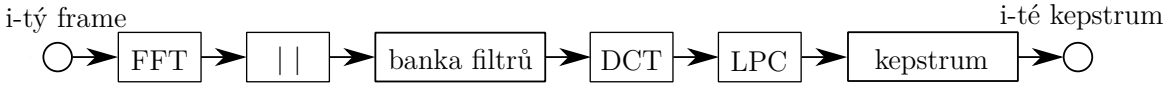
$$\Psi(z) = \begin{cases} 0 & \text{pro} \quad z < -2.5 \\ 10^{z+0.5} & \text{pro} \quad -2.5 \leq z \leq -0.5 \\ 1 & \text{pro} \quad -0.5 < z < 0.5 \\ 10^{-2.5(z-0.5)} & \text{pro} \quad 0.5 \leq z \leq 1.3 \\ 0 & \text{pro} \quad z > 1.3 \end{cases} \tag{2.2.5}$$

Lidský sluch vnímá různé frekvence se stejnou intenzitou jako různě hlasité. PLP tento jev zachycuje různým zesílením filtrů podle křivek stejné hlasitosti. Výslednou banku filtrů pro výpočet PLP ukazuje obrázek 2.2.4.



Obrázek 2.2.4 Banka filtrů pro výpočet PLP.

Spektrum s aplikovanou bankou filtrů je v dalším kroku PLP analýzy aproximováno metodami lineární prediktivní analýzy. Nalezená aproximace se dále transformuje na kepstrální koeficienty. Blokové schéma ukazuje obrázek 2.2.5.



Obrázek 2.2.5 Blokové schéma výpočtu PLP-kepstra pro i-tý frame.

2.2.3 Dynamické koeficienty

Melovské kepstrální koeficienty ani PLP koeficienty neobsahují informaci o kontextu signálu. Příznakový vektor je vypočítán s použitím hodnot pro jediný segment signálu. Kontextovou informaci lze přidat rozšířením příznakového vektoru o dynamické delta a delta-delta koeficienty. Tyto dynamické koeficienty vyjadřují dynamiku časové změny vektoru příznaků a určují se lineární regresí $2L + 1$ po sobě jdoucích mikrosegmentů řečového signálu. Výpočet delta a delta-delta koeficientů ukazují rovnice 2.2.6 a 2.2.7 [1].

$$[\Delta c_m(j)]_n = \frac{\sum_{\kappa=-L}^{L} \kappa [c_m(j)]_{n+\kappa}}{\sum_{\kappa=-L}^{L} \kappa^2} \tag{2.2.6}$$

$$[\Delta^2 c_m(j)]_n = \frac{\sum_{\kappa=-L}^{L} \kappa [\Delta c_m(j)]_{n+\kappa}}{\sum_{\kappa=-L}^{L} \kappa^2} \tag{2.2.7}$$

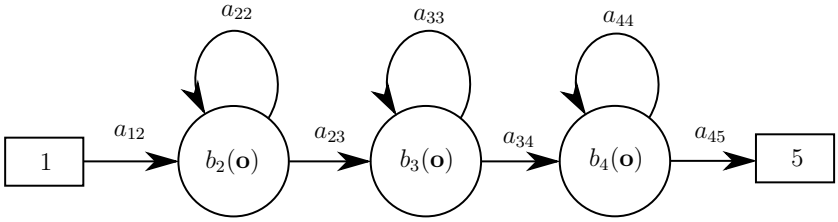
Každý příznakový vektor je pak rozšířen o tyto další příznaky. V ojedinělých případech je možné použít dynamické koeficienty vyšších řádů, ale tato praktika není rozšířená.

2.3 Akustické modelování

Akustický model pracuje s příznakovými vektory O. Jeho úkolem je poskytnout odhad pravděpodobnosti $P(O|W)$. Tento odhad musí být schopen flexibilně reagovat na různé podmínky provozu. Jedná se jednak o parametrizací ne zcela odfiltrované zarušení (zvuky na pozadí, odlišná kvalita záznamu, ...), tak i o sekvenci příznakových vektorů, které se v trénovací množině vůbec nevyskytovaly (rozdílní mluvčí, tempo řeči, artikulace, ...). Další požadavky na akustický model jsou přesnost v odlišení výslovnostně podobných, ale významově zcela odlišných slov a rychlost poskytnutí odhadu, která je důležitá zvlášť při použití rozpoznávače v real-time aplikacích. K akustickému modelování se nejčastěji používají systémy na bázi skrytých Markovových modelů (HMM), které se ukázaly jako velmi vhodné.

2.3.1 Skryté Markovovy modely

HMM je dnes nejrozšířenější metoda pro akustické modelování. Pracuje se statistickými modely rozpoznávaných jednotek a myšlenkou, že se příznakové vektory náležící jedné rozpoznávané jednotce od sebe příliš nevzdalují (jejich vzájemná vzdálenost je malá). HMM je konečný stavový automat, jehož stavy jsou propojeny do přímé posloupnosti. Pětistavový dopředný model se třemi emitujícími stavy a přeskoky obstav ukazuje obrázek 2.3.1.



Obrázek 2.3.1 Schéma levopravého pětistavového modelu

Takovýto HMM je popsán maticí přechodů a funkcemi b_i, které náleží každému emitujícímu stavu. Matice přechodů udává pravděpodobnosti přechodu mezi jednotlivými stavy a defakto udává strukturu HMM. Vzhledem k tomu, že celková pravděpodobnost přechodu na jednom stavu musí být jedna, nebo nula, součet na řádku matice přechodu musí být také jedna, nebo nula. Příklad matice přechodu pro HMM z obrázku 2.3.1 je v rovnici 2.3.1.

$$\mathbf{A} = \begin{bmatrix} 0 & 0.8 & 0 & 0 & 0 \\ 0 & 0.6 & 0.3 & 0 & 0 \\ 0 & 0 & 0.55 & 0.40 & 0 \\ 0 & 0 & 0 & 0.7 & 0.3 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.3.1}$$

Pravděpodobnostní funkce b_i vyjadřují hustotu pravděpodobnosti jednotlivých shluků reprezentujících podobné akustické podmínky v prostoru příznakových vektorů O. Jsou modelovány nejčastěji gaussovskými funkcemi, jejichž výhodou je možnost reprezentace středními hodnotami a rozptyly, které lze získat z trénovacích dat. N-rozměrná gaussovská funkce je dána předpisem 2.3.2.

$$b_i(\mathbf{o}) = \frac{1}{\sqrt{(2\pi)^n |\mathbf{C}_i|}} \cdot \exp\left(-\frac{1}{2}\sum_{k=1}^{n}(\mathbf{o}-\boldsymbol{\mu}_i)^\intercal \mathbf{C}_i^{-1}(\mathbf{o}-\boldsymbol{\mu}_i)\right) \tag{2.3.2}$$

Pokud jsou data dekorelována, kovarianční matice $\mathbf{C}$ je pak diagonální a vztah lze zjednodušit tak, jak ukazuje rovnice 2.3.3.

$$b_i(\mathbf{o}) = \frac{1}{\sqrt{(2\pi)^n \prod_{k=1}^{n} \sigma_{ik}^2}} \cdot \exp\left(-\frac{1}{2}\sum_{k=1}^{n}\frac{(o_k-\mu_{ik})^2}{\sigma_{ik}^2}\right) \tag{2.3.3}$$

V případě, že rozložení trénovacích dat nelze uspokojivě modelovat s použitím pouze jedné gaussovské funkce, používají se tzv. směsi. Funkce b_i je pak lineární kombinací několika gaussovských funkcí. Což ukazuje vztah 2.3.4.

$$b_i(\mathbf{o}) = \sum_{m=1}^{M} c_{im} \cdot \frac{1}{\sqrt{(2\pi)^n \prod_{k=1}^{n} \sigma_{imk}^2}} \cdot \exp\left(-\frac{1}{2}\sum_{k=1}^{n}\frac{(o_k-\mu_{imk})^2}{\sigma_{imk}^2}\right) \tag{2.3.4}$$

Pravděpodobnostní funkce b_i je možné nahradit pravděpodobností vypočtenou jinými metodami. Pro výpočet pravděpodobností je možné použít i neuronovou síť. Pak se hovoří o DNN-HMM.

Pro správnou funkci HMM je potřeba ho natrénovat. Při tomto procesu se nastavují pravděpodobnosti v matici přechodů a v případě použití gaussovských funkcí b_i i jejich parametry. V případě použití DNN-HMM je nutné natrénovat umělou neuronovou síť.

2.3.2 Slučování parametrů

Při výpočtu příznaků se často používají různé metody, jak do příznakového vektoru přidat kontext. Tento kontext má zásadní vliv na úspěšnost rozpoznávání. Je vhodné vložit nějakou formu kontextu i do akustického modelu. Tento kontext je možné vytvořit vhodnou volbou rozpoznávané jednotky. Nejzákladnější rozpoznávanou jednotkou je jeden fon (monofon). Monofony poskytují jen základní informaci o kontextu zřetězením jejich HMM do slov s pomocí přepisu. Toto nemusí být dostatečné, neboť akustické vlastnosti fonému jsou ovlivněny okolními fonémy a to i mezi slovy. Z tohoto důvodu se jako základní elementy z nichž se vytváří skryté Markovovy modely, uvažují fonémy s kontextem několika fonémů před i za.

Typicky se pracuje s monofony, difony a trifony. Přepisy pro vytváření modelů pak mohou vypadat i takto:

- **promluva:** pěkný den
- **monofony:** sil p j e k n yy sp d e n sil
- **difony:** sil sil-p p-j j-e e-k k-n n-yy sp yy-d d-e e-n sil
- **trifony:** sil sil-p+j p-j+e j-e+k e-k+n k-n+yy n-yy+d sp yy-d+e d-e+n e-n+sil sil

Samozřejmě je možné vygenerovat HMM pro libovolný -n/+m kontext. Z technického hlediska lze -n/+m fony vygenerovat z monofonů a není tedy třeba distribuovat data s jiným než monofonovým přepisem. Každý -n/+m fon je jeden skrytý Markovův model. Z toho je jasné, že počet těchto modelů s větším požadovaným kontextem prudce roste a zároveň pro každý z těchto modelů je potřeba mít k dispozici dostatečná trénovací data. V praxi se pro rozpoznávání používá trifonový model.

Slučování stavů

Kontextově závislé subslovní elementy jako jsou trifóny lépe modelují promluvy, nicméně větší počet modelů vyžaduje větší trénovací množinu. Toho se v praxi obtížně dosahuje. Řešením je sdílení stavů mezi několika modely. Tato metoda využívá myšlenky, že některé stavy jsou si akusticky dostatečně podobné. Spojením těchto stavů do jednoho pak vede na vznik zobecněného trifonu, redukci trénovací množiny a tím pádem i menší nároky na trénovací data.

2.4 Jazykové modelování

Úkolem jazykového modelu je poskytovat odhad apriorní pravděpodobnosti $P(W)$ pro libovolnou posloupnost slov W. Každý jazyk má své zákonitosti, slova ve slovníku a jejich

výběr a řazení do celků. Slova navíc mohou mít více výslovnostních variant (např. nashledanou/naschledanou). Jazykový model se snaží tyto zákonitosti najít a modelovat, čímž stanovuje pro posloupnosti slov W určitá omezení. Tato omezení mohou být deterministická, kdy model striktně nedovoluje některé posloupnosti, nebo pravděpodobnostní, kdy jsou tyto posloupnosti pouze méně pravděpodobné. Častým deterministickým omezením je přípustnost pouze slovníkové výslovnosti.

V úloze rozpoznávání spojité řeči musí jazykový model předpokládat libovolnou sekvenci slov a žádná sekvence by neměla mít nulovou pravděpodobnost, abychom ji nevyloučili z rozpoznávání. Takovému modelu se říká stochastický jazykový model. Výpočet pravděpodobnosti posloupnosti W o K slovech je uveden v rovnici 2.4.1.

$$\begin{aligned} P(W) = P(w_1^K) = P(w_1 w_2 w_3 \dots w_K) = \\ = P(w_1)P(w_2|w_1)P(w_3|w_1 w_2)\dots P(w_K|w_1 w_2 \dots w_{K-1}) = \\ = P(w_1)P(w_2|w_1^1)P(w_3|w_1^2)\dots P(w_K|w_1^{K-1}) = \prod_{i=1}^{K} P(w_i|w_1^{i-1}) \end{aligned} \tag{2.4.1}$$

Z rovnice 2.4.1 je vidět, že pravděpodobnost výskytu slova je dána pouze jeho historií. To je výhodné, pokud má rozpoznávač pracovat již v průběhu promlouvání.

Pro konstrukci jazykového modelu je třeba znát pravděpodobnosti všech posloupností slov libovolné délky. Taková data je téměř nemožné získat a proto se uvažuje jen n předcházejících slov. Takovým modelům se říká n-gramové. Výpočet pravděpodobnosti uvedené v rovnici 2.4.1 je aproximován, jak ukazuje rovnice 2.4.2.

$$P(w_1^k) \approx \prod_{i=1}^{k} P(w_i|w_{i-n+1}^{i-1}) \tag{2.4.2}$$

V praxi se nejčastěji používají bigramy, nebo trigramy. Vyhodnocení pravděpodobností těchto n-gramů spočívá v podstatě ve zjišťování relativní četnosti této posloupnosti v rozsáhlých textových korpusech daného jazyka. Množství trénovacích dat pro jazykové modely je hlavní problém při vytváření jazykových modelů zvlášť v rozpoznávačích s rozsáhlým slovníkem.

2.5 Dekódování & WFST

Dekódování je proces hledání posloupnosti slov W z posloupnosti příznakových vektorů O. Kromě příznakových vektorů, jsou při dekódování k dispozici pravděpodobnosti $P(O|W)$ vyčíslované z akustického modelu a pravděpodobnosti $P(W)$ počítané jazykovým modelem. Úkolem je najít posloupnost slov $\hat{W}$ takovou, která maximalizuje součin $P(O|W)$ a $P(W)$, tak jak ukazuje rovnice 2.5.1.

$$\hat{W} = \arg\max_{W} P(W)P(O|W) \tag{2.5.1}$$

Přímý dekódovací algoritmus, který by prohledal všechny posloupnosti slov je mimo jakékoli výpočetní možnosti i pokud nemusí probíhat v reálném čase. Je proto nutné hledat takové metody řešení, které tuto úlohu dokáží vyřešit uspokojivě v řádu několika málo milisekund.

Aby bylo možné dosáhnout požadované rychlosti výpočtu, provádí se při prohledávání prořezávání grafu, kdy se uvažují jen pravděpodobné návaznosti slov. Někdy je vhodné místo jedné nejpravděpodobnější sekvence slov hledat několik nejpravděpodobnějších. To přináší výhodu v možnosti omezit prohledávaný prostor pouze na okolí těchto promluv.

Jazykový model popsaný regulární gramatikou (n-gramový model), lze vyjádřit orientovaným grafem, kde uzly jsou slova a hrany určují možnost rozšířit aktuální posloupnost o další slovo. Hrany jsou ohodnoceny číslem, udávajícím pravděpodobnost, že posloupnost bude rozšířena tímto novým slovem. Tento graf je schopen generovat jakoukoli posloupnost slov s pravděpodobností danou jazykovým modelem. Po vygenerování cesty grafem a tedy posloupnosti slov je pravděpodobnost této posloupnosti určena jako součin pravděpodobností na hranách, kterými tato cesta prochází.

Akustický model je zpravidla konstruován pomocí HMM, jedná se tedy o konečný stavový automat. Ten lze vyjádřit pomocí orientovaného grafu.

Spojením akustického a jazykového modelu, lze vybudovat hierarchický model promluvy pro každou posloupnost slov W. Každý takový model je tvořen posloupností skrytých Markovových modelů slov, které jsou složeny z HMM menších základních jednotek (monofonů, trifonů). Každý model promluvy má tedy podobu pravděpodobnostního konečného automatu a tedy orientovaného grafu. Dekódování je úloha hledání cesty tímto grafem s největším ohodnocením.

WFST (Weighted Finite-State Transducer) je konečný stavový automat vytvořený z rozpoznávací sítě její optimalizací. Jednotlivé části rozpoznávače jsou stavové automaty. Tyto stavové automaty lze kombinovat a slučovat. Z akustického a jazykového modelu, výslovnostního slovníku a dalších informací dostupných o dané úloze, vznikne stavový automat, který lze vyjádřit orientovaným grafem, pak hovoříme o rozpoznávací síti HCLG. Ta je však silně neoptimalizovaná, jednak redundantními cestami, jež se liší jen v ohodnocení a ze kterých stačí uvažovat jen tu s nejlepším ohodnocením a jednak shodnými částmi cest. Pomocí různých dostupných nástrojů dochází k minimalizaci počtů hran a uzlů, což vede k optimalizované síti a tedy rychlejšímu prohledávání a menší paměťové náročnosti.

2.6 LVCSR

Lze definovat různé úlohy rozpoznávání řeči.

Rozpoznávání fonémů, kde základní rozpoznávaná jednotka je foném. Využívá se při rozpoznávání řečníka, rozpoznávání jazyka promluvy a dalších úlohách. Je rychlé a jednoduché díky malé velikosti gramatiky.

Rozpoznávání s malým slovníkem je vhodné pro rozpoznávání izolovaných slov, příkazů, frází. Naproti tomu rozpoznávače s velkým slovníkem nalézají využití při rozpoznávání spojité, přirozené řeči.

LVCSR (Large Vocabulary Continuous Speech Recognition) je typ úlohy s velkým slovníkem. Používá jako základní rozpoznávanou jednotku slova a větné celky modelované pomocí jazykového modelu. Slova jsou modelována pomocí výslovnostního slovníku, akustického modelu a na základní úrovni skrytými Markovovými modely subslovních elementů jako jsou monofony, trifony atd. Výstupem je rozpoznaná promluva na úrovni slov. LVCSR je sestaven na rozsáhlém slovníku obsahujícím i desítky tisíc slov a je navržen a implementován tak, aby dokázal rozpoznat i spojitou řeč. Dnešní LVCSR dokáží operovat i v reálném čase.

3 Umělé neuronové sítě & TANDEM

Již v roce 1993 bylo dosaženo dílčích úspěchů při použití Multi-Layer Perceptronu (MLP) při odhadu pravděpodobností v HMM modelu [2]. V té době nebyly k dispozici dostatečně efektivní algoritmy, ani výkonný hardware a tak převládla metoda využívající GMM. Návrat k použití umělých neuronových sítí při odhadu pravděpodobností stavů skrytých Markovových modelů přišel až v posledních letech.

Umělé neuronové sítě (Artificial Neural Network - ANN) napodobují princip fungování biologických neuronových sítí. Jejich základním prvkem je neuron. Tyto neurony jsou uspořádány do vzájemně propojených vrstev. ANN zpravidla obsahují několik vrstev a v každé může být i několik tisíc neuronů. Trénování neuronové sítě je výpočetně velice náročná úloha, a proto k jejich rozšíření došlo až s nárůstem výpočetního výkonu počítačů.

3.1 Neuron

Základem neuronové sítě je neuron. Nejrozšířenější model neuronu je tzv. formální neuron, který má několik vstupů a jeden výstup. Skládá se z obvodové a aktivační funkce. Obvodová funkce je dána vztahem 3.1.1, kde x_i jsou vstupy w_i jsou váhy a Θ je aktivační práh, výsledkem je aktivační potenciál neuronu $\bar{y}$.

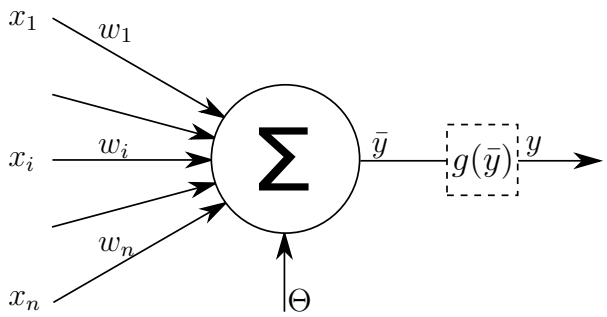
$$\bar{y} = \sum_{i=1}^{n} x_i w_i + \Theta \tag{3.1.1}$$

Tento aktivační potenciál pak aktivační funkce $g(\bar{y})$ transformuje na výstup neuronu. Aktivační funkce je často *sigmoida*, *tanh*, *skoková funkce* nebo funkce *softmax*, v případě že výstupem neuronu má být pravděpodobnostní příznak. Schématický nákres formálního neuronu ukazuje obrázek 3.1.1.

Existují i další typy neuronů vhodné pro různé aplikace, nicméně v rozpoznávání řeči je nejpoužívanější právě formální neuron.

3.2 Typy sítí

Spojením mnoha neuronů do sítě vzniká neuronová síť, jež je schopna řešit i obtížné úlohy nelineárního mapování. Existuje mnoho typů neuronových sítí s různými počty



Obrázek 3.1.1 Nákres formálního neuronu

vrstev a neuronů v nich. ANN se však vždy skládá ze vstupní vrstvy, která provádí lineární transformaci příznaků, jedné, nebo několika skrytých vrstev a výstupní vrstvy, která upravuje hodnoty výstupního vektoru podle do požadované podoby. V případě pravděpodobnostního výstupu, například pokud je potřeba klasifikovat vstupní vektor k rozpoznávané třídě, se výstupy normují pomocí funkce *softmax* tak, aby jejich součet byl roven jedné.

Neuronovou síť je vždy nutné nejprve natrénovat, kdy se pomocí učení s učitelem nastaví vstupní váhy jednotlivých neuronů ve všech vrstvách.

3.2.1 Multi-Layer Perceptron

MLP je typ neuronové sítě, který se skládá ze tří vrstev, z nichž jedna je skrytá. Neurony v každé vrstvě mají nelineární aktivační funkci a každý neuron vrstvy je spojen se všemi neurony následující vrstvy. Výstupní vrstva je zpravidla normovaná *softmax* funkcí, která součet výstupů sítě normuje na jedničku a výstupy jsou pak pravděpodobnostního charakteru.

Trénování MLP je učení s učitelem a probíhá pomocí algoritmu zpětné propagace chyby, který je zobecněným algoritmem nejmenších čtverců. Tento algoritmus je relativně jednoduchý a rychlý oproti jiným metodám používaným ve složitějších sítích.

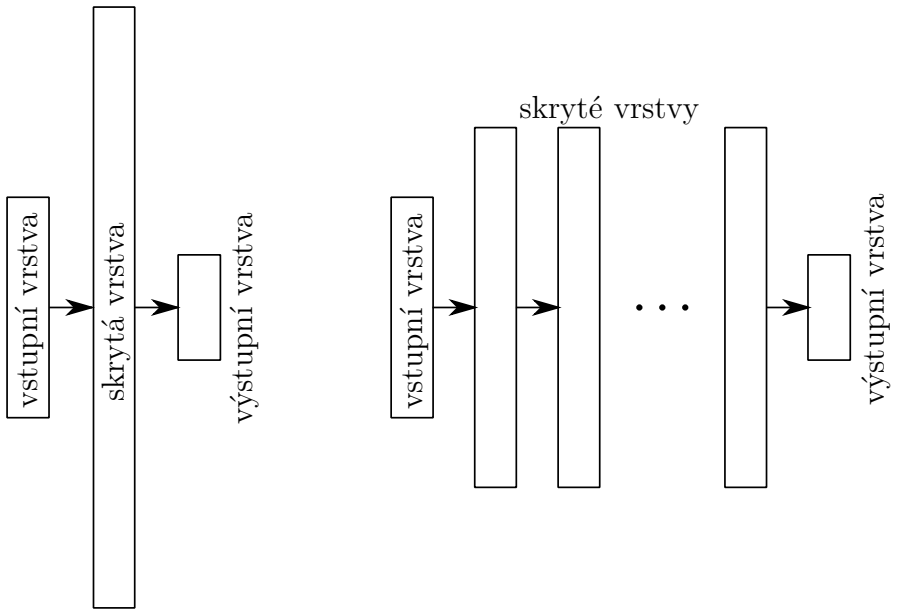
MLP byl s nárůstem výkonu výpočetní techniky vytlačen a zcela nahrazen složitějšími sítěmi s více vrstvami, které poskytují daleko lepší výsledky.

Schématický nákres takové sítě ukazuje levá část obrázku 3.2.1.

3.2.2 Deep Neural Network

Ukazuje se, že umělé neuronové sítě, jejichž rozvoj je úzce svázán s nárůstem výpočetního výkonu počítačů, mohou řešit problémy daleko úspěšněji než jiné metody. V takovém případě jsou však jednoduché třívrstvé sítě typu MLP nedostačující. DNN je pokročilý typ neuronové sítě, který se vyznačuje tím, že má více než jednu skrytou vrstvu a tyto vrstvy jsou předtrénované pomocí deep learning algoritmu, jako je například Restrictive

Boltzman Machine (RBM). Pokud je síť místo toho inicializovaná jen náhodnými hodnotami, jedná se spíše o obecnější variantu MLP. Každá skrytá vrstva může mít různý počet neuronů. Většinou však mají všechny skryté vrstvy stejný počet neuronů. S nárůstem počtu neuronů i vrstev sítě prudce roste výpočetní náročnost trénování a provozu takové sítě. Tento problém částečně odklání bouřlivý pokrok ve vývoji hardware, zvláště pak grafických karet, jejichž paralelní zpracování dat je pro použití v ANN ideální. Složitější struktura sítí však také zvyšuje nároky na množství trénovacích dat. Správně natrénovaná DNN dostatečné velikosti pak může představovat velmi kvalitní nástroj pro klasifikaci. Schématický nákres takové sítě ukazuje pravá část obrázku 3.2.1.



Obrázek 3.2.1 Schematický nákres MLP sítě (vlevo) a DNN (vpravo).

Vedle tradičnějších DNN, které mají v každé skryté vrstvě typicky stejný počet neuronů, existují i tzv. Bottleneck sítě, kde má jedna skrytá vrstva výrazně méně neuronů. V tomto místě dochází k redukci stavového prostoru ANN a tím ke zhuštění informace. Bottleneck ANN se trénuje stejným způsobem jako DNN, ale po jejím natrénování se jako výstup používají výstupní hodnoty z bottleneck vrstvy. Tyto parametry obsahují zhuštěnou informaci s jen malou redundancí a jsou vhodné pro použití v dalším rozpoznávání. Příznaky z bottleneck vrstvy nejsou transformovány funkcí *softmax* a nemají charakter pravděpodobností.

3.3 Trénování ANN

Trénování umělých neuronových sítí spočívá v nastavení vah jednotlivých vstupů a prahu aktivační funkce jednotlivých neuronů v celé síti tak, aby pro vektor vstupů bylo dosa-

ženo žádaného výstupu. Jedná se tedy o učení s učitelem. To může být problematické, neboť je potřeba mít k trénovacím datům informaci o příslušnosti ke klasifikované třídě a to na úrovni jednotlivých příznakových vektorů.

Trénování i jednoduchých umělých neuronových sítí bylo dlouho velmi problematické. Díky zvýšení výpočetního výkonu aktuálních počítačů a metodách algoritmů strojového učení, je možné v dnešní době efektivně trénovat i rozsáhlé sítě v přijatelném časovém horizontu.

3.3.1 Backpropagation

Algoritmus zpětného šíření chyby je základní metoda používaná pro trénování neuronových sítí. Tato metoda vyžaduje znalost správného výstupu pro každý vstupní vektor, kvůli nutnosti spočítat hodnotu ztrátové funkce a aktivační funkce neuronu musí mít derivaci.

Nejprve přivedeme vstupní příznakový vektor a necháme síť vypočítat výstup. Z této hodnoty a očekávaného výstupu vypočteme chybu na neuronu a celkovou chybu výstupní vrstvy. Cílem trénování je minimalizovat tuto hodnotu. S pomocí gradientu klesání lze zjistit změnu každé váhy neuronu. Tato hodnota se před přičtením násobí hodnotou *learning rate*, tedy hodnotou která určuje rychlost trénování. Pokud je zvolena příliš malá, trénování bude pomalé, pokud bude příliš velká, hodnoty se mohou rozkmitat a jen těžko pak budou konvergovat k optimálnímu řešení.

Nastavování vah probíhá od výstupní vrstvy směrem ke vstupní. Vyčíslení derivace pro více skrytých vrstev rychle nabývá na složitosti a prakticky znemožňuje použití tohoto algoritmu pro vícevrstvé sítě.

3.3.2 Stochastické gradientní klesání

Stochastické gradientní klesání (Stochastic gradient descent - SGD) je aproximace metody gradientního klesání. Zatímco metoda gradientního klesání počítá výslednou hodnotu analyticky, což komplikuje řešení pro hluboké sítě, SGD pracuje iterativně a postupně konverguje k výsledku. Aby při konvergenci nedošlo k zacyklení, trénovací množina se po každém projití algoritmu náhodně zamíchá. Tento přístup umožňuje SGD odhadovat váhy neuronů při trénování ANN i s více než jednou skrytou vrstvou a tím umožňuje trénování hlubokých neuronových sítí.

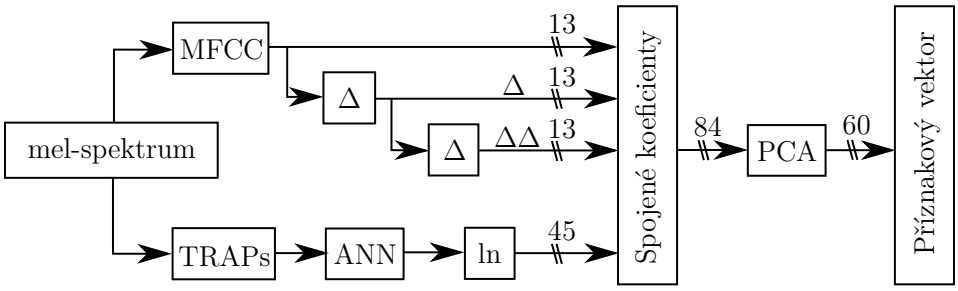
3.4 TANDEM architektura

Každý typ příznaků nese trochu jiný typ informace a přináší v rozpoznávání výhody i nevýhody. Nabízí se spojit příznakové vektory do jednoho a vytvořit tak robustnější

příznakový vektor, který potlačuje nevýhody a využívá výhody výchozích příznakových vektorů.

TANDEM architektura je označení pro metodu, jež spojuje zpravidla dva typy příznakových vektorů. Jedním z nich jsou aposteriorní odhady fonémů ve frame, získané například pomocí umělých neuronových sítí. Druhý příznakový vektor je tvořen kepstrálními příznaky, ať už PLP, nebo častěji MFCC, s možným rozšířením o dynamické koeficienty. Pro vstup do ANN za účelem získání aposteriorních pravděpodobností mohou sloužit kepstrální koeficienty i s dynamickými parametry, případně se používají TRAP příznaky, z nichž TANDEM architektura také historicky vychází.

Schématický náčrt 3.4.1 ilustruje strukturu příznakového vektoru TANDEM architektury při použití příznakového vektoru TRAP na vstupu ANN. Z mel-spektra jsou vypočteny Melovské kepstrální příznaky a jejich dynamické koeficienty. Osvědčená nastavení pracují s třinácti prvními kepstry MFCC a stejným počtem pro delta a delta-delta koeficienty. Tyto hodnoty dohromady vytváří první část spojeného příznakového vektoru. Druhá část vychází z TRAPs příznaků vypočtených ze stejného mel-spektra, jako první část příznakového vektoru. TRAP příznaky, které mají běžně i několik stovek příznaků se pomocí umělé neuronové sítě namapují na prostor s menší dimenzí. Díky funkci *softmax* na výstupu sítě se jedná o pravděpodobnostní příznaky. Tyto příznaky nemají gaussovské rozdělení potřebné k dalšímu zpracování, proto je nutné je zlogaritmovat. Spojením s paralelně vypočtenými MFCC příznaky vznikne spojený příznakový vektor. Ten je vzhledem k rozdílné povaze vstupních příznaků málo dekorelovaný a obsahuje stále nějakou nadbytečnou informaci. Z tohoto důvodu se provádí PCA. Tato transformace data dekoreluje a zhustí na prvních dimenzích příznakového vektoru. Díky tomuto lze ještě snížit délku příznakového vektoru. Toto snížení však nesmí být moc razantní, aby z příznakového vektoru nevymizelo příliš mnoho užitečné informace.

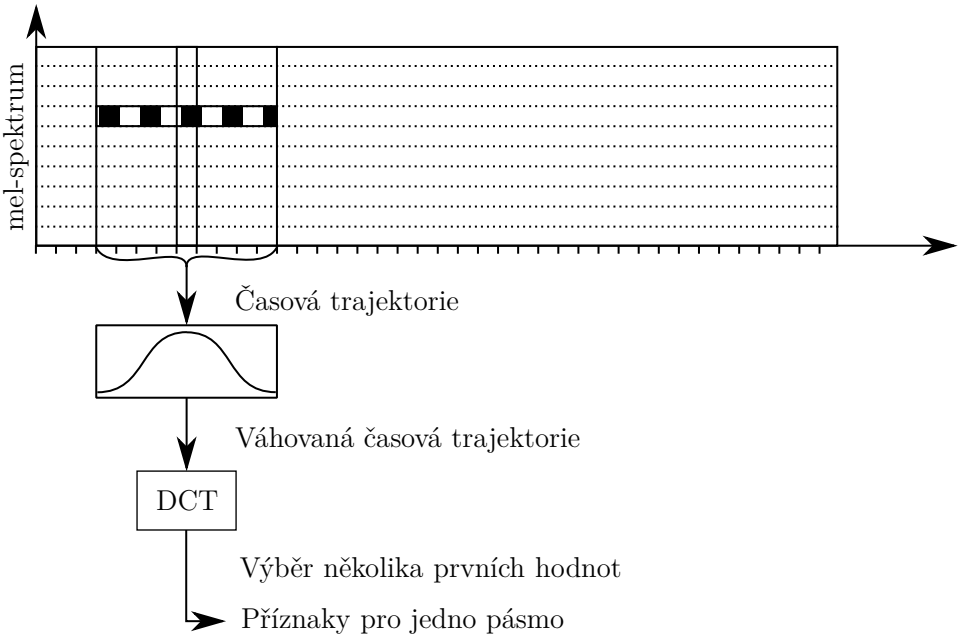


Obrázek 3.4.1 Znázornění výpočtu příznakového vektoru TANDEM architektury.

3.4.1 TRAP příznaky

Statické příznaky neposkytují dostatečnou informaci o kontextu segmentu. V případě, že chceme uvažovat i delší kontext, lze použít komprimované Trajektorie mel-spektra

(TRAP). Tyto příznaky se počítají pro každé pásmo mel-spektra zvlášť. Kontext tvoří několik sousedních segmentů, jež tvoří tzv. trajektorii, jejíž délka může být až jedna vteřina. Na váhované trajektorii se provede DCT, ze které se vybere několik prvních příznaků s největší informací. Tento postup se provede pro všechna pásma a výsledné hodnoty se spojí do jednoho příznakového vektoru. Obvyklé schéma výpočtu TRAP z mel-spektra ukazuje obrázek 3.4.2.



Obrázek 3.4.2 Výpočet TRAP příznaků pro jedno pásmo z mel-spektra

Příznakový vektor TRAP může obsahovat i několik stovek prvků a obsahuje množství nadbytečné a málo dekorelované informace. Jedním ze způsobů omezení dimenze příznakového vektoru je nelineární mapování na prostor s menší dimenzí. K tomuto účelu se používají umělé neuronové sítě.

3.4.2 Příznakové vektory TANDEM architektury

Oba typy příznaků mají ze své podstaty velmi odlišnou formu. Kepstrální příznaky obsahují velmi zhuštěnou, dekorelovanou informaci ve formě vhodné pro další zpracování. Naproti tomu aposteriorní pravděpodobnosti nemají formu vhodnou pro další strojové zpracování, nejsou dekorelované a nemají gaussovské rozdělení. Pravděpodobnosti však dokáže bez větších problémů interpretovat i člověk.

Aby bylo možné tyto dva příznakové vektory spojit do jednoho, musí mít aposteriorní pravděpodobnosti gaussovské rozdělení. Transformace zajišťující tento předpoklad je prosté zlogaritmování. Po této operaci jsou pravděpodobnostní připraveny ke spojení s kepstrálními příznaky.

Pro trénování GMM-HMM rozpoznávače je vhodné mít dekorelovaná data. Tím zajistíme, že kovarianční matice bude diagonální, nebo skoro diagonální. Kepstrální příznaky jsou dekorelované díky použití DCT, nicméně aposteriorní příznaky žádnou dekorelací neprošly. Existuje několik metod, jak data dekorelovat. Stejně tak způsobů, jak tyto metody aplikovat. Je možné dekorelovat logaritmované aposteriorní příznaky, spojit je s kepstrálními příznaky. Případně nový příznakový vektor znovu dekorelovat, to může mít za následek menší dimenzi výsledného příznakového vektoru. Také je možné, spojit logaritmované aposteriorní pravděpodobnosti s kepstrálními příznaky a dekorelovat až spojený příznakový vektor.

3.4.3 Metoda hlavních komponent

Jedním ze způsobů, jak dekorelovat příznaky je metoda hlavních komponent (Principal Component Analysis - PCA). Jedná se o transformaci, jež hledá směry největšího rozptylu vstupních dat. To znamená, že je přizpůsobena přímo datům pro něž byla vypočtena, na rozdíl od například DCT. Tato transformace také koncentruje informaci na nízkých dimenzích a může tedy sloužit i ke kompresi dat. Transformace je dána vztahem 3.4.1.

$$\mathbf{Y} = \mathbf{V} \cdot \mathbf{X} \tag{3.4.1}$$

Vektor $\mathbf{Y}$ je výstupním vektorem s redukovanou dimenzí m. Vektor $\mathbf{X}$ je vstupním vektorem s dimenzí n. Matice $\mathbf{V}$ je transformační matice, v řádcích obsahující vlastní vektory o dimenzi shodné se vstupním vektorem. Těchto vektorů je n. Snížení dimenze se provádí odebráním vlastních vektorů v matici $\mathbf{V}$ na počet požadovaných výstupních dimenzí m.

Data pro výpočet PCA musí být zarovnána na střed přes střední hodnotu na celém trénovacím setu. Z kovarianční matice vypočtené ze zarovnaných dat pomocí rovnice 3.4.2, se vypočtou vlastní čísla a vektory. Vlastní vektory se seřadí podle velikosti vlastních čísel a vybráním m prvních dochází k redukci výstupního příznakového vektoru.

$$\mathbf{C}_x = \frac{1}{k} \sum_{i=1}^{k} \tilde{\mathbf{x}}_i \tilde{\mathbf{x}}_i^{\intercal} \tag{3.4.2}$$

Kde $\mathbf{C}_x$ je kovarianční matice, k je množství trénovacích dat a $\tilde{\mathbf{x}}_\mathbf{i}$ jsou vycentrované příznakové vektory trénovací množiny.

4 Implementace & Metacentrum

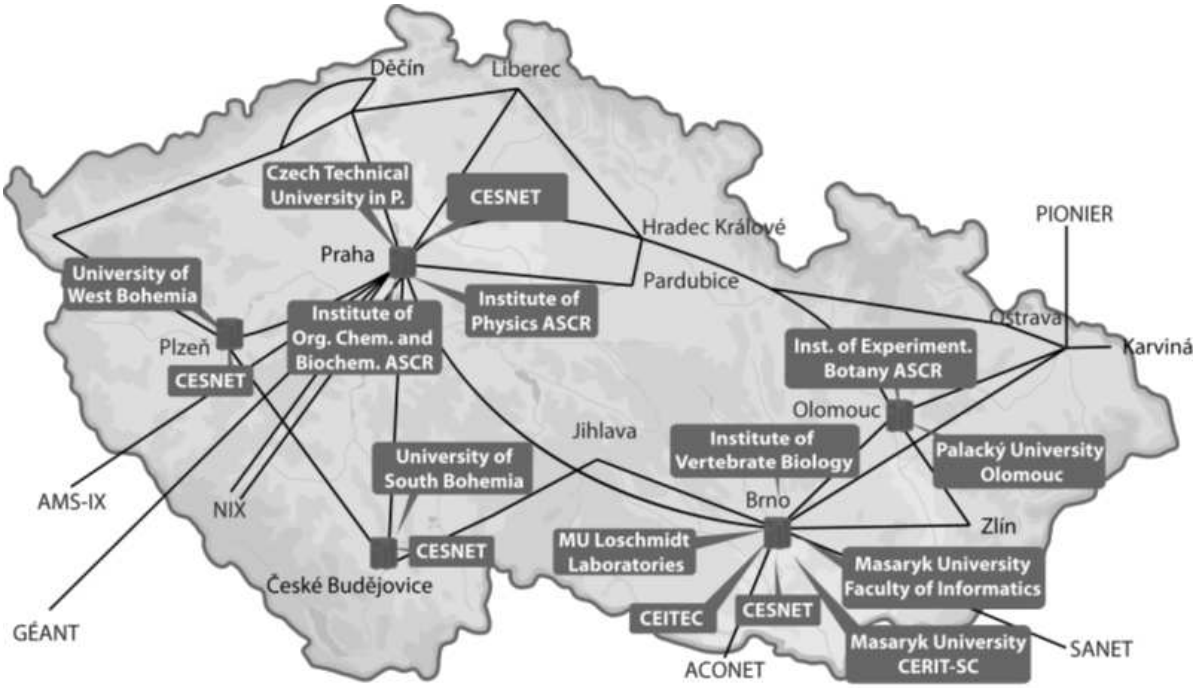
Existuje několik volně dostupných nástrojových sad, které jsou vyvíjené a používané ve výzkumu v oblasti zpracování řeči. Mezi nejznámější patří HTK [7], Open-Source Large Vocabulary CSR Engine Julius [9], CMU Sphinx [10], Quicknet [8]. Rychle se vyvíjejícím a hojně používaným balíčkem nástrojů je Kaldi [5], který byl zvolený jako platforma k realizaci úloh prezentovaných v této práci.

Úlohy rozpoznávání řeči jsou velmi náročné na výpočetní výkon. Jedná se zvláště o trénování, kde se pracuje s velmi složitými systémy, které je potřeba natrénovat a optimalizovat. Z praktického hlediska je potřeba tento výkon zajistit dostatečně výkonným hardware, paralelizací úloh nebo optimalizací algoritmů. Výpočetní prostředky dostupné na pracovišti, na kterém byla tato práce realizována, nebyly dostačující hlavně z důvodu většího počtu probíhajících projektů ostatních členů pracovní skupiny. Z tohoto důvodu bylo potřeba zajistit externí výpočetní prostředky. Pro provedení experimentů byly využity výpočetní a úložné zdroje národní gridové struktury Metacentrum [6].

4.1 Metacentrum

Metacentrum je společnost provozující mimo jiné i Národní Gridovou Strukturu, která je uznávanou součástí Evropské Gridové struktury. Jedná se o síť výpočetních a úložných prostředků dostupnou akademické komunitě v ČR. Přístup k prostředkům Metacentra je pro akademické pracovníky a studenty členů sdružení CESNET zdarma. Jediná podmínka pro využití výpočetních prostředků Metacentra je nutnost do všech publikací vzniklých s přispěním Metacentra vložit text s poděkováním. Tyto publikace je také nutné zadat do systému Metacentra. Tím je uživateli prodloužen přístup k výpočetním prostředkům Metacentra.

Metacentrum je skvělá příležitost pro akademické pracovníky a studenty, jak získat přístup k výpočetnímu výkonu. Výpočetní zdroje Metacentra se skládají z clusterů a SMP strojů, na kterých běží unixové operační systémy. Zjednodušenou strukturu, umístění serverů a spojení na další evropské servery ukazuje obrázek 4.1.1.



Obrázek 4.1.1 Mapa sítě serverů Metacentra. *Zdroj: http://metavo.metacentrum.cz/*

4.1.1 Přístup a struktura

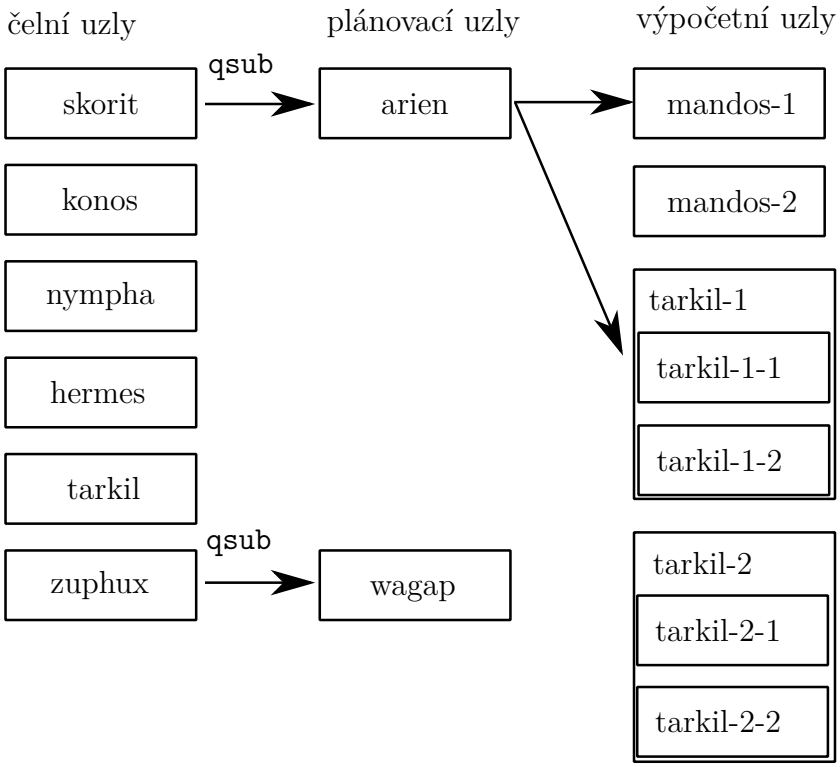
Pro přístup do sítě Metacentra a možnost využívat výpočetní zdroje je potřeba podat si přihlášku. Potvrzení totožnosti nutné ke kladnému vyřízení přihlášky je možné v případě velkých univerzit elektronicky přes eduID.cz. Pokud zastupující instituce není členem České akademické federace identit, je nutné provést potvrzení identity jinými prostředky.

Do sítě Metacentra se přistupuje pomocí ssh klienta přes několik čelních uzlů. Tyto uzly nejsou určeny k náročným výpočtům a jakýkoli uživatelský proces běžící déle než hodinu je násilně ukončen. Pro výpočetně náročné úlohy, u kterých nelze nebo není žádoucí spuštění v dávce, je možné spustit interaktivní úlohu na některém z výpočetních uzlů. Na výpočetní uzly je možné se přihlásit přes ssh, nicméně tento postup slouží jen pro zkontrolování stavu běžících úloh a neměl by sloužit k obcházení plánovacího systému Metacentra. Mezi čelními a výpočetními uzly jsou stroje plánovacího systému PBS. Ty řídí plánovací systém a nedá se na ně přihlásit žádným způsobem. Výpočetní uzly jsou tvořeny jednotlivými clustery, které eventuálně sdílejí diskový prostor *scratch*, ovšem nikoli operační paměť. Jednotlivé stroje v clusteru jsou SMP stroje, jež jsou virtualizované a tyto virtualizace sdílejí jak diskový prostor *scratch*, tak i operační paměť. Díky tomuto systému je možné plánovací systém požádat o stroj se specifickými parametry, jako je počet výpočetních jader a vláken na nich. Čelní uzly, plánovací servery a výpočetní uzly ilustrativně ukazuje obrázek 4.1.2.

Všechny čelní uzly Metacentra jsou přístupné přes doménu `.metacentrum.cz`. Přihlášení na ně je možné pomocí jakéhokoli ssh klienta pomocí příkazu:

`ssh <uziv_jmeno>@<celni_uzel>.metacentrum.cz`.

Je vhodné nastavit ssh klienta tak, aby v pravidelných intervalech žádal o oživení spojení. Toto opatření zabrání Metacentru v ukončení spojení z důvodu neaktivity. Každý uživatel má k dispozici několik domovských adresářů fyzicky rozmístěných na serverech po celé ČR. Tyto adresáře jsou omezeny velikostí na jednoho uživatele v řádu stovek GB. Omezení je pouze virtuální a může se měnit v závislosti na fyzické obsazenosti daného serveru daty dalších uživatelů. Informace o dostupných domovských adresářích, jako umístění, využité a volné místo, celkový počet CPU a dostupný počet CPU se zobrazují při přihlášení na Metacentrum.



Obrázek 4.1.2 Ilustrativní náčrt struktury serverů na Metacentru.

4.1.2 Moduly

Systém Metacentra poskytuje kromě výpočetního výkonu také různý software. Ten je ve formě modulů, které je možné naimportovat do úlohy a používat je. Seznam dostupných modulů je možné vyvolat příkazem:

```
module avail
```

Aby bylo možné modul používat, je nutné ho importovat příkazem:

```
module add <jmeno_modulu>
```

Tím se žádaný modul přidá do systémové proměnné `PATH`. Je důležité si uvědomit, že tato proměnná se v základním nastavení nepředává do spouštěných úloh a je tedy nutné ji předat, nebo provést přidání modulů u každé spuštěné úlohy zvlášť.

V případě, že požadovaný software není k dispozici, je možné ho, vzhledem k virtualizaci pracovního prostoru a tedy z uživatelského hlediska nezávislosti na typu hardware, samostatně zkompilovat do vlastního umístění a toto umístění pak přidat do proměnné `PATH`, když je potřeba. Toto je i případ nástrojů, které byly použity v této práci. Zkompilované nástroje byly umístěny do společného prostoru skupiny, tak aby byly přístupné všem jejím členům.

Kompilace všech nástrojů probíhala většinou bez větších obtíží. Bylo třeba vzít v úvahu nepřístupnost základní lokace pro výstup kompilace (`/usr/bin`) a správně přesměrovat kompilaci do náhradního adresáře (`/storage/projects/CTU_Speech_Lab/tools`). Také není vhodné provádět kompilaci na čelním uzlu Metacentra, neboť při jejím delším trvání by ji mohl plánovací systém násilně ukončit. Je vhodné vytvořit interaktivní úlohu (`qsub -I`) a všechny operace provádět v ní.

4.1.3 Skupina CTU Speech Lab a její pracovní prostor

S Metacentem je možné pracovat i bez členství ve výzkumné skupině. Postačující je akademická příslušnost. Pokud mají být prostředky Metacentra využívány více členy jedné skupiny, je výhodné na Metacentru požádat o vytvoření skupiny. Skupina má k dispozici společný diskový prostor, který je vhodný například pro uložení databází. Její správa je možná přes webové rozhraní Perun a je v režii správce skupiny. Po přihlášení do Metacentra tedy bylo požádáno o vytvoření skupiny pro výzkumnou skupinu CTU Speech Lab.

Ve sdílené složce (`/storage/projects/CTU_Speech_Lab`) byla vytvořena struktura složek s databázemi, pracovním prostorem pro jednotlivé členy skupiny a složkou s nástroji, které nejsou dostupné na metacentru a bylo nutné je zkompilovat zvlášť. K tomuto účelu slouží složka `tools` obsahující zdrojové soubory používaných nástrojů. Zkompilované nástroje jsou ve složce `tools/release`. Tuto složku je možné přidat do systémové proměnné `PATH`. Složka `data` obsahuje dostupné databáze signálů. Jedná se o databáze řečových signálů, které bylo potřeba naimportovat z úložného prostoru výzkumné skupiny. V rámci použití Metacentra i dalšími členy skupiny byly naimportovány i databáze v této práci nevyužité. Jedná se o databáze *GLOBALPHONE*, *SPEECON*, *SPEECHDAT* a *TEMIC* v dostupných jazycích. Složka `workspace` pak poskytuje diskový prostor jednotlivým členům skupiny, pokud nejsou dostačující jiné úložiště Metacentra.

```
|-- data
```

```
|-- tools
|   |-- release
|       |-- bin
|       |-- include
|       |-- lib
|       |-- share
|-- workspace
```

4.1.4 Plánovací systém

Metacentrum využívá plánovací systém Torque PBS, což je Open-source varianta plánovacího systému SGE od Oracle. Torque PBS pracuje na principu několika front, do kterých zařazuje úlohy, podle požadavků na výpočetní zdroje. Bez plánovacího systému by zdroje Metacentra mohly být využívány neefektivně, nebo jen několika málo uživateli. Pro práci s plánovacím systémem je potřeba umět předat úlohu plánovači, zjistit stav úlohy a případně ji smazat. K tomu slouží následující nástroje.

qsub

Úloha se předává plánovacímu systému pomocí příkazu `qsub`. Při spuštění úlohy je potřeba specifikovat tyto parametry:

- počet strojů - Určuje kolik fyzických strojů bude pro úlohu vyčleněno. Je potřeba si uvědomit, že každý takový stroj má vlastní scratch.
- počet CPU - Určuje kolik vláken bude k dispozici na jednom stroji. Pokud je skutečný výkon vyšší, než jaký dokáže poskytnout žádaný stroj, úloha je ukončena.
- množství RAM - Maximální množství RAM paměti dostupné pro úlohu. Pokud je tato hodnota překročena, úloha je ukončena.
- velikost scratch úložiště a typ - Maximální velikost lokálního úložiště úlohy a její typ. Typy scratch úložišť jsou:
  - local - Standardní lokální úložiště uzlu s velkou kapacitou.
  - ssd - Velmi rychlé úložiště. Omezeno velikostí.
  - shared - Standardní velkokapacitní úložiště sdílené v clusteru.

  Ne všechny clustery, nebo stroje mají k dispozici ssd a shared scratch. Volbou těchto typů scratch tedy razantně omezuje množství použitelných strojů.
- walltime - Určuje maximální čas přiřazený úloze plánovacím systémem. Pokud úloha neskončí sama do tohoto limitu, je plánovacím systémem násilně ukončena.

Pokud některý z parametrů není specifikován, je nastaven na základní hodnotu, jež udává tabulka 4.1.1.

počet strojů	1
počet CPU	1
množství RAM	400 MB
scratch	400 MB, local
walltime	24 hodin

Tabulka 4.1.1 Základní nastavení parametrů úloh Metacentra.

Mimo parametrů, je možné specifikovat několik přepínačů:

- -q - Umožňuje uživateli rozhodnout, do jaké fronty bude úloha zařazena. Přesto je však nutné specifikovat výše uvedené parametry. Pokud žádaná fronta nedokáže splnit požadavky specifikované parametry (hlavně walltime), úloha se nikdy nespustí. Tento přepínač je nutné použít, pokud mají být pro výpočet použity grafické karty.

- -v - Umožňuje předat proměnné do úlohy. Pokud je potřeba předat proměnnou PATH, provede se to zde pomocí příkazu `-v PATH=$PATH`. Je možné předat více proměnných.

qstat

Pro zjištění stavu úlohy se používá příkaz `qstat`. Tento příkaz bez dalších parametrů zobrazí všechny běžící a čekající úlohy a dokončené úlohy za posledních 24 hodin všech uživatelů Metacentra. Pokud je zadáno číselné ID úlohy, zobrazí informace o žádané úloze. Tyto informace jsou unikátní identifikátor úlohy, jméno úlohy, jméno uživatele, který spustil tuto úlohu, celkový čas úlohy přepočítaný na jeden stroj, stav úlohy a frontu, do které je úloha zařazena. Pokud je také zadán přepínač `-f`, je výpis daleko podrobnější a obsahuje například informace o alokovaných prostředcích a označení stroje, na kterém úloha běží. To je užitečné, pokud je potřeba se na tento stroj přihlásit a zkontrolovat stav úlohy v lokálním adresáři scratch. Další možností je přepínač `-u` následovaný uživatelským jménem. Tento příkaz zobrazí čekající a běžící úlohy a dokončené úlohy za posledních 24 hodin zadaného uživatele.

qdel

Pokud je potřeba zabránit spuštění úlohy, nebo přerušit již běžící úlohu, lze zavolat příkaz `qdel` následovaný číselným ID úlohy. Úloha je okamžitě označena za dokončenou a pokud již běžela, jsou vyčištěny všechny zabrané systémové prostředky.

4.1.5 Webové nástroje

Metacentrum poskytuje několik webových nástrojů zjednodušujících správu účtu, nebo skupin, poskytujících pomocné služby pro práci s plánovacím systémem a žádosti o výpočetní výkon. Všechny tyto nástroje jsou přístupné ze stránek metacentra [6] a k jejich použití je nutné se přihlásit.

Můj účet - Perun

Perun je webová aplikace pokrývající správu účtu a skupiny. Umožňuje nastavit detaily uživatele, základní nastavení zdrojů, změnu hesla a také nahrát publikaci s poděkováním, která je nutností pro prodloužení práv využívání výpočetních prostředků Metacentra. Také umožňuje spravovat skupinu. Přidávat a odebírat správce a členy.

Stav zdrojů

Tento nástroj umožňuje procházet celou historii úloh spuštěných uživatelem, také je zde pomocný nástroj umožňující zjistit, zda úloha s žádanými parametry může běžet, případně na kolika strojích celkově ji lze spustit a kolik jich je v tuto chvíli k dispozici. Mimoto tu jsou informace o veškerém hardware dostupném uživateli.

4.1.6 Spouštění úloh

Úloha na metacentru se dá pustit několika způsoby. Nedílnou součástí této práce bylo vyzkoušet tyto metody a najít takovou, která bude spolehlivá a vhodná pro úlohy rozpoznávání řeči. Výsledkem této části jsou dva způsoby výpočtu, kdy každý je vhodný pro jiný typ úlohy.

Paralelizace na clusteru

Úlohy, které pracují na velkém počtu nezávislých dat, je možné paralelizovat v clusteru, případně celém Metacentru. Výhodou tohoto přístupu jsou opravdu velké možnosti paralelizace. Takové úlohy mohou používat lokální scratch a po dokončení výpočtu data zkopírovat do trvalého úložiště společného pro všechny úlohy. To však v konečném důsledku může vést ke zpomalení úlohy z důvodu kopírování dat. Což bylo při pokusech pozorováno. Případně mohou data získávat a rovnou ukládat na trvalé úložiště. Tento postup však není doporučeno používat, neboť neúměrně zatěžuje infrastrukturu Metacentra a to jak disky trvalých úložišť, která nejsou na takovýto provoz vhodná, tak přenosové linky. V limitním případě tento postup může způsobit pád sítového úložiště a nedostupnost dat mnoha uživatelů.

Správně navrhnuta paralelizovaná úloha, přenese na lokální scratch potřebná data, provede výpočty, jejich výsledky ukládá lokálně a až konečný výsledek přenese na trvalé úložiště. S tímto postupem se však musí počítat už při návrhu aplikace. Pozdější implementace tohoto systému může být problematická, jak se i ukázalo při pokusech aplikovat tento postup na úlohu rozpoznávače s TANDEM architekturou.

Paralelizace na stroji

Další možností paralelizace je využít velkého množství procesorů v jednom stroji a požádat plánovač o jeden stroj s velkým počtem CPU. Výhoda tohoto systému je v dostupnosti jednoho lokálního scratch adresáře pro všechna vlákna úlohy. Další výhodou je rychlost dostupnosti dat. Většina strojů disponuje ssd scratch úložištěm s kapacitou v rozsahu desítek GB. To je pro úlohy prováděné v této práci dostačující a v okamžiku, kdy jsou všechna data nakopírovaná na toto úložiště, úlohu již nezpomaluje přenos dat, což se ukázalo i při experimentech. Nevýhodou může být nedostatek výkonu spojený právě s omezeným počtem procesů. Tento nedostatek je kompenzován high-end hardwarem. Ve výsledku úlohy spouštěné touto metodou běží v řádu hodin, zatímco předchozí metoda si vyžádá v nejzazším případě i několik dní.

Při tomto způsobu paralelizace v podstatě dochází k nakopírování celé úlohy na scratch a tak ji neohrožují pády diskových polí, ani krátkodobé latence.

Hlavní skript

Při spouštění samotné úlohy je potřeba udělat několik věcí. Nastavit parametry `qsub`, nakopírovat vstupní data a potřebné skripty do scratche, přepnout pracovní složku tamtéž, naimportovat moduly a nastavit další proměnné. K tomuto účelu slouží hlavní skript, který ve své hlavičce obsahuje parametry pro plánovací systém. Následně zkopíruje úlohu do scratche přiděleného úloze plánovačem. Z praktických důvodů se ukázalo vhodné po skončení kopírování dát uživateli najevo, že kopírování skončilo, úloha je na scratchi a je tedy bezpečné upravovat zdrojové skripty pro další konfiguraci. To se děje vytvořením souboru COPY_DONE ve zdrojové složce. Úloha pak změní pracovní adresář do scratche, naimportuje potřebné moduly a proměnné a spustí pracovní skript. Po dokončení výpočtu se scratch složka s výsledky nakopíruje do trvalého adresáře s výsledky. V případě, že kopírování selže, scratch se nesmaže a uživatel může výsledky vykopírovat ručně.

Praktická implementaci hlavního skriptu je vidět zde:

```
#/bin/bash
#PBS -l mem=32gb
#PBS -l walltime=1d
```

```
#PBS -l nodes=1:ppn=8
#PBS -l scratch=80gb
#PBS -q gpu
trap 'clean_scratch' TERM EXIT
workDir="s5_tandem_1"
DATADIR="/storage/projects/CTU_Speech_Lab/workspace/brichale/$workDir"
OUTDIR="$DATADIR/../results"
cp -R $DATADIR $SCRATCHDIR || exit 1
touch $DATADIR/COPY_DONE
cd $SCRATCHDIR/$workDir || exit 2
. ./path.sh
. ./cmd.sh
./run.sh
cp -R $SCRATCHDIR $OUTDIR || export CLEAN_SCRATCH=false
```

Řádky začínající `#PBS` jsou parametry pro plánovač a jsou ekvivalentní se zápisem:

```
qsub -l mem=32gb ... -l scratch=80gb -q gpu main_script.sh
```

Takto stačí volat:

```
qsub main_script.sh
```

4.2 CtuCopy

Pro výpočet příznaků byl použit nástroj CtuCopy. Ten vznikl na katedře zpracování signálu, ČVUT v Praze. Pracuje s různými typy vstupních a výstupních souborů. Umožňuje aplikovat různé banky filtrů, redukovat šum a jako výstup má širokou paletu příznakových vektorů a jejich diferencí. Nastavení může být pomocí přepínačů, nebo z konfiguračního souboru. Jednoduchost a přehlednost z něj dělá užitečný nástroj pro výpočet příznaků.

Před samotnou kompilací CtuCopy bylo potřeba zkompilovat knihovnu FFTW [11], což je rychlá opensource knihovna pro výpočet Fourierovy transformace. Zkompilovaná knihovna byla umístěna do společného adresáře skupiny. Pro kompilaci samotného CtuCopy bylo potřeba přesměrovat výstupní lokaci do společného adresáře skupiny a tamtéž hledat i FFTW. Tato změna se týkala tří souborů s nastavením:

- objects.mk - Řádek 7. Parametr `-L` s cestou k FFTW knihovně.
- subdir.mk - Řádek 63 a 70. Parametr `-I` cesta k hlavičkovému souboru FFTW knihovny.

- makefile - Řádek 53. Parametr `-I` cesta k hlavičkovému souboru FFTW knihovny.

Výsledek kompilace je spustitelný soubor *ctucopy4*, který je potřeba nakopírovat do sdílené složky skupiny s nástroji `tools/release/bin`.

4.3 Kaldi

Kaldi je balíček nástrojů pro rozpoznávání řeči. Má rozsáhlou dokumentaci a je volně k dispozici. Začal vznikat v roce 2009. Tehdy byl založen na HTK. Již od roku 2010 je prezentován jako balíček nástrojů nezávislý na jiných balíčcích. V dnešní době jde o jeden z nejpoužívanějších balíčků. Snaží se být moderním, flexibilním, snadno upravitelným a rozšiřitelným. Pracuje na moderních knihovnách pro matematické operace BLAS a LAPACK. Při kompilaci pak využívá OpenFST k integraci konečných stavových transducerů (FST).

Díky spolupráci mezi vývojovým týmem a komunitou vzniklo mnoho receptů. Jedná se o vzorové skripty distribuované spolu s nástroji Kaldi. Na těchto receptech je jednak možné prozkoumat, jakým způsobem Kaldi funguje a mělo by se používat a jednak slouží jako solidní základ pro vlastní experimenty.

Recepty reprezentují příklady práce s různými databázemi na rozličných experimentech. Recepty obsahují jednoduché rozpoznávače povelů, číslic, které jsou díky své jednoduchosti vhodné pro seznámení s Kaldi nástroji a pomocnými skripty. V receptech jsou zastoupeny i velmi složité úlohy na rozsáhlých databázích, jako jsou Wall Street Journal, které obsahují desítky hodin dat. Tyto recepty jsou často velmi složité, nicméně přehledně implementované a slouží jako kvalitní základ pro vlastní experimenty. Skripty vytvořené a prezentované v této práci vycházejí z velké části právě z receptu rozpoznávače na databázi Wall Street Journal.

4.3.1 Kompilace Kaldi

Kompilace Kaldi probíhala až na drobnosti podle originálního návodu dostupného u zdrojových souborů. Kompilaci je možné provést na hlavním uzlu, ale zabere nějaký čas a může se stát, že ji plánovač přeruší. Z toho důvodu je jistější spustit interaktivní úlohu s dostatkem paměti a v případě využití vícevláknové kompilace i s více CPU.

- Při ověřování dostupnosti závislostí je nutné aktivovat modul subversion pomocí `module add svn-1.7.6`.
- Při kompilaci je ve složce `tools/` možné nastavit verzi knihovny OpenFst.
- Ve složce `src/`, při kompilaci samotného Kaldi, je potřeba správně nastavit příkaz `configure`, tak aby se úspěšně zkompilovala CUDA:

– Nejprve je potřeba zjistit umístění CUDA knihoven pomocí

```
module add cuda-7.0
echo $PATH
```

Metacentrum umožňuje přidat i modul `cuda-7.5`, ale s tímto modulem nejsou kompatibilní všechny grafické karty dostupné na Metacentru. Použití verze 7.0 tento problém odstraňuje. Cestu k CUDA knihovnám lze zístak ze systémové proměnné `$PATH` a vypadá takto `/software/cuda/7.0/bin`. Pro správnou kompilaci Kaldi, je třeba zavolat příkaz

```
./configure --cudatk-dir="/software/cuda/7.0"
```

– V nastavení `configure` je možno provést další úpravy, například použití statických knihoven.

– Také je možné pustit kompilaci na více vláknech, ovšem v takovém případě je nutné si požádat o vícevláknový stroj.

4.3.2 Kaldi recepty

Jak již bylo několikrát naznačeno, Kaldi je úzce provázáno se vzorovými recepty, které jsou s ním přímo distribuovány. Adresářová struktura receptů je standardizovaná a vychází z receptů pro databázi Wallstreet Journal. Základem jsou tyto soubory a složky:

- conf - Složka obsahující konfigurační soubory.
- local - Složka obsahující skripty specifické pro danou úlohu.
- steps - Jedna se složek obsahující společné skripty pro každý recept. Tyto skripty reprezentují větší celky výpočtu, neměnící se pro různé experimenty. Skripty v této složce se zásadně nemodifikují a ve většině řešeních jsou jen nalinkované z nějakého společného umístění.
- utils - Druhá složka obsahující společné skripty pro každý recept. Obsahuje spíše pomocné skripty. Skripty v této složce se zásadně nemodifikují a ve většině řešeních jsou jen nalinkované z nějakého společného umístění.
- cmd.sh - Tento skript nastavuje proměnné důležité pro správný běh skriptů. Jedná se například o specifikaci skriptu spouštějícího paralelní úlohy, jeho parametry, počet paralelních úloh. V tomto skriptu je možné změnit paralelní způsob výpočtu na lokální bez nutnosti hledat a zasahovat v dalších skriptech.
- path.sh - Tento skript přidává do proměnné `PATH` Kaldi nástroje. Volá se automaticky na všech místech, kde je třeba mít jistotu, že jsou tyto nástroje k dispozici. Pokud je

tedy potřeba zajistit dostupnost nějakého nástroje v celé úloze, tento skript je nejlepší způsob, jak toho dosáhnout. Přidávání modulů, knihoven a dalších nástrojů do tohoto souboru zajistí jejich dostupnost v každé paralelní úloze a každém spouštěném skriptu bez nutnosti je modifikovat a přidávat je ručně.

Na kostře výše uvedených souborů a složek je výhodné postavit další úlohy. Při dodržení pár základních pravidel tak zůstane zachována modulárnost. Složky `steps` a `utils` jsou na Metacentru k dispozici ve sdíleném adresáři `tools/kaldi/kaldi-trunk`.

4.4 Příklad implementace

K této práci je přiloženo několik složek, které obsahují úlohy s rozdílnou funkcí. Tyto úlohy dodržují konvence zavedené Kaldi recepty. Pro spouštění s plánovacím systémem Metacentra obsahují hlavní skript `main_run_xxx.sh`. Ten pak spouští samotnou úlohu ve skriptu `run.sh`. Specifický běh úlohy bude popsán pro každou úlohu zvlášť. Přesto většina použitých skriptů je pro všechny úlohy totožná. Je potřeba si uvědomit, že mnohé skripty jsou univerzální a tak některé generované soubory v úloze nejsou použity.

4.4.1 Skripty pro přípravu dat

Tyto skripty pracují přímo s databází dat. Generují seznamy, připravují slovníky a počítají příznaky.

- **local/speecon_data_prep.sh** - Tento skript připravuje seznamy dat pro zpracování. Z `CONTENT` souboru databáze vybere signály z požadovaného prostředí a kanálu. Z tohoto seznamu odeberou defektní mluvčí, jejichž seznam je uveden v souboru `conf/speecon/bad_spk`. Tento opravený list rozdělí na trénovací, testovací a development množinu v poměru 8:1:1. V tuto chvíli se operace skriptu ztrojí pro každý z těchto listů. Vytvoří se soubory mapující mluvčí na pohlaví, seznam signálů, přičemž pro test a dev množinu se vybírají pouze věty. Dále se generují soubory s přepisem. Soubory přiřazující mluvčímu signály a signálům mluvčího. Fonetické přepisy a textový korpus.

- **local/speecon_prep_dict.sh** - Tento skript připravuje slovník. Transformuje originální slovník databáze do formátu používaného dalšími nástroji.

- **utils/prepare_lang.sh** - Připravuje složku `data/lang`.

- **local/speecon_format_test_G.sh** - Vytváří graf gramatiky.

- **utils/copy_data_dir.sh** - Skript poskytující kopírování pro složku obsahující listy generované skriptem `speecon_data_prep.sh`. Při kopírování umožňuje přidat předpony a přípony mluvčího a promluv.

- **utils/subset_data_dir_tr_cv.sh** - Tento skript dělí seznamy na trénovací množinu a krosvalidační množinu v požadovaném poměru.

4.4.2 Skripty pro zpracování dat

Tyto skripty zajišťují výpočty s pomocí předpřipravených souborů a seznamů.

- **local/make_fea_ctucopy.sh** - Skript počítající akustické příznaky s pomocí nástroje *ctucopy4*. Implementuje paralelizaci výpočtu. Spravuje rozdělení dat do paralelně zpracovávaných množin a jejich následné spojení po dokončení parametrizace. Nastavení parametrů *ctucopy* probíhá pomocí konfiguračního souboru.

- **steps/compute_cmvn_stats.sh** - Normalizuje střední hodnoty a rozptyly promluv pro mluvčího.

- **local/nnet/train_nnet.sh** - Tento skript se stará o definici, inicializaci a trénování umělých neuronových sítí. Umožňuje nastavit počet vstupních, výstupních neuronů. Počet skrytých vrstev a neuronů v nich. Pokud není poskytnuta inicializace sítě, provede inicializaci pomocí náhodných čísel. Umožňuje provádět transformace na vstupních příznakových vektorech a splicing, tedy spojení několika příznakových vektorů do jednoho k získání kontextu na vstupu sítě. Dokáže vytvořit i síť typu bottleneck, ovšem nepodporuje inicializaci vhodnou pro DNN a tedy pokud není poskytnuta již nainicializovaná síť, vždy vytvoří jen síť typu MLP. Umožňuje výpočet na grafické kartě.

- **local/get_posteriors.sh** - Tento skript vyhodnocuje průchod příznakového vektoru již natrénovanou sítí a slouží tedy k evaluaci a získání aposteriorních pravděpodobností. Dovoluje také pravděpodobnosti rovnou logaritmovat a tím jim zajistit gaussovské rozdělení. Umožňuje výpočet na grafické kartě.

- **local/ctu/merge_feats.sh** - Provádí spojení dvou příznakových vektorů do jednoho.

- **local/tandem/train_mono.sh** - Provádí trénování monofonů. Během trénování přidává Gaussovské komponenty.

- **local/tandem/train_deltas.sh** - Provádí trénování trifonů. Během trénování přidává Gaussovské komponenty.

- **local/tandem/align_si.sh** - Zarovnává data s použitím napočítaných modelů.
- **utils/mkgraph.sh** - Vytváří dekódovací HCLG graf, který reprezentuje jazykový model, lexikon, kontextovou závislost a HMM strukturu modelů. Výsledný graf je typu WFST.
- **steps/decode.sh** - Provádí dekódování z HCLG grafu.

4.4.3 Implementace GMM-HMM

Složka `s5_tandem_GMM` obsahuje implementaci GMM-HMM rozpoznávače s kepstrálními příznaky s delta a delta-delta koeficienty. Tento recept slouží k získání zarovnaných modelů pro trénování umělých neuronových sítí. Také umožňuje získat referenční výsledky netandemového systému. Skript má několik fází:

1. **Příprava dat** - Během této fáze se připravují seznamy pro práci s databází, generuje se slovník a gramatika. Volají se skripty `speecon_data_prep.sh`, `speecon_prep_dict.sh`, `prepare_lang.sh` a `speecon_format_test_G.sh`.
2. **Výpočet příznaků** - V této fázi dochází k výpočtu příznakových vektorů melovských kepstrálních koeficientů (skript `make_fea_ctucopy.sh` a normalizace jejich CMV (skript `compute_cmvn_stats.sh`).
3. **Trénování monofonů a trifonů** - Pomocí skriptu `train_mono.sh` dojde k natrénování monofonů. Natrénované modely jsou použity k zarovnání dat pomocí skriptu `align_si.sh`. Zarovnaná data jsou dále použita k trénování trifonů pomocí skriptu `train_deltas.sh`. V tomto případě se trénuje 2500 trifonů s celkovým maximálním počtem Gaussovských komponent 80000.
4. **Generování HCLG grafů** - V této fázi se pomocí skriptu `mkgraph.sh` vytváří HCLG grafy pro monofony a trifony.
5. **Dekódování** - HCLG grafy vytvořené v předchozí fázi jsou v této fázi využity k dekódování (`decode.sh`).
6. **Generování zarovnaných dat pro ANN** - Poslední fáze je není nedílnou součástí běžného běhu skriptu. Slouží k vytvoření zarovnaných dat pro účely trénování ANN. Ke správnému vygenerování těchto dat je nutné, aby proběhly tyto fáze: Příprava dat, Výpočet příznaků, Trénování monofonů a trifonů. Na tyto fáze navazuje tato fáze, která provede přetrénování trifónů na lépe přizpůsobené. Pomocí těchto nových trifónů jsou následně zarovnána data použitelná pro trénování ANN.

4.4.4 Trénování sítí

Složka `s5_tandem_ANN` obsahuje úlohu trénování ANN. Běh této úlohy je rozdělen na dvě fáze:

1. **Příprava dat** - První část této fáze je shodná s fází 1 z předchozího skriptu. Po její evaluaci se do úlohy nakopírují zarovnaná data, která byla rovněž vypočtena předchozím skriptem. Následně se zkopírují seznamy generované v první části do nového umístění (skript `copy_data_dir.sh`) a vypočítají se příznaky a CMV normalizace. Jako poslední se trénovací množina rozdělí na trénovací data a krosvalidační pomocí skriptu `subset_data_dir_tr_cv.sh`.

2. **Trénování ANN** - Skript `train_nnet.sh` trénuje ANN. Pomocí parametrů je potřeba specifikovat typ sítě, její rozměry, zřetězení příznakových vektorů na vstupu, learning rate.

4.4.5 Trénování rozpoznávače

Složka `s5_tandem_train` obsahuje úlohu, ve které se sestavuje a trénuje rozpoznávač pro specifický kanál. Její běh má šest fází:

1. **Příprava dat** - Příprava dat pro rozpoznávač je téměř shodná s přípravou při trénování sítí. Rozdíl je v tom, že kromě příznaků pro ANN se musejí napočítat i kepstrální příznaky, které budou druhou částí příznakového vektoru TANDEM architektury.

2. **Výpočet aposteriorních pravděpodobností** - V této fázi se nejprve do úlohy nakopíruje dříve natrénovaná ANN. Ta se použije k namapování vstupních příznaků na aposteriorní pravděpodobnosti, které jsou rovnou zlogaritmovány a je na nich provedena normalizace. Mapování je záležitostí skriptu `get_posteriors.sh`, výpočet CMVN pak již použitého `compute_cmvn_stats.sh`.

3. **Spojení příznaků** - V této fázi skript `merge_feats.sh` spojí kepstrální koeficienty s transformovanými aposteriory.

4. **PCA** - Aby bylo možné aplikovat PCA, je nejprve nutné vyčíslit transformační matici, pomocí `est_pca.sh`. Následně je možné tuto matici aplikovat pomocí `apply_pca.sh`.

5. **Trénování a HCLG grafy** - V této fázi jsou natrénovány monofony, sestaven HCLG graf monofonů a vypočteno zarovnání monofonů. Následně jsou natrénovány trifony a sestaven druhý HCLG graf.

6. **Dekódování** - V poslední fázi je možné sestavený rozpoznávač ihned otestovat na trénovacím kanále.

4.4.6 Testování rozpoznávače

Ve složce `s5_tandem_test` je implementována úloha testující sestavený rozpoznávač. Tento rozpoznávač je možné automaticky otestovat na vybraných kanálech jedním spuštěním skriptu. Skript má pět fází, které jsou téměř totožné s úlohou Trénování rozpoznávače:

1. **Příprava dat** - Tato fáze je zcela totožná s Přípravou dat při Trénování rozpoznávače.

2-4. **Výpočet aposteriorních pravděpodobností, Spojení příznaků, PCA** - V těchto fázích je jediná změna. Vzhledem k pouhému testování rozpoznávače, není potřeba provádět výpočet pro trénovací množinu.

5. **Dekódování** - V této fázi dojde k importu rozpoznávače monofonů a trifonů natrénovaných v úloze Trénování rozpoznávače. Po dekódování monofonů jsou natrénované i trifony.

Celý tento postup je uzavřen do cyklu, který iteruje přes požadované testovací kanály. Vzhledem k používané struktuře adresářů by se při dalším cyklu původní výsledky přepsaly. Z tohoto důvodu jsou soubory s výslednými WER překopírovány do unikátního adresáře.

5 Experimentální část

Tato část shrnuje realizované experimenty, na kterých byla testována úspěšnost a robustnost rozpoznávání příznaků různé TANDEM architektury na úloze s velkým slovníkem. Nejprve bylo nutné najít optimální parametry rozpoznávače pro příznaky TANDEM architektury. Optimální rozpoznávač pak testoval příznakové vektory vytvářené pomocí 3 a 8-vrstvé MLP sítě z TRAPs, MFCC_0 a MFCC_0_D_A příznaků na různě zarušených datech a prostředích.

Jako hodnotící kritérium byla použita hodnota Word Error Rate (WER), která je dána vztahem 5.0.1. Tato metrika je standardní v úlohách rozpoznávání řeči.

$$WER = \frac{S + D + I}{N} \tag{5.0.1}$$

Kde S je počet substitucí v promluvě, tedy kolik slov bylo klasifikováno chybně. D je počet smazaných slov, tedy takových, které v promluvě jsou, ale rozpoznávač je nerozpoznal. I je počet chybně vložených slov, tedy nikoli nesprávně rozpoznaných, ale vložených i přesto, že v testovací promluvě na tomto místě žádné slovo není. N je pak součet $S + D + C$, kde C je počet správně klasifikovaných slov. Výsledek WER je po vynásobení 100 v procentech.

Data, použitá pro výpočet příznaků k trénování i testování, jsou z české části databáze SPEECON [17]. Tato databáze obsahuje promluvy od různých mluvčích, z různých prostředí a z různých záznamových zařízení. V této práci byla použita data z prostředí OFFICE, CAR_ON, CAR_OFF, PUBLIC_OPEN a PUBLIC_HALL. OFFICE představuje data nahrávaná v tichém prostředí. CAR_OFF jsou data zaznamenávaná v automobilu s vypnutým motorem, CAR_ON také v automobilu, ale se zapnutým motorem, PUBLIC_OPEN jsou nahrávky z otevřeného veřejného prostranství a PUBLIC_HALL pak z uzavřeného prostoru. Pro trénování byl použit kanál CS0 z prostředí OFFICE, který obsahuje jen málo šumu. Testování probíhalo na všech kanálech, tedy CS0, CS1, CS2 a CS3. Každý z těchto kanálů reprezentuje jeden mikrofón s různou kvalitou záznamu a umístěním vůči mluvčímu. Kanál CS0 je nejkvalitnější s nejčistším signálem vysoké kvality. Kanály CS1 a CS2 reprezentují data od mikrofonů s různou kvalitou záznamu zhruba ve stejné vzdálenosti od mluvčího. Kanál CS3 je tvořen daty z mikrofonu daleko od mluvčího s nevysokou kvalitou záznamu. V tomto kanále je nejvíc šumu a ruchů z prostředí. Databáze byla rozdělena na trénovací, develop a testovací data v poměru

8:1:1. Testovací a develop množiny byly tvořeny pouze promluvami s celými větami.

5.1 Rozpoznávač

Všechny experimenty byly realizovány na úloze rozpoznávání spojité řeči s velkým slovníkem (LVCSR). Pro porovnání robustnosti příznakových vektorů, probíhalo trénování a dekódování na monofonech a trifonech. Pro příznakové vektory TANDEM architektury bylo potřeba najít optimální nastavení rozpoznávače. Sledované parametry byly dimenze PCA, celkový počet Gaussovských komponent a počet trénovacích operací. Byly provedeny analýzy za účelem zjištění hodnot blízkým optimálním. Pro zjednodušení byly tyto hodnoty považovány za nezávislé a testovány každá zvlášť. Pro testovací účely byl zvolen příznakový vektor spojený z MFCC_0_D_A a aposteriorních pravděpodobností vypočítaných s pomocí MLP sítě z TRAP příznaků s 4096 neurony. Optimální hodnoty byl hledány na DEV množině a posléze ověřeny na TEST množině dat.

5.1.1 PCA - dekorelace a redukce dimenze příznaků

Jako první byla hledána hodnota zkrácení dekorelovaných příznaků po PCA pro optimální počet použitých příznaků tak, aby výsledný příznakový vektor obsahoval pouze malé množství neužitečné informace. Počet Gaussovských komponent byl 15000 a počet trénovacích iterací 35. Dimenze příznaků byla volena 30, 40, 50, 60 a 70. Výsledné hodnoty WER pro DEV i TEST množinu na trifonech ukazuje tabulka 5.1.1.

Dimenze PCA	WER DEV [%]	WER TEST [%]
30	39.67	34.14
40	36.41	31.15
50	34.19	29.27
60	33.21	29.22
70	33.52	30.21

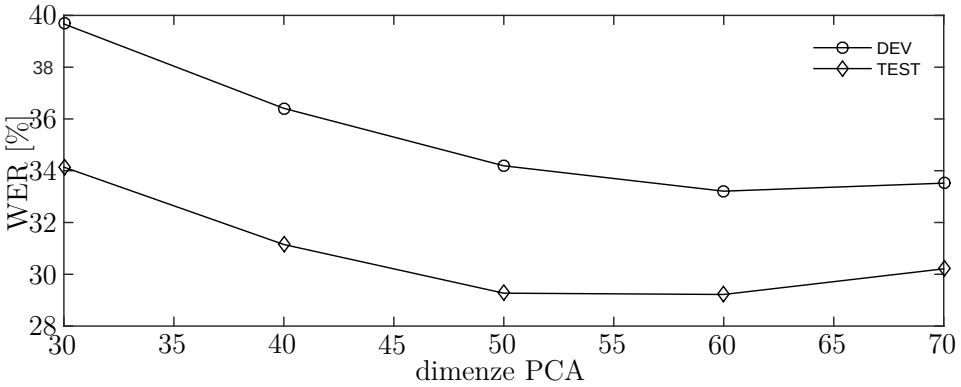
Tabulka 5.1.1 Tabulka vlivu dimenze PCA.

Jak je vidět, optimální dimeze PCA je blízko 60. Trend je lépe vidět z grafu 5.1.1.

5.1.2 Optimalizace počtu komponent GMM modelu

Jako další bylo potřeba najít optimální počet Gaussovských směsí u akustického modelu. Dimenze PCA byla zvolena optimálních 60, počet trénovacích iterací 35. Výsledné hodnoty WER pro DEV i TEST množinu pro zvolené počty komponent na trifonech ukazuje tabulka 5.1.2.

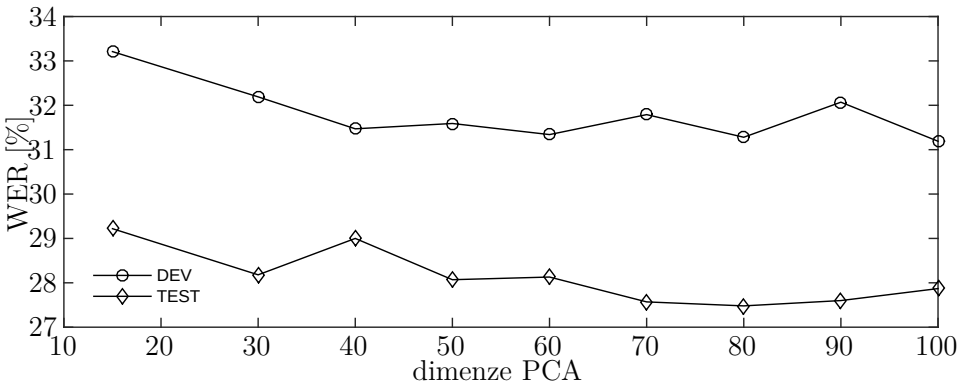
Tentokrát není minimum tak jednoznačné, jako v předchozím případě. Trendy ukazuje graf 5.1.2. Jako optimální hodnota bylo zvoleno 80000 Gaussovských komponent.



Obrázek 5.1.1 Trend úspěšností DEV a TEST množiny v závisloti na dimenzi PCA.

Počet komp.	WER DEV [%]	WER TEST [%]
15000	33.21	29.22
30000	32.19	28.18
40000	31.47	29.00
50000	31.59	28.07
60000	31.34	28.13
70000	31.79	27.57
80000	31.28	27.48
90000	32.07	27.60
100000	31.19	27.87

Tabulka 5.1.2 Tabulka vlivu počtu Gaussovkých komponent na WER.



Obrázek 5.1.2 Trend úspěšností DEV a TEST množiny v závisloti na počtu Gaussovských komponent.

5.1.3 Optimalizace trénovacích kroků

Fluktuace v úspěšnostech v předcházející analýze může být způsobena malým počtem trénovacích kroků, kdy hodnoty nestihly konvergovat k minimu. Otestování této hypotézy bylo provedeno pro PCA dimenzi 60 s 80000 Gaussovskými komponentami, postupně na 35, 55 a 75 iteracích. Výsledky ukazuje tabulka 5.1.3.

Z tabulky je vidět, že pro 75 iterací dojde k přetrénování. Pro použitý typ příznakového

Počet iterací.	WER DEV [%]	WER TEST [%]
35	31.28	27.48
55	31.14	26.95
75	31.47	27.35

Tabulka 5.1.3 Tabulka vlivu počtu přetrénovacích kroků na úspěšnost.

vektoru se ukazuje optimální hodnota kolem 55 iterací.

5.2 TANDEM

V experimentech byly použity dvě neuronové sítě, Multi Layer Perceptron se standardními 3 vrstvami (1 skrytá) a pak rozšířený s 8 vrstvami (6 skrytých). K získání pravděpodobností z těchto sítí byly na jejich vstupu použity tři typy příznakových vektorů:

- TRAP
- Melovské kepstrální koeficienty statické
- Melovské kepstrální koeficienty s delta a delta-delta koeficienty

Pravděpodobnostní výstupy ze sítě byly po zlogaritmování spojeny s Melovskými kepstrálními koeficienty s delta a delta-delta koeficienty, čímž vznikl příznakový vektor TANDEM architektury. Na tyto spojené příznaky byla aplikována PCA za účelem dekorelace příznaků a omezení jejich dimenze.

V případě kepstrálních příznaků bylo na vstupu ANN řetězeno několik příznakových vektorů k zajištění dalšího kontextu. Kontext tvořilo konkrétně sedm příznakových vektorů na každou stranu od středového vektoru. Pro statické MFCC tak byl na vstupu ANN vektor o délce 195 (15 vektorů po 13 příznacích). Pro MFCC s dynamickými parametry pak 585 (15 vektorů po 39 příznacích).

Při použití TRAP příznaků tento kontext není potřeba, neboť TRAPs zahrnují dostatečný kontext. Zřetězení těchto příznakových vektorů vede ke snížení úspěšnosti tak, jak ukazuje tabulka 5.2.1. Tento pokles může být způsoben nadbytkem informace, kdy díky kontextu dochází k přílišnému překrývání příznaků a tím k jejich znehodnocení.

Příznaky	WER DEV [%]	WER TEST [%]
TRAP bez kontextu	39.67	34.14
TRAP -1 0 +1 kontext	40.71	34.49

Tabulka 5.2.1 Tabulka vlivu kontextu TRAP na vstupu ANN.

Jako druhá složka výsledného příznakového vektoru TANDEM architektury byly použity MFCC s delta a delta-delta koeficienty. Ty se spojily s logaritmovanými aposterior-

ními pravděpodobnostmi a na tomto vektoru byla prováděna PCA. Výsledný příznakový vektor byl dimenze 60, což se ukázalo jako optimální.

5.2.1 TRAP

Pro výpočet TRAP příznakových vektorů je zvolená délka trajektorie 15 frame na každou stranu. To při kroku 10 ms vytváří trajektorii o celkové délce 310 ms. Počet příznaků vybraných po DCT transformaci je 16. Počet filtrů aplikovaných na spektrum je 23. To dohromady vytváří příznakový vektor délky 368.

5.2.2 Kepstrální příznaky

Pro výpočet Melovských kepstrálních koeficientů je na krátkodobé spektrum aplikována banka filtrů s 30 filtry. Z těchto hodnot je napočítáno kepstrum, z něhož je vybráno 13 prvních příznaků. Po výpočtu delta a delta-delta koeficientů je výsledný počet kepstrálních příznaků 39.

5.2.3 3-vrstvá MLP síť

3-vrstvá MLP neuronová síť byla trénována s 4096 neurony ve skryté vrstvě. Inicializace sítě proběhla náhodnými hodnotami. Trénovací množinu tvořilo 45515 signálů. Cross-validační množinu pak 5114 signálů. Počet iterací potřebných k natrénování a úspěšnost sítě ukazuje tabulka 5.2.2.

Příznaky	počet iterací	přesnost na cv [%]
MFCC_0	16	83.8942
MFCC_0_D_A	18	85.0118
TRAP	15	85.1144

Tabulka 5.2.2 Tabulka přesnosti natrénování 3-vrstvých MLP sítí.

5.2.4 8-vrstvá MLP síť

Použitou hlubokou neuronovou síť tvoří šest skrytých vrstev po 2048 neuronech. Inicializace sítě byla provedena náhodnými čísly. Trénovací množinu tvořilo 45515 signálů. Cross-validační množinu pak 5114 signálů. Počet iterací potřebných k natrénování a úspěšnost sítě ukazuje tabulka 5.2.3.

5.3 Výsledky

Bylo provedeno několik experimentů porovnávající robustnost různých příznakových vektorů TANDEM systémů pro prostředí OFFICE. Byly vyzkoušeny dva typy neu-

Příznaky	počet iterací	přesnost na cv [%]
MFCC_0	18	84.6754
MFCC_0_D_A	17	86.7898
TRAP	20	87.3327

Tabulka 5.2.3 Tabulka přesnosti natrénování 8-vrstvých MLP sítí.

ronových sítí s parametry popsanými v předchozí části. Oba typy sítí byly trénovány na třech různých příznakových vektorech a to na TRAP a Melovských kepstrálních koeficientech jednou bez dynamických koeficientů a podruhé s nimi. Po nalezení nejúspěšnější kombinace příznaků, byla tato konfigurace otestována na dalších prostředích.

5.3.1 GMM-HMM

Standardní GMM-HMM rozpoznávač se statickými, dynamickými a akceleračními Melovskými kepstrálními příznaky byl použit pro získání kvalitních zarovnaných dat pro trénování a jako referenční ukázka robustnosti netandemových architektur. Výsledky na trifonovém rozpoznávači ukazuje tabulka 5.3.1.

kanál	CS0	CS1	CS2	CS3
WER DEV [%]	31.86	44.98	49.12	65.97
WER TEST [%]	28.52	36.67	44.86	58.47

Tabulka 5.3.1 GMM-HMM výsledky

5.3.2 3-vrstvá MLP síť

První sada experimentů probíhala na umělé neuronové síti MLP s jednou skrytou vrstvou. Počet neuronů ve skryté vrstvě byl nastaven na 4096.

TRAP

TRAP příznaky mají díky velkému kontextu vysokou dimenzi. Nicméně také obsahují velké množství potenciálně užitečné informace. Jejich namapováním pomocí MLP dochází k redukci dimenze a tedy ke zhuštění informace. Velký kontext navíc může přinášet výhodu ve zvýšené robustnosti pro méně kvalitní vstupní data. Výsledky na trifonovém rozpoznávači s příznakovým vektorem složeným z MFCC_0_d_a a aposteriorních pravděpodobností vypočtených MLP sítí z TRAPs příznaků ukazuje tabulka 5.3.2.

MFCC_0

Příznakový vektor statických Melovských kepstrálních příznaků má dimenzi 13. Pokud tento vektor použijeme samotný jako vstup ANN, neuronová síť tento vektor musí na-

kanál	CS0	CS1	CS2	CS3
WER DEV [%]	31.59	59.72	61.84	96.43
WER TEST [%]	26.95	46.73	61.30	93.29

Tabulka 5.3.2 Tabulka výsledků příznaků TANDEM architektury složené z pravděpodobnostních příznaků z MLP sítě s TRAP příznaky.

mapovat na 45 výstupů. Výhoda pro jeden vstupní vektor je tedy diskutabilní. Použití statických kepstrálních příznaků na vstupu sítě má smysl v případě, že je zřetězeno několik kepstrálních příznakových vektorů. V experimentu bylo použito 15 příznakových vektorů, což znamená 7 na každou stranu od středového. Toto zřetězení přidá informaci o kontextu a rozšíří množství příznaků na vstupu sítě, které mohou být redukovány do zhuštěné informace. Výsledky na trifonovém rozpoznávači s příznakovým vektorem složeným z MFCC_0_d_a a aposteriorních pravděpodobností vypočtených MLP sítí z řetězených MFCC_0 příznakových vektorů ukazuje tabulka 5.3.3.

kanál	CS0	CS1	CS2	CS3
WER DEV [%]	30.93	61.76	65.28	96.47
WER TEST [%]	26.80	51.24	65.49	92.83

Tabulka 5.3.3 Tabulka výsledků příznaků TANDEM architektury složené z pravděpodobnostních příznaků z MLP sítě s MFCC_0 příznaky.

MFCC_0_D_A

Kepstrální příznaky s delta a delta-delta koeficienty již poskytují informaci o kontextu. Jejich zřetězení může mít podobný důsledek jako zřetězení TRAPs příznaků diskutované výše. Výsledky na trifonovém rozpoznávači s příznakovým vektorem složeným z MFCC_0_d_a a aposteriorních pravděpodobností vypočtených MLP sítí ze zřetězených MFCC_0_d_a příznakových vektorů ukazuje tabulka 5.3.4.

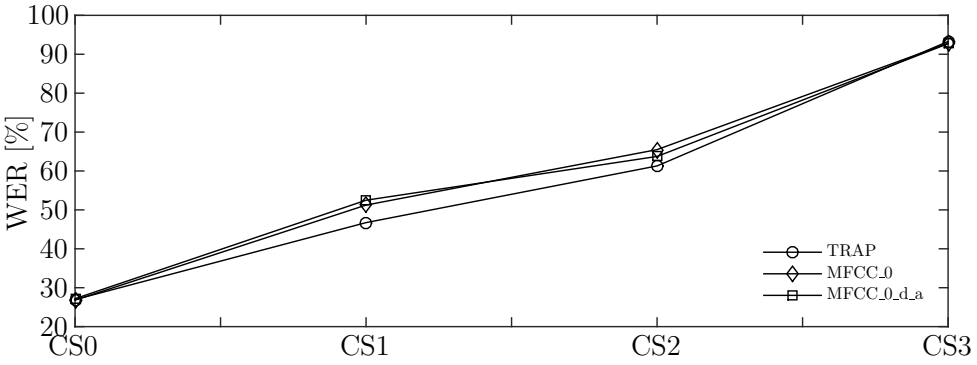
kanál	CS0	CS1	CS2	CS3
WER DEV [%]	31.59	64.67	63.55	95.84
WER TEST [%]	27.18	52.49	63.74	92.71

Tabulka 5.3.4 Tabulka výsledků příznaků TANDEM architektury složené z pravděpodobnostních příznaků z MLP sítě s MFCC_0_d_a příznaky.

Zhodnocení robustnosti

Graf 5.3.1 ukazuje úspěšnosti rozpoznávání všech tří typů příznakových vektorů na TEST množině. Jak je vidět, celkové nejnižšího WER dosahuje příznakový vektor vycházející z TRAPs příznaků. To je velmi pravděpodobně způsobeno rozsáhlým kontextem,

který zvýší celkovou robustnost příznakového vektoru. Příznakový vektor MFCC_0 ukazuje, že kontext získaný zřetězením je dostačující pro sledování stejného trendu růstu mezi CS1 a CS2 jako v případě TRAPs, ale celková informace je menší a tudíž vede na horší absolutní úspěšnost. Zajímavý je vývoj příznakového vektoru vycházejícího z MFCC_0_d_a. Mezi kanály CS1 a CS2 nekopíruje trend předchozích vektorů a roste o něco pomaleji. To by mohlo ukazovat na zvýšenou míru robustnosti vůči vstupním datům. Tato robustnost je však zastíněna celkovým zhoršením WER způsobeném nejspíš překrýváním dat v příznakovém vektoru.



Obrázek 5.3.1 Úspěšnost rozpoznávání TEST množiny pro typy příznaků s MLP sítí.

5.3.3 8-vrstvá MLP síť

Další experimenty proběhly na ANN s 8 vrstvami. Počet skrytých vrstev byl nastaven na 6 s 2048 neurony v každé z nich. Trénování těchto sítí není triviální záležitostí a zabralo velké množství času. Složitější a delší trénování je zpravidla vyváženo větší úspěšností.

TRAP

Kvůli vysoké vstupní dimenzi lze předpokládat, že bude TRAPs příznaky obtížnější namapovat na menší dimenzi příznakového vektoru. Vícevrstvá ANN má díky své složitější struktuře lepší předpoklady k úspěšnějšímu namapování než třívrstvá. Jak je vidět na dosažených úspěšnostech, tento předpoklad je správný. Výsledky na trifonovém rozpoznávači s příznakovým vektorem složeným z MFCC_0_d_a a aposteriorních pravděpodobností vypočtených ANN sítí z TRAPs příznaků ukazuje tabulka 5.3.5.

kanál	CS0	CS1	CS2	CS3
WER DEV [%]	30.50	51.86	54.10	95.02
WER TEST [%]	27.45	40.97	54.45	91.49

Tabulka 5.3.5 Tabulka výsledků příznaků TANDEM architektury složené z pravděpodobnostních příznaků z 8-vrstvé ANN s TRAP příznaky.

MFCC_0

Výsledky na trifonovém rozpoznávači s příznakovým vektorem složeným z MFCC_0_d_a a aposteriorních pravděpodobností vypočtených 8-vrstnou MLP sítí zřetězených MFCC_0 příznakových vektorů ukazuje, tabulka 5.3.6.

kanál	CS0	CS1	CS2	CS3
WER DEV [%]	30.40	60.07	62.00	95.88
WER TEST [%]	25.30	51.47	62.74	91.06

Tabulka 5.3.6 Tabulka výsledků příznaků TANDEM architektury složené z pravděpodobnostních příznaků z 8-vrstvé ANN s MFCC_0 příznaky.

MFCC_0_D_A

Výsledky na trifonovém rozpoznávači s příznakovým vektorem složeným z MFCC_0_d_a a aposteriorních pravděpodobností vypočtených 8-vrstnou MLP sítí zřetězených MFCC_0_d_a příznakových vektorů, ukazuje tabulka 5.3.7.

kanál	CS0	CS1	CS2	CS3
WER DEV [%]	30.26	59.55	59.86	96.17
WER TEST [%]	26.57	50.89	60.96	91.26

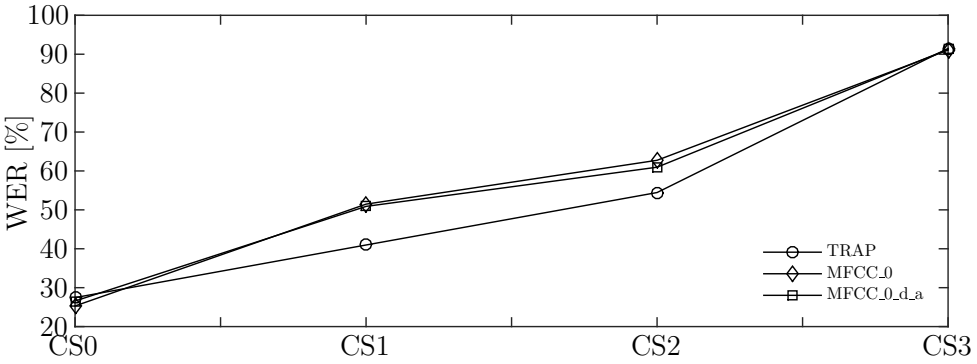
Tabulka 5.3.7 Tabulka výsledků příznaků TANDEM architektury složené z pravděpodobnostních příznaků z 8-vrstvé ANN s MFCC_0_d_a příznaky.

Zhodnocení robustnosti

Graf 5.3.2 ukazuje úspěšnosti rozpoznávání všech tří typů příznakových vektorů na TEST množině. Jak je vidět, celkové WER jsou menší a to díky mapujícím schopnostem 8-vrstvé ANN. Vývoj úspěšností příznakového vektoru založeného na TRAP ukazuje, že s vhodnou sítí, schopnou složitějšího mapování, má tento typ příznakového vektoru vysoký potenciál využít velkou kontextovou informaci.

5.3.4 Jiná prostředí

Robustnost příznakového vektoru a rozpoznávače je možné otestovat i v zásadně odlišných prostředích. Pro tento účel byly použity nahrávky z automobilu s vypnutým a zapnutým motorem, otevřeného veřejného prostranství a uzavřeného veřejného prostranství. Veřejné prostranství je specifické přítomností dalších osob, jejich hovorem a jiným hlukem, který se v případě uzavřeného prostoru navíc odráží. Tyto vlivy ovlivňují míru rušení v signálu. V případě nahrávek z automobilu, tedy malého uzavřeného prostoru,



Obrázek 5.3.2 Úspěšnost rozpoznávání TEST množiny pro typy příznaků s 8-vrstvou MLP sítí.

mohou v signálu být odražené dozvuky samotného řečníka, zvuk motoru, či hluk okolního provozu. Pro otestování úspěšností z těchto prostředí byl zvolen příznakový vektor kombinovaný z MFCC_0_D_A a aposteriorních pravděpodobností získaných z TRAPs pomocí 8-vrstvé ANN. Výsledky ukazuje tabulka 5.3.8.

prostředí	CS0	CS1	CS2	CS3
CAR_OFF TEST [%]	34.95	41.83	39.18	40.86
CAR_ON TEST[%]	39.57	46.17	45.85	45.99
PUBLIC_OPEN TEST [%]	36.47	48.85	61.25	77.54
PUBLIC_HALL TEST [%]	35.53	51.63	86.35	95.77

Tabulka 5.3.8 Tabulka výsledků pro další prostředí s TANDEM architekturou.

Pro porovnání výsledků byla tato prostředí otestována i na netandemovém GMM-HMM rozpoznávači. Výsledky ukazuje tabulka 5.3.9.

prostředí	CS0	CS1	CS2	CS3
CAR_OFF TEST [%]	40.47	47.96	45.72	47.38
CAR_ON TEST[%]	41.73	49.75	48.70	52.31
PUBLIC_OPEN TEST [%]	40.42	54.32	70.56	85.20
PUBLIC_HALL TEST [%]	38.22	60.04	89.54	96.13

Tabulka 5.3.9 Tabulka výsledků pro další prostředí bez TANDEM architektury.

Jak je z obou tabulek vidět, TANDEM systém ve všech případech dosáhl lepších výsledků. V některých případech je úspěšnost vyšší skoro o deset procent. Tyto výsledky jasně ukazují, že optimalizovaný TANDEM systém má vyšší robustnost i úspěšnost než srovnatelně optimalizovaný netandemový systém.

5.3.5 Celkové zhodnocení

Tabulka 5.3.10 obsahuje všechny napočítané hodnoty, aby je bylo možné lépe porovnat.

	CS0	CS1	CS2	CS3
GMM-HMM bez TANDEMu DEV [%]	31.86	44.98	49.12	65.97
GMM-HMM bez TANDEMu TEST [%]	28.52	36.67	44.86	58.47
MLP3 TRAP DEV [%]	30.93	61.76	65.28	96.47
MLP3 TRAP WER TEST [%]	26.80	51.24	65.49	92.83
MLP3 MFCC_0 DEV [%]	30.93	61.76	65.28	96.47
MLP3 MFCC_0 TEST [%]	26.80	51.24	65.49	92.83
MLP3 MFCC_0_D_A DEV [%]	31.59	64.67	63.55	95.84
MLP3 MFCC_0_D_A TEST [%]	27.18	52.49	63.74	92.71
MLP8 TRAP DEV [%]	30.50	51.86	54.10	95.02
MLP8 TRAP TEST [%]	27.45	40.97	54.45	91.49
MLP8 MFCC_0 DEV [%]	30.40	60.07	62.00	95.88
MLP8 MFCC_0 TEST [%]	25.30	51.47	62.74	91.06
MLP8 MFCC_0_D_A DEV [%]	30.26	59.55	59.86	96.17
MLP8 MFCC_0_D_A TEST [%]	26.57	50.89	60.96	91.26
CAR_OFF bez TANDEMu TEST [%]	40.47	47.96	45.72	47.38
CAR_ON bez TANDEMu TEST[%]	41.73	49.75	48.70	52.31
PUBLIC_OPEN bez TANDEMu TEST [%]	40.42	54.32	70.56	85.20
PUBLIC_HALL bez TANDEMu TEST [%]	38.22	60.04	89.54	96.13
CAR_OFF TEST [%]	34.95	41.83	39.18	40.86
CAR_ON TEST[%]	39.57	46.17	45.85	45.99
PUBLIC_OPEN TEST [%]	36.47	48.85	61.25	77.54
PUBLIC_HALL TEST [%]	35.53	51.63	86.35	95.77

Tabulka 5.3.10 Tabulka všech výsledků.

6 Závěr

V této práci byla realizována implementace rozpoznávače na bázi TANDEM architektury. K implementaci byly použity volně dostupné nástroje Kaldi a CtuCopy.

V souvislosti s touto prací vzniklo několik skupin skriptů s implementací různých částí úlohy. Tyto skripty jsou určeny pro práci s databází SPEECON a umožňují řešit tyto podúlohy: Trénování umělých neuronových sítí pro potřeby TANDEM architektury. Trénování rozpoznávače s TANDEM architekturou na specifickém kanálu databáze. Testování rozpoznávače na libovolném kanále. Implementaci standardního GMM-HMM rozpoznávače bez TANDEM architektury, určeného k výpočtu referenčních výsledků a generování kvalitních zarovnaných dat pro trénování umělých neuronových sítí. Tyto recepty vycházejí z Kaldi receptů a dodržují jejich konvenci.

Dílčím výsledkem práce je vytvoření funkčního programového a datového zázemí v Národní Gridové Struktuře Metacentrum pro vývojovou skupinu CTU Speech Lab. Tato platforma poskytuje skupině potřebné nástroje a databáze pro realizaci výpočetně náročných experimentů souvisejících s výzkumem v oblasti rozpoznávání řeči. Pro účely skupiny byla na Metacentru založena skupina CTU Speech Lab se sdíleným pracovním prostorem. V tomto pracovním prostoru se nacházejí databáze signálů a zkompilované nástroje Kaldi, CtuCopy a další podpůrné nástroje, které na Metacentru nebyly k dispozici a tedy musely být doinstalovány.

Experimentální část tvoří v podstatě tři části. V první části bylo hledáno optimální nastavení rozpoznávače pro TANDEM architekturu. Nastavované parametry byly dimenze výsledného příznakového vektoru po PCA, celkový počet Gaussovských komponent v GMM modelu a počet trénovacích kroků. Hodnoty shledané jako optimální jsou: Dimenze po PCA 60, celkový počet komponent 80000, při 55 iteracích.

V druhé části experimentů byl hledán optimální příznakový vektor TANDEM architektury. Toto hledání bylo provedeno na datech z prostředí OFFICE, které bylo považováno za nejstabilnější. Výsledky rozpoznávání však ukázaly, že naopak mělo největší rozdíly v kvalitě dat mezi jednotlivými kanály. Zatímco chybovost klasifikace WER, se na CS0 pohybovala kolem 30 %, na CS3 dosahovala víc než 90 %. Přesto se z experimentů ukázal jako jasně nejvhodnější pro další použití, příznakový vektor spojený z MFCC_0_D_A a TRAPs příznaků mapovaných pomocí 8-vrstvé neuronové sítě.

V třetí části experimentů byl tento příznakový vektor použit při testech robustnosti na

ostatních prostředích dostupných v databázi. Ve všech testech tento TANDEM systém dosáhl vyšší úspěšnosti než systém bez TANDEMu. Například v prostředí CAR_ON, dosáhl TANDEM systém WER 39.57 % pro CS0, 46.18 % pro CS1, 45.85 % pro CS2 a 45.99 % pro CS3. Naproti tomu standardní GMM-HMM systém bez TANDEM architektury dosáhl hodnot 41.73 %, 49.75 %, 48.70% a 52.31 %.

Tato práce dala vzniknout několika skriptům, které jsou relativně snadno rozšiřitelné a poskytují modulární základnu pro další experimenty. Jako případné další úkoly navazující na tuto práci by mohlo být použití hlubokých neuronových sítí s RBM předtrénováním. Další možností je rozšířit DNN o další vrstvy a vytvořit bottleneck síť.

Literatura

[1] Josef Psutka et al. *Mluvíme s počítačem česky*. Academia, 2006.

[2] Herve A. Bourlard a Nelson Morgan. *Connectionist Speech Recognition: A Hybrid Approach*. Norwell, MA, USA: Kluwer Academic Publishers, 1993. ISBN: 0792393961.

[3] Geoffrey Hinton et al. "Deep Neural Networks for Acoustic Modeling in Speech Recognition". In: *IEEE Signal Processing Magazine* 29.6 (lis. 2012), s. 82–97. URL: http://research.microsoft.com/apps/pubs/default.aspx?id=171498.

[4] Geoffrey Hinton et al. "Deep Neural Networks for Acoustic Modeling in Speech Recognition". In: *Signal Processing Magazine* (2012).

[5] Daniel Povey et al. "The Kaldi Speech Recognition Toolkit". In: *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Catalog No.: CFP11SRW-USB. Hilton Waikoloa Village, Big Island, Hawaii, US: IEEE Signal Processing Society, pros. 2011.

[6] CESNET. *MetaCentrum NGI*. URL: http://metavo.metacentrum.cz/.

[7] Steve Young. *Hidden Marcov Model Toolkit*. 1989. URL: http://htk.eng.cam.ac.uk/.

[8] *Quicknet*. URL: http://www1.icsi.berkeley.edu/Speech/icsi-speech-tools.html.

[9] *Open-Source Large Vocabulary CSR Engine Julius*. URL: http://julius.osdn.jp/en_index.php.

[10] *CMU Sphinx*. URL: http://cmusphinx.sourceforge.net/.

[11] Matteo Frigo a Steven G. Johnson. "The Design and Implementation of FFTW3". In: *Proceedings of the IEEE* 93.2 (2005). Special issue on "Program Generation, Optimization, and Platform Adaptation", s. 216–231.

[12] Jan Uhlíř et al. *Technologie hlasových komunikací*. Praha: Nakladatelství ČVUT, 2007.

[13] Partha Lal a Simon King. "Cross-Lingual Automatic Speech Recognition Using Tandem Features". In: *Audio, Speech, and Language Processing, IEEE Transactions on* (2013), s. 2506–2515.

[14] Daniel Ellis, Rita Singh a Sunil Sivadas. "Tandem Acoustic Modeling In Large-Vocabulary Recognition". In: *in Proc. ICASSP-2001*. 2001, s. 517–520.

[15] Qifeng Zhu et al. "Tandem Connectionist Feature Extraction for Conversational Speech Recognition". In: *Proceedings of the First International Conference on Machine Learning for Multimodal Interaction*. Martigny, Switzerland, 2005.

[16] Daniel P. W. Ellis a Manuel J. Reyes Gomez. *Investigations into Tandem Acoustic Modeling for the Aurora Task*. 2001.

[17] *SPEECON*. URL: `http://speech.fit.vutbr.cz/cs/projects/speecon`.

[18] Xian Tang. "Hybrid Hidden Markov Model and Artificial Neural Network for Automatic Speech Recognition". In: *Circuits, Communications and Systems, 2009. PACCS '09. Pacific-Asia Conference on*. 2009, s. 682–685.

[19] Amlan Kundu a Aruna Bayya. "Speech recognition using hybrid hidden markov model and NN classifier". In: *International Journal of Speech Technology* (1998), s. 227–240.

[20] Rongfeng Su et al. "Efficient Use of DNN Bottleneck Features in Generalized Variable Parameter HMMs for Noise Robust Speech Recognition". In: *Sixteenth Annual Conference of the International Speech Communication Association*. 2015.

[21] Xuedong Huang, Alex Acero a Hsiao-Wuen Hon. *Spoken Language Processing: A Guide to Theory, Algorithm, and System Development*. 1st. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2001. ISBN: 0130226165.

[22] Léon Bottou. "Stochastic Learning". In: *Advanced Lectures on Machine Learning*. Ed. Olivier Bousquet a Ulrike von Luxburg. Lecture Notes in Artificial Intelligence, LNAI 3176. Berlin: Springer Verlag, 2004, s. 146–168. URL: `http://leon.bottou.org/papers/bottou-mlss-2004`.