Univerzita Komenského v Bratislave
Fakulta Matematiky, Fyziky a Informatiky
Katedra Aplikovanej Informatiky

Multimediálny šlabikár

Systém na rozpoznávanie prehovorených písmen slovenskej abecedy

(diplomová práca)

Martin Zlatý

Školiteľ: RNDr. Marek Nagy

Bratislava, 2007

Čestne prehlasujem, že som túto diplomovú prácu vypracoval samostatne s použitím uvedenej literatúry.

V Bratislave, 7.Mája 2007 .....

Martin Zlatý

Poďakovanie

Chcel by som poďakovať najmä mojmu diplomovému vedúcemu Marekovi Nagyovi. Bez jeho pomoci a dobrých rád by táto práca nemohla vzniknúť. Ďalej by som chcel poďakovať vedeniu Základnej školy Černyševského v Branislave, ktoré mi umožnilo nahrávanie vzoriek. Rovnako by som chcel poďakovať aj ďeťom, ktoré sa nahrávania zúčastnili. Poďakovanie patrí aj mojim rodičom, vďaka ktorým som mohol študovať.

Obsah

1 Úvod **1**

1.1 Cieľ diplomovej práce 2

1.2 Členenie diplomovej práce 3

2 Prehľad problematiky **4**

2.1 Automatický rozpoznávač izolovaných slov 4

2.1.1 Rozpoznávanie hraníc slov 5

2.1.2 Sekundárne spracovanie signálu 6

2.1.3 Klasifikátor 7

2.2 Fonetika slovenských hlások 12

2.2.1 Samohláskové zvuky 12

2.2.2 Spoluhláskové zvuky 13

3 Databáza **16**

3.1 Príprava na nahrávanie 17

3.2 Aplikácia 17

3.3 Nahrávanie 18

3.4 Výsledky nahrávania 18

4 Návrh **20**

4.1 Teoretický návrh 20

4.2 Návrh systému 23

4.3 Subklasifikátory 24

4.3.1 Neurónová sieť 25

4.3.2 DTW subklasifikátor 26

5 Trénovanie a testovanie subklasifikátorov **28**

5.1 Neurónová sieť 28

5.1.1 Trénovanie neurónovej siete 28
5.1.2 Normalizácia 29
5.1.3 Výsledky 30
5.2 DTW subklasifikátor 36
5.2.1 Testovanie 36
5.2.2 Výsledky 36

6 Klasifikátor **38**
6.1 Popis vyhodnocovacích algoritmov 38
6.2 Testovanie 39
6.3 Výsledky 39

7 Záver **43**

A Ďalšie výsledky **46**

B Obsah CD **56**
B.1 Abeceda 56
B.2 Databáza 56
B.3 Decider 56
B.4 Dokumenty 57
B.5 TestingApp 57
B.6 WavToMFCC 57

Kapitola 1

Úvod

Ako asi bude vyzerať počítač budúcnosti? Neviem. Možno bude stále kremíkový, kvantový alebo biologický. Určite bude oveľa rýchlejší ako dnešné počítače. A čo by mal dokázať? Asi by mal poskytovať lepší spôsob, ako doň vkladať informácie, ako pomocou klávesnice alebo myšky. Mal by dokázať s užívateľom plnohodnotne komunikovať - porozumieť jeho požiadavkám, spracovať ich a poskytnúť výsledok v očakávanej podobe, teda napríklad odpovedať rečou.

Už v dnešnej dobe, by mal softvér, ktorý dokáže dobre rozpoznávať plynulú reč, veľmi veľké uplatnenie v mnohých oblastiach. Napriek pomerne veľkému úsiliu viacerých firiem a mnohých univerzít stále takýto softvér neexistuje. Dôvodov je veľa. Asi tie najpodstatnejšie sú nekvalitný vstup s množstvom šumu v pozadí, obrovská redundancia vstupných dát, rôzni rečníci s rôznymi prízvukmi a neexistencia páuz medzi slovami. Keď nevieme vyriešiť komplexný problém, skúsme si ho rozdeliť na jednoduchšie podproblémy, tie vyriešiť a potom tieto riešenia spojiť. Jedným z možných zjednodušení je obmedziť sa na rozpoznávanie izolovaných slov.

Keď počujeme cudzie slovo alebo meno, ktorému nerozumieme (nikdy predtým sme ho nepočuli) a potrebujeme ho napísať, požiadame rečníka, aby nám ho vyhláskoval. Je to potrebné kvôli nedokonalému spôsobu zápisu vyslovených zvukov, keď viacero rôznych zápisov odpovedá rovnakej výslovnosti. Ak by existoval systém na rozpoznávanie plynulej reči, asi najlepšie by malo fungovať rozpoznávanie hlások danej abecedy, lebo pri každom zlom rozpoznaní je nutné slovo vyhláskovať [1]. Preto som sa vo svojej diplomovej práci

[1] Akosi implicitne predpokladám, že pod rozponaním slov sa vlastne myslí nájdenie zobrazenia zo zvukového signálu na zápis v abecede, aj keď to nutne nemusí byť tak.

rozhodol zaoberať rozpoznávaním prehovorených písmen slovenskej abecedy.

1.1 Cieľ diplomovej práce

Cieľom mojej diplomovej práce je navrhnúť a implementovať klasifikátor, ktorý by bol schopný rozpoznávať písmená slovenskej abecedy, ktoré sú totožné s hláskami. To znamená, že sa nechcem zaoberať písmenami ako "ä" alebo "w" ("dvojité v"), ale ani hláskami, ktoré sa nevyskytujú pri hláskovaní, ale len pri plynulej reči. "Ä" síce môže byť aj samostatnou hláskou, ale väčšinou sa miesto neho používa normálne "e" alebo pri hláskovaní "široké e". Navyše by systém mal byť schopný pracovať v reálnom čase, čo znamená, že po zadaní vstupu by mal rozhodnúť do 0.3 až 0.5 sekundy. Návrh a implementácia by mali byť také, aby sa dali využiť aj v ďalších prácach, bez nutnosti začínať opäť úplne od začiatku. Výsledný produkt by mohol byť použitý buď ako samostatný rozponávač alebo ako podporný systém pre rozponávanie plynulej reči.

Hlavný problém, ktorý musím riešiť je fakt, že spoluhlásky ako napríklad "k" a "g" alebo "m" a "n" sa na seba veľmi podobajú a sú veľmi krátke. To znamená, že zvuk, ktorý je diferenčným príznakom (rozhoduje či je to "m" alebo "n") je veľmi krátky. Na druhej strane, krátky vstup má tú výhodu, že môžem použiť aj výpočtovo a časovo náročnejšie algoritmy a pritom sa stále stihnúť odozvu v reálnom čase.

Ďalší problémom je neexistencia databázy vzoriek, s ktorou by som mohol pracovať. Aj keď vo svete existujú, takéto voľne dostupné databázy, nič podobné zatiaľ v slovenskom jazyku nevzniklo. Existuje komerčná databáza slovenských vzoriek SpeechDat(E) [16], ktorá obsahuje len celé slová navyše v telefónnej kvalite, a preto je pre moje potreby nevyhovujúca. Keďže môj školiteľ má dobré skúsenosti s deťmi, rozhodol som sa nahrať vzorky na základnej škole, spracovať ich a vytvoriť tak základ takejto databázy. Na nej by som chcel trénovať a testovať úspešnosť môjho rozpoznávacieho softvéru. Takto natrénovaný rozpoznávač by mal byť základom multimediálneho šlabikára - systému, ktorý by pomáhal deťom učiť sa správne a čisto hláskovať.

1.2 Členenie diplomovej práce

Druhá kapitola je rozdelená na dve časti. V prvej oboznamujem čitateľa so základnými pojmami z oblasti rozpoznávania reči, v druhej časti sú to základy fonetiky a fonológie. V tretej kapitole je popísaná tvorba databázy hlások. Vo štvrtej kapitole predstavím návrh. V piatej popíšem trénovanie a testovanie subklasifikátorov. V šiestej potom spojím natrénované subklasifikátory do finálneho klasifikátora a nakoniec vyvodím záver.

Kapitola 2

Prehľad problematiky

2.1 Automatický rozpoznávač izolovaných slov

Štandardná architektúra automatického rozpoznávača izolovaných slov je zobrazená na obrázku 2.1. Vstupom je signál, buď z mikrofónu alebo zo súboru so zvukovým obsahom. Primárne spracovanie signálu sa snaží jednoduchými úpravami vstupu, zlepšiť úspešnosť zisťovania hraníc slov[1]. Sekundárne spracovanie signálu sa snaží odstrániť nepotrebné informácie zo vstupného signálu (napríklad farba hlasu rečníka), ale pritom zachovať informáciu nutnú pre úspešné rozpoznanie. Klasifikátor zodpovedá za samotné rozponanie, teda priradenie k vstupnému signálu triedu do ktorej patrí.

Obrázok 2.1: Architektúra automatického rozponávača izolovaných slov

[1] anglicky End-point detecion

Vo väčšine prác odpadá problém vytvárania databázy vzorov, lebo existuje niekoľko databáz anglických slov na internete. Tie obsahujú slová a prislúchajúci signál a teda netreba zisťovať začiatok a koniec slova.

2.1.1 Rozpoznávanie hraníc slov

Cieľom zistovania hraníc slov je rozdelenie vstupného signálu na reč a šum (nerečový signál). Presné určenie hraníc slov je veľmi dôležité, lebo chyba ktorá vznikne sa ďalej šíri a teda zvyšuje nepresnosť finálneho rozpoznania. Zvyčajne sa snažíme neurčovať príliš presné hranice, lebo sa nám môže stať, že odsekneme podstatnú časť slova. Pri množstve spoluhlások rozhoduje začiatok, a preto pri vynechaní tejto časti stratím podstatnú informáciu. Preto je lepšie pridať nejakú časť signálu pred detekovaný začiatok slova a za detekovaný koniec slova. Samozrejme príliš dlhé pridané časti zbytočne zväčujú vstup a tým spomaľujú výpočet, čo je podstatné pri aplikáciách, ktoré majú pracovať v reálnom čase.

Pri plynulej reči je takéto určovanie hraníc nemožné, lebo neexistujú. Často vyslovujeme slová a predložku bez pauzy medzi nimi, prípadne sa celé slová zlievajú do jedného. Tieto hranice možno určiť napríklad pri nádychoch, koncoch viet alebo čiarkách. Pravdepodobne až po uspokojivom vyriešení problému rozpoznávania izolovaných slov, budeme schopný riešiť aj problém rozpoznávania plynulej reči.

Najbežnejší spôsob detekcie hraníc slov je pomocou funkcie krátkodobej energie signálu a krátkodobej funkcie stredného počtu priechodov nulov [5], lebo sú výpočtovo nenáročné a pomerne spoľahlivé [6]. Vstupný signál, ako potentciálne nekonečná postupnosť, sa spracúva po častich (oknách), na ktorých sa určia hodnoty funkcií. Je nutné určiť hraničný prah, podľa ktorého sa bude rozdelovať reč od nerečovej časti signálu a počet okien rozponaných ako reč (šum), ktoré musia nasledovať po sebe, aby sme danú časť vyhlásili za začiatok slova (koniec slova). Navyše treba určiť pomer v akom budeme brať výsledky oboch funkcií.

Jedna z novších metód [6] [7] je pomocou výpočtu Teager Energy a Entropy-Energy Features. Teager Energy je vlastne iný spôsob výpočtu energie signálu, založený nielen na intenzite ale aj frekvencii signálu. Na jej výpočet ale treba vypočítať diskrétnu Furierovu transformáciu(DFT) [5] a preto je pomalšia. V [6] bol použitý na približné určenie hraníc, na presné určenie sa využíva Entropy-

Energy Features. Lepšie výsledky dosahuje najmä v zašumených prostrediach.

2.1.2 Sekundárne spracovanie signálu

Pri bežnom rozhovore vysloví človek asi 80-130 slov za minútu, čo predstavuje frekvenciu výskytu asi 10 foném za sekundu. Ak uvážime, že priemerná informácia na jednu fonému je 3-4 bity, dostaneme pre súvislú reč prenosovú rýchlosť 30-40 bit/s. Tento odhad však zanedbáva ďalšie informácie ako napríklad intonácia, tempo reči, farba hlasu alebo dialekt. Ak by sme chceli presne zapísať rečový signál, potrebovali by sme informačnú rýchlosť aspoň 200000 bit/s [5]. Z dôvodov obrovskej informačej redundancie využíva človek pri vnímaní významu interné mechanizmy, ktoré potláčajú v rečovom signále nepotrebné údaje a zdôrazňujú základnú informáciu, ktorá je zhodná pre rovnaké slová. Sekundárne spracovanie signálu má teda za úlohu čo možno najviac znížiť informačný obsah a zároveň zachovať a zdôrazniť informácie, potrebné na rozpoznanie slova.

Sekundárne spracovanie signálu je zvyčajne založené na diskrétnej Fourierovej transfromácii (DFT), diskrétnej kosínusovej transformácii (DCT) a prípadne lineárnej prediktívnej analýze (LPC).V [9] Davis porovnával viacero typov spracovania: Mel Frequency Cepstrum Coefficients (MFCC), Linear Frequency Cepstrum Coefficients (LFCC), Linear Prediction Coefficients(LPC), Linear Prediction Cepstrum Coefficents (LPCC) a Reflection Coefficients (RC). Hlavným záverom tejto práce bolo, že MFCC sú najlepšie pre automatický rozpoznávač reči. Aj keď táto práca vznikla v roku 1980, MFCC s malými obmenami je stále asi najbežnejšie používaný spôsob extrakcie príznakov. Využíva DFT, DCT a Mel scale – logaritmickú stupnicú, ktorá odpovedá tomu, že človek lepšie rozlišuje nižšie ako vyššie frekvencie. Najväčší problém je určiť správne parametre pre danú metódu, lebo rôzne nastavenia môžu dávať výrazne rôzne výsledky. Zatiaľ najúspešnejší (mne známy) rozpoznávač izolovaných hlások používal 12 MFCC koeficientov, 12 delta koeficientov, energia a energia delta koeficientov. Používal ISOLET databázu[17] a pracoval s úspešnosťou 97.37%[10].

Iná metóda popísaná v [3] a [11] využíva Discret Cosine Series Coefficients (DCSC). Je podobná ako MFCC spolu s výpočtom delta MFCC, lepšie sa však analyzuje kvôli nižšiemu počtu parametrov. Autori však neuvádzajú výrazne lepšie výsledky, ako tie, ktoré dosiahnuté s MFCC koeficientami.

2.1.3 Klasifikátor

Úloha klasifikátora je rozpoznať vektor príznakov alebo postuponosti takýchto vektorov. Rozpoznanie v tomto prípade znamená zaradenie vstupu do jednej z predom definovanej triedy. Pri systéme rozpoznávania reči je väčšinou najdôležitejšie aby pracoval v reálnom čase. To znamená, že klasifikátor by sa mal rozhodnúť asi 0.5 - 0.7 sekundy po vyslovení slova. Napriek zävčšujúcej sa výpočtovej sile počítačov je stále najdôležitejšia rýchosť a efektívnosť klasifikačného algoritmu. Klasifikátor je zvyčajne založený na jednej (alebo viacerých) z nasledujúcich metód.

Dynamic time warping (DTW)

Prvé klasifikátory zaraďovali vstupné slovo do tej triedy, ku ktorej referenčnému vzoru mali najmenšiu vzdialenosť. Táto vzdialenosť sa počítala pomocou algoritmu DTW, alebo jeho modifikácie (bližšie popísané v [5]). Veľkým problémom bolo určiť referenčné vzory pre jednotlivé triedy, teda nejaký priemerný vektor, ktorý vystihuje celú triedu. Preto dosahujú lepšie výsledky metódy, ktoré sa samé "naučia" rozpoznávať triedu len z trénovacej množiny. Každopádne však tento prístup poskytuje pomerne rýchly a jednoduchý spôsob klasifikácie, schopnej pracovať v reálnom čase.

Skryté Markovove modely (HMM)

Dnes asi najpoužívanejšia metóda je založená na skrytých Markovovych modeloch. Prvý krát bola použitá už na začiatku sedemdesiatych rokov, populárna sa stala v polovici osemdesiatych rokov, po niekoľkých úspešných aplikáciách. Princíp tejto metódy vychádza z predstavy o vytváraní reči. Môžme si predstaviť, že pri generovaní reči človekom sú hlasové ústroje počas nejakého krátkeho časového intervalu (mikrosegment) v nejakom stave. Tento stav je jeden z konečného počtu artikulačních konfigurácii (napríklad generuje nejakú fonému). V uvažovanom mikrosegmente je potom hlasovými ústrojmi generovaný krátky signál, ktorý závisí od stavu v ktorom boli hlasové ústroje.

Matematicky možmo Markovov model G vyjadriť päticou

$$G = (Q, V, N, M, \pi) \tag{2.1}$$

kde :

- $Q = \{q_1, \ldots, q_N\}$ je množina stavov Markovovho modelu
- $V = \{v_1, \ldots, v_L\}$ je abeceda L výstupných symbolov
- $N = [n_{ij}]$ je matica prechodov medzi stavmi. Určuje pravdepodbnosť s akou v ľubovolnom čase t prechádza systém zo stavu q_i do stavu q_j v čase $t + 1$.
- $M = [m_{il}] = [m_j(l)]$ je matica pravdpodobností vygenerovania vzoru l v stave q_i
- $\pi = [\pi_i]$ určuje pravdepodnosť začiatku v stave q_i

Ak rozpoznávame medzi N triedami, pre každú triedu vytvoríme jeden Markovov model. Pre vstupnú postupnosť O určíme do ktorej triedy patrí tak, že pre všetky modely vypočítame pravdepobnosť, s akou by ju model vygeneroval a vyberieme z nich maximálnu. Táto aproximácia pravdepodobnosti sa počíta pomocou modifikovaného Virterbi algoritmu(bližšie v [5]):

1.Inicializácia

$$\phi_1(i) = \log \pi_i + \log m_i(o_i) \tag{2.2}$$

pre $i = 1, \ldots, N$

2.Rekurzia pre $t = 2, 3, \ldots, T$ a $j = 1, \ldots, N$

$$\phi_t(i) = \max_j [\phi_{t-1}(i) + \log n_{ij}] + \log m_j(o_t) \tag{2.3}$$

kde $i = 1, \ldots, N$

3. Výsledná prevdepodobnosť

$$\log P^* = \max_i [\phi_T(i)] \tag{2.4}$$

kde $i = 1, \ldots, N$

Pri pôvodnom algoritme hrozilo podtečenie premenných, lebo sa násobili veľmi malé čísla. Preto sa v modifikovanom algoritme počíta logartimus pravdepodonosti - využíva sa vlastnosť $\log x.y = \log x + \log y$.

Neurónové siete

Perceptrón (percept = vnem) je model neurónu, ktorý prijíma vstupné signály $\overrightarrow{x} = (x_1, x_2, ...x_n)$ cez synaptické váhy tvoriace váhový vektor $\overrightarrow{w} = (w_1, w_2, ...w_n)$. Vstupný vektor $\overrightarrow{x}$ sa nazýva vzor alebo obrazec (angl. pattern). Zložky vstupného vektora môžu nadobúdať reálne alebo binárne hodnoty. Výstup perceptrónu o je daný vzťahom

$$o = f(net) = f(\overrightarrow{x}.\overrightarrow{w}) = f\Big(\sum_{j=1}^{n} w_j x_j\Big) \tag{2.5}$$

kde f je aktivačná funkcia. Najčastejšie sa používajú spojité aktivačné funkcie, ktoré majú sigmoidálny priebeh.

$$\text{Unipolárna sigmoida:} \qquad f(net) = \frac{1}{1 + e^{-\lambda net}} \tag{2.6}$$

$$\text{Bipolárna sigmoida:} \qquad f(net) = \frac{2}{1 + e^{-\lambda net}} - 1 \tag{2.7}$$

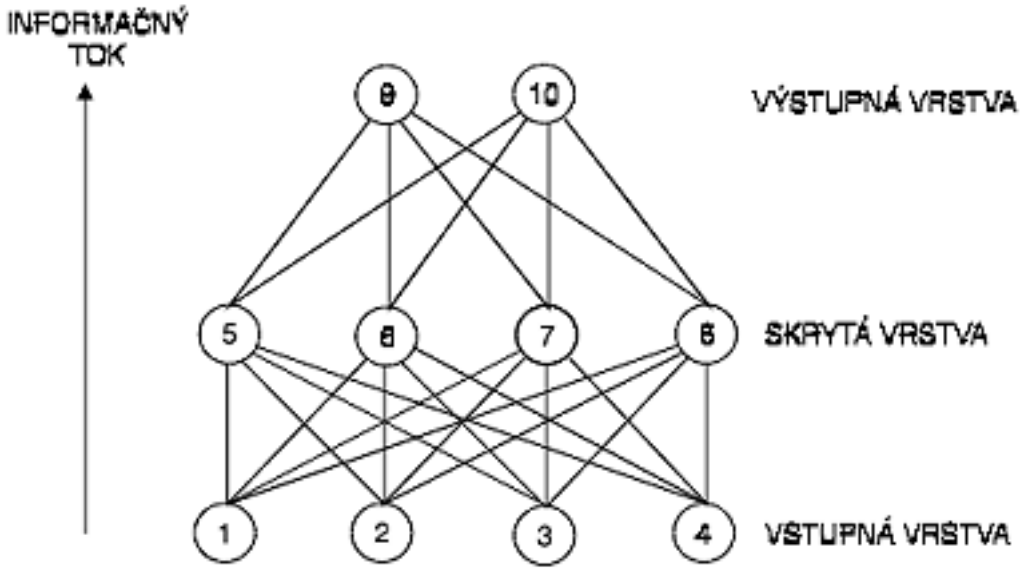
Konštanta $\lambda > 0$ sa nazýva strmosť sigmoidy. Zvyčajne sa používa $\lambda = 1$. V limite pre $\lambda \to \infty$ bipolárna sigmoida prejde na funkciu signum a unipolárna sigmoida na krokovú funkciu. Bolo dokázané, že jeden neurón vie riešiť len problémy, ktoré sú lineárne separovateľné[12].

Keď usporiadame perceptróny do vrstiev a prepojíme, dostávame doprednú umelú neurónovú sieť (Obrázok 2.2). Bolo dokázané, že dopredná neurónová sieť s jednou skrytou vrstvou je schopná aproximovať ľubovolnú spojitú funkciu. Ak máme množinu vektorov (napríklad MFCC koeficienty) a ich funkčné hodnoty (trieda ktorej prislúchajú), rozdelíme ju na dve množiny - trénovaciu A_{train} a testovaciu A_{test}. Samozrejme musí platiť, že A_{train} a A_{test} sú disjunktné množiny. Optimálny pomer rozdelenia je 70% ku 30%. Na trénovacej množine sieť natrénujeme, zvyčajne pomocou metódy spätného šírenia chýb[2], ktorá sa snaží minimalizovať chybu výstupu (bližšie v [12]) E_p.

$$E_p = \frac{1}{2} \sum_{k=1}^{K} (d_{pk} - o_{pk})^2 \tag{2.8}$$

kde K počet výstupných neurónov siete, o_{pk} je k-ty výstup na p-tej vzorke A_{train} a d_{pk} je očakávaný výstup siete. Súčet takýchto chýb cez všetky vzorky sa volá

[2] anglicky Backpropagation algorithm



Obrázok 2.2: Znázornenie trojvrstvovej neurónovej siete s dopredným šírením

kumulovaná chyba. Trénujeme dovtedy kým bude klesať kumulovaná chyba množiny A_{test}. Akonáhle začne rásť dochádza k preučeniu (overfitting). Za úspešnosť siete sa považuje, úspešnosť rozpoznania vzorov z A_{test}.

Doprednú neurónovú sieť možno trénovať pomocou metódy spätného šírenia chýb aj na klasifikáciu vzorov do K tried, ktoré majú nelineárne deliace hranice. Vtedy má sieť K výstupných neurónov. Napríklad pre 3 triedy, požadované výstupné vektory navrhneme takto: (1,-1,-1) , (-1, 1, -1) a (-1, -1, 1), ak je aktivačná funkcia výstupných neurónov bipolárna sigmoida. Táto metóda sa volá "One-hot", lebo na určenie triedy musí, byť aktívny práve jeden neurón a všetky ostané musia byť neaktívne. Skutočný výstup musíme nejako interpretovať. Zvyčajne určíme maximálnu chybu ε, ktorú akceptujeme. Napríklad ak bude $\varepsilon = 0.3$, tak vektor (0.7, -0.7, -0.7) je stále prvok prvej triedy. Takúto interpretáciu používame pre výpočet počtu chybných výsledkov na A_{train} a A_{test}. Pod chybným výsledkom rozumieme nesprávne klasifikovaný vzor.

Treba spomenúť aj problémy, ktoré treba vyriešiť, ak chceme použiť neurónové siete. Ako pretransformovať reálne vstupy na konštatný počet vstupov neurónovej siete? Ako chápať výsledok? Aké parametre siete použiť? Koľko vstupných

neurónov, koľko skrytých, akú rýchlosť učenia, ... ? Navyše treba určiť jednu z mnohých modifikácií základnej architektrúry neurónovej siete.

Peknú porovnávaciu štúdiu urobili v [4], kde skúmali úspešnosť štyroch typov neurónových sietí - viacvrstvová neurónová sieť (MLP), viacvrstvová neurónová sieť s viacerými výstupými vrstvami (MOLP), Time-Delay Neural Network (TDNN) a Kohonenovu samoorganizujúca sa mapa (SOM). Boli testované na dvoch databázach. Prvá pozostávala zo slov nahovorených v miestosti relatívne bez šumu, druhá bolo databáza nahrávaná cez telefón (oveľa viac šumu). Obe pozostávali z 12 izolovaných slov, nahovorených 25 ľuďmi. Každé slovo bolo parametrizované na postupnosť pätnástich 18-rozmerných príznakových vektorov. Tie sa skladali z 8 MFCC koeficientov, 8 delta MFCC koeficientov a oboch energií. Podľa tejto práce sú tieto štyri techniky schopné konkurovať DTW a HMM. MLP dosahovala dobré výsledky pri bezšumovej databáze (úspešnosť 93,3%), výrazné zhoršenie nastalo pri druhej databáze (76,3%). MOLP a TDNN dosahovali podobne, veľmi dobrú úspešnosť – pri prvej databáze 95.5%, pri druhej 90,%. Víťazom sa stala SOM s úspešnosťami 97,1% a 96,8%. Nevýhoda SOM spočíva v pomerne dlhom čase, ktorý je potrebný na rozpoznanie slova a preto je ťažie použiteľná v aplikáciách pracujúcich v reálnom čase.

Spájanie modelov

Všetky typy klasifikátorov boli úspešne použité vo viacerých aplikáciách, a preto ak by sa nám ich podarilo spojiť so zachovaním predošlích vlastností, výsledný klasifikátor by mohol dosiahnuť lepšie výsledky.

Asi najjednoduchším spôsobom ako spojiť HMM a neurónové siete je jednoducho niektoré časti HMM nahradiť neurónovou sieťou. Aj keď takýto systém nedosahuje vyššiu úspešnosť, je lepšie paralelizovateľný [13].

Zatiaľ najúspešnejší prístup bol predstavený v [14]. V HMM nahrad'uje Radial Basis Function (RBF) sieť viacvrstvovou neurónovou sieťou, ktorá je natrénovaná tak, aby počítala pravdepodbnosť vygenerovania symbolu v stave.

V [3] je popísaná metóda binary-pair partitioning (BPP) a funguje nasledovne: ak máme N klasifikačných tried, prácu jedného klasifikátora rozdelíme na $\binom{N}{2}$ (počet všetkých dvojíc) klasifikačných úloh. Na každú takúto úlohu použijeme samostatný subklasifikátor, v [3] to bola neurónová sieť. Výstup

neurónovej siete sa dá chápať ako odhad pravdepodobnosti a teda stačí určiť tú najpravdepodobnejšiu možnosť. Označme m-tú kategóriu G_m a jej pravdepodobnosť $P_m = P(G_m)$. Ďalej označme $P_{ij} = P(G_i|G_i \vee G_j)$

Potom

$$P_m = \frac{\prod_{j=1,j\neq m}^{N} P_{mj}}{\sum_{j=1,j\neq m}^{N}\left(\prod_{k=1,k\neq m,k\neq j}^{N} P_{mk}\right) - (N-2)N\prod_{j=1,j\neq m}^{N} P_{mj}} \tag{2.9}$$

Tento výpočet je dosť časovo náročný, a preto zvolili výslednú pravdepodbnosť ako

$$P_m = \frac{1}{N}\sum_{j=1,j\neq m}^{N} P_{jm} \tag{2.10}$$

Takto počítaná pravdepodobnosť sa len málo líšila od reálnej a teda úspešnosť sa skoro vôbec nezmenila.

Tento prístup má niekoľko výhod oproti klasifikátoru, ktorý pozostáva len z jednej veľkej neurónovej siete.

- pre každý pár môžme použiť rôzny spôsob extrakcie príznakov a typ klasifikátora
- znižuje sa citlivosť siete na zvolené parametre
- znižuje sa čas a množstvo dát potrebných na trénovanie siete
- ako "základný" klasifikátor sa nemusí použiť neurónová sieť, ale ľubovolný klasifikátor, ktorý vypočíta potrebné pravdepodobnosti

2.2 Fonetika slovenských hlások

2.2.1 Samohláskové zvuky

Rozlišujú sa krátke a dlhé samohlásky, monoftongy (jednohlásky) a diftongy (dvojhlásky). Z funkčného hľadiska sú všetky tieto samohláskové zvuky slabikotvorné (sonanty). To znamené, že sú jadrom slabiky, takže v slove je najviac toľko slabík, koľko je v ňom samohláskových zvukov. Všetky slovenské samohlásky sú znelé - pri ich artikulácii hlasivky kmitajú. Dlhé samohlásky trvajú v tom istom rečovom tempe prbližne dvakrát toľko ako krátke samohlásky.

Pri štúdiu dlhých a krátkych slovenských samohlások zisťujeme, že dlhé á je otvorenejšie než krátke a, no všetky ostatné dlhé samohlásky sú zatvorenejšie. Diferencie sú malé, akustické dôsledky sú z percepčného hľadiska nesignifikantné. Z fyzikálneho hľadiska sú samohlásky zložené zvuky so zosilnenými harmonickými tónmi v určitých frekvenčných oblastiach. Na oscilogramoch možno bez ťažkostí určiť jednotlivé hlasivkové cykly (jeden hlasivkový cyklus je jedno zatvorenie a otvorenie hlasiviek). Modulácia hlasivkových kmitov je pri každej samohláske iná. Zreteľne sa prejavuje najmä modulácia prvým a druhým formantom. Kvôli týmto vlastnostim by mali byť samohlásky rozponávané z najlepšou úspešnosťou. Pri určení, či ide o krátku alebo dlhú samhlásku, sa stačí zamerať len na dĺžku prehovoreného zvuku.

2.2.2 Spoluhláskové zvuky

Základnou spoločnou vlastnosťou spoluhlások z artikulačného hľadiska je prekážkovosť a z akustického hľadiska je šumovosť. Artikuláciou spoluhlások sa modifikuje i prerušuje zvukový prúd tónov samohlások a moduluje sa rytmus reči v slabikách, lebo spoluhlásky tvoria okraje slabík. Ak je priechod hlasovému a vzduchovému prúdu úplne zablokovaný, vznikajú záverové spoluhlásky, ak je prekážkou úžinam vznikajú úžinové spoluhlásky. Pri niektorých spoluhláskach sa ústna prekážka môže kombinovať s podnebnohltanovým otvorom. Vtedy vznikajú nosové spoluhlásky. Ak po závere nenasleduje explózia, ale frikcia, vznikajú záverovoúžinové spoluhlásky. Ak sa prekážka v strede ústnej dutiny kombinuje s voľným priechodom na bokoch ústnej dutiny, vznikajú bokové spoluhlásky, a ak je prekážka prerušovaná uvoľňovaním priechodu, artikulujú sa kmitavé spoluhlásky. Nosové, bokové, kmitavé a kĺzavé hlásky nebývajú šumové.

Z hľadiska účasti hlasivkového tónu na artikulácii tvoria spoluhlásky triedu znelých a neznelých spoluhlások, no znelosť - neznelosť je v slovenčine asimilovateľná vlastnosť, takže v určitých situáciách sa namiesto neznelej vyslovuje znelá spoluhláska a naopak.

Výraznou skupinovou konsonatickou vlastnosťou je sykavosť. Majú ju spoluhlásky s, z, c, dz, š, ž, č, dž. Označenie sykavosť sa opiera o sluchový vnem. V rámci sykavosti sa z hľadiska sluchováho dojmu rozlišujú dve podskupiny: ostré sykavé spoluhlásky (s, z, c, dz) a tupé sykavé spoluhlásky (š, ž, č, dž). V slovenčine sa často dávajú do protikladu tvrdé a mäkké spoluhlásky.

Z fonetického hľadiska možno objektívne podložiť tento protiklad iba v skupinách t, d, n, ľ - ť, ď, ň, ľ ak mäkkosť stotožníme so širokým dotykom jazyka o podnebnú klenbu, tvrdosť stotožníme s úzkym dotykom. Ďalej by som chcem zhrnúť slovenské hlásky v skupinách podľa ich podobnosti a odlišnosti[3]. Obyčajne hlásky v spoločnej skupine majú množstvo spoločných vlastností, odlišujú sa medzi sebou po pároch jednou výraznou vlasnosťou, ktorá reprezentuje ich diferenčný príznak.

a) Delenie spoluhlások podľa miesta artikulácie

   - pernoperné: p, b, m
   - pernozubné: f, v
   - predoďasnové: t, d, n, s, z, c, dz
   - zadoďasnové: š, ž, č, dž, r, ŕ, l, Ĺ
   - ďasnopodnebné: ť, ď, ň, ľ
   - tvrdopodnebné: j
   - mäkkopodnebné: k, g, ch
   - hrtanové: h

b) Delenie podľa spôsobu artikulácie

   - záverové: p, b, m, t, d, n, ť, ď, ň, k, g
   - pernozubné: f, v, s, z, š, ž, r, l, ŕ, Ĺ, ľ, j, ch, h
   - záverovoúžinové: c, dz, č, dž

c) Delenie podľa artikulujúceho orgánu

   - pernoperné: p, b, m
   - pernozubné: f, v
   - jazyk
     - končekové: t, d, n, s, z, c, dz, š, ž, č, dž, r, ŕ, l, Ĺ
     - predochrbtové: ť, ď, ň, ľ, j

[3]Obmedzím sa na hlásky, ktoré prichádzajú do úvahy pri hláskovaní slov

- zadochrbtové: k, g, ch

– hlasivkové: h

d) Delenie podľa sluchového dojmu

– explozívne: p, b, m, t, d, n, ť, ď, ň, k, g

– frikatívy: f, v, h s, z, š, ž, r, ŕ, l, Ĺ, ľ, j, ch

– afrikované: c, dz, č, dž

e) Delenie podľa účasti hlasu

– znelé: b, d, ď, g, v, z, ž, dz, dž, j

– neznelé: p, t, ť, k, f, s, š, c, č, ch

– nepárové (sonóry): r, ŕ, l, Ĺ, m, n, ň, ľ, h

f) Delenie podľa účasti nosovej dutiny

– nosné: m, n, ň

– ústne: všetky ostatné

g) Delenie podľa trvania

– dlhé: ŕ, Ĺ

– krátke: všetky ostatné

Ďalej možno slovenské hlásky začleniť do nasledovných tried:

- sykavé: s, z, c, dz; š, ž, č, dž
- ostré: s, z, c, dz; t, d, n, ľ; i, e
- tupé: p, b, m; k, g; ch, j; h; v, f; u, o, ô

Kapitola 3

Databáza

Každý rozpoznávač reči potrebuje databázu vzoriek, na ktorej sa natrénuje. Preto mojou prvou úlohou bolo vytvoriť databázu nahovorených písmen. Od jej kvality závisí aj výsledná (ne)úspešnosť klasifikátora. Inými slovami, bez kvalitnej databázy nemožno vytvoriť dobre fungujúci rozpoznávač. Pojem "kvalitná" zavisí od toho na čo má byť použitá. Ak sa má napríklad rozpoznávať reč len jedného človeka, stačí databáza slov nahovorených týmto človekom. Ak však má rozpoznávač fungovať nezávisle od rečníka, treba databázu zozbieranú od rôznych rečníkov. V mojom prípade ide o rozponávač pracujúci nezávisle od rečníka, a preto som od databázy očakával nasledovné vlasnosti:

- veľa rôznych rečníkov
- veľa nahovorených vzoriek od každého rečníka
- dobre rozpoznateľné hranice slov
- žiadny ďalší rečový signál v pozadí

Čím je datáza väčšia a bohatšia, tým môže klasifikátor lepšie zovšeobecňovať. Ideálne by bolo, keby som mal pre každú hlásku rádovo stovky vzoriek. Pri štyridsiatich hláskach sa potom celkový počet vzoriek šplhá do desaťtisícov. Takéto množstvo vzoriek sa mi asi nepodarí zozbierať. Chcel by som vytvoriť databázu asi 50 vzoriek na jednu hlásku. Aj tento cieľ je pomerne odvážny, lebo to znamená spolu 2000 vzoriek.

3.1 Príprava na nahrávanie

Ako som už spomenul v úvode, nahrávať vzorky som sa rozhodol s deťmi na základnej škole. K dispozícii som mal osem počítačov s mikrofónmi, ktoré však boli v jednej miestnosti. Kedže mojou prioritou bolo zobierať čo možno najviac vzoriek, rozhodol som sa pre paralelné nahrávanie. Jeho výhodou oproti nahrávaniu na jednom počítači, je samozrejme rýchlosť, nevýhodou je fakt, že sa v pozadí mohli objaviť iné rečové signály, ktoré môžu výrazne znižovať úspešnosť rozpoznávača. Ja som v tomto prípade nemal na výber, lebo som mal na nahrávanie len niečo viac ako hodinu.

Keďže nahrávali žiaci druhého ročníka, musel som vytvoriť vlastnú aplikáciu, ktorá okrem schopnosti nahrávania, musela spĺňať ešte dve podmienky. Prvou bolo mať jednoduché ovládanie. Počítal som s tým, že niektoré deti sa nikdy predtým nestretli s počítačom a že im spôsob ovládania budem musieť vysvetliť čo možno najrýchlejšie. Druhou podmienkou bolo, aby mala aplikácia pekný design, aby boli deti ochotné s ňou pracovať. Bál som sa, že deti nebudú ochotné spolupracovať, ako sa ukázalo neskôr, moje obavy boli neopodstatnené.

3.2 Aplikácia

Program na nahrávanie vzoriek fungoval nasledovane:

- na obrazovke sa zobrazilo písmenko
- počas držania medzerníka sa nahrával zvuk
- po pustení medzerníka sa zaznamenaný zvuk uložil do príslušného priečinka
- zobrazilo sa nové písmeno
- po nahratí všetkých písmen abecedy sa aplikácia ukončila

Snažil som sa, aby nahrávanie nebolo až také fádne, a preto sa písmená abedcedy zobrazovali v náhodnom poradí. Ako som už spomínal, nechcem sa zaoberať všetkymi písmenami abecedy, ale len tými, ktoré korešpondujú s nejakou hláskou. Teda písmená Ä, Y, Ý, W a X som nahrával len pre úplnosť. Nahraté zvuky som ukladal do formátu *.wav. Program sa volá *Abeceda.exe* a možno ho aj so zdrojovými kódmi nájsť na priloženom CD.

3.3 Nahrávanie

Samotné nahrávanie sa uskutočnilo na Základnej škole Černyševského v Bratislave. Mal som k dispozícii tri skupinky detí po osem. Každej skupinke som musel vysvetliť spôsob akým sa program obsluhuje. Aj keď deti obsluhu programu pochopili, viaceré mali problém skoordinovať držanie medzerníka a hovorenie. Nahrávanie ich však bavilo, a väčšina detí nahrala celý sled písmeniek štyri-päťkrát.

3.4 Výsledky nahrávania

Dokopy sa nahralo asi 4000 vzoriek. S týmto číslom som bol veľmi spokojný, aj keď sa neskôr ukázalo, že veľké množstvo vzoriek je nepoužiteľných. Musel som ich ručne spracovať - vybrať použiteľné vzorky, určiť presný začiatok a koniec písmena a označiť, o aké písmeno ide. Zvuk bol označený názvom súboru, teda napríklad Ť2.wav je tretie nahraté písmeno Ť (začínalo sa od 0). Najčastejšie chyby, kvôli ktorým som vzorku nemohol zaradiť do databázy, boli:

- zlá a nezrozumiteľná výslovnosť nahratého slova
- príliš tiché vyslovenie
- odseknutý začiatok alebo koniec písmena
- šum alebo reč v pozadí
- žiadna vzorka

Moja snaha bola, aby bola databáza čo najkvalitnejšia, ale zároveň čo najväčšia. Keďže si tieto dve podmienky v podstate protirečia, musel som nájsť nejaký dobrý kompromis. Mikrofóny neboli veľmi kvalitné a niektoré deti mali problémy vysloviť niektoré písmená. Tieto dva faktory ešte znižovali množstvo použiteľných vzoriek. Prvé a najzákladnejšie pravidlo, ktorým som sa riadil pri triedení bolo, že ak danú vzorku nedokážem rozpoznať na prvýkrát, tak ju nemôžem zaradiť. Často sa stávalo, že až po viacerých vypočutiach som určil o aké písmeno ide. Dokonca občas som si po prvom vypočutí myslel, že viem o aké písmeno ide a po ďalších vypočutiach som zistil, že je to iné písmeno[1].

[1] je vážne zaujímavé pozorovať ako asi pracuje ľudský mozog

Najčastejšie sa takto mýlili písmená P-B, K-G-CH, S-Z-C-Dz, M-N, T-D, Ľ-J a Š-Č.

Celkové výsledky zozbieraných písmen neboli žiadnym prekvapením. Počty vzoriek jednoznačne kopírujú ľudské (moje) schopnosti rozpoznávania. Tabuľka 3.1 zobrazuje výsledné početnosti. Najviac vzoriek v databáze majú jednoznačne samohlásky. Je to čiastočne tým, že sú znelé a teda veľmi dobre rozpoznateľné a možno čiastočne tým, že deti prvého ročníka sa ako prvé učia samohlásky, a preto ich asi nejlepšie vyslovujú. Celkovým víťazom je A-Á (opäť žiadne prekvapenie). Pri samohláskach som mal občas problém rozpoznať či ide o krátku alebo dlhú hlásku, a preto vo výslednej databáze môžu byť A a Á pomiešané. Rozdiel medzi A a Á je však len v dĺžke, a preto ich možno rozdeliť jednoduchým určením prahu.

Spoluhlások je zozbieraných približne rovnaký počet. Výrazne menej je len Dz a F, preto očakávam najväčšie problémy pri rozpoznávaní týchto spoluhlások. Aj keď sa mi pri väčšine hlások nepodarilo zozbierať ani 50 vzoriek, budem sa snažiť pracovať aj s takouto databázou. Očakávam, že početnosť zozbieraných vzoriek bude priamo úmerná úspešnosti môjho rozpoznávača, teda že najlepšie sa budú rozponávať samohlásky a najhoršie to bude pri spoluhláskach, ktoré sa mi mýlili pri triedení nahratých vzoriek.

Tabuľka 3.1: Početnosti zozbieraných hlások

Hláska	AÁ	B	C	Č	D	Ď	Dz	Dž	EÉ	F	G
Početnosť	136	38	23	24	25	30	13	35	98	16	35
Hláska	H	Ch	IÍ	J	K	LĹ	Ľ	M	N	Ň	OÓ
Početnosť	37	20	88	34	41	41	30	32	38	45	86
Hláska	ô	P	RŔ	S	Š	T	Ť	UÚ	V	Z	Ž
Početnosť	30	38	80	25	32	41	33	82	42	30	28

Kapitola 4

Návrh

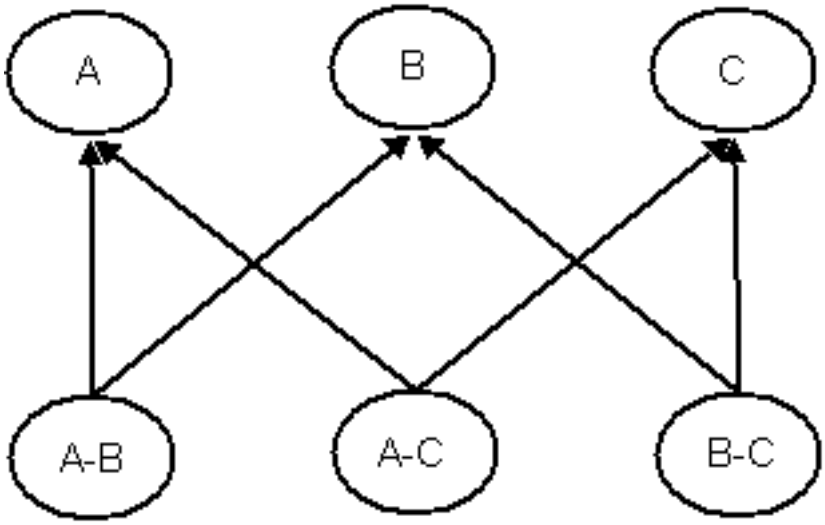
V tejto kapitole by som sa chcel venovať základu celého rozpoznávacieho systému - klasifkátoru. Nebudem sa zaoberať určovaním hranice vstupného slova, pretože toto som vyriešil ručným spracovaním vstupov, kde som tieto hranice určil presne. Takisto ma nezaujíma, aký je vstup klasifikátora. Môžu ním byť MFCC koefcienty, LPC koefcienty, alebo v podstate ľubovoľná, konečná postupnosť vektorov príznakov. Teda klasifikátor presnejšie definujem ako algoritmus, ktorý spĺňa:

- Vstup: Postupnosť $\overrightarrow{a_1}, \overrightarrow{a_2} \ldots \overrightarrow{a_M}$, kde a_i , $1 \leq i \leq M$, je N rozmerný vektor príznakov. Treba počítať s tým, že každá vstupná postupnosť môže byť rôzne dlhá, ale N je konštantné.
- Výstup: Rozpoznaná hláska (prípadne ešte aj pravdepodobnosť, s akou je určená)
- Podmienka: Rýchlosť - systém má pracovať v reálnom čase, navyše aj keď sa nebudem zaoberať predspracovaním signálu, treba uvažovať čas nutný na jeho vykonanie. Preto sa budem snažiť, aby klasifikátor ukončil výpočet do 0.3 sekundy.

4.1 Teoretický návrh

Pri teoretickom návrhu klasifikátora som sa nechal inšpirovať BPP prístupom, ktorý som spomenul v prehľadovej kapitole. Ak by som chcel použiť presne rovnaký postup ako v [3], znamenalo by to pre mňa vytvoriť pre každú dvojicu písmen samostatný rozpoznávač - **subklasifikátor**. Tento by mal za úlohu

určiť pravdepodobnosť, že vstupná postupnosť zodpovedá prvej alebo druhej hláske. Pre každú hlásku by sa tieto pravdepodobnosti sčítali a hláska s najvyšším súčtom by bola výstupom. Napríklad pre hlásky A,B,C by takýto rozpoznávač vyzeral ako na obrázku 4.1.



Obrázok 4.1: Klasifikátor pre triedy A,B,C

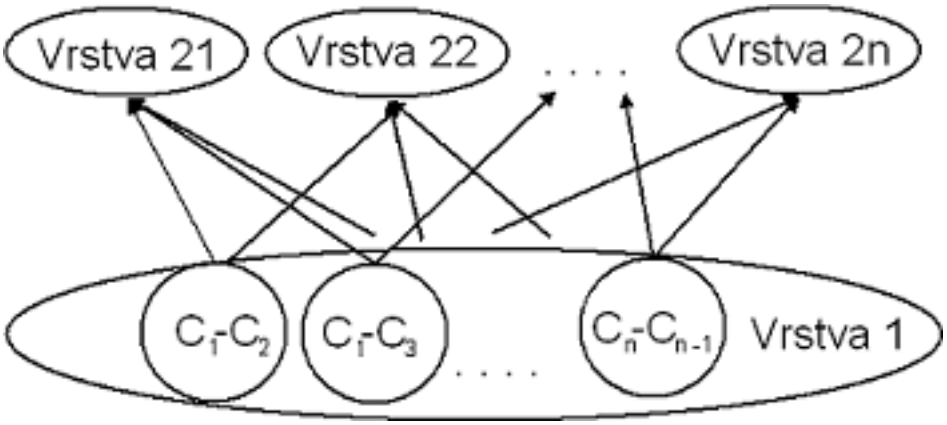
Hlavný problém pri tomto prístupe je rýchlosť. Mnou navrhnutý klasifikátor by mal rozpoznávať štyridsať hlások. To pri tomto prístupe znamená $\binom{40}{2} = 780$ potrebných subklasifikátorov, z ktorých by sa skladal výsledný klasifikátor. Toto číslo sa mi zdá privysoké, a preto sa ho ďalej snažím znížiť. Najjednoduchší spôsob, ako to urobiť, je znížiť počet rozpoznávaných tried.

Pre dlhé samohlásky aj spoluhlásky platí, že sa veľmi nelíšia od krátkej formy. Hlavný rozdiel je v dĺžke. Preto som sa ich rozhodol považovať za rovnaké písmeno - akési dvojpísmeno. Ak klasifikátor určí ako výstup takéto dvojpísmeno výsledné rozpoznanie určím podľa nejakého pevne stanoveného prahu. Týmto spôsobom znížim počet tried o 7 (AÁ, UÚ, IÍ, EÉ, OÓ, RŔ, LĹ) na 33. Je to stále priveľa, lebo $\binom{33}{2} = 528$. Preto použijem trochu iný prístup. Skúsim nerozpoznávať medzi každými dvomi hláskami, ale usporiadať si hlásky do tried, rozhodovať medzi triedami, a potom vybrať víťaza v rámci tejto (alebo každej) triedy. Rozdelím teda klasifikáciu do dvoch úrovní.

Označím pojmom vrstva (layer) systém, ktorý funguje podobne ako BPP. Rozhoduje medzi množinami (triedami) $C_1, C_2, ...C_n$, ktoré musia byť disjunktné.

To znamená, že sa vytvorí pre každú dvojicu tried $C_i, C_j\, i \neq\ j$, subklasifikátor K_{ij}, ktorý určí hodnotu v_{ij}. Táto hodnota by mala byť úmerná pravdepodobnosti, že vstup patrí do triedy C_i v porovnaní s C_j a musí platiť $v_{ij} = -v_{ji}$. Z čiastkových "súbojov" každej triedy sa vypočíta výsledné číslo $P_i = \sum_{j=1,j\neq\ i}^{n} w_{ij}.v_{ij}$, kde w_{ij} je váha medzi triedami C_i, C_j. Toto číslo je určené na vyrovnávanie rozdielov medzi rôznymi typmi subklasifikátorov. Ak takéto vyrovnávanie nie je nutné, stačí všetky w_{ij} nastaviť na hodnotu 1.

Klasifikátor sa bude skladať z nasledujúcich vrstiev. V prvej vrstve L_1 budú triedy $C_1^1, C_2^1, \ldots C_n^1$, ktorých zjednotením budú všetky rozpoznávané hlásky. Ďalej vytvorím vrstvu L_{2k}, pre každú triedu C_k^1 z vrstvy L_1, ktorá bude rozhodovať medzi prvkami triedy C_k^1. Triedy vrstiev L_{2k} budú už jednoprvkové, a preto netreba d'alšie vrstvy. Bude to vyzerať ako na obrázku 4.2



Obrázok 4.2: Usporiadanie do vrstiev

Existuje viacero spôsobov, ako určiť výstup klasifikátora:

1. z L_1 vybrať triedu s najvyššou hodnotou P_i^1 a určiť ako víťaza najpravdepodobnejší prvok z vrstvy L_{2i}
2. pre všetky hlásky prenásobiť P_i^1 z L_1 s P_j^{2i} z príslušného L_{2i} a určiť ako víťaza hlásku s najväčšou hodnotou takéhoto súčinu
3. určiť prah pre P_i^1 z L_1, a potom počítať ako v predchádzajúcom bode, ale len s triedami, ktoré prekročia tento prah

Neviem jednoznačne povedať, ktorú možnosť si vybrať, inak ako ich otestovať a vybrať tú najúspešnejšiu.

Druhý problém, ktorý prirodzene vzniká znie: "Ako určiť triedy $C_1^1, C_2^1, \ldots C_n^1$, teda ako rozdeliť všetky hlásky do takýchto tried?"Rozdelenie v prvej vrstve je kľúčové, pretože zbytok je už jednoznačne daný. V tomto prípade som sa rohodol skombinovať teoretické vedomosti zhrnuté v prehľadovej kapitole s praktickými skúsenosťami z nahrávania. Ako prirodzené triedy považujem hlásky, ktoré sa mi mýlili pri ručnom spracovaní. Teda začnem s triedami ako sykavé, tupé, ostré ..., postupne ich budem testovať, obmieňať a hľadať optimálnu kombináciu.

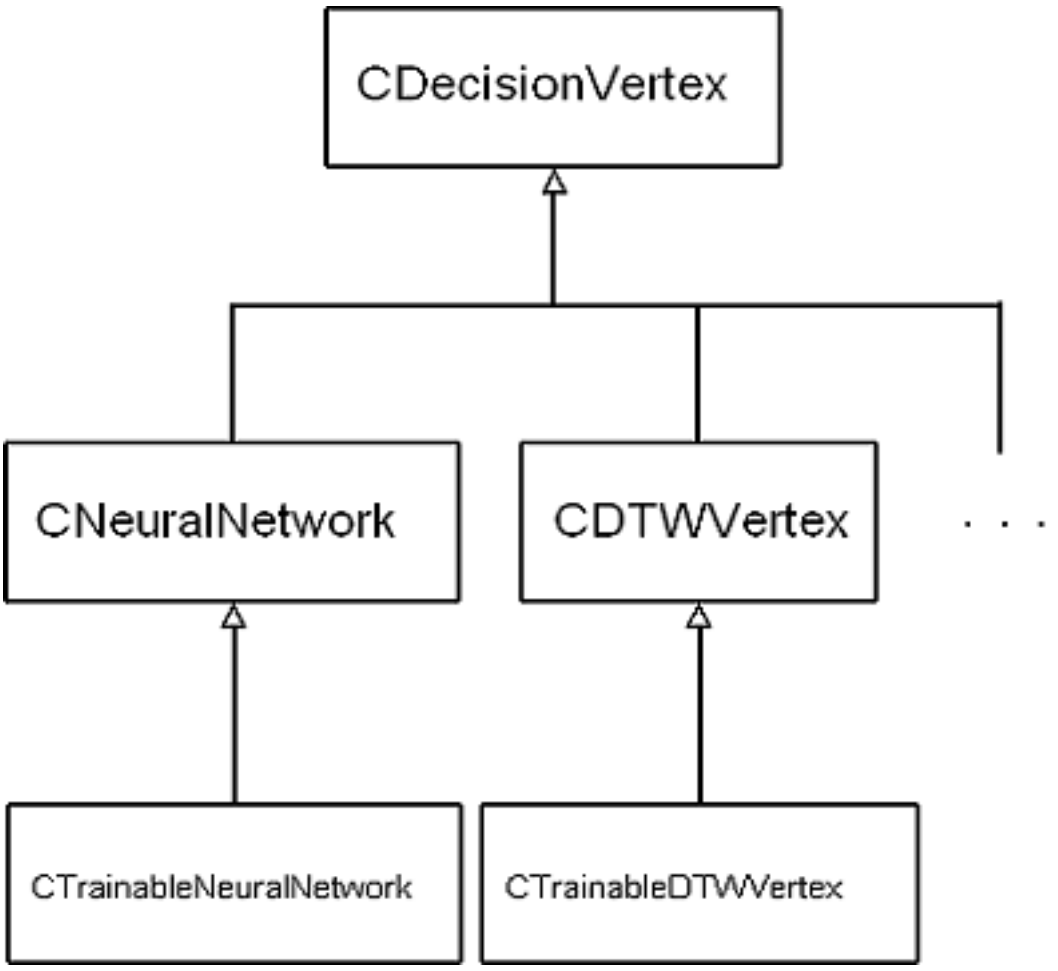
Pri použití subklasifikátora, ktorý sa trénuje (Skryté Markovove modely, neurónovú sieť, ...) platí, že potrebuje vzorky na natrénovanie. Bez dostatočného množstva je schopný len obmedzenej generalizácie. Preto bolo vytvorenie tried hlások nutné, a preto očakávam aj lepšie výsledky, ako keby som použil pôvodný prístup s $\binom{n}{2}$ klasifikátormi.

4.2 Návrh systému

V predchádzajúcej časti som sa venoval teoretickému návrhu rozpoznávača, pritom som sa nezaujímal o typy subklasifikátorov. Možno sa na ne pozerať ako na čierne skrinky, ktoré ma zatiaľ nezaujímajú ako pracujú. Pri tom chcem ešte chvíľu zotrvať. Dôvod je jednoduchý - zmena subklasifikátorov. Povedzme, že vytvorím taký rozpoznávač, že všetky subklasifikátory budú neurónové siete. Každá bude natrénovaná na rozpoznávanie medzi svojimi triedami. Niektoré budú dosahovať dobré úspešnosti, niektoré však nie. Zistím napríklad, že medzi triedami (jednoprvkovými) K a G sa dosahuje úspešnosť rozpoznania len 50 %. To znamená, že môj rozpoznávač pracuje rovnako dobre a ako keby určoval triedu náhodne. Tento subklasifikátor nahradím napríklad algoritmom DTW, ktorý by mohol pracovať lepšie. Takto budem hľadať optimálnu stavbu klasifikátora.

Definujem si triedu abstraktnú triedu CDecisionVertex. Táto predstavuje subklasifikátor. Oddelené triedy musia implementovať metódu LikelihoodDistribution. Vracia dvojicu hodnôt - v_{ij} a v_{ji}. Mnou použité triedy sú zobrazené na obrázku 4.3.

Triedy CTrainableNeuralNetwork a CTestDTWDecisionVertex sú obohatené o metódy určené na trénovanie a testovanie úspešnosti subklasifikátorov. Vrstve je ekvivalentná trieda CLayer. Tá používa CDecisionVertexy ako subklasifiká-



Obrázok 4.3: Triedy subklasifikátorov

tory. Z inštancií CLayer je zložený finálny klasifikátor. Pre implementáciu vlastného subklasifikátora stačí oddediť od CDecisionVertex a implementovať metódu LikelihoodDistribution.

4.3 Subklasifikátory

Najskôr sa budem zaoberať vstupmi pre subklasifikátory. Nechcem opäť testovať rôzne typy vektorov príznakov, lebo na túto tému vzniklo už viacero prác. Použijem radšej ich výsledky. Ako vstup som sa rozhodol použiť MFCC koeficienty s delta koeficientami a príslušnými energiami. Na ich výpočet použijem *HList.exe* z knižnice HTK Toolkit[18]. Presné konfiguračné hodnoty možno nájsť na priloženom CD v adresáry *WavToMFCC*.

Ďalšou otázkou je, aké typy subklasifikátorov použiť. Ja som sa rozhodol zamerať na neurónové siete, pretože ich používali aj v pôvodnom BPP návrhu. Očakávam, že druhé vrstvy nebudú pracovať veľmi dobre, kvôli nízkemu počtu vzoriek. Tieto neúspešné subklasifikátory skúsim nahradiť subklasifikátorom používajúcim DTW algoritmus.

4.3.1 Neurónová sieť

Jedna z nevýhod neurónovej siete je množstvo parametrov, ktoré treba určiť. Pri niektorých budem hľadať optimálnu hodnotu, inde určím hodnotu bez skúšania na základe mojich predchádzajúcich skúsenostiach a podľa [12]. Vstupné vektory príznakov sú rôzne dlhé, ale štandardná dopredná neurónová sieť potrebuje konštantnú dĺžku vstupu. Preto mám dve možnosti:

- upraviť ľubovoľný vstup na konštantnú dĺžku
- použiť rekurentnú neurónovú sieť, ktorá môže spracovať teoreticky nekonečne dlhý vstup

Rozhodol som sa pre prvú možnosť, a teda použiť doprednú neurónovú sieť s jednou skrytou vrstvou. Ako trénovací algoritmus použijem back-proapagation s rýchlosťou učenia $\alpha = 0.1$. Počiatočné váhy budú nahodné z intervalu $\langle -0.3, 0.3 \rangle$.

Vstupná vrstva

Spôsob, akým upraviť vstup, sa volá "metóda segmentácie spektrálnej stopy" a bola popísaná v [1]. Princíp je jednoduchý. Majme na vstupe N-rozmerné vektory, ich počet je M ale potrebuejeme K. Tieto vektory určujú body v N-rozmernom priestore. Keď ich berieme v poradí od 1 po M, tak vlastne určujú M-1 úsečiek. Spolu udávajú jednu krivku, na ktorej si zafixujeme prvý a posledný bod a ostatných K-2 bodov budú na tejto krivke v rovnakej vzdialenosti od seba. Krivka je teda aproximovaná polygónom.

Vstupom teda budú konštantne dlhé postupnosti vektorov príznakov. Ďalej použijem bipolárnu aktivačnú funkciu, teda potrebujem vstupy normalizovať na interval $\langle -1, 1 \rangle$. Treba si však uvedomiť, že rozdiel 0.5 medzi dvomi hodnotami MFCC koeficientov nemá rovnakú váhu ako takýto rozdiel medzi dvomi

deltami. Preto budem skúšať rôzne normalizačné funkcie a hľadať tú najoptimálnejšiu trojicu.

Skrytá vrstva

Použijem jednu skrytú vrstvu, pretože viac vrstiev už nezlepšuje vlastnosti [12]. Nechcem sa zaoberať prílišným testovaním počtu skrytých neurónov. Určím dve alebo tri možnosti, tie natrénujem a vyberiem tú najlepšiu možnosť.

Výstupná vrstva

Pri každom subklasifikátore ide o rozpoznávanie medzi dvomi triedami. Použijem metódu "one-hot", teda dva výstupné neuróny. Úlohou je z dvoch hodnôt výstupných neurónov, vypočítať hodnoty v_{ij} respektíve v_{ji}. Nech na vstupe $\overrightarrow{a_1}, \overrightarrow{a_2}, \ldots \overrightarrow{,} a_M$ majú výstupné dva neuróny hodnoty o_1 a o_2. Potom postavím v_{ij} rovné rozdielu týchto vstupov:

$$v_{ij} = o_1 - o_2 \tag{4.1}$$

$$v_{ji} = o_2 - o_1 = -(o_1 - o_2) = -v_{ij} \tag{4.2}$$

Hodnoty o_1 a o_2 sú z intervalu $\langle -1, 1 \rangle$, preto v_{ij} a v_{ji} patria $\langle -2, 2 \rangle$.

4.3.2 DTW subklasifikátor

Ako som už spomenul, ak nebude dobre pracovať neurónová sieť, pokúsim sa použiť algoritmus DTW. Sieť bude pravdpodobne neúspešná preto, že nebude mať dostatočné množstvo vzoriek na natrénovanie. Toto sa budem snažiť využiť v môj prospech.

Nech subklasifikátor rozhoduje medzi triedami C_i a C_j. Potom bude pracovať nasledovane:

1. Použijem štandardný DTW algoritmus[1], kde sa výsledná vzdialenosť normalizuje dĺžkou cesty
2. Vypočíta sa vzdialenosť od všetkých vzoriek triedy C_i a vyberie sa tá najmenšia. Pri výpočte vzdialenosti použijem Euklidovskú metriku

[1] existuje množstvo obmien, možno ich nájsť v [5]

3. Rovnako sa spočítajú vzdialenosti od vzoriek triedy C_j a vyberie sa tá najmenšia

4. Triede s menšou vzdialenosťou ku vstupu sa priradí hodnota 2, tej druhej -2

Hodnoty 2 a -2 som zvolil zámerne, kvôli kompatibilite s používanými neurónovými sieťami. Ak by som zvolil hodnoty inak, treba upraviť príslušné váhy w_{ij}. Očakávam, že výpočet nebude trvať pridlho, lebo vzoriek nebude priveľa.

DTW algoritmus pracuje som vstupom postupností rozličných dĺžok, a preto dĺžku vstupu nemusím upravovať. Vzniká však rovnaký problém ako pri neurónových sieťach a to rôzna sila MFCC koeficientov, delta koeficientov a energie. Preto budem musieť opäť normalizovať tieto hodnoty.

Kapitola 5

Trénovanie a testovanie subklasifikátorov

V tejto kapitole popíšem trénovanie a testovanie subklasifikátorov. Najskôr sa zamerám na trénovanie neurónových sietí. Pomocou dosiahnutých úspešností sa budem snažiť hľadať optimálne rozdelenie hlások na triedy. Potom otestujem DTW subklasifikátory, vyberiem tie najúspešnejšie, ktoré nakoniec použijem vo finálnom klasfikátore. Na trénovanie aj testovanie som použil aplikáciu *TestingApp.exe*.

5.1 Neurónová sieť

5.1.1 Trénovanie neurónovej siete

Ako som už spomenul, neurónová sieť má viacero parametrov, ktoré treba nastaviť. Preto som si rozdelil tieto parametre na tie, ktoré určím a tie, ktoré budem testovať a hľadať optimálnu hodnotu. Parametre, ktoré som pevne stanovil:

- počet vstupných neurónov - vstup som upravoval na desať 26-rozmerných vektorov a teda výsledný počet bol 260
- rýchlosť učenia - $\alpha = 0.1$
- počet neurónov skrytej vrstvy - 50
- počiatočné váhy - náhodné hodnoty z intervalu $\langle -0.3, 0.3 \rangle$

Snažil som sa určiť:

- pre každú dvojicu tried optimálne normalizačné hodnoty MFCC koeficientov, delt aj energií
- rozdelenie tried prvej vrstvy

Neurónovú sieť som trénoval štantdardným algoritmom back-propagation. Množinu všetkých vzoriek som rozdelil na testovaciu a trénovaciu množinu v pomere 70% ku 30%. Sieť najskôr prešla celou trénovacou množinou desaťkrát - desať epoch, a potom som trénoval, pokiaľ nenastala aspoň jedna z nasledujúcich možností:

a) začala rásť chyba trénovacej množiny

b) kumulovaná chyba trénovacej množiny klesla pod hodnotu 0.3

c) trénovanie trvalo dlhšie ako 100 epoch

Trénovanie som opakoval päťkrát pre každý typ normalizácie a vybral tie najlepšie výsledky. Vyjadroval som aj percentuálnu chybu neurónovej siete, ako podiel počtu zle rozpoznaných vzoriek a všetkých vzoriek. Pre dobré rozpoznanie musel mať výstupný neurón triedy hodnotu v intervale $\langle 0.5, 1\rangle$ a druhý musel mať hodnotu z intervalu $\langle -0.5, -1\rangle$.

5.1.2 Normalizácia

Po podrobnejšom preskúmaní vstupných vektorov príznakov som zistil, že MFCC koeficienty sa zvyčajne pohybujú v intervale $\langle -20, 20\rangle$. Delta koeficienty sú samozrejme menšie a to približne z intervalu $\langle -10, 10\rangle$. Najmenšie sú energie - $\langle -1, 1\rangle$. Normalizoval som tak, že som určil kladné číslo a ním som všetky hodnoty predelil. Ak výsledné číslo bolo väčšie ako 1 (menšie ako -1), tak som mu priradil 1 (-1). Pri normalizácii nulou sú všetky normalizované hodnoty nulové. Pri hľadaní optimálnych hodnôt som postupoval nasledovne:

1. vybral som päť hodnôt pre každý typ koeficientu
2. preskúšal všetky možnosti
3. vybral tú najlepšiu

Takto sa nedalo postupovať pri celom trénovaní, lebo $5^3 = 125$ možností bolo priveľa z časového hľadiska - trénovanie prvej vrstvy by trvalo rádovo dni. Preto som takýto postup zvolil len pre pár subklasifikátorov. Podľa výsledkov som vybral dvanásť zmysluplných (s najlešou úspešnosťou) trojíc hodnôt, a potom používal už len tieto. Trojice boli nasledovné: [20,20,20][1] [20,20,5] [20,10,10] [10,10,0] [20,5,0.5] [20,1,1] [10,2,0.5] [20,10,1] [20,2,0.1] [20,0.5,0.1] [20,5,0] [20,1,0]. Čím vyššia hodnota, tým sa viac potláčajú dané koeficienty, čím menšia tým sa viac zosilňujú. Najoptimálnejšia trojica by mala byť [20,10,1].

5.1.3 Výsledky

Pri rozdelení hlások do tried som sa snažil naraz splniť tieto podmienky:

- dobrá úspešnosť neurónovej siete pri rozhodovaní medzi triedami
- malý počet tried
- približne rovnako veľké triedy (žiadna trieda nesmie byť priveľká)

Prvý bod zaručuje dobrú výslednú úspešnost, druhý o dostatočnú rýchlosť rozpoznania. Tretí prihliada na počet subklasifikátorov na druhej vrstve. Ak má totiž nejaká trieda prvej vrstvy N prvkov, v druhej triede bude treba ďalších $\binom{N}{2}$ subklasifikátorov.

Najskôr som vyskúšal dať do jednej triedy samohlásky. Tento prístup sa ukázal ako neúspešný, lebo trieda bola priveľká, bez dostatočne veľa podobných znakov. Vytvoril som teda pre každú samohlásku (dvoj)triedu. Rovnako som postupoval aj pri RŔ a LĹ. Teda prvá vrstva obsahovala triedy AÁ, ôUÚ, IÍ, EÉ, OÓ, RŔ, LĹ. Skúsil som zaradiť ô k UÚ aj k OÓ, trieda ôUÚ dosahovala lepšie výsledky.

Ťažšie bolo rozdeliť zvyšné spoluhlásky. Prvotné rozdelenie bolo nasledovné P,B,M T,D,N Ť,Ď,Ň K,G,Ch S,Z,C,Dz Š,Ž,Č,Dž. Z týchto postupným skúšaním možností vznikli triedy P,B,V,T,D Ť,Ď,Ň M,N K,G,Ch S,Z,C,Dz Š,Ž,Č,Dž. Nevedel som kam zaradiť F, H, J, Ľ. Po ďalšom testovaní sa mi podarilo zaradiť J a Ľ k Ť,Ď,Ň bez výrazného zhoršenia úspešností. Skúšal som F zaradiť k PBVTD a H ku K,G,Ch, lebo tieto triedy sa mi zdali byť najpríbuznejšie. Výsledky však neboli uspokojivé, preto som ich nakoniec nechal

[1] v poradí normalizačná hodnota pre MFCC koeficienty, delta koeficienty a energie

ako samostatné triedy. Tabuľka 5.1 zobrazuje najlepšie dosiahnuté výsledky prvej vrstvy (ďalšie výsledky, pre iné rozdelenia, možno nájsť v prílohe). Stĺpce znamenajú:

1. počet epoch potrebných na natrénovanie

2.-3. dvojica tried, medzi ktorými sa rozhoduje

4. počet neurónov na vstupnej vrstve

5. počet neurónov na skrytej vrstve

6.-8. normalizačné parametre

9. kumulovaná chyba siete na trénovacej množine

10. kumulovaná chyba siete na testovacej množine

11. percentuálna chyba siete na trénovacej množine

12. percentuálna chyba siete na testovacej množine - podľa tejto chyby sú dáta usporiadané

Tabuľka 5.1: Výsledné úspešnosti prvej vrstvy

Epochy	Triedy		Parametre siete		Normalizácia			Chyby			
10	F	ŤĎŇĽJ	260	50	20	1	0	0.76	0.32	0.00%	0.00%
10	PBVTD	ŤĎŇĽJ	260	50	20	20	5	0.69	0.43	0.00%	0.00%
21	PBVTD	AÁ	260	50	20	20	20	0.49	0.46	0.00%	0.00%
10	F	LĹ	260	50	20	1	0	0.84	0.42	0.00%	0.00%
10	F	RŔ	260	50	20	10	10	0.9	0.3	0.00%	0.00%
10	MN	IÍ	260	50	20	10	10	0.68	0.28	0.00%	0.00%
13	MN	LĹ	260	50	20	5	0	0.54	0.5	0.00%	0.00%
15	MN	RŔ	260	50	20	10	10	0.54	0.47	0.00%	0.00%
10	PBVTD	OÓ	260	50	20	1	0	4.44	0.36	0.71%	0.00%
10	PBVTD	ôUÚ	260	50	20	5	0	0.63	0.34	0.00%	0.00%
10	PBVTD	ŠŽČDŽ	260	50	20	20	20	0.6	0.25	0.00%	0.00%
10	PBVTD	MN	260	50	20	20	20	0.58	0.25	0.00%	0.00%
10	SZCDZ	OÓ	260	50	10	10	0	0.69	0.34	0.00%	0.00%

10	SZCDZ	ôUÚ	260	50	20	10	10	0.68	0.42	0.00%	0.00%
10	MN	EÉ	260	50	20	20	5	0.65	0.28	0.00%	0.00%
10	SZCDZ	IÍ	260	50	20	20	5	0.65	0.31	0.00%	0.00%
10	ŠŽČDŽ	AÁ	260	50	20	20	20	0.63	0.26	0.00%	0.00%
10	ŠŽČDŽ	OÓ	260	50	20	10	10	0.72	0.29	0.00%	0.00%
10	MN	ŠŽČDŽ	260	50	20	20	20	0.64	0.26	0.00%	0.00%
12	MN	OÓ	260	50	10	10	0	0.6	0.46	0.00%	0.00%
10	SZCDZ	EÉ	260	50	10	10	0	0.63	0.29	0.00%	0.00%
10	MN	H	260	50	20	10	10	0.98	0.47	0.00%	0.00%
10	MN	ôUÚ	260	50	20	20	5	0.79	0.36	0.00%	0.00%
10	F	ŠŽČDŽ	260	50	10	10	0	0.84	0.31	0.00%	0.00%
10	F	EÉ	260	50	20	20	5	0.65	0.29	0.00%	0.00%
12	PBVTD	F	260	50	20	10	10	4.47	0.48	0.76%	0.00%
10	F	ôUÚ	260	50	20	20	5	0.87	0.35	0.00%	0.00%
10	KGCH	IÍ	260	50	10	10	0	0.62	0.29	0.00%	0.00%
10	F	AÁ	260	50	10	10	0	0.68	0.29	0.00%	0.00%
14	KGCH	H	260	50	10	10	0	4.33	0.46	1.15%	0.00%
10	F	OÓ	260	50	10	10	0	0.66	0.28	0.00%	0.00%
10	MN	AÁ	260	50	20	20	20	0.86	0.24	0.00%	0.00%
21	F	KGCH	260	50	20	20	20	0.37	0.49	0.00%	0.00%
10	MN	F	260	50	20	20	5	0.77	0.46	0.00%	0.00%
10	KGCH	ŠŽČDŽ	260	50	20	20	20	0.61	0.25	0.00%	0.00%
10	KGCH	LĹ	260	50	20	5	0	0.73	0.34	0.00%	0.00%
11	KGCH	ôUÚ	260	50	20	10	1	0.64	0.48	0.00%	0.00%
10	PBVTD	EÉ	260	50	20	20	5	0.61	0.38	0.00%	0.00%
10	PBVTD	IÍ	260	50	10	10	0	0.64	0.3	0.00%	0.00%
12	F	SZCDZ	260	50	10	2	0.5	4.43	0.46	1.45%	0.00%
10	KGCH	OÓ	260	50	20	20	5	0.66	0.27	0.00%	0.00%
10	KGCH	AÁ	260	50	20	20	20	0.71	0.31	0.00%	0.00%
14	KGCH	ŤĎŇĽJ	260	50	20	10	10	0.48	0.46	0.00%	0.00%
10	F	IÍ	260	50	20	5	0	0.74	0.34	0.00%	0.00%
10	F	H	260	50	10	2	0.5	0.843	0.419	0.00%	0.00%
10	KGCH	EÉ	260	50	20	5	0	0.62	0.29	0.00%	0.00%
10	ŤĎŇĽJ	AÁ	260	50	20	10	1	0.65	0.27	0.00%	0.00%
10	ŤĎŇĽJ	OÓ	260	50	10	10	0	0.7	0.29	0.00%	0.00%

10	LĹ	RŔ	260	50	20	20	5	0.93	0.33	0.00%	0.00%
10	RŔ	AÁ	260	50	20	10	10	0.78	0.37	0.00%	0.00%
10	RŔ	OÓ	260	50	20	20	5	0.76	0.27	0.00%	0.00%
10	ŤĎŇĽJ	ŠŽČDŽ	260	50	20	20	20	0.74	0.28	0.00%	0.00%
52	LĹ	H	260	50	20	20	5	0.12	0.5	0.00%	0.00%
10	LĹ	ŠŽČDŽ	260	50	20	20	20	0.78	0.3	0.00%	0.00%
10	ŤĎŇĽJ	RŔ	260	50	20	20	5	0.87	0.38	0.00%	0.00%
10	ŤĎŇĽJ	H	260	50	20	10	10	0.69	0.41	0.00%	0.00%
10	ŤĎŇĽJ	SZCDZ	260	50	20	10	10	0.72	0.31	0.00%	0.00%
15	RŔ	H	260	50	10	2	0.5	0.51	0.48	0.00%	0.00%
10	LĹ	AÁ	260	50	20	20	5	0.69	0.38	0.00%	0.00%
10	RŔ	EÉ	260	50	20	10	1	0.7	0.34	0.00%	0.00%
10	ŤĎŇĽJ	LĹ	260	50	20	1	0	0.76	0.46	0.00%	0.00%
10	LĹ	EÉ	260	50	20	1	1	0.63	0.3	0.00%	0.00%
10	LĹ	IÍ	260	50	20	5	0	0.74	0.35	0.00%	0.00%
10	RŔ	IÍ	260	50	10	10	0	0.69	0.29	0.00%	0.00%
10	RŔ	ŠŽČDŽ	260	50	20	20	20	0.72	0.28	0.00%	0.00%
18	LĹ	OÓ	260	50	20	10	10	0.44	0.48	0.00%	0.00%
10	LĹ	ôUÚ	260	50	20	5	0	0.85	0.42	0.00%	0.00%
10	RŔ	ôUÚ	260	50	10	10	0	0.7	0.37	0.00%	0.00%
10	OÓ	EÉ	260	50	20	10	10	0.75	0.4	0.00%	0.00%
10	OÓ	IÍ	260	50	20	20	5	0.71	0.3	0.00%	0.00%
10	H	AÁ	260	50	10	10	0	0.75	0.42	0.00%	0.00%
10	ôUÚ	IÍ	260	50	20	5	0	0.83	0.38	0.00%	0.00%
10	EÉ	IÍ	260	50	20	10	1	0.65	0.28	0.00%	0.00%
10	OÓ	ôUÚ	260	50	10	10	0	0.74	0.43	0.00%	0.00%
10	H	OÓ	260	50	20	20	20	1.27	0.34	0.00%	0.00%
10	H	ôUÚ	260	50	10	2	0.5	0.75	0.42	0.00%	0.00%
10	H	EÉ	260	50	20	20	5	0.64	0.25	0.00%	0.00%
10	H	IÍ	260	50	10	10	0	0.67	0.31	0.00%	0.00%
11	H	SZCDZ	260	50	20	20	20	4.57	0.48	1.20%	0.00%
10	SZCDZ	ŠŽČDŽ	260	50	20	20	20	0.62	0.27	0.00%	0.00%
10	H	ŠŽČDŽ	260	50	20	20	20	0.69	0.27	0.00%	0.00%
10	AÁ	ôUÚ	260	50	20	20	20	0.65	0.28	0.00%	0.00%
10	AÁ	EÉ	260	50	20	20	20	0.64	0.28	0.00%	0.00%

10	ŤĎŇĽJ	ôUÚ	260	50	10	10	0	0.68	0.34	0.00%	0.00%
10	ŤĎŇĽJ	EÉ	260	50	20	20	20	0.69	0.24	0.00%	0.00%
10	ŤĎŇĽJ	IÍ	260	50	20	20	5	0.92	0.42	0.00%	0.00%
10	ŠŽČDŽ	EÉ	260	50	20	20	20	0.62	0.26	0.00%	0.00%
10	ŠŽČDŽ	ôUÚ	260	50	20	0.5	0.1	0.71	0.31	0.00%	0.00%
10	ôUÚ	EÉ	260	50	20	20	20	0.68	0.25	0.00%	0.00%
10	AÁ	IÍ	260	50	20	20	20	0.64	0.27	0.00%	0.00%
10	ŠŽČDŽ	IÍ	260	50	20	20	5	0.7	0.3	0.00%	0.00%
100	PBVTD	RŔ	260	50	20	5	0.5	0.05	3.99	0.00%	1.37%
23	AÁ	OÓ	260	50	20	10	10	0.27	0.49	0.00%	1.64%
68	PBVTD	H	260	50	10	10	0	4.02	2.73	0.68%	1.67%
28	PBVTD	LĹ	260	50	20	10	10	0.24	0.79	0.00%	1.75%
100	SZCDZ	AÁ	260	50	20	5	0	0.05	0.63	0.00%	2.17%
17	MN	KGCH	260	50	20	10	10	0.48	2.12	0.00%	2.17%
47	RŔ	SZCDZ	260	50	20	20	20	0.14	1	0.00%	3.03%
52	LĹ	SZCDZ	260	50	20	10	10	0.12	0.5	0.00%	3.23%
23	MN	ŤĎŇĽJ	260	50	20	20	20	0.3	0.48	0.00%	3.45%
100	KGCH	SZCDZ	260	50	20	20	5	0.06	2.19	0.00%	3.92%
93	MN	SZCDZ	260	50	20	5	0	0.06	2.7	0.00%	4.55%
24	PBVTD	SZCDZ	260	50	20	5	0	8.12	10.96	1.12%	5.26%
100	KGCH	RŔ	260	50	20	1	1	0.06	3.45	0.00%	5.71%
28	PBVTD	KGCH	260	50	20	10	10	8.13	13.97	1.09%	6.41%

S výsledkami som spokojný, sú však trochu skreslené, lebo pri niektorých triedach sa mi nepodarilo dosiahnuť, aby bola chyba trénovacej množiny nula percent. Na druhej strane však vo výslednom rozpoznávači budem brať do úvahy rozdiel výstupných neurónov a nie len binárnu hodnotu rozpoznané-nerozpoznané, ako v tomto prípade. Tento fakt bude úspešnosť trochu zlepšovať.

Podľa očakávania sa dobre správali triedy AÁ, ôUÚ, IÍ, EÉ, OÓ, RŔ, LĹ. Najhoršie úspešnosti vznikali pri veľkých triedach.

Očakával som nejakú výraznejšiu prevahu niektorej normalizačnej množiny. Z môjho pohľadu sú však tieto parametre približne rovnako úspešné.

Tabuľka 5.2 zobrazuje konečné výsledky druhých vrstiev.

Tabuľka 5.2: Výsledné úspešnosti druhých vrstiev

Epochy	Triedy		Parametre siete		Normalizácia			Chyby			
10	P	B	260	50	20	1	1	0.97	0.45	0.00%	0.00%
25	P	V	260	50	20	20	20	0.5	0.5	0.00%	4.55%
21	P	T	260	50	20	10	10	0.52	6.65	0.00%	9.09%
11	P	D	260	50	20	5	0.5	0.67	0.45	0.00%	0.00%
21	B	V	260	50	10	10	0	0.66	7.8	0.00%	23.81%
10	B	T	260	50	10	2	0.5	0.68	0.35	0.00%	0.00%
14	B	D	260	50	10	2	0.5	0.85	0.49	0.00%	0.00%
10	V	T	260	50	10	2	0.5	0.73	0.34	0.00%	0.00%
72	V	D	260	50	10	10	0	0.08	1.52	0.00%	5.56%
20	T	D	260	50	20	10	10	0.41	0.5	0.00%	0.00%
81	M	N	260	50	20	20	20	0.1	3.06	0.00%	5.26%
11	S	Z	260	50	10	10	0	4.41	0.45	2.78%	0.00%
53	S	C	260	50	20	1	0	0.12	5.33	0.00%	15.38%
10	S	DZ	260	50	10	10	0	0.96	0.45	0.00%	0.00%
18	Z	C	260	50	20	5	0.5	0.38	0.48	0.00%	0.00%
18	Z	DZ	260	50	10	10	0	2.52	0.37	7.14%	0.00%
31	Z	DZ	260	50	20	5	0.5	0.24	3.41	0.00%	18.18%
10	C	DZ	260	50	20	5	0	1.12	0.35	0.00%	0.00%
40	K	G	260	50	20	20	20	0.19	0.5	0.00%	0.00%
23	K	CH	260	50	20	2	0.1	0.32	1.53	0.00%	5.88%
10	G	CH	260	50	10	2	0.5	0.71	0.36	0.00%	0.00%
18	Š	Ž	260	50	20	5	0	0.46	0.48	0.00%	0.00%
16	Š	Č	260	50	20	10	1	2.47	12.73	2.33%	29.41%
21	Š	DŽ	260	50	10	10	0	0.34	0.5	0.00%	0.00%
21	Ž	Č	260	50	20	20	20	0.45	0.48	0.00%	0.00%
70	Ž	DŽ	260	50	20	1	1	0.09	4.15	0.00%	11.76%
29	Č	DŽ	260	50	20	1	0	0.21	0.49	0.00%	0.00%
49	Ť	Ď	260	50	20	5	0	0.13	0.5	0.00%	0.00%
10	Ť	Ň	260	50	10	2	0.5	0.65	0.28	0.00%	0.00%
10	Ť	Ľ	260	50	20	1	1	0.7	0.4	0.00%	0.00%
13	Ť	J	260	50	10	10	0	1.25	0.3	0.00%	0.00%

67	Ď	Ň	260	50	20	0.5	0.1	0.09	0.5	0.00%	5.00%
100	Ď	Ľ	260	50	20	10	10	0.06	0.81	0.00%	6.67%
100	Ď	J	260	50	20	5	0	0.06	0.96	0.00%	5.88%
13	Ň	Ľ	260	50	20	2	0.1	4.37	0.49	2.00%	0.00%
22	Ň	J	260	50	20	5	0.5	0.38	0.49	0.00%	0.00%
10	Ľ	J	260	50	20	1	1	0.95	0.47	0.00%	0.00%
10	ô	UÚ	260	50	20	20	20	0.71	0.28	0.00%	0.00%

Chybu do 6-7% považujem aspoň nateraz za prípustnú. Preto sa budem snažiť zlepšiť úspešnosti pri rozpoznávaní medzi P-T, B-V, S-C, Z-Dz, Š-Č a Ž-Dž. Pri subklasifikátore Z-Dz som uviedol dva výsledky, ktoré mi vyšli. Ťažko totiž určiť, ktorý subklasifikátor je úspešnejší. Pravdepodobne sa niektoré vzorky Z a Dz tak podobajú, že sa sieť nedokáže správne natrénovať. Nakoniec som použil subklasifikátor s nulovou chybou trénovacej množiny.

Podarilo sa mi teda pôvodných 40 tried znížiť na 15. V počte subklasifikátorov je to zníženie zo 780 na 105 v prvej vrstve. V druhej vrstve je počet subklasifikátorov v rozmedzí 0 až 10. Spolu ich je 38, takže aj keby som vyhodnocoval všetky, tak výsledný počet subklasifikátorov, oproti pôvodnému počtu sa znížil asi päťkrát.

5.2 DTW subklasifikátor

5.2.1 Testovanie

Pri testovaní úspešnosti DTW subklasifikátora som opäť normalizoval hodnoty. Použil som rovnaké trojice ako pri neurónových sieťach. Množinu všetkých vzoriek danej triedy som náhodne rozdelil na porovnávaciu a testovaciu množinu. Opäť to bolo v pomere 70% ku 30%. Pre všetky vzorky testovacej množiny som vypočítal výsledky a určil úspešnosť. Toto som urobil desaťkrát pre každý typ normalizácie. Celková úspešnosť je priemerom všetkých úspešností.

5.2.2 Výsledky

Testoval som dvojice tried druhých vrstiev, ktoré nemali nulovú chybu pri neurónových sieťach. Očakával som však trochu lepšie výsledky. DTW sub-

klasifikátory dosahovali rádovo o 10-20% horšiu úspešnosť (tabuľka 5.3) ako neurónové siete. Trochu porovnateľné výsledky sú len pri Ď-Ľ a Ž-Dž. Preto žiadny subklasifikátor založený na DTW algoritme vo finálnom klasifikátore nepoužijem.

Tabuľka 5.3: Úspešnosti DTW subklasifikátorov

Triedy		Normalizácia			Priemerná úspešnosť
P	V	20	20	5	82.38%
P	T	10	10	0	81.9%
B	V	20	20	20	65.71%
V	D	20	20	20	83.33%
M	N	10	10	0	77.89%
S	C	20	1	0	60%
Z	Dz	20	0.5	0.1	72.73%
K	Ch	20	0.5	0.1	68.12%
Š	Č	20	0.5	0.1	68%
Ž	Dž	20	20	20	80%
Ď	Ň	20	20	5	86.5%
Ď	Ľ	20	10	10	91.43%
Ď	J	20	5	0	75%

Kapitola 6

Klasifikátor

6.1 Popis vyhodnocovacích algoritmov

Pri implementácii klasifikátora som postupoval presne podľa návrhu. Skúsil som implementovať dva druhy vyhodnocovacích algoritmov:

- *winner-only* algoritmus - vyhodnotia sa všetky subklasifikátory prvej vrstvy. Pre každú triedu sa spočíta súčet $V_i^1 = \sum_{j=1, i \neq j}^{N} v_{ij}$ a vyberie sa trieda k s najvyšším súčtom V_k^1. Ak je trieda jednoprvková, rozpoznávač skončil. Inak vyhodnocuje vrstvu L_{2k}. Táto vrstva sa vyhodnocuje rovnako. Výsledkom je víťazná trieda z L_{2k} (všetky sú už jednoprvkové), tj. trieda (hláska) l s maximálnou hodnotou V_l^{2k}.

- *threshold* algoritmus - pre prvú vrstvu sa určí prah T_1 (threshold), pre druhé vrstvy prah T_2. Postupuje sa ako pri *winner-only*, ale vyhodnocujú sa všetky druhé vrstvy L_{2i}, pre ktoré platí $V_i^1 \geq T_1$. Všetky súčty druhých vrstiev V_k^{2i} sa prenásobia príslušnou hodnotou V_i^1. Triedam, ktoré nemajú druhú vrstvu (sú jednoprvkové), sa prenásobí hodnota V_i^1 samou sebou. Z takýchto súčinov sa vyberú tie ktoré prekonajú prah T_2 a trieda s maximálnym ziskom je víťazná. Ak žiadna hláska nepresiahne T_2, výstupom je hodnota UNRECOGNIZED.

Pri prvom algorime sa jednoducho stále vyberá víťazná trieda. Pri druhom je motivácia asi takáto: Môže sa stať, že dve triedy, A a B, získajú výsoký súčet v prvej vrstve. Nech súčet A je o málo väčší ako súčet B. Pri ďalšom rozhodovaní v rámci triedy A však výsledok bude neurčitý, zatiaľ čo v rámci

B vyhrá jednoznačne nejaká hláska. Pri použití prvého algoritmu vyhrá prvok triedy A, pri druhom stále môže vyhrať prvok triedy B.

6.2 Testovanie

Obidva vyhodnocovacie algoritmy som testoval na dvoch množinách dát.

Prvá množina sú náhodne vybraté vzorky z dát, na ktorých som trénoval a testoval neurónové siete. Pre každú hlásku som vybral desať vzoriek, teda dokopy 330. Na týchto dátach som chcel vyskúšať, či je výsledný klasifikátor vôbec schopný pracovať. Rovnako otestujem aj rýchlosť rozponania.

Druhá množina pozostáva zo vzoriek, ktoré som si predom odložil z nahratých vzoriek databázy. Neboli teda použité ani na trénovanie ani testovanie. Ich počet je 105, a početnosť jednotlivých hlások kolíše v rozmedzí 1 až 10. Početnosti kopírojú celkové zobierané množstvá vzoriek databázy - najviac samohlások, najmenej spoluhlások. Týmyto vzorkami budem testovať reálnu schopnosť rozpoznávať. Neočakávam veľmi dobré výsledky, lebo trénovacích vzoriek bolo primálo na dobré zovšeobecnenie. Navyše ani výsledné úspešnosti nebudú objektívne, lebo niektoré hlásky majú len jednu vzorku v testovacej množine, a preto môžu dosiahnuť len úspešnosť 0% alebo 100%.

6.3 Výsledky

Tabuľky 6.1 a 6.2 zobrazujú dosiahnuté výsledky obidvoch algoritmov na prvej testovacej množine. Hlásky, ktoré nie sú v tabuľke dosiahli 100% úspešnosť. Celková úspešnosť záhŕňa aj triedy so 100% úspešnosťou.

Pri *Threshold* algoritme bolo nutné určiť dva prahy. Pri prílíš nízkych hodnotách, sa vyhodnocujú všetky vrstvy. Na druhej strane pri vysokých hodnotách prahov je málo rozpoznaných vzoriek - väčšina výstupných hodnôt je UNRECOGNIZED. Ideálne by bolo, keby som dosiahol rovnakú úspešnosť ako pri *winner-only* a zároveň chybné rozpoznania by určil ako UNRECOGNIZED. Najlepšie výsledky som dosiahol pri $T_1 = 1.5$ a $T_2 = 2.5$.

Tabuľka 6.1: Úspešnosť *winner-only* algoritmu - prvá testovacia množina

Hláska	Rozpoznané ako	Úspešnosť
M	OÓ	90%
N	D	90%
F	AÁ, C, V	70%
G	B	90%
Ch	K	90%
Ň	J	90%
LĹ	Dz	90%
H	OÓ	90%
S	C	80%
Dz	V	90%
Č	Š	90%
Dž	B	90%
Celková úsepšnosť		94.55%
Priemerné trvanie vyhodnotenia		138ms

Tabuľka 6.2: Úspešnosť *threshold* algoritmu - prvá testovacia množina

Hláska	Rozpoznané ako	Úspešnosť
P	UNRECOGNIZED	90%
F	UNRECOGNIZED(2x), V	70%
Ch	K, UNRECOGNIZED(3x)	60%
Ť	Č	90%
Ď	Dž	90%
Ň	UNRECOGNIZED	90%
Ľ	UNRECOGNIZED	90%
LĹ	ô, UNRECOGNIZED	80%
H	OÓ	90%
S	Š, UNRECOGNIZED(2x)	80%
Dz	V	90%
Š	Č, UNRECOGNIZED	80%

Č	Š	90%
Dž	UNRECOGNIZED	90%
Celková úsepšnosť		92.42%
Priemerné trvanie vyhodnotenia		162ms

Ako vidno, *threshold* algoritmus dosiahol trošku horšie výsledky. V niektorých prípadoch sa chyba opravila (napríklad pri M), pri iných sa zhoršila. S dĺžkou trvania výpočtu[1] som veľmi spokojný. Hodnoty 138ms respektíve 162ms poskytujú minimálne ďalších 150ms na ďalšie výpočty k zlepšeniu úspešnosti.

Tabuľky 6.3 a 6.4 zobrazujú dosiahnuté výsledky obidvoch algoritmov na druhej testovacej množine. Hlásky, ktoré nie sú v tabuľke dosiahli 100% úspešnosť.

Tabuľka 6.3: Úspešnosť *winner-only* algoritmu - druhá testovacia množina

Hláska	Rozpoznané ako	Úspešnosť
P	G	50%
B	V, N	50%
V	G, B	33.3%
T	K(2x)	33.3%
D	G	0%
M	Ľ	50%
F	K	0%
Ch	K(2x)	0%
Ď	Dž(2x), Ž	40%
Ľ	P, T	33.3%
H	V, G	33.3%
S	Š(2x)	50%
Z	Ž, G	0%
C	Š, Z	33.3%
Dz	Z	50%
Ž	J, Š	0%
Č	Dž	50%

[1] počítané na procesore AMD Athlon 3000+, 1.81GHz

Dž	Ž	66.6%
Celková úsepšnosť		70.1%

Tabuľka 6.4: Úspešnosť *threshold* algoritmu - druhá testovacia množina

Hláska	Rozpoznané ako	Úspešnosť
P	T	50%
B	UNRECOGNIZED, N	50%
V	G, UNRECOGNIZED	33.3%
T	UNRECOGNIZED, K	33.3%
D	G	0%
M	Ľ	50%
F	K	0%
Ch	K(2x)	0%
Ď	Dž(4x), Ž	0%
Ľ	UNRECOGNIZED, T	50%
H	UNRECOGNIZED(3x)	25%
S	Š(2x)	50%
Z	Ž(2x)	0%
C	Š, UNRECOGNIZED, K	0%
Dz	UNRECOGNIZED	50%
Ž	UNRECOGNIZED	50%
Č	Š	50%
Dž	Ž	66.6%
OÓ	UÚ	87.5%
UÚ	UNRECOGNIZED	85.7%
EÉ	Ň	88.9%
Celková úsepšnosť		65.42%

Úspešnosť je rádovo horšia ako pri prvej testovacej množine. Toto som v podstate očakával a dôvod je nízky počet vzoriek trénovacej množiny. Úspešnosť samohlások je v oboch prípadoch skoro 100%, spoluhlásky však dopadli zle.

Kapitola 7

Záver

Očakávaným záverom je vyhodnotenie naplnenia cieľov práce. Myslím si, že obidva ciele, ktoré mala táto diplomová práca splniť, splnila.

Podarilo sa mi zozbierať databázu prehovorených písmen slovenskej abecedy. Vzorky sú v dobrej kvalite a bez výrazného šumu v pozadí. Aj keď je zatiaľ táto databáza primalá na vytvorenie úspešného klasifikátora, ktorý by pracoval nezávisle od rečníka, je to základ ktorý sa dá ďalej rozširovať.

Navrhol som a implementoval klasifikátor, ktorý rozpoznáva medzi hláskami tejto zozbieranej databázy. Je založený na BPP princípe, pričom všetky subklasifikátory sú neurónové siete. Použitie algoritmu DTW sa ukázalo ako neúspešné. Hlavnou podmienkou bolo, aby klasifikátor pracoval dostatočne rýchlo - v reálnom čase. Tento cieľ sa mi podarilo splniť. Druhou rovnako dôležitou úlohou bolo dosiahnuť čo najlepšiu úspešnosť. Ukázalo sa, že úspešnosť nie je dostatočná, ale to je hlavne spôsobené malou databázou.

Rýchlosťou klasifikácie sa utvára priestor na ďalšie zlepšovanie. Klasifikátor môže pracovať dvakrát pomalšie a stále umožňuje real-time rozhodovanie aj s prihliadnutím na čas potrebný na predspracovanie. Spôsobov vylepšenia je viacero. Prvým je samozrejme použitie väčšej databázy. Pri použití neurónových sietí by sa výsledné rozponávanie nespomalilo. Druhou možnosťou je hľadanie iného rozdelenia hlások na triedy. Možno pri väčšej databáze by vyšli niektoré triedy úspešnejšie ako je tomu teraz. Treťou možnosťou je implementácia iných subklasifikátorov. Môžu byť použité rekurentné neurónové siete, skryté Markovove modely, samoorganizujúce mapy, ...

Literatúra

[1] J. Ondriska "Pozorovanie úspešnosti rozponávania izolovaných slov rôznymi prístupmi pri telefónnej kvalite", Fakulta Matematiky, Fyziky a Informatiky, Univerzita Komenského, 2002

[2] Om Deshmukh, Carol Y., Espy-Wilson, Amit Juneja "Acoustic-phonetic speech parametes for speaker–independent speech recognition", IEEE International Conference on Volume 1, Issue, 2002 Page(s): I-593 - I-596 vol.1

[3] Stephen A. Zahorian, Zaki B. Nossair "A partitioned neural network approach for vowel classification using smoothed time/frequency features", IEEE Transactions on Volume 7, Issue 4, Jul 1999 Page(s):414 - 425

[4] F.J. Owens, R. Andonie, G.H. Zheng, A. Cataron, S. Manciulea "A comparative study of the multi-layer perceptron, the multi–output layer perceptron,the time-delay neural network and the Kohonen self–organising map in an automatich speech recognition task", in Proceedings of the EIS'98 International ICSC Symposium on Engineering of Intelligent Systems, 1998

[5] J. Psutka "Komunikace s počítačem mluvenou řečí", Academia, Praha, 1995

[6] Lingyuan Gu, Stephen A. Zahorian "A new robust algorithm for isolated word endpoint detection", IEEE International Conference on Volume 4, Issue, 2002 Page(s): IV-4161 vol.4

[7] Chuan JIA,Bo XU "An improved entropy-based endpoint detection algorithm", Chinese Academy of Sciences, 2002

[8] G.S.Ying, C.D.Mitchell ,L.H. Jamieson "Endpoint detection of isolated utterances based on a modified teager energy measurement", IEEE International Conference on Volume 2, Issue , 27-30 Apr 1993 Page(s):732 - 735 vol.2

[9] S.B. Davis, P. Mermelstein "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences", IEEE Transactions on Volume 28, Issue 4, Aug 1980 Page(s): 357 - 366

[10] P.C. Loizou ,Spanias A.S. "High–perfomance alphabet recognition", IEEE Transactions on Volume 4, Issue 6, Nov 1996 Page(s):430 - 445

[11] M. Karnjanadecha, S.A. Zahorian "Signal modeling for high-performance robust isolated word recognition", IEEE Transactions on Volume 9, Issue 6, Sep 2001 Page(s):647 - 654

[12] V. Kvasnička, Ľ. Beňušková, J. Pospíchal, I. Farkaš, P. Tiňo, A. Kráľ "Úvod do teórie neurónových sietí", IRIS, Bratislava, 1997

[13] Joe Tebelskis "Speech recognition using Neural Networks", PhD thesis, School of Computer Science, Pittsburgh, 1995

[14] H. Bourlard, N. Morgan "Connectionist Speech Recognition - A Hybrid Approach", Kluwer Academic Press, 1994.

[15] A. Kráľ "Fonetika a fonológia", Slovenské pedagogické nakladateľstvo, Bratislava, 1989

[16] SpeechDat(E) `http://www.fee.vutbr.cz/SPEECHDAT-E/`

[17] Cole R., Muthusamy, Y., and Fanty M., "The ISOLET spoken letter database", Tech. Rep. 90-004, Oregon Graduate Inst., 1990.

[18] Steve Young, Mark Gales, "The HTK Book", Cambridge University Engineering Department, 2005

[19] Wikipedia - the free encyclopedia `http://www.wikipedia.org/`

Dodatok A

Ďalšie výsledky

Tabuľka A.1: Tabuľka úspešnosti pri rozdelení PBM-TDN-V-F-MN-KGCh-ŤĎĽŇJ-SZCDz-ŠŽČDž-LĹ-RŕŔ-H-AÁ-EÉ-ôUÚ-IÍ-OÓ

Epochy	Triedy		Parametre siete		Normalizácia			Chyby			
29	PBM	TDN	260	50	10	10	0	30.492	28.733	5.71%	17.24%
45	PBM	V	260	50	10	10	0	15.503	20.211	4.04%	22.5%
56	PBM	F	260	50	20	5	0	0.104	0.625	0%	3.03%
26	PBM	KGCH	260	50	20	20	5	7.578	11.839	1.48%	7.27%
21	PBM	ŤĎŇĽJ	260	50	20	20	20	0.36	3.774	0%	1.3%
19	PBM	LL	260	50	20	5	0	4.188	0.475	1.01%	0%
17	PBM	RR	260	50	20	10	1	0.493	7.88	0%	5.88%
100	PBM	H	260	50	20	10	10	0.145	1.201	0%	2.63%
28	PBM	SZCDZ	260	50	20	5	0	4.156	0.498	0.78%	1.89%
17	PBM	ŠŽCDŽ	260	50	20	1	1	0.351	1.097	0%	1.61%
16	PBM	AÁ	260	50	20	20	5	4.536	4.848	0.62%	4.55%
17	PBM	OÓ	260	50	20	10	10	0.373	0.469	0%	1.92%
100	PBM	ôUÚ	260	50	20	5	0.5	0.053	2.297	0%	8.33%
10	PBM	EÉ	260	50	20	20	20	0.646	0.357	0%	0%
10	PBM	IÍ	260	50	20	20	20	0.64	0.267	0%	0%
58	TDN	V	260	50	20	10	10	4.061	5.118	1.03%	5%
10	TDN	F	260	50	20	5	0	0.907	0.452	0%	0%
62	TDN	KGCH	260	50	20	10	10	4.07	23.795	0.75%	14.55%

41	TDN	ŤĎŇĽJ	260	50	20	20	20	4.137	9.674	0.55%	6.49%
28	TDN	LL	260	50	20	5	0.5	0.205	0.492	0%	0%
28	TDN	RR	260	50	10	2	0.5	0.294	7.977	0%	3.92%
60	TDN	H	260	50	20	1	1	0.098	2.125	0%	2.63%
66	TDN	SZCDZ	260	50	20	5	0	7.938	7.641	1.59%	9.43%
14	TDN	ŠŽCDŽ	260	50	20	20	20	0.574	8.251	0%	6.45%
41	TDN	AÁ	260	50	20	20	5	0.143	0.577	0%	1.52%
10	TDN	OÓ	260	50	20	20	20	0.679	0.347	0%	0%
59	TDN	ôUÚ	260	50	20	10	10	0.098	0.855	0%	3.33%
10	TDN	EÉ	260	50	10	10	0	0.605	0.275	0%	0%
10	TDN	IÍ	260	50	20	20	5	0.612	0.244	0%	0%
79	V	F	260	50	20	10	10	0.077	0.499	0%	0%
22	V	KGCH	260	50	20	2	0.1	7.938	0.496	2.17%	0%
12	V	ŤĎŇĽJ	260	50	20	10	10	0.614	0.439	0%	0%
29	V	LL	260	50	20	2	0.1	0.288	0.935	0%	4.55%
10	V	RR	260	50	10	10	0	0.725	0.477	0%	0%
24	V	H	260	50	20	5	0.5	0.272	3.47	0%	10%
55	V	SZCDZ	260	50	20	10	10	4.032	4.178	1.18%	2.86%
10	V	ŠŽCDŽ	260	50	20	10	10	0.676	0.346	0%	0%
10	V	AÁ	260	50	10	10	0	0.621	0.259	0%	0%
10	V	OÓ	260	50	20	20	5	0.706	0.347	0%	0%
10	V	ôUÚ	260	50	20	20	20	0.723	0.323	0%	0%
10	V	EÉ	260	50	20	20	20	0.726	0.286	0%	0%
10	V	IÍ	260	50	20	20	5	0.647	0.277	0%	0%
17	F	KGCH	260	50	20	10	10	0.436	3.589	0%	10%
45	F	ŤĎŇĽJ	260	50	20	2	0.1	0.123	0.494	0%	0%
10	F	LL	260	50	20	1	0	0.702	0.452	0%	0%
10	F	RR	260	50	20	5	0	0.683	0.325	0%	0%
13	F	H	260	50	20	20	20	0.893	0.423	0%	0%
14	F	SZCDZ	260	50	20	5	0	0.603	0.427	0%	0%
10	F	ŠŽCDŽ	260	50	10	10	0	0.731	0.268	0%	0%
10	F	AÁ	260	50	20	0.5	0.1	0.623	0.28	0%	0%
10	F	OÓ	260	50	10	10	0	0.657	0.273	0%	0%
10	F	ôUÚ	260	50	10	10	0	0.642	0.263	0%	0%
10	F	EÉ	260	50	20	20	20	0.661	0.231	0%	0%

10	F	IÍ	260	50	20	20	5	0.66	0.284	0%	0%
13	KGCH	ŤĎŇĽJ	260	50	20	1	0	8.259	7.459	1.13%	2.7%
100	KGCH	LL	260	50	20	5	0.5	0.055	1.033	0%	5.41%
76	KGCH	RR	260	50	20	1	0	3.996	8.048	0.86%	4.17%
28	KGCH	H	260	50	10	2	0.5	4.073	3.795	1.14%	2.86%
18	KGCH	SZCDZ	260	50	20	20	5	0.445	4.244	0%	10%
86	KGCH	ŠŽCDŽ	260	50	10	10	0	0.067	4.679	0%	3.39%
10	KGCH	AÁ	260	50	20	10	10	0.638	0.292	0%	0%
10	KGCH	OÓ	260	50	20	20	5	0.652	0.323	0%	0%
10	KGCH	ôUÚ	260	50	10	2	0.5	0.655	0.388	0%	0%
10	KGCH	EÉ	260	50	20	5	0	0.631	0.29	0%	0%
10	KGCH	IÍ	260	50	20	20	20	0.625	0.261	0%	0%
11	ŤĎŇĽJ	LL	260	50	20	5	0	0.555	0.483	0%	0%
14	ŤĎŇĽJ	RR	260	50	10	2	0.5	0.593	0.281	0%	0%
28	ŤĎŇĽJ	H	260	50	20	0.5	0.1	11.77	0.494	2.19%	1.75%
25	ŤĎŇĽJ	SZCDZ	260	50	20	5	0	0.252	3.417	0%	1.39%
31	ŤĎŇĽJ	ŠŽCDŽ	260	50	20	10	1	0.321	21.256	0%	9.88%
10	ŤĎŇĽJ	AÁ	260	50	20	20	20	0.62	0.224	0%	0%
10	ŤĎŇĽJ	OÓ	260	50	20	20	20	0.639	0.243	0%	0%
10	ŤĎŇĽJ	ôUÚ	260	50	10	10	0	0.603	0.286	0%	0%
11	ŤĎŇĽJ	EÉ	260	50	20	0.5	0.1	0.647	0.471	0%	0%
20	ŤĎŇĽJ	IÍ	260	50	10	10	0	0.4	0.234	0%	0%
10	LL	RR	260	50	10	10	0	0.651	0.305	0%	0%
16	LL	H	260	50	20	20	5	0.651	0.492	0%	0%
13	LL	SZCDZ	260	50	20	10	1	0.533	0.477	0%	0%
10	LL	ŠŽCDŽ	260	50	10	2	0.5	0.716	0.301	0%	0%
10	LL	AÁ	260	50	20	10	10	0.644	0.264	0%	0%
10	LL	OÓ	260	50	20	20	20	0.895	0.433	0%	0%
98	LL	ôUÚ	260	50	20	20	5	0.061	1.277	0%	2.38%
10	LL	EÉ	260	50	20	20	20	0.656	0.271	0%	0%
10	LL	IÍ	260	50	20	5	0	0.627	0.295	0%	0%
10	RR	H	260	50	10	2	0.5	4.371	0.352	1.32%	0%
10	RR	SZCDZ	260	50	20	2	0.1	0.671	0.407	0%	0%
20	RR	ŠŽCDŽ	260	50	20	5	0	0.301	0.492	0%	0%
10	RR	AÁ	260	50	20	5	0	0.653	0.382	0%	0%

10	RR	OÓ	260	50	20	20	5	0.651	0.327	0%	0%
15	RR	ôUÚ	260	50	20	10	1	0.411	0.471	0%	0%
10	RR	EÉ	260	50	20	10	10	0.671	0.315	0%	0%
10	RR	IÍ	260	50	20	20	20	0.631	0.289	0%	0%
25	H	SZCDZ	260	50	10	10	0	7.083	0.433	2.47%	3.03%
10	H	ŠŽCDŽ	260	50	20	1	1	0.682	0.296	0%	0%
53	H	AÁ	260	50	20	10	1	0.106	0.55	0%	2.17%
10	H	OÓ	260	50	20	1	0	4.387	0.318	1.27%	0%
10	H	ôUÚ	260	50	10	2	0.5	0.626	0.334	0%	0%
10	H	EÉ	260	50	20	10	10	0.656	0.274	0%	0%
10	H	IÍ	260	50	10	2	0.5	0.635	0.274	0%	0%
23	SZCDZ	ŠŽCDŽ	260	50	20	10	10	0.315	7.128	0%	5.26%
51	SZCDZ	AÁ	260	50	20	0.5	0.1	4.021	0.707	0.68%	1.64%
10	SZCDZ	OÓ	260	50	20	5	0.5	0.616	0.317	0%	0%
11	SZCDZ	ôUÚ	260	50	20	0.5	0.1	2.009	0.438	1.54%	0%
10	SZCDZ	EÉ	260	50	20	20	5	0.646	0.274	0%	0%
10	SZCDZ	IÍ	260	50	20	20	20	0.64	0.261	0%	0%
10	ŠŽCDŽ	AÁ	260	50	20	20	5	0.614	0.276	0%	0%
10	ŠŽCDŽ	OÓ	260	50	20	20	20	0.606	0.298	0%	0%
10	ŠŽCDŽ	ôUÚ	260	50	10	2	0.5	0.59	0.28	0%	0%
10	ŠŽCDŽ	EÉ	260	50	10	2	0.5	0.588	0.265	0%	0%
10	ŠŽCDŽ	IÍ	260	50	20	10	10	0.627	0.311	0%	0%
10	AÁ	OÓ	260	50	20	10	1	0.67	0.284	0%	0%
10	AÁ	ôUÚ	260	50	20	5	0	0.605	0.258	0%	0%
10	AÁ	EÉ	260	50	20	20	20	0.619	0.243	0%	0%
10	AÁ	IÍ	260	50	20	20	20	0.593	0.272	0%	0%
100	OÓ	ôUÚ	260	50	20	2	0.1	0.054	1.388	0%	1.85%
13	OÓ	EÉ	260	50	20	10	10	0.604	0.488	0%	0%
10	OÓ	IÍ	260	50	20	20	20	0.625	0.251	0%	0%
10	ôUÚ	EÉ	260	50	10	10	0	0.586	0.258	0%	0%
10	ôUÚ	IÍ	260	50	10	10	0	0.589	0.259	0%	0%
10	EÉ	IÍ	260	50	20	20	20	0.646	0.246	0%	0%

Tabuľka A.2: Tabuľka úspešnosti pri rozdelení PBVTD-MN-F-KGChH-ŤĎĽŇJ-SZCDz-ŠŽČDž-LĹ-RŕŔ-AÁ-EÉ-ôUÚ-IÍ-OÓ

Epochy	Triedy		Parametre siete		Normalizácia			Chyby			
13	PBVTD	MN	260	50	20	10	10	0.921	21.993	0%	11.27%
13	PBVTD	F	260	50	20	20	5	0.628	1.849	0%	5.45%
24	PBVTD	KGCHH	260	50	20	1	1	34.519	44.201	4.37%	14.77%
71	PBVTD	ŤĎŇĽJ	260	50	20	20	5	0.079	0.616	0%	1.64%
58	PBVTD	LĹ	260	50	20	10	10	0.109	3.725	0%	5.26%
50	PBVTD	RŕŔ	260	50	20	10	10	0.133	4.101	0%	5.08%
13	PBVTD	SZCDZ	260	50	10	2	0.5	19.582	13.135	2.81%	5.26%
10	PBVTD	ŠŽČDŽ	260	50	20	20	20	0.578	0.248	0%	0%
13	PBVTD	AÁ	260	50	10	10	0	0.457	4.111	0%	1.39%
30	PBVTD	OÓ	260	50	10	10	0	0.189	0.494	0%	1.69%
16	PBVTD	ôUÚ	260	50	20	20	20	0.385	0.483	0%	0%
10	PBVTD	EÉ	260	50	20	10	10	0.623	0.374	0%	0%
10	PBVTD	IÍ	260	50	20	1	1	0.591	0.279	0%	0%
100	MN	F	260	50	10	10	0	0.055	2.683	0%	12.5%
26	MN	KGCHH	260	50	20	10	10	0.298	10.234	0%	5.36%
10	MN	ŤĎŇĽJ	260	50	20	5	0	0.72	0.343	0%	0%
13	MN	LĹ	260	50	20	20	20	0.58	0.465	0%	0%
37	MN	RŕŔ	260	50	20	20	20	0.19	2.341	0%	3.57%
30	MN	SZCDZ	260	50	10	10	0	0.208	3.807	0%	2.27%
10	MN	ŠŽČDŽ	260	50	20	20	20	0.665	0.266	0%	0%
20	MN	AÁ	260	50	20	20	5	0.32	0.484	0%	0%
10	MN	OÓ	260	50	20	20	20	0.772	0.374	0%	0%
10	MN	ôUÚ	260	50	20	10	10	0.739	0.302	0%	0%
10	MN	EÉ	260	50	20	20	5	0.665	0.291	0%	0%
10	MN	IÍ	260	50	20	10	10	0.657	0.261	0%	0%
27	F	KGCHH	260	50	20	20	5	0.259	1.616	0%	4.88%
10	F	ŤĎŇĽJ	260	50	20	1	0	0.832	0.397	0%	0%
10	F	LĹ	260	50	20	2	0.1	0.757	0.43	0%	0%
10	F	RŕŔ	260	50	20	1	1	0.806	0.385	0%	0%

26	F	SZCDZ	260	50	20	5	0	0.258	4.935	0%	6.9%
10	F	ŠŽČDŽ	260	50	10	10	0	0.822	0.298	0%	0%
10	F	AÁ	260	50	20	10	1	0.681	0.359	0%	0%
10	F	OÓ	260	50	10	10	0	0.764	0.378	0%	0%
10	F	ôUÚ	260	50	20	10	10	0.856	0.328	0%	0%
10	F	EÉ	260	50	20	10	10	0.645	0.255	0%	0%
10	F	IÍ	260	50	20	20	5	0.764	0.297	0%	0%
65	KGCHH	ŤĎŇĽJ	260	50	20	1	1	0.083	0.805	0%	2.17%
27	KGCHH	LĹ	260	50	20	5	0	0.247	3.368	0%	2.33%
17	KGCHH	RŔ	260	50	20	5	0	0.408	7.184	0%	11.11%
13	KGCHH	SZCDZ	260	50	10	10	0	11.989	11.643	2.07%	8.2%
10	KGCHH	ŠŽČDŽ	260	50	20	20	20	0.599	0.249	0%	0%
19	KGCHH	AÁ	260	50	10	10	0	9.233	0.46	2.92%	0%
100	KGCHH	OÓ	260	50	10	2	0.5	0.053	1.004	0%	2.22%
13	KGCHH	ôUÚ	260	50	20	0.5	0.1	0.612	0.457	0%	0%
10	KGCHH	EÉ	260	50	20	20	5	0.626	0.291	0%	0%
10	KGCHH	IÍ	260	50	20	20	5	0.645	0.283	0%	0%
10	ŤĎŇĽJ	LĹ	260	50	20	5	0	0.777	0.325	0%	0%
13	ŤĎŇĽJ	RŔ	260	50	20	20	5	0.625	0.453	0%	0%
10	ŤĎŇĽJ	SZCDZ	260	50	10	10	0	0.687	0.284	0%	0%
10	ŤĎŇĽJ	ŠŽČDŽ	260	50	20	20	20	0.699	0.279	0%	0%
10	ŤĎŇĽJ	AÁ	260	50	10	10	0	0.632	0.264	0%	0%
10	ŤĎŇĽJ	OÓ	260	50	10	10	0	0.714	0.352	0%	0%
10	ŤĎŇĽJ	ôUÚ	260	50	10	2	0.5	0.674	0.344	0%	0%
10	ŤĎŇĽJ	EÉ	260	50	10	10	0	0.636	0.26	0%	0%
30	ŤĎŇĽJ	IÍ	260	50	20	20	5	0.359	0.452	0%	0%
13	LĹ	RŔ	260	50	20	5	0.5	0.711	0.45	0%	0%
10	LĹ	SZCDZ	260	50	10	10	0	0.742	0.303	0%	0%
10	LĹ	ŠŽČDŽ	260	50	20	20	20	0.733	0.274	0%	0%
10	LĹ	AÁ	260	50	20	20	5	0.733	0.348	0%	0%
22	LĹ	OÓ	260	50	20	10	10	0.319	0.493	0%	7.14%
10	LĹ	ôUÚ	260	50	20	20	20	0.978	0.442	0%	0%
10	LĹ	EÉ	260	50	20	20	20	0.653	0.262	0%	0%
10	LĹ	IÍ	260	50	10	10	0	0.686	0.297	0%	0%
100	RŔ	SZCDZ	260	50	10	10	0	0.054	1.404	0%	9.09%

10	RŔ	ŠŽČDŽ	260	50	20	10	10	0.761	0.299	0%	0%
10	RŔ	AÁ	260	50	20	20	5	0.706	0.257	0%	0%
10	RŔ	OÓ	260	50	10	10	0	0.704	0.344	0%	0%
10	RŔ	ôUÚ	260	50	10	10	0	0.763	0.361	0%	0%
10	RŔ	EÉ	260	50	20	20	5	0.677	0.319	0%	0%
10	RŔ	IÍ	260	50	20	20	5	0.749	0.299	0%	0%
10	SZCDZ	ŠŽČDŽ	260	50	20	20	20	0.653	0.262	0%	0%
10	SZCDZ	AÁ	260	50	20	20	20	0.707	0.257	0%	0%
10	SZCDZ	OÓ	260	50	20	20	20	0.676	0.298	0%	0%
10	SZCDZ	ôUÚ	260	50	10	10	0	0.649	0.246	0%	0%
10	SZCDZ	EÉ	260	50	20	10	10	0.65	0.27	0%	0%
10	SZCDZ	IÍ	260	50	20	1	0	0.699	0.346	0%	0%
10	ŠŽČDŽ	AÁ	260	50	20	20	20	0.633	0.259	0%	0%
10	ŠŽČDŽ	OÓ	260	50	10	10	0	0.735	0.293	0%	0%
10	ŠŽČDŽ	ôUÚ	260	50	10	10	0	0.719	0.3	0%	0%
10	ŠŽČDŽ	EÉ	260	50	20	20	20	0.623	0.257	0%	0%
10	ŠŽČDŽ	IÍ	260	50	20	10	10	0.684	0.294	0%	0%
10	AÁ	OÓ	260	50	20	20	5	0.772	0.448	0%	0%
10	AÁ	ôUÚ	260	50	20	5	0.5	0.648	0.299	0%	0%
10	AÁ	EÉ	260	50	20	20	20	0.611	0.276	0%	0%
10	AÁ	IÍ	260	50	20	20	20	0.646	0.293	0%	0%
10	OÓ	ôUÚ	260	50	20	20	5	0.85	0.352	0%	0%
10	OÓ	EÉ	260	50	20	10	10	0.782	0.418	0%	0%
10	OÓ	IÍ	260	50	10	10	0	0.665	0.28	0%	0%
10	ôUÚ	EÉ	260	50	10	10	0	0.646	0.289	0%	0%
10	ôUÚ	IÍ	260	50	10	10	0	0.698	0.377	0%	0%
10	EÉ	IÍ	260	50	20	20	20	0.677	0.226	0%	0%

Tabuľka A.3: Tabuľka úspešnosti pri rozdelení PBVTDF-MN-KGChH-ŤĎĽŇJ-SZCDz-ŠŽČDž-LĹ-RŔ-AÁ-EÉ-ôUÚ-IÍ-OÓ

Epochy	Triedy		Parametre siete		Normalizácia			Chyby			
23	PBVTDF	MN	260	50	20	20	20	8.239	26.745	1.13%	13.33%

34	PBVTDF	KGCHH	260	50	20	5	0	31.158	47.182	3.69%	19.57%
10	PBVTDF	ŤĎŇĽJ	260	50	20	1	1	0.622	0.437	0%	0%
28	PBVTDF	LL	260	50	20	20	20	0.263	2.75	0%	4.84%
17	PBVTDF	RR	260	50	20	5	0	0.471	3.214	0%	3.12%
75	PBVTDF	SZCDZ	260	50	20	20	5	7.995	14.268	1.06%	7.5%
10	PBVTDF	ŠŽCDŽ	260	50	20	20	20	0.576	0.244	0%	0%
28	PBVTDF	AÁ	260	50	20	5	0	3.476	0.984	0.55%	1.3%
24	PBVTDF	OÓ	260	50	10	10	0	0.236	0.688	0%	1.56%
41	PBVTDF	ôUÚ	260	50	20	10	10	0.14	1.995	0%	1.59%
10	PBVTDF	EÉ	260	50	20	10	10	0.614	0.319	0%	0%
10	PBVTDF	IÍ	260	50	20	20	5	0.624	0.273	0%	0%
52	MN	KGCHH	260	50	20	20	20	0.162	0.495	0%	1.79%
17	MN	ŤĎŇĽJ	260	50	20	10	10	0.423	0.488	0%	0%
10	MN	LL	260	50	20	20	20	0.866	0.36	0%	0%
16	MN	RR	260	50	20	20	20	0.555	4.015	0%	14.29%
21	MN	SZCDZ	260	50	20	1	1	7.955	0.497	1.9%	0%
10	MN	ŠŽCDŽ	260	50	20	20	20	0.642	0.261	0%	0%
90	MN	AÁ	260	50	20	2	0.1	0.06	0.747	0%	2.44%
15	MN	OÓ	260	50	20	1	1	0.415	0.479	0%	0%
10	MN	ôUÚ	260	50	20	5	0	0.708	0.363	0%	0%
10	MN	EÉ	260	50	20	5	0	0.623	0.308	0%	0%
10	MN	IÍ	260	50	20	20	20	0.679	0.254	0%	0%
10	KGCHH	ŤĎŇĽJ	260	50	20	1	1	0.645	0.33	0%	0%
21	KGCHH	LL	260	50	20	20	20	0.349	3.774	0%	2.33%
16	KGCHH	RR	260	50	20	10	10	3.725	7.558	0.94%	8.89%
100	KGCHH	SZCDZ	260	50	20	20	20	11.891	5.014	2.07%	4.92%
10	KGCHH	ŠŽCDŽ	260	50	20	20	20	0.604	0.248	0%	0%
11	KGCHH	AÁ	260	50	10	2	0.5	4.432	0.47	0.73%	0%
10	KGCHH	OÓ	260	50	20	20	5	0.729	0.278	0%	0%
18	KGCHH	ôUÚ	260	50	20	5	0	0.332	0.484	0%	0%
10	KGCHH	EÉ	260	50	20	10	10	0.636	0.257	0%	0%
10	KGCHH	IÍ	260	50	10	10	0	0.606	0.292	0%	0%
10	ŤĎŇĽJ	LL	260	50	20	20	5	0.796	0.336	0%	0%
10	ŤĎŇĽJ	RR	260	50	20	10	10	0.853	0.309	0%	0%
10	ŤĎŇĽJ	SZCDZ	260	50	20	20	20	0.749	0.284	0%	0%

10	ŤĎŇĽJ	ŠŽCDŽ	260	50	20	20	20	0.725	0.262	0%	0%
10	ŤĎŇĽJ	AÁ	260	50	20	20	5	0.638	0.276	0%	0%
10	ŤĎŇĽJ	OÓ	260	50	10	10	0	0.687	0.312	0%	0%
10	ŤĎŇĽJ	ôUÚ	260	50	10	10	0	0.79	0.339	0%	0%
10	ŤĎŇĽJ	EÉ	260	50	20	20	5	0.656	0.295	0%	0%
10	ŤĎŇĽJ	IÍ	260	50	20	20	5	0.879	0.336	0%	0%
10	LL	RR	260	50	20	20	20	1.047	0.383	0%	0%
18	LL	SZCDZ	260	50	20	5	0.5	0.346	0.5	0%	0%
10	LL	ŠŽCDŽ	260	50	10	10	0	0.751	0.284	0%	0%
10	LL	AÁ	260	50	20	20	20	0.721	0.266	0%	0%
17	LL	OÓ	260	50	10	2	0.5	0.441	0.483	0%	0%
10	LL	ôUÚ	260	50	10	10	0	0.743	0.431	0%	0%
10	LL	EÉ	260	50	20	10	10	0.642	0.282	0%	0%
10	LL	IÍ	260	50	20	20	20	0.778	0.378	0%	0%
74	RR	SZCDZ	260	50	20	20	20	0.085	0.498	0%	0%
10	RR	ŠŽCDŽ	260	50	20	20	20	0.749	0.292	0%	0%
10	RR	AÁ	260	50	20	20	5	0.72	0.257	0%	0%
10	RR	OÓ	260	50	20	20	20	0.813	0.306	0%	0%
10	RR	ôUÚ	260	50	10	10	0	0.8	0.368	0%	0%
10	RR	EÉ	260	50	20	5	0.5	0.677	0.291	0%	0%
10	RR	IÍ	260	50	20	20	20	0.729	0.286	0%	0%
10	SZCDZ	ŠŽCDŽ	260	50	20	20	20	0.62	0.256	0%	0%
16	SZCDZ	AÁ	260	50	20	1	0	0.44	0.478	0%	2.17%
10	SZCDZ	OÓ	260	50	10	10	0	0.644	0.278	0%	0%
10	SZCDZ	ôUÚ	260	50	20	20	20	0.724	0.434	0%	0%
10	SZCDZ	EÉ	260	50	10	2	0.5	0.619	0.357	0%	0%
10	SZCDZ	IÍ	260	50	20	10	10	0.667	0.361	0%	0%
10	ŠŽCDŽ	AÁ	260	50	20	20	20	0.633	0.261	0%	0%
10	ŠŽCDŽ	OÓ	260	50	10	10	0	0.731	0.293	0%	0%
10	ŠŽCDŽ	ôUÚ	260	50	20	20	20	0.738	0.31	0%	0%
10	ŠŽCDŽ	EÉ	260	50	20	20	20	0.624	0.259	0%	0%
10	ŠŽCDŽ	IÍ	260	50	20	20	5	0.698	0.296	0%	0%
60	AÁ	OÓ	260	50	20	20	20	0.097	0.733	0%	3.45%
10	AÁ	ôUÚ	260	50	20	10	10	0.663	0.27	0%	0%
10	AÁ	EÉ	260	50	20	20	20	0.634	0.264	0%	0%

10	AÁ	IÍ	260	50	10	10	0	0.633	0.297	0%	0%
10	OÓ	ôUÚ	260	50	20	5	0	0.885	0.475	0%	0%
10	OÓ	EÉ	260	50	20	20	5	0.799	0.492	0%	0%
10	OÓ	IÍ	260	50	10	10	0	0.682	0.284	0%	0%
10	ôUÚ	EÉ	260	50	20	20	20	0.666	0.268	0%	0%
10	ôUÚ	IÍ	260	50	20	20	5	0.751	0.361	0%	0%
10	EÉ	IÍ	260	50	20	10	1	0.655	0.329	0%	0%

Dodatok B

Obsah CD

B.1 Abeceda

Abeceda.exe je program[1], ktorý som použil na nahrávanie vzoriek na základnej škole. Po spustení sa na obrazovke náhodne zobrazujú písmená abecedy. Nahráva sa počas držania medzerníka, po pustení sa objaví ďalšie písmeno. Po nahraní všetkých písmen, aplikácia skončí. Nahraté zvuky sa nachádzjú v adresáry *Vzorky*, v príslušných adresároch. Na nahrávanie používam knižnicu FMOD.

B.2 Databáza

V adresáry databáza sa nachádzajú všetky zozbierané a upravené vzorky, ktoré som použil na trénovanie a testovanie. V podadresáry *Testovanie* sú vzorky, ktoré boli použité ako druhá testovacia množina. V podadresáry *Nezaradené* je príklad zvukov, pri ktorých som sa nevedel rozhodnúť o akú hlásku ide.

B.3 Decider

Decider.exe je aplikácia na testovanie úspešnosti klasifikátora. Po spustení sa najskôr načítavajú subklasifikátory - trvá to asi desať sekúnd. Stlačením tlačidla *Successfulness* sa spustí testovanie na vybratej testovacej množine a s vybratými parametrami. Stlačením tlačidla *Evaluate* sa vypočíta výsledok pre jeden zvolený

[1] všetky aplikácie som programoval v prostredí Borland C++ Builder

vstupný **.mfcc* súbor. Jeho meno treba napísať do editovacieho okna vpravo hore.

B.4 Dokumenty

Súbor *Dokumenty* obsahuje niektoré práce, z ktorých som čerpal.

B.5 TestingApp

Súbor obsahuje aplikáciu vytvorenú na trénovanie a testovnaie neurónových sietí a DTW algoritmu. Do editovacieho okna treba zadať triedy, ktoré sa majú testovať. Triedy musia byť oddelené čiarkou, prvky tried musia byť oddelené medzerou a v úvodzovkách - napríklad *"P B M","T D N"*. DTW očakáva len dve triedy. Pri trénovaní neurónových sietí sa natrénuje sieť pre všetky dvojice tried. Tieto siete sú exportované v podadresáry *ExportedNN* s príponou *.nne*. Výsledky trénovania sa zapíšu do súboru v adresáry *Results*. Na trénovanie sa používajú MFCC koeficienty, ktoré sa nachádzajú v adresáry *MFCC*.

B.6 WavToMFCC

WavToMFCC.exe je jednoduchá konzolová aplikácia na výpočet MFCC koeficientov z *.wav* súborov. Očakáva dva parametre, cestu k adresáru s *.wav* súbormi a adresár kam sa majú uložiť výstupy. Na výpočet MFCC koeficientov používa *HList.exe* z knižnice HTKToolkit. Súbor *mfcc.cfg* obsahuje, konfigurčné hodnoty pre *HList.exe*.