University of Novi Sad
Faculty of Technical Sciences
Department of Power, Electronic and
Telecommunication Engineering

A Novel Method for Speaker Adaptation in Parametric Speech Synthesis

PhD thesis

Candidate: Darko Pekar

Mentors: Prof. Vlado Delić, Prof. Marko Janev

Novi Sad, 2020

Members of the jury:

Milan Sečujski, PhD (University of Novi Sad)

Nikša Jakovljević, PhD (University of Novi Sad)

Jerneja Žganec-Gros, PhD (Faculty of Industrial Engineering, Novo Mesto)

Branislav Popović, PhD (University of Novi Sad)

Marko Janev, PhD (Mathematical Institute SASA, Novi Sad)

Vlado Delić, PhD (University of Novi Sad)

Affiliations

Faculty of Technical Sciences, University of Novi Sad, Serbia

AlfaNum Speech Technologies Ltd, Novi Sad, Serbia

УНИВЕРЗИТЕТ У НОВОМ САДУ ● **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА**
21000 НОВИ САД, Трг Доситеја Обрадовића 6

КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Врста рада:	Докторска дисертација
Име и презиме аутора:	Дарко Пекар
Ментор (титула, име, презиме, звање, институција)	др Владо Делић, ред. проф, Факултет техничких наука, Нови Сад др Марко Јанев, виши научни сарадник, Математички институт САНУ, Београд
Наслов рада:	Нова метода адаптације на говорника у параметарској синтези говора
Језик публикације (писмо):	Енглески / Српски (латиница)
Физички опис рада:	Унети број: Страница 157 Поглавља 9 Референци 76 Табела 4 Слика 26 Графикона 13 Прилога 1
Научна област:	Електротехничко и рачунарско инжењерство
Ужа научна област (н. дисциплина):	Телекомуникације и обрада сигнала
Кључне речи / предметна одр.:	синтеза говора, адаптација на говорника, неуронске мреже, скривени Марковљеви модели
Резиме на језику рада:	У дисертацији је описано и упоређено неколико метода адаптације на говорника помоћу дубоких неуронских мрежа. Метода дообуке система, метода дељених и засебних слојева за различите говорнике, као и адаптација у две фазе. Последња метода као полазну тачку има систем обучен на више говорника и обучени простор говорника. Адаптација на новог говорника се одвија у две фазе: тражење оптималне тачке у простору говорника и адаптација параметара остатка мреже. Показано је да се најбољи резултати добијају коришћењем последње методе, путем поређења објективних мера, као и преко тестова слушања.
Датум прихватања теме од стране надлежног већа:	31.10.2019.
Датум одбране: (Попуњава одговарајућа служба)	
Чланови комисије: (титула, име, презиме, звање, институција)	Председник: др Милан Сечујски, ред. проф, Факултет техничких наука, Нови Сад Члан: др Никша Јаковљевић, ван. проф, Факултет техничких наука, Нови Сад Члан: др Јернеја Жганец-Грос, ред. проф, Факултет за индустријски инжењеринг, Ново место, Словенија Члан: др Бранислав Поповић, виши научни сарадник, Факултет техничких наука, Нови Сад Ментор: др Владо Делић, ред. проф, Факултет техничких наука, Нови Сад Ментор: др Марко Јанев, виши научни сарадник, Математички институт САНУ, Београд
Напомена:	

UNIVERSITY OF NOVI SAD ● **FACULTY OF TECHNICAL SCIENCES**
21000 NOVI SAD, Trg Dositeja Obradovića 6

KEY WORDS DOCUMENTATION

Document type:	Doctoral dissertation
Author:	Darko Pekar
Supervisor (title, first name, last name, position, institution)	Prof. Vlado Delić, PhD, Faculty of Technical Sciences, Novi Sad Senior research associate Marko Janev, PhD, Mathematical Institute of the Serbian Academy of Sciences and Arts, Belgrade
Thesis title:	A novel method for speaker adaptation in parametric speech synthesis
Language of text (script):	English / Serbian language (latin script)
Physical description:	Number of: Pages 157 Chapters 9 References 76 Tables 4 Illustrations 26 Graphs 13 Appendices 1
Scientific field:	Electrical and computer engineering
Scientific subfield (scientific discipline):	Telecommunications and signal processing
Subject, Key words:	text-to-speech synthesis, neural networks, hidden Markov models, speaker adaptation
Abstract in English language:	The thesis describes and compares several methods of adaptation to the speaker using deep neural networks. Simple method of system adaptation, method proposing separate layers for different speakers, as well as adaptation in two phases. The last method starts from multi-speaker model and a trained speaker space. Adaptation to a new speaker takes place in two phases: 1) searching for the optimal point in the speaker embedding space; 2) adapting the parameters of the rest of the network. It has been shown that the last approach yields the best results, by comparing objective measures, as well as by listening tests.
Accepted on Scientific Board on:	31.10.2019.
Defended: (Filled by the faculty service)	
Thesis Defend Board: (title, first name, last name, position, institution)	President: Prof. Milan Sečujski, PhD, Faculty of Technical Sciences, Novi Sad Member: Associate Prof. Nikša Jakovljević, PhD, Faculty of Technical Sciences, Novi Sad Member: Prof. Jerneja Žganec-Gros, PhD, Faculty of Industrial Engineering, Novo mesto, Slovenia Member: Senior research associate Branislav Popović, PhD, Faculty of Technical Sciences, Novi Sad Mentor: Prof. Vlado Delić, PhD, Faculty of Technical Sciences, Novi Sad Mentor: Senior research associate Marko Janev, PhD, Mathematical Institute of the Serbian Academy of Sciences and Arts, Belgrade
Note:	

Sažetak

Sinteza govora na osnovu teksta je tehnologija koja omogućava računarima pretvaranje pisanog teksta u ljudski govor. Pristupi sintezi govora mogu se podeliti u dve grupe, konkatenativne i parametarske. Parametarski pristupi su popularni zbog mogućnosti lakše modifikacije generisanog govora, ali su donedavno bili inferiorni u pogledu kvaliteta sintetizovanog glasa. Sa pojavom najnovijih metoda parametarske sinteze (npr. WaveNet), ova razlika se ne samo smanjila, već su parametarske metode u pogledu kvaliteta prevazišle konkatenativne. U ovoj disertaciji ukratko su opisane dve najpopularnije parametarske metode: sinteza govora na bazi skrivenih Markovljevih modela i sinteza govora na osnovu dubokih neuronskih mreža.

Za praktičnu primenljivost određenog sistema veoma je bitno da ne zahteva mnogo ulaganja prilikom realizacije novog glasa ili stila. Donedavno se za realizaciju novog glasa i/ili stila morala snimati potpuno nova govorna baza, često veća od 10 časova govora, i morao se uložiti dugotrajan rad na anotaciji te baze (fonetskoj i prozodijskoj). U ovoj disertaciji su predstavljene metode koje mogu da generišu novi glas ili stil koristeći mnogo manju količinu snimljenog materijala (čak i ispod jednog minuta). Jasno je da one zahtevaju i mnogo manje rada, kako na snimanju novog govornika ili stila, tako i na anotaciji.

Sve analizirane metode baziraju se na neuronskim mrežama i metodama poznatim pod zajedničkim nazivom adaptacija na govornika. Ove metode koriste početni model obučen na jednom ili više govornika, uglavnom na većoj količini materijala, koji se potom dodatno obučava (adaptira) na novog govornika korišćenjem relativno male količine materijala. Slične tehnike su korišćene još na skrivenim Markovljevim modelima, a eksperimenti izloženi u ovoj tezi obuhvataju nekoliko metoda baziranih na neuronskim mrežama. U tezi se predlaže nekoliko metoda i modela adaptacije. Sprovedena su objektivna i subjektivna poređenja svih metoda, a zatim su predstavljeni njihovi rezultati i izvučeni su odgovarajući zaključci. Takođe su predloženi i dalji pravci istraživanja koji se tiču efikasne i stabilne adaptacije na govornika i stil, u cilju brzog i isplativog generisanja novih glasova.

Abstract

Text-to-speech synthesis is the technology which enables machines to convert written text into human speech. Approaches to speech synthesis can be divided into two groups, concatenative and parametric. Parametric approaches are popular because they provide means of modifying generated speech, but until recently they have been inferior in terms of synthesized voice quality. With the emergence of new approaches in parametric synthesis (e.g. WaveNet), this gap has not only narrowed, but the situation reversed to the advantage of parametric approaches. In this thesis the two most popular methods of parametric synthesis have been described: speech synthesis based on hidden Markov models and based on deep neural networks.

For practical application of a certain system it is very important that it does not require a large effort for building of a new voice or style. Until recently, for these purposes it was necessary to record a completely new speaker database for each new voice and/or style, often containing more than 10 hours of speech data, and to invest significant effort into the annotation of that database (phonetic and prosodic). This thesis presents methods which can generate a new voice or style by using a much smaller amount of recorded material (even less than one minute). Obviously, this requires a much smaller effort both for the recording of the new speaker and the annotation.

All analyzed methods are based on neural networks and methods jointly known as speaker adaptation. These methods use initial model trained on one or more speakers, usually on large amounts of speech data, which is then additionally trained (adapted) on a new speaker by using a relatively small amount of speech data. Similar techniques have been used in the past with hidden Markov models, and in this thesis several methods based on neural networks are examined.

Several methods and models of adaptation are proposed in this thesis. Both objective and subjective comparisons of all the methods have been conducted, and the results have been presented and discussed. Future research plans have been proposed, regarding efficient and stable adaptation to a new speaker and/or style, for the purpose of fast and cost-effective generation of new voices.

Acknowledgements

The completion of multiple years of research naturally goes together with acknowledgements to all the people and organizations who have supported me, collaborated with me and advised me in multiple ways.

My first words obviously go to my mentors, Prof. Vlado Delić and Senior research associate Marko Janev. They were the ones to convince me that PhD research was the new challenge I was looking for and this thesis project would have never been launched without their invaluable help and support. While this three-party configuration may have seemed rather complex at first, it offered me an ideal research context in which I was left with much freedom to create but also advice to be reassured.

My gratitude is next addressed to each colleague at the Faculty of Technical Sciences and the company AlfaNum, who took part in this research, in some of its segments even to greater extent than myself. Special thanks go to Milan Sečujski, Siniša Suzić, Tijana Nosek, Anton Smirnov, Pieter Scholtz, Peter Glushkov, Tatjana Burčer, Marko Ćorić, Edvin Pakoci and others who did their best to make the results of this research possible.

I would also like to thank the Faculty of Technical Sciences, companies AlfaNum and Speech Morphing Systems for financing this research, providing access to speech databases and computational resources, without which this research would not have been possible.

I am thankful to all the members of the committee who contributed to the quality of this thesis with their comments and suggestions.

I owe special gratitude to my family, who has always given me unconditional support and love.

Darko Pekar

Prošireni izvod na srpskom jeziku

Sinteza govora (engl. *Text-to-Speech Synthesis* - TTS) je tehnologija sa širokim spektrom aplikacija. Koristi se za čitanje tekstualnog sadržaja za slepe, u pozivnim centrima za prenos različitih informacija korisnicima ili čak potpunu zamenu živog agenta u određenim scenarijima. Sa porastom broja pametnih telefona ova tehnologija je pronašla svoje mesto u raznim aplikacijama virtuelnih asistenata, kao i u navigacionim sistemima.

U sintezi govora trenutno dominiraju dva glavna pristupa: selekcija segmenata i parametarska sinteza. Sintetizatori koji koriste selekciju segmenata (konkatenativni) pristup biraju segmente govora iz velike baze govora i spajaju ih da bi generisali konačni niz. Parametarski pristupi sintezi govora zasnivaju se na parametrizaciji govornog signala čiji je tekstualni oblik poznat (faza analize) i razvoju modela koji može uspešno da generiše parametre za dati tekst (faza sinteze). Do nedavno, ove metode su bile inferiorne u poređenju sa selekcijom segmenata, ali su svoju primenu pronašli u mnogim aplikacijama zbog njihove fleksibilnosti i mogućnosti manipulacije osobinama generisanog govora. Sa nedavnim razvojem neuralnih vokodera, ove metode ne samo da su sustigle selekciju segmenata, nego su je čak i nadmašile [3].

Tema i glavni doprinosi

Dva glavna zahteva koja bi sintetizovani govor trebalo da ispuni su razumljivost i prirodnost [4]. U istraživačkoj zajednici postoji konsenzus da savremeni sistemi sinteze govora postižu dobre rezultate po ovim kriterijumima, ali se često naglašava da sintetizovani glas zvuči previše monotono. Drugi problem je vezan za efikasno kreiranje novih govornika i stilova. Naime, sa trenutnim pristupima, uključujući parametarski, obično je potrebno imati nekoliko sati snimka novog govornika kako bi se proizvela sinteza visokog kvaliteta. Proces izrade novog glasa ili stila zahteva snimanje novog govornika, ali i neki oblik poluautomatske anotacije i pripreme baze podataka.

Glavni cilj ovog istraživanja je da se ispita mogućnost izgradnje novih glasova (govornika sa odgovarajućim stilovima), uz istovremeno značajno smanjenje količine potrebnog govornog materijala, a samim tim i rada potrebnog za pripremu baze. Fokus će biti na korišćenju Dubokih neuronskih mreža (engl. *Deep Neural Network*, DNN) kao trenutno najnaprednijeg parametarskog pristupa. Svi pristupi će biti testirani na bazi govora sa

relativno malom količinom materijala za adaptaciju. Hipoteza koja treba da se testira je da li primenom novih metoda na ograničenu količinu novog govornog materijala mogu da se dobiju rezultati visokog kvaliteta, koji se mogu uporediti sa onima dobijenim korišćenjem velikih govornih baza.

Svi eksperimenti su rađeni na američkiom engleskom, pošto su te audio baze bile dostupne. Međutim, s obzirom da je ovo istraživanje fokusirano na jezički nezavisni deo TTS-a, koji se uglavnom bavi generisanjem signala (tzv. *back-end*), može se pretpostaviti da su rezultati primenljivi na bilo koji jezik.

Glavni doprinosi istraživanja predstavljenog u disertaciji su:

- Prilagođavanje otvorenog Merlin alata [5] za rad sa savremenijim okruženjima mašinskog učenja, kao što su TensorFlow [6] i CNTK [7].
- Pravljenje novog glasa kretanjem od modela obučenog na velikoj bazi, uz naknadnu adaptaciju na novog govornika za koji postoji mala količina materijala.
- Pravljenje multi-speaker TTS modela, koji je u stanju da generiše govor velikog broja spikera, istovremeno proizvodeći "prostor govornika" za sve govornike, koji bolje odražava sličnosti između govornika (*embedding space*).
- Pravljenje novog glasa tako što se počne od modela sa više govornika, a koristi se veoma mala količina novog audio materijala.

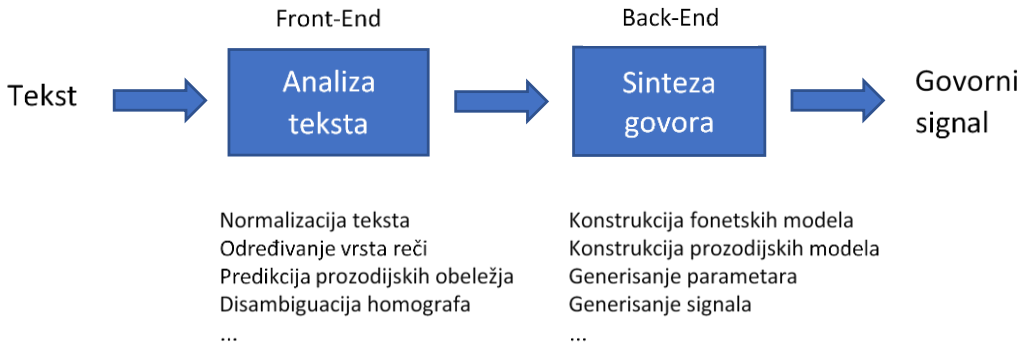
Pristupi sintezi govora

Kao što je prikazano na slici 1, tipičan sistem sinteze govora sastoji se od dve komponente: prednjeg i zadnjeg dela (engl. *front-end and back-end*). Front-end vrši analizu unetog teksta i izdvajanje informacija neophodnih za *back-end* modelovanje. To uključuje normalizaciju teksta (npr. konvertovanje brojeva u reči), određivanje tipova reči (npr. imenica, glagol, pridev), prozodijska obeležja (npr. ToBI) i disambiguaciju homografa. U ovoj disertaciji korišćena su sledeća leksička obeležja:

- Identitet trenutnog i susednih fonema (±1 and ±2 kontekst).
- Pozicija leksičkog naglaska.
- Pozicija fonema u odnosu na granicu sloga/reči/stope/fraze.
- Pozicija reči u odnou na granicu fraze.
- Broj fonema u slogu/stopi/reči.

- Broj reči u frazi/rečenici.
- Prozodijska obeležja zavisna od jezika (ToBI ili slična).

Back-end komponenta prihvata rezultate analize iz *front-enda* i kombinuje ih sa informacijama u govoru za potrebe modelovanja. Tokom procesa sinteze, *back-end* generiše izlazni govorni signal koristeći ulaz iz *front-enda* i obučene akustičke modele.



Slika 1: *Standardna aritektura TTS sistema*

Ostatak ove disertacije biće fokusiran na *back-end* komponentu i različite pristupe koji se primenjuju za njeno modelovanje.

Sinteza putem selekcije segmenata

Ovaj pristup koristi uskladištene instance govornih segmenata koje imaju različite fonetske i prozodijske realizacije. Poznat je i kao konkatenativni TTS pošto funkcioniše putem spajanja (konkatenacije) segmenata. Segmenti se skladište u bazi podataka, a zatim povezuju u skladu sa definisanim pravilima i cenama. Odgovarajući segment se bira iz baze podataka na osnovu dva tipa cene – cene do ciljanog sadržaja i cene konkatenacije.

Cena cilja (*target cost*) izražava koliko su akustička obeležja segmenta iz baze podataka slične željenim obeležjima, pošto svaka digitalna obrada koja bi se koristila za približavanje izabranog akustičnog segmenta specifikaciji može da uvede neželjenu distorziju.

Cena konkatenacije (*concatenation cost*) je mera koliko se akustičke karakteristike dva segmenta podudaraju u tačkama u kojima bi trebalo da budu spojene. Prevelike razlike bi ili bile čujne, ili bi opet bilo potrebe za dodatno digitalnom obradom signala.

Nakon inicijalnog ocenjivanja, vrši se iscrpna pretraga kako bi se iz baze govora izabrali optimalni govorni segmenti. Viterbi pretraga se često koristi za izbor segmenata iz baze, kako bi se minimizovala ukupna akumulirana cena (cena cilja i konkatenacije).

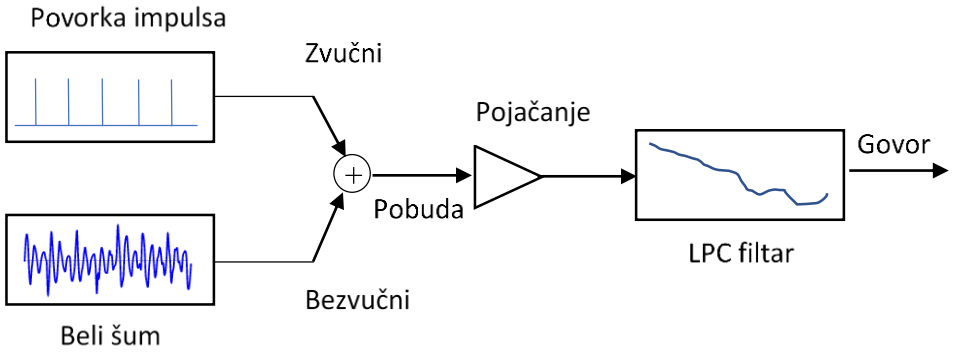
Da bi se krajnja austička obeležja bolje podudarala sa ciljnim (*f0*, energija, trajanje fonema) i da bi prelazi između segmenata bli glatki, obično se koristi neka od metoda za generisanje krajnjeg signala:

- Pitch Synchronous Overlap and Add (PSOLA)
- Frequency Domain Pitch Synchronous Overlap and Add (FD-PSOLA)
- Linear Prediction Pitch Synchronous Overlap and Add (LP-PSOLA)
- Time Domain Pitch Synchronized Overlap and Add (TD-PSOLA)
- Epoch Synchronous Non Overlap and Add (ESNOLA)

Parametarska sinteza govora

Kada je reč o sintezi govora baziranoj na modelu, posebno kada je cilj da se ovaj model nauči iz podataka, govorimo o parametarskoj sintezi govora. Model je parametarski jer predstavlja govor koristeći parametre, a ne uskladištene zvučne uzorke.

Široko korišćeni model za parametrizaciju govora je model pobude i filtra. On modeluje govor kao kombinaciju izvora (pobude), kao što je harmonijski signal iz glasnih žica ili šum iz pluća; i linearni akustički filtar, koji opisuje vokalni trakt i karakteristike zračenja sa usana. Ilustrovan je na slici 2.



Slika 2: *Pobuda-filtar model produkcije govora*

Pobuda je obično kombinacija impulsa i belog šuma. Odnos njihovih pripadajućih doprinosa izvornom signalu, koji se obično definiše po spektralnom opsegu (*band*), naziva se (a)periodičnost opsega. Frekvencija povorke impulsa definiše visinu ili osnovnu učestanost krajnjeg signala (*f0*). Akustički filtar opisuje spektralnu obvojnicu, koja daje konačni oblik spektra izlaznog signala (dok se pobuda smatra belom). Različiti fonemi mogu se razlikovati po svojstvima pobude i njihovoj spektralnoj obvojnici. Pored ovih, akustičkih, parametara

TTS sistemi moraju da parametrizuju i modeluju i trajanje fonema (ili čak subfonema – stanja).

Sistem koji može da izvrši ekstrakciju opisanih akustičkih parametara iz govornog signala (analiza) i generisanje govornig signala na osnovu parametara (sinteza) naziva se vokoder. Ovde će biti data samo kratka lista nekih od najpopularnijih vokodera koji se koriste u TTS-u: vokoder sa impulsnom pobudom (*Impulse Excitation vocoder*), mešoviti (*Multi-Band*) vokoder, čiji su najpoznatiji prestavnici STRAIGHT i WORLD, kao i neuralni vokoderi, čiji je najpoznatiji predstavnik WaveNet.

Postoje dva glavna parametarska pristupa sintezi govora: statistička parametarska sinteza govora (engl. *Statistical Parametric Speech Synthesis*, SPSS) zasnovana na skrivenim Markovljevim modelima i Gausovim smešama (engl. *Hidden Markov models - Gaussian Mixture Models*, HMM-GMM) i sinteza govora pomoću dubokih neuronskih mreža.

U sistemima koji su ispitivani u ovoj tezi koristili su se sledeći parametri govora:

- Trajanja fonema i stanja, izražena u broju blokova.
- 40 MGC koeficijenata [12] koji opisuju spektralnu obvojnicu.
- 1 ili 3 *band aperiodicity* parametra.
- Logaritam osnovne učestanosti ($\log f_0$), koji uključuje i informaciju da li je određeni segment ima harmonijski deo ili ne.

Evaluacija TTS sistema

Postoji nekoliko pristupa koji se koriste za procenu kvaliteta određenog TTS sistema. Mogu se podeliti u sledeće grupe [14]:

- objektivne mere,
- subjektivne mere,
- poređenje sistemskih karakteristika.

Objektivne mere se dobijaju direktnim poređenjem parametara izdvojenih iz prirodnog govora sa generisanim parametrima. Veliko neslaganje između generisanih parametara i parametara izdvojenih iz prirodnog govora obično je znak ozbiljnog problema u dizajnu sistema. Međutim, problem sa ovim pristupom je činjenica da vrednosti objektivnih mera nisu uvek povezane sa subjektivnim utiskom slušalaca. Moguće je da sistem generiše audio

sekvence koje se značajno razlikuju od originalnih snimaka u smislu objektivnih mera, ali i dalje zvuče sasvim prirodno i razumljivo.

Subjektivne mere se zasnivaju na ocenama koje daju živi subjekti u vezi sa određenim karakteristikama govora. Ova grupa testova može se podeliti u dve podgrupe:

1. Ocena razumljivosti,
2. Ocena kvaliteta sinteze.

Opšte je prihvaćeno da savremeni sintetizatori imaju dobru razumljivost [15], stoga se metode predstavljene u disertaciji ocenjuju preko kvaliteta sinteze. Konkretno, korišćeni su testovi srednje subjektivne ocene (engl. *Mean Opinion Score*, MOS) i višestrukih stimulansa sa skrivenom referencom i sidrom (engl. *Multiple Stimuli with Hidden Reference and Anchor*, MUSHRA) [16].

Sinteza govora korišćenjem DNN

Iako SPSS ima mnoge prednosti u odnosu na konkatenativni pristup ([38] [39] [40] [41] [42]), njeno glavno ograničenje je kvalitet sintetizovanog govora. Jedan od glavnih faktora koji pogoršava kvalitet sintetizovanog govora je tačnost akustičkih modela. Konvencionalni pristupi statističkoj parametarskoj sintezi govora obično koriste HMM zavisne od konteksta klasterizovane korišćenjem stabla odluke da bi predstavili raspodelu gustine verovatnoće govornih parametara. Ovaj pristup je prilično efikasan, ali ima nekoliko ograničenja, npr. stabla odluke nisu efikasna u modelovanju složenih kontekstnih zavisnosti. Prvo, nisu sposobna da izraze složene kontekstne zavisnosti kao što su XOR, paritet ili multipleks problemi [43]. Da bi mogla da predstavljaju i takve slučajeve, stabla odlučivanja bi morala biti izuzetno velika. Drugo, ovaj pristup deli ulazni akustički prostor što rezultira fragmentacijom podataka [44].

Stabla odluke mogu biti zamenjena sa DNN, za koju se pokazalo da bolje generalizuje ulazne podatke. Slika 20 (strana 53) ilustruje okvir za sintezu govora zasnovan na DNN. Dati tekst koji treba sintetizovati prvo se pretvara u niz ulaznih obeležja $\{x_n^t\}$, gde x_n^t označava n-to ulazno obeležje u bloku t. Ulazna obeležja mogu biti binarni odgovori na pitanja o lingvističkom kontekstu (npr. za identitet fonema: „da li je trenutni fonem M“) i numeričke vrednosti (npr. broj reči u frazi, trajanje fonema). Trajanja fonema mogu se dobiti korišćenjem odvojenog DNN-a ili se sve može generisati pomoću jedne mreže.

Zatim se ulazna obeležja preslikavaju na izlazna $\{y_m^t\}$ korišćenjem obučene DNN, gde y_m^t označava *m*-to izlazno obeležje u bloku *t*. Izlazna obeležja uključuju parametre govora i njihove izvode u vremenu. Težine DNN mogu se trenirati korišćenjem parova ulaznih i izlaznih obeležja dobijenih iz podataka za obuku, dok u fazi sinteze DNN generiše izlazna obeležja, odnosno parametre govora. Konačno, vokoder generiše talasni oblik na osnovu tih parametara.

Uvod u adaptaciju na govornika

U sintezi govora, adaptacija na govornika odnosi se na spektar tehnika kojima se TTS sistem prilagođava akustičkim karakteristikama određenog govornika koristeći mali uzorak snimljenog govora. Poslednjih godina razvoj sistema sinteze govora značajno je napredovao; međutim, ovi sistemi i dalje zahtevaju velike anotirane baze podataka (deset ili više sati govora) da bi bili dobro obučeni. Naime, konačni sistem bi trebalo da bude u stanju da proizvede bilo koji fonem u bilo kojem kontekstu (fonetskom i prozodijskom, u relativno širokom vremenskom prozoru), što podrazumeva da svaki takav akustički fenomen treba da postoji („bude viđen“) u bazi podataka za obuku. S obzirom da je broj ovih fenomena veoma velik (preko milion), praktično je nemoguće imati ih sve čak i u veoma velikoj bazi. Iz tog razloga, modeli bi trebalo da budu u stanju da dobro generalizuju i da proizvodu prihvatljive izlaze čak i za neviđene kontekste. U slučaju malih baza procenat viđenih konteksta postaje gotovo zanemarljiv, što znači da sve ostalo treba generalizovati iz tog malog uzorka i nekih prethodnih informacija.

U HMM-GMM pristupu, postojalo je nekoliko tehnika kojima se pokušava rešiti ovaj problem, od kojih su najuspešnije sledeće: procena parametara maksimizacijom aposteriorne verovatnoće (engl. *Maximum A Posteriori Parameter Estimation*, MAP) [11][23] i linearna regresija bazirana na maksimizaciji verodostojnosti (engl. *Maximum Likelihood Linear Regression*, MLLR) [12][13].

Da bi se iskoristila poboljšanja kvaliteta postignuta prelaskom na DNN, predložene su razne tehnike adaptacije na govornika. Wu et al. [58] predložili su adaptaciju na govornika koristeći i-vektore kao ulaz, prilagođavanjem doprinosa skrivenim slojevima [59], primenom izlaznih transformacija definisanih GMM-om, ili kombinacijom ovih metoda. Fan et al. [60] pretpostavili su da izlazni sloj u DNN sadrži većinu informacija o govorniku i uveli različite izlazne slojeve za različite govornike, pri čemu su ostali skriveni slojevi i dalje bili zajednički

za sve govornike. U [62], autori su obučili sistem na 135 govornika i koristili „diskriminišuće kodove“ za mapiranje početnog one-hot vektora u prostor govornika. U fazi adaptacije koristili su algoritam propagacije unazad za ažuriranje kodova govornika i minimizaciju srednje kvadratne greške predikcije koristeći malu količinu podataka koju izgovorenu od strane ciljni govornika. DNN arhitektura sa dodatnim ulazima zavisnim od govornika predložena je u [63], a ovaj pristup je dalje proširen dopunjavanjem ulaza informacijom o polu i starosti govornika [62]. Da bi se omogućilo mreži da reprodukuje glas određenog govornika u stilu koji nije prisutan u korpusu za obuku, autori su u [64] predložili mrežnu arhitekturu koja eksplicitno razdvaja doprinose govornika i stila, dok je model predstavljen u [65] izgrađen na DNN sa više govornika sa deljenim skrivenim slojevima predloženim u [60], proširujući ga jednim ulazom koji zavisi od stila i uvodeći dodatni uski sloj (*bottleneck*). Drugi pravci istraživanja, poput onog predstavljenog u [66], fokusirali su se na razvoj metoda za prilagođavanje višeslojnog DNN sa jednim govornikom, glasu novog govornika.

U ovoj tezi predstavljamo dve metode za efikasno stvaranje novih TTS glasova, zasnovane na relativno maloj količini podataka o adaptaciji. Jedna metoda u početku obučava TTS zasnovan na DNN na relativno velikoj količini materijala za obuku (3+ sata) i koristi taj model kao polaznu tačku za adaptaciju. To znači da novi model nije obučen na slučajno inicijalizovanoj, već na već prethodno obučenoj mreži, što je rezultiralo mnogo boljim performansama (veći kvalitet sintetizovanog govora). Drugi pristup, predlaže stvaranje početnog modela sa više govornika i odgovarajućeg prostora govornika (*embedding*). Tokom adaptacije izvode se dve faze. U prvoj fazi se traži optimalna tačka u prostoru govornika za novog govornika, sa idejom da se generiše govor koji već liči na njega, pa su u drugoj fazi potrebne samo minimalne promene DNN-a. U drugoj fazi je fiksirana pronađena tačka u prostoru govornika, a ostatak DNN je prilagođen na isti način kao u prvom pristupu. Ovaj dvofazni pristup dao je još bolje rezultate i može stvoriti glasove s količinom materijala od samo 30 sekundi.

Merlin: DNN TTS sistem otvorenog kôda

2016. godine Centar za istraživanje govornih tehnologija Univerziteta u Edinburgu objavio je sopstveni skup alata otvorenog koda za razvoj TTS-a zasnovanog na DNN. Poput HTS-a [47], Merlin nije kompletan TTS sistem. Pruža osnovne funkcije akustičkog

modelovanja: vektorizaciju lingvističkih obeležja, normalizaciju akustičkih i lingvističkih obeležja, obuku akustičkih modela neuronske mreže i generisanje parametara govora. Napisan je na Python programskom jeziku, i baziran na biblioteci Theano, a tim kompanije AlfaNum i Fakulteta tehničkih nauka (AN-FTS) obezbedili su da radi i sa CNTK [7] i TensorFlov [6] okvirima za duboko učenje. AN-FTS tim je takođe unapredio tehniku poravnanja foenma, implementiranu u osnovnu verziju alata.

Adaptacija sa početnog na ciljanog govornika

Metoda koristi podatke koji odgovaraju ciljnom govorniku za potrebe doobuke DNN koja je već obučena za TTS zadatak na početnom govorniku. Dakle, započinjemo obuku sa početnim vrednostima parametara prethodno obučene mreže, umesto nasumično inicijalizovanim.

Korišćena je standardni pristup preko dve mreže, za trajanja fonema (tj. stanja) i akustičke parametre, kako je to ranije opisano. Korišćena su 554 binarna leksička obeležja, pomeraj bloka od 5 ms, 5 stanja po fonemu, MLPG, a za generisanje krajnjeg signala je korišćen WORLD vokoder [13].

Obe mreže imaju 4 skrivena sloja i 1024 neurona po sloju sa *tanh* funkcijom aktivacije. Prva tri sloja su *feed-forward*, dok je poslednji skriveni sloj LSTM tipa, a izlazni sloj je linearan (bez aktivacione funkcije). Dodatna normalizacija obeležja se izvodi za ulaz (normalizovano na interval [0, 1]), kao i izlazna obeležja (normalizovana tako da imaju nultu srednju vrednost i jediničnu varijansu). Ciljna funkcija koja se koristi je srednja kvadratna greška.

Predložena metoda omogućava bržu i ekonomičniju TTS adaptaciju, jer ne zahteva postojanje modela prosečnog govornika kao u konvencionalnim metodama prilagođavanja govorniku, a istovremeno zahteva mnogo manje podataka u poređenju sa obukom DNN-TTS modela od nule. Uticaj izbora početnog modela na predloženu metodu prilagođavanja takođe je predmet istraživanja.

Rezultati eksperimenata

U ovom odeljku upoređujemo predloženi model sa osnovnim modelom na zadatku stvaranja novog TTS glasa. Model se ocenjuje na skupu rečenica koje se sintetišu na osnovu fonetskih i prozodijskih informacija preuzetih iz originalnih izgovora. U svim prikazanim

eksperimentima snimci su podeljeni u deo za obuku, validaciju i test. Za sve eksperimente korišćeni su isti skupovi koji su se sastojali od 5 ili 10 izgovora. U svakom eksperimentu, slučajno je izabrano 10% materijala koji se koristio za validaciju, dok je ostatak korišćen za trening.

Rađena je objektivna i subjektivna evaluacija rezultata. Za objektivnu evaluaciju korišćene su ranije objašnjene metode i parametri. Za subjektivnu procenu sprovedena su dva MUSHRA testa [69]. U oba su učestvovala 22 subjekta u kontrolisanom okruženju i sa kvalitetnim slušalicama. Svaki ispitanik je procenio određeni broj testnih izgovora upoređujući ih sa referentnim (originalni snimak), pri čemu je svaki put jedna od test rečenica bila identična referentnoj. Izgovori su ocenjeni u smislu ukupnog kvaliteta (razumljivost i prirodnost). Svaki snimak je dobio ocenu od 0 do 100, sa jednim ograničenjem - jedna od 5 rečenica je morala da dobije ocenu 100. Izračunate su prosečne ocene i korišćen je t-test kako bi se proverile statistički značajne razlike u srednjim vrednostima.

Tačnost poravnanja

Kad postoji dovoljna količina podataka, standardno poravnanje zasnovano na monofonima postiže zadovoljavajuću tačnost. Međutim u situacijama kada je dostupno znatno manje podataka, ranije opisani metod postiže bolje rezultate, što je prikazano na slici 25 (strana 72). Slika predstavlja procenat fonema čija su granična odstupanja ispod određenog praga u poređenju sa ručno postavljenim granicama. Takođe, uticaj metode poravnanja na objektivne mere odgovarajućeg TTS modela predstavljen je u tabeli 1 (strana 72), gde se može videti da je predloženo poravnanje postiglo gotovo iste rezultate kao i obuka sa konvencionalnim poravnanjem kada baza ciljnog govornika sadrži 10 ili 15 minuta govora, ali znatno bolji rezultat kada baza sadrži samo 3 ili 5 minuta materijala. Zbog toga je u svim eksperimentima početno poravnanje izvedeno predloženom metodom, dok se za potrebe obuke početnog modela koristila konvencionalna metoda.

Prvi skup eksperimenata

Osnovni modeli su se nasumično inicijalizovali i obučili koristeći 5, 10, 15, 30, 60 i 180 minuta podataka muškog govornika. Predloženi modeli su napravljeni polazeći od modela prethodno obučenog na 3 sata materijala ženskog govornika, a zatim ga prilagodili muškom

govorniku koristeći 3, 5, 10 i 15 minuta govora. Kao što se može videti na slici 26 (strana 73), sve objektivne mere, sa izuzetkom VUV, pokazuju da je kretanjem od modela koji je već obučen dovoljno 15 minuta govora ciljnih govornika da bi se postigao kvalitet dobijen započinjanjem od nasumično inicijalizovanog modela i obučavanjem na 30 minuta materijala (videti npr. MCD na slici 26 (strana 73) a). Takođe, kretanjem od obučenog modela, dovoljno je 5 minuta govora da se postigne ili nadmaši kvalitet dobijen treningom nasumično inicijalizovanog modela na 15 minuta.

Iako se 50% manje matrijala potrebnog za postizanje istog kvaliteta može smatrati dobrim rezultatom, nezadovoljavajuće je što 15 minuta ciljnih podataka još uvek nije dovoljno za pretvaranje već obučenog modela u model sposoban da proizvede govor kvaliteta uporediv sa modelom obučenim na 3h govora (i treniranim od nule).

S obzirom na to da objektivne mere ne odražavaju u potpunosti subjektivnu percepciju, izvršeni su dodatni testovi slušanja. Uključeno je 10 rečenica u kojima je korišćen originalni snimak, zajedno sa 4 snimka sintetizovana korišćenjem 4 različita sintetizatora navedena u tabeli 2 (strana 74). Sintetizatori predstavljeni u tabeli predstavljaju podskup svih sistema prikazanih na slici 26 (strana 73), dok su rezultati testova slušanja predstavljeni na slici 27 (strana 75).

Vidi se da model obučen sa 10 minuta materijala predloženom metodom zvuči blisko modelu obučenom na 1h materijala počevši od nule. Njihove prosečne ocene bliske su ocenama modela obučenog na 3h materijala (sa t-testom $\alpha = 0{,}05$). Iako se čini da je prosečna ocena sinteizatora 1.4 takođe blizu ostalih, t-test pokazuje statistički značajnu razliku. Stoga se može zaključiti da 3 minuta ciljnog govornika pružaju zadovoljavajuće rezultate, ali još uvek se ne može očekivati da sintetizovani govor zvuči kao govor sintetizovan modelom obučenim na relativno velikoj bazi podataka.

Drugi skup eksperimenata

U drugom nizu eksperimenata ispitujemo uticaj početnog modela. Izvršene su adaptacije u okviru istog pola i između različitih polova, kretanjem od modela obučenog na 3h materijala u svim slučajevima, a adaptacija je vršena sa 3, 5 ili 10 minuta materijala ciljnog govornika. Na slici 28 (strana 76) date su objektivne mere za ovaj skup eksperimenata. Moglo bi se zaključiti početni model nije od velike važnosti.

Takođe je izvršeno poređenje spomenutih adaptacija preko subjektivnih testova slušanja, predstavljenih u tabeli 3 (strana 76). Test slušanja obuhvatio je 10 rečenica, od toga polovinu ženskih i polovinu muških govornika. Za svaku od rečenica korišćeni su originalni snimci i još četiri sintetizovana modelima navedenim u tabeli 3.

Rezultati su predstavljeni na slici 29 (strana 77). Može se videti da kada je ciljni govornik bio ženskog pola (slika 29a), adaptacija sa 10 minuta podataka ciljnog govornika daje bolje rezultate ako je umesto ženskog korišćen muški početni model. Međutim, ako se koristi samo 3 minuta materijala za adaptaciju, rezultati i za muški i za ženski početni model su gotovo isti. S druge strane, kada je ciljni govornik bio muškarac (slika 29b), adaptacija sa samo 3 minuta, počevši od muškog početnog modela, postiže bolje rezultate od adaptacije sa 10 minuta, počevši od ženskog početnog modela. Možemo zaključiti da, koristeći ograničene raspoložive resurse, početni model ima određeni, mada ne i značajan uticaj na adaptaciju.

Adaptacija na govornika u dva koraka

Ideja je da se prvo obuči model na više govornika i više stilova (engl. *Multi Speaker Multi Style* – MSMS) kako bi se dobio dobar polazni model za adaptaciju i takođe stvorio „ugrađeni“ prostor govornika, slično kao u [18]. Korišćeni model, ulazni i izlazni parametri su isti kao i kod prethodne metode. Ugrađivanje je moćna tehnika dubokog učenja zasnovana na mapiranju diskretnih (često binarnih) vektora iz prostora velike dimenzionalnosti do vektora kontinualnih vrednosti u prostoru male dimenzionalnosti. U kontekstu sinteze govora, i govornik i govorni stil tradicionalno su predstavljeni kao vektori sa jednim nenultim elementom (*one-hot*), što se može smatrati suboptimalnim, jer sličnost dva glasa nije ni na koji način povezana sa rastojanjem između odgovarajućih tačaka u visoko-dimenzionalnom prostoru [21]. Ovaj nedostatak se prevazilazi izgradnjom zajedničkog ugrađenog prostora govornika i stila, predstavljajući ih u prostoru male dimenzionalnosti na intuitivniji način, što pomaže mreži da efikasno generalizuje neviđene govornike i stilove.

Sa idejom poboljšanja modela sa više govornika kao polazne tačke za prilagođavanje novom govorniku i stilu, mi nadopunjavamo ulaze obe neuronske mreže informacijama o govorniku, stilu govora i klasteru (engl. *Speaker Style Cluster* - SSC) u ugrađenom obliku, kao što je prikazano na slici 30 (strana 82). Na ovaj način se prepušta mreži da predstavlja određeni SSC u prostoru niže dimenzije (u našem istraživanju broj SSC je 67, a dimenzija ugrađenog prostora je N = 15). Kada bude obučena, mreža će moći da sintetiše govor koji

odgovara određenom SSC-u s obzirom na odgovarajuću tačku u ugrađenom prostoru. Pored toga, ukoliko se odabere slučajna tačka u ugrađenom prostoru, mreža će moći da proizvede novi, prethodno „neviđeni“ glas.

Arhitektura i postupak obuke predloženi u ovom istraživanju rezultiraju u multi-govorničkom modelu za sintezu govora u više stilova, koji može reprodukovati veoma kvalitetan govor u bilo kojoj kombinaciji govornika / stila / klastera prisutnog u početnom korpusu za treniranje, ali je takođe lako prilagodljiv novom govorniku i stilu, uz relativno malu količinu podataka za adaptaciju.

Procedura adaptacije u dva koraka

Prva faza ima za cilj nalaženje tačke u ugrađenom prostoru za novog govornika i stil, a započinje nasumičnom inicijalizacijom vrednosti u ugrađenim slojevima obe mreže. U ovoj fazi adaptacije samo se vrednosti u ugrađenim slojevima ažuriraju tokom obuke, dok se ostatak mreže ne menja. Model sa ugrađenim slojevima prilagođenim na ovakav način može da sintetiše govor koji u određenoj meri već liči na ciljnog govornika i stil. Međutim, kvalitet sintetizovanog govora može se dalje poboljšati kroz drugu fazu prilagođavanja, u kojoj se ponovo koriste isti podaci za trening, ali je ugrađeni sloj zamrznut, dok se težine u mrežama modifikuju.

Podaci

Podaci korišćeni za izradu modela sastoje se od ukupno 8 sati i 38 minuta govora 6 govornika sa američkog govornog područja, čiji količina varira u broju stilova govora kao i akustičkom kvalitetu, što je prikazano u tabeli 4 (strana 87). Dva govornika čiji doprinosi obuhvataju najveći broj stilova govora i čiji je doprinos neutralnom stilu najveći ćemo obeležiti sa M1 i F1. Kako bi se izbegla pristrasnost modela prema M1 i F1, kao i povećala osnova za model sa više govornika, raspoloživi snimci veštački su umnoženi uvođenjem promena u visini, brzini i spektralnoj obvojnici kod svih 6 inicijalnih govornika. Koristeći različite delove originalnog korpusa, kao i umnožene izgovore, stvoreno je 10 novih veštačkih govornika, čime je ukupan broj porastao na 16 (sa 67 jedinstvenih kombinacija govonika / stila / klastera) i ukupno trajanje govornog korpusa na 21 sat i 50 minuta.

Da bi se procenila sposobnost sistema da se prilagodi novom govorniku i stilu, korišćena su dva relativno mala korpusa, od kojih jedan sadrži govor ženskog govornika (F4), a drugi

muškog (M4). Oba ova korpusa su izuzeta iz obuke MSMS modela. Stil govora u ova dva korpusa može se nazvati grubo neutralnim, mada su ove informacije zapravo nevažne, pošto je model u stanju da se prilagodi nepoznatom govorniku, ali takođe i nepoznatom stilu.

Osnovne metode

U našim eksperimentima upoređivali smo performanse predloženog postupka adaptacije u dva koraka sa dve osnovne metode. Prva metoda koja se koristi kao osnovna metoda (bazna metoda 1), predstavljena je ranije (Adaptacija sa početnog na ciljanog govornika). Za potrebe ovog istraživanja, govorni materijal dva govornika, M1 i F1, korišćen je za dobijanje dva TTS modela zavisna od govornika (engl. *Speaker Dependent TTS* – SD TTS), koji su poslužili kao osnova za prilagođavanje govornicima M4 i F4.

Druga osnovna metoda (bazna metoda 2) predstavlja malu modifikaciju pristupa detaljno opisanog u [32], gde se koristi „odvojeni izlazni sloj“. Ovaj pristup se temelji na ideji predstavljenoj u [33], koja predlaže arhitekturu zasnovanu na deljenim skrivenim slojevima i višestrukim izlaznim slojevima (za svakog govornika po jedan). U ovom pristupu pretpostavlja se da deljeni deo mreže modeluje globalnu jezičku transformaciju, dok se zasebni izlazni slojevi koriste za različite kombinacije govornik / stil. U fazi adaptacije prilagođava se samo određeni izlazni sloj koji zavisi od govornika / stila, koristeći raspoložive podatke o govorniku / stilu, prateći postupak adaptacije predložen u [33]. Modifikacija u odnosu na [32] leži u uvođenju dodatnog skrivenog sloja koji zavisi od govornika / stila. Slično kao u slučaju sa osnovnim modelom 1, ulazi se proširuju sa stilovima i klaster kodovima u obliku *one-hot* vektora, ali u ovom slučaju su se svi govorni podaci koristili za obuku MSMS modela koji je kasnije prilagođen M4 i F4.

Eksperimenti

U ovom istraživanju predloženi model se obučava na istim podacima kao i dva osnovna modela opisana u prethodnom odeljku. Međutim, dok su modeli sa više govornika (osnovni model 2 i predloženi model) bili obučavani na celoj govornoj bazi, osnovni model 1 (koji je SD TTS) obučavan je samo na M1 i F1 kako bi se napravila dva modela zavisna od govornika. Da bi se testirala sposobnost sva tri modela za prilagođavanje nepoznatom govorniku i nepoznatom stilu, za adaptaciju su korišćeni snimci govornika M4 i F4. Budući da je cilj ovog istraživanja ispitati slučaj kada je količina ciljnog govornog materijala vrlo

mala, eksperimenti su izvedeni na bazama koje sadrže 10 minuta i samo 30 sekundi. Za prilagođavanje osnovnog modela 1, korišćen je inicijalni model istog pola zavisan od govornika. Kako osnovni model 2 obuhvata 16 različitih govornika, oni koji su korišćeni kao polazišta za adaptaciju u ovom istraživanju su oni koji odgovaraju M1 ili F1 (u zavisnosti od pola ciljnog govornika). U predloženom modelu, dimenzija ugrađivanja je postavljena na N = 15, mada je pokazano da je od iznenađujuće malog značaja za performanse sintetizatora (testirane su vrednosti u rasponu od 4 do 40). Sposobnost predloženog modela da sintetizuje govor koji odgovara predviđenom govorniku / stilu, najpre je procenjena preko objektivnih mera, nakon čega je usledio niz testova slušanja posebno usmerenih na utvrđivanje relevantnosti položaja SSC tačaka u svakom od dva ugrađena prostora, relevantnosti svake faze u dvofaznom procesu prilagođavanja, kao i količine podataka za adaptaciju.

Objektivne mere

Rezultati su predstavljeni na slici 33 (strana 92). Može se videti da korelacija između generisane f0 krive i stvarne, kao i korelacija između generisanog trajanja fonema i stvarnog, pokazuje samo male razlike između tri modela, ali da predloženi model konzistentno postiže najbolje performanse, bez obzira da li je za prilagođavanje korišćeno 10 minuta ili 30 sekundi govora. Takođe se može primetiti da su razlike nešto veće u slučaju prilagođavanja na manje ciljnog materijala. Izgleda da je osnovni model 1 najosetljiviji na smanjenje količine podataka za adaptaciju, mada razlike ni u ovom slučaju nisu značajne. Razlike između modela su mnogo očiglednije u slučaju RMSE f0 i trajanja fonema. U većini slučajeva predloženi model nadmašuje dva osnovna modela, a osnovni model 1 je najmanje uspešan. Razlike među modelima su opet vidljivije u slučaju manjeg skupa podataka za prilagođavanje.

Subjektivna evaluacija

Sproveden je niz testova slušanja kako bi se potvrdili rezultati objektivne procene i utvrdio uticaj različitih faktora na kvalitet sintetizovanog govora nakon što se početni model prilagodi ciljnom govornom materijalu.

Eksperiment 1

Cilj ovog eksperimenta bio je da se istraži uticaj položaja SSC tačaka u svakom od dva ugrađena prostora na stepen sličnosti sintetizovanog govora i ciljnog govornika. Dalje,

eksperiment takođe ilustruje pozitivan efekat druge faze procesa adaptacije, za koji se pokazalo da povećava sličnost sintetizovanog govora sa planiranom kombinacijom govornik / stil. Eksperiment istražuje samo predloženi model i ne uključuje poređenje sa osnovnim modelima.

Eksperiment je postavljen kao MUSHRA test slušanja, a učestvovalo je 26 slušalaca. Svakom slušaocu je predstavljeno 10 zadataka, uključujući 5 rečenica izgovorenih glasovima 2 govornika (M4 ili F4). U svakom zadatku, slušaocima je predstavljeno sledećih 5 verzija iste rečenice, nasumičnim redosledom:

- Skriveni snimak reference (originalni snimak izvornog govornika);
- Sinteza nakon prve faze adaptacije sprovedene na početnom modelu;
- Sinteza nakon što je izvršena prva faza adaptacije, a zatim je dobijeni ugrađeni vektor modifikovan za 10%;
- Sinteza nakon što je izvršena prva faza adaptacije, a zatim je dobijeni ugrađeni vektor modifikovan za 20%;
- Sinteza nakon oba faze adaptacije izvršene na početnom modelu bez modifikacije ugrađenih vektora dobijenih u prvoj fazi.

U ovom eksperimentu prilagođavanje je izvršeno korišćenjem 10 minuta ciljanih govornih podataka. Slušaoci su zamoljeni da ocene sličnost govornika između reference i svakog od 5 primera na skali od 0 do 100. Kako slušaoci imaju tendenciju da daju niže ocene manje privlačnim glasovima, što bi prikrilo uticaj faktora koji su smatrani kao relevantni za ovaj eksperiment, ocena koja je data skrivenoj referenci je skalirana do maksimalne ocene, a ostale ocene su skalirane u skladu sa tim. Nadalje, da bi se pojednostavila poređenje rezultata u svim eksperimentima, sve ocene su prikazane kao ponovo postavljene na interval 0-5.

Rezultati, prikazani na slici 34 (strana 95), pokazuju da je prva faza adaptacije sama po sebi dovoljna da model proizvede govor koji u određenoj meri liči na glas ciljnog govornika. Takođe je pokazano da je položaj ugrađenih vektora dobijen početnom obukom modela relevantan, jer ako se modifikuje, gubi se sličnost sa ciljnim govornikom. Eksperiment je takođe pokazao važnost druge faze adaptacije, jer je ocena dobijena nakon obavljene obe faze adaptacije značajno veća od bilo koje ocene dobijene posle same prve faze. Još uvek postoji relativno široka margina između sintetizovanog i originalnog govora, i vrlo je

verovatno da je to posledica relativno slabe pokrivenosti prostora za ugradnju od strane SSC-a koji postoje u trenažnom korpusu.

Eksperiment 2

Cilj ovog eksperimenta bio je da se kvalitet sinteze predloženog modela uporedi sa dva osnovna modela nakon adaptacije, bez obzira na sličnost govornika u odnosu na referencu, kroz MUSHRA test slušanja sa 24 učesnika. U svakom od 20 zadataka, slušaoci su obavešteni da referenca, označena kao takva, predstavlja snimak prirodnog govora, i od njih je zatraženo da ocene razumljivost i prirodnost, a zanemare sličnost govornika, sledeće verzije iste rečenice, koje su se pojavljivale nasumičnim redosledom:

- Skriveni referentni snimak (originalni snimak govornika);
- Sinteza po osnovnom modelu 1 nakon adaptacije;
- Sinteza po osnovnom modelu 2 nakon adaptacije;
- Sinteza po predloženom modelu nakon što se ugrađeni vektor resetuje na 0 i sprovede se samo druga faza prilagođavanja;
- Sinteza predloženog modela nakon obe faze adaptacije.

Od 20 zadataka, 10 je prilagođeno korišćenjem 10 minuta ciljnih govornih podataka, a preostalih 10 korišćenjem samo 0,5 minuta ciljnih govornih podataka. U svakom od ova dva slučaja bilo je po 5 izgovaranja od strane svakog od dva govornika (M4 i F4).

Rezultati, prikazani na slici 35 (strana 97), pokazuju da su, bez obzira na količinu ciljnih govornih podataka koji su korišćeni za adaptaciju, slušaoci smatrali da je osnovni model 2 najmanje uspešan, dok su dve verzije predloženog modela dobili najviše ocene. Zanimljivo je napomenuti da, iako razlika između prosečnih ocena za osnovni model 1 i predloženi model nije značajna u slučaju kad je korišćeno 10 minuta materijala, predloženi model značajno nadmašuje osnovni model 1 u slučaju kad se prilagođenje vrši sa samo 0,5 minuta govornih podataka. Još jedna zanimljivost koja se odnosi na predloženi model je da, ako se ugrađeni vektor dobijen u prvoj fazi adaptacije resetuje na 0 i izvrši se samo druga faza, to ne umanjuje značajno kvalitet sinteze.

Eksperiment 3

Postavke eksperimenta 3 bile su potpuno iste kao u slučaju eksperimenta 2, ali ovog puta od slušalaca je traženo da procene sličnost govornika umesto opšteg kvaliteta sinteze.

Eksperiment se sastojao od 10 zadataka (5 za svakog od dva govornika, M4 i F4), a učestvovalo je 20 slušalaca. Kako bi se slušaoci fokusirali na sličnost govornika, adaptacija je rađena samo sa 10 minuta materijala (eksperiment 2 je pokazao da kvalitet sinteze značajno opada kod nekih modela, kada je količina materijala jako mala). Kao što se može videti na slici 36 (strana 98), predloženi model nadmašuje oba modela po pitanju generisanja glasa koji podseća na izvornog govornika, čak i u slučaju kad je ugrađivanje resetovano na 0 i izvršena samo druga faza prilagođavanja.

Eksperiment 4

Opširnija evaluacija performansi predloženog modela uključivala bi njegovu poređenje sa drugim osnovnim SD TTS modelom, koristeći ne samo male već i velike količine snimaka govornika za obuku. Međutim, nismo bili u mogućnosti da direktno izvršimo takvu procenu zbog dostupnosti samo male količine podataka za govornike M4 i F4, imajući u vidu da su svi preostali dostupni govornici već korišćeni za obuku inicijalnog modela. Ovaj eksperiment predstavlja pokušaj zaobilaženja ovog ograničenja uključivanjem dve vrste zadataka MUSHRA (10 zadataka svake vrste). U obe vrste zadataka, 32 učesnika u testu slušanja obavešteni su da je referentni izgovor zapravo snimak prirodnog govora, a zadatak je bio da procene opšti kvalitet 3 izgovora datih slučajnim redosledom. U zadacima tipa 1 ponuđena su sledeća 3 izgovora:

- Skriveni referentni snimak (originalni snimak M1 ili F1);
- Sinteza po osnovnom modelu 1 obučena na svim raspoloživim podacima za M1 ili F1, bez daljeg prilagođavanja;
- Sinteza po predloženom modelu, pomoću ugrađenih vektora koji odgovaraju M1 ili F1, bez daljeg prilagođavanja;

dok su zadaci tipa 2 uključivali sledeća 3 izgovora:

- Skriveni referentni snimak (originalni snimak M4 ili F4);
- Sinteza predloženog modela nakon obe faze adaptacije na M4 ili F4, koristeći 10 minuta podataka ciljnih govornika;
- Sinteza predloženog modela posle obe faze adaptacije na M4 ili F4, korišćenjem 0,5 minuta podataka ciljanih govornika.

U svakom zadatku sve 3 rečenice odgovaraju istom govorniku kako bi se eliminisala preferenca koju slušalac može imati prema nekom od glasova. Svi govornici su bili jednako zastupljeni tokom eksperimenta, tj. svaki od njih se pojavio u 5 zadataka.

Rezultati eksperimenta, sa rezultatima skaliranim na interval 0-5, prikazani su na slici **37** (strana 99). Pre nego što se donesu bilo kakvi opšti zaključci, treba primetiti da iako se M1 i F1 nisu pojavljivali u istim zadacima kao M4 i F4, još uvek je moguće uporediti perceptivni kvalitet sinteze između modela i/ili verzija koji se nisu pojavili u istim zadacima. Sinteza bazirana na osnovnom modelu 1 obučena na svim raspoloživim podacima za M1 ili F1 bez daljeg prilagođavanja i sinteza predloženog modela nakon dvofazne adaptacije na M4 ili F4, koristeći 10 minuta podataka (stavke (a) i (c) na slici **37**) su ocenjene sličnim ocenama. To pokazuje da predloženi model, kada krene od dobro obučenog MSMS modela, i koristi samo 10 minuta adaptacionog materijala, može postići kvalitet sinteze uporediv sa onim koji ima standardni SD TTS model obučen na mnogo više audio materijala (~3,5 sata u slučaju M1 i ~2,5 sata u slučaju F1). Dalje, sinteza dobijena osnovnim modelom 1 obučena na svim raspoloživim podacima za M1 ili F1 smatra se da je istog kvaliteta kao i sinteza MSMS modela, koristeći ugrađene vektore koji odgovaraju M1 ili F1, bez daljeg prilagođavanja. Može se zaključiti da je razumnije koristiti određenu količinu podataka govornika kao osnovu za model sa više govornika koji se zasniva na tehnici ugradnje nego za obuku jednog SD TTS modela. Na kraju, treba napomenuti da je prilagođavanje predloženog modela korišćenjem 0,5 minuta podataka dalo sintetički govor koji je, kako se i očekivalo, ocenjen kao lošijeg kvaliteta nego u slučaju da je prilagođavanje izvršeno na 10 minuta podataka.

Zaključak

U ovom istraživanju bavimo se problemom stvaranja visokokvalitetnih sintetičkih glasova, kada je dostupna samo mala količina podataka. Ova tema je u fokusu mnogih studija već decenijama, jer ima brojne namene i potencijalno značajno smanjuje potreban rad za stvaranje novih glasova, čineći ga mnogo bržim i jeftinijim. Nakon uvoda i poređenja uobičajenih pristupa sintezi govora, ilustrovali smo i niz prethodnih pokušaja rešavanja ovog problema. Neki od njih su se zasnivali na starijim pristupima (npr. HMM-TTS), dok su neki noviji pokušali da ponude rešenje korišćenjem DNN arhitekture.

U ovoj tezi predložena su dva različita pristupa adaptaciji. Oba su modeli sinteze govora zasnovani na DNN, sposobni za prilagođavanje određenom govorniku i stilu govora. Prva

metoda inicijalno obučava TTS na relativno velikoj količini materijala za obuku (3+ sata) i koristi taj model kao polaznu tačku za adaptaciju. To znači da novi model nije obučen na slučajno inicijalizovanoj mreži, već na prethodno obučenoj mreži, što je rezultiralo većim kvalitetom sintetizovanog govora. Nije primećena značajna razlika kada su korišćeni različiti početni modeli. Takođe su korišćene različite veličine adaptacionog materijala i upoređen je kvalitet dobijenog govora. Kao što se očekivalo, više materijala dalo je bolje rezultate, ali pokazalo se da prilagođavanje čak i sa relativno malom količinom podataka može pružiti uporedive rezultate modelima istreniranim od nule sa mnogo više govornog materijala. Evaluacija se zasnivala na objektivnim merama, ali i na testovima slušanja.

Druga metoda je postupak adaptacije u dva koraka u kojem prvo pronalazimo optimalno ugrađivanje za ciljni glas na iterativni način. Pre toga se napravi model treniran na mnogo govornika i tokom tog procesa se gradi prostor za ugradnju. Drugi korak sastoji se od prilagođavanja ostatka neuronske mreže, optimizacijom svih težina i pomeraja (engl. *bias*), tako da rezultujuća mreža može proizvesti govor ciljnog govornika. Budući da je izlaz nakon prve faze već blizu cilja, količina promena primenjenih na mreži je relativno mala. Ovo sprečava preobučavanje mreže i omogućava mnogo bolju generalizaciju neviđenih događaja.

Druga metoda je pokazala da nadmašuje dva druga nedavno predložena parametarska modela sinteze govora zavisna od govornika i stila, posebno u slučaju kad je količina dostupnih podataka za adaptaciju izuzetno mala. To se postiglo zahvaljujući zajedničkoj reprezentaciji govornika, stila i klastera njihovim ugrađivanjem u prostor niže dimenzije, pri čemu je model u stanju da utvrdi sličnosti među govornicima i stilovima.

Pristup sa ugrađivanjem otvara niz zanimljivih mogućih primena predloženog modela u bilo kom domenu gde je potrebno brzo i efikasno prilagođavanje sinteze govora novom govorniku ili stilu.

Contents

Acronyms

AN-FTS – AlfaNum and Faculty of Technical Sciences

ANN – Artificial Neural Networks

BLSTM – Bidirectional LSTM

CNN - Convolutional Neural Network

CNTK – The Microsoft Cognitive Toolkit

CSTR – Centre for Speech Technology Research, University of Edinburgh

DAVO – Dynamic Analog of the VOcal tract

DCT – Discrete Cosine Transform

DNN – Deep Neural Networks

EPB – Error Back Propagation

ESNOLA – Epoch Synchronous Non Overlap and Add

FB – Filter Banks

FD-PSOLA – Frequency Domain Pitch Synchronous Overlap and Add

FF – Feed Forward networks

FFT – Fast Fourier Transform

GD – Gradient Descent

GMM – Gaussian Mixture Models

GRU – Gated Recurrent Unit

HMM – Hidden Markov Models

HMM-GMM – Hidden Markov Models and Gaussian Mixture Models

HNM – Harmonic plus Noise Model

HSMM – Hidden Semi-Markov Model

HTS – HMM-based Speech Synthesis System

tanh – Hyperbolic Tangent function

LHUC (approach) – Learning Hidden Layer Contribution

LPC – Linear Predictive Coding

LP-PSOLA – Linear prediction pitch synchronous overlap and Add

LSF – Line Spectral Frequency

LSP – Line Spectral Pairs

LSTM (neuron) – Long Short-Term Memory

MAP – Maximum A Posteriori Parameter Estimation

MBE – Mixed (Multi-Band) Excitation Vocoder

MCD – Mean Cepstral Distortion

MDL – Minimum Description Length criterion

ME – Mixed Excitation

MFCC – Mel Frequency Cepstral Coefficients

MGC – Mel-Generalized Cepstral Ccoefficients

MIT – Massachusetts Institute of Technology

ML – Machine Learning

MLLR – Maximum Likelihood Linear Regression

MLPG – Maximum Likelihood Parameter Generation

MOS (test) – Mean Opinion Score

MS – Multi-Speaker

MSD – Multi-Space probability Distribution

MSE – Mean Squared Error

MSMS (model) – Multi-Speaker Multi-Style

MUSHRA (test) – Multiple Stimuli Hidden Reference and Anchor

NN – Neural Network

OVE – Orator Verbis Electris

PAT – Parametric Artificial Talker

PLATINUM – PLATform INference by removing Underlying Material

PSOLA – Pitch Synchronous Overlap and Add

RC – Regression Class of equivalence

ReLU – Rectified Linear Unit

RMSE – Root Mean Square Error

RNN – Recurrent Neural Networks

SD TTS – Speaker-Dependent Text-To-Speech

SGD – Stochastic Gradient Descent

SMAP – Structural Maximum A Posteriori Parameter Estimation

SPSS – Statistical Parametric Speech Synthesis

SSC – Speaker, Speaking style and Cluster

STRAIGHT – Speech Transformation and Representation using Adaptive Interpolation of weiGHTed spectrum

TBE – Two-Band Excitation

TD-PSOLA – Time Domain Pitch Synchronized Overlap and Add

ToBI – Tones and Break Indices

TTS – Text-to-Speech Synthesis

VOCODER – Voice Coder

VODER – Voice Operating Demonstrator

VUV (feature) – Voiced/Unvoiced

1. Introduction

Communication is the process by which individuals conduct interactions between each other and use symbols to create and interpret meaning [1]. Speech is one of the main media for communication. First attempts to artificially generate speech date back to XVIII century [2]. Namely, in 1779. Christian Gotlieb Kratzenstein won the first prize at the competition announced by the Royal academy of arts and sciences in Saint Petersburg, for work in which he described the differences between vowels from a physiological perspective. He also presented a mechanical device which could reproduce these sounds. Since the presentation of Kratzenstein's device generation of human speech in artificial ways has travelled a long way. Contemporary approaches to this problem are based on the use of computers. Having that in mind, Text-to-Speech Synthesis (TTS) is defined as technology which provides computers with means to convert text into a signal closely resembling human speech.

Speech synthesis is a technology with a wide spectrum of applications. Initially, it was used for reading textual content for the blind, but today this technology can be of significant importance for people with various speech impairments. TTS is used in call centers to convey various information to users or even completely replace a human agent in certain scenarios. With the rise of smartphones this technology has found its place in various virtual assistant applications as well as navigation systems. In recent times there is a growing popularity of audio books, and TTS enables significantly faster and simpler generation of such audio material by using computers instead of long and tedious recording performed by professional speakers.

Speech synthesis is currently dominated by two main approaches: unit selection and parametric synthesis. Synthesizers that use unit selection (concatenative) approach select speech segments from a large speech database and concatenate them in order to generate a final sequence. Parametric approaches to speech synthesis are based on parametrization of speech signal whose textual form is known (analysis phase) and development of the model which can successfully generate parameters for a given text (synthesis phase). Until recently,

these methods were inferior to unit selection, but they have found their application in many use cases because of their flexibility and the possibility to manipulate the character of generated speech. With recent developments in neural vocoding these methods not only caught up with unit selection, but even surpassed it [3].

1.1. Subject and Main Contributions

Two main requirements that synthesized speech should meet are intelligibility and naturalness [4]. Research community is in agreement that modern speech synthesis systems achieve good results by these criteria, but it is often emphasized that the synthesized voice sounds too monotonous, and that, regardless of the use case, speech is usually generated only in one available style, frequently designated as neutral.

Another problem is related to the efficient creation of new speakers and styles. Namely, with current approaches, including parametric, it is usually required to have several hours of new speaker's recordings in order to produce high quality synthesis. The process of building a new voice or style also requires recording of a new speaker, but also some form of semi-automatic annotation and preparation of the database. This obviously makes the process longer, more expensive and less scalable in commercial applications.

The main goal of this research is to examine the possibility to build new voices (speakers with corresponding styles) while significantly reducing the amount of required speech material and consequently the work required to prepare the database. The focus will be on the use of Deep Neural Networks (DNN) as the currently most advanced parametric approach. All the approaches will be tested on the speech database with a relatively small amount of adaptation material (speech from the new speaker). The hypothesis to be tested is that by applying novel methods on a limited amount of new speech material, high quality results can be obtained, comparable to those obtained by using large speech databases.

All experiments are done for American English language, since those audio databases were readily available. But since this research focuses on the language independent part of

TTS, which mostly deals with signal generation (so called back-end), it can be assumed that the results will be applicable to any language.

Main contributions of the research presented in the dissertation are:

- Adapting open source Merlin toolkit [5] to work with more contemporary deep learning frameworks, such as TensorFlow [6] and CNTK [7]. Improving the toolkit and implementing new version of forced alignment (Sections: 4.4 and 4.4.1).
- Building a new voice by starting from previously trained single-speaker TTS model. Source model is trained on large amount of data, while target speaker has only limited amount (Section 5).
- Creating a multi-speaker (MS) TTS model, able to produce speech of many speakers, while producing a "speaker space" for all the speakers, i.e. an embedding space, which reflects similarities between speakers (Section 6.1).
- Building a new voice by starting from the MS model and using a very small amount of new speech material (Section 6.2).

1.2. Thesis organization

The thesis contains six chapters, and is organized as follows.

Section 2 provides the introduction to speech synthesis, its history, and describes the main approaches. It also explains the basic concepts and problems related to speaker adaptation and describes methods used for TTS system evaluation.

Section 3 focuses on statistical parametric approach and illustrates methods for speaker adaptation which were used with that approach.

Section 4 deals with neural networks (NN) and their application in speech synthesis. It provides a brief history of NNs and an introduction to basic concepts of deep learning. It proceeds to explain how NNs are used in contemporary speech synthesis systems and compares the statistical and the NN approach. Standard approaches for speaker adaptation are also presented. This chapter also gives a brief introduction to open-source Merlin toolkit, which was used as a starting point for this research.

Section 5 describes the proposed method of speaker adaptation from source to target speaker model, where there is sufficient amount of audio data for source speaker, but only a small amount for target speaker. Instead of training target speaker model from randomly initialized neural network, proposed method starts from already built source speaker model, which had sufficient data for conventional training.

Section 6 explains in detail the proposed method for speaker adaptation, which starts by building multi-speaker model and low-dimensional speaker space (embedding), followed by two-step adaptation procedure. In the first step optimal point in the embedding space for target speaker is found. During the second step adaptation of the rest of the neural network is performed, which results in a voice highly resembling target speaker.

The advantages of the novel methods are corroborated by numerous experiments and evaluations.

Section 7 provides conclusions and outlines the directions for future work in this area.

2. Introduction to Speech Synthesis

The term speech synthesis refers to computer-generated simulation of human speech. Speech synthesis is used to translate written information into audio information, and is used whenever audio information is more convenient, especially in mobile applications such as voice-enabled e-mail and unified messaging. It is also used to assist the visually impaired so that, for example, the contents of a display screen can be automatically read aloud to a blind user. Speech synthesis is the counterpart of speech recognition.

Speaker adaptation is a text-to-speech technique by which a TTS system can be customized to the voice characteristics and manner of speaking of a specific speaker, typically in a short time and at a low cost. In conventional TTS systems, at least several hours of speech data are required to create a voice that represents a specific speaker's voice characteristics, in order for the system to be able to convert any text to speech. Speaker adaptation techniques convert an existing voice model with sufficient phonetic and linguistic coverage into a model having the voice characteristics and manner of speaking of a specific speaker, based on a small quantity of speech data of that speaker. These techniques make it possible to create a voice for a specific speaker with a high level of voice quality from a very small amount of speech data.

2.1. History of Speech Synthesis

Artificial speech has been a dream of the humankind for centuries. The earliest efforts to produce synthetic speech were made over two hundred years ago. In 1779 in St. Petersburg Russian Professor Christian Kratzenstein explained physiological differences between five long vowels (/a/, /e/, /i/, /o/, and /u/) and designed an apparatus to produce them artificially. He constructed acoustic resonators similar to the human vocal tract and activated them with vibrating reeds as it is done in music instruments. Several years later, in 1791 in Vienna, Wolfgang von Kempelen introduced his "Acoustic-Mechanical Speech Machine", which was

able to produce single sounds and some sound combinations. In mid 18th century Charles Wheatstone constructed his famous version of von Kempelen's speaking machine. It was somewhat more complicated and was capable to produce vowels and most of the consonant sounds. Some sound combinations and even full words were also possible to produce. In late 18th century Alexander Graham Bell with his father, inspired by Wheatstone's speaking machine, constructed a similar speaking machine. The research and experiments with mechanical and semi-electrical analogs of the human vocal system were made until 1960's, but with no remarkable success.

The first full electrical synthesis device was introduced by Stewart in 1922 [8]. The synthesizer had a buzzer as excitation and two resonant circuits to model the acoustic resonances of the vocal tract. The machine was able to generate single static vowel sounds with two lowest formants, but not consonants or connected utterances. In 1932 Japanese researchers Obata and Teshima discovered the existence of the third formant in vowels [9]. The three first formants are generally considered to be sufficient for intelligible synthetic speech. The first device to be considered as a speech synthesizer was VODER (Voice Operating Demonstrator) introduced by Homer Dudley at New York World's Fair in 1939 [8] [10]. VODER was inspired by VOCODER (Voice Coder), developed at Bell Laboratories in the mid-thirties. The original VOCODER was a device for analyzing speech into slowly varying acoustic parameters that could then drive a synthesizer to reconstruct the approximation of the original speech signal. It was finally shown that intelligible speech can be produced artificially. Actually, the basic structure and idea of VODER is very similar to present systems which are based on the source-filter model of speech.

The first formant synthesizer, PAT (Parametric Artificial Talker), was introduced by Walter Lawrence in 1953 [8]. PAT consisted of three electronic formant resonators connected in parallel. The input signal was either a buzz or noise. At about the same time Gunnar Fant introduced the first cascade formant synthesizer OVE I (Orator Verbis Electris), which consisted of formant resonators connected in cascade. PAT and OVE synthesizers started a conversation how the transfer function of the acoustic tube should be modeled, in parallel or in cascade. First articulatory synthesizer was introduced in 1958 by George Rosen at the Massachusetts Institute of Technology (MIT) [8]. The DAVO (Dynamic Analog of the

VOcal tract) was controlled by tape recording of control signals created by hand. In mid 1960's, first experiments with Linear Predictive Coding (LPC) were carried out [2].

The first full text-to-speech system for English was developed in the Electrotehnical Laboratory, Japan 1968 by Noriko Umeda and his collaborates [8]. It was based on an articulatory model and included a syntactic analysis module with sophisticated heuristics. In 1979 Allen, Hunnicutt, and Klatt demonstrated the MITalk laboratory text-to-speech system developed at MIT. Two years later Dennis Klatt introduced his famous Klattalk system, which used a new sophisticated voicing source. The first reading aid with optical scanner was introduced by Kurzweil in 1976. The Kurzweil Reading Machines for the Blind were capable to read multifont written text quite well. In late 1970's and early 1980's, a wide range of commercial text-to-speech and speech synthesis products was introduced. Dominant systems in the 1980's and 1990's were the DECtalk system, based largely on the work of Dennis Klatt at MIT, and the Bell Labs system; the latter was one of the first multilingual language-independent systems, making extensive use of natural language processing methods.

From late 1990's until 2016 and the introduction of WaveNet [3], the market was dominated by unit selection versions of synthesizers, produced by several major companies (Nuance, Acapela, Google). Even though parametric approach based on hidden Markov models (HMM) was widely known, its quality was inferior to unit selection based solutions. After 2016, WaveNet based solutions started to catch up and even surpass the quality of unit selection methods. Beside speech quality, the big advantage of WaveNet is that it is parametric and as such more flexible for many applications.

2.2. Approaches to Speech Synthesis

As shown in Figure 1, a typical speech synthesis system consists of two components: front-end and back-end. The front-end component performs analysis of text input and extraction of information necessary for back-end modelling. This includes text normalization (e.g. converting numbers into words), parts of speech (e.g., noun, verb, adjective) annotation, prosodic features (Tones and Break Indices (ToBI) or like, see Section 4.5), and disambiguation of homographs. The back-end component accepts the front-end analysis

results and combines the speech and text information for modelling. During the process of synthesis, the back end generates the output speech signals using the text input and well-trained acoustic models.

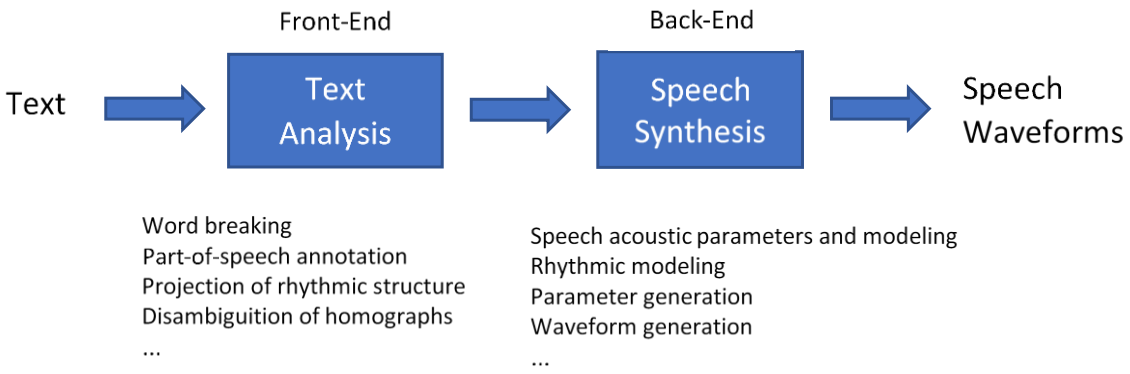


Figure 2: *Typical speech synthesis architecture*

The rest of this work will focus on the back-end component and different approaches applied therein.

2.2.1. **Unit Selection Synthesis**

In this approach multiple instances of speech units having different prosodic features are stored. This approach is known as Unit Selection Based Concatenative TTS. Units are stored in database and then assembled in accordance to defined rules and costs (Figure 2). An appropriate unit is selected from the database based on two types of costs – a target cost and a concatenation cost.

Target cost expresses how similar the features of a database speech unit are to the features of the desired speech unit, with the idea that any digital processing that may be used to bring the selected acoustic segment closer to the specification may introduce unwanted distortion. The target cost comprises of target subcosts. Each target subcost is a cost of a single attribute of a speech unit [11] such as energy, pitch etc. The target cost can be calculated as:

$$C_t(t_i, v_i) = \sum_{j=1}^{p} w_{tj} C_{tj}(t_i, v_i) \quad (1)$$

where t_i is the target unit, v_i is the candidate unit, p is the number of sub-costs used. C_{tj} is the j-th target sub-cost, and w_{tj} it is the weight given to the j-th target sub-cost.

Concatenation cost is a measure of how well the acoustic features of two acoustic segments match at the point where they should be concatenated. The concatenation cost also comprises of multiple subcosts. Each of these subcosts is related to a specific continuity metric such as spectral continuity etc. The concatenation cost can be calculated as:

$$C_c(v_{i-1}, v_i) = \sum_{j=1}^{q} w_{cj} C_{cj}(v_{i-1}, v_i) \tag{2}$$

where v_{i-1} and v_i are candidate speech units for the $(i-1)$-th and i-th target speech units, q is the total number of subcosts used, and w_{cj} is the weight associated with the subcost C_{cj}.

An exhaustive search is performed so as to select optimum speech units from the speech database. The Viterbi search is frequently used to select the units to be concatenated from the speech inventory, so as to reduce the total accumulated target cost and concatenation cost.

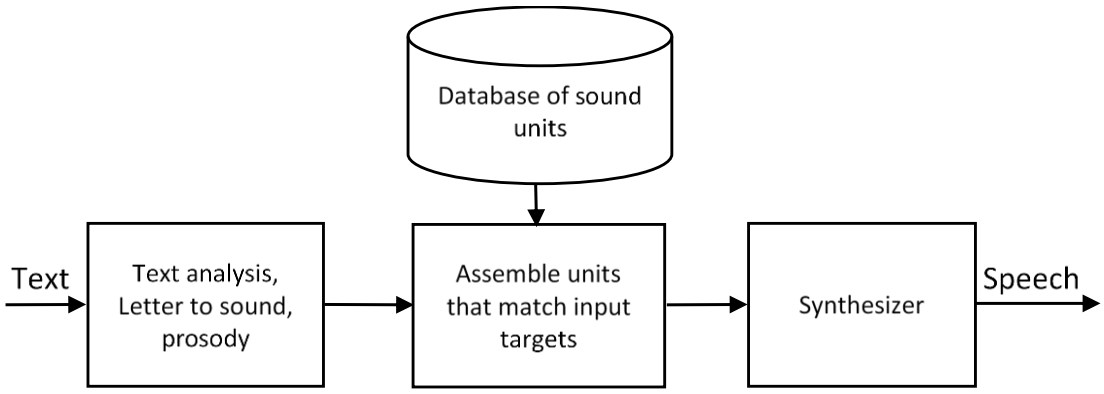


Figure 3: *Unit selection TTS system architecture*

In order to match the target more precisely (f_0, energy, phone duration) and to make transitions between units as smooth as possible, several methods were used:

- Pitch Synchronous Overlap and Add (PSOLA)
- Frequency Domain Pitch-Synchronous Overlap and Add (FD-PSOLA)
- Linear Prediction Pitch-Synchronous Overlap and Add (LP-PSOLA)

- Time Domain Pitch-Synchronized Overlap and Add (TD-PSOLA)
- Epoch Synchronous Non Overlap and Add (ESNOLA)

2.2.2. **Parametric Speech Synthesis**

When it comes to a model-based approach to speech synthesis, particularly when the goal is to learn this model from data, we are talking about a parametric speech model. The model is parametric because it represents the speech using parameters, rather than stored sound samples.

A widely used model for speech parametrization is the *source-filter model*. The source–filter model models speech as a combination of a sound source, such as periodic buzz from vocal cords or noise from lungs; and a linear acoustic filter, which describes the vocal tract and lip radiation characteristic. It is illustrated in the Figure 3.

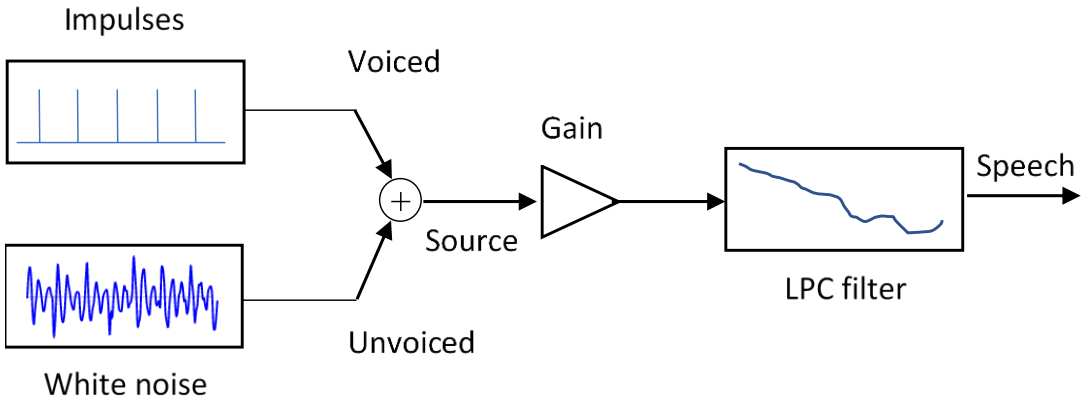


Figure 4: *Source-filter model of voice production*

Source is usually a combination of impulse train and white noise. The ratio of their respective contributions to the source signal, which is usually defined per spectral band, is called band periodicity/aperiodicity. The frequency of the impulse train defines pitch or fundamental frequency (f_0 in Figure 4) of the final phone. The acoustic filter describes the spectral envelope (Figure 4), which provides the final shape of the spectrum of the output signal (while source is kept white). Different phonemes can be distinguished by the properties of their source(s) and their spectral envelopes. For example, vowels and sonants

have a source due to mostly periodic glottal excitation, which can be approximated by an impulse train in the time domain and by harmonics in the frequency domain, while the corresponding filter depends on vocal tract shape and lip position. On the other hand, fricatives have a source mostly due to turbulent noise produced at a constriction in the oral cavity (e.g. “s” and “f”) or noise made by glottis and lungs (e.g. “h”). But most sounds actually have two sources – one at the glottis and one at the supra-glottal constriction, and the ratio of voiced to unvoiced components (per spectral band) is described by band aperiodicity feature.

The task of the acoustic filter is to represent the spectral envelope and to appropriately shape the source. Spectral envelope represents a smoothed version of a spectrum, which should leave aside the spectral line structure while preserving the general form of the spectrum. If the signal contains only harmonic parts, spectral envelope is the curve that passes through the local peaks. In this case, peak values have to be retrieved and an interpolation scheme should exist in order to complete the curve between the peaks. If the sound contains parts that are not harmonic (i. e. that are noisy), the notion of a spectral envelope becomes completely dependent on the definition of what belongs to the source and what belongs to the filter.

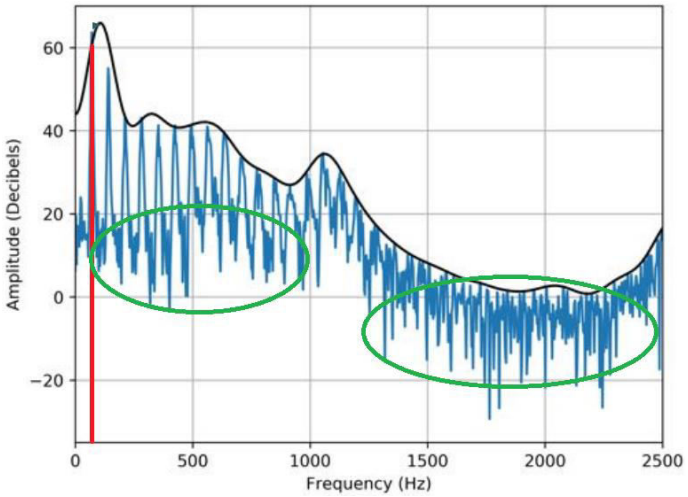


Figure 5: *Speech signal spectrum*
black line - spectral envelope; red line - f_0; green areas - aperiodic energy

As already noted, signal contains both harmonic and aperiodic (noise) component, and the ratio of their presence in the signal is not constant over the whole spectrum. For that reason, band aperiodicity parameter is introduced. It represents the ratio between aperiodic and periodic energy, averaged over certain frequency bands, i.e., total power divided by sine wave power.

There are numerous methods for extracting the spectral envelope, with the following being most widely used:

- Channel vocoder or filter banks (FB). This approach is based on frequency bands and performs estimations of the amplitude of the signal inside these bands, thus approximating the spectral envelope.
- Linear prediction coding. This method estimates an all-pole filter that matches the spectral content of a sound. When the order of this filter is low, only the formants are taken, while a higher order would describe a more detailed representation. The LPC predictor coefficients can be used to efficiently model the vocal tract spectral envelope, but they are not robust in terms of quantization or statistical modeling: even though the autocorrelation method guarantees a stable filter, a small error in the coefficient values may cause the synthesis filter to become unstable.
- Line Spectral Pairs (LSP). Many methods have been proposed for robust representation of LPC coefficients, such as reflection coefficients or log area ratios. One of the most prominent methods of presenting LPC data is the Line Spectral Frequency (LSF) representation, with line spectral frequencies being the roots of the LSP polynomials.
- (Mel) Cepstral coefficients (MFCC). Along with LPC, cepstral analysis of speech is one of the most widely used methods for the extraction of the spectral envelope. This technique performs the approximation of the logarithm of the fast Fourier transform (FFT) spectrum by using discrete cosine transform (DCT). Again, the more DCT coefficients are used the finer representation of the spectral envelope is preserved.
- Mel-generalized cepstral coefficients (MGC). The generalized cepstral analysis method is viewed as a unified approach to the cepstral method and the linear prediction method, in which the model spectrum varies continuously from all-pole to cepstral according to the value of a scalar parameter γ. Since the human ear has higher resolution at low frequencies, using Mel scale to model spectrum, it is represented more efficiently.

Besides these, vocoder parameters, TTS systems have to parameterize and model phone durations as well (or even subphone – state durations). These are not considered for the task of mere speech coding and transmission, because the transmission is synchronous so durations don't change. But when it comes to converting text into speech, the system has to find a way to estimate these durations correctly.

To summarize, when speaking about parameters in parametric speech synthesis, these are usually considered:

- Phone and state durations, usually given in frames or milliseconds.
- Some form of spectral envelope representation.
- Pitch (f_0) representation, which includes the information whether a sound segment is voiced or unvoiced. Logarithm of f_0 is used more frequently, because we are more interested in relative rather than absolute distance between two f_0 values.
- Band aperiodicity.

A typical representation may use between 40 and 60 parameters per frame (usually 5 ms) to represent the spectral envelope, band aperiodicity, the value for f_0, and the degree of voicing (usually binary). Before training the models, the encoding stage of the vocoder is used to extract a vector containing these vocoder parameters from the speech signal, at a certain constant frame rate. During synthesis, the vector of parameters is generated by the models, and fed to a vocoder which generates the output signal. Here is the list of some of the most popular vocoders used in TTS:

- Impulse Excitation vocoder. The most basic vocoder used in statistical parametric speech synthesis essentially exemplifies the unified source-filter model: the speech signal is divided into source and filter parameters, and the source signal is modeled as a pulse train for voiced segments, and as white Gaussian noise for unvoiced segments.
- Mixed (Multi-Band) Excitation vocoder (MBE). The main idea of the Mixed Excitation vocoder is based on the observations of the spectral characteristics of the residual: the residual has been found to have different degrees of periodicity and noise in different frequency bands. If the residual is modeled completely periodically (with a pulse train), the resulting voice will sound "buzzy". Similarly, if the residual is modeled completely with noise, the resulting voice will sound "hissy". With a correct combination of periodic and noise components in the excitation (residual), the synthesized speech will show a

great increase in quality. The most popular MBE vocoders are: Mixed Excitation (ME) vocoder, Two-Band Excitation (TBE) vocoder based on Harmonic plus Noise Model (HNM), STRAIGHT vocoder as well as WORLD vocoder. We are providing additional information for some of these:

- STRAIGHT (Speech Transformation and Representation using Adaptive Interpolation of weiGHTed spectrum) [12] belongs to MBE group, but deserves to be singled out, since it is the most established of the more sophisticated vocoding methods. This speech analysis, modification and synthesis system is an extension of the classical channel VOCODER that utilizes the progress in information processing technologies and a new understanding of the role of repetitive structures in speech sounds. It uses pitch-adaptive spectral analysis combined with a surface reconstruction method in the time-frequency region, and an excitation source design based on phase manipulation.
- WORLD vocoder [13] also belongs to MBE group. It provides high-quality speech output, similar to STRAIGHT, but also meets the requirements of real-time processing. The f_0 contour is estimated with the procedure named DIO [14] and the spectral envelope is estimated with CheapTrick [15], which uses not only the waveform but also the f_0 information. The excitation signal is estimated using PLATINUM (PLATform INference by removing Underlying Material [16]) and used as an aperiodic parameter (whose definition is different from that of STRAIGHT). PLATINUM uses the waveform, f_0, and spectral envelope information.

- WaveNet [3] is a neural network based vocoder. During training, it is presented with the spectrogram or some other set of parameters (extracted by a conventional vocoder) and the actual signal is set as a target. Thus, WaveNet learns to generate naturally sounding speech when presented with its parameters during inference.

There are two main approaches which utilize speech parametrization for TTS: statistical parametric speech synthesis based on hidden Markov models and Gaussian mixture models (HMM-GMM) and speech synthesis by using deep neural networks (DNNs). Both will be discussed in more detail in the remainder of this work.

2.3. Introduction to Speaker Adaptation

In speech synthesis, speaker adaptation refers to the range of techniques whereby a TTS system is adapted to the acoustic features of a specific speaker using a small sample of utterances. In recent years the practical development of speech synthesis systems has seen significant progress; however, these systems still require large annotated databases (ten or more hours of speech) in order to be trained well. Much hope has therefore been placed on the establishment of speaker adaptation techniques that can bring the performance of a single speaker system trained from scratch, up to that of a speaker-adapted one using the smallest possible amount of data.

One of the main problems in training TTS models is data sparsity. Namely, the final system should be able to produce any phoneme in any context (phonetic or prosodic, in a relatively wide time window), which implies that each such acoustic phenomenon should be seen in the training database. Since the number of these phenomena is very large (millions or more), it is virtually impossible to have them all even in a very large training database. For this reason, models should be able to generalize well and to produce reasonable outputs even for unseen contexts. This problem becomes much harder if the available amount of adaptation data is very small (several minutes or less). In that case the percentage of seen contexts becomes almost negligible, meaning that everything else should be generalized from that small sample and some prior information.

In the HMM-GMM approach, there were several techniques which attempt to address this problem, of which the most successful are: maximum a posteriori (MAP) parameter estimation and maximum likelihood linear regression (MLLR), which will be described later. When it comes to DNN-based TTS, research is still under way, but some of the promising methods include transfer learning and speaker embedding.

2.4. TTS System Evaluation

There are several different approaches used for the evaluation of a specific TTS system. All of them can be divided into following groups [17]:

- evaluation based on objective measures,

- evaluation based on subjective measures,
- comparison of system characteristics.

Objective measures are obtained by direct comparison of parameters extracted from the natural speech with the generated parameters. This type of measures can be particularly useful in the development phase of TTS system. Namely, a large discrepancy between generated parameters and parameters extracted from natural speech is usually a sign of serious problem in the system design or implementation. Early problem detection is important since it can accelerate the development of a new system. However, the problem with this approach is the fact that the values of objective measures are not always correlated with the subjective evaluation by human listeners. It is possible to have a system (or e.g. a set of generated utterances) that differ significantly from natural speech in terms of objective measures, but still sound quite natural and intelligible to humans. The reason for this is the natural variability of speech. Namely, there are numerous ways in which a certain text could be spoken, even when certain constraints are given (e.g. style and prosodic guidelines). A TTS system provides only one rendition, while a human speaker could generate a significantly different one. Even if TTS system provides a perfectly natural output, there still can be significant differences with respect to the rendition produced by the speaker. In this dissertation the following objective measures were used:

1. Mean Cepstral Distortion (MCD), defined by [18]

$$MCD = \frac{1}{T}\frac{10\sqrt{2}}{\ln 10}\sum_{t=0}^{T-1}\sqrt{\sum_{d=0}^{D-1}\left(v_d^{gen}(t) - v_d^{ref}(t)\right)^2}, \tag{3}$$

where T is the total number of frames in referent sequence v^{ref}, and generated sequence v^{gen}, and D is the number of cepstral coefficients extracted per each frame.

2. Distortion of aperiodicity coefficients (if used), calculated in the same way as in (3).
3. Root Mean Square Error (RMSE) of the fundamental frequency (f_0).
4. Voiced/Unvoiced (VUV) error, calculated as the ratio of the number of wrongly predicted frames and total number of frames.
5. Correlation for the features of voicing and fundamental frequency.

Subjective measures are based on the scores given by human subjects regarding certain characteristics of the speech. This group of tests can be divided into two subgroups:

3. Intelligibility score,
4. Synthesis quality score.

It is widely accepted that contemporary synthesizers have good intelligibility [19], hence the methods presented in the dissertation are scored by using synthesis quality score approach. Specifically, Mean Opinion Score (MOS) and Multiple Stimuli with Hidden Reference and Anchor (MUSHRA) tests were used [20].

During MOS testing, subjects are expected to score certain speech characteristic with the scores from 1 (poor) to 5 (excellent). The final result of this test is the average score from all subjects.

MUSHRA is a methodology initially used for conducting a codec listening test to evaluate the perceived quality of the output from lossy audio compression algorithms. In MUSHRA, the listener is presented with the reference (labeled as such), a certain number of test samples, a hidden version of the reference and one or more anchors. The recommendation specifies that a low-range and a mid-range anchor should be included in the test signals. The purpose of the anchors is to calibrate the scale so that minor artifacts are not unduly penalized. This is particularly important when comparing or pooling results from different labs. The listeners' task is to firstly determine the utterance of the highest quality, compared to the reference, and then to score the rest of the utterances by assigning scores from 1 to 100.

Subjective tests are the best way to score the quality of the synthesized speech. The main disadvantages are time and human resources required for conducting a proper test, as well as substantial variability in subjects' perception of speech quality.

3. Statistical Parametric Speech Synthesis

Until recently, Statistical Parametric Speech Synthesis (SPSS) was the only widely used approach which utilized parameters to represent speech. It is called statistical because it defines those parameters using statistics (usually means and variances of probability density functions) which describe the distribution of parameter values found in the training data. The initial motivation for the use of statistical parametric speech synthesis was the success of the hidden Markov models for automatic speech recognition. The existence of efficient training algorithms (Expectation-Maximization), automatic methods for model complexity reduction (tying of parameters) and computationally efficient search algorithms (Viterbi) make the HMM an obvious choice. The performance of the model depends critically on choosing an appropriate configuration. Two principal aspects of this configuration are the parameterization of the speech signal (observations) and the choice of modelling unit. The modelling unit is usually a context-dependent phoneme, so this choice means selecting which contextual factors need to be taken into account. As for the speech parametrization, the speech signal is represented as a set of vocoder parameters at some fixed frame rate.

The model most commonly used in statistical parametric speech synthesis is not the conventional HMM. The duration model (i.e., the state self-transitions) in the HMM is not optimal, so a better model for phoneme duration prediction is required for high-quality speech synthesis. When explicit duration modelling is added to the HMM, it is no longer a Markov model in the mathematical sense. Transitions between states still exist, and the model is Markov at that level, but the exact model of state durations is not Markov. Such a model is referred to as Hidden Semi-Markov Model (HSMM).

The vocoder parameters themselves are the only thing needed to control the output stage of the vocoder and generate speech. However, the key to producing natural-sounding speech using HMM synthesis lies in the modelling of not only the statistical distribution of these parameters, but also in the dynamics of their change over time. The vocoder parameters are known as the static coefficients, their first-order derivatives as the delta coefficients and second-order derivatives as the delta-delta coefficients. These three subsets of parameters are

combined together into a single observation vector for the model. During training, the model learns the distributions of all these parameters. During synthesis, the model generates parameter trajectories which have optimal statistical properties.

During synthesis, the input text is analyzed and a sequence of full context labels is produced. The sequence of models corresponding to this sequence of labels is then joined into a single vector of (HMM-GMM) states. From this model, the vocoder parameters are generated using the maximum likelihood parameter generation (MLPG) algorithm [21] illustrated in Figure 5. Parameters are generated from the model based on the criterion of maximum likelihood, used to generate the optimal sequence of observations (vocoder parameters). State durations (the number of frames of parameters to be generated by each state of the model) are determined in advance – they are simply the means of the state duration distributions. Finally, the generated vocoder parameters are used to produce a speech waveform.

A naive method for parameter generation would generate the most likely observation from each state, taking into account only the static parameters. The most likely observation is the mean of the Gaussian in that state. Therefore, this method would generate piecewise constant parameter trajectories, which would change their values abruptly at each state transition. Obviously, this would not sound natural – natural speech usually does not have such parameter trajectories. This problem is solved by the MLPG algorithm, which takes the statistical properties of the delta and delta-delta coefficients into account. Before generating parameters, a state sequence is selected using the duration model, and the number of frames for each state is also determined. The figure illustrates the sequence of output distributions for each state, frame by frame. MLPG finds the most likely sequence of generated parameters (0th cepstral coefficient is used as an example in Figure 5), given the distributions for the static, delta and delta-delta parameters. It can be seen that the most likely parameter trajectory is smoothly changing in a statistically appropriate way.

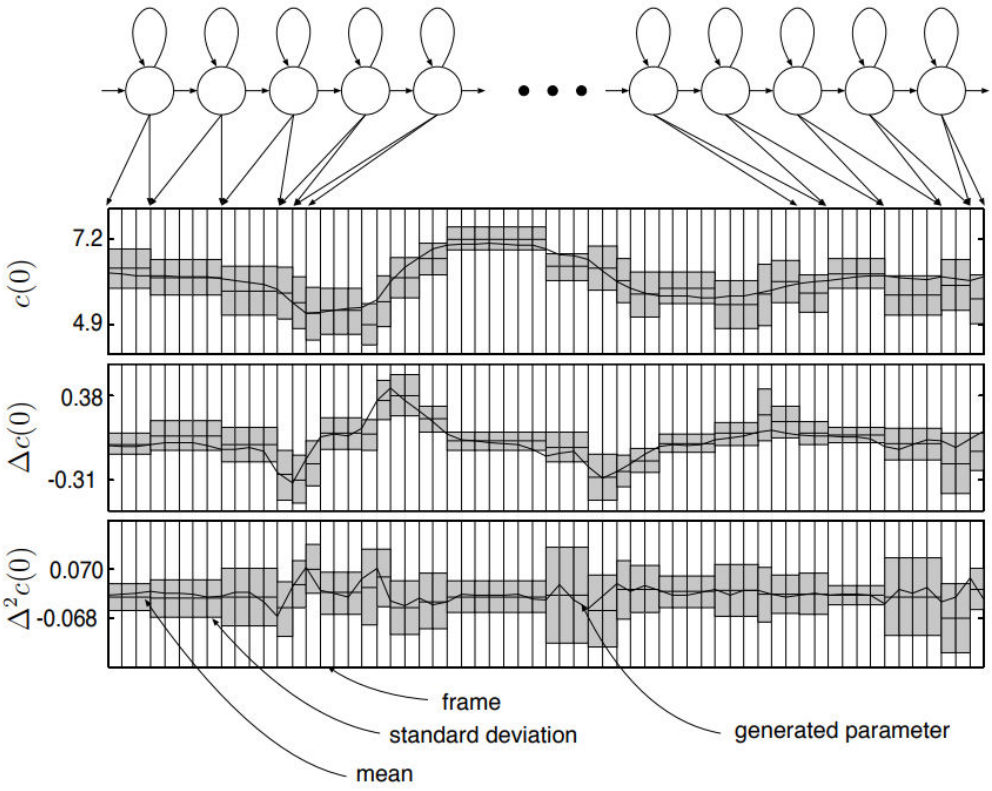


Figure 6: *Illustration of Maximum Likelihood Parameter Generation algorithm*

3.1.1. **Speaker Adaptation in HMM-GMM**

As previously described, HMM-GMM model is represented by a set of states, each containing usually only one Gaussian component which represents the position of that state in the entire parameter "acoustic space". There are usually separate models for duration, f0 and spectral envelope parameters. When performing speaker adaptation, all these model parameters should be transformed so that they better represent target speaker observations (adaptation data). In other words, we are transforming acoustic space of the initial speaker (which may also be an average of multiple speakers) towards target speaker's data. After that, synthesis by using the new model should sound more like the target speaker. We will present two most popular adaptation techniques in HMM-GMM: MAP and MLLR.

3.1.2. **MAP Adaptation**

If parameters of the model θ are assumed to be random vector, which is to be estimated from samples x_t drown from probability density function $f(\cdot\,|\theta)$ and for θ we assume some prior p.d.f. g, and if we denote posterior probability density as $g(\theta|x)$, then we obtain MAP estimate of θ using the Bayesian theorem, as follows:

$$\theta_{\text{MAP}} = \text{argmax}_\theta g(\theta|x) = \text{argmax}_\theta f(x|\theta)g(\theta) \tag{4}$$

In (15), if prior g is non-informative, i.e., $g \equiv \text{const}$, we obtain that the ML estimate coincident with the MAP estimate. Let now $x = [x_1 \cdots x_T]$ be the vector of T independent, identically distributed (i.i.d.) observations drawn from the GMM with K Gaussian p dimensional components, i.e. the following holds:

$$\begin{gathered} f(x|\theta) = \prod_{t=1}^{T}\sum_{k=1}^{K} \omega_k N(x_t|m_k, r_k), \\ \omega_k \geq 0, \sum_{k=1}^{K} \omega_k = 1\,, \theta = [\omega_1 \cdots \omega_K, m_1 \cdots m_K, r_1 \cdots r_K] \end{gathered} \tag{5}$$

where ω_k denotes the mixture gain for k-th mixture component and

$$N(x|m_k, r_k) \propto |r_k|^{1/2}\exp[-\frac{1}{2}(x-m_k)^T r_k (x-m_k)] \tag{6}$$

where m_k is p dimensional mean vector and r_k is the precision matrix (inverse of covariance matrix) and $|\cdot|$ denotes determinant. Recall that the statistic $t = t(x)$ is the *sufficient statistics* for underlying parameter θ of the density $f(\cdot\,|\theta)$ iff there exists factorization $f(x|\theta) = h(x)k(\theta, t(x))$, where kernel density k depends on θ explicitly, but on x only trough $t(x)$. If $f(x|\theta)$ posses sufficient statistics of fixed length, then the natural solution is to chose the prior density g in the form of conjugated family $k(\cdot, \phi)$, which includes the kernel density k, so that the posterior has the form $g(\theta|x) \propto k(\theta, \phi') = k(\theta, \phi)k(\theta, t(x))$, where we assume that for kernel k it holds $k(\theta, \phi_1)k(\theta, \phi_1) = k(\theta, \phi)$ for some $\phi = \phi(\phi_1, \phi_2)$ for all ϕ_1, ϕ_2 (very often $\phi(\phi_1, \phi_2) = \phi_1 + \phi_2$). However, among the distribution families of interest, only exponential families have a sufficient statistic of fixed length, so that only for them, there exists conjugate prior, which greatly constraints possible

applications [22]. Consequently, there is no simple conjugate prior for GMMs (and thus HMMs with GMM state observation densities). Nevertheless, an approximate approach which uses EM algorithm could be employed, as it is presented in [23]. Note first, that a finite mixture density $f(x|\theta)$ given by (5) can be interpreted as a density associated with a statistical population which is a mixture of K component populations with mixing proportion $[\omega_1 \cdots \omega_K]$. It can be seen as a marginal distribution of the joint distribution of the parameters θ expressed as the product of a multinomial density (for the sizes of the component populations) and multivariate Gaussian densities (for the components). Gains ω_k for each mixture density have the joint distribution is in the form of a multinomial distribution with density given by $q(n_1, \dots, n_K; T, \omega_1, \dots, \omega_K) \propto \omega_1^{n_1} \cdots \omega_K^{n_K}$, $n_1 + \cdots + n_K = T$, $n_k \in N$. Then choice for prior is the Dirichlet density, so that $g(\omega_1, \dots, \omega_K; \nu_1, \dots, \nu_K) \propto \prod_{k=1}^{K} \omega_k^{\nu_k - 1}$ with hyper-parameters $\nu_k \geq 0$. Next, for the simplicity we consider r_k to be known, i.e., not subject to adaptation so it is considered deterministic constant. Thus, the conjugate prior for m_k is Gaussian and given by

$$g(m_k; r_k, \tau_k) \propto |r_k|^{\frac{\alpha_k - p}{2}} \exp[-\frac{\tau_k}{2}(m_k - \xi_k)^T r_k (m_k - \xi_k)] \tag{7}$$

with hyper-parameters satisfying $\alpha_k > p - 1$, $\tau_k > 0$. Assuming independence between the parameters of the individual mixture components and the set of the mixture weights, the joint prior g is the product of the prior density defined in (6) and (7), so that

$$g(\theta) = g(\omega_1, \dots, \omega_K) \prod_{k=1}^{K} g(m_k; r_k, \tau_k) \tag{8}$$

The EM algorithm is an iterative procedure for approximating ML estimates in the context of incomplete data cases such as GMMs and HMM with GMMs [24]. This procedure consists of maximizing at each iteration the auxiliary target function $Q(\theta, \hat{\theta})$ defined as the conditional expectation of the complete-data log-likelihood $\ln h(y|\theta)$, where $y = [xz]$ is complete data vector and $z = [z_1 \cdots z_T]$ are the unobserved labels referring to the mixture components (whose mixture label z_t generated particular x_t), given the incomplete data x and the current fit $\hat{\theta}$, i.e.,

$$Q(\theta,\hat{\theta}) = E_{z|x,\hat{\theta}}[\ln h(y|\theta)] \tag{9}$$

where $E_{z|x,\hat{\theta}}[] = \int [] h(z|x,\hat{\theta})\mathrm{d}z$ is the conditional expectation operator. The EM procedure is based on the fact that $h(y|\theta) = h(z|x,\theta)f(x|\theta)$ (Bayesian theorem), so that after performing logarithm, taking expectation $E_{z|x,\hat{\theta}}[]$ and using $\int h(z|x,\hat{\theta})\mathrm{d}z = 1$, one obtains $\ln f(x|\theta) = Q(\theta,\hat{\theta}) - H(\theta,\hat{\theta})$, with

$$H(\theta,\hat{\theta}) = E_{z|x,\hat{\theta}}[\ln h(y|x,\theta)] \tag{10}$$

As it holds, $H(\theta,\hat{\theta}) < H(\hat{\theta},\hat{\theta})$, by assuring $Q(\theta,\hat{\theta}) \geq Q(\hat{\theta},\hat{\theta})$, as $\ln(\cdot)$ is monotony increasing one obtains $f(x|\theta) > f(x|\hat{\theta})$, so that the iterative process leads toward maximization of likelihood. In the case of MAP estimation, the same procedure can be obtained, by maximizing $R(\theta,\hat{\theta}) = Q(\theta,\hat{\theta}) + \ln g(\theta)$. By direct calculation, one can obtain [22]:

$$Q(\theta,\hat{\theta}) = \sum_{t=1}^{T}\sum_{k=1}^{K} \frac{\hat{\omega}_k N(x_t|\hat{m}_k, r_k)}{\sum_{l=1}^{K}\hat{\omega}_l N(x_t|\hat{m}_l, r_l)} \ln \omega_k N(x_t|m_k, r_k) \tag{11}$$

We maximize $\Psi(\theta,\hat{\theta}) = \exp(R(\theta,\hat{\theta}))$ (which is equivalent to maximizing $Q(\theta,\hat{\theta})$), so we define the following notation:

$$\begin{aligned} c_{\mathrm{kt}} &= \frac{\hat{\omega}_k N(x_t|\hat{m}_k, r_k)}{\sum_{l=1}^{K}\hat{\omega}_l N(x_t|\hat{m}_l, r_l)}, \\ c_k &= \sum_{t=1}^{T} c_{\mathrm{kt}}, \\ \bar{x}_k &= \sum_{t=1}^{T} c_{\mathrm{kt}}\, x_t / c_k \end{aligned} \tag{12}$$

It follows from the definition of $f(x|\theta)$, given by (5), that the following holds

$$\Psi(\theta,\hat{\theta}) \propto g(\theta)\prod_{k=1}^{K} \omega_k^{c_k} |r_k|^{c_k/2} \exp[-\frac{c_k}{2}(m_k - \bar{x}_k)^T r_k (m_k - \bar{x}_k)] \tag{13}$$

so that from (7), (8) and the shape of $g(\omega_1, \ldots, \omega_K)$, one concludes that $\Psi(\cdot,\hat{\theta})$ belongs to the same family of distributions as $g(\cdot)$, but with the parameters

$$\begin{aligned} \nu_k' &= \nu_k + c_k, \\ \tau_k' &= \tau_k + c_k, \\ \alpha_k' &= \alpha_k + c_k, \\ \mu_k' &= \frac{\tau_k \mu_k + c_k \bar{x}_k}{\tau_k + c_k} \end{aligned} \tag{14}$$

So that the family of densities defined by (8) is therefore a conjugate family for the complete data density. The mode of $\Psi(,\hat{\theta})$ denoted $(\hat{\omega}_k, \hat{m}_k)$ may be obtained from the modes of the Dirichlet and normal distribution as: $\hat{\omega}_k = (\nu_k' - 1)/\sum_{k=1}^{K}(\nu_k' - 1)$, $\hat{m}_k = \mu_k'$, which are actually MAP re-estimations $\hat{\theta}$. Thus, we can resume:

If the prior mean is $\boldsymbol{\mu}_0$, then the MAP estimate for the adapted mean $\hat{\boldsymbol{\mu}}$ of Gaussian is given by

$$\hat{\boldsymbol{\mu}} = \frac{\tau \boldsymbol{\mu}_0 + \sum_n \gamma(n) \boldsymbol{x}_n}{\tau + \sum_n \gamma(n)} \tag{15}$$

where:

- τ is a hyperparameter that controls the balance between the ML estimate of the mean, i.e. its prior value (typically, it is in the range 2–20),
- $\boldsymbol{x}_n$ is the adaptation vector at time n,
- $\gamma(n)$ is the probability of this Gaussian at this time.

As the amount of training data increases, the MAP estimate converges to the ML estimate. The main drawback to MAP adaptation is that it is local. Only the parameters belonging to Gaussians of observed states will be adapted, which is illustrated in Figure 6. Large speech synthesis systems have thousands of Gaussians, most will not be adapted, or will not be adapted to a sufficient extent. Structural MAP (SMAP) approaches have been introduced to share Gaussians. MAP adaptation is very useful for domain adaptation (e.g. adapting a conversational telephone speech system to studio recordings).

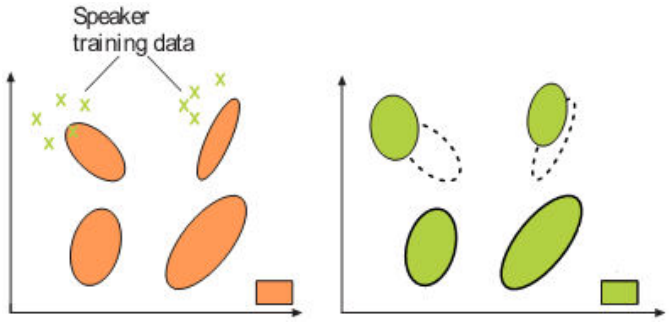


Figure 7: *Gaussian adaptation by using MAP*

3.1.3. **MLLR transformation**

Another approach is to estimate a set of transformations that can be applied to model parameters. If these transformations can capture general relationships between the original model set and the current speaker or new acoustic environment, they can be effective in adapting all HMM distributions. One such transformation approach is MLLR [22][23], which estimates a set of linear transformations for the mean parameters of a mixture Gaussian HMM system, so that the likelihood of the adaptation data is maximized. As many components are assumed to share the same transformation, it is possible to adapt all the components of the system with little data. It should be noted that while MLLR was initially developed for speaker adaptation, since it reduces the mismatch between a set of models and adaptation data, it can also be used for speaking style adaptation, or even to perform environmental compensation by reducing a mismatch due to channel or additive noise effect.

If the original mean vector for a certain state (i.e. its Gaussian component) is $\boldsymbol{\mu}$, then the new estimate of the mean, $\hat{\boldsymbol{\mu}}$, is obtained by

$$\hat{\boldsymbol{\mu}} = \widehat{\boldsymbol{W}}\boldsymbol{\xi} \tag{16}$$

where $\widehat{\boldsymbol{W}}$ is the $n \times (n+1)$ transformation matrix (n is the dimensionality of the data) and $\boldsymbol{\xi}$ is the extended original mean vector:

$$\boldsymbol{\xi} = [1\ \mu_1\ \ldots\ \mu_n]^T \tag{17}$$

We can write $\widehat{\boldsymbol{W}}$ in the form:

$$\widehat{\boldsymbol{W}} = [\widehat{\boldsymbol{b}}\ \widehat{\boldsymbol{A}}] \tag{18}$$

where $\widehat{\boldsymbol{b}}$ is a bias on the mean and $\widehat{\boldsymbol{A}}$ is a transformation matrix, which may be full, block diagonal, or diagonal. The aim is to find the transformation $\widehat{\boldsymbol{W}}$ that maximizes the likelihood of the adaptation data.

Gaussian covariance matrices are updated by

$$\boldsymbol{\Sigma} = \boldsymbol{B}^T \widehat{\boldsymbol{H}}\ \boldsymbol{B} \tag{19}$$

where $\widehat{\boldsymbol{H}}$ is the linear transformation to be estimated and $\boldsymbol{B}$ is the inverse of the Choleski factor of $\boldsymbol{\Sigma}^{-1}$, so:

$$\boldsymbol{\Sigma}^{-1} = \boldsymbol{C}\ \boldsymbol{C}^T \tag{20}$$

and:

$$\boldsymbol{B} = \boldsymbol{C}^{-1} \tag{21}$$

The entire optimization procedure is described in [24].

Two problems should be resolved when performing MLLR adaptation. The first problem is to decide how components should be clustered together, so that they all can share the same transformation matrix. The second problem is how to decide how many transformations to generate, given a particular set of adaptation data.

To better illustrate the whole procedure, let us look at Figure 7, which depicts parametric representation of one phoneme in context (e.g. P-**A**+D). Generally, context is much more complex than this, and usually contains a wider window of phonetic context, linguistic context, prosodic context (ToBI tags or like), etc. Parameters (1 and 2) can be any acoustic features (spectrum (MFCCs), pitch, aperiodicity) and while the example is presented in a 2D acoustic space, in practice the dimension of this space is much higher. A0, A1 and A2 are consecutive states of the phoneme in context. Parameter values are calculated from the database of speaker X during the training procedure of HMM-GMM.

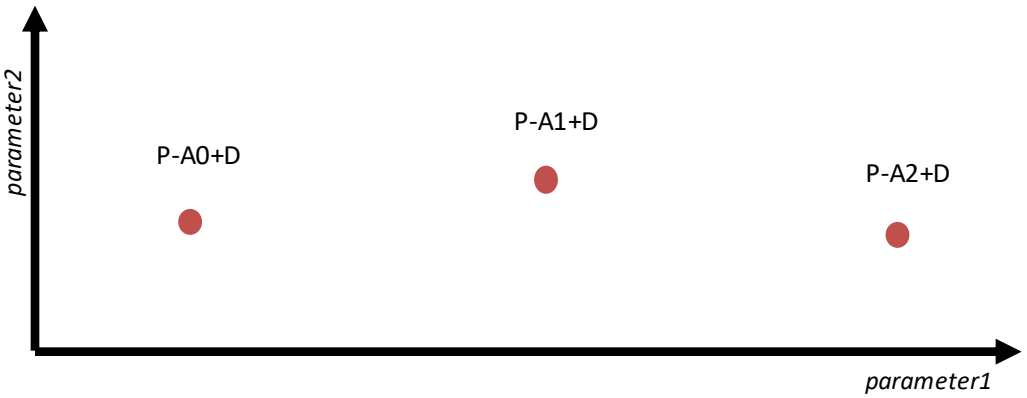


Figure 8: *Parametric representation of phoneme (actually, its states) in context*

During enrollment (providing speech adaptation data), speaker Y produced some acoustic observations – his/her versions of the same phoneme (i.e. its states). All these parameters are calculated per frame (e.g. 5 ms), therefore several observations exist for each state (Figure 8). Each observation is assigned to the appropriate state, which is accomplished by the process of forced alignment.

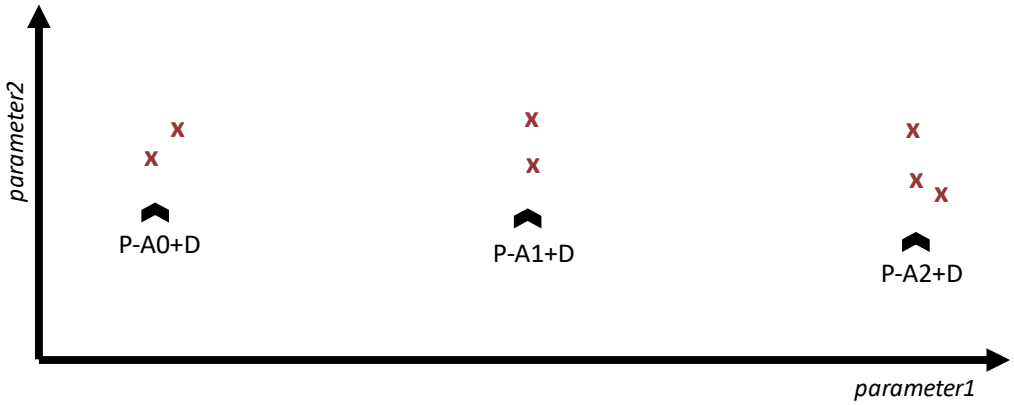


Figure 9: *Observations from the adaptation sequence*

In the next step, states from speaker X are transformed "towards" speaker Y's observations. Since we know "what should go where", it is possible to calculate a transformation. We are not transforming each state separately, but rather calculate a joint transformation for some cluster of states (Figure 9). This provides more robust results. A logical approach is to transform speaker X parameters (of each state) by using the

precalculated transformation. After this, we basically have speaker Y parameters, instead of X.

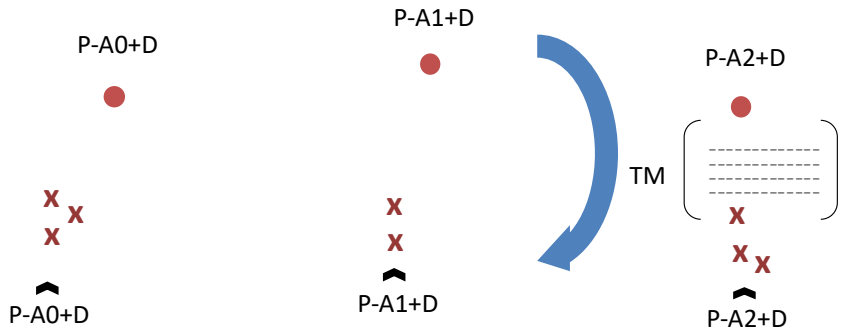


Figure 10: *Calculation of transformation matrix*

In the model X there will be numerous states which do not have their representatives in the speaker Y's observations, i.e. they are not "seen" in the training sample (Figure 10). However, they are (more or less) close to some "seen" states. The basic intuition of MLLR or similar method is that these ("unseen") states should be transformed in the same way as their close "seen" states.

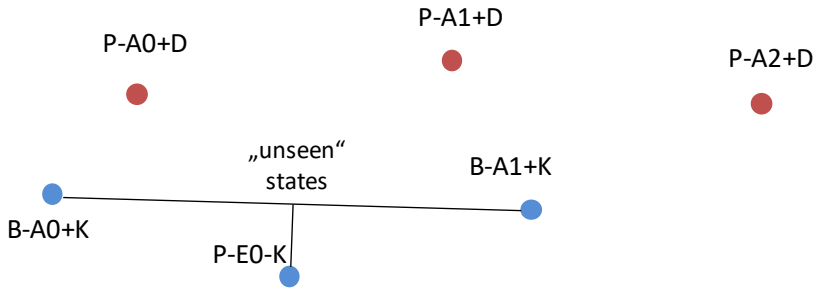


Figure 11: *Illustration of "unseen" states from model X*

By applying the same transformation to neighboring "unseen" states, we obtain new "positions" for those states as well (Figure 11). Since the same transformation was applied to those states, their relative position to the "seen" states will remain the same, which is the desired behavior of the procedure. It should be noted that initially, these three states (A0, A1 and A2) were marked as close, thus having joint transformation. This usually will not be the case, since they represent different segments of a phone. It is more likely that, e.g. P-A0+D will be similar to B-A0+T, than P-A1+D.

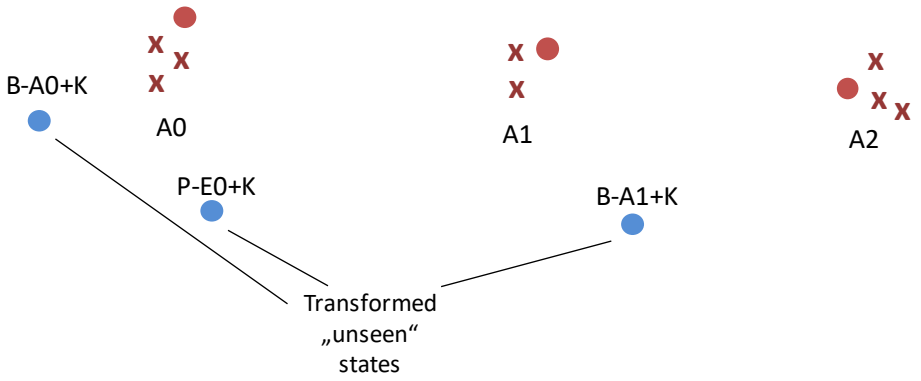


Figure 12: *Appropriate transformation is applied to "unseen" states as well*

To summarize the rationale behind MLLR:

- The whole acoustic space is clustered into regression classes of equivalence (RC).
- Each RC contains states close to each other.
- The same transformation is applied to every state in a RC.
- Some states in RC are "seen" and some are "unseen".
- Observations from the "seen" states participate in the calculation of transformation for that RC.
- If some RC has an insufficient number of observations, it is merged with some neighboring RC.
- In conclusion, if we have a large number of observations, we can handle a large number of RCs, and have better transformations. Otherwise, we are forced to merge RCs more often, and perform suboptimal transformations.

Some of the problems in MLLR are the following:

- Insufficiently robust transformations if we aim at a large number of them (small amount of observations per transformation). This is even more noticeable if we use full transformation matrices.
- Hard splitting of acoustic space (some states that were very close can become very distant after transformation, and vice versa).

4. ANN Speech Synthesis

An Artificial Neural Network (ANN) is an efficient computing system whose central theme is borrowed from the analogy of biological neural networks. ANNs are parallel computing devices, which basically attempt to represent a computer model of the brain. The main objective is to develop a system to perform various computational tasks faster than traditional systems. These tasks include pattern recognition and classification, approximation, optimization, and data clustering. ANNs are also named as "artificial neural systems" or "parallel distributed processing systems" or "connectionist systems". ANN acquires a large collection of units that are interconnected in some pattern to allow communication between the units. These units, also referred to as nodes or neurons, are simple processors which operate in parallel. Every neuron is connected with other neurons through a connection link. Each connection link is associated with a weight that multiplies the output from some other neuron or an input to the network. Weights excite or inhibit the signal that is being communicated, which is the basic mechanism by which neural network solves a particular problem. Each neuron is also characterized by its activation function. Output signals, which are produced by combining the input signals and the activation function, may be sent to other units or provided as outputs of the neural network.

Modern neural networks usually contain several layers of neurons instead of just one. With a sufficient number of layers, such an architecture is usually called a Deep Neural Network (DNN). In the remainder of this thesis, terms ANN and DNN will be used interchangeably.

4.1. ANN Basics

There are several types of architectures for neural networks:

- Feed Forward networks (FF), which contain one or more hidden layers of neurons, propagating information and signals from one layer to the next, without feedback.

- Recurrent Neural Networks (RNN), used for sequential data such as text or times series.
- Convolutional Neural Networks (CNN), particularly adapted for image processing.

All these types of networks are based on deep cascades of layers of neurons, containing large amount of parameters, thus demanding stochastic optimization algorithms, and initialization, with the emphasis on the choice of topological structure of network. Any ANN is formally described by a nonlinear continuous mapping $y = F(x, \theta) = F_\theta(x)$, where $x \in R^n$ is the input vector, $y \in R^m$ is the output vector, while $\theta \in R^p$ is the vector of network parameters. As usual in statistical learning, the parameters θ are estimated from a set of observations in a supervised (then learned on paired input and output data) or unsupervised manner (just input data is available). Having learned from data in a supervised manner, neural networks can be used for regression or classification, depending on what kind of information is brought to their output (desired output, in the case of regression, or class information in the case of classification). As function to minimize is non-convex, by design, learning procedures constitute non-convex optimization problems, and thus local minimizers are used to solve them.

4.1.1. **Artificial neurons**

An artificial neuron (illustrated in Figure 12) is a function of an input vector $\boldsymbol{x} = [x_1 \cdots x_n]^T \in R^n$, weighted by connection weights $\boldsymbol{w} = [w_1 \cdots w_n]^T \in R^n$, translated by bias $b \in R$ and then finally passed through a fixed nonlinear activation function $\varphi: R \to R$, i.e., $y = f(\boldsymbol{x}) = \varphi(\langle \boldsymbol{w}, \boldsymbol{x} \rangle + b)$, where $\langle \boldsymbol{w}, \boldsymbol{x} \rangle$ is an Euclidean scalar product in R^n, so that $\langle \boldsymbol{w}, \boldsymbol{x} \rangle = \sum_{i=1}^{n} w_i x_i$. Today, there are numerous activation functions in use [25], of which we will describe several most common ones.

The sigmoid function

$$\varphi(x) = \frac{1}{1 + \exp(-x)}, x \in R \tag{22}$$

The hyperbolic tangent (tanh) function

$$\varphi(x) = \tanh(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)} = \frac{\exp(2x) - 1}{\exp(2x) + 1}, x \in R \tag{23}$$

The hard threshold function

$$\varphi(x) = 1_{x>\beta}(x) = \begin{cases} 1, x > \beta \\ 0, x \leq \beta \end{cases}, x \in R, \beta \in R \tag{24}$$

The Rectified Linear Unit (ReLU) function

$$\varphi(x) = \max\{0, x\}, x \in R \tag{25}$$

The Leaky ReLU function

$$\varphi(x) = \max\{0, x\} + \alpha \min\{0, x\}, x \in R \tag{26}$$

where $\alpha > 0$ is fixed and usually much smaller than 1.

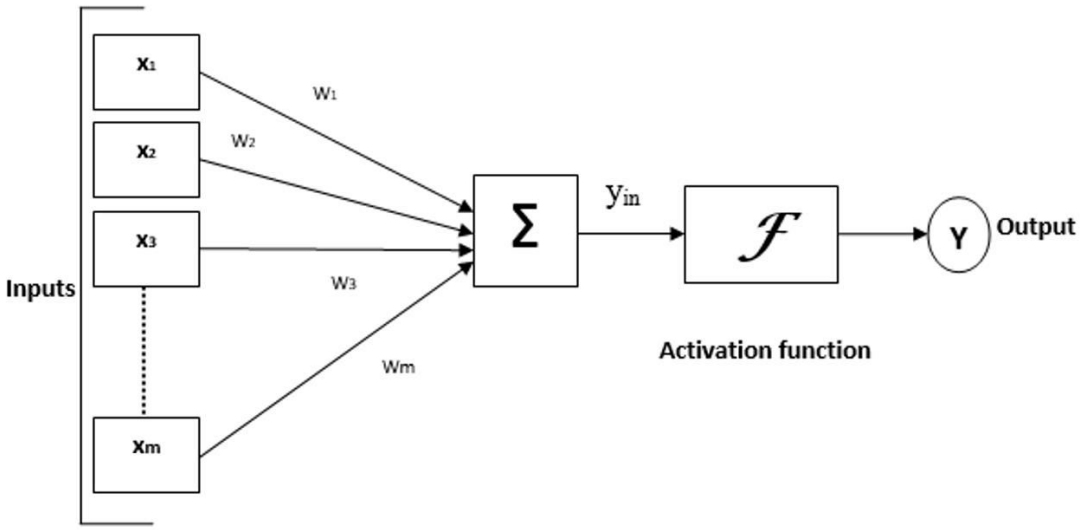


Figure 13: *General model of artificial neuron*

Although the sigmoid activation function given by (22) is a smooth function i.e. has bounded and continuous first derivative on R and has its values contained in the interval [0,1]

(which is the reason why it was largely used in ANNs with a smaller number of layers). It has the drawback that the absolute value of its derivative (and thus the whole gradient w.r.t. parameters of the ANN, due to the chain derivative rule) is close to zero on the large part of R (cases when $|x|$ is not close to zero). Namely, in the ANNs with a large number of layers (which defines the actual "Deep Learning" paradigm), due to the usage of back-propagation algorithm in learning procedure, the problem of so-called "vanishing gradient" occurs (more details in 4.2). It means that the norm of gradient becomes close to zero due to the chain differentiation rule which is executed in the process of calculating gradient w.r.t. parameters of ANN, when propagating through layers of ANN (where one actually encompasses the composition of mappings).

Contrary to the previous, in the case of ReLU activation function, the singularity at $x = 0$ is present, making the ANN learning process formally a non-smooth optimization problem. Nevertheless, the probability of encountering the zero value of the argument of the activation function is zero, so in practice it does not cause significant problems. On the other hand, as the absolute values of gradient of ReLU activation function are equal to 1, for $x > 0$, the benefit of avoiding the *vanishing gradient* problem (see Section 4.2.3) in the ANN learning procedure is huge. As the ReLU activation function and also its derivative have zero values for $x < 0$, Leaky ReLU is introduced as its modification, which avoids such a drawback.

4.1.2. **Feed Forward Networks**

Feed forward ANN is a non-recurrent network having processing units/nodes in layers and all the nodes in a layer connected with the nodes of the previous layers (Figure 13). The connections may have different weights. There is no feedback loop, which means that the signal can only flow in one direction, from input to output. One widely used example of feed forward ANN is multilayer perceptron.

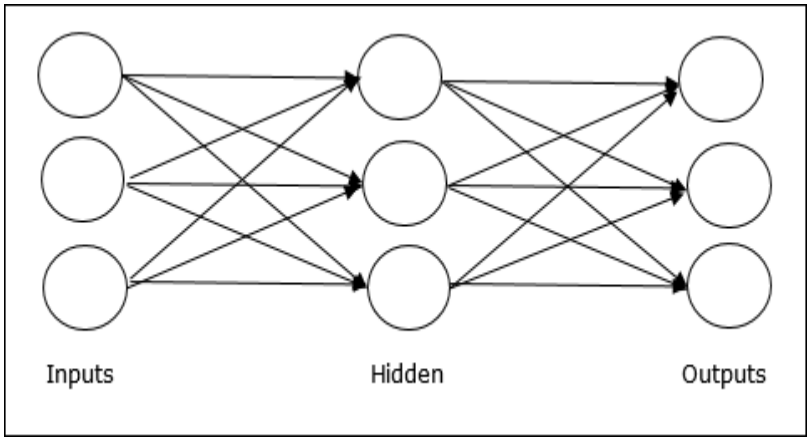


Figure 14: *An example of feed-forward neural network*

Multilayer perceptron, or a fully-connected multilayer feed-forward ANN, is a structure composed by several layers of neurons where the output of every neuron of a layer becomes the input of every neuron of the next layer. Those layers that are not input and output layers (i.e. the layers in between) are called *hidden layers*. Moreover, there is the constraint that the output of a neuron in some layer can only be the input of a neuron of the next layer (Figure 13). To the last layer, i.e. the output layer, one may apply a different activation function than in the case of hidden layers which depends on the type of problem that is tackled: regression or classification. Actually, in the regression task, no activation function is imposed on the output layer. For binary classification, the output gives an estimation of $P(y = 1|\boldsymbol{x})$. For a multi-class classification task, the output layer contains C neurons, where C is the number of classes, with one neuron representing each class $c \in \{1, \dots, C\}$ in the soft manner, giving the estimation of the conditional probability $P(y = c|\boldsymbol{x})$. As those probabilities have to sum to 1 when summing all classes, the *softmax* function is usually used for the output layer, i.e.,

$$\text{softmax}_c(\boldsymbol{z}) = \frac{\exp(z_c)}{\sum_{c=1}^{C} \exp(z_c)} \tag{27}$$

Multilayer perceptron neural network with L hidden layers can be formally formulated as follows:

$$\begin{aligned} &\boldsymbol{h}^{(0)}(\boldsymbol{x}) = \boldsymbol{x}, \\ &\boldsymbol{a}^{(k)}(\boldsymbol{x}) = \boldsymbol{W}^k \boldsymbol{h}^{(k-1)}(\boldsymbol{x}) + \boldsymbol{b}^k, \qquad k = 1, \dots, L+1, \end{aligned} \tag{28}$$

$$\boldsymbol{h}^{(k)}(\boldsymbol{x}) = \varphi\left(\boldsymbol{a}^{(k)}(\boldsymbol{x})\right), \qquad k = 1, \dots, L,$$
$$\boldsymbol{h}^{(L+1)}(\boldsymbol{x}) = \psi\left(\boldsymbol{h}^{(L)}(\boldsymbol{x})\right)$$

where $\boldsymbol{h}^{(k)}, k = 1, \dots, L$ are hidden layers, $\boldsymbol{h}^{(0)}, \boldsymbol{h}^{(L+1)}$ are input and output layers respectively, $\boldsymbol{W}^k \in R^{\dim(\boldsymbol{h}^{(k)}) \times \dim(\boldsymbol{h}^{(k-1)})}$ are network weight matrices, while $\boldsymbol{b}^k \in R^{\dim(\boldsymbol{h}^{(k)})}$ are network biases. The parameters of the network are then $\theta = \{[\boldsymbol{W}^k, \boldsymbol{b}^k], k = 1, \dots, L+1\}$. Additionally, φ is the activation function of the hidden layers, while ψ is the activation function of the output layer (in most cases chosen to be different from φ).

4.1.3. **Recurrent Neural Networks**

Feedback network has feedback paths, which means that the signal can flow in both directions using loops. This makes it a non-linear dynamic system, which changes continuously until it reaches a state of equilibrium. One example of feedback networks are recurrent networks, i.e. feedback networks with loops closed within the same layer (Figure 14).

Recurrent neural networks are introduced in order to tackle the machine learning (ML) tasks involving sequential data such as text or time series, as well as video data. Simple

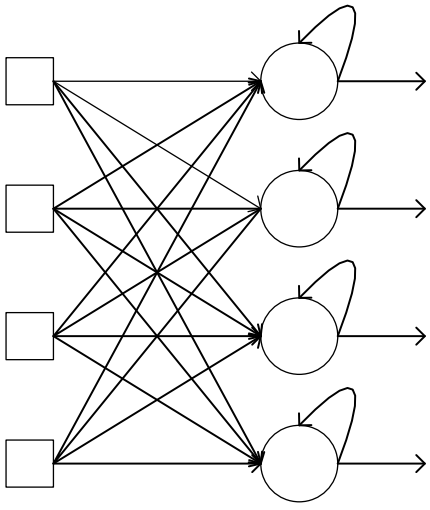


Figure 15: *An example of feedback (recurrent) neural network*

RNNs are introduced by Elman [26] and Jordan [27]. In the model introduced in [26], feed-

forward ANN with one hidden layer is looped back onto itself. It means that the following relations hold:

$$\begin{aligned} \boldsymbol{h}_t &= \sigma_h(\boldsymbol{W}_h \boldsymbol{x}_t + \boldsymbol{U}_h \boldsymbol{h}_{t-1} + \boldsymbol{b}_h), \\ \boldsymbol{y}_t &= \sigma_y(\boldsymbol{U}_y \boldsymbol{h}_t + \boldsymbol{b}_y), \end{aligned} \tag{29}$$

where $\boldsymbol{x}_t \in R^n$ is input vector at time t, $\boldsymbol{h}_t \in R^l$ is hidden layer output at time t, $\boldsymbol{W}_h \in R^{m \times n}$, $\boldsymbol{U}_h, \boldsymbol{U}_y \in R^{m \times l}$, $\boldsymbol{b}_h, \boldsymbol{b}_y \in R^m$ weight matrices and bias vectors, respectively, to be learned. Also, σ_y and σ_h are activation functions for output and hidden layer respectively. It can be seen from (47), that hidden layer at time t, depends on hidden layer at previous time, i.e., $t-1$. In the model introduced in [27], hidden layer at time t, instead of depending on the output of the hidden layer, depends on the output of the network at previous time step, i.e., the following relations hold:

$$\begin{aligned} \boldsymbol{h}_t &= \sigma_h(\boldsymbol{W}_h \boldsymbol{x}_t + \boldsymbol{U}_h \boldsymbol{y}_{t-1} + \boldsymbol{b}_h), \\ \boldsymbol{y}_t &= \sigma_y(\boldsymbol{U}_y \boldsymbol{h}_t + \boldsymbol{b}_y), \end{aligned} \tag{30}$$

where, $\boldsymbol{y}_{t-1}$ is output of the network at time step $t-1$.

The basic version of RNN fails to learn long time dependencies [28], so that new architectures have been introduced to tackle this problem. *Long Short-Term Memory* (LSTM) RNNs are introduced by Hochreiter and Schmidhuber, in order to tackle the mentioned problem and they found their application in many tasks such as speech recognition, translation, etc. The main difference between LSTM and simple RNN is the following: an LSTM cell at time t, contains not only the hidden layer $\boldsymbol{h}_t \in R^l$, but also a state $\boldsymbol{c}_t \in R^q$. This cell at time t, is the function of the following vectors: current input to the hidden layer $\boldsymbol{x}_t$, hidden layer output at previous timestep $\boldsymbol{h}_{t-1}$, as well as the previous cell state $\boldsymbol{c}_{t-1}$. Inside the LSTM, the *gates* are defined that decide on the transmission of information, so that LSTM cell is described by the following set of equations:

$$\begin{aligned}
&\text{Update gate: } \boldsymbol{u}_t = \boldsymbol{\sigma}(\boldsymbol{W}_u \boldsymbol{h}_{t-1} + \boldsymbol{I}_u \boldsymbol{x}_t + \boldsymbol{b}_u),\\
&\text{Forget gate: } \boldsymbol{f}_t = \boldsymbol{\sigma}(\boldsymbol{W}_f \boldsymbol{h}_{t-1} + \boldsymbol{I}_f \boldsymbol{x}_t + \boldsymbol{b}_f),\\
&\text{Cell candidate: } \tilde{\boldsymbol{c}}_t = \mathbf{tanh}(\boldsymbol{W}_c \boldsymbol{h}_{t-1} + \boldsymbol{I}_c \boldsymbol{x}_t + \boldsymbol{b}_c),\\
&\text{Cell output: } \boldsymbol{c}_t = \boldsymbol{f}_t \circ \boldsymbol{c}_{t-1} + \boldsymbol{u}_t \circ \tilde{\boldsymbol{c}}_t,\\
&\text{Output gate: } \boldsymbol{o}_t = \boldsymbol{\sigma}(\boldsymbol{W}_o \boldsymbol{h}_{t-1} + \boldsymbol{I}_o \boldsymbol{x}_t + \boldsymbol{b}_o),\\
&\text{Hidden output: } \boldsymbol{h}_t = \boldsymbol{o}_t \circ \mathbf{tanh}(\boldsymbol{c}_t)
\end{aligned} \tag{31}$$

where $\boldsymbol{W}_f$, $\boldsymbol{W}_u$, $\boldsymbol{W}_c$, $\boldsymbol{W}_o$, $\boldsymbol{W}_y$ are weight matrices of adequate dimensions, while $\boldsymbol{b}_u$, $\boldsymbol{b}_f$, $\boldsymbol{b}_c$, $\boldsymbol{b}_o$, $\boldsymbol{b}_y$ are biases of adequate dimensions, $\circ$ is point-wise multiplication, while $\boldsymbol{\sigma}(\cdot)$ and $\mathbf{tanh}(\cdot)$ are vector-level activation functions.

4.1.4. **Convolutional Neural Networks**

For image type of data, multilayer perceptrons are not adequate, as images should be vectorized, thus losing the spatial information contained in the images, such as forms, geometry, texture, etc. CNNs introduced by LeCun in [29] revolutionized image processing, and removed the need for manual extraction of features, which was essential in that area and demanded specific expert knowledge. Namely, CNNs act directly on matrices, or even on tensors for images with three RGB color channels. A Convolutional Neural Network is composed of the following layers: convolutional layers, pooling layers and fully connected layers.

Convolutional layer

The cross-correlation between two 2D (discretized) finite sequences $\boldsymbol{u}$ and $\boldsymbol{v}$ is a finite 2D sequence defined as

$$(\boldsymbol{u} * \boldsymbol{v})(i,j) = \sum_{m,n} \boldsymbol{u}(m,n)\boldsymbol{v}(m+i,n+j) \tag{32}$$

Even though CNNs use this formula, they are called convolutional, thus disregarding the strict definition of convolution. For example, one can set $\boldsymbol{u} = \boldsymbol{K}$ to be the convolution kernel acting on grayscale image signal $\boldsymbol{v} = \boldsymbol{I}$, but the definition (32) could be generalized onto

finite sequences of arbitrary dimension, so for example, $\boldsymbol{I}$ could be an image containing RGB channels (thus a 5D finite sequence) and thus $\boldsymbol{K}$ would be a corresponding 5D kernel.

In practice 2D convolution is calculated by dragging convolution kernel $\boldsymbol{K}$ throughout the image $\boldsymbol{I}$. At each position, we get the convolution between the kernel and the part of the image that is currently treated. Then, the kernel moves by a number s of pixels, s is called *stride*. When the stride is small, we get redundant information. Sometimes, one can also add a *zero padding*, which is a margin of size p containing zero values around the image in order to control the size of the output. Assume that we apply C_o kernels (also called filters), each of size $k \times k$ on an image. If the size of the input image is $W_i \times H_i \times C_i$ (W_i denotes the width, H_i the height, and C_i the number of channels), the format of the output is $W_o \times H_o \times C_o$, where C_o corresponds to the number of kernels that we consider, and the following holds:

$$\begin{aligned} W_o &= \frac{W_i - k + 2p}{s} + 1, \\ H_o &= \frac{H_i - k + 2p}{s} + 1. \end{aligned} \tag{33}$$

In convolution layer of a CNN, convolution operations are further composed with an activation function φ as well as bias $\boldsymbol{b}$, as $o = \varphi(\boldsymbol{K} * \boldsymbol{u} + \boldsymbol{b})$, where $\boldsymbol{u}$ is the input and o is the output of the convolution layer.

In the particular learning process (similar to ANNs described in previous sections) the CNN will learn convolution kernels that are the most useful for the given task.

Pooling layer

CNN also has pooling layers, which allow it to reduce the dimension, also referred as sub-sampling, by taking the mean or the maximum on patches of the image (mean-pooling or max-pooling). Like the convolution layers, pooling layers act on small patches of the image, and we also have a stride. If we consider 2×2 patches, over which we take the maximum value to define the output layer, and a stride $s = 2$, we divide by 2 the width and height of the image. Of course, it is also possible to reduce the dimension using the convolution layer, by taking a stride larger than 1, and without zero padding, but another advantage of pooling is that it makes the network less sensitive to small translations of input images.

Fully connected layer

After several convolution and pooling layers, the CNN usually ends with several fully-connected perceptron layers, depending on the particular task.

4.1.5. **Dense vs. Sparse Layers**

In general, neural networks are represented as tensors. Each layer of neurons is represented by a matrix. Each entry in the matrix can be thought of as representative of the connection between two neurons. In a simple neural network, like a classic fully-connected (dense) feed-forward neural network, every neuron on a given layer is connected to every neuron on the subsequent layer. This means that each layer must have n^2 connections represented, where n is the size of both of the layers. In large networks, this can take a lot of memory and time to propagate. Since different parts of a neural network often work on different subtasks, it can be unnecessary for every neuron to be connected to every neuron in the next layer. In fact, it might make sense for a neural network to have most pairs of neurons with a connection weight of 0. Training a neural network might result in these less significant connection weights adopting values very close to 0 but accuracy would not be significantly affected if the values were exactly 0.

A matrix in which most entries are 0 is called a sparse matrix. These matrices can be stored more efficiently and certain computations can be carried out more efficiently on them provided the matrix is sufficiently large and sparse. Neural networks can leverage the efficiency gained from sparsity by assuming most connection weights are equal to 0.

4.1.6. **Approximation capabilities of ANN**

Hornik [30] has shown that any bounded and continuous mapping between Euclidean spaces can be approximated with arbitrary precision by a neural network with one hidden layer containing a finite (but possibly very large, often unacceptable, from the computational point of view) number of neurons, having the same activation function, and one linear output neuron (no activation function on that neuron). This result was earlier proved by Cybenko [31] in the particular case of the sigmoid activation function given by (22). The *Universal Approximation Theorem* (by Cybenko) can be formally stated as:

Theorem 1: Let φ be a continuous and bounded activation function additionally non-decreasing. Let $K \subset R^n$ be compact in R^n (i.e., bounded and closed). Then, for arbitrary fixed continuous $f: K \to R$, there exists the function $G_s: K \to R$ defined by $G_s(\boldsymbol{x}) = \sum_{i=1}^{s} v_i(\langle w_i, \boldsymbol{x} \rangle + b_i)$, that could be made arbitrary close to f in the space of the continuous functions on K, denoted by $C(K)$ (equipped with the supremum norm $\sup_{\boldsymbol{x} \in K} |\cdot|$), by increasing s. This means, that for arbitrary fixed $\varepsilon > 0$ there exist $s \in N$, $w_i \in R^n$, $b_i, v_i \in R$ such that $\sup_{\boldsymbol{x} \in K} |f(\boldsymbol{x}) - G_s(\boldsymbol{x})| < \varepsilon$ holds.

The Theorem extends straightforward to the cases of ANNs with a finite number of hidden layers and a finite number of outputs (the case $f: K \to R^m$) . Note that all activation functions defined by (22)-(26) satisfy the regularity condition from Theorem 1.

Another important theoretical result in a similar manner (approximation possibility) is delivered for networks with a large number of hidden layers and not necessary a large number of neurons in particular layers. It is formulated by Zhou et all, as [32]:

Theorem 2: For any integrable function $f: R^n \to R$ and arbitrary fixed $\varepsilon > 0$, there exists a fully-connected feed-forward ANN with finite depth (number of hidden layers) with ReLU activation functions (defined by (25)), with the width d (i.e., number of neurons in arbitrary hidden layer) satisfying $d \leq n + 4$, given by its function $G: R^n \to R$, such that $\|f - G\|_1 < \varepsilon$, where $\|\cdot\|_1 = \int_{R^n} |\cdot| \, \mathbf{dx}$ is l_1 norm, defined on set of all absolutely integrable functions on R^n.

Note that the width of the DNN in Theorem 2, cannot be arbitrary small, but the upper bound of $n + 4$ is given, which depends on the number of inputs to the network. Of course, a lower width must be compensated by a larger depth.

Note first, that the nonlinearity (actually, non-polynomiality) of activation functions mentioned in Theorems 1 and 2 is crucial for good approximation properties of ANNs and DNNs, meaning that there are counter-examples if it is not involved (see [32]). Note second, that many other systems of functions have good approximation properties in various normed spaces of functions (for example polynomials in spaces of continuous or absolutely integrable functions, etc.) but did not come close to reaching the level of applicability, as it is

the case for ANNs. Note finally, that although Theorems 1 and 2 explain from the theoretical point of view, good approximation properties of ANNs, the actual "learning" of ANNs (i.e., obtaining optimal parameters of some ANN, i.e. those that minimize specific target criteria driven by specific task, with available data) "is the task of its own", encompassing many specific optimization and computational issues as well as heuristics, in order to efficiently reach the theoretical bounds marked by previous theorems.

4.1.7. **Learning in ANN**

A Machine Learning (ML) system is is formally a mapping $F_\theta: X \to Y$, where X and Y are input and output domains (formally represented as $X = R^n$, $Y = R^m$) respectively and $\theta \in \Theta \subseteq R^p$ are parameters of the system. Learning in the broader context of ML, is obtaining the optimal parameters of ML system $\hat{\theta} \in \Theta$, in the sense that those are minimizers of the specific target function which is formed according to the specific task, subject to certain specified constraints (that define Θ), with available data that are also associated to that particular task. Thus, learning in the context of artificial neural networks, is the process of modifying the parameters of ANNs, i.e. weights of connections between the neurons of a specified network as well as biases, so that the specified target function is minimized and it is all driven by the available task specific data. From the Statistical Estimation Theory perspective, training of ML system is process of estimating the parameters of ML system so that the expected loss is minimized (i.e., the previously mentioned task specific target function becomes some specified expected loss) where the available task specific data is considered to be observations (i.e. vector of random variables) drawn from some underlying unknown joint probability distribution $p(x, y)$. Note that the optimization problem associated to the ANN learning process is almost in all cases obtained as unconstrained, i.e., $\Theta = R^p$. Learning in ANN can be roughly classified into three categories: supervised, unsupervised and reinforcement learning.

Supervised Learning. As the name suggests, this type of learning is done under supervision, meaning that the paired input-output observations $(x_1, t_1), \dots, (x_M, t_M)$, drawn from unknown underlying probability distribution $p(x, y)$ are available as data during the training. Those data are referred as *labeled data*, and t_i are referred as *target* vectors or

labels. The target criterion function that is optimized during training is thus function of known (x_i, t_i) as well as the parameters θ (to be determined) of the network. In a more illustrative manner, the process could be explained as follows: During the training of ANN by supervised learning, the particular input vector x_i is presented to the network, which will give an output vector y_i (Figure 15). This output vector is compared with the desired target vector t_i and an error signal $|t_i - y_i|$ is generated as a measure of the difference between the actual output and the desired target vector. On the basis of this error signal, the weights are adjusted until the actual output is matched with the desired output. More precisely, the previously mentioned target function is obtained as a function of those error signals (e.g. the square error function defined by $\sum_{i=1}^{M}(t_i - y_i)^2$) .

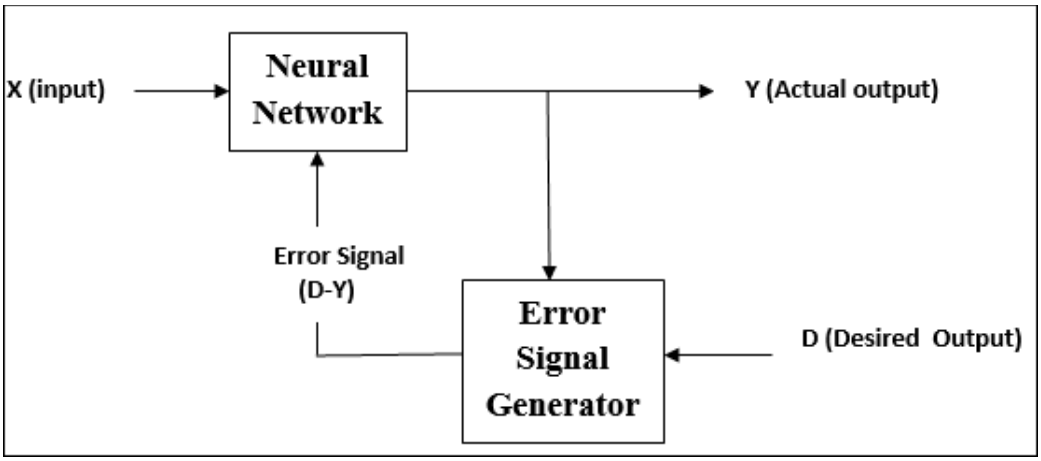


Figure 16: *Supervised learning scheme*

Unsupervised learning is carried out without supervision, meaning that only the observations $(x_1, \ldots, x_M)$ drawn from the unknown marginal probability distribution $p(x) = \int p(x, y)\mathrm{dy}$ (or $\sum_y p(x, y)$), are available during the training, i.e., no labeled data is present. Such data are referred to as *unlabeled data.* The target criterion function that is optimized during training is thus the function of x_i as well as the parameters θ of the network, to be determined. During the training of ANN by unsupervised learning, the parameters of the network tend to those that tend to group the input vectors of similar type into clusters, where the clustering is driven by the type of target criteria used in training. When a new input pattern is applied, the neural network gives an output response indicating the cluster to which

the input pattern belongs (Figure 16). There is no feedback from the environment as to what should be the desired output and if it is correct or incorrect. Hence, in this type of learning, the network itself must discover, trough the learning (i.e., training) process, the patterns and features from the input data, and the relation between the input data and the output.

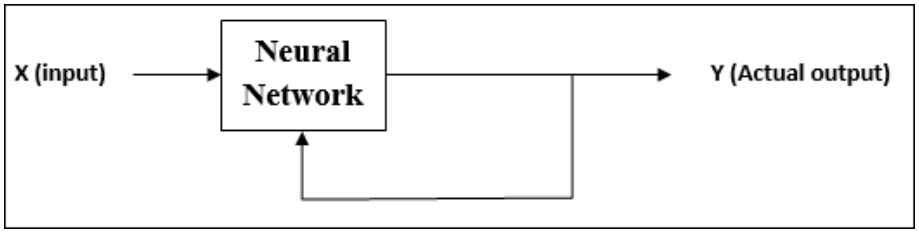


Figure 17: *Unsupervised learning scheme*

Reinforcement learning is concerned with how software agents ought to take actions in an environment in order to maximize the notion of cumulative reward. Reinforcement learning differs from supervised learning in not needing labelled input/output pairs to be presented, and in not needing sub-optimal actions to be explicitly corrected. Instead, the focus is on finding a balance between exploration (of uncharted territory) and exploitation (of current knowledge).

During the training of network under reinforcement learning, the network receives some feedback from the environment (Figure 17). When the agent's performance is compared to that of an agent that acts optimally, the difference in performance gives rise to the notion of regret in decision theory. In order to act near optimally, the agent must reason about the long-term consequences of its actions (i.e., maximize future income), although the immediate reward associated with this might be negative.

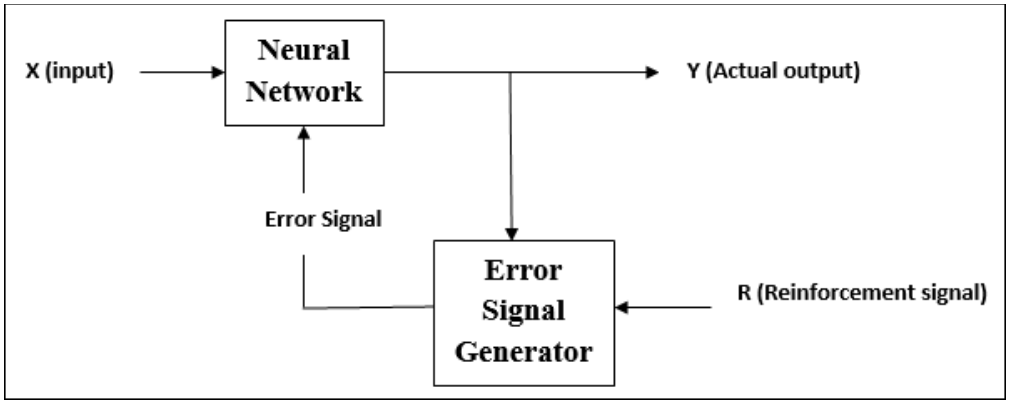


Figure 18: *Reinforcement learning scheme*

Thus, reinforcement learning is particularly well-suited to problems that include a long-term versus short-term reward trade-off. It has been applied successfully to various problems, including robot control, elevator scheduling, telecommunications, backgammon, checkers and go (AlphaGo).

4.1.8. **Target Criterion Functions**

In forming the Target criterion function to be minimized during ANN learning process (supervised learning case), a convenient approach is to use the statistical perspective and to maximize the expected log likelihood, i.e., to minimize the following loss:

$$l(\theta) = -E_{p(x,y)}\left[\ln p(y \mid x, \theta)\right] \tag{34}$$

For a regression task, if we assume a Gaussian error model, we obtain the quadratic loss, i.e.

$$l(\theta) = -E_{p(x,y)}\left[\left\|F_\theta(x) - y\right\|^2\right] \tag{35}$$

which is most common, and is used in the following Sections of this thesis.

For a binary classification task, one has $y \in \{0,1\}$, and the following holds:

$$p(y|x,\theta) = F_\theta(x)^y (1 - F_\theta(x))^{1-y} \tag{36}$$

so that one obtains:

$$L(\theta) = -E_{p(x,y)}[y\ln F_\theta(x) + (1-y)(1-F_\theta(x))] \tag{37}$$

while, by same reasoning, for multiclass classification task, one obtains:

$$L(\theta) = -E_{p(x,y)}[\sum_{c=1}^{C} 1_{y=c} \ln p(y=c|x,\theta)] \tag{38}$$

where:

$$1_{y=c}(y) = \begin{cases} 1, y = c \\ 0, y \neq c \end{cases} \tag{39}$$

and $p(y=c|x,\theta)$ in ANNs are expressed by using softmax activation function in the output layer, as discussed previously.

It should be noted that one can express the loss in the following way:

$$L(\theta) = E_{p(x,y)}[l(F_\theta(x), y)] \tag{40}$$

as, for example in the quadratic loss case (35), where $l(F_\theta(x), y) = \|F_\theta(x) - y\|^2$. As the true underlying joint probability distribution $p(x,y)$ is always unknown, we use (supervised case) available paired data samples $\{(x_i, y_i)\}_1^M$, in order to estimate unknown parameters θ and minimize the only available Empirical loss defined as

$$L(\theta) = \frac{1}{M}\sum_{i=1}^{M} l(F_\theta(x_i), y_i). \tag{41}$$

Furthermore, in some cases, mostly for the uniqueness of the solution (one could interpret it as convexification of loss function $L(\cdot)$, since a strictly convex function has the unique minimizer on any convex closed set), we add the convex regularization term $\Psi()$, so that the loss function becomes:

$$L(\theta) = \frac{1}{M}\sum_{i=1}^{M} l(F_\theta(x_i), y_i) + \lambda\Psi(\theta) \tag{42}$$

for some fixed $\lambda > 0$. The regularization term also has its statistical interpretation, in the sense that it is associated to the prior to network parameters as $p(\theta) \propto \exp(-\lambda\Psi(\theta))$, so that

in that case, instead of ML estimate expressed through (34), we perform a MAP estimate, i.e., minimize $L(\theta) = -E_{p(x,y)}[\ln p(\theta|x,y)]$ and, as for the posterior the following holds:

$$p(\theta|x,y) \propto p(y|x,\theta)p(\theta) \tag{43}$$

one obtains (41) as the loss. Most common regularization terms are quadratic (or l_2), defined as $\Psi() = \|\cdot\|_2^2$, where $\|\cdot\|_2$ is the Euclidean norm in R^p, so that $\Psi(\theta) = \sum_{i=1}^{p} \theta_i^2$, as well as sparse (or l_1), defined as $\Psi() = \|\cdot\|_1$, where $\|\cdot\|_1$ is the l_1 norm in R^p, so that $\Psi(\theta) = \sum_{i=1}^{p} |\theta_i|$.

Next, the question of optimization method to be used (gradient descent based), as well as the way of efficiently calculating the gradient, is to be discussed.

4.2. Gradient Descent And Back Propagation

Gradient descent, also known as the steepest descent, is an iterative optimization algorithm used to find a local minimum of a function. While minimizing the function, we are concerned with the cost or error to be minimized. This algorithm is extensively used in deep learning, which is useful in a wide variety of situations. The point here to be remembered is that we are concerned with local optimization and not global optimization (Figure 18).

We can understand the main working idea of gradient descent with the help of the following steps:

- First, start with an initial guess of the solution.
- Then, calculate the gradient of the function at that point.
- Subsequently, repeat the process by stepping the solution in the direction contrary to the gradient.

By following the above steps, the algorithm will eventually converge to a point where the gradient is equal to zero.

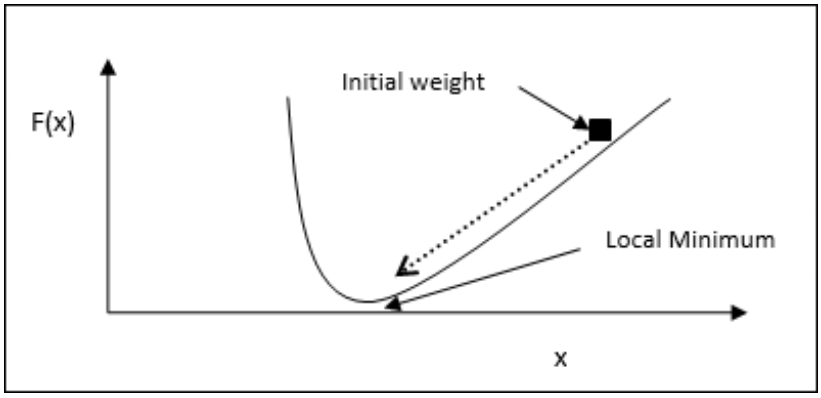


Figure 19: *Gradient descent illustration*

Gradient descent is at the core of error back propagation algorithm [33].

4.2.1. **Stochastic Gradient Descent Optimization Algorithm**

In application of ANNs, there is, in almost all cases, a large amount of training data and a large size of the ANN parameter space. For that reason, conventional gradient descent (unconstrained) optimization algorithm is inappropriate, since the calculation of high dimensional gradient on a whole corpus of training data is computationally too expensive. On the other hand, empirical risk (41) to be minimized in an ANN learning optimization task is the sum of terms $l(F_\theta(x_i), y_i)$ evaluated for each training data observation pair (x_i, y_i) separately, so that the iteration of the classical *Gradient Descent* (GD) algorithm can be expressed as

$$\theta_{k+1} = \theta_k - \mu \nabla_\theta L(\theta)|_{\theta_k} = \theta_k - \mu \sum_{i=1}^{M} \nabla_\theta l(F_\theta(x_i), y_i)|_{\theta_k} \tag{44}$$

where $\nabla_\theta L(\theta)|_{\theta_k} = [\partial L/\partial \theta_1 \cdots \partial L/\partial \theta_p]|_{\theta_k}$ is actually the gradient of $L(\theta)$ evaluated at θ_k($\partial L/\partial \theta_i$ are partial derivatives of L with respect to θ_i). In applications in statistics, due to the usage of exponential families of distributions (such as Gaussian, Laplace, etc.) there is a simple close-form expression for summation of terms $l(F_\theta(x_i), y_i)$ and $\nabla_\theta l(F_\theta(x_i), y_i)$ in (44). Contrary to that, in the cases of ANNs, $\nabla_\theta l(F_\theta(x_i), y_i)$ require expensive evaluations and the evaluation of $\nabla_\theta l(F_\theta(x_i), y_i)$ for all observations, in each iteration step, which thus

becomes unacceptable computationally expensive. Thus, *Stochastic Gradient Descent* (SGD) algorithm is introduced, where the gradient $\nabla_\theta L(\theta)|_{\theta_k}$, instead of its precise value $\sum_{i=1}^{M} \nabla_\theta l(F_\theta(x_i), y_i)\,|_{\theta_k}$ is approximated by

$$\tilde{\nabla}_\theta L(\theta) = \frac{1}{m}\sum_{i\in B} \nabla_\theta l(F_\theta(x_i), y_i)\,|_{\theta_k} \tag{45}$$

where $B \subset \{1, \dots, M\}$ is the subset of observation indices of (always fixed) size m, chosen randomly (uniformly, without replacement), for each iteration k. This subset of observations is called a minibatch, while iteration over all the training examples (all minibatches) is called an epoch of SGD algorithm.

4.2.2. **Error Back Propagation Algorithm**

The goal of the Error Back Propagation (EBP) algorithm is to find the gradients of loss function versus ANN parameters, i.e., partial derivatives of loss function with respect to all weight coefficients, which are to be used in the process of certain loss function minimization. Here we present the method for the quadratic, i.e., mean squared error (MSE) loss function defined by (35), which we write as

$$E = \frac{1}{2}\sum_{j=1}^{K} (y_j - t_j)^2, \tag{46}$$

where *K* represents the number of neurons in the output layer, y_j is the current output of the neuron at position *j,* and t_j is the target output value.

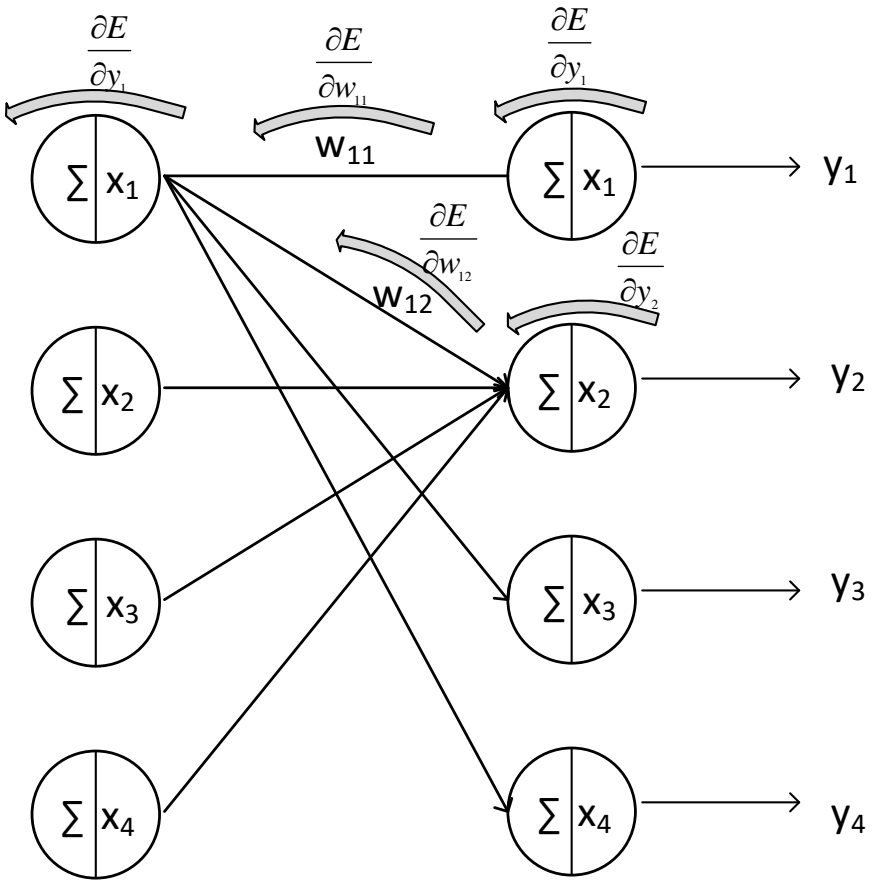


Figure 20: *Hidden and output layer of neural network*

A portion of the network is shown in Figure 19 where some neuron connections have been omitted, for the sake of simplicity. For the purpose of determining how much each output y_j influences the total loss function, we have to calculate partial derivatives

$$\frac{\partial E}{\partial y_j} = y_j - t_j. \tag{47}$$

What we really need to find out is how a change of certain coefficient influences the loss value, i.e. we have to calculate partial derivatives of the loss function with respect to parameters of ANN. Thus, due to the chain rule for derivatives, we first need to find how the change of neuron input x influences the loss value.

$$\frac{\partial E}{\partial x_j} = \frac{\partial E}{\partial y_j}\frac{dy_j}{dx_j}. \tag{48}$$

Algorithm: Optimization of neural network weight coefficients

Input
Set of training samples $(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2) \dots (\boldsymbol{x}_m, y_m)$
Learning rate

Output
Optimized weight coefficients

Set initial values of weight coefficients
For each training sample
1. *Calculate outputs of each neuron in the network (forward pass)*
2. *Calculate loss as the difference between target value and network output*
3. *Calculate gradients for each weight coefficient in the network*
4. *Calculate update weight coefficients by using gradients*

The derivative dy_j/dx_j depends on the shape of activation function. By using previous equations, the influence of the weight coefficient w_{ij} which connects the neuron i in one layer with the neuron j in the next layer, on the loss function value are actually the partial derivatives of the loss function with respect to parameters of ANN (we exclude biases, for simplicity). By the chain rule for derivatives, those are given as:

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial x_j}\frac{\partial x_j}{\partial w_{ij}}. \tag{49}$$

Since input x_j to the neuron j is equal to the linear combination of the outputs of neurons from the previous layer i.e. $x_j = \sum_k w_{kj} y_k$, we have $\frac{\partial x_j}{\partial w_{ij}} = y_i$, so the equation (49) can be written as

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial x_j} y_i. \tag{50}$$

If we observe only the previous hidden layer, then the influence of neuron i output in the hidden layer on the total loss can be described by the sum of the values given in equation (50):

$$\frac{\partial E}{\partial y_i} = \sum_j \frac{\partial E}{\partial w_{ij}}. \tag{51}$$

Partial derivatives given by (50) are called *gradients* and are usually denoted as Δw_{ij}. They are used in the calculation of new weight coefficients in some iteration of a (conventional) GD algorithm, by using:

$$\widehat{w}_{ij} = w_{ij} - \xi \Delta w_{ij}, \tag{52}$$

where ξ is a scalar value which represents the learning rate, and is set heuristically, or could be varied in every iteration.

As we have already stated previously, weights could be updated after each sample in the training database, which would generate very noisy gradients and would introduce significant overhead in calculations. The opposite extreme would be to perform the update only after accumulating gradients over the entire training database. This would provide a very stable update, but would take a huge amount of time to converge. An intermediate approach is usually applied, in which a certain portion of the training database, referred to as *batch*, is processed, the gradients are averaged on that batch, and the update is performed. This eliminates overhead, reduces noise (although a certain amount of noise is actually desirable, especially in the beginning of the training) and leads to reasonably quick convergence.

This baseline algorithm is still quite slow, and can be accelerated by using numerous techniques proposed in literature. One of the ways to do this is to slightly modify the update calculation based on *momentum* [34], as follows:

$$\Delta w(t) = -\xi \frac{\partial E}{\partial w} + \alpha \Delta w(t-1), \tag{53}$$

where α is a scalar value. In this way, the modification of weights and the entire training process will proceed in smaller steps if the directions of the most recent steps vary significantly, and it will proceed in bigger steps if this variation is smaller.

4.2.3. **Vanishing Gradient Problem**

DNNs may be hard to train because of the way in which the gradients in previous and next layer are related, and the fact that there is a large number of such layers. In order to explain the vanishing gradient problem, let us, without loss of generality, assume that DNN has a single node at each hidden layer, and that it is H hidden layers deep, denoted as $h_1, \dots, h_H$. Let us denote the weights between subsequent layers (i.e., nodes) as $w_1, \dots, w_H$

and assume that the sigmoidal activation function $\varphi(\cdot)$ given by (22) is applied at each layer. Let also y be the final output of the network, and x be the input. Namely, for some given loss function L, due to the derivative chain rule, we obtain

$$\frac{\partial L}{\partial h_t} = \varphi'(h_{t+1}) w_{t+1} \frac{\partial L}{\partial h_{t+1}}. \tag{54}$$

Let us assume that weights w_t are initialized from the standard normal distribution with expected value equal to 1. It holds that $\varphi'(x) = \varphi(x)(1 - \varphi(x))$, $x \in R$, $|\varphi'(x)| < 1$ and has its supremum equal to 0.5, so that the value of $|\varphi'(h_{t+1})|$ can not exceed 0.25. Since the expectation of w_{t+1} is 1, it follows that each weight update will typically cause the value of $\frac{\partial L}{\partial h_t}$ to be less than $0.25 \frac{\partial L}{\partial h_{t+1}}$. Therefore, after moving by about r layers, this value will typically be less than 0.25^r. As a consequence, during back propagation, lower layers will receive much smaller updates than the upper layers, which is referred as the vanishing gradient problem. Using different activation function with larger gradients helps, but it is a tradeoff as the opposite situation is also possible (the gradient explodes in the backward direction instead of vanishing). The tanh function given by (23) fares better than the sigmoid function because the gradient of 1 is equal to zero, but the gradient saturates rapidly at increasingly large absolute values of the argument, making it also "vulnerable" to the vanishing gradient problem. In the case of ReLU activation function given by (25), the vanishing gradient problem tends to occur less often, as long as most of these units operate within the intervals where the gradient is 1.

4.3. Speech Synthesis by Using DNN

Statistical parametric speech synthesis based on hidden Markov models has various advantages over the concatenative speech synthesis approach [35], such as the flexibility to change its voice characteristics [36], small footprint [37], and robustness to lower quality and consistency of audio material [38][39]. However, its major limitation is the quality of the synthesized speech. One major factor that degrades the quality of the synthesized speech is the accuracy of acoustic models. Conventional approaches to statistical parametric speech

synthesis typically use context-dependent HMMs clustered using decision trees to represent probability densities of speech parameters given a text. Speech parameters are generated from the probability densities to maximize their output probabilities, and then a speech waveform is reconstructed from the generated parameters. This approach is reasonably effective but has several limitations, e.g. decision trees are not efficient in modeling complex context dependencies. Firstly, they are incapable of expressing complex context dependencies such as XOR, parity or multiplex problems [40]. To represent such cases as well, decision trees would be prohibitively large. Secondly, this approach divides the input space and uses separate parameters for each region, with each region associated with a terminal node of the decision tree. This results in fragmentation of the training data and reduction of the amount of the data that can be used in clustering other contexts and estimating their distributions [41]. Having a prohibitively large tree and fragmenting training data will both lead to overfitting and degrade the quality of the synthesized speech.

However, decision trees can be replaced by an artificial neural network, which has been shown to generalize better. Figure 20 illustrates a speech synthesis framework based on a DNN. A given text to be synthesized is first converted to a sequence of input features $\{x_n^t\}$, where x_n^t denotes the n-th input feature at frame t. The input features include binary answers to questions about linguistic contexts (e.g. for phoneme identity: “is current phoneme M?”) and numeric values (e.g. the number of words in the phrase, the relative position of the current frame in the current phoneme, and duration of the current phoneme). Durations of phonemes can be obtained by using a separate DNN, or everything can be generated by single network.

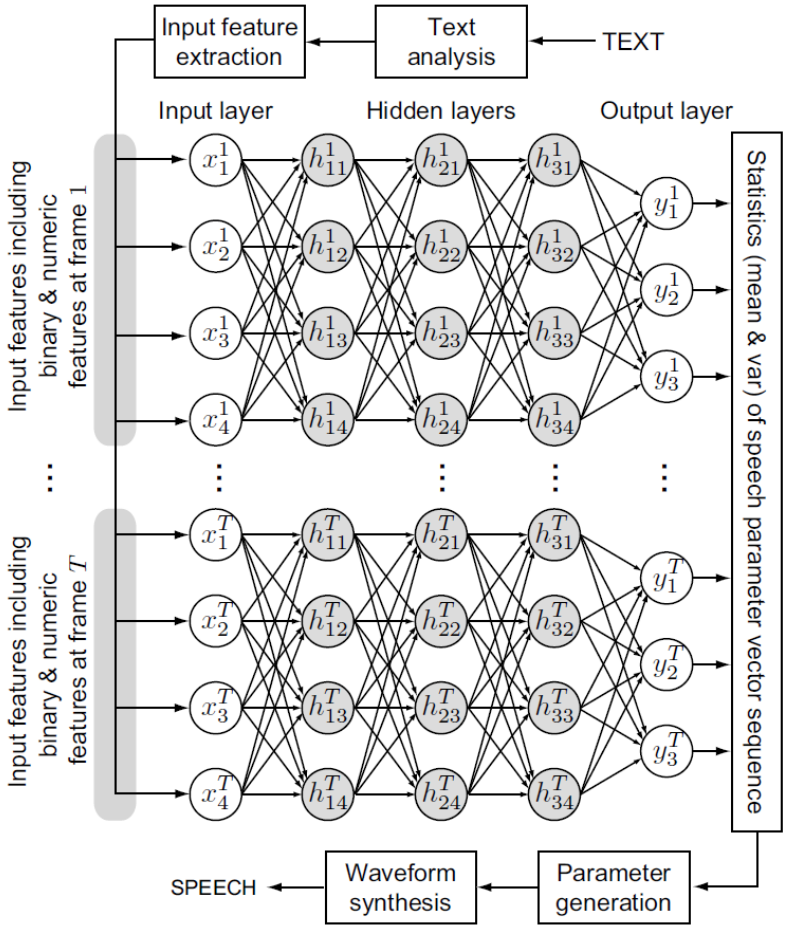


Figure 21: *DNN-based speech synhtesis framework*

Then the input features are mapped to output features $\{y_m^t\}$ by a trained DNN using forward propagation, where y_m^t denotes the m-th output feature at frame t. The output features include spectral and excitation parameters and their time derivatives (dynamic features). The weights of the DNN can be trained using pairs of input and output features extracted from training data. In the same fashion as the HMM-based approach, it is possible to generate speech parameters; by setting the predicted output features from the DNN as mean vectors and pre-computed variances of output features from all training data as covariance matrices, the MLPG algorithm can generate smooth trajectories of speech parameter features which satisfy both the statistics of static and dynamic features. Finally, a waveform synthesis module outputs a synthesized waveform given the speech parameters. Note that in this approach, the text analysis, speech parameter generation, and waveform

synthesis modules of the DNN-based system can be shared with the HMM-based one, i.e. only the mapping module from context-dependent labels to statistics needs to be replaced.

The comparison between DNNs and decision trees can be summarized as follows:

- Decision trees are inefficient to express complicated functions of input features, such as XOR, d-bit parity function, or multiplex problems [40]. To represent such cases, decision trees would be prohibitively large. On the other hand, each of these problems can be compactly represented by DNNs [42].
- Decision trees rely on a partition of the input space and using a separate set of parameters for each region associated with a terminal node. This results in a reduction of the amount of the data per region and poor generalization. Yu et al. showed that "weak" input features such as word-level emphasis in reading speech were thrown away while building decision trees [43]. DNNs provide better generalization as weights are trained from all training data. They also offer the possibility of incorporation of high-dimensional, disparate features as inputs.
- Training a DNN by back-propagation usually requires a much larger amount of computation than building decision trees. At the prediction stage, DNNs require a matrix multiplication at each layer, while decision trees just need to be traversed from their root to terminal nodes using a subset of input features.
- The induction of decision trees can produce rules that are interpretable by humans, while weights induction in a DNN is typically hard or impossible to interpret.

4.4. Merlin: An Open Source DNN TTS

In 2016. the Centre for Speech Technology Research, University of Edinburgh (CSTR), released its own open source toolkit for development of DNN-based TTS. Since most of the work in this thesis is based on that toolkit, it will be described here in more detail.

Even though there has been an explosion in the use of neural networks for speech synthesis, a truly open source toolkit was missing. Such a toolkit would underpin reproducible research and allow for more accurate cross-comparisons of competing techniques, in very much the same way that the HMM-based Speech Synthesis System

(HTS) toolkit [44] has done for HMM-based work. Like HTS, Merlin is not a complete TTS system. It provides the core acoustic modeling functions: linguistic feature vectorization, acoustic and linguistic feature normalization, neural network acoustic model training, and generation. It is written in Python, based on the Theano library, and the team at AlfaNum Company and The Faculty of Technical Sciences (AN-FTS) has adapted it to work with both CNTK [7] and TensorFlow [6] deep learning frameworks. It comes with documentation for the source code and a set of "recipes" for various system configurations.

Extraction of linguistic features is carried out at the phoneme level. Therefore, linguistic features usually contain information on phoneme identity, as well as the phonemic context, accent, level of emphasis, proximity to phrase breaks, etc. Merlin does not perform these operations on its own, but requires an external front-end module, such as Festival [45] or a custom one, such as modules that have been developed at AN-FTS for Serbian and a number of kindred South Slavic languages [46]. It is easy to interface to different front-end text processors. The front-end output must be formatted as HTS style labels with state-level alignment. The toolkit converts such labels into vectors of binary and continuous features for neural network input.

Standard Merlin TTS architecture consists of two neural networks, one for modeling phoneme durations and the other one for modeling acoustic parameters of speech. Both of them are fed with aforementioned linguistic features and are trained with appropriate outputs set as targets (Figure 21). The "duration network" models either phoneme or HMM state durations, so these values are set as targets in the training phase. It operates at the phoneme level, which means that it produces one set of outputs for each phoneme.

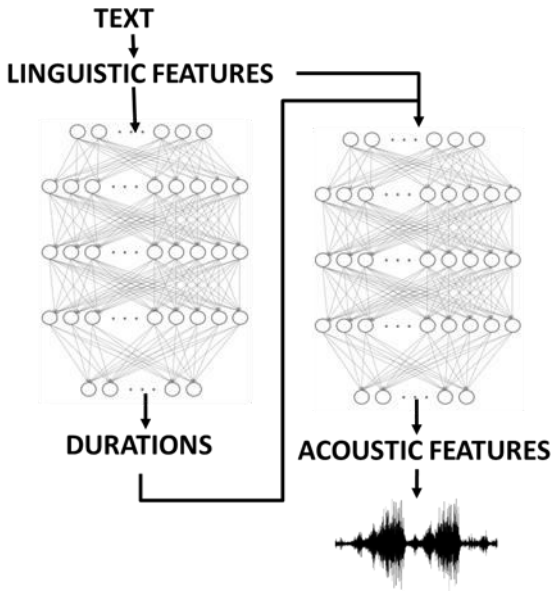


Figure 22: *Standard Merlin TTS architecture*

"Acoustic network" operates at the frame level (a frame usually being 5 ms), and produces vocoder acoustic parameters, subsequently used for speech generation. It also accepts linguistic features as inputs, but it in addition to that, it requires additional inputs specifying current frame positional information, state and phoneme durations, etc. Specifically, the subphone feature set used by Merlin includes the following features:

- fraction through state (forwards)
- fraction through state (backwards)
- length of state in frames
- state index (counting forwards)
- state index (counting backwards)
- length of phone in frames
- fraction of the phone made up by current state
- fraction through phone (backwards)
- fraction through phone (forwards).

This information is necessary for the network in order to generate smoothly changing acoustic parameters, appropriately positioned in time.

In order to get durations at the state level, as targets during the training, initial alignment has to be performed. The frame alignment and state information is obtained from forced alignment using HMM-based system with several emitting states per phone. This procedure is usually done automatically from scratch (no manual alignment required), by using monophone models with up to 8 mixture components, and several rounds of Baum-Welch algorithm [47], followed by the application of the HTS alignment tool (HVite). The use of monophones for this purpose is known to be suboptimal, but it performs well when there is a large amount of training material available. In cases of scarce training material for a new target speaker, some changes had to be introduced to this algorithm, which will be discussed later.

Before training a neural network, it is important to normalize features. The toolkit supports two normalization methods: min-max, and mean-variance. The min-max normalization will normalize features to the range of [0.01 0.99], while the mean-variance normalization will normalize features to zero mean and unit variance. By default, the linguistic features undergo min-max normalization, while output duration and acoustic features undergo mean-variance normalization.

Initially, the waveform generation module supported two vocoders: STRAIGHT [12] and WORLD [13], but the toolkit is easily extensible to other vocoders. For example, the team at AN-FTS has recently added support for the WaveRNN vocoder [48]. Standard parameters used for speech encoding are:

- Spectral envelope representation. Usually by (40+) MGCs or (~80) filter banks.
- Band aperiodicity, with 3-5 coarse aperiodicity parameters in the case or WORLD vocoder, or up to 30 band aperiodicity parameters in the case of STRAIGHT vocoder.
- Pitch and voiced/unvoiced (VUV) estimation.

As already mentioned, these parameters are usually estimated on a frame level, one frame being ~5 ms. In addition to these, static parameters, delta and delta-delta parameters are calculated and included in the training procedure. It has been shown that they provide greater stability in the synthesized voice after MLPG procedure is applied during synthesis.

Merlin includes implementations of several popular neural network models, each of which comes with an example "recipe" to demonstrate its use. Similarly as in HTS, separate models are used to predict phoneme state durations and vocoder parameters, referred to as acoustic features. These models are trained separately and can have different architectures. Dense, feedforward neural network is the simplest type of network. It takes linguistic features as input and predicts the output through several hidden layers. The input is used to predict the output via several layers of hidden units, each of which performs a nonlinear function. In the toolkit, sigmoid and hyperbolic tangent activation functions are supported for the hidden layers. In a feedforward network, linguistic features are mapped to vocoder parameters frame by frame without considering the sequential nature of speech. In contrast, RNNs are well suited for sequence-to-sequence mapping. The use of LSTM [28] units is a popular way to realize an RNN. In a unidirectional RNN, only contextual information from past time instances is taken into account, while bidirectional RNNs (or BLSTMs) can learn from information propagated both forwards and backwards in time. Other variants of neural networks are also implemented, such as gated recurrent units (GRUs) [49], simplified LSTM [50], as well as other variants on LSTMs and GRUs described in [50]. All these basic units can be assembled together to create a new architecture by simply changing a configuration file, for example:

[TANH, TANH, TANH, TANH]

which describes a network consisting of 4 feedforward layers, with tanh activation functions. Similarly, a hybrid bidirectional LSTM-based RNN can be specified as:

[TANH, TANH, TANH, BLSTM]

in the configuration file.

There are other hyper-parameters available in Merlin, aimed at tuning the training process:

- Learning rate, momentum, and scheduling of these values over the course of training.
- The length of the "warm-up period", in which momentum and learning rate are somewhat lower.
- Mini-batch size and the number of epochs (for duration and acoustic networks separately).

- Early stopping criteria.
- L1 and L2 regularization.

In the stage of synthesis, state-level durations predicted from the first network are used to extract additional features for the second network, which predicts acoustic features required by vocoder to produce waveforms. It was found that better results are achieved when dynamic acoustic features are used along with the static ones. For this reason, first and second derivatives of acoustic features are also used as targets during the training of the second network. In the synthesis stage, after those are predicted, they are only used by MLPG algorithm [21] in order to slightly correct static acoustic features trajectories. After this procedure, recalculated static features are propagated to the vocoder. One should note that, in contrast to HMM model, we do not have states here, and consequently we do not have per-state variances of the features either. Instead, global variances of all the features (static, delta and delta-delta) are used.

4.4.1. **Improvements in Merlin Made by AN-FTS Team**

As already stated, Merlin was initially developed with a support for Theano [51], which is a Python library that allows efficient definition, optimization and evaluation of mathematical expressions involving multi-dimensional arrays. Although it was very powerful at the time of its introduction, and despite the progress made in recent years, there remain some limitations or shortcomings in Theano. Today it has been largely surpassed by some newer frameworks, such as TensorFlow [6], CNTK [7] and PyTorch [52]. The AN-FTS team made a significant effort to integrate Merlin with CNTK and TensorFlow frameworks and thus make it more efficient and flexible. It also allowed the development of the architecture described in this thesis, which will be described later.

Merlin performs phoneme and state alignment of the audio material by relying on monophone HMM models. Initial improvement to this approach was described in [53], where authors proposed alignment method based on the use of full-context models obtained during the training of HMM TTS system and the HMM synthesizer itself. The first step in this procedure is synthesizing the sentence which is to be aligned using the HMM synthesizer. It also enables assigning each synthesized frame to a corresponding state. The second step is

feature extraction from original audio files by using the same speech representation as in the training of the HMM system. The final step is to align generated and original frames, which is performed by using the Viterbi search. This method outperformed Merlin's default version, but it also struggled when the amount of the data for a new speaker was extremely small. In those cases, we first had to perform MLLR adaptation of HTS models to the new speaker, by preserving the already established set of states and context dependency tree. This led not only to better alignment but also accelerated the process several times.

4.5. Lexical Features Used in Proposed TTS Models

In Section 2.2 we already mentioned front-end as a part of TTS system which accepts raw text and coverts it into a stream of data which is sufficient for back-end to create natural sounding speech. In order to achieve that, front-end usually has to perform the following tasks:

Text cleaning: Get rid of items (HTML mark-up, etc.) that are not to be synthesized. It's often language-independent.

Text normalization: Transforms items such as dates, time, numbers, currency, phone numbers, addresses, and abbreviations into normal orthographic form.

Examples:

- *Dr. King Dr.* becomes *Doctor King Drive*
- *1 oz.* becomes *one ounce*
- *2 oz.* becomes *two ounces*

Grapheme-to-Phoneme conversion (phonetization): Transforms a (normalized) orthographic string into the "phones" of the language:

The brown fox -> DH AX B R AW N F AO K S

Syllabification and lexical stress prediction: Divides a word's phonetic representation into syllables, and marks lexical stress:

Speech Synthesis -> S P IY(1) CH || S IH(1) N | TH AX(0) | S IX(0) S

Part-of-Speech (POS) tagging:

She came to record the record ->

She(PRN) came(VB) to record(VB) the(DET) record(NOUN)

Syntactical analysis:

[NP The brown fox] [VP jumped] [PP over] [NP the lazy dog.]

Semantic analysis such as named-entity recognition (is it a person? a place? An organization? a quantity? etc.):

Jim bought 300 shares of Acme Corp. in 2006. -> Jim(PERSON) bought 300(QUANTITY) shares of Acme Corp.(ORGANIZATION) in 2006(DATE)

Generating prosodic tags: These provide the most explicit information about the prosody on syllable, word and sentence level. Prosodic tags can describe emphasis level of certain syllables, pitch movement, word and phrase breaks, elongations and more. If prosodic tags are provided by front-end, it usually makes POS, syntactic and semantic information obsolete for back-end. Actually, front-end usually relies on some of these in order to predict prosodic tags.

Lexical features used by our back-end consist of:

- Phone identity and identity of neighboring phones (±1 and ±2 context).
- Position of lexical stress.
- Phone position related to syllable/foot/word/phrase boundary.
- Word position related to phrase boundary.
- Number of phones in syllable/foot/word.
- Number of words in phrase/utterance.
- Language specific prosodic tags.

For Serbian and kindred languages the following prosodic tags are used:

- Vowel accent. For stressed vowels there are four accent types. Unstressed vowels can have post-tonic length or not.
- Four types of phrase breaks: weak, medium, strong and sentence end break.
- Presence of positive or negative emphasis (on a word level).
- Phone position related to all these tags.

For English language the following is used:

- A subset of most relevant standard ToBI tags, including pitch accents, break and boundary tone indices, as described in [54].
- Emphasis (E+) tag, which marks extra emphasized words.
- Compressed pitch (CF0) tag, which marks deemphasized words.
- Phone position related to all these tags.

Lexical features are provided in the form of binary questions (actually, answers to those questions), so they could be used by classification trees used in HMM-TTS. For example, answer to the question "is current phoneme P" would be a single (binary) lexical feature. For the same reason, some of the questions are combined into so-called complex questions in the form of logical expressions, if that combination seemed important for classification. For example, question

{name='is_silence_R', def='is_R AND is_left_phoneme_silence'}

returns *true* only if current phoneme is "R" and it is the first phoneme after silence. Phoneme "R" exhibits different acoustic features in this case, which is why it was singled out. Of course, simple questions could also resolve this and reach the same conclusion, but it would require more branching and more final nodes in classification tree, which would in turn require more data to be available for the process to be successful. Note that each question is given a name, which provides the possibility of creating even more complex questions.

Complex questions were not used in the case od DNN TTS, because DNN transforms lexical to acoustic features in a different way (see Section 4.3), and does not really need this type of manual augmentation of the features. DNN could also accept non-binary questions

(e.g. “number of phonemes from the eprevious break”), but in order to keep the same tools in use for both HMM and DNN case, the same features were used for DNN.

It is also noteworthy that our lexical features provide quite wide contextual information for every phoneme in the utterance, which is necessary for HMM, but also helpful for DNN model. If only local information was provided (only for the current phone), HMM would generate quite poor output, because no context would be taken into account. DNN could overcome this problem by utilizing convolutional or bidirectional recurrent architecture, but because of the presence of wide context, feed-forward network also yields decent results.

4.6. Comparison of HMM and DNN Based Speech Synthesis

As mentioned earlier, for a long time HMM-based synthesis represented the state of the art in parametric speech synthesis. DNN approach seems to promise a change in this paradigm, owing to the improvements in mapping lexical to acoustic features.

In order to compare parametric approaches for speech synthesis in terms of overall quality a set of experiments was performed in [46]. HMM system was based on parameters extracted by WORLD vocoder – 40 MGCs representing spectral envelope, logarithm of fundamental frequency and 2 band aperiodicity parameters were used (see Section 2.2.2). Five-state, left-to-right, no-skip hidden semi-Markov models (HSMMs) were used. The logarithm of f_0 and band aperiodicity parameters were modeled using multi-space probability distribution (MSD). The number of lexical questions used for context-clustering was 617 (see Section 4.5) and default values of 1 for parameters controlling tree size were used (Minimum description length (MDL) criterion was used). For DNN system, the same vocoder, acoustic parameters and database were used. For the DNN architecture the one with best objective measures was chosen – 4 tangent hyperbolic feed-forward hidden layers with 512 units per layer. Both systems were trained on the same database consisting of 3 hours of speech (the best DNN objective measures are obtained with this architecture and database). In order to objectively compare HMM and DNN synthesis, the trajectories of generated acoustic parameters were compared with the trajectories of parameters extracted from the original recordings. The trajectories of several lowest MGC coefficients were found to be almost the

same for both DNN and HMM, and to follow the original trajectory almost perfectly. Significant differences between HMM and DNN trajectories, as well as differences from the original utterance, start to occur on the 6th MGC coefficient (Figure 22) and differ from the original one to a greater extent. Nonetheless, it can be seen that the DNN trajectory follows the original one much better than the HMM trajectory, and that there are no significant deviations. For higher coefficients, the differences between HMM and DNN are more emphasized and deviations from the original one increase. In other words, neither DNN nor HMM are able to accurately predict spectral details. All three trajectories (HMM, DNN and original) of fundamental frequency are more similar among themselves than the trajectories of MGC coefficients (Figure 23). The DNN trajectory is, again, a better match to the original than the HMM trajectory.

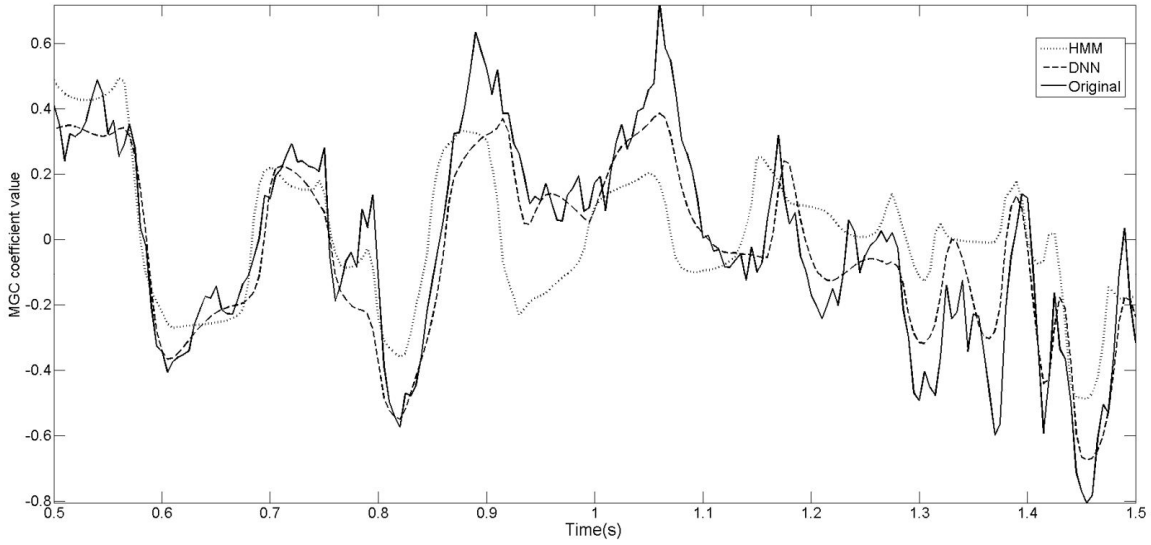


Figure 23: *Trajectory of the 6th MGC coefficient*

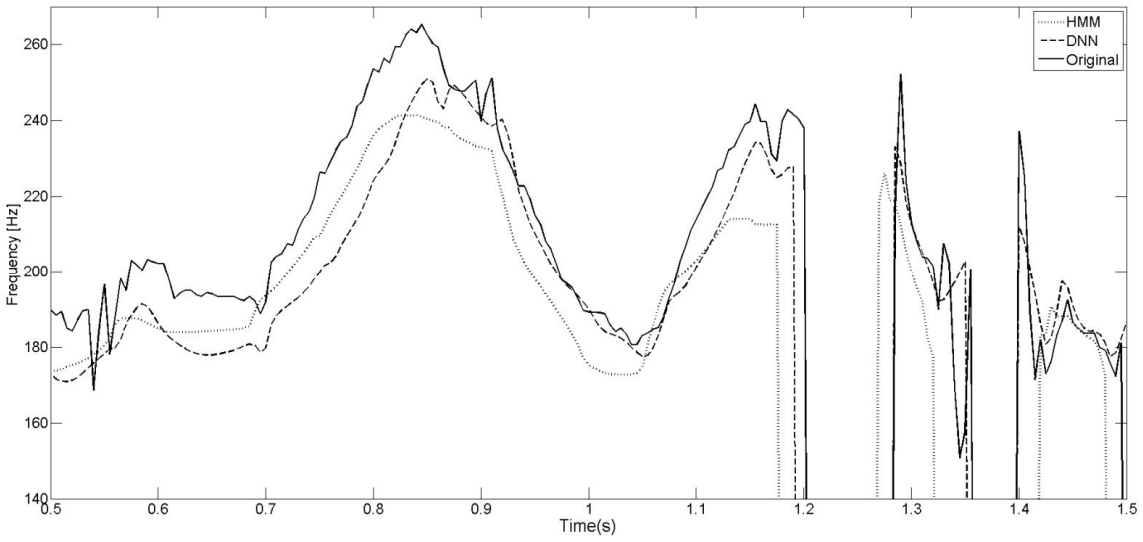


Figure 24: *Trajectory of the fundamental frequency*

Furthermore, in Figure 23, it can be clearly seen that DNN is a better predictor of whether a frame should be voiced or not.

Subjective evaluation of the quality of synthesized speech for HMM and DNN approach was carried out by listening tests. Participants were 40 students, native speakers, without expert knowledge of speech technology. In each test there were 3 audio files, each containing the same 4 unrelated utterances. The first file is generated by the HMM model, the second by the DNN model and the last one represents original recordings (natural speech). More details about the test conditions can be found in [46]. Figure 24 presents average grades for two main features of synthesized speech – intelligibility and naturalness. DNN was found to perform better than HMM as regards both intelligibility and naturalness by almost half of a grade, while it lags behind original recordings for just 0.25 as regards intelligibility and 0.43 as regards naturalness. The overall average grade, calculated by averaging the grades for naturalness, intelligibility and the overall impression, for original recording is 4.7, for DNN is 4.3 and for HMM is 3.8.

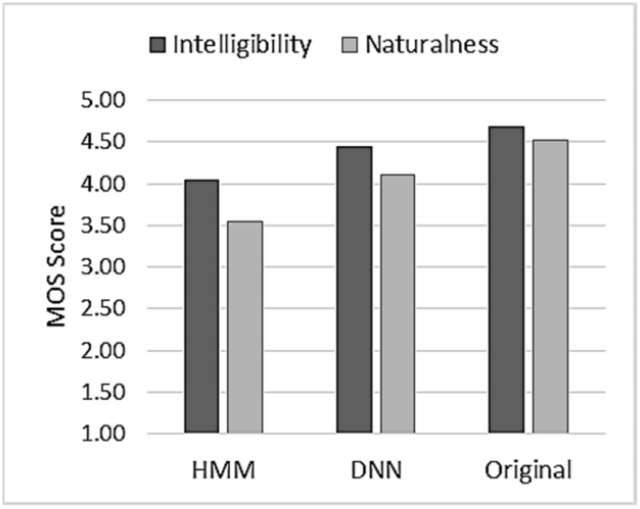


Figure 25: *Comparison of intelligibility and naturalness of synthesized and natural speech, when using HMM and DNN approaches*

4.7. Speaker Adaptation in DNN Speech Synthesis

To harness quality improvements achieved over HMM based speech synthesis, a variety of speaker adaptation techniques have been proposed for DNN-based acoustic models. Wu et al. [55] proposed speaker adaptation using i-vectors as input, or by adapting hidden unit contributions (LHUC [56]), or by applying output transforms defined by GMMs, or combinations of these. Fan, et al. [57] assumed that the output layer in the DNN captures most speaker differences, and considered estimating speaker-dependent output layers using multi-speaker data, while keeping the hidden network layers shared across all speakers. This also allowed the model to be adapted to new speakers by only updating the regression layer [57]. However, their experiments only used four different speakers, with a relatively large amount of data (one hour) from each. There have also been attempts to enable control of DNNs similar to multiple regression HMMs. In [58], two-dimensional per-sentence control-vector inputs to a DNN synthesizer were learned in an unsupervised fashion from a corpus of expressive speech. It was found that one direction in the (unlabeled) control-vector space had a consistent and interpretable influence on the generated speech, but the orthogonal direction did not. In [59], authors trained a system on 135 speakers and used “discriminant condition

codes" to map initial one-hot vector to speaker space. In the adaptation phase they used back propagation algorithm to update the speaker codes and minimize the mean square prediction error over a small amount of data uttered by the target speaker. They obtained promising results by using only a small amount of adaptation data. A DNN architecture with additional speaker-dependent inputs was proposed in [60], and this approach was further extended by supplementing the input information by speaker gender and age [59]. To enable the network to reproduce the voice of a particular speaker in a style that is absent from the training corpus (which is referred to as emotion or style transplantation), the research presented in [61] proposed a network architecture which explicitly separates speaker and speech style contributions, while the one presented in [62] built on the multi-speaker DNN with shared hidden layers proposed in [57], by extending it with a single style-dependent input and introducing an additional bottleneck layer. Other lines of research, such as the one presented in [63], focused on the development of methods for adaptation of a multi-style single-speaker DNN to a new speaker's voice. In one way or another, all these approaches address the practical impossibility of recording and processing a new training speech corpus for each new speaker/style combination for which the need may arise.

In this thesis we present two methods for efficient creation of new TTS voices, based on relatively small amount of adaptation data. In Section 5 we describe a method which initially trains DNN-based TTS on a relatively large amount of training material (3+ hours) and uses that model as a starting point for adaptation. This means that new model is not trained on a randomly initialized network (weights and biases), but on an already pretrained one, which results in much better performance (higher quality of synthesized speech).

Second approach, described in Section 6, proposes creating initial multi-speaker model and corresponding speaker embedding space. During adaptation, two phases are performed. In the first phase optimal embedding for the new speaker is found, with the idea of generating speech that already resembles target speaker, so only minimal changes to the DNN are required in the second phase. In the second phase, the new embedding is fixed, and the rest of the DNN is adapted in the same way as in the first approach. This two-phase approach yielded even better results and can generate voices with the amount of material as small as 30 seconds.

5. Adaptation from Source to Target Speaker

It is very important to reduce the quantity of target speaker data needed for producing high quality synthetic speech in target speaker's voice. In this section, a simple but very efficient method for creating a new DNN-based TTS voice with a small amount of data, developed and published in [64], is presented. The idea is to use data that correspond to target speaker and to retrain DNN already trained in TTS task on source speaker data. Thus, we start re-training with initial values of network parameters of pretrained network, instead of randomly initialized. This approach reduces the quantity of target speaker data needed for producing high quality synthetic speech in target speaker's voice.

5.1. DNN adaptation

The method deals with the generation of natural sounding speech signals from text that has already been linguistically processed. It is thus assumed that all the necessary phonetic and prosodic information is known at the time of synthesis, and the problems of natural linguistic processing required to recover this information from text are abstracted away. Besides lexical features related to prosody (such as lexical stress), we also use explicit features related to specific choices of prosodic events (pitch accents, phrase breaks and the corresponding phrase accents and boundary tones, see Section 4.5) that the speaker makes when forming the prosodic plan of the utterance.

Speech is obtained by linguistically pre-processing the text and then using the improved Merlin toolkit (Section 4.4.1) for the final wave form generation by using the WORLD vocoder [13]. In the training phase, each frame of speech is parameterized into 40 mel-generalized cepstral coefficients, logarithm of f_0, band aperiodicity parameter, as well as a binary feature which indicates frame voicing (see Section 2.2.2), where the 5 ms frame shift is used. At synthesis time, the values of these parameters are predicted for new, usually previously unseen phonetic and prosodic contexts. After that, predicted features for certain frame are converted to speech by using vocoder. As network is not constrained to produce

smooth trajectories of output features, for modeling of speech dynamics, for all features except voicing, corresponding dynamic features (first and second derivatives) are additionally supplied. Thus, there are altogether 127 acoustic features for each frame so that the static output features are smoothed taking into account the dynamic output features using MLPG algorithm (see Section 3). The prediction of acoustic features from which speech is generated is divided into two stages, and performed by two DNNs trained simultaneously. First network predicts the durations of phonetic segments, i.e. states (5 states per phoneme were used) from linguistic features extracted from text. Second network uses the information related to the durations of each phonetic state (in frames) obtained as the prediction result from the first network, in order to predict acoustic features. Input for the first network contains 554 binary linguistic features (see Section 4.5). Durations obtained from alignment procedure are used as target features, in order to train the first network, while for the second network, the same input as for the first one is used, with additional 9 features specifying state and phone durations as well as frame position inside current state (see Section 4.4). The output features of the second network are previously mentioned 127 acoustic features, which are extracted by the vocoder from the original recordings and used as target in the training.

Both networks have 4 hidden layers and 1024 units per layer with the activation function used for the input and the first 3 hidden layers set to be hyperbolic tangent given by (23). The last hidden layer uses LSTM units (see Section 4.4), while the output layer is linear (no activation function). Additional feature normalization is performed for input (normalized to the unit interval), as well as output features (normalized so as to have zero mean and unit variance). The objective function used is mean square error.

5.2. Proposed TTS adaptation procedure

Common approaches to training a DNN based TTS system usually start from a DNN model with random values for weights and biases. Starting from such a model, several hours of single speaker speech material is typically required to train the network to produce intelligible and relatively natural sounding speech, which will resemble the voice of the original speaker to significant extent. However, the preparation of annotated speech database

of several hours is an expensive and extremely time consuming process. The main idea of this approach is to use an existing model trained on a large database of source speaker as the initial model for adapting to the target speaker. It thus enables rapid and less expensive TTS adaption, since it does not require the existence of an average speaker model as in conventional speaker adaptation methods, and at the same time it requires far less training data than the amount needed to build a DNN-based TTS voice from scratch. The influence of the choice of the starting model on the proposed adaptation method is also a matter for investigation.

The training of DNN used in speech synthesis requires the initial state alignment since state-level alignments yield much better results in comparison to phone-level alignment [65]. In Merlin toolkit, it is achieved by forced alignment using the monophone models trained on the same database on which DNN is trained and it has been shown to be outperformed by the method described in Section 4.4.1. The accuracy of this procedure obviously decreases when the amount of training material is small.

5.3. Experimental Results

In this section, we compare the quality of the proposed model against the baseline model in the task of the new TTS voice creation. The model is evaluated on a set of utterances (excluded from the training process) that are synthesized on the basis of phonetic and prosodic information taken from utterances actually pronounced by target speakers. In all presented experiments, available utterances were divided into training, validation and test sets. For all experiments, the same test sets were used, consisting of 5 or 10 utterances. In each experiment, 10% of utterances were randomly chosen to be used for validation, while the rest was used for training.

For objective evaluation of the results, mean squared error for MGCs (MCD) and band aperiodicities error were used, both given in dB, correlation between predicted and original f_0 and durations of phones, the error of frame voicing prediction and root mean squared error for f_0 (see Section 2.4). For subjective evaluation, two MUSHRA tests [66] were conducted. Both of them were performed by 22 subjects in a controlled environment and with good

quality headphones. Each subject evaluated a certain number of test utterances by comparing them with the reference one (the original recording), where each time one of the test sentences was identical to the reference. The utterances were evaluated in terms of overall quality (intelligibility and naturalness). Each recording was given a grade from 0 to 100 (with one limitation – one of 5 sentences had to be given grade 100). Average grades were calculated and t-test was used in order to check for a statistically significant mean value differences.

5.3.1. **Alignment performances**

For a sufficient amount of data, standard forced alignment based on monophones achieves satisfactory accuracy. Nevertheless, in the situations when significantly less data is available the proposed method (see Section 4.4.1) achieves better results in predicting alignments, which is presented in Figure 25. The figure represents the percentage of the phonemes whose boundary deviations are below a certain threshold compared to manually set boundaries. Also, the influence of alignment method on the objective measures of corresponding TTS model is presented in Table 1, where it can be seen that the proposed alignment achieved almost the same results as training with conventional forced-alignment when the target speaker database contains 10 or 15 minutes of speech, but notably better results when it contains just 3 or 5 minutes of material. Thus, in all experiments the initial alignment was performed with proposed method, while training procedures starting from randomly initialized models used the conventional method.

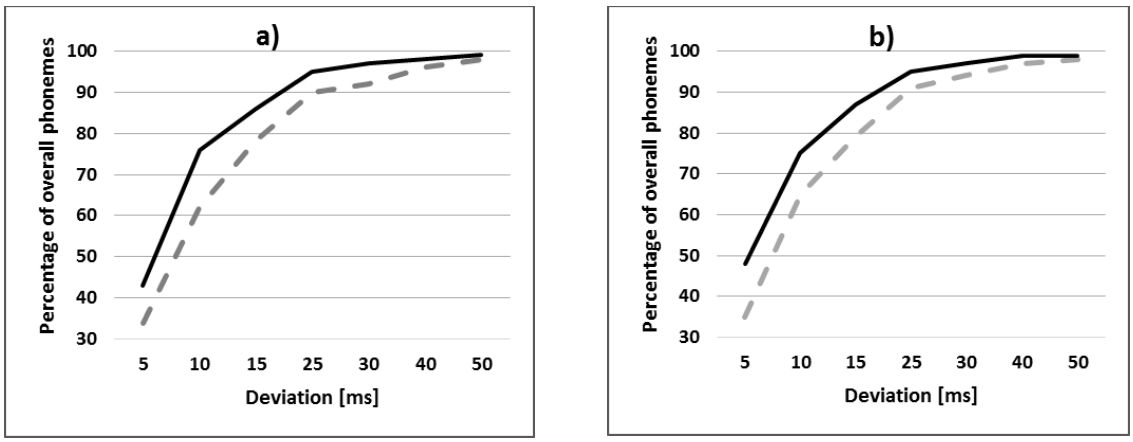


Figure 26: *Comparison of baseline monophone alignment (dashed grey line) and proposed alignment (solid black line) for 3 min (a) and 5 min (b) of speech.*

Db	Alignment	MCD	BAP	RMSE F0	VUV	RMSE DUR
3 min	Monophones	5.39	0.18	23.58	8.53	5.49
	Proposed	5.25	0.17	22.58	7.66	4.90
5 min	Monophones	5.18	0.18	23.06	7.80	5.16
	Proposed	5.12	0.17	22.28	7.63	4.87
10 min	Monophones	5.00	0.17	22.62	7.43	5.18
	Proposed	5.02	0.17	22.00	7.46	4.95
15 min	Monophones	4.90	0.16	21.50	7.33	4.81
	Proposed	4.94	0.16	21.17	7.33	4.92

Table 1: *TTS objective measures comparison, depending on the alignment method and the amount of material used*

5.3.2. **First set of experiments**

Here, we tend to verify that one could use significantly smaller amount of target speaker data if starting from model trained on source speaker data, than if starting from model utilizing randomly initialized weights. Namely, baseline models where randomly initialized using 5, 10, 15, 30, 60 and 180 minutes of male speaker's data, respectively. Proposed models where built starting from a model previously trained on 3h of female speaker's data, then adapting it to a male speaker using 3, 5, 10 and 15 minutes of target data, respectively. As can be seen in Figure 26, all objective measures, with the exception of VUV, show that

when starting from the model already trained on source speaker data, 15 minutes of target speaker data is sufficient to reach the quality obtained by starting from a randomly initialized model and training it with 30 minutes of data (see e.g. MCD in Figure 26 a). Also, starting from the trained model, 5 minutes of speech is sufficient to reach or surpass the quality obtained when training a randomly initialized model on 15 minutes.

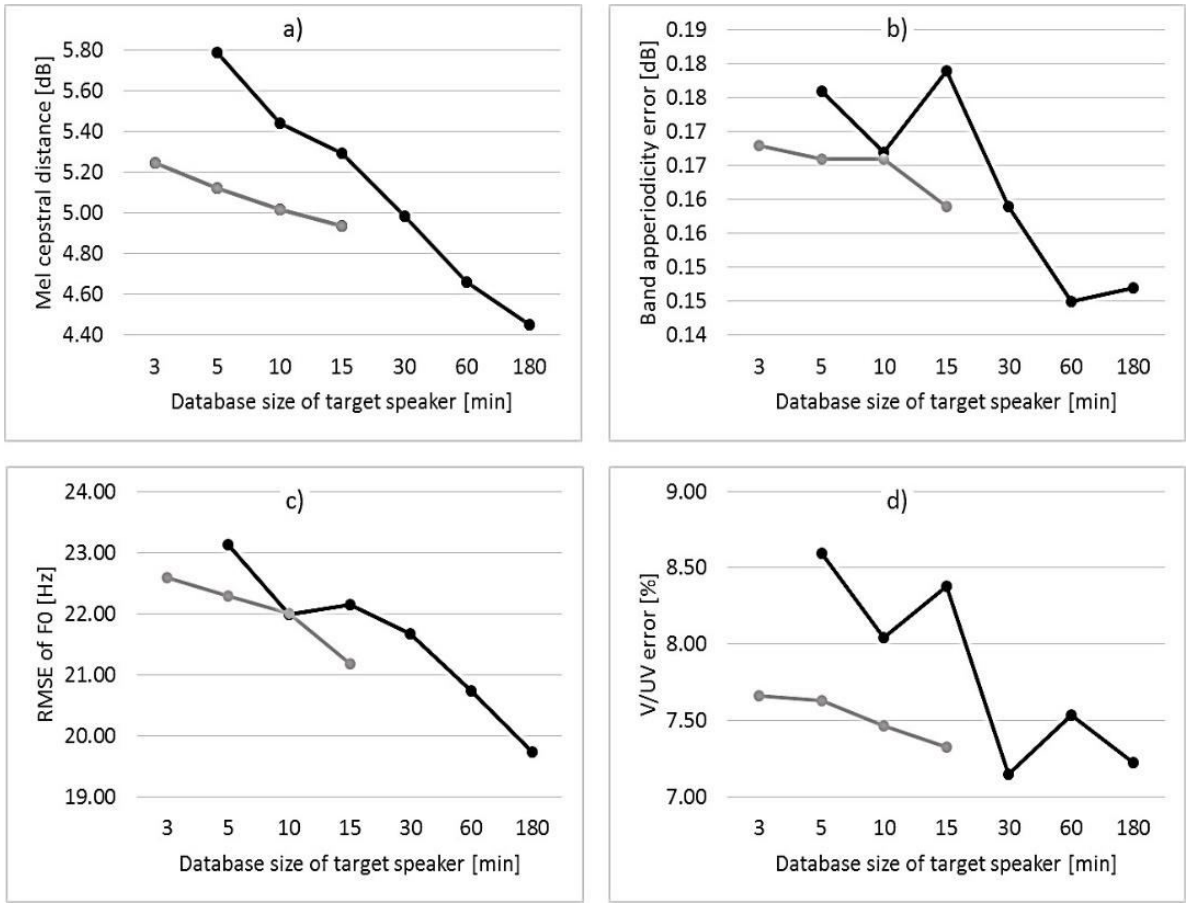


Figure 27: *Objective measures for MCD (a), BAP (b), RMSE for f0 (c) and VUV (d). Black line represents the results when the initial model is randomly initialized, while the gray one represents the results when the initial model is trained on 3 h of female speaker data. The size of the male speaker's database used for training/adaptation spans from 3 to 180 min.*

Although 50% less data being needed to achieve the same quality may be considered a good result, it is unsatisfactory that 15 minutes of target data is still not enough to convert an already trained model into a model able to produce speech of a quality comparable to that of a model trained on a 3h database after being randomly initialized.

Since the objective measures do not fully reflect subjective perception, additional listening tests were performed. Those included 10 sentences where the original recording was used together with 4 utterances synthesized by 4 different synthesizers listed in Table 2, randomly shuffled. The synthesizers presented in Table 2 represent the subset of all systems shown in Figure 26, while the results of listening tests are presented in Figure 27.

Synthesizer	**Starting model**	**Database**	**Recording**
1.1	Random	Male 3h	Studio
1.2	Random	Male 1h	Studio
1.3	Female (3h)	Male 10min	Studio
1.4	Female (3h)	Male 3min	Studio

Table 2: *Synthesizers used in the first subjective test, with information about starting model, size of training database as well as speaker gender and database recording conditions*

It can be seen that 10 minutes of target speaker starting from the model trained on source speaker sounds closely to 1 hour of target speaker starting from a randomly initialized model. Their average results are close to the results of 3 hours of target speaker (with the significance t-test α=0.05). Although it may seem that the average grade of synthesizer 1.4 is also close to the others, the t-test shows a statistically significant difference. It can thus be concluded that 3 minutes of target speaker provides satisfactory results, but the synthesized speech still cannot be expected to sound like speech synthesized by a model trained on a relatively big database.

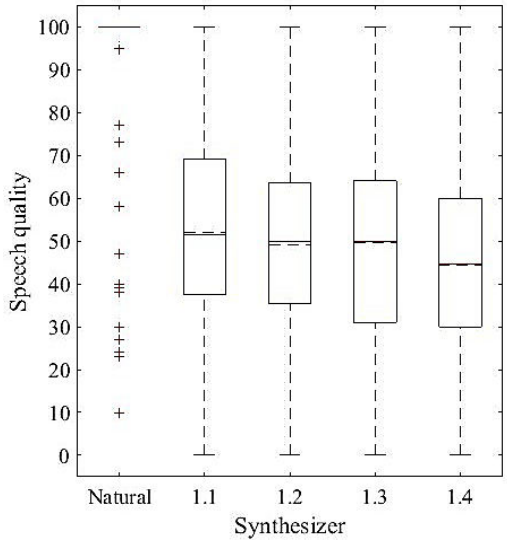


Figure 28: *Grades for synthesizers 1.1 - 1.4 and for natural speech. Boxes are limited with 25th and 75th percentile values, solid lines in the boxes present the median values, while the dashed lines present the mean values.*

5.3.3. **Second Set of Experiments**

In the second set of experiments we examine if the initial model has any significant influence on overall result of the adaptation procedure. Inter and intra-gender adaptations were performed, starting from model trained on 3h database in all cases, and adaptation was done with 3, 5 or 10 minutes of target speaker's data. In Figure 28, objective measures for this set of experiments are given. It could be concluded that there is no difference regarding the male or female starting models in cases of male or female target speakers.

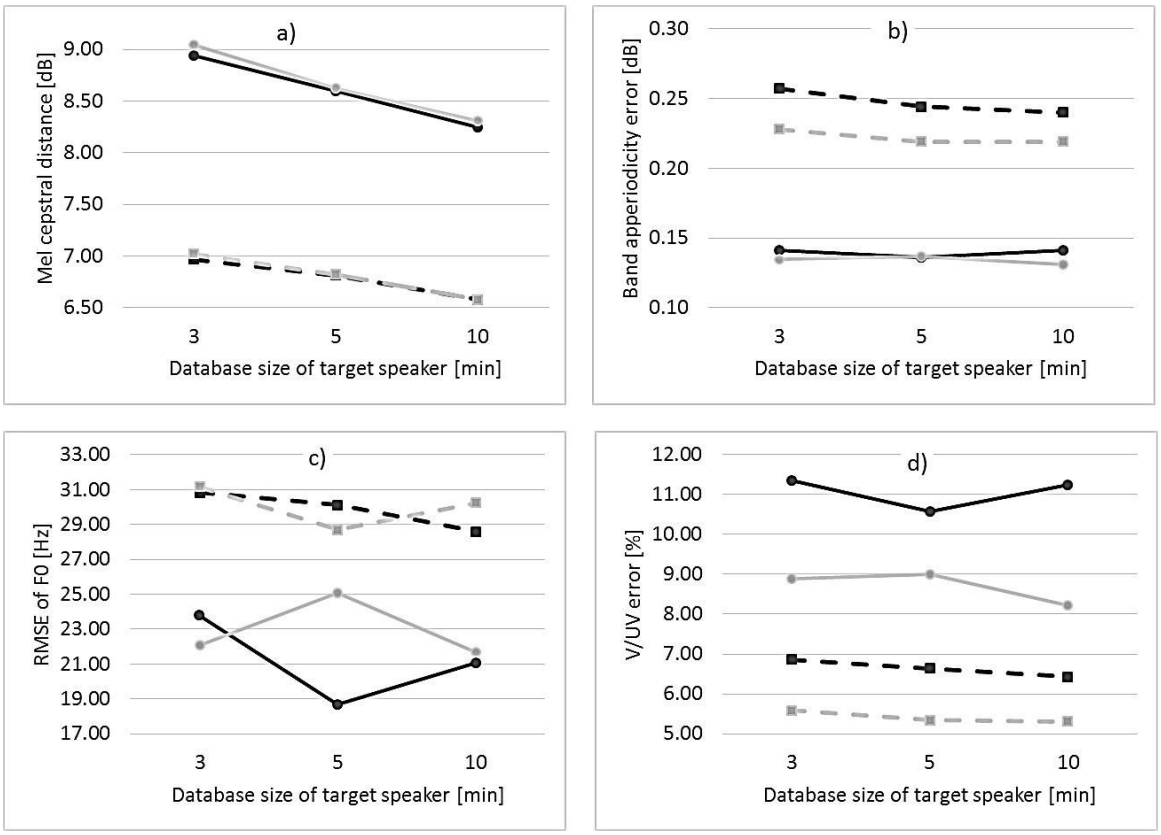


Figure 29: *Objective measures for MCD (a), BAP (b), RMSE for f0 (c) and VUV (d). Black line represents cases when the initial model is male, while the gray line represents cases when the initial model is female. Solid lines represent cases when the target speaker is male, while dashed lines represent cases when the target speaker is female.*

The inter-gender vs. intra-gender adaptations comparison in the form of subjective listening tests were also performed, presented in Table 3. The listening test included 10 sentences, half of them female and half male speakers. For each of the sentences, the original recordings and four more produced by synthesizers listed in Table 3 were used, randomly shuffled.

Synthesizer	**Starting model**	**Database**	**Recording**
2.1	Female (3h)	3 min	From YouTube
2.2	Male (3h)	3 min	From YouTube
2.3	Male (3h)	10 min	From YouTube
2.4	Female (3h)	10 min	From YouTube

Table 3: *Synthesizers used in the second subjective test, with information about starting model, size of target speaker database used for adaptation and database recording conditions*

The results are presented in Figure 29. It can be seen that when the target speaker was female (Figure 29a), adaptation with 10 minutes of target speaker's data yields better results if a male initial model was used instead of female. However, if only 3 minutes of adaptation data are used, the results for both male and female initial models are almost the same. The t-test shows that with either 10 or 3 minutes of adaptation data, there is no statistically significant difference between utterances synthesized with models which were originally male or female. On the other hand, when the target speaker was male (Figure 29b), adaptation with just 3 minutes of target speaker's data, starting from a male initial model achieves better results than adaptation with 10 minutes of data, starting from a female initial model. The t-tests showed that there is a statistically significant difference when comparing utterances synthesized by models which were initially male with those which were initially female. It turned out that the model which was initially male, was more appropriate.

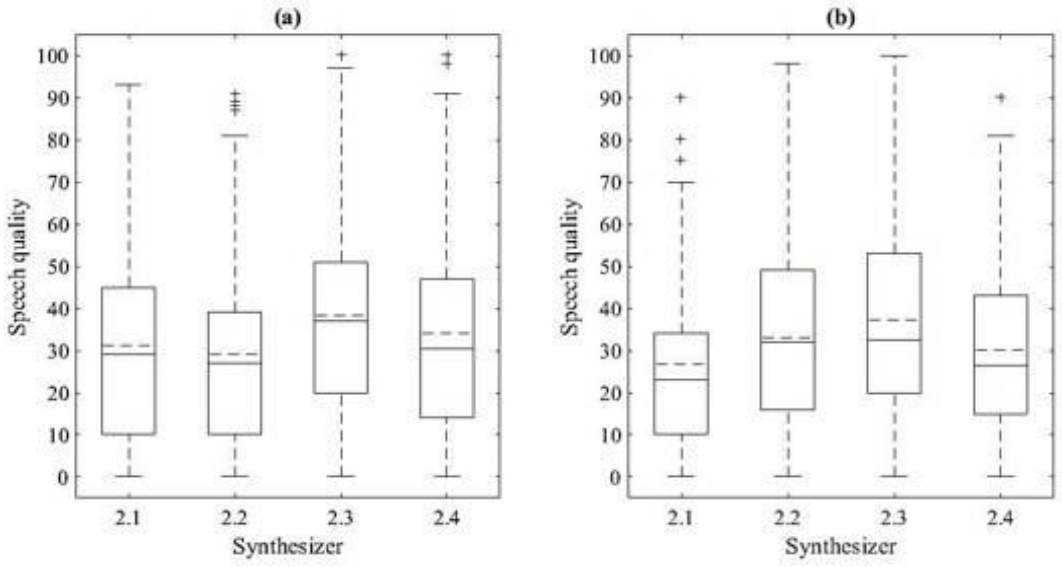


Figure 30: *Grades for synthesizers 2.1, 2.2, 2.3 and 2.4 for (a) the female target speaker and (b) the male target speaker. Boxes are limited with 25th and 75th percentile values. Solid lines in the boxes present the median values, while the dashed lines present the mean values.*

It is noteworthy that some listeners pointed out that overall synthesis quality was lower in the second set of experiments. This is expected since this database was not recorded in a studio environment. The actual influence of target database quality on overall quality of synthesized speech is a matter for further research. We can conclude that using the limited

resources available, the initial model has some, though not significant influence on the adaptation. Thus, further research could focus on the case when an average speaker model is used as the starting point, as well as the possibility of restricting the adaptation procedure to higher NN layers.

6. Two-Step Approach to Speaker Adaptation

This approach proposes a DNN architecture and a two-step adaptation procedure aimed at obtaining speaker/style-dependent speech synthesis based on very small quantities of training data by the target speaker and in the target speech style, which produces synthesized speech of very good quality. All the procedures and models are based on Merlin toolkit, which has been significantly upgraded by AN-FTS team as described in 4.4.1.

6.1. Model Description

The model is based on a cascade of two independent neural networks – one predicting phonetic segment durations, and the other predicting acoustic feature vectors for each frame. The principal input to both networks is the vector of 577 linguistic features extracted from text, related to the current phone. In the synthesis stage, the output of the duration model is used as supplementary input of the acoustic model, augmented with the information on the duration of particular HMM states of each phone, which is obtained in the training phase from HMM models through the alignment procedure described in [53]. In all experiments each of the two networks has 4 hidden layers of size 1024, where the first three are feed-forward dense layers, while the fourth one is composed of LSTM units. All of them use tangent hyperbolic activation function. Stochastic gradient descent was used as optimizer in back propagation algorithm, using one utterance as a batch. In other words, back propagation occurs after the networks have seen one utterance, regardless of the number of phones (in the case of the first network) or frames (in the case of the second one).

Initially, multi-speaker multi-style (MSMS) model was trained on a number of speakers in order to get a good baseline model for adaptation and also to create speaker embedding, similarly as in [60]. Embedding is a powerful deep learning technique based on mapping discrete (often binary) vectors from a high-dimensional space to continuous vectors in a low-dimensional space, and which has been used for a variety of ML tasks ranging from text tagging [67] to automatic image captioning [68]. In the context of speech synthesis, both

speaker and speech style are traditionally represented as one-hot vectors, which can be considered an ignorant representation, since the similarity of two voices is not related in any way to the distance between corresponding points in the high-dimensional space [69]. The architecture and training procedure presented in this thesis overcome this deficiency by performing joint embedding of the speaker and style, representing them in a low-dimensional space in a more intuitive way, which helps the network to efficiently generalize on unseen speech data. To use the available speech data even more economically, the embedding is jointly performed not only on speaker and style ID's, but on cluster ID's as well, where the term "cluster" refers to the portion of a speaker/style dependent speech corpus which is consistent in terms of acoustic and prosodic quality. Namely, one of the practical problems in obtaining a high quality speech corpus for training, which is rarely mentioned in the literature, is maintaining the consistency of the acoustic and prosodic quality of the voice and speaking style, especially when the recording is performed in multiple sessions or the speaker takes a break within a session. This often results in parts of the corpus being slightly different in volume, timbre or even the particular way the speaker has chosen to render a speech style (e.g. "happy"). Rather than discarding the parts of a speech corpus that deviate from the corpus segment that can be termed as "default", we have opted for dividing each speaker/style-specific speech corpus into consistent clusters. Consequently, instead of supplying two non-linguistic inputs to the network (speaker ID and speech style ID), now a third input (cluster ID) is added, and these three inputs are jointly represented as a single one-hot vector, which is converted into an appropriate joint embedding through the training procedure. The effects of the division of speech data into clusters have been analyzed in [70], and it has been shown to slightly improve the quality of speech synthesis.

With the idea of improving the multi-speaker model as a starting point for speaker/style adaptation, we supplement the inputs of both neural networks (one that predicts durations and the other, which predicts acoustic features) with the information about the speaker, speaking style and cluster (SSC) in an embedded form, as shown in Figure 30. As previously explained, both networks are presented with 577 binary linguistic features (related to US English) at their inputs, with the output of the duration network serving as an additional input for the network predicting acoustic features. However, in the proposed model the input layer of each network is extended with an N-dimensional vector containing the joint embedding of

the speaker ID, speaking style ID and cluster ID, all of them originally represented in the form of a single one-hot vector of length 67, which is the number of unique SSCs existing in the training corpus. In this way it is left to the network to represent a particular SSC in a space of lower dimensionality (in our research it was set to $N = 15$). The idea of representing the speaker, the style and the cluster using 3 separate one-hot vectors was discarded since it would imply the questionable assumption that every speaker renders a speaking style in a similar way. The main advantage of the approach based on embedding is that the network itself has the opportunity to establish similarities and differences between particular speakers, styles or clusters, and based on this information, it is expected to position particular SSC combinations closer or farther from each other in the embedding space. This, in turn, will help the main network to generalize more easily, since the distance between two SSCs in the embedding space will correspond to the general difference between them. Once trained, the network will be able to synthesize speech that corresponds to a particular SSC given the corresponding point in the embedding space. Furthermore, given a random point in the embedding space, the network will be able to produce a new, previously "unseen" voice.

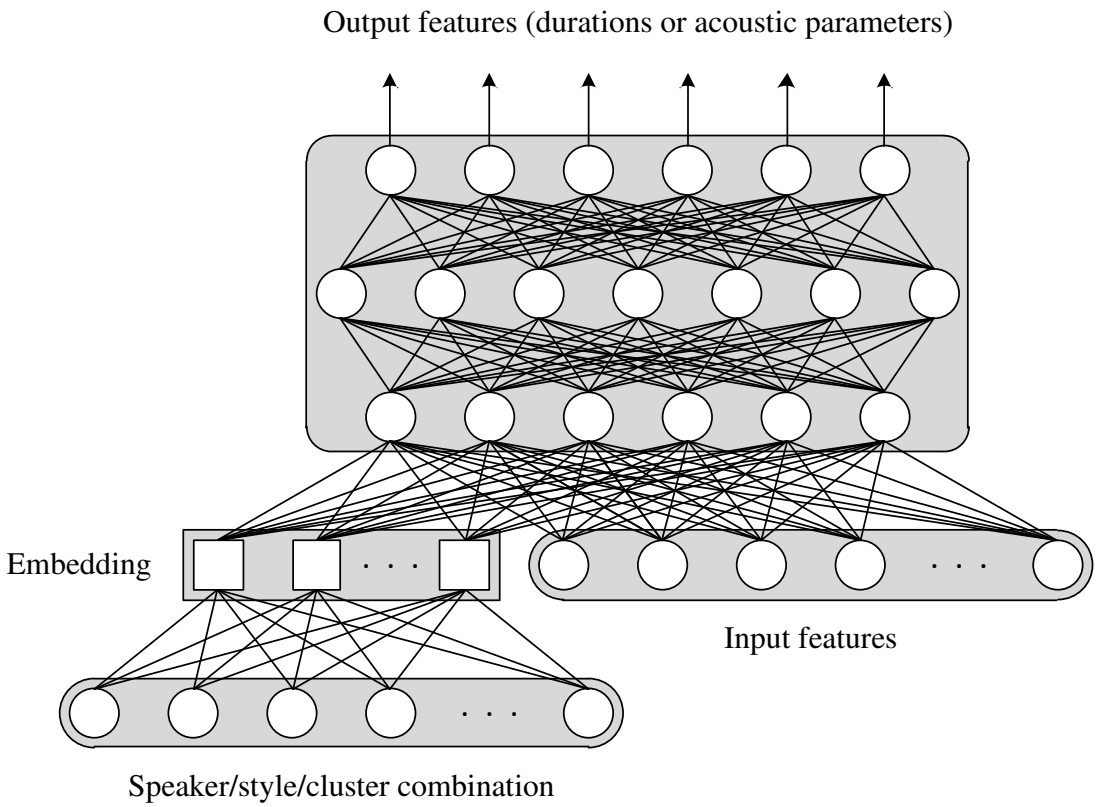


Figure 31: *The architecture of the proposed model for the two neural networks that predict either phonetic segment durations or acoustic features.*

During the initial training of the MSMS model, input to each of the networks were linguistic features as well as the one-hot vector representing the speaker, style and cluster combination, while the output were corresponding values (durations or acoustic features). By doing so the networks themselves build the embeddings for each SSC, and as a result, each SSC will be represented by two points in the corresponding embedding spaces – one in the embedding of phonetic segment durations and the other in the embedding of acoustic features. In both cases the expected outcome is that the closeness of two SSCs in embedding spaces will reflect their subjective similarity. The outcome of the initial training is the MSMS model, able to provide speech sounding like any SSC seen in the training, provided with the correct embeddings in both networks.

The described architecture and training procedure result in a MSMS TTS synthesis able to generate speech of high quality in any speaker/style/cluster combination seen in the

training corpus, but can also be easily adapted to a new speaker/style, with a relatively small amount of adaptation data.

6.2. Two-Step Adaptation Procedure

Presented model which uses trained embedded representations of SSCs can be adapted to a new speaker or style using a relatively small amount of new audio data, through a two-phase procedure. The goal of the first phase is to establish the embedding for the new speaker/style, and it starts by random initialization of the weights in the embedding layers of both duration and acoustic network. In this stage of the adaptation, only the weights in the embedding layers of both networks are updated by SGD procedure while the rest of the network is frozen (not updated). After first phase and updated embeddings, the model is capable to synthesize speech similar to the target speaker/style to some extent. The resemblance of synthesized speech to target speaker and style can be further improved through the second phase of the adaptation process, in which the same adaptation data is reused, but now the embedding layer is kept constant, while the rest of the parameters in the networks are updated by using SGD algorithm. In the following section we describe the experiments which demonstrate the ability of the initially trained MSMS model to synthesize speech in speaker/style combinations seen during the initial training, but also its ability to generate speech in a speaker/style combination not seen in the original training set (even for unseen speaker and an unknown style) after the second phase of adaptation. Through these experiments we also measure the influence of different factors, such as the perceived importance of each phase of the described adaptation process as well as the amount of target speaker's data available for adaptation.

6.3. Data Augmentation

Recent advances in deep learning models are largely attributed to the amount and diversity of data collected over the past period. Data augmentation is a strategy that allows researchers to significantly increase the diversity of available data for model training without actually having to collect new data. During augmentation, changes are made to the data

(image, sound, text ...) which can be of different scope, provided that the newly generated data must look as if they were created in a natural way, i.e. there must be no clear indication they are actually augmentations. Although the data obtained in this way are correlated with the original data, the augmentation implicitly regularizes the model and improves its ability to generalize [71]. As such, augmenting data as an approach to overcoming data sparsity has been used since the earliest days of ML [72].

In image processing, there are augmentation techniques such as: rotation, resizing, flipping, applying various filters, adding noise, or combinations of several techniques. Examples of these augmentations in the case of digit recognition can be seen in Figure 31.

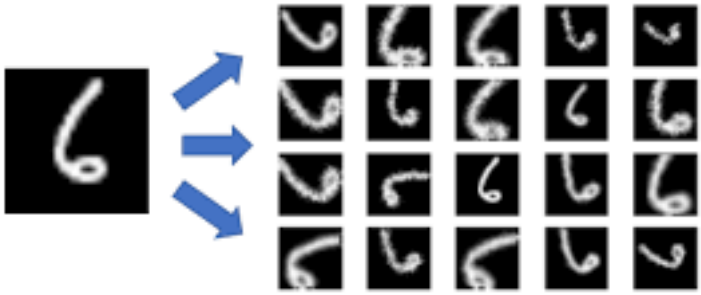


Figure 32: *Image augmentation example*

Audio data can also be augmented in a variety of ways, several of which are listed here:

- **Adding noise**. Noise can be added either by generating a random signal of a certain distribution or an audio database of variety of noises can be acquired from different sources, and then randomly added to the original recordings.
- **Time shifting**. A shift of just a few ms will lead to certain changes in the appearance of the parameters for each frame, which will make the models more robust.
- **Speed change**. This technique can be implemented either at the signal level, leading to a change in f_0 and the spectral envelope, or in the parametric domain, where it can be performed independently from other modifications.
- **Change in f_0**. The technique of modifying fundamental frequency, which can be implemented at the signal level, but is mostly done in the parametric domain.
- **Spectral envelope modifications**. After the parameters are estimated by a vocoder, it is very easy to make independent changes over different parameters. A very important

parameter (i.e. a set of parameters, as described in 2.2.2) is also the spectral envelope. **Scaling** gives different colors to voices, and after a significant change (combined with changes in f_0) there may even be a change in the perceived gender of the original speaker. **Filtering** a spectrum over the frequency scale can either make formants more prominent (band-pass filter, usually applied in order to reduce oversmoothing) or make spectral envelope more flat (low-pass filter).

- **Frequency masking**. In this procedure, certain frequency channels are masked [f_1, f_1 + $f2$). $f2$ is selected from the uniform distribution from 0 to the frequency masking parameter F, and f_1 is selected from the range (0, v-f_2) where v is the number of frequency channels [73].
- **Time masking**. t consecutive time steps [t_0, $t_0 + t$) are masked. t is chosen from a uniform distribution from 0 to the time mask parameter T, and t_0 is chosen from [0, $\tau - t$).

Virtually all the above techniques are used in speech recognition, even those which lead to noticeable speech degradation, because they contribute to the robustness of the final acoustic model. Speech synthesis usually does not use techniques which significantly degrade the recording, such as adding noise, time and frequency masking, unless a specific application requires it. Other techniques are used, which, if applied to a small extent, lead to the generation of speech that can be attributed to the same speaker, while a significant change in some of the parameters leads to the creation of a new speaker. Both variants have their benefits and are widely used in speech synthesis systems.

Changes in f_0, spectral envelope, and speed were used in this research. Speech parameterization consisted of extraction of the spectral envelope using a WORLD vocoder [13] and f_0 curve estimated by an algorithm based on autocorrelation [74]. Scaling of the spectral envelope, the f_0 curve, and the speed was chosen so that the speech, which the WORLD vocoder resynthesized, sounded like a new, yet natural speaker. Examples of three such augmentations are shown in Figure 32.

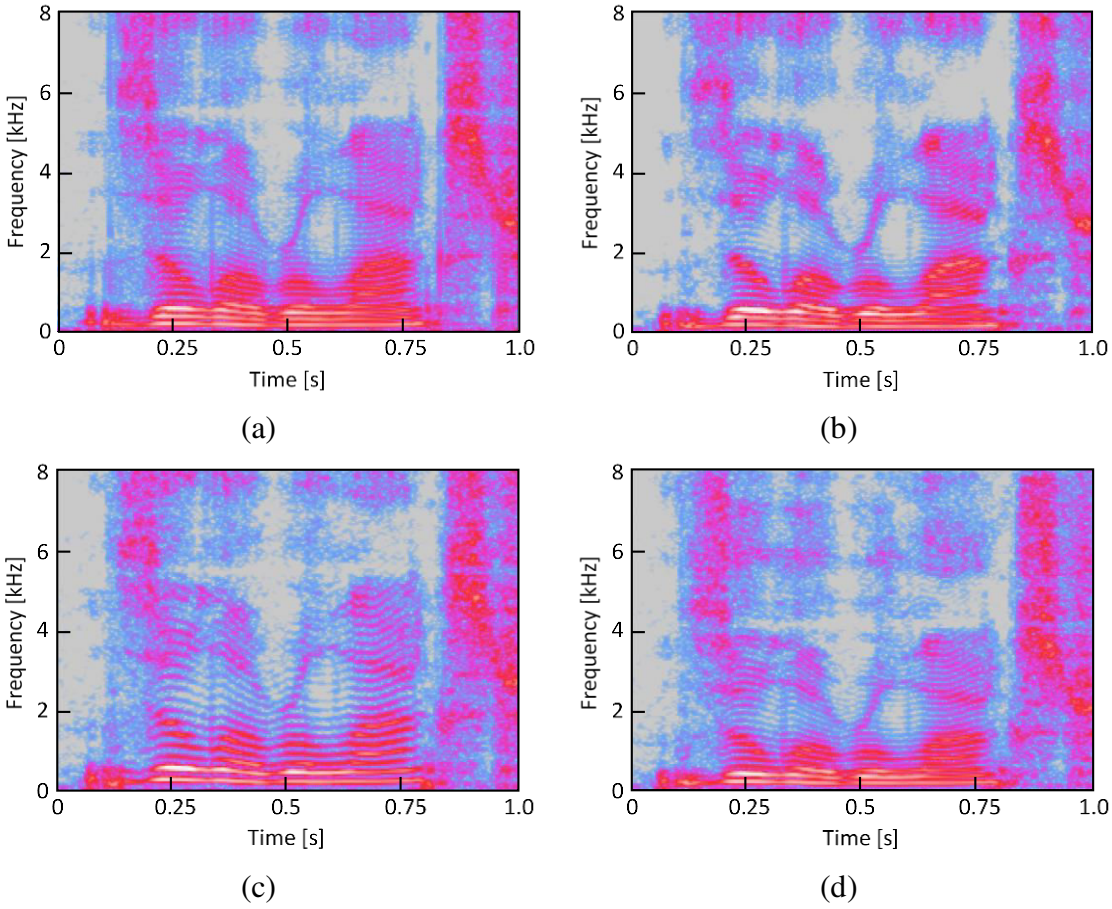


Figure 33: *Spectrograms of augmented speech: (a) original; (b) post-filtered spectral envelope (band-pass); (c) changed f0; (d) down-scaled spectral envelope*

6.4. Data

The data used to build the MSMS TTS model, as well as other models used in some of the experiments, consists of 8 hours and 38 minutes of speech from 6 American English speakers, where the quantity of speech data per speaker varied in sizes, speech styles and also acoustic quality, as shown in Table 4. All speech data was sampled at a rate of 22.05 kHz and 16 bits per sample were used.

Speaker	Gender	Quality	Style	Time [hh:mm:ss]	Total time per speaker [hh:mm:ss]
F1	female	studio	Neutral	01:30:03	02:32:59
			Apologetic	00:17:42	
			Happy	00:21:24	
			Promotional	00:23:50	
M1	male	studio	Neutral	01:38:07	03:34:11
			Angry	00:16:55	
			Apologetic	00:15:58	
			Happy	00:26:13	
			Promotional	00:28:04	
			Stern	00:28:54	
F2	female	studio	Friendly	00:31:42	01:00:25
			Promotional	00:28:43	
M2	male	studio	Friendly	00:18:26	00:39:46
			Promotional	00:21:20	
F3	female	source: YouTube	Neutral	00:26:46	00:26:46
M3	male	source: YouTube	Neutral	00:24:17	00:24:17
Total time [hh:mm:ss]:					08:38:24

Table 4: *Speech corpora used for construction of MSMS model ("time" refers to the time left when leading and trailing silences are trimmed and silent phonetic segments, such as mid-phrase silences, excluded)*

As can be seen, there are two main speakers, M1 and F1, whose data include the largest number of speech styles, and whose influence to the neutral style is the greatest. Four clusters were identified (manually) in the neutral segments of each of these two speakers. This was easy to detect since clusters usually contain contiguous utterances and the boundaries between clusters correspond to breaks within or between sessions. In order to avoid the bias towards speakers M1 and F1, and also to expand the base for the MSMS model, the available speech data was artificially augmented by introducing changes in speed, f_0 and spectral envelope into the utterances of all 6 original speakers, as described in 6.3. Using different portions of the original speakers' data, as well as additional utterances from some of them, 10 new artificial speakers were created, resulting in the total number of speakers being 16 (with 67 unique SSC combinations), and the total duration of the data available to 21 hours and 50 minutes. In 7 of the 10 artificially created speakers augmentation resulted in audible gender

switch, but a approximate balance between genders in the resulting speech corpus was preserved. The speech style ID was copied from the original corpus, while slightly different modifications of the original data were performed in order to generate different clusters of the neutral speech style. The speaker/style combinations created by augmenting speakers with less available data (F2, M2, F3 and M3) were generated by modifying clips that already exist in the original speech corpus. Speaker/style combinations created from F1 and M1 were generated by modifying both utterances from the original F1 and M1 corpora, and some previously unseen utterances, because the availability of speech data for these speakers is greater. The whole speech database was phonetically and prosodically annotated, with prosodic annotation following the extended ToBI set of conventions, as described in Section 4.5.

In the process of evaluating the ability of the system to adapt to a new speaker and style, two relatively small speech databases were used, one from a female speaker (F4) and the other from a male speaker (M4). Both these database were not present in the training of the MSMS model. The speech style in these two data sets can be named as neutral, although it should be noted that this information it actually not used.

6.5. Baseline Methods

In our experiments we compared performance of the proposed two-step adaptation procedure to two baseline methods. The first method used as a baseline (Baseline 1), presented in [64], represents one of the simplest methods for creating a voice of new speaker with a very small amount of speech training data. Its main idea is to create a speaker-dependent text-to-speech (SD TTS) model, initially trained on a large speech corpus from one speaker, and then adapt it to another speaker with a very small quantity of training data. The adaptation process differs from the standard training of SD TTS [75][46] only in the starting point, i.e. it starts from an already trained model instead of a randomly initialized one, and it proceeds in an identical way. It was shown that such an approach, using only 10 minutes of training data from the target speaker, produces results that are comparable to the results obtained from a regular SD TTS trained on a 3-hour speech corpus. Due to the limited

availability of training data, the research presented in [64] analyzed 2 SD TTS models: one based on a speech corpus from the male speaker M1 and the other based on a speech corpus from the female speaker F1, both in American English, which were identical to the ones used in this research. Since both corpora included multiple speech styles, the inputs to SD TTS models were extended with the information related to the style and cluster, both in the form of a one-hot vector, as was previously done in [76]. For the purpose of this research, speech data from the same two speakers, M1 and F1, was used to obtain two speaker-dependent TTS models that served as a basis for adaptation to the speakers M4 and F4, respectively.

The second method used as a baseline (Baseline 2) represents a slight modification of the approach described in detail in [76], where it is referred to as "separate output layer". This approach builds upon the idea presented in [57], which proposes an architecture based on shared hidden layers and multiple speaker-dependent output layers. In the second baseline approach the shared part of the network is assumed to represent a global linguistic feature transformation, while separate output layers are used for different speaker/style combinations. In the adaptation phase only a specific speaker/style-dependent output layer is adapted using the limited speaker/style-specific data, following the adaptation procedure proposed in [57]. The modification with respect to [76] lies in the introduction of an additional speaker/style-dependent hidden layer into the network structure. Similarly to the case of baseline model 1, the inputs are extended with the style and cluster codes in the form of one-hot vectors, but in this case all of the speech data listed in the Table 4. was used for training the multi-speaker/multi-style model that was subsequently adapted to M4 and F4.

It should be noted here that we did not conduct experiments in which the entire model was trained from scratch (randomly initialized network) on a very small amount of data (10 minutes or less). These experiments were tried even before the described baseline methods, but the results obtained were hardly intelligible, let alone natural sounding. Therefore, we have dismissed that approach and decided not to include it in our comparison.

6.6. Experiments

In this thesis the proposed model is trained on the same speech data as the two baseline models described in previous section. However, while the MS models (proposed model and baseline 2) were trained on the entire database presented in Table 4, the baseline model 1 was trained only on M1 and F1 in order to create two speaker-dependent models (not multi-speaker models). To test the ability of all three models to adapt to a new speaker and style, for adaptation purposes speech data from speakers M4 and F4 were used. Since the specific aim of this thesis is to explore the case when the amount of target speech data is very limited, the adaptation experiments were conducted with speech databases containing from 30 seconds to 10 minutes of target speech data. The initial speaker-dependent model of the same gender was used in each case for the adaptation of the baseline model 1. Since the baseline 2 model actually contains 16 different speakers (6 genuine and 10 obtained by augmentation), those used as initial points for adaptation in this thesis were the ones that correspond to M1 or F1 (the one that matches the gender of the target speaker). The dimension of the embedding was set to 15, although it was observed that it is of surprisingly little importance to the quality of the output (values ranging from 4 to 40 were tried). The ability of the proposed model to synthesize speech in the voice of the intended speaker/style was evaluated by both objective and subjective measures. Objective measures are represented through the distance between corresponding acoustic features of the original and synthetic speech, while subjective evaluation consists of a series of listening tests. Both measures are specifically aimed at establishing the importance of the position of the SSC points in the two embedding spaces, the relevance of each phase in the adaptation process, as well as the influence of training data. Speech samples used for both objective and subjective evaluation are available at the URL: www.alfanum.ftn.uns.ac.rs/embedding.

6.6.1. **Objective Evaluation**

In order to get objective evaluation of the three models, the values of state durations and acoustic parameters were compared between synthesized speech and original target speech data in case both phases of the adaptation process were performed. For evaluation we used target speech data which was not present in any of the training phases. The acoustic

parameters included in objective measures were the root mean square error and correlation for f_0, RMSE and correlation for the duration of phones as well as mel cepstral distance as explained in Section 2.4. The results are presented in Figure 33.

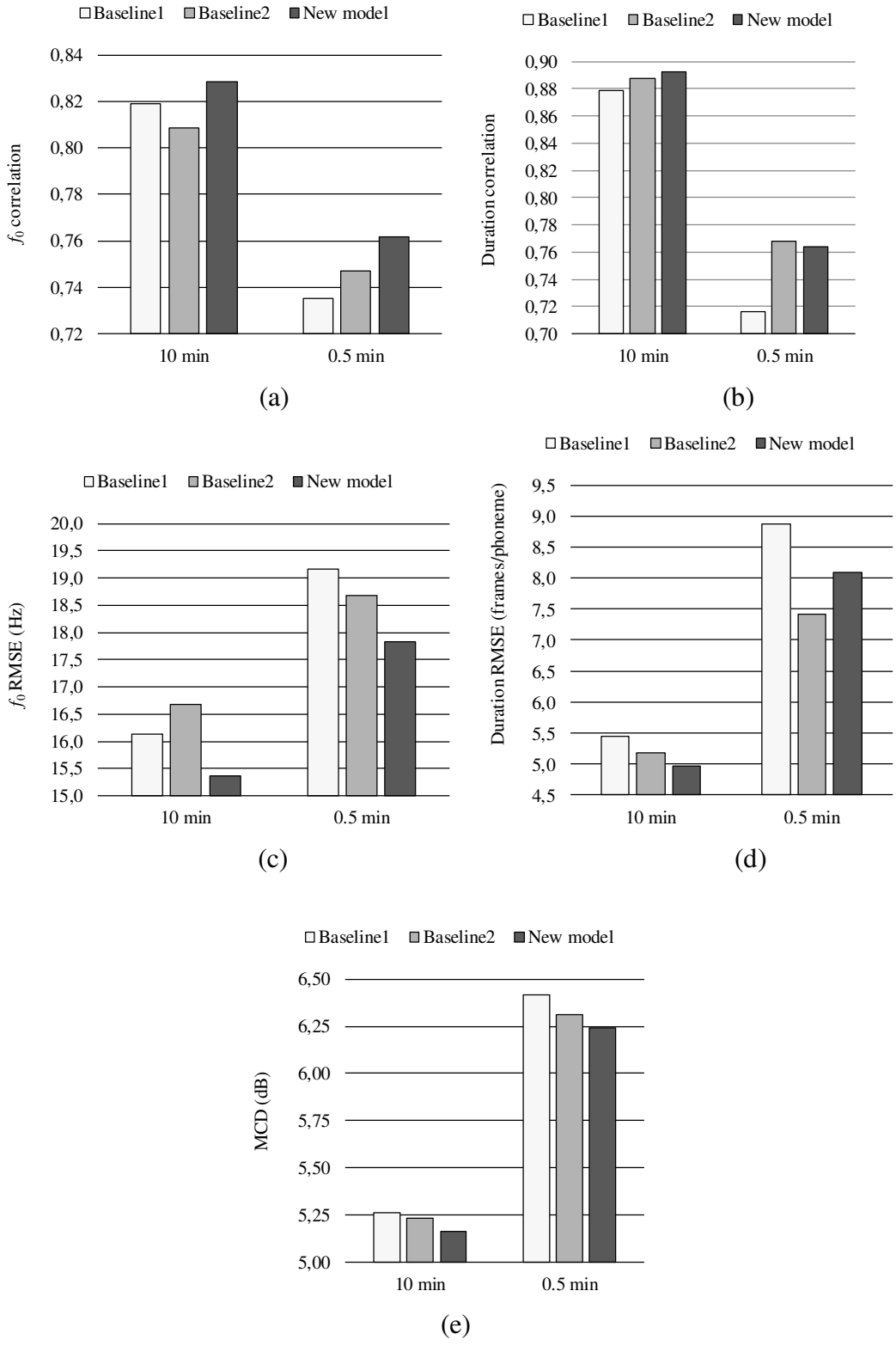


Figure 34: *The results of the objective evaluation of the proposed model against the two baseline models: (a) correlation of f_0; (b) correlation of phone durations; (c) RMSE of f_0; (d) RMSE of phone durations; (e) MCD.*

It can be seen that the correlation between the predicted *f0* contour and the ground truth (*f0* in the original clip), as well as the correlation between the predicted phone durations and the ground truth, is quite high for all three models, but that the proposed model consistently outperforms the other two, regardless of the amount of speech used for adaptation. It can also be seen that the differences are slightly higher in case when less target speech data was used. The baseline model 1 seems to degrade the most with the decrease of the quantity of adaptation data, although the differences are not substantial in this case either. The differences between the models are more significant in case of RMSE of *f0* and phone durations. The proposed model performs better the two baseline models in most cases, and the baseline model 1 appears to be least successful. The differences between the models are again more pronounced in case of the smaller adaptation set.

As for MCD, the differences among the models are almost negligible, but the proposed model consistently outperforms the others, and the baseline model 1 performs the worst.

6.6.2. **Subjective Evaluation**

A number of listening tests was performed in order to compare the results of the objective evaluation with the subjective perception and to establish the influence of various factors to the quality of synthesized speech after adaptation of the initial model to the target data.

6.6.3. **Experiment 1**

The aim of this experiment was to evaluate how successful the proposed model is in generating speech that is intended to be similar to a particular speaker and speaking style in case only a small quantity of target speech data is available. It also examines the influence of the relationship between the position of the SSC points in embedding spaces and the degree to which the synthesized speech resembles to the target speaker and style. Furthermore, the experiment also proves the importance of the second phase of the adaptation process, which has been shown to increase the similarity of the synthesized speech to the target speaker/style combination. The experiment explores only the proposed model and does not compare it to the baseline models.

The experiment was set up as a MUSHRA listening test, and conducted among 26 listeners. Each listener was presented with 10 tasks, including 5 sentences in the voices of 2 speakers (M4 or F4). In each task, the listener was presented with the following 5 versions of the same utterance, in a randomized order:

- Hidden reference recording (original recording of the source speaker);
- Synthesis after just the first adaptation phase has been performed on the initial model;
- Synthesis after the first adaptation phase has been performed and then the obtained embedding was modified by 10%;
- Synthesis after the first adaptation phase has been performed and then the obtained embedding was modified by 20%;
- Synthesis after both phases of the adaptation procedure have been performed on the initial model without modifying the embedding obtained in the first phase.

In this experiment adaptation was performed using 10 minutes of target speech data. In cases the obtained embedding was modified, the modification was performed for each of the 15 dimensions of the embedding, in the following way. Firstly, the reference range for each dimension was calculated as the sample standard deviation of its 67 points (one for each SSC) multiplied by 6. After that the actual coordinate was modified by ±10% or ±20% of the calculated reference range. The reference recording was explicitly marked as such (usual practice in MUSHRA tests), but it was also hidden among the 5 utterances chosen for grading. The listeners were asked to rate speaker similarity between the reference and each of the 5 utterances on a scale of 0 to 100. Since there is a tendency of giving lower grades to less appealing voices, which might blur the influence of the factors that were considered as relevant, the grade given to the hidden reference was scaled up to the maximum grade, and the rest of the grades were scaled accordingly. Furthermore, in order to make comparison of the results across all experiments more simple, all grades are presented as rescaled to the interval 0-5.

The results, shown in Figure 34, suggest that the first phase of the adaptation alone is sufficient for the model to be capable of producing speech that roughly resembles the target speaker. It can also be seen that the position of the embedding generated through initial model training is significant, since if it is modified, resemblance to the target speaker is

partially lost (variation of each coordinate by 10% leads to a relatively small change, but an increase to 20% of the initial value reduces the mean score from 2.5 to 1.2). This experiment has also demonstrated the relevance of the second phase, since the grade achieved after both adaptation phases is substantially higher than any grade seen after performing only the first phase of adaptation. A substantial margin still exists between the original and the synthesized speech, and one explanation could be that it is because of the relatively poor coverage of the embedding space by the SSCs present in the training database. If more diverse data was used for training the initial MSMS model, it could be expected that the synthesis after adaptation to a new speaker and style would exhibit less audible artefacts, and would be perceived as more similar to the original speaker by the listeners.

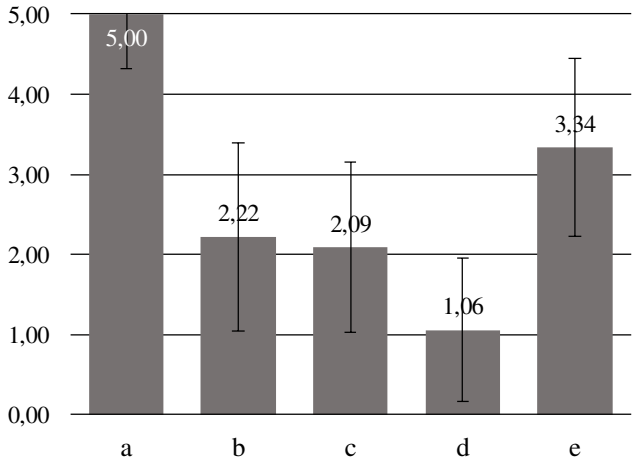


Figure 35: *Subjective assessment of speaker similarity to the reference recording, rescaled to 5.00: (a) reference recording; (b) synthesis after the first phase of adaptation of the initial model; (c) synthesis after the first phase of adaptation and thus obtained embedding modified by 10%; (d) synthesis after the first phase of adaptation of the initial model and thus obtained embedding modified by 20%; (e) synthesis after both phases of adaptation.*

6.6.4. **Experiment 2**

The purpose of this experiment was to compare the quality of synthesized speech by the proposed model with the two baseline models after adaptation, not taking into account speaker similarity with the reference speaker, through a MUSHRA listening test with 24 subjects. In each of the 20 tasks, the subjects were informed that the reference audio clip is a

recording of natural speech, and they were asked to grade the quality (intelligibility and naturalness) not taking into account speaker similarity, the following modifications of the same utterance:

- Hidden reference recording (original recording of the source speaker);
- Synthesis by the baseline model 1 after adaptation;
- Synthesis by the baseline model 2 after adaptation;
- Synthesis by the proposed model after the embedding obtained in the initial training is reset to 0 and only the second phase of adaptation is carried out;
- Synthesis by the proposed model after both phases of adaptation.

All utterances appeared in a randomized order.

Among these 20 tasks, 10 of them had models adapted by using 10 minutes , and the remaining 10 by using only 30 seconds of target speech data. In each of these two instances there were 5 utterances by each of the 2 speakers (M4 and F4).

The results (Figure 35) show that regardless of the quantity of target speech used for adaptation of the model, baseline model 2 was considered worst by the subjects, while the two versions of the proposed model received the highest grades. It should be noted that, although average grades for baseline model 1 and the proposed model do not differ that much in case when 10 minutes of adaptation data were used, the proposed model gets significantly higher grades than the baseline model 1 in case adaptation is performed using only 30 seconds of target speech. That is to say, the proposed model seems to be more robust to small amount of adaptation data compared to any of the baseline models. Also worth mentioning is that, if the initial embedding is reset to 0 and we perform only the second phase of adaptation, it does not significantly degrade the quality of synthesis. However, it is still slightly higher if the initial embedding adapted to the new speaker is used.

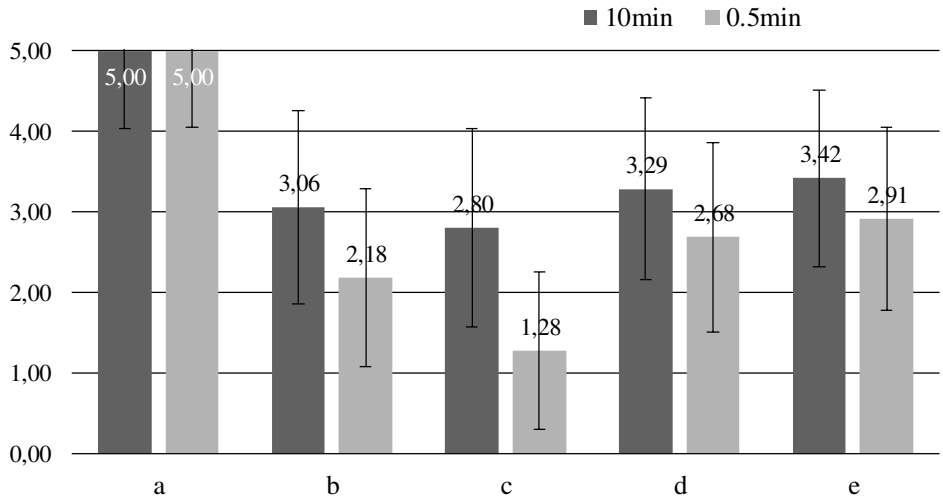


Figure 36: *Comparison of the quality of synthesis obtained in different conditions, rescaled to 5.00: (a) reference recording; (b) synthesis by the baseline model 1 after adaptation; (c) synthesis by the baseline model 2 after adaptation; (d) Synthesis by the proposed model after the embedding is reset to 0 and only the second phase of adaptation is carried out; (e) synthesis by the proposed model after both phases of adaptation.*

6.6.5. **Experiment 3**

Experiment 3 was performed in the same way as Experiment 2 regarding the versions of synthesized speech that were presented to the subjects in each task, but this time the subjects were asked to evaluate similarity between speakers instead of the general quality. The experiment involved 10 tasks (5 for each of the speakers, M4 and F4), and 20 subjects performed evaluation. As Experiment 2 has shown that the general quality of synthesized speech is quite different for the three models in case of adaptation on very small amount of data, adaptation was conducted only on 10-minute target speaker datasets, to prevent the subjects from being distracted by this difference so that they could focus only on speaker similarity. As shown in Figure 36, the proposed model performs better than both baseline models regarding production of synthesized speech in a voice that resembles the original speaker, even when the embedding is reset to 0 and only the second phase of adaptation is performed.

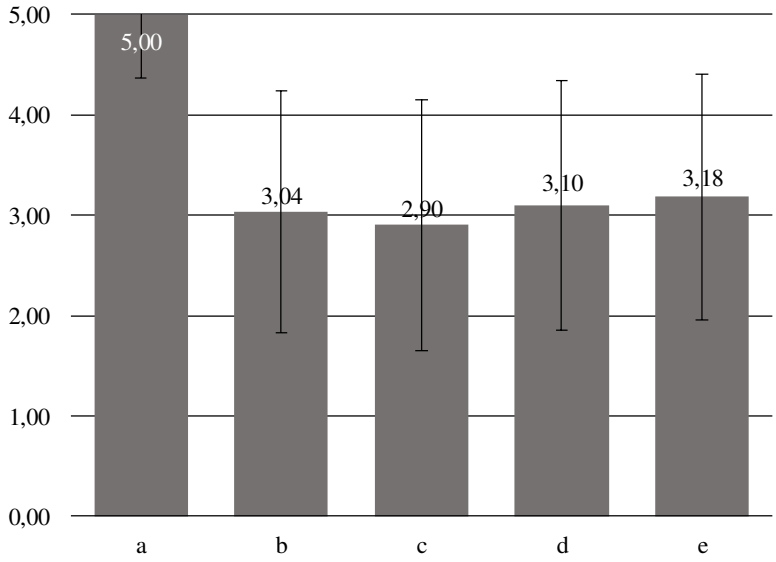


Figure 37: *Comparison of the speaker similarity obtained in different conditions, rescaled to 5.00: (a) reference recording; (b) synthesis by the baseline model 1 after adaptation; (c) synthesis by the baseline model 2 after adaptation; (d) Synthesis by the proposed model after the embedding is reset to 0 and only the second phase of adaptation is carried out; (e) synthesis by the proposed model after both phases of adaptation.*

6.6.6. **Experiment 4**

In order to measure full capability of the proposed model it is necessary to compare it with another speaker-dependent baseline model using large quantities of target speaker data for training. However, we were unable to perform such an evaluation directly because only a small amount of speaker data for the speakers M4 and F4 was available, and the remaining speakers were already used for the initial model training. In order to circumvent this limitation we conducted experiment by including two types of MUSHRA tasks (10 tasks of each type). In both types of tasks, the 32 subjects in the listening test received information that the reference utterance is actually a natural recording of speech, and were asked to grade the general quality (intelligibility and naturalness) of 3 utterances provided in random order. In the tasks of type 1 the following 3 utterances were presented:

- Hidden reference recording (original recording of M1 or F1);
- Synthesis by the baseline model 1 trained on all available data for M1 or F1 (see Table 4), without further adaptation;

- Synthesis by the proposed model using embeddings corresponding to M1 or F1, without further adaptation;

while the tasks of type 2 consisted of the following 3 utterances:

- Hidden reference recording (original recording of M4 or F4);
- Synthesis by the proposed model after both phases of adaptation to M4 or F4, using 10 minutes of target speaker data;
- Synthesis by the proposed model after both phases of adaptation to M4 or F4, using 30 seconds of target speaker data.

In each task the 3 given utterances originated from the same speaker in order to eliminate the preference that a subject may have for some of the voices. This is also the reason why we separated these two tasks instead of conducting only one. All speakers were equally distributed in the experiment, i.e. each of them appeared in 5 tasks.

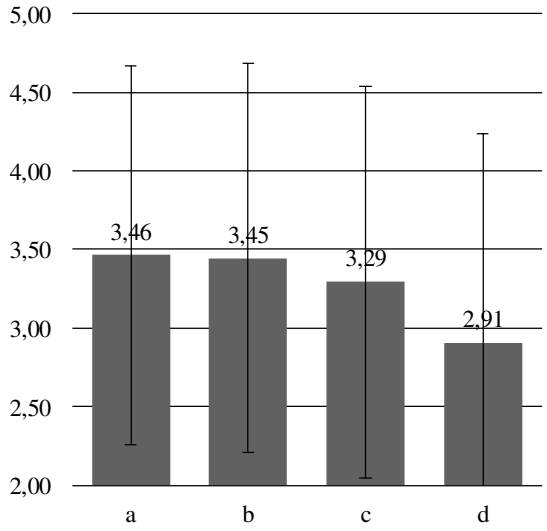


Figure 38: *Comparison of the quality of synthesis obtained in different conditions, rescaled to 5.00: (a) Synthesis by the baseline model 1 trained on all available data for M1 or F1 without further adaptation; (b) Synthesis by the proposed model using embeddings corresponding to M1 or F1 without further adaptation; (c) Synthesis by the proposed model after both phases of adaptation to 10 minutes of speech data from M4 or F4; (d) Synthesis by the proposed model after both phases of adaptation to 30 seconds of speech data from M4 or F4.*

The results of the experiment are shown in Figure 37 (scores rescaled to the interval 0-5). Before commenting the results, it is important to mention that although M1 and F1 were not in the same tasks as M4 and F4, we can still compare the subjective quality of synthesis between models and versions that were not present in the same tasks. Noteworthy, the synthesis by the baseline model 1 trained on all available data for M1 or F1 and synthesis by the proposed model after two-phase adaptation to M4 or F4, using 10 minutes of speech (items (a) and (c) in Figure 37) were graded as similar in quality. This shows that the proposed model, given properly trained MSMS model as a starting point, and using as little as 10 minutes of adaptation material, is able to reach a quality of synthesis similar to that of a standard speaker-dependent model trained on much more speech data (~3.5 hours in case of M1 and ~2.5 hours in case of F1). Additionally, synthesized speech generated by the baseline model 1 trained on all available data for M1 or F1 (~3.5 and ~2.5 hours respectively) seems to be of the equal quality as the synthesis by the proposed model using embedding points corresponding to M1 or F1 and no additional adaptation. This means that given quantity of training data for a certain speaker can be used as a basis for a multispeaker model based on embeddings and to train a single speaker-dependent model, with similar outcomes. As a final point, it should be noted that the adaptation of the proposed model using seconds of material yielded synthetic speech that was rated as being of lower quality than in case the adaptation was performed on 10 minutes of speech. Nevertheless, the difference in scores is only 0.38, which is quite small taking into account the difference in the amount of adaptation data.

7. Conclusion

In this research we deal with the problem of creating high quality synthetic voices when only small amount of data is available. The subject has been the focus of many studies for decades, because it has numerous uses and potentially significantly reduces the effort for creating new voices, by making it much faster and less expensive. After introducing and comparing usual approaches to speech synthesis, we also illustrate a number of previous attempts to address this problem. Some of them were based on older approaches (e.g. HMM-TTS), while some more recent ones tried to offer solution by using DNN architecture.

Two different adaptation approaches were proposed in this thesis. Both are deep neural network based speech synthesis models, capable of adaptation to a particular speaker and speaking style. The first method initially trains DNN-based TTS on relatively large amount of training material (3+ hours) and uses that model as a starting point for adaptation. This means that the new model is not trained on a randomly initialized network (weights and biases), but on an already pretrained one, which resulted in much better performance (higher quality of synthesized speech). Because of the small amount of adaptation material for new speaker, we had to devise a new alignment procedure, which outperformed the default one, provided in the tool. Both male and female initial models were trained and used as starting points for adaptation, and the outcomes compared. No significant difference was observed when different starting models were used. Also, different sizes of adaptation material were used and quality of the obtained speech was compared. As expected, more material yielded better results, but it was shown that adapting even with a relatively small amount of data could provide comparable results to models trained from scratch with much more speech material. The evaluation was based on objective measures, but also on listening tests.

The second method is the two step adaptation process in which we first find the optimal embedding for the target voice in an iterative way. Before that we have to build a multi-speaker multi-style model, based on many speakers, during which process the embedding space is built. The second step consists of adaptation of the rest of the neural network, by optimizing all the weights and biases so the resulting network can produce the speech of the

target speaker. Since the output after phase one is already close to the target, the amount of changes applied to the network is relatively small. This prevents the network from overfitting and enables much better generalization of unseen events.

The second method has been shown to outperform two other recently proposed parametric speaker/style-dependent speech synthesis models, particularly in case the quantity of available adaptation data is extremely small. This is achieved owing to the joint representation of speaker, speaking style and cluster by their low-dimensional embedding, whereby the model is able to establish the similarities or differences among speakers and styles, and consequently generalize more accurately.

The embedding approach opens up a range of interesting possible applications of the proposed model in any domain where the possibility of quick and efficient adaptation of speech synthesis to a new speaker and/or style is required.

7.1. Future Work

A limitation of this research that cannot be disregarded is the relatively small quantity of speech data on which it was based. Namely, for the second approach (which yielded the best results), only 8 hours and 38 minutes of actual speech from 6 speakers was available for training, and a total of 20 minutes was available for adaptation, which is why data augmentation had to be applied. Although this is a valid technique aimed at overcoming data scarcity, the question remains to what extent a stronger multi-speaker/multi-style basis, including a greater number of speakers/styles, would improve the ability of the proposed system to produce synthetic speech of high intelligibility, naturalness and similarity to the target speaker/style. For that reason, the model will certainly be reinvestigated as soon as a significantly greater amount of training data becomes available, and this will also be an opportunity to study the influence of data augmentation to the performance of the model.

Another issue that will be further investigated is the influence of the difference in the quantities of available training data related to particular speakers/styles. This research in particular may have suffered from two speakers (M1 and F1) being overrepresented in the

training data. In the future versions of the proposed model we intend to equalize the influence of all speakers/styles on the training process by introducing weight coefficients corresponding to their relative contributions.

As explained in Section 2.2.2, WaveNet is a neural vocoder which produces output speech of almost perfect quality. It could be used instead of WORLD vocoder and additionally raise the quality of output speech and similarity to the original voice. The process would be quite similar to what has been done with current NNs: first train WaveNet with speaker embeddings; in the first step of adaptation, estimate the optimal embedding for the target speaker; then additionally optimize WaveNet to better represent the target speaker.

Similar architecture and approach could be applied to style transplantation and polyglot TTS. Style transplantation is a process where we make TTS generate sentences in a certain voice and style, even though the target speaker never actually produced any utterance in the target style. The idea is to transform the embedding point of that speaker in appropriate way, so when synthesis is run by using that new "speaker" we get desired style. The process is, of course, far from trivial and will require additional research.

Multilingual or polyglot TTS is a system able to produce synthetic speech in a certain voice in several languages even though the original speaker provided speech data in only one language. Besides the obvious advantage of being able to provide personalized speech in several languages, this approach also offers building a more comprehensive speaker embedding space, by using data from multiple languages. In other words, just as a multi-speaker model is able to learn general characteristics of human speech and of a particular language regardless of the differences between the voices of particular speakers, a multi-language model goes one step beyond and learns about human speech by "listening" to speech samples in different languages.

Direct voice conversion (from audio to audio, without generating text) is also a very promising technology with a wide variety of applications. Although it has certain similarities to what was researched in this thesis, it is based on somewhat different approaches and requires additional research.

8. Literature

[1] J. T. Wood, *Communication mosaics: An introduction to the field of communication*. Cengage Learning, 2013.

[2] M. R. Schroeder, "A brief history of synthetic speech," *Speech Commun.*, vol. 13, no. 1–2, pp. 231–237, 1993.

[3] A. van den Oord *et al.*, "Wavenet: A generative model for raw audio," *arXiv Prepr. arXiv1609.03499*, 2016.

[4] S. King, "A beginners' guide to statistical parametric speech synthesis," Edinburgh, 2010.

[5] Z. Wu, O. Watts, and S. King, "Merlin: An open source neural network speech synthesis system," *Proc. SSW, Sunnyvale, USA*, 2016.

[6] M. Abadi *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI '16)*, 2016, pp. 265–283.

[7] F. Seide and A. Agarwal, "CNTK: Microsoft's open-source deep-learning toolkit," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, p. 2135.

[8] D. H. Klatt, "Review of text-to-speech conversion for English," *J. Acoust. Soc. Am.*, vol. 82, no. 3, pp. 737–793, 1987.

[9] E. Ostrom, L. Schroeder, S. Wynne, and others, *Institutional incentives and sustainable development: infrastructure policies in perspective.* Westview Press, 1993.

[10] K. Ishizaka and J. L. Flanagan, "Synthesis of voiced sounds from a two-mass model of the vocal cords," *Bell Syst. Tech. J.*, vol. 51, no. 6, pp. 1233–1268, 1972.

[11] H. Sak, T. Güngör, and Y. Safkan, "A corpus-based concatenative speech synthesis

system for Turkish," *Turkish J. Electr. Eng. Comput. Sci.*, vol. 14, no. 2, pp. 209–223, 2006.

[12] H. Kawahara, "Speech representation and transformation using adaptive interpolation of weighted spectrum: vocoder revisited," in *1997 IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1997, vol. 2, pp. 1303–1306.

[13] M. Morise, F. Yokomori, and K. Ozawa, "WORLD: a vocoder-based high-quality speech synthesis system for real-time applications," *IEICE Trans. Inf. Syst.*, vol. 99, no. 7, pp. 1877–1884, 2016.

[14] M. Morise, H. Kawahara, and H. Katayose, "Fast and reliable F0 estimation method based on the period extraction of vocal fold vibration of singing voice and speech," in *Audio Engineering Society Conference: 35th International Conference: Audio for Games*, 2009.

[15] M. Morise, "CheapTrick, a spectral envelope estimator for high-quality speech synthesis," *Speech Commun.*, vol. 67, pp. 1–7, 2015.

[16] M. Morise, "Platinum: A method to extract excitation signals for voice synthesis system," *Acoust. Sci. Technol.*, vol. 33, no. 2, pp. 123–125, 2012.

[17] H. Cryer and S. Home, "Review of methods for evaluating synthetic speech," *RNIB Cent. Access. Inf. (CAI), Tech. Rep.*, vol. 8, 2010.

[18] J. Kominek, T. Schultz, and A. W. Black, "Synthesizer voice quality of new languages calibrated with mean mel cepstral distortion," in *Spoken Languages Technologies for Under-Resourced Languages*, 2008.

[19] N. Campbell, "Evaluation of speech synthesis," in *Evaluation of text and speech systems*, Springer, 2007, pp. 29–64.

[20] I. Recommendation, "Bs. 1534-1. method for the subjective assessment of intermediate sound quality (mushra)," *Int. Telecommun. Union, Geneva*, 2001.

[21] K. Tokuda, T. Yoshimura, T. Masuko, T. Kobayashi, and T. Kitamura, "Speech

parameter generation algorithms for HMM-based speech synthesis," in *Acoustics, Speech, and Signal Processing, 2000. ICASSP'00. Proceedings. 2000 IEEE International Conference on*, 2000, vol. 3, pp. 1315–1318.

[22] C. J. Leggetter and P. C. Woodland, "Flexible speaker adaptation for large vocabulary speech recognition," in *Fourth European Conference on Speech Communication and Technology*, 1995.

[23] C. J. Leggetter and P. C. Woodland, "Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models," *Comput. Speech Lang.*, vol. 9, no. 2, pp. 171–185, 1995.

[24] C. J. Leggetter, "Improved acoustic modeling for HMMs using linear transformations.," University of Cambridge, 1995.

[25] C. C. Aggarwal and others, *Neural networks and deep learning*. Springer, 2018.

[26] J. L. Elman, "Finding structure in time," *Cogn. Sci.*, vol. 14, no. 2, pp. 179–211, 1990.

[27] M. I. Jordan, "Artificial neural networks," *Attractor Dyn. parallelism a Connect. Seq. Mach.*, pp. 112–127, 1990.

[28] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[29] Y. Le Cun *et al.*, "Handwritten digit recognition: Applications of neural network chips and automatic learning," *IEEE Commun. Mag.*, vol. 27, no. 11, pp. 41–46, 1989.

[30] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural networks*, vol. 4, no. 2, pp. 251–257, 1991.

[31] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Math. Control. signals Syst.*, vol. 2, no. 4, pp. 303–314, 1989.

[32] M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken, "Multilayer feedforward networks with a nonpolynomial activation function can approximate any function," *Neural networks*, vol. 6, no. 6, pp. 861–867, 1993.

[33] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning internal representations by error propagation,” 1985.

[34] B. T. Polyak, “Some methods of speeding up the convergence of iteration methods,” *USSR Comput. Math. Math. Phys.*, vol. 4, no. 5, pp. 1–17, 1964.

[35] A. J. Hunt and A. W. Black, “Unit selection in a concatenative speech synthesis system using a large speech database,” in *Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on*, 1996, vol. 1, pp. 373–376.

[36] M. Tamura, T. Masuko, K. Tokuda, and T. Kobayashi, “Adaptation of pitch and spectrum for HMM-based speech synthesis using MLLR,” in *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No. 01CH37221)*, 2001, vol. 2, pp. 805–808.

[37] A. Gutkin, X. Gonzalvo, S. Breuer, and P. Taylor, “Quantized HMMs for low footprint text-to-speech synthesis,” in *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[38] J. Yamagishi, Z. Ling, and S. King, “Robustness of HMM-based speech synthesis,” 2008.

[39] J. Yamagishi *et al.*, “Robust speaker-adaptive HMM-based text-to-speech synthesis,” *IEEE Trans. Audio. Speech. Lang. Processing*, vol. 17, no. 6, pp. 1208–1230, 2009.

[40] S. Esmeir and S. Markovitch, “Anytime learning of decision trees,” *J. Mach. Learn. Res.*, vol. 8, no. May, pp. 891–933, 2007.

[41] K. Yu, H. Zen, F. Mairesse, and S. Young, “Context adaptive training with factorized decision trees for HMM-based statistical parametric speech synthesis,” *Speech Commun.*, vol. 53, no. 6, pp. 914–923, 2011.

[42] Y. Bengio and others, “Learning deep architectures for AI,” *Found. trends®in Mach. Learn.*, vol. 2, no. 1, pp. 1–127, 2009.

[43] K. Yu, F. Mairesse, and S. Young, “Word-level emphasis modeling in HMM-based speech synthesis,” in *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2010, pp. 4238–4241.

[44] H. Zen *et al.*, “The HMM-based speech synthesis system (HTS) version 2.0.,” in *SSW*, 2007, pp. 294–299.

[45] A. Black, P. Taylor, R. Caley, and R. Clark, “The Festival speech synthesis system.” 1998.

[46] T. Delić, M. Sečujski, and S. Suzić, “A review of Serbian parametric speech synthesis based on deep neural networks,” *Telfor J.*, vol. 9, no. 1, pp. 32–37, 2017.

[47] L. E. Baum, T. Petrie, G. Soules, and N. Weiss, “A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains,” *Ann. Math. Stat.*, vol. 41, no. 1, pp. 164–171, 1970.

[48] N. Kalchbrenner *et al.*, “Efficient neural audio synthesis,” *arXiv Prepr. arXiv1802.08435*, 2018.

[49] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, “Empirical evaluation of gated recurrent neural networks on sequence modeling,” *arXiv Prepr. arXiv1412.3555*, 2014.

[50] Z. Wu and S. King, “Investigating gated recurrent networks for speech synthesis,” in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 5140–5144.

[51] T. T. D. Team *et al.*, “Theano: A Python framework for fast computation of mathematical expressions,” *arXiv Prepr. arXiv1605.02688*, 2016.

[52] A. Paszke *et al.*, “PyTorch: An imperative style, high-performance deep learning library,” in *Advances in Neural Information Processing Systems*, 2019, pp. 8024–8035.

[53] S. Suzić, T. Delić, D. Pekar, and V. Ostojić, “Novel alignment method for DNN TTS

training using HMM synthesis models," in *2017 IEEE 15th International Symposium on Intelligent Systems and Informatics (SISY)*, 2017, pp. 271–276.

[54] M. Sečujski, S. J. Ostrogonac, S. B. Suzić, and D. J. Pekar, "Learning prosodic stress from data in neural network based text-to-speech synthesis," *Трудьı СПИИРАН*, vol. 59, no. 0, pp. 192–215, 2018.

[55] Z. Wu, P. Swietojanski, C. Veaux, S. Renals, and S. King, "A study of speaker adaptation for DNN-based speech synthesis," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[56] P. Swietojanski and S. Renals, "Learning hidden unit contributions for unsupervised speaker adaptation of neural network acoustic models," in *2014 IEEE Spoken Language Technology Workshop (SLT)*, 2014, pp. 171–176.

[57] Y. Fan, Y. Qian, F. K. Soong, and L. He, "Multi-speaker modeling and speaker adaptation for DNN-based TTS synthesis," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, 2015, pp. 4475–4479.

[58] O. Watts, Z. Wu, and S. King, "Sentence-level control vectors for deep neural network speech synthesis," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[59] H.-T. Luong, S. Takaki, G. E. Henter, and J. Yamagishi, "Adapting and controlling DNN-based speech synthesis using input codes," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 4905–4909.

[60] N. Hojo, Y. Ijima, and H. Mizuno, "An Investigation of DNN-Based Speech Synthesis Using Speaker Codes," in *INTERSPEECH*, 2016, pp. 2278–2282.

[61] K. Inoue, S. Hara, M. Abe, N. Hojo, and Y. Ijima, "An investigation to transplant emotional expressions in DNN-based TTS synthesis," in *2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, 2017, pp. 1253–1258.

[62] S. Suzić, T. Delić, D. Pekar, V. Delić, and M. Sečujski, "Style transplantation in neural network based speech synthesis," *Acta Polytech. Hungarica*, vol. 16, no. 6, 2019.

[63] J. Parker, Y. Stylianou, and R. Cipolla, "Adaptation of an expressive single speaker deep neural network speech synthesis system," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 5309–5313.

[64] T. Delić, S. Suzić, M. Sečujski, and D. Pekar, "Rapid Development of New TTS Voices by Neural Network Adaptation," in *2018 17th International Symposium INFOTEH-JAHORINA (INFOTEH)*, 2018, pp. 1–6.

[65] O. Watts, G. E. Henter, T. Merritt, Z. Wu, and S. King, "From HMMs to DNNs: where do the improvements come from?," in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, 2016, pp. 5505–5509.

[66] E. Vincent, M. Jafari, and M. Plumbley, "Preliminary guidelines for subjective evalutation of audio source separation algorithms," in *UK ICA Research Network Workshop*, 2006.

[67] P. Wang, Y. Qian, F. K. Soong, L. He, and H. Zhao, "A unified tagging solution: Bidirectional lstm recurrent neural network with word embedding," *arXiv Prepr. arXiv1511.00215*, 2015.

[68] J. Mao, W. Xu, Y. Yang, J. Wang, and A. L. Yuille, "Explain images with multimodal recurrent neural networks," *arXiv Prepr. arXiv1410.1090*, 2014.

[69] J. Lorenzo-Trueba, G. E. Henter, S. Takaki, J. Yamagishi, Y. Morino, and Y. Ochiai, "Investigating different representations for modeling and controlling multiple emotions in DNN-based speech synthesis," *Speech Commun.*, vol. 99, pp. 135–143, 2018.

[70] T. Nosek, S. Suzić, M. Sečujski, and D. Pekar, "Improvement of the quality of neural network based speech synthesis through intra-speaker clustering," *Proc. Tak.*, 2019.

[71] V. N. Vapnik and A. Y. Chervonenkis, “On the uniform convergence of relative frequencies of events to their probabilities,” in *Measures of complexity*, Springer, 2015, pp. 11–30.

[72] P. Simard, B. Victorri, Y. LeCun, and J. Denker, “Tangent prop-a formalism for specifying selected invariances in an adaptive network,” in *Advances in neural information processing systems*, 1992, pp. 895–903.

[73] D. S. Park *et al.*, “Specaugment: A simple data augmentation method for automatic speech recognition,” *arXiv Prepr. arXiv1904.08779*, 2019.

[74] D. Pekar and R. Obradović, “C++ library for digital signal processing-slib,” in *Proc. Telfor 2001*, 2001.

[75] H. Ze, A. Senior, and M. Schuster, “Statistical parametric speech synthesis using deep neural networks,” in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, 2013, pp. 7962–7966.

[76] S. Suzić, T. Delić, V. Jovanović, M. Sečujski, D. Pekar, and V. Delić, “A comparison of multi-style DNN-based TTS approaches using small datasets,” in *MATEC Web of Conferences*, 2018, vol. 161, p. 3005.

Прилог 1

План третмана података

Назив пројекта/истраживања
Novel method for speaker adaptation in parametric speech synthesis **(Нова метода адаптације на говорника у параметарској синтези говора)**
Назив институције/институција у оквиру којих се спроводи истраживање
а) АлфаНум доо, Нови Сад **б) Speech Morphing System, Inc, Saj Jose, California** **в) Факултет техничких наука, Нови Сад**
Назив програма у оквиру ког се реализује истраживање
/
1. Опис података
1.1 Врста студије *Укратко описати тип студије у оквиру које се подаци прикупљају* **У питању су докторске студије, које су се одвијале у паралели са развојем комерцијалних прозивода за горе наведене компаније.**

1.2 Врсте података

а) квантитативни

б) квалитативни

1.3. Начин прикупљања података

а) анкете, упитници, тестови

б) клиничке процене, медицински записи, електронски здравствени записи

в) генотипови: навести врсту _____

г) административни подаци: навести врсту _____

д) узорци ткива: навести врсту_____

ђ) снимци, фотографије: навести врсту_____

е) текст, навести врсту ___ _____

ж) мапа, навести врсту _____

з) остало: мерење растојања између параметара (објективне мере)

1.3 Формат података, употребљене скале, количина података

1.3.1 Употребљени софтвер и формат датотеке:

а) Excel фајл, датотека (више њих)

b) SPSS фајл, датотека _____

c) PDF фајл, датотека _____

d) Текст фајл, датотека _____

e) JPG фајл, датотека _____

f) Остало, датотека _____

1.3.2. Број записа (код квантитативних података)

а) број варијабли: **у зависности од експеримента, мерено је између 2 и 7 параметара**

б) број мерења (испитаника, процена, снимака и сл.) **број испитаника се кретао од 20-30, а број снимака од 5-20**

1.3.3. Поновљена мерења

а) да

б) не

Уколико је одговор да, одговорити на следећа питања:

а) временски размак између поновљених мера је

_____

б) варијабле које се више пута мере односе се на

_____

в) нове верзије фајлова који садрже поновљена мерења су именоване као

_____

Напомене: _____

Да ли формати и софтвер омогућавају дељење и дугорочну валидност података?

а) Да

б) Не

Ако је одговор не, образложити

_____

_____

2. Прикупљање података

2.1 Методологија за прикупљање/генерисање података

У тестовима слушања користиле су се MOS (енгл. *Mean Opinion Score*) и MUSHRA (енгл. *MUltiple Stimuli with Hidden Reference and Anchor*) методе.

За мерење објективних мера користило се растојање између параметара синтетизованог и оригиналног говора. Мерени параметри су: мел кепстрали, основна учестаност, степен звучности, степен апериодичности по фреквенцијским опсезима.

2.1.1. У оквиру ког истраживачког нацрта су подаци прикупљени?

а) експеримент, навести тип: MOS, MUSHRA.

б) корелационо истраживање, навести тип: поређење наведених параметара.

ц) анализа текста, навести тип _____

д) остало, навести шта _____

2.1.2 Навести врсте мерних инструмената или стандарде података специфичних за одређену научну дисциплину (ако постоје).

У експериментима слушања су се користиле слушалице.

Формат аудио фајлова је био 22kHz, 16bit, PCM.

2.2 Квалитет података и стандарди

Табеле су у стандардном (Excel) формату.

2.2.1. Третман недостајућих података

а) Да ли матрица садржи недостајуће податке? Да **Не**

Ако је одговор да, одговорити на следећа питања:

а) Колики је број недостајућих података? _____

б) Да ли се кориснику матрице препоручује замена недостајућих података? Да Не

в) Ако је одговор да, навести сугестије за третман замене недостајућих података

_____

__

2.2.2. На који начин је контролисан квалитет података? Описати

Махом ручно, односно праћењем тока експеримента.

2.2.3. На који начин је извршена контрола уноса података у матрицу?

Софтвер прилагођен за ове сврхе је вршио ту функцију, уз накнадну ручну

контролу.

3. Третман података и пратећа документација

3.1. Третман и чување података

*3.1.1. Подаци ће бити депоновани у **компанијски** репозиторијум.*

3.1.2. URL адреса

https://drive.google.com/drive/folders/1CKyablYERuHKMiierDiP3gMEAsBzVIib?usp=sharing

3.1.3. DOI

_____

3.1.4. Да ли ће подаци бити у отвореном приступу?

а) ***Да***

б) *Да, али после ембарга који ће трајати до*

_____

в) *Не*

Ако је одговор не, навести разлог _____

3.1.5. Подаци неће бити депоновани у репозиторијум, али ће бити чувани.

Образложење

_____

__

_____

__

3.2 Метаподаци и документација података

3.2.1. Који стандард за метаподатке ће бити примењен?

Слободна форма у оквиру ецџел докумената.

3.2.1. Навести метаподатке на основу којих су подаци депоновани у репозиторијум.

У оквиру метаподатака је наведен број експеримената, субјеката и снимака који су оцењивани, као и њихове најважније карактеристике.

Ако је потребно, навести методе које се користе за преузимање података, аналитичке и процедуралне информације, њихово кодирање, детаљне описе варијабли, записа итд.

3.3 Стратегија и стандарди за чување података

3.3.1. До ког периода ће подаци бити чувани у репозиторијуму? неограничено

3.3.2. Да ли ће подаци бити депоновани под шифром? Да **Не**

3.3.3. Да ли ће шифра бити доступна одређеном кругу истраживача? Да Не

3.3.4. Да ли се подаци морају уклонити из отвореног приступа после извесног времена?

Да **Не**

Образложити

4. Безбедност података и заштита поверљивих информација

Овај одељак МОРА бити попуњен ако ваши подаци укључују личне податке који се односе на учеснике у истраживању. За друга истраживања треба такође размотрити заштиту и сигурност података.

4.1 Формални стандарди за сигурност информација/података

Истраживачи који спроводе испитивања с људима морају да се придржавају Закона о

заштити података о личности

(*https://www.paragraf.rs/propisi/zakon_o_zastiti_podataka_o_licnosti.html*) и одговарајућег институционалног кодекса о академском интегритету.

4.1.2. Да ли је истраживање одобрено од стране етичке комисије? Да **Не**

Ако је одговор Да, навести датум и назив етичке комисије која је одобрила истраживање

_____

__

4.1.2. Да ли подаци укључују личне податке учесника у истраживању? Да **Не**

Ако је одговор да, наведите на који начин сте осигурали поверљивост и сигурност информација везаних за испитанике:

а) Подаци нису у отвореном приступу

б) Подаци су анонимизирани

ц) Остало, навести шта

5. Доступност података

5.1. Подаци ће бити

а) јавно доступни

б) доступни само уском кругу истраживача у одређеној научној области

ц) затворени

Ако су подаци доступни само уском кругу истраживача, навести под којим условима могу да их користе:

Ако су подаци доступни само уском кругу истраживача, навести на који начин могу приступити подацима:

5.4. Навести лиценцу под којом ће прикупљени подаци бити архивирани.

1 - Ауторство

6. Улоге и одговорност

6.1. Навести име и презиме и мејл адресу власника (аутора) података

Дарко Пекар, pekard@gmail.com

6.2. Навести име и презиме и мејл адресу особе која одржава матрицу с подацима

Дарко Пекар, pekard@gmail.com

6.3. Навести име и презиме и мејл адресу особе која омогућује приступ подацима другим истраживачима

Дарко Пекар, pekard@gmail.com