Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Távközlési és Médiainformatikai Tanszék

Kis Kornél István

MÉLY NEURÁLIS HÁLÓZAT ALAPÚ GÉPI BESZÉDKELTÉS MAGYAR NYELVEN

KONZULENSEK

Dr. Tóth Bálint Pál

Dr. Németh Géza

BUDAPEST, 2016

Tartalomjegyzék

Köszönetnyilvánítás

Ezúton szeretnék köszönetet mondani konzulenseimnek, Dr. Tóth Bálint Pálnak és Dr. Németh Gézának a TDK dolgozat megvalósításához nyújtott szakmai és technikai segítségükért. A dolgozat nem jöhetett volna létre a konzultációk során és a kritikus pillanatokban nyújtott odafigyelésük és segítségük nélkül.

Összefoglaló

A gépi beszédkeltés kutatásában az elmúlt évtizedben a korábbi elemkiválasztásos (ún. unit selection) rendszerekkel szemben beszéd paramétereinek statisztika alapú modellezése előtérbe került. Mára már kiemelkedően népszerűvé váltak a mély neurális hálózatok, számos tudományterületen jelentős előrelépést hoztak a korábbi megoldásokhoz képest. Jelen TDK kutatómunka során a hazai és nemzetközi irodalomra támaszkodva mély neurális hálózat (Deep Neural Network, DNN) alapú gépi beszédkeltő rendszert dolgoztam ki, melynek működését a dolgozat során részletesen bemutatom.

Munkám során a szöveg fonetikus átirata és a beszédparaméterek közötti kapcsolatot DNN modellezi. Ehhez első lépésként a Budapesti Műszaki és Gazdaságtudományi Egyetem Távközlési és Médiainformatikai Tanszék Beszédtechnológia és Intelligens Interakciók Laboratóriumának korábbi rendszereiből és hang adatbázisaiból kiindulva a DNN tanításához szükséges tanító adatbázist létrehozó eljárás kidolgozására volt szükség. Ezek után a tanító adatbázist a nemzetközi irodalomra támaszkodva többféle neurális hálózati architektúrán tanítottam. Munkámban elsődlegesen a beszéd alapfrekvencia (f0) és az ún. spekrális paraméterek DNN-ekkel történő modellezését vizsgálom magyar nyelven. A számos lehetséges hiperparaméter-beállítás között az optimálishoz minél közelebb lévő beállítást sztochasztikus elven működő hiperparaméter-optimalizással kerestem. Mind az alaprendszer kidolgozását, mind pedig a hiperparaméter optimalizálást kísérleti mintarendszerben demonstrálom.

Dolgozatomban a gépi beszédkeltő rendszer alapvető komponenseinek működésén és ezek kapcsolódásán kívül nagy hangsúlyt fektettem arra, hogy a jelenlegi beszédkeltő rendszerek gyengeségeit - például a hosszabb szövegek generálásakor jelentkező zavaró monotonitást – minél inkább mérsékeljem. Ennek megvalósítására egy speciális – ún. aggregált (ensemble) architektúrát alkalmaztam a mély neurális hálózatok tervezésekor. A kutatómunka ezen részét konzulenseimmel konferenciacikk formájában a SPECOM 2016 nemzetközi konferencián publikáltuk, melyet előadás formájában is bemutattam angol nyelven.

Jelen dolgozat rávilágít, hogy a DNN-el történő beszédparaméter modellezés – optimálishoz közeli hiperparaméterek használata esetén - érdemleges előrelépést nyújthat a korábbi megoldásokhoz képest.

Abstract

In the recent years, a consistent shift of focus is apparent in the text-to-speech research leading from unit selection systems to statistical parametric speech synthesis. In these days, the research area of neural networks has seen a previously unprecedented popularity, revolutionizing several other fields of science. In this TDK paper, based on Hungarian and international scientific literature, a deep neural network based text-to-speech system (TTS) is presented and described in greater detail.

In this approach, the relationship between the phonetic transcription of the input text and speech parameters is modelled by a DNN. The first step to build an efficient DNN model was to create a training database for the neural network. Based on previous solutions and speech corpora of the Speech Technology and Smart Interactions Laboratory, Department of Telecommunications and Media Informatics of the Budapest University of Technology and Economics, this data was used to train several state of the art neural network architectures. In the process of finding the best network architecture and hyperparameters for our purposes and transforming the training database to a representation which is best for training, several theoretical questions – sometimes from outside the field of TTS – were considered. In this TDK paper, both the theoretical and implementation elements are discussed in detail. The primary speech parameters under consideration is the fundamental frequency of voiced speech (f0) and the speech spectral parameters. To further optimize the search for better solutions, a stochastic process based hyperparameter-optimization method was also implemented. The method is also described in this TDK paper.

During my research, a great deal of effort was put into reducing the performance-degrading effect of bad prosodic stress estimation. This part of the TDK research has also appeared in the form of a lecture and a conference paper – authored by my thesis advisors and me – at SPECOM 2016 conference.

The goal of this TDK paper is to reveal that – given a near-optimal setup of hyperparameters – enhancements can be achieved over today's other competing TTS technologies.

1 Bevezető

Jelen TDK dolgozat fő célja, hogy bemutasson egy mély neurális hálózatokon alapuló kísérleti rendszert, mely alkalmas az emberi beszéd paraméterfolyamjainak a becslésére és ezáltal gépi beszédet generálni. A kutatómunka célja a korábbi gépi szövegfelolvasó eljárásoknál természetesebb hangzású gépi beszéd előállítása volt.

A problémakör igen aktuális, hiszen a mesterséges beszédkeltésre egyre növekvő igény mutatkozik modern digitális világunkban. Az első hallásra logikusnak tűnő célcsoport, a vakok és gyengénlátók kiszolgálása mellett napjainkban – például a személyi asszisztens rendszerek részeként - az átlagfelhasználó is sokféle módon hasznosíthatja a gépi beszédet generáló rendszereket (pl. IBM Watson, Apple Siri, Microsoft Cortana). A problémakör igen szerteágazó, hiszen a beszédnek érthetőnek, emberi hangzásúnak kell lennie, felhasználási helytől függően rugalmasnak, esetleg a többnyelvűség is elvárt. A tématerület kutatóit a jelenleg elérhető műszaki lehetőségek határai optimalizálási feladatok sora elé állítják. Kiemelkedő, ipari rendszerekben is megbízható minőséget csak kötött témakör esetén lehet biztosítani (ekkor alkalmazhatók az ún. elemkiválasztásos rendszerek, például időjárás-előrejelzés felolvasók), más területeken a rendkívül széles szókincs és nagyfokú rugalmasság biztosítása okoz kihívást a kielégítő beszédminőség mellett (jó példa ilyen alkalmazásra a már említett okostelefonokon található személyi asszisztensek). Sokszor kompromisszumokat kell kötni a rendszerek számítás- és helyigénye és a minőség között is. A későbbiekben többször említett rejtett Markov modellen [1] alapuló rendszereknek nagyobb helyigénye van a régebbi, diádos megoldásokhoz képest, jobb minőséget is képes előállítani, de még mindig jellemző, hogy a beszéd sokszor nem elég természetes hangzású. Bár a beszédhangzás rengeteget fejlődött, még ma is nagy kihívás hosszú szövegek élvezhető felolvasása kiterjedt szókincs mellett. Ezen problémák orvoslása esetén várhatóan tovább javulna az ember-gép kommunikáció minősége, mely végső soron a következő lépcsőfokot jelenthetné a kognitív számítástechnika fejlődésében.

A jelen dolgozatban tárgyalt kísérleti mintarendszer a beszédképző rendszerek egyik legmodernebb csoportjába, a statisztikai parametrikus beszédszintetizáló rendszerek közé tartozik. A rendszer újszerűségét az adja, hogy a beszédparaméterek mindegyikét mély neurális hálózatokkal (Deep Neural Network, DNN) [2] modellezzük. Magyarországon a gépi

beszédkeltés céljára ismereteim alapján az elsők között végzek kutatást mély neurális hálózatokkal. A DNN-ek és azok működését biztosító kísérleti mintarendszer elemeit a dolgozat későbbi fejezeteiben részletes bemutatom. Az elvégzett munkám fő komponensei a következő felsorolásban olvashatók:

1. A gépi beszéd előállításához szükséges paraméterfolyamok (alapfrekvencia, spektrális- és időzítési paraméterek) előállítása előtanított mély neurális hálózatokkal
2. A paraméterfolyamok becslésére szolgáló mély neurális hálózatok tanításának lépései (ez magában foglalja az adatelőkészítést, a tanítások elvégzését, és a hiperparaméter-optimalizálást)
3. A gépi beszéd természetességének további növelését szolgáló, a beszédhangsúlyok jobb becslését lehetővé tevő rendszer elemeinek bemutatása
4. Az eredmények részletes kiértékelése mind objektív, mind pedig szubjektív eszközökkel.

Dolgozatom első fejezete megadja a munka értelmezéséhez szükséges beszédtechnológiai alapokat, valamint bemutatja a kutatómunka során használt műszaki modelleket. A második fejezetben bemutatásra kerülnek a gépi tanulás legfontosabb elméleti jellegzetességei, az alapvető hálózati struktúrák, különös tekintettel a munka során használt architektúrákra és tanító algoritmusokra. A harmadik fejezet a kutatómunka indulásakor a rendelkezésre álló, a BME TMIT Beszédtechnológia és Intelligens Interakciók Laboratóriumának (a továbbiakban: TMIT Speechlab) birtokában lévő beszédparaméter- és hangadatbázist mutatja be. A negyedik fejezet bemutatja a probléma megoldására felhasznált tanító adatbázis létrehozását, a vizsgált neurális hálózati architektúrákat, a tanítás folyamatát, és a hiperparaméter-optimalizálást. Az ötödik fejezet beszámol az eddig elért eredményekről, és bemutatja azokat a szubjektív és objektív teszteket, melyekkel a rendszer eredményességét teszteltem. Az utolsó fejezet a jelenleg még folyamatban lévő fejlesztéseket írja le.

1.1 A beszédképző rendszerek fejlődése

A beszédképző rendszerek – jelenlegi formájuk és teljesítményük eléréséig – rengeteg fejlődési lépcsőn mentek keresztül. A felhasználók részéről folyamatos igény jelentkezik az emberi beszéd analízisére, feldolgozására, és szükség esetén pedig a jó minőségű (emberire emlékeztető) gépi beszédkeltésre. Már jóval a digitális kor eljövetele előtt készültek olyan mechanikus, és később analóg elektronikus szerkezetek, amelyek képesek voltak gépi beszédkeltésre. A XX. század első felében a legsikeresebb rendszerek közé sorolható például Homer Dudley „VODER" nevű gépe (1939) [3], vagy az 40-es években Dr. Franklin S. Cooper által készített szerkezet [4], amely tulajdonképpen egy kézi paramétervezérlésű beszédkódoló segítségével állított elő hangokat. Az első, több nyelven működő berendezés az 1980-as években készült el Dennis Klatt kutató jóvoltából [5].

Az első digitális beszédképzők rendkívül gépies hangzásúak voltak, a beszéd sokszor akadozott, és nagyon nehezen volt érthető. Az elmúlt évtizedekben a beszéd érthetősége folyamatosan javult, ez főképp a számítógépek adatfeldolgozási képességeinek és a fejlettebb matematikai modelleknek köszönhető. A tökéletesség azonban még messze van: a jelenlegi beszédképző rendszerek által szolgáltatott hangok az emberi fül számára még mindig megkülönböztethetők a valóságos emberi beszédtől általános tématerület esetén.

A mai rendszerek többféle megközelítést is alkalmazhatnak, ezek például a diád- és triád alapú beszédszintézis, vagy a nagyméretű adatbázisokon alapuló korpuszos beszédszintézis. A beszédkeltő rendszerek legfrissebb ága a statisztikai paramterikus szövegfelolvasók megjelenése volt mintegy 10 évvel ezelőtt. Az első sikereket a kutatók a rejtett Markov modelleken alapuló rendszerekkel érték el (még pár éve ezek a modellek működtették a legfejlettebb általános célú statisztikai parametrikus szövegfelolvasás alapú „beszéd alkalmazásokat") [1]. A rejtett Markov modell alapú rendszerek azonban számos nehézséggel is küzdenek: egy bizonyos pontosság elérése után csak igen sok új beszédminta bevitelével fokozható tovább a teljesítmény (ha egyáltalán ez lehetséges), továbbá a rejtett Markov modellben [2] használt döntési fák pedig nem alkalmasak a összetett környezetfüggőségeket együttes modellezésére. A gépi tanulás fejlődésével lehetőség nyílt egy alternatív megközelítésre, a mély neurális hálózatok felhasználására [2].

1.2 A beszéd elemi paraméterei

A statisztikai parametrikus beszédszintézisben a beszédkódolókban használt megoldáshoz hasonlóan paraméterfolyamokra bontják a beszédet. A beszéd így tulajdonképpen időben eltolt keretek („frame”-ek) egymás után helyezéséből áll. Egy-egy kerethez tartozik az adott paraméterfolyam egy-egy eleme. Ezek a paraméterfolyamok a kutatómunka során használt modellben az alapfrekvenciát (f0) és az adott beszélőre jellemző spektrális paramétereket jelenti (például az ún. mel-kepsztrális felbontás). Szintézis időben szükségünk van még az ún. időzítési paraméterekre is, ezek határozzák meg a beszédkeltés során, hogy az egymást követő hangkarakterek milyen hosszúak és milyen ütemben követik egymást. Ezen időzítési paramétereket egy kisebb, külön neurális hálózattal modelleztem.

1.2.1. Az alapfrekvencia függvény

Az alapfrekvencia (fundamental frequency, f0) a beszéd bonyolult, adott beszélőre jellemző nem folytonos függvénye. Az f0 a vizsgált beszélő egyedi hangmagassága mellett függ a beszélő nemétől, életkorától, sőt még olyan külső paraméterektől is, mint például a vizsgálat ideje [6,36]. (A reggel felvett hangfelvételeken jellemzően alacsonyabb f0 értékeket mérhetünk, mint a nap későbbi részén felvett hanganyagoknál). Ilyen bonyolult ok-okozati függés modellezése nehéz feladat mind a hagyományosabb modellekkel, mind pedig a neurális hálózatok felhasználásával.

A neurális hálózatok esetében a modellezni kívánt függvény szakadásai jelentenek elméleti szempontból leküzdendő kihívást. Az alapfrekvencia értékek nemcsak hogy nem alkotnak minden mondat esetén folytonos függvényt, hanem az általam használt modellben zöngétlen hangkarakterek kiejtésénél az általam használt modellben nem is definiáltak. Az ilyen hangkarakterek gerjesztése itt zajszerű, ebből következően nincs egy jól definiálható alapharmonikusa. Ahhoz, hogy eredményesebben modellezzük az ilyen függvényt, a tanító adatbázis elkészítése során módosításokat, például interpolációt érdemes alkalmaznunk az

adatsorok egyes részein. Az interpoláció után bevezetünk egy új változót, az ún. zöngés/zöngétlen bitet (Voiced/Unvoiced bit). A neurális hálózatnak kimenetként ezen bináris változót is modelleznie kell, tehát el kell döntenie, hogy az adott minta zöngés vagy zöngétlen hangkaraktertől származik-e. A neurális hálózat a modellezni kívánt függvény vagy más összefüggés mintázatait numerikus értékekké alakítja, és saját, iteratív úton kialakított belső reprezentációjában tárolja. Ebből következően a tanító adatbázis minden elemének tartalmaznia kell egy definiált értéket, hiszen a „nem definiált érték" számokkal nem reprezentálható. Erről, és ehhez hasonló problémákról az előfeldolgozás alfejezetben részletesebben lesz szó. A neurális hálózatnak ebből a részből tehát egy 2 db egydimenziós függvényt kell modelleznie, ami 2 kimenetet jelent (f0, zöngés-zöngétlen flag).

1.2.2. A spektrális paraméterek

Az emberi beszéd az alapfrekvencia-függvény mellett a magasabb frekvenciákon is jelentős energia és információtartalmat hordoz [37,38]. Ezen egyedi hatások minél pontosabb megragadása, illetve a minél precízebb és tömörebb reprezentációk készítése jelenleg is aktív kutatási terület. Jelen kutatómunka során az emberi beszéd hullámformájának mélyebb elemzésével nem foglalkoztam, a használt mély neurális hálózatok egy, a spektrális tartományt leíró, gyakran használt modelljét használtam. A modell neve LPC (linear predictive code). Ennek mélyebb bemutatása a dolgozatnak nem témája, azonban érdemes megjegyezni, hogy a modell változtathatóan mintegy 12-48 dimenziójú paraméterfolyammal dolgozik, melyből én a 24+1 dimenziós felbontást használtam. További technikai részlet, hogy az LPC paraméterek közvetlen formájukban a zajra különösen érzékenyek, ezért a munka során egy további transzformációt hajtottam végre a paraméterfolyamon. Ennek az adatreprezentációnak a neve LSP (Linear Spectral Pairs). Az LSP-t kimondottan az LPC paraméterek valós telekommunikációs csatornán történő átvitelére fejlesztették ki [40]. A neurális hálózati tanítás is közvetlenül az LSP paramétereken történik, mely gépi tanulás szempontjából is sokkal „stabilabb", mint az LPC. Az eredmény LSP paraméterfolyamokat pedig a használat előtt visszaátalakítjuk LPC paraméterekké. A neurális hálózatnak ebből a szakaszból 24+1 darab paramétert kell

modelleznie, ami az előző, alapfrekvencia szakaszban olvasott két paraméterrel együtt 27 paramétert jelent összesen minden egyes keret esetén.

1.2.3. Az időzítési paraméterek

Az alapfrekvencia és a spektrális paraméterek modellezése mellett a beszéd gépi előállítása során szükségünk van ún. időzítési paraméterekre. Ezek adják meg a beszédkódolónak, hogy egy adott hangkarakter mikor kezdődik, és mikor végződik. Mivel a hangkarakterek hosszúsága az alapfrekvenciát és a spektrális paramétereket modellező neurális hálózatban bemenetként szerepel (a részletekért lásd későbbi fejezetek), nem lehetséges ezen időzítési paramétereket ugyanebben a hálózatban modellezni. Szükség van egy kisebb, de hasonló felépítésű mély neurális hálózatra, amely ezeket az időzítési paramétereket előállítja. Az időzítési adatok is részben beszélő függők.

1.3 Beszédhangsúlyok

A beszédhangsúlyok helyes modellezése nagy jelentőséggel bír a természetesnek hangzó gépi beszéd előállításánál. A jelenlegi általános szövegfelolvasó rendszereink mindegyikében észlelhető a hosszabb szövegek esetén monotonitás, amely a felhasználói élményt nagymértékben ronthatja. A beszédhangsúly és a dinamika javítására többféle lehetőség kínálkozik. A kutatómunka során használt beszédhang-adatbázis (lásd 3. fejezet) fonetikus átiratában csupán egy numerikus bemenet jelképezte a hangsúlyokat. Ez az egy bemenet az összes bemenő adat nagy számához képest a tapasztalatok szerint nem ad elég figyelmet eme fontos beszédparaméternek.

Mivel a hangsúly megjelenítése az alapfrekvencia függvényen keresztül történik, szükség volt egy olyan megoldás kidolgozására, mely képes a beszédhangsúlynak, mint bemenetnek nagyobb jelentőséget adni a teljes rendszeren belül. A későbbi fejezetekben (aggregált modell néven) bemutatásra kerül a mintarendszer egy változata, mely a beszédhangsúly bemenetnek erőteljesebb szerepet ad az alapfrekvencia-függvény becslés során. Ennek hatására a hangsúly-

bemenetre jobban érzékeny becsült függvények keletkeztek a neurális hálózat kimenetén, és ez hallható különbséget eredményezett a szubjektív, meghallgatásos tesztek során is. A mintarendszer ezen része a SPECOM 2016 nemzetközi konferencián konferenciacikk és előadás formájában is be lett mutatva. A konferenciacikk elkészítésében társszerzőként vettem részt konzulenseim illetve Dr. Szaszák György mellett. Az eredményeket a konferencián 15 perces előadás formájában mutattam be angol nyelven.

A teljes kutatómunka kivonata 45 perces angol nyelvű előadás formájában is bemutatásra került a „6th Deep Learning Meetup in Vienna” elnevezésű szakmai rendezvényen Bécsben, amelyen meghívott előadóként vettem részt 2016 októberében.

2 Neurális hálózatok tervezése és használata

Ez a fejezet tárgyalja a neurális hálózatok működésének megértéséhez szükséges nélkülözhetetlen alapismereteket, különös figyelmet szentelve a szerteágazó tudományterületen belül a TDK dolgozat szempontjából releváns részeknek. Szó lesz a neurális hálózatok alkalmazásának rövid történetéről, az alkalmazott technológiákról, az általam alkalmazott konkrét architektúrákról és a tanítás során alkalmazott optimalizálásokról.

2.1 A neurális hálózat alapfogalmai és hálózati struktúrák

A neurális hálózatok tudománya sokéves múltra tekint vissza [7]. A mérnökök szerettek volna létrehozni tetszőleges folytonos függvény becslésére alkalmas modelleket. Ezen neurális hálózatok működése alapvetően eltér a hagyományos hálózatoktól. Közös, és egyben általános jellemzőjük, hogy a neurális rendszerek a megoldani kívánt feladatot nem valamilyen belső, tudatosan tervezett működés (pl.: állapotgép), szabályok segítségével valósítják meg, hanem a környezetből vett minták értelmezése révén. A neurális hálózat alapvető jellemzője tehát, hogy bemenetére kerülő mintázatok alapján valamilyen tanulási stratégia segítségével képes a belső paramétereit a kitűzött feladat minél jobb megoldása érdekében "jó irányba" módosítani [8].

Egy gépi tanulással operáló algoritmus tervezése során nincs szükség az adott folyamat, vagy jelenség teljes mértékű ismeretére. Ez a megközelítés nagy segítség olyan, nehezen megfogható problémák esetén, ahol a feladat explicit leírása valamilyen okból kifolyólag nem lehetséges. Ezen okok lehetnek például a túlzott erőforrásigény, a modellezni kívánt rendszer komplex volta. Jellemzően ilyen problémák például az explicit képfelismerési feladatok (például arcok, alakok, tárgyak felismerése képekről vagy videóról). Könnyen belátható, hogy például egy általános alakfelismerési probléma nehezen definiálható a hagyományos módon, hiszen a bemeneti képek igen sokfélék lehetnek. A gépi tanuló rendszerek fejlesztési lépései is eltérnek: a legfontosabb különbség, hogy az architektúra kiválasztása után a rendszer nem lesz automatikusan alkalmas az adott feladat megoldására. Ahhoz, hogy a hálózat ténylegesen el tudja

végezni a kitűzött feladatot, először be kell tanítani. Ennek során a hálózatot kapcsolatba hozzuk példa bemenetekkel (nemellenőrzött tanulás), vagy bemenet/kimenet párokkal (ellenőrzött tanulás). A hálózaton a példa adatsorokat végigfuttatva, az a nagyszámú belső paramétereit a látott adatoknak megfelelően alakítja, ekkor egy un. „belső reprezentációját" készítve el a feladatnak. Amennyiben a tanítás sikeres volt, ez a belső reprezentáció jól modellezi a feladatot, és képessé teszi a hálózatot arra, hogy valós bemenetekre is kis hibával adjon közelítést a tesztelés, és a tényleges használat közben.

A neurális hálózatokkal kapcsolatos kutatások az elmúlt években ismét erős lendületet vettek. Minden eddiginél nagyobb sebességgel születnek az új eredmények a tudományterületen. A gépi tanulás legújabb tudományos forradalmát (többek között) Hinton 2006-ban megírt cikke [9] gyorsította fel, mert hatékony eljárást adott több rejtett rétegű hálózatok hatékony tanítására. A fejlődést nagyban segíti továbbá, hogy elterjedtek a modern GPU-k (Graphics Processing Unit, magyarul grafikai processzor), melyek jelentős mértékben, akár nagyságrendekkel is felgyorsíthatják a számításokat. Így olyan problémák is kezelhetőek lesznek, melyekre korábban nem volt elegendő számítási kapacitás. Érdemes még megemlíteni az egyre nagyobb mennyiségben elérhetővé vált tárkapacitást, és a nagyméretű, részletes adatbázisok meglétét is. A neurális hálózatokban nagy potenciál rejtőzik, különösen azért, mert folyamatosan javuló eredményeket produkálnak, vagy legalábbis ígérnek tipikusan olyan területeken (például általános képfeldolgozás, bonyolult osztályzási problémák), ahol a hagyományos paradigmák alapján működő, korábbi rendszerek gyengén teljesítenek. A tudományterületen jelenleg azonban nagy kihívás igazán hatékony rendszert építeni, hiszen kevés az olyan analitikus eredmény, mely a gyakorlatban is jól használhatóan adna támpontot ilyen rendszerek ún. hiperparamétereinek a beállításához. A gyakorlatban ezért a legtöbb esetben az empirikus módszer, a már létező rendszerek vizsgálata képezi az újabb hálózatok kiindulását, de további, heurisztikus alapon működő eljárások is használatban vannak. Szerencsére már számos tanulmány készült a különböző rendszerek alkalmazhatóságát illetően, így hasonló problémákat vizsgálva támpontot kaphatunk a tervezés kiindulásánál. További előny, hogy matematikailag bizonyított a neurális hálózat folytonos függvényekre vonatkozó univerzális approximátor képessége. Ez azt jelenti, hogy az elérhető eredményeket a gyakorlatban az adatbázisunk minősége, és a tanítási eljárásunk hatékonysága korlátozza. Az univerzális approximátor képességhez a hálózatnak legalább egy nemlineáris függvényt felhasználó réteggel kell rendelkeznie. A gyakorlati problémák

megoldásánál általában több rejtett réteget alkalmazunk, és többségében a rejtett rétegek mellett a kimeneti réteget is nemlineáris neuronokkal építjük fel. Ezek alapján a hálózat elméletben felhasználható tetszőleges osztályzási és regressziós problémákhoz is (például címkékkel ellátott kategóriákba sorolás, vagy folytonos függvények modellezése).

A neurális hálózatok egyik fő csoportját az ún. előrecsatolt hálózatok („feedforward network") képzik. Ezek gráfelméleti reprezentációja nem tartalmaz irányított kört. A TDK dolgozatban megvalósított TTS rendszer során kizárólag előrecsatolt, több rejtett réteget tartalmazó neurális hálózatok kerültek felhasználásra. Mély neurális hálózatról jellemzően akkor beszélünk, ha hálózatunk több, a külvilággal közvetlen kapcsolatban nem álló rejtett réteget tartalmaz.

Az előrecsatolt hálózatok egyik leggyakrabban használt alkategóriája az ún. Multi-Layer-Perceptron (MLP) [35]. A hálózat részletes ismertetését terjedelmi okokból mellőzöm, itt csupán néhány, a munkám szempontjából fontosabb gyakorlati részletre térek ki.

A MLP elemi műveletvégzője általában tartalmaz egy nemlineáris függvényt. A nemlineáris függvény sokféle lehet, régebben valamilyen küszöbfüggvény jellegű összefüggést használnak, például előjel ($sgn(x)$), hiperbolikus tangens ($tanh(x)$), vagy ún. szigmoid függvényt. Az elmúlt néhány évben népszerűvé vált az ún. Rectified Linear Unit (ReLu) [10] alkalmazása is. A ReLu gyakorlati problémák esetén a tapasztalatok szerint nagyobb pontosságot eredményez [11], így én is ezt használtam.

Az MLP architektúra építése során a nemlinearitás mellett számos egyéb szabad paramétert módosíthatunk céljainknak megfelelően. Az első ilyen a hálózat mérete (ebbe a rejtett rétegek száma is beletartozik). Az egy rejtett rétegen belüli neuronok számának eldöntése alapvető fontosságú kérdés (a bemeneti és kimeneti neuronok számát meghatározza a probléma dimenziószáma, például egy 3 bemenetű, 2 kimenetű függvény közelítésekor 3 input neuron és 2 output neuron kell). Ha túl kevés rejtett rétegbeli neuront használunk, a hálózatunk nem fog megfelelően tanulni, túlzottan sok használata esetén pedig számítási kapacitást és időt pazarolunk el. Általánosságban kijelenthető, hogy bonyolultabb, illetve nagyobb mintaszámmal rendelkező problémák több rejtett rétegbeli neuront igényelnek, de ennél több konkrétum sajnos nem adható.

2.2 A hálózatok tanítása

Az ebben a fejezetben olvasható információk elsősorban a fontosabb, műszaki tervezésnél használt szempontokat, irányelveket tartalmazzák. Jelen TDK dolgozat forrásjegyzékében található irodalmak sokkal részletesebben és matematikai precizitással tárgyalnak egy-egy részterületet vagy problémakört.

Munkám során az MLP tanítása ellenőrzött módon történik. Az ellenőrzött tanítás során a tervező rendelkezésére áll egy bemenet-kimenet párokból álló mintakészlet, amely a valós, modellezni kívánt rendszer alapján készült. A tanítás során, miután meghatároztuk a hálózat méretét, a súlyok kezdeti értékeinek beállítása a következő lépés. Itt sok esetben véletlen inicializálást választunk, figyelve arra, hogy a nemlineáris függvény értékkészletével és az bemeneti és kimeneti adatokkal összemérhető nagyságú értékek legyenek a súlyok kezdeti értékei. Amikor a tanító mintáinkat végigfuttatjuk a hálózaton, azok kimeneteket generálnak (becsült kimenet). Ezeket a kimeneteket összehasonlítjuk az elvárt, eredeti rendszerről vett referencia kimenetekkel, és egy hibafüggvényt képzünk. A hibafüggvény megválasztása alapvetően befolyásolja a hálózat viselkedését. MLP esetén a hiba-visszaterjesztéses algoritmus sajátosságai miatt népszerű választás a négyzetes hibafüggvény, azonban vannak más lehetőségek is. Négyzetes hibafüggvény használata esetén is csak az biztos, hogy a hiba a tanítandó paraméterek folytonos, differenciálható függvénye (azaz a „hibafelület” nem kvadratikus). Ezen tulajdonság miatt megvan a veszélye annak, hogy a hibaminimalizálás során „leragadunk” egy lokális minimumhelyen, ahonnan a rendszer nem képes kimozdulni. Ennek elkerülésére többféle technika létezik, amelyek egy részéről a következő alfejezetekben lesz szó.

A hiba-visszaterjesztéses algoritmus teljes bemutatása meghaladja a jelen dolgozat kereteit. A teljes leírás helyett inkább röviden összegzem a működését. Az algoritmus egy iteratív eljárás, amely minden egyes tanító minta rendszeren való átfutása után (online mód) vagy az összes tanító minta egyszeri lefuttatása után (batch update mód) a súlyfüggvény értékeinek módosításával csökkenti a hibát. A problémát az jelenti, hogy tudnunk kell, hogy a detektált kimenetben melyik súlyérték mekkora részben és milyen „irányban” vesz részt. (Erre ad választ az algoritmus). Az algoritmus minden iterációban végig futtatja a tanító mintákat a rendszeren, elmenti a hibát, és annak megfelelően módosítja a súlyértékeket. Optimális esetben a hiba kezdetben nagyon gyorsan, majd

később lassabban, de csökken az elméleti zéróig. Természetesen a gyakorlatban ez nem így valósul meg; semmi sem garantálja ugyanis, hogy a folyamat monoton, konvergens, így nem megfelelő beállítások esetén a hiba nem hogy nem csökken, de akár végtelenhez is tarthat.

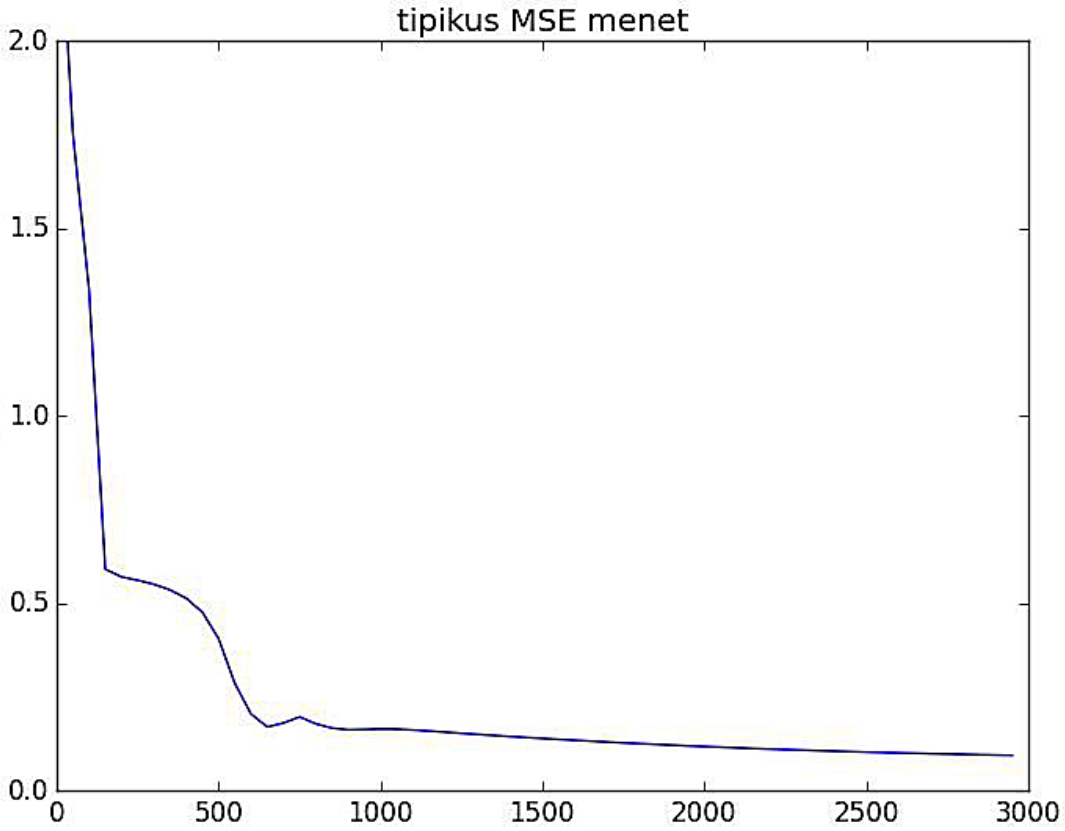
Az algoritmus egy belső paramétere az ún. tanulási ráta (μ), amely egy szorzó tagként szerepel a súlymódosításkor. Vázlatosan ez a következőt jelenti:

$$v_{k+1} = -\mu * \nabla f(W_{k+1}) + \eta * \Delta v_k \quad (1)$$

$$W_{k+1} = W_k + v_{k+1} \quad (2)$$

ahol v_k a súlymódosítás értéke az adott iterációban, W_k a jelenlegi súlyérték, W_{k+1} pedig a következő iteráció súlyértéke, és μ a tanulási ráta, $\nabla f(\dots)$ pedig a gradiensképzés operátora. A tanulási rátát pozitív (nulla és egy közötti valós) számként definiáltuk, ezért szükséges a negatív előjel a képletben. Az utolsó tag a momentum módszer alkalmazásakor számít, itt η a momentum módszer paramétere. A tanulási ráta egyszerűbb esetekben lehet konstans a teljes tanulás alatt, de számos adaptív tanulási ráta módosító stratégia is létezik [15,16,17]. Nagyobb tanulási ráta gyorsabb tanulást eredményezhet, azonban ha túl nagyra választjuk, a konvergencia elvész. A hibafelület lokális minimumaiból való kimozdulást segítheti elsősorban a momentum módszer alkalmazása is, mely tehetetlenséget, és egy további szabad paramétert tesz a rendszerbe. A hiba-visszaterjesztésről részletesen [12]-ben lehet olvasni.

A gyakorlati alkalmazásoknál (különösen kevesebb neuron esetén) a hiba csökkenése kezdetben nagyon gyors, majd ez tempó lassul. Ezt szemléletesen mutatja az 1. ábra. A tanítás során általában meghatározunk egy hibafüggvény minimum értéket, amit a folyamat sikeres tanítás esetén elér. Itt érdemes leállítani az algoritmust, mert amennyiben sokáig még tovább folytatódik a tanítás, a „túltanulás” jelensége következik be, és a rendszer teljesítménye a valós adatokon az optimálishoz képest romlani fog.



1. ábra – tipikus hibafüggvényérték-menet a tanítás során. x tengely: iterációk száma, y tengely – középhiba (mean square error, MSE)

A tanítás során el kell döntenünk hogy milyen módon csoportosítjuk a rendelkezésünkre álló tanító mintákat. Egyszerűbb esetben két csoport lesz, a tanító és a validációs adatok. Bonyolultabb, gyakorlati alkalmazásoknál – jelen TDK dolgozat esetén is - egy ezektől teljesen külön kezelt, a tanítás végén végső ellenőrzésre használt ún. teszt adatbázist is használnak. A tanító halmazban lévőkön futtatjuk a hiba-visszaterjesztéses algoritmust, a maradék adatokon ellenőrizzük a pontosságot. Fontos látni, hogy ez az adat szelektáló kérdés is döntő lehet: ha túl kevés a tanító adat, nem fog megfelelően tanulni a rendszer, ha pedig túl kevés az ellenőrző adat, nem fogjuk látni a folyamat sikerességét. Ez a kompromisszum különösen olyan alkalmazásoknál éles, ahol nehézkes vagy drága a mintákhoz való hozzájutás. Gyakorlatban működő szabályként általában az adatok mintegy 50-80%-át szokták a tanító halmazba, a maradékot pedig a validációs és teszt halmazba tenni. A tanítástól független vizsgálatokhoz teszt adatbázist használunk, ezt a

tanító adatoktól elkülönítetten kell kezelni, így a hálózat paramétereinek végső beállítása után objektíven tudunk tesztelni.

2.3 A tanítás hatékonyságának javítása

A neurális hálózatok tanításának egyik nehézségét a matematikai háttér hiánya okozza [13]. Sokszor nem állnak rendelkezésre olyan tételek, amelyek a gyakorlatban is jól használhatóan adnának felső korlátot egy probléma megoldásához szükséges hálózatméretre, vagy tanítási hiperparaméterre. Ebből kifolyólag a tanítás során nagy szerepe van a tervezők korábbi tapasztalatainak, illetve a hasonló problémák esetén már működő eljárások vizsgálatának. A témakörrel hosszú ideje foglalkozó szakértők sok, a gyakorlatban működő eljárást dolgoztak ki, itt ezek közül ismertetek néhányat. Ezen eljárások többségének működése a témakörben mélyebb ismeretekkel rendelkezők számára intuitív módon végig követhető, annak ellenére, hogy ezek nem mindegyike formális matematikai precízséggel megalkotott módszer.

Optimalizációs stratégiák

[13]-ban is olvasható, hogy a gradiens módszer egy minimumhely kereső eljárás. Mint minden ilyen, ez is szenved a lokális minimum elérésének problémájától. A legegyszerűbb problémáktól eltekintve a sokdimenziós hibafelület változatos alakú, így ahhoz, hogy ténylegesen megállapodjunk a globális minimumhoz közel, időnként a hibafelületen a nagyobb hiba irányába („felfelé”) is el kell mozdulnunk. A gradiens módszer alap esetben erre nem képes, hanem leragadhat egy nem szükségszerűen optimális lokális minimumban. Az egyik módszer a leragadás kiküszöbölésére a momentum módszer alkalmazása. Ha a gradiens gyakran vált előjelet, a momentum módszer képes kisimítani az utat a vélelmezett minimum felé. A momentum módszer a tanulási rátához hasonló, 0 és 1 között állítható paraméterrel rendelkezik. Nagy momentum alkalmazása esetén a tapasztalatok szerint érdemes a tanulási rátát kisebbre venni, különben gyorsan túlhaladhatunk a minimumon.

A momentum módszerhez képest további optimalizálást tesz lehetővé Nesterov megoldása [14]. A Nesterov-momentumnak is nevezett módszer módosítja a (1)-es képletet. A Nesterov-momemtum alkalmazása esetén a súlyfrissítési képlet a következő:

$$v_{k+1} = -\mu * \nabla f(W_{k+1} + \eta * \Delta v_k) + \eta * \Delta v_k \quad (3)$$

$$W_{k+1} = W_k + v_{k+1} \quad (4)$$

A módosítás értelmét szemléletesen úgy lehetne kifejezni, hogy amennyiben nagy (egyhez közeli) momentum paraméter használata esetén a súlyfrissítési lépésekben ez a tag túlzottan domináns lesz, azzal végső soron eltávolodunk eredeti céljainktól. A Nesterov-momentum alkalmazása ezen hatás ellen dolgozik, ami várhatóan javítja a végső eredményt.

Léteznek másfajta optimalizációs módszerek is. Az elmúlt néhány évben elterjedtek a olyan megoldások, melyek a tanulási rátát automatikusan állítják. Ilyen módszer például az ún. Adagrad [15], az RMSProp [16], vagy az Adam módszer [17]. Ezekkel sok esetben jobb eredményeket lehet elérni, mint a manuális beállításokkal. Másik előny, hogy itt nincs szükség a tanulási ráta kézi beállítására, mert ezek a módszerek maguknak állítják be a tanulási rátát. A javulást azonban nem minden problématípus esetén lehet egyértelműen kimutatni.

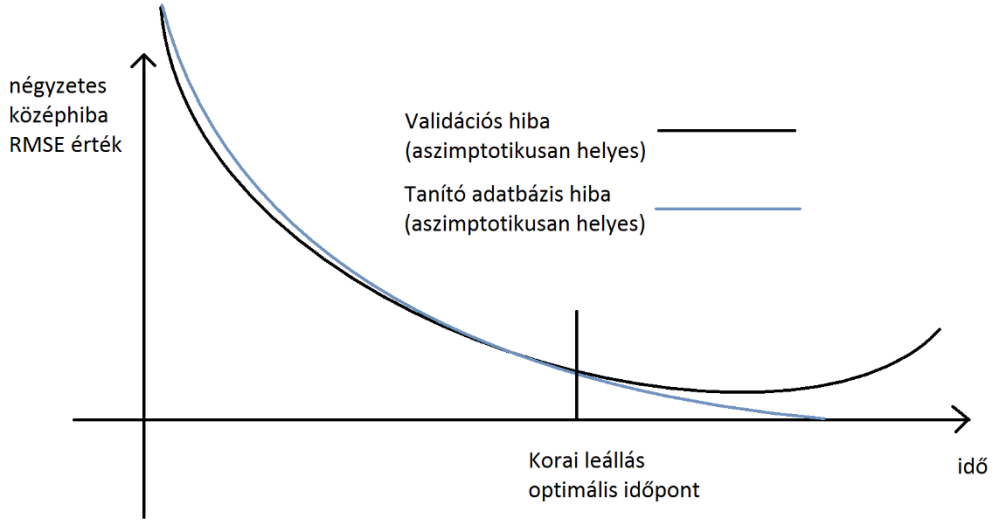
A gépi tanuló rendszerek alkalmazásának egyik legfőbb kérdése a túltanulás, illetve ennek elkerülése. A túltanulás mint probléma azt jelenti, hogy a rendszer nem a tanító adatokból próbál általános belső reprezentációt készíteni, hanem magukat a tanító adatokat tekinti a rendszer egészének. Amennyiben nem sikerül a rendszert általánosításra rávenni, az nem fog működni valós teszt adatok esetén. A tapasztalatok azt mutatják, hogy a rendszer hajlamosságát a túltanulásra a tanítandó paraméterek száma és a tanító adatok arányának hányadosa írja le jól. Ez azt jelenti, hogyha kevés adattal rendelkezünk a feladat bonyolultságához képest, vagy túl bonyolult architektúrát használunk egy egyszerűbb problémára, a rendszerünk erősebben túl fog tanulni.

A túltanulás elkerülésének egyik módszere (a hálózati architektúra adatokhoz való szabása mellett) a tanító adatbázis növelése. Azonban ennek vég nélküli fokozása sem lehet megoldás, hiszen nagyobb adatbázis nagyobb tároló és főleg számítási kapacitást kíván. Másik probléma, hogy sok valós probléma esetén költséges vagy egyenesen lehetetlen a további adatok

beszerzése. Ilyen esetben más módszerekhez kell folyamodnunk a túltanulás elkerülése, vagy legalábbis késleltetése érdekében.

Korai leállás (early stopping)

A túltanulás detektálása ellenőrzött tanítás során a validációs és a tanító adatbázis által számolt hibafüggvény összehasonlításával történik. Amennyiben a hiba átlaga a validációs és a tanító adatbázison közel egyező mértékben csökken, túltanulás nem lépett fel, és a tanítás biztonságosan folytatható. Ha a tanítást folytatva azt tapasztaljuk, hogy a validációs adatbázison a hiba újra növekedni kezd, miközben a tanító adatokon a hiba tovább csökken, bekövetkezett a túltanulás jelensége. Ennek elkerülésére egy logikusnak tűnő megoldás az ún. *early stopping* [18] technika, amely megkísérli megtalálni azt a pontot, ahol a validációs halmazon a hiba még nem kezdett el újra növekedni. A következő sematikus ábrán látható az *early stopping* technika lényege.



2. ábra - Korai leállás

A 2. ábrán látható időpillanat a legjobb a tanítás leállítására. A valóságban a megfelelő időpont megtalálása nem könnyű, de az is igaz, hogy már az optimálishoz közeli megállással is sokat nyerhetünk.

Dropout

Az *early stopping* technika mellett gyakran alkalmazott technológia az ún. *dropout* módszer [19]. Ez szintén elsődlegesen a túltanulás késleltetésére alkalmazható, de egészen más az elve, mint az *early stoppingnak*. Dropout alkalmazásakor tanítás során avatkozik bele a tervező a hálózatba, az egyes iterációk során a rendszer véletlenszerűen kihagy elemi neuronokat a rejtett rétegekből. Ez a technika megakadályozza a neuronok koadaptációját. A kihagyott neuronok arányát egy százalékos értékben határozzák meg. Ez általában 10 és 50% között van. A módszer során fontos, hogy a neuronokat tényleg véletlenszerűen kapcsoljuk ki-be. A tapasztalatok szerint a *dropout* technika igen hatékonyan képes csökkenteni a káros túltanulást és ezáltal javítja a hálózat teljesítményét. Az alkalmazott technikákról az eredményeket tárgyaló fejezetben lesz szó.

Batch méret

A tanítás során el kell még döntenie a tervezőnek, hogy egy-egy tanítási iteráció során (ezt *epoch*nak is neveik) a tanító adatbázis mekkora csomagokban („batch”) dolgozza fel a súlymódosítás céljára. A két szélsőséges eset az ún. *full-batch learning* és az *online learning. Full-batch* esetén a tényleges gradiens kiszámítása történik minden iterációban, az összes adat alapján (tehát itt egy „csomagban” dolgozzuk fel az összes mintát). Ez sokszor szükségtelenül lassú, hiszen nehezebb párhuzamosítani a folyamatot. A másik szélsőség, az *online learning* használata esetén egy véletlenszerűen választott tanító adat alapján állítják a súlyokat. Ennek a módszernek a gyengesége pedig a rendkívül zajos gradiensek. A két szélsőséges beállítást viszonylag ritkán használják, gyakorlatban a kettő között található ún. *mini-batch learning* terjedt el annak nagyobb sebessége, és pontossága miatt. Itt a tanító adatbázis tervező által

meghatározott nagyságú, az összes adat közül véletlenszerűen mintavételezett adatokból álló csomagokban („batch”-ekben) dolgozzák fel az adatbázist, és ezek alapján számolják a gradiens egy közelített értékét. Bár ez az érték nyilvánvalóan különbözik a valós gradienstől valamilyen mértékben, ez a bizonytalanság (zaj) a lokális minimumok elkerülése szempontjából még előnyös is lehet. Az optimális gradiens méret szintén az adott probléma függvénye. Jelen TDK dolgozat során a tanítások többségében *mini-batch learning*-et alkalmaztam.

2.4 A hiperparaméter-optimalizáció

Amint azt az előző alfejezetben olvashattuk, egy neurális hálózat feladathoz való tervezése során igen sokféle hiperparamétert kell megfelelően beállítanunk az eredményes tanításhoz. Gyakorlatban használható szabályok, törvényszerűségek nélkül, az ezzel kapcsolatos optimalizálási kérdések központi részét alkotják a neurális hálózatok tudományterületének. Az ún. hiperparaméter-optimalizáció, vagyis a hiperparaméterek minél jobb beállításának elméleti háttere ma is intenzíven kutatott terület. A fő kihívások, amikkel a tervezőknek itt meg kell küzdeni, elsősorban a keresési tér hatalmas méretei, és a térben egy mintapont vizsgálatához szükséges idő nagysága. A keresési tér ebben az esetben a hiperparaméterek tere, amely – választástól függően az időbeli koráltok miatt jellemzően 2-7 dimenziós. A dimenziószám attól függ, hogy a tervező az adott feladat esetén mely hiperparamétereket akarja optimalizálni. A leggyakoribb választások MLP hálózatok esetén:

- Tanulási ráta
- Rétegek száma
- Neuronok száma egy rétegenként
- *Batch size*
- *Dropout* nagysága

Ezen felül – ritkábban alkalmazott megoldásként – felvehető még a momentum módszer paramétere. A felsorolásból látható, hogy ilyen esetben egy 5 dimenziós térben kell keresnünk. További problémát okoz az a tény, hogy 1 mintapont korrekt kiértékeléséhez (az adott pontbeli hiba megállapításához) tulajdonképpen az adott paraméterekkel végig kell tanítani a hálózatot a

teljes adatbázison. Mivel nagy adatbázisok esetén ez a folyamat órákig, de akár napokig is eltarthat, a teljes optimalizálási folyamat akár csupán néhány tucat mintapont vizsgálata esetén is elfogadhatatlanul megnőhet.

Hogy kordában tarthassuk a futási időt, köthetünk egy mérnöki „kompromisszumot”. Ha feltételezzük, hogy a tanítás elején gyorsabban csökkenő hibával rendelkező elrendezések a későbbiekben is megtartják előnyüket (erre semmilyen elméleti garancia nincsen), akkor lehetőségünk van az első néhány iteráció után leállítani a tanításokat, megtakarítva sok időt. Így azonban sosem lehetünk biztosak abban, hogy tényleg a *tanítás végén* legjobb elrendezést választottuk győztesnek.

A mintapontok kiválasztásakor többféleképpen is eljárhatunk. A legegyszerűbb az ún. *manuális keresés* (Manual search), amikor intuitív módon, kézzel próbálunk ki hiperparaméter-elrendezéseket. Ez az esetek többségében lassú, és nem hatékony.

Szintén nem igazán hatékony megoldás a *szisztematikus keresés* (Grid search), elsősorban a keresési tér méretei miatt.

A legjobb megoldást a tapasztalatok szerint [20] valamilyen *véletlen keresés* alapú megoldás jelenti. Ez azért is valószínű, mert a legtöbb gyakorlati problémánál a különböző hiperparaméterek hangolásának végső hibára való hatása nem egyenlő (vannak „fontosabb” és „kevésbé fontos” hiperparaméterek). Egy intelligensebb kereső algoritmus ezt is figyelembe véve hatékonyabban választhatja ki a mintapontokat.

2.5 Aggregált (ensemble) tanítás

Gyakran előfordul, hogy egy gépi tanulással megoldható problémára többféle modellt sikerült készíteni. Ilyen esetekben lehetőség van arra, hogy az egyes modellek becsléseit összegezzük, és egy ún. aggregált modellben helyezzük el. Az ilyen aggregált, vagy idegen szóval *ensemble* modellek megfelelő tervezéssel a tapasztalatok szerint képesek valamivel jobb teljesítményt nyújtani, mint az egyes tag modellek külön-külön. Ez a hatás akkor a legerősebb, ha az egyes tag modellek egymástól nagyon különböznek (a tag modelleknek nem is kell feltétlenül neurális hálózatnak lenniük, választhatunk más gépi tanuló eljárást, például döntési fákat,

Szupport Vektor gépeket [21] stb). Az eltérő modellek egyike sem képes a probléma tökéletes megtanulására, viszont eltérő belső működésüknek köszönhetően várhatóan lesznek olyan minták a teszt adatbázisban, ahol az egyes tag modellek egymás hibáját kiátlagolják.

A tag modellek kimeneten való összegzésére is sokféle módszer kínálkozik, a legegyszerűbb megoldásoktól (algebrai átlagszámítás), a bonyolultabb elrendezésekig (elterjedt megoldás erre a feladatra egy külön kis neuronhálózat betanítása, mely képes szabályozni, hogy mikor melyik tag modellt milyen mértékben kell figyelembe venni). Fontos általános jellemzője az ilyen elrendezéseknek, hogy az egyes tag modellek ugyanazon az adatbázison tanulnak.

Létezik azonban egy másik aggregált módszer is. Itt az egyes tag modellek az adatbázis különálló részein tanulnak, így ezen kisebb rész „szakértőivé" válnak.

Az adatbázis szétbontását egy olyan bemeneti változó (vagy változók) mentén érdemes megtenni, amely a feladat szempontjából kiemelt jelentőséggel bír. Jelen TDK kutatómunka során egy ilyen aggregált modellt fogok bemutatni.

Az adatbázis szétbontása ebben az esetben egy, a beszéd dinamikájára és ezáltal a felhasználói élményre jelentős hatással lévő inputon, a beszédhangsúly-szinten keresztül történt. Az alapvető cél itt az volt, hogy az egymodelles rendszerhez képest kedvezőbb dinamikájú alapfrekvencia függvényeket lehessen előállítani. Az ezzel kapcsolatos eredmények és a munka részletesebb leírása a következő 4. és az 5. fejezetekben olvasható.

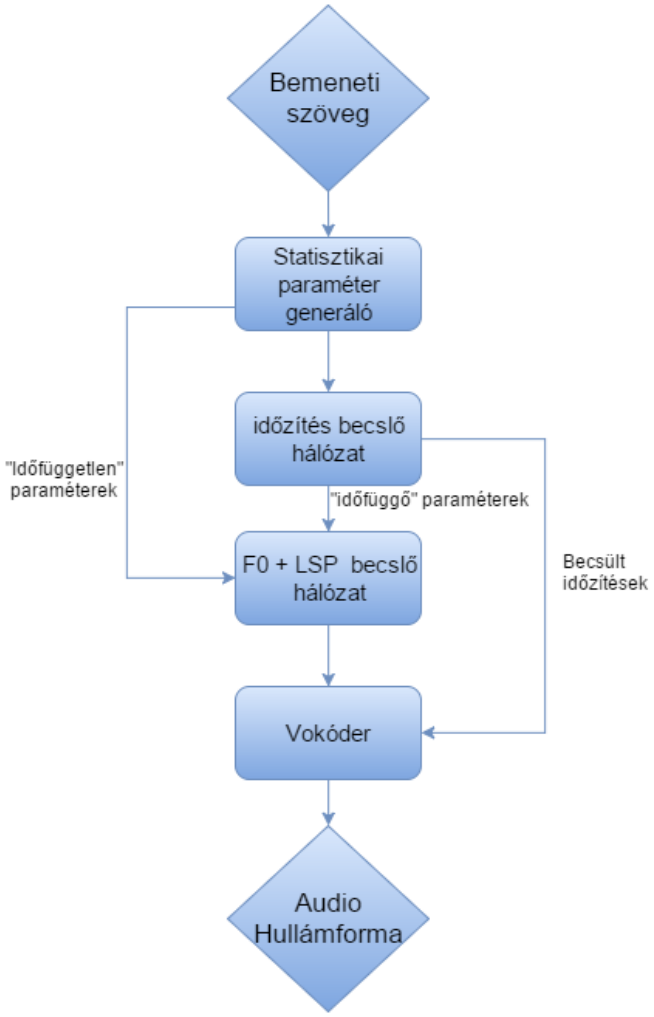
3 Kiindulási állapot – beszédparaméter adatbázisok

A munka kezdetén a BME-TMIT SmartLab a kutatómunka céljaira rendelkezésemre bocsátotta beszédhang adatbázisának [39] egy részét. Az adatbázis emberi beszélőktől (stúdiókörülmények között) felvett rövid kijelentő mondatokat tartalmaz. Minden mondat egy külön .wav formátumú fájlban található. Minden hangfájlhoz egy adatbázis fájl is tartozik, mely a mondatban elhangzó hangkaraktereket tárolja a hozzá tartozó időbélyegekkel. A teljes hangkarakter-lista a Függelékben található. Az adatbázis fájlokban ezen kívül számos más kontextus paraméter is található, melyek kiválóan alkalmazhatók statisztikai parametrikus beszédszintetizáló rendszerek tervezésekor. Ezen paraméterek közül néhány a hangkaraktert a mondaton belül tágabb kontextusba helyezi (jelölve vannak a hangkaraktert egy és kettő távolsággal megelőző és az azt követő hangkarakterek is), valamint foglalkozik a beszéd nagyobb egységeinek a leírásával is (szótag, szó, tagmondat és mondat szinten) [1]. A kutatómunka során felhasználásra került, a hangkaraktereket leíró paraméterek teljes listája a Függelékben megtalálható.

4 Az elvégzett munka bemutatása

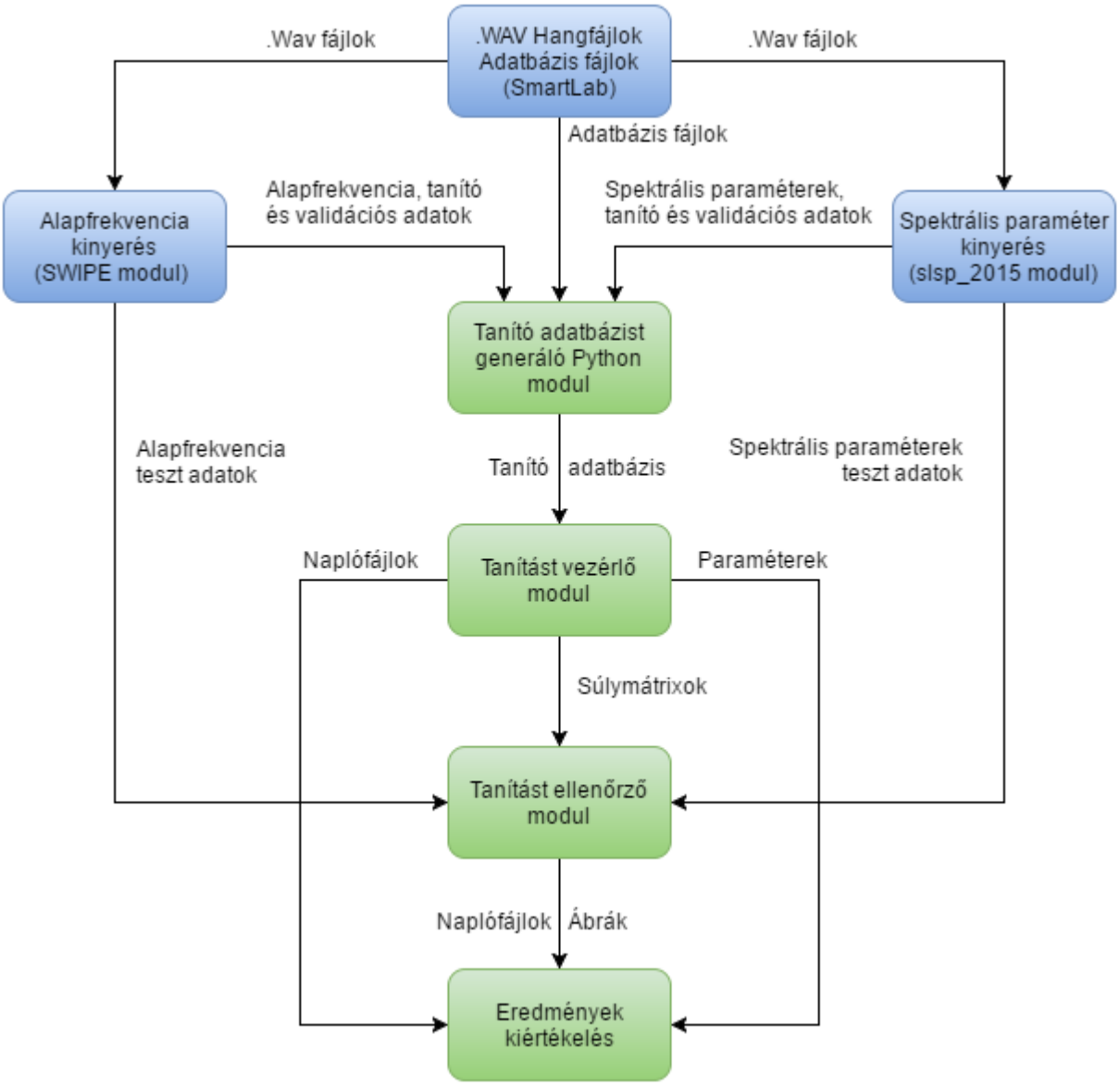
Ez a fejezet részletesen ismerteti a meglévő beszédparaméter-adatbázis neurális hálózat kompatibilis tanító adatbázissá való konvertálásának lépéseit (előfeldolgozás), a tanítás folyamatát, a hiperparaméter-optimalizálást és az aggregált modellel kapcsolatos részleteket.

A teljes TTS folyamatot, azaz a szöveg beszéddé alakítását a 3. ábra szemlélteti. A folyamat elején a rendszer megkapja a felolvasni kívánt szövegrészt. A rendszer a szövegből legenerálja nem időfüggő statisztikai paramétereket. Ezek segítségével az időzítéseket becslő neurális hálózat előállítja az időfüggő statisztikai paramétereket. Ezen paraméterek összessége alapján az alapfrekvencia és a spektrális paraméter becslő (f0+LSP) hálózat megbecsüli a hullámforma generáláshoz szükséges összes paramétert. Az utolsó lépésben a korábban már említett vokóder szoftver az alapfrekvencia, a spektrális paraméterek és az időzítések figyelembe vételével elkészíti a hullámformát, ami a bemeneti írásos mondat hangos változata lesz annak a beszélőnek a hangjával, akinek a kezdeti hullámforma adatbázisával a neurális hálózatok tanítva lettek.



3. ábra - A teljes TTS rendszer működése

A 4. ábrán csak az alapfrekvencia és a spektrális paraméterek kinyerését és tanítását ábrázoltam. A 4. ábrán zölddel jelöltem azokat az elemeket, melyek a TDK munkám részeként készültek. Az első lépés a tanító adatbázist generáló modul létrehozása a Smartlab adatbázisából. A kész tanító adatbázissal a hálózat tanítása elvégezhető, ezek után a tanítást ellenőrző modul a tanított hálózattal előállítja a becsült paramétereket és az eredményesség ellenőrzéséhez szükséges naplófájlokat és adatokat.



4. ábra - f0 és spektrális (LSP) paraméterek tanítása

4.1 Az adatbázis előfeldolgozása

Az előfeldolgozás szinte minden gépi tanulással kapcsolatos munkában előkerül. Az adatok, amelyet a rendszer tanításához szeretnénk használni, általában olyan állapotban és formátumban van, hogy az közvetlenül nem használható. A legtöbb esetben azonban az előfeldolgozás nem csupán az egyes formátumok közötti gépies másolásból áll. A hálózatot és az adatbázist kölcsönösen egymáshoz kell illeszteni a hatékony tanítás érdekében. A következő alfejezetekben a TDK kutatómunka során alkalmazott előfeldolgozási lépéseket ismertetem.

4.1.1 Az alapfrekvencia referencia adatok előkészítése

Mivel az MLP hálózat ellenőrzött tanulást kíván, mindenképp szükség van egy, a hangfájlokból előállított referencia alapfrekvencia adatbázisra, amelyhez képest a neurális hálózat hibafüggvényét képezhetjük. Az alapfrekvencia hangfelvételekből való kinyerésére többféle megoldás is létezik. Jelen kutatás során az ingyenesen elérhető SWIPE (A Sawtooth Waveform Inspired Pitch Estimator) [22] szoftvert használtam, ami C nyelven íródott. A programnak a hangfájl mellé bemeneti paraméternek meg kell adni a mintavételi időt és a minimum-maximum alapfrekvencia értékeket, amelyek az adott hangfájl beszélőjére jellemzők. A program ezekből az adatokból generál egy szöveges állományt, amely a választott mintavételi időnként számol egy alapfrekvencia értéket. A generálás során a használt adatbázisokhoz (1 férfi és 1 női beszélő lett feldolgozva) tartozó frekvencia-szélsőértékek: a férfi beszélőnél 85Hz és 180Hz, a női beszélőnél 165Hz és 255Hz, a választott mintavételi idő mindkét esetben 0.005s. Az emberi hangképzés sajátosságait, és az alapfrekvencia változásakor fellépő fiziológiai változásokat figyelembe véve ez az alapfrekvencia igen finom modellezését teszi lehetővé. (200 minta/másodperc).

4.1.2 Az spektrális paraméter referencia adatok előkészítése

A beszélőre jellemző spektrális paraméterek kinyerése is célszoftverrel történt. A [23]–ban ismertetett szoftver képes közvetlenül a hullámformából kinyerni mind az alapfrekvenciát,

mind pedig a spektrális paramétereket is. A szoftver segítségével bármelyik paraméter az eredetivel kicserélhető, és a becsült f0 menetekkel egy új hullámforma fájlba összeállítható. Ez a szoftver a szubjektív, meghallgatásos teszteknél is felhasználásra került, ekkor az általunk becsült paraméterekből állítottam össze új, generált hullámformákat. Ez utóbbi eljárást „vocoding”-nak nevezik, az erre képes eszközt pedig vokódernek (Vocoder), ami a „Voice encoder-decoder” rövidítése [3].

4.1.3 Adatok skálázása

Mint ahogy azt korábban már említettem, a neurális hálózatokon a tanítás során úgy lehet sikerrel eljárni, ha a hálózat és a tanító adatbázis sajátosságait is figyelembe vesszük [24]. A hálózat oldaláról - bár az elmélet szerint, ha a függvény folytonos, azt egy neurális hálózat képes közelíteni - könnyű végiggondolni, hogy az alkalmazott nemlineáris függvények értékkészletének korlátossága miatt nagy kimeneti érték előállítása problémás. Jelen kutatás során a hasonló nemzetközi próbálkozásokat tanulmányozva a bemeneti és a kimeneti adatokra különböző transzformációkat alkalmaztam. A kimeneti adatokat egy hagyományos standardizáló algoritmussal skáláztam 0.01 és 0.99 közé. Azért nem érdemes pontosan nullát és egyet választani, mert bizonyos hálózatok esetén a 0 és az 1 kimeneti érték csak végtelenbe tartó bemeneteknél valósul meg, így itt instabillá válhat a hálózat. A bemeneti adatok esetén célszerűbb úgy áttranszformálni az adatbázist, hogy egy közel 0 várható értékű halmaz kerüljön a bemenetre minden iteráció során. Ennek megvalósítására egy alkalmas megoldás egy Gauss-féle standard normál skálázó használata (a munka során a Scikit-learn [25] előfeldolgozást segítő (preprocessing) moduljának StandardScaler függvénye lett felhasználva). Az algoritmus nulla várható értékű, és egységi szórású adatbázist hoz létre, így megfelel az ilyen téren támasztott kritériumoknak. A skálázást globálisan, a teljes adatbázison egyszerre végeztem el, figyelve arra, hogy a különálló teszt adatokat ne illesszük, hanem a tanító adatbázisnál meghatározott skálázási paraméterek továbbvitelével csak transzformáljuk. Ha nem így járnánk el, hanem külön illesztenénk és skáláznánk, akkor ezzel extra („jövőbeli”) információhoz juttatnánk a rendszert, és ez meghamisítaná az objektív méréseket a teszt adatbázison. Az eredmények kiértékelésekor a

kimeneti adatokon a vissza transzformációt is végre kell hajtani, ekkor is az eredetileg kialakított skálázó mátrixot használjuk.

4.1.4 Interpoláció

A neurális hálózat a külvilágból vett információit egy belső numerikus reprezentációvá alakítja át. Amennyiben a numerikus reprezentáció jól modellezi a megoldani kívánt problémát, a tanítás sikeres volt, és a hálózat a gyakorlati alkalmazás esetén is várhatóan jól fog teljesíteni. Ezen numerikus sajátosságokból adódóan azonban a nem folytonos függvények modellezése általában nagyobb problémát jelent a hálózatok számára. Az alapfrekvencia függvény modellezése során probléma, hogy a függvény nem folytonos, sőt nem is mindenhol definiált a vizsgált hangfájlokban. A zöngés hangoknak az általam használt modellben jól meghatározható alapfrekvenciája van, a zöngétlen hangkarakterek gerjesztése azonban zajszerű, így ezen elemek esetén az alapfrekvencia értéke ebben a modellben nem is definiált.

A neurális hálózat viszont csak numerikus adatokkal tud dolgozni, így amennyiben a tanítás végeztével mi egy a gyakorlatban is használható alapfrekvencia becslőt szeretnénk kapni, mindenképpen generálni kell szimulált kimenet értékeket azokhoz a mintákhoz is, amikhez a valóságban nem definiált f0 érték tartozik. Természetesen az interpoláció nem az egyetlen lehetséges megoldás a függvény folytonossá tételére, de mivel a jelen architektúra esetén ez került alkalmazásra, a továbbiakban ezzel foglalkozom.

A hálózattól elvárjuk, hogy az alapfrekvencia becslése mellett képes legyen különbséget tenni a valós és a szimulált f0 értékek között (ez tulajdonképpen önmagában tekintve egy bináris osztályozási probléma). Ezt egy ún. zöngés/zöngétlen f0 kimenet bevezetésével oldottam meg. A hálózat a tanítás során ezt a kimenetet is optimalizálja az f0 érték mellett.

Az interpoláció során figyelnünk kell arra, hogy az f0 függvény alapvető jellegzetességei megmaradjanak. A diplomamunka során csak kijelentő mondatok alkották a tanító adatbázist. A kijelentő mondatokra jellemző, hogy az f0 menet egy mondaton belül enyhén csökkenő tendenciát mutat. Egy hangfájl esetén az interpolálni szükséges mintasorozatok változó

hosszúságúak, interpolált adat a hangfájl legelején és a végén is előfordulhat. Ezt az interpoláció során figyelembe kell venni. Az alkalmazott interpolációs technika a következő:

- minden interpolált szakasz lineáris, a kezdőpont a legutolsó definiált érték a szakasz előtt, a végpont a legelső definiált érték az interpolált szakasz után, ha léteznek ilyen pontok
- a lineáris interpolálás során kiszámoljuk a két adat különbségét, és az iránytól (növekvő vagy csökkenő) függően egyenletes lépésekkel alakítjuk ki a szimulált értékeket

Ha a hangfájl elején vagy a végén is interpolálni szükséges sorozat van, az interpoláció a fájlban az első definiált érték 110%-ától indul (lefelé), ha a végén, akkor az utolsó definiált f0 érték 90%-ig interpolálunk (itt is lefelé). Ezzel a megoldással az interpoláció hibája egy elfogadható szintre csökkenthető. Hangsúlyozandó, hogy ez a megoldás sem lehet tökéletes, többek között azért sem, mert a kijelentő mondatok esetén sem mindig igaz az f0 függvény szigorú monotonitása.

4.1.5 A tanító adatbázis felépítése

A tanító adatbázist elkészítő script kimenete egy szöveges fájl, ahová soronként kerülnek be az egyes adatbázis elemek. Egy sorba, vesszővel elválasztva, a következő adatok kerülnek:

Bemenetek	Kimenetek
68*5 db bináris bemenet	24+1 db spektrális paraméter
26 numerikus bemenet	1 db f0 paraméter
	1 db zöngés-zöngétlen flag
Összesen: 366 bemenet	Összesen: 27 kimenet

1. Táblázat – neurális hálózat bemenetek és kimenetek

A 366 bemenet túlnyomó többsége bináris (ún. „one-hot” kódolású): 68 darab beszédhang közül lehet választani, ezek mindegyike kapott egy bemenetet, így a 68 bemenet közül egy időben csak pontosan egy darab 1-es lesz. A bemenet ilyen formán történő képzése gyakori a neurális hálózatok esetén. A 68 darab beszédhang egymás után ötször szerepel, mert a modellben rögzítésre kerül a korábban említetteknek megfelelően az adott hangkaraktert megelőző, valamint az azt kettővel megelőző beszédhang, valamint az azt követő és az azt kettővel követő beszédhang is. Ez 68*5 = 340 bemenetet jelent. A maradék 26 bemenet numerikus, és különböző kiegészítő információkat tartalmaz az adott elemmel kapcsolatban. Ilyen bemenet például az adott adatbázis elem százalékos helyzete a beszédhangon belül (egy hangkarakterhez kb. 20-30 adatbázis elem tartozik, hiszen a korábban már említett SWIPE szoftver időfelbontása jóval precízebb, mint egy beszédhang átlagos hosszúsága), de itt jelennek meg azok a már korábban említett paraméterek is, melyek az adott hangkaraktert a mondaton belül tágabb környezetben helyezik el.

4.2 A hálózati modellek összeállítása

Munkámhoz a Python [26] nyelvet választottam. A Python nyelv előnye, hogy a tudományos kutatásoknál használt többi megoldással szemben (pl. MATLAB) általánosabb célú programnyelv, így a fejlesztők lehetőségei szinte korlátlanok a felmerülő problémák megoldását illetően, ingyenessége és könnyű bővíthetősége pedig nagyfokú rugalmasságot tesz lehetővé. A másik ok, hogy a mély neurális hálózatok kezelésére legmodernebb keretrendszerek többsége Python nyelven íródott. Az GPU-n történő alacsony szintű programozás megvalósításában nagy segítséget nyújtott a Theano [27] nevű, erre a célra kitalált Python könyvtár. A Python nyelv adatkezelését nagyban könnyítik az ingyenes Numpy és a Scipy [19] csomagok, a hatékony és könnyed adat vizualizációt pedig a Matplotlib [28] könyvtár támogatja.

A neurális hálózati modelleket is Python programnyelv segítségével készítettem, az ingyenesen elérhető Keras [29] csomag felhasználásával. A programcsomagban lehetőségünk van többféle neurális hálózati architektúra, köztük MLP felépítésére is. A programcsomagnak jelentős felhasználói tábora van közérthetősége és a gyors prototípusfejlesztést támogató metódusai miatt. Egy MLP összeállításához definiálnunk kell egy hálózati struktúrát, majd ezután be kell vinnünk

a legfontosabb paramétereket (például a tanulási ráta). A hibafüggvény és a tanító algoritmus pontos definiálása után az általunk használni kívánt tanítást segítő módszereket (amennyiben azok elérhetők) is hozzá kell adnunk a modellhez. A szoftver lehetőséget nyújt arra, hogy a tanítás végeztével az eredmény súlymátrixot elmentsük, amit teszteléskor vagy a tanulás tovább folytatásánál újra használhatunk. A kutatómunka során sokféle hálózati architektúra és paraméter beállítás tesztelésre került ennek a szoftvernek a segítségével, ezekről bővebben az Eredmények fejezetben lesz szó.

Az aggregált modell esetén a módszer igen hasonló, hiszen itt a végső eredmények kiszámítása előtt egyesével, egymás után be kell tanítani a tag modelleket az adatbázis nekik megfelelő részén. A tag modellek mérete úgy lett meghatározva, hogy a számítási idő egy aggregált modell és egy nagy modell esetén közel azonos legyen.

A tanszéken lehetőségem volt gyors processzorokkal és modern grafikus processzorral felszerelt szervereken tanítani a hálózatokat. (Nvidia GeForce GTX 970 és később Nvidia GeForce GTX Titan X GPU típusok).

4.3 Hiperparaméter-optimalizálás

Jelen TDK kutatómunka során hiperparaméter-optimalizálással kíséreltem meg egy minél kisebb hibát okozó hiperparaméter-elrendezést találni. A keresési teret rétegenként szélválasztottam, és csak a 3 és 4 réteges hálózatok további vizsgálatával foglalkoztam, mivel a korábbi, manuális optimalizálás során ezek adták a legjobb teljesítményt. A keresési tér 3 dimenziós volt, az egyes dimenziók és az keresési intervallumok a következő táblázatban olvashatók. A 3 és a 4 réteges keresésnél azonos keresési intervallumokat alkalmaztam. Az egyes mintapontok vizsgálatánál az első 15 iteráció (epoch) alatt nyújtott teljesítményt vettem figyelembe.

Hiperparaméter	**Intervallum**
Rétegenkénti neuron szám	2000-4000
Tanulási ráta	0.005-0.1

Batch méret	20-100

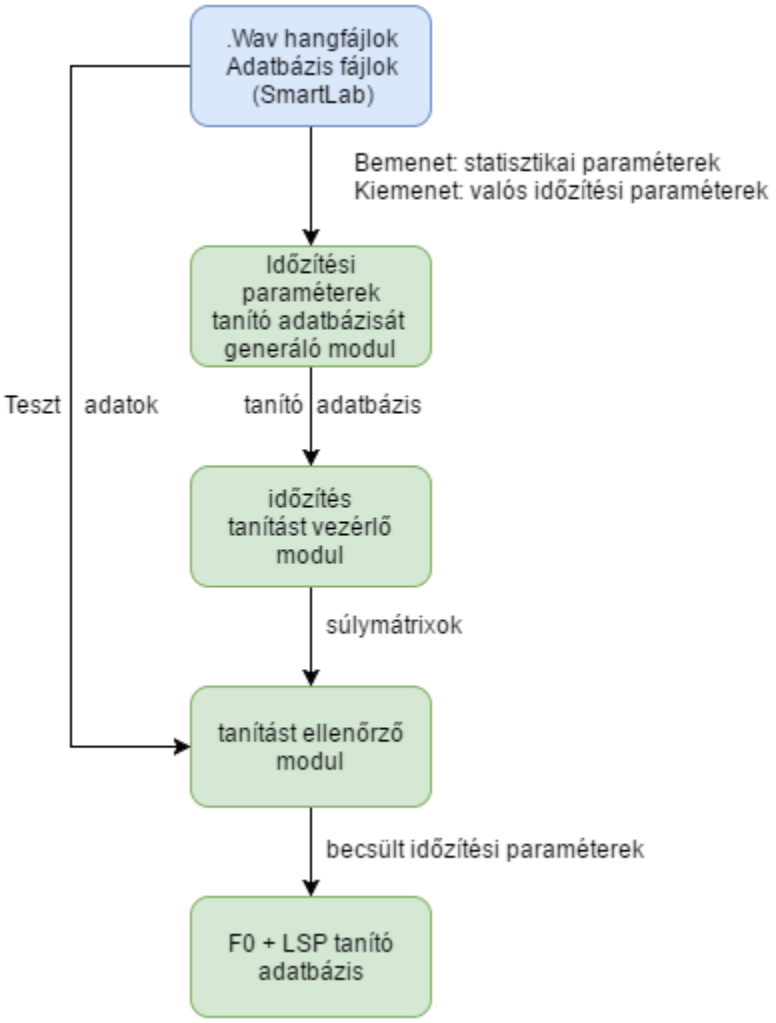
2. Táblázat – hiperparaméter-optimalizáció intervallumok

A hiperparaméter optimalizálás összességében eredményes volt, a legjobb manuális beállításnál némileg jobb beállításokat sikerült tanulni. A nagyon nagy eredménykülönbségek hiánya azonban arra utal, hogy a hibafelületek meglehetősen „simák”, ami jó minőségű adatbázisra utal. Az eredményekről részletesen a következő fejezetben számolok be.

A hiperparaméter-optimalizáláshoz a szintén közkedvelt Hyperopt [30] Python csomagot használtam. A szoftver többféle heurisztikus keresési eljárást is támogat.

4.4 Az időzítést becslő hálózat

Az 5. ábrán az időzítési paraméterek előállítása látható. Az egész rendszert elrendezését bonyolítja, hogy az alapfrekvencia és a spektrális paraméterek paramétereinek előállítását végző neurális hálózat bemenetként olyan adatokat is fogad, melyeket az időzítési paraméterekből számol a rendszer (például az adott hangkarakter hossza). Így nem lehetséges a teljes rendszert egy neurális hálózattal betanítani. Az időzítési paraméterek tanítása az alapfrekvencia és a spektrális paraméterek tanításhoz hasonlóan történik, de a kisebb adatbázis miatt a hálózat mérete is kisebb lesz. A megoldás menete a fejezet elején a 4. ábrán látható módszerhez nagyon hasonló. Itt is előállítjuk a hálózat tanításához szükséges adatbázist, elvégezzük a tanítást, és a végén kiértékeljük az eredményeket. Ezen hálózat esetén az eredmények (az időzítési paraméterek) a 4. ábrán található hálózatban bemenetként jelentkeznek, ezt az 5. ábra végén az f0+ LSP tanító adatbázisba való irányítással jeleztem.



5. ábra - az időzítési paraméterek előállítása

A tanítási feladat, és az adatbázis itt jóval kisebb, hiszen itt nem kell 25 ms-os keret szinten feldolgozni a mondatokat, elég a beszédhangokat, mint elemi adatelemeket kezelni. A kisebb adatbázis miatt a hálózat méretei is kisebbek lehetnek. A sokkal kisebb tanítási idő azt is eredményezi, hogy ha erre a kis adatbázisra hiperparaméter-optimalizálást tervezünk, végigtaníthatjuk az összes jelölt hálózatot, így a ténylegesen legjobb architektúrát választhatjuk ki az optimalizálás végén.

Az általam készített, időzítést becslő hálózatot 5 dimenziós hiperparaméter-térben optimalizáltam. A paraméterek a következők voltak:

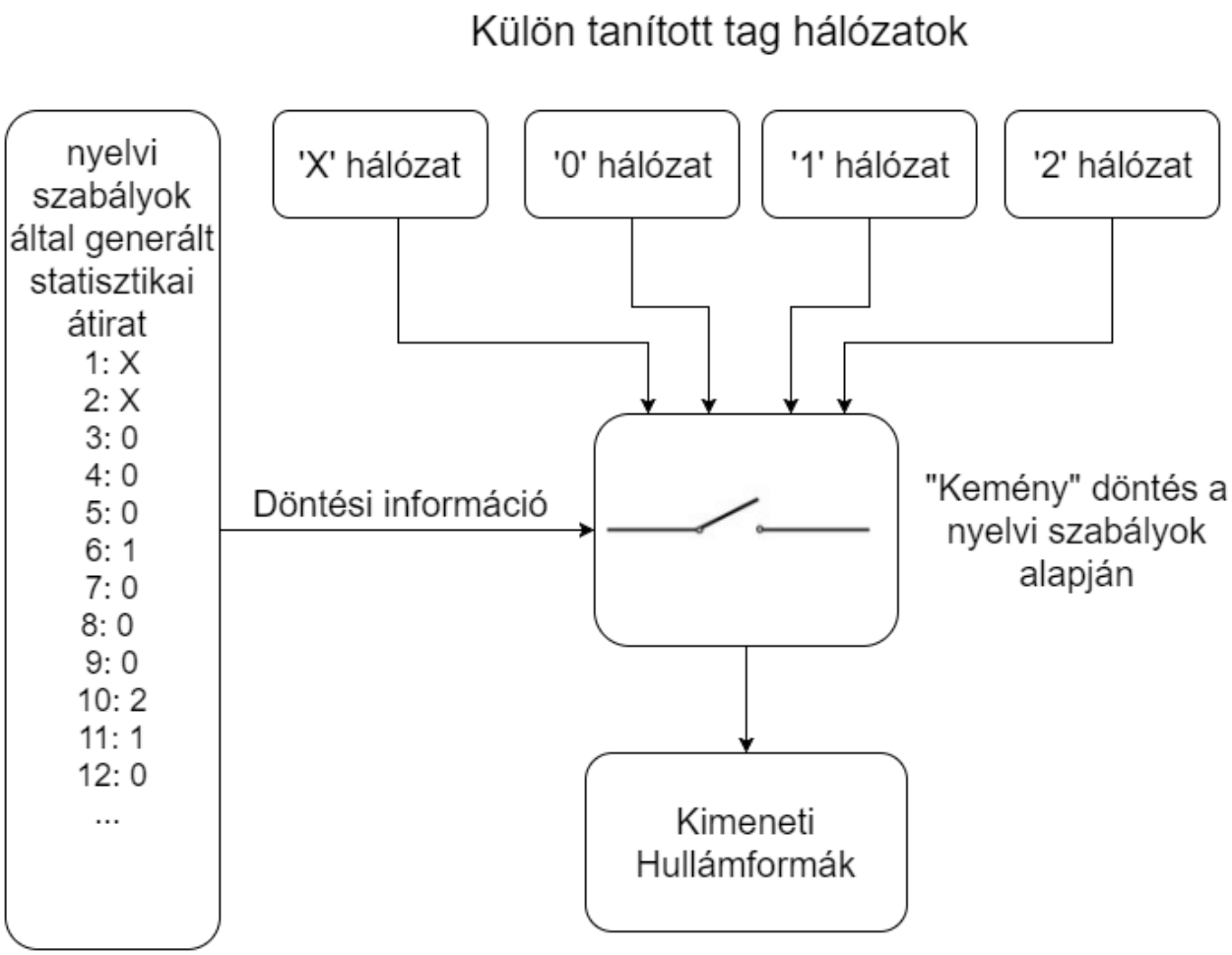
- Tanulási ráta

- Batch méret
- Neuronok száma
- Rétegek száma
- Dropout nagysága

Az eredmények a legjobb beállításokkal az 5. fejezetben olvashatók.

4.5 Az aggregált (ensemble) architektúra

A gépi beszédkeltő rendszerek gyakori problémája a beszédhangsúlyok pontatlan kezelése, és a (részben) ebből következő túl monoton beszéd. Az egymodelles neurális hálózatok is jelentkezik ez a probléma. Bár a tanító adatbázisban a hangsúly többször megjelenik, mint beszédparaméter, a néhány hangsúlyra vonatkozó bemenet, a bemenetek nagy száma (jelen kutatómunka keretében 366 db, lásd 4.1.5 fejezet) miatt nem kap elegendő figyelmet. Az aggregált modellek ismeretében természetesen adódik a felvetés, hogy vizsgáljuk meg egy olyan aggregált modell teljesítményét, ahol az egyes tagok („szakértők”) a hangsúly szerint vannak szétválogatva. Szintézis időben a rendszer hangsúly szerint választ az egyes tag hálózatok között. A legegyszerűbb elrendezés szerint az lesz a végső kimenet az adott időpillanatban, amit a hangsúly szerinti szakértő becsül. A folyamat a következő ábrán jól követhető:



6. ábra – Aggregált architektúra – tesztelési fázis

Kérdésként felmerülhet, hogy a tanuló adatbázisban milyen módszerrel határozzuk meg a hangsúlyokat. A meghatározásra több lehetőség van. Tanítási és teszt időben is rendelkezésünkre állnak a nyelvi szabályokból számított hangsúlyok. A valóságos beszélők által képzett hangsúlyok azonban általában eltérnek, illetve nem pontosan ott vannak, mint ahogy azt a nyelvi szabályok diktálnák. (Ezek az eltérések is beszélőfüggők.) A kutatómunka során rendelkezésre álltak az olyan hangsúly címkék is, amelyek közvetlenül a hullámformából (magából a felvett hanganyagból) [31] lettek becsülve. A tag neurális hálózatok ezekből tanultak. Figyelni kell azonban arra, hogy valós alkalmazási helyzetben nem áll rendelkezésünkre a hullámforma, így ott csak a nyelvi szabályokat tudjuk használni.

A jelenlegi rendszerben a hullámforma alapján számolt hangsúlyok 4 szintben vannak rendezve. A minták legnagyobb része 0 címkét kapott, ami a „nincs hangsúly”-t jelenti. Az 1 és a 2 a gyenge és az erősebb hangsúlyt jelzi. A negyedik pedig az X címkével ellátott, ami a nincs információ, vagy a „*don’t care*”-t jelenti (ez általában hangfájlok legelején és legvégén jelenik meg). A nyelvi szabályok alapján becsült hangsúlyok azonban összesen 7 szinten jelennek meg, így a következő átírást kell alkalmazni:

Nyelvi szabályok szerinti becslő	Hullámforma becslő	Magyarázat
0	0	Nincs hangsúly
1, 2	1	Gyenge hangsúly
3,4,5	2	Erős hangsúly
X	X	Ismeretlen/hangsúly nem értelmezett

3. Táblázat – kombinált modell tagok átírása

A kutatómunka ezen, jól elkülöníthető, aggregált tanításra vonatkozó része a 2016-os SPECOM (International Conference on Speech and Computer) nemzetközi konferencián jelent meg konferenciacikk formájában. Jelen TDK dolgozat szerzője a cikkben [32] társszerzőként vett részt konzulensei és Dr. Szaszák György mellett.

5 Eredmények

Ebben a fejezetben ismertetem a tanításhoz használt hálózati architektúrákat, a tanítás alapvető lépéseit, és az eredmények összehasonlítását a különböző architektúrák között. Az első részben a hálózatok tanítása során a legjobb elért RMSE (Root Mean Square Error [33], négyzetes hibafüggvény) hibaértékek kerültek a táblázatokba.

5.1 Egy hálózatos architektúra eredményei

A TDK kutatómunka teljes ideje alatt több mint 200 különálló tanítást végeztem el. Számos különféle elrendezést, optimalizáló eljárást, hiperparaméter-beállítást vizsgáltam, egy részüket szakirodalomban található példák, más részüket saját intuíció alapján. Ezekből rengeteg egyéb konklúzió született, azonban terjedelmi okokból itt a legjobb architektúrák mellett csak a fontosabb konklúziókat említem meg. A következő táblázatban szereplő értékek esetén a nem felsorolt hiperparaméterek a következők voltak:

- *Nemlinerális függvény az elemi neuronokban:* ReLU, az utolsó rétegben szigmoid
- *Learning decay*: 5e-6 / iteráció
- *Tanítás ideje:* hálózatnagyságtól függően 16-22 óra (rendszerfüggő)
- *Early stopping:* 50 változatlan hibájú interáció után

A fentebb felsorolt hiperparaméterek optimalizálásával többet azért nem foglalkoztam, mert vagy a tapasztalatok szerint nagyon kis hatásuk volt a végső eredményre (pl: learning decay), vagy pedig az alternatívák szignifikánsan (hullámformában hallhatóan is) rosszabbul teljesítettek (pl: ReLU nemlineáris függvény szigmoidra vagy tanh() függvényre való cseréje.)

Rétegek száma	Neuron egy rétegben	Tanítási ráta	Momemtum parameter	Batch méret	RMSE a teszt adatbázison
3	3500	0.050	0.9(Nesterov)	32	0.00401
4	3000	0.08	0.85(Nesterov)	32	0.00416

3	3179	0.0678	0.9(Nesterov)	25	0.00293
3	3500	0.0714	0.9(Nesterov)	25	0.00325

4. Táblázat – egymodelles (egy hálózatos) rendszer legjobb eredmények

A TDK kutatómunka kezdetén az első eredmények a táblázatban felsoroltak elrendezéseknél legalább 1, de előfordult, amikor 2 nagyságrenddel nagyobb hibát eredményeztek. Ezekből még nem sikerült érthető beszédet generálni képes rendszert létrehozni. A legkorábbi tanítások még nem tartalmaztak ReLU aktivációs függvényt (szigmoid, illetve tanh() függvény volt a nemlinearitás). A ReLU bevezetése jelentős javulást eredményezett, ez a megfigyelés teljesen összhangban van a nemzetközi szakmai publikációk megállapításaival [11].

A batch méret fokozatos csökkentése, bár jelentős többletszámítás-igényt, és ezzel az egy tanításhoz szükséges idő jelentős növekedését hozta magával (az időközben megerősített hardver környezetben is), arányaiban mégis megéri a kompromisszum, mert a hiba sokkal jelentősebben csökkent, mint ahogy a tanítási idő nőtt. A tapasztalatok azt mutatják, hogy a batch méret további csökkentése a jelenlegi (20-50) szintről már nem javít, sőt, inkább ront a hibaértéken.

A rétegek száma és a rétegenkénti neuronszám (minden rétegben azonos neuronszámot alkalamaztam) hiperparaméterekkel kapcsolatos megállapítások is megfeleltethetők a szakirodalomban találhatókkal: jelen adatbázissal az 5-6 rétegnél nagyobb, illetve a rétegenkénti néhány ezer neuronnál többet tartalmazó MLP hálózatokat nehéz tanítani, illetve az eredmények nem lesznek olyan jók, mint a kevesebb réteget tartalmazó változatok. A fenti táblázatból látható, hogy itt 3 és 4 körül sikerült egy optimumot találni a rétegnagyságot illetően, neuronszám esetén pedig 3000 és 3500 körül voltak a legjobb eredmények.

5.2 „Hangsúlykiemelt” architektúra eredmények

Az aggregált modellben kizárólag az alapfrekvencia került tanításra. A következő táblázatban felsorolt hiba értékek tehát közvetlenül nem összevethetők az előző pontban felsorolt elrendezésekkel, hiszen más a hálózat kimenete. A táblázatban a legjobb kombinált összeállítás

tag neurális hálózatainak eredményei láthatók. A hálózati paraméterek úgy lettek beállítva, hogy az 4 tag hálózat tanítási ideje közel egyenlő legyen egy nagy, átlagos hálózat tanítási idejével.

Tag hálózat azonosítója	Neuron egy rétegben	Tanítási ráta	Momemtum paraméter	Batch méret	RMSE a teszt adatbázison
0	1000	0.03	0.9(Nesterov)	25	0.01280
1	1000	0.03	0.9(Nesterov)	25	0.01674
2	1000	0.03	0.9(Nesterov)	25	0.01369
X	1000	0.03	0.9(Nesterov)	25	0.01027

5. táblázat – „hangsúlykiemelt” rendszer legjobb eredmények

A következő, ábrákat tartalmazó alfejezetben az aggregált architektúra pozitív hatásai szemléletesebben láthatók.

5.3 Időzítést becslő hálózat eredmények

A következő táblázatban a kedvező tanítási sebesség (8-10 perc egy tanítás lefolytatása) miatt manuális beállításokkal nem kísérleteztem, így itt kizárólag a hiperparaméter-optimalizálás eredményei olvashatók.

Rétegek száma	Neuronok száma egy rétegben	Tanulási ráta	Dropout	Batch méret	RMSE a teszt adatbázison
4	253	0.0283	0.2626	21	0.0003207
2	104	0.0097	0.2894	15	0.0002175

2	349	0.0121	0.2513	9	0.0002171
2	121	0.0144	0.3332	9	0.0001280

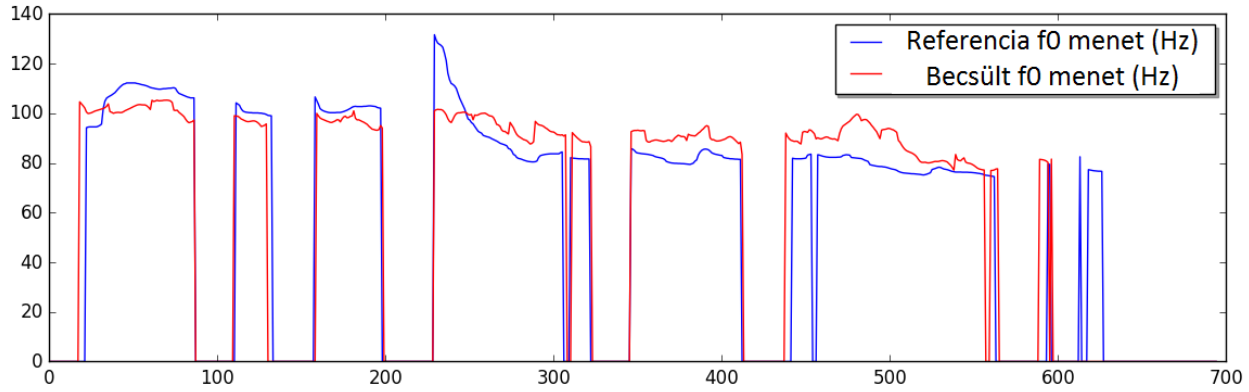
6. Táblázat – időzítés becslő hálózat legjobb eredmények

Érdemes megfigyelni az hiperparaméterek nagy szórását, és ehhez képest a hibaértékek közötti rendkívül kis különbséget. Voltak más – táblázatban nem felsorolt – mintapontok, ahol a hibaérték a táblázatban szereplő értékek 40-50-szerese volt. Ez arra utal, hogy itt a hibafelületek cseppet sem „simák”, azaz egy rossz irányba történő kezdeti lépés után már nem tud a rendszer egy jó minőségű minimumhelyre megérkezni.

Ez a konklúzió cseppet sem meglepő, ha figyelembe vesszük, hogy ez az adatbázis kb. 20-szor kisebb mintaszámban, mint az alapfrekvencia és spektrális paraméterek becslésére használt (itt egy beszédhang egy adat, ellentétben a „nagy” adatbázisnál, ahol keretekkel dolgoztam, 1 beszédhang pedig akár 20-25 keretet is jelenthetett). A kisebb adatbázisoknál a tanítás közbeni mintavételezés különbségei jóval erősebben jelentkeznek, mint a „nagy” adatbázisnál, ahol ezek a hatások kiátlagolódnak.

5.4 Eredmények vizualizációja

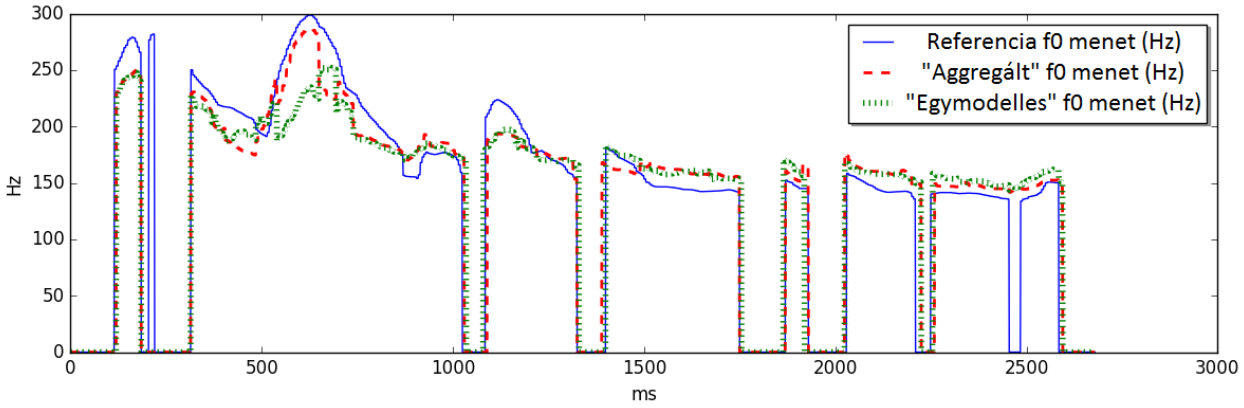
Bonyolult bemenettel és kimenettel rendelkező hálózatok esetén az eredmények látványos megjelenítése, összehasonlítása külön megoldandó feladat. A konkrét hibaérték, melyek a hálózat az adott hiperparaméter-beállítással elér, csak korlátozottan alkalmas a tényleges teljesítmény felmérésére. Mindig érdemes más ellenőrző módszereket, például szubjektív, meghallgatásos tesztet, vagy grafikus eredménymegjelenítést is használnunk. A következőkben az eredmények közül az 1 dimenziós alapfrekvencia függvény viselkedését mutatom be (az alapfrekvencia függvény viselkedése sokkal szemléletesebb, mint a spektrális paramétereké). Először az egymodelles neurális hálózat működését a referencia kimenetekkel összehasonlítva, majd utána egy másik ábrán a „hangsúlykiemelt” aggregált modell teljesítményét lehet megvizsgálni a referenciával és az egymodelles rendszer eredményeivel.



7. *ábra - Hangfájl alapfrekvencia becslése, példa . x tengely: minták, y tengely: frekvencia érték (Hz)*

A 7. ábrán látható, hogy az egyik legfontosabb tulajdonságot, az enyhén ereszkedő tendenciát sikeresen követte le a hálózat. A szubjektív teszteken az ilyen mértékű offszet jellegű eltérések (amennyiben nem túl nagyok) várhatóan nem fognak túl nagy problémát okozni a minőségben, hiszen ez csupán azt jelenti, hogy a beszélő hangja egy kicsit magasabb, vagy alacsonyabb lesz. Azt, hogy a piros görbének hol kell nullát illetve nem nullát adnia, szintén ugyanazon tanításkor becsült bináris zöngés/zöngétlen (flag) bit dönti el. (Ennek a becslése sem száz százaléking pontos, az átlagos pontosság ebben az esetben 99% körül volt). Ez az eltérés okozza azokat a szakaszokat, ahol a kék (referencia) görbe nulla értéket ad, a piros (becsült) nem nullát.

Az aggregált modell esetén ábrákon hasonlítom össze a rendszer teljesítményét a referenciával és az egymodelles rendszerrel. Itt az ábrákon három görbe lesz: kék színnel a referencia, zöld színnel a legjobb egymodelles elrendezés által becsült alapfrekvencia-függvény, piros színnel pedig az aggregált modell eredménye. A következő ábra jól mutatja, hogy mely esetekben nyújthat jobb teljesítményt a több szakértőből álló modell, mint az egymodelles elrendezés.



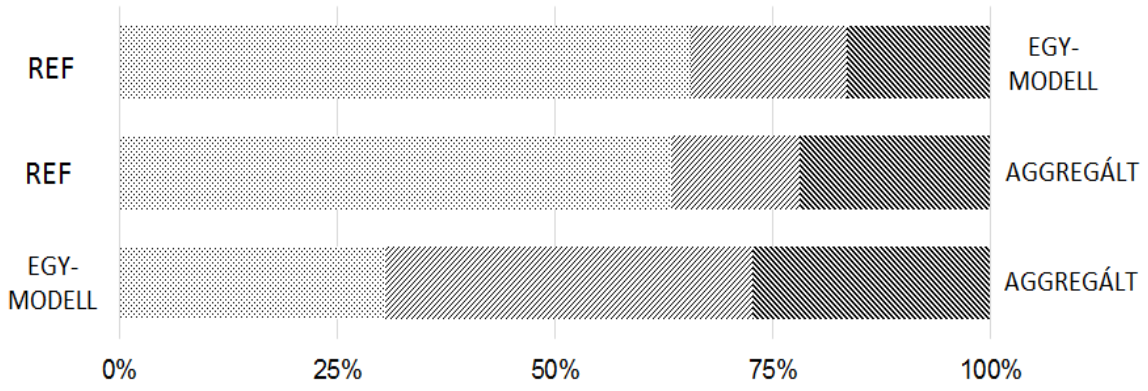
8. ábra - aggregált és egyelemes modell összehasonlítás

A 8. ábrán a legérdekesebb tartomány az 500 ms és 700 ms között van, itt látszik, hogy a nagyon hangsúlyos részen az erős hangsúlyra tanított „szakértő” tag neuronhálózat „belép”, és sokkal kevésbé átlagol, mint az egymodelles elrendezés.

5.5 Meghallgatásos tesztek

A gépi beszédkeltésnél – mint minden ember-gép interfészt megvalósító rendszernél – kiemelkedő jelentősége van a szubjektív teszteknek. Az ilyen tesztek során, amennyiben azok független tesztelőktől származnak, a készítők fontos tapasztalatokat szerezhetnek a rendszerük erősségeiről és gyengéiről. A neurális hálózatok által előállított hibaértékek, sőt még az generált ábrák is csak korlátozottan képesek előre jelezni azt a hatást, ami végső soron a használatkor javulásként vagy romlásként érzékelődik a felhasználókban. A szubjektív teszt ilyenkor is segíthet.

A TDK dolgozat készítésekor a hangsúly becslésének javítását szolgáló aggregált modell került összehasonlításra a legjobb egymodelles elrendezéssel, illetve a referencia hangfájlokkal. A tesztelők száma 16 fő volt (5 nő és 11 férfi). A tesztelőknek arra a kérdésre kellett válaszolniuk, hogy a párban lejátszott rövid hangfájlok közül (melyek egy-egy generált mondatból álltak) melyiket érzik természetesebbnek a hangsúly szerint. Az eredmények a következő ábrán láthatók:



9. ábra – szubjektív teszt eredmények

A 9. ábrán az egyes teszt eredmények akkor kerülnek a bal, vagy a jobb sávba, ha az adott tesztelő azt természetesebbnek találta. Középre kerül a teszt eredmény, ha ugyanolyannak találta a két hangfelvételt a tesztelő a természetesség szempontjából.

A felső sorban a referencia és egymodelles rendszer összehasonlítása látható, a nagy, második sorban pedig a referencia és az aggregált modell összehasonlítása szerepel. Az aggregált modellhez tartozó sáv hosszabb, mint az egymodelles rendszer sávja. A legalsó sávban az aggregált és az egymodelles elrendezés sávja közel azonos méretű.

Ezek alapján látszik, hogy a kombinált modellt kissé jobbra értékelték, ha az a természetessel van összehasonlítva, viszont az egymodelles és a kombinált modell közötti összehasonlításnál nem mutatható ki statisztikailag szignifikáns különbség.

6 Összefoglalás

A TDK kutatómunka során sor került egy mély neurális hálózat alapú beszédkeltő rendszer bemutatására. A rendszer egyes részegségeinek (alapfrekvencia-becslő, spektrális paraméter becslő, időzítés-becslő) elkészítése részletesen bemutatásra került. Az MLP hálózatok segítségével sokféle hiperparaméter beállítás mellett megvizsgáltam, hogyan érhető el a legjobb eredmény ezen részegységek megvalósítására. Sztochasztikus hiperparaméter-optimalizálással kíséreltem meg a manuális keresésnél jobb paraméter-beállítást találni. Bemutattam ezen kívül egy módszert, amellyel az alapfrekvencia-becslés javítható, ezáltal jobb beszéd-dinamika, és végső soron jobb felhasználói élmény érhető el.

A jelenlegi tesztrendszer fejlesztése terén két jelentős fejlesztési irány képzelhető el: az egyik az adatreprezentáció további finomhangolása, fejlesztése. Itt jelenleg javítandó tényezőnek felsorolható a zöngétlen hangok és a hangfájl eleji és végi szünetek közötti összemosás csökkentése, és az interpoláció által bevitt hiba minimalizálása. A másik fő fejlesztési irány a neurális hálózati architektúrák további hangolása a megoldandó feladathoz. Jelenleg gyorsan megvalósíthatónak látszik a [34] által is említett ReLU modellek továbbfejlesztéseinek (például az ún. Leaky ReLU) bevezetése. Másik jelentős irány a rekurrens hálózatokkal történő tanítás (pl. Long Short-Term Memory, azaz LSTM hálózatok) az MLP kiváltására vagy kiegészítésére. Ezek a módosítások további javulást hozhatnak az eddigi eredményekhez képest –várhatóan hosszabb tanítási idők mellett.

Irodalomjegyzék

[1] Tóth, B., & Németh, G. (2008). Rejtett Markov-modell alapú mesterséges beszédkeltés magyar nyelven. LXIII. köt, 2-6.

[2] Zen, Heiga, Andrew Senior, and Mike Schuster. "Statistical parametric speech synthesis using deep neural networks."International Conference on.Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE 2-3.

[3] Dudley, H. (1940). The Vocoder—Electrical Re-Creation of Speech. Journal of the Society of Motion Picture Engineers, 34(3), 272-278.

[4] Liberman, A. M., Cooper, F. S., Shankweiler, D. P., & Studdert-Kennedy, M. (1967). Perception of the speech code. Psychological review, 74(6), 431.

[5] Sproat, R. W. (1997). Multilingual text-to-speech synthesis. KLUWER academic publishers. 29-38.

[6] Markó Alexandra – Bóna Judit 2012. Fundamental frequency patterns: The factors of age and speech type. In: Calamai, Silvia – Celata, Chiara – Ciucci, Luca (eds.) Proceedings of 'Sociophonetics, at the crossroads of speech variation, processing and communication'. Edizioni della Normale. Pisa. 45–48.

[7] Rosenblatt, Frank. "The perceptron: a probabilistic model for information storage and organization in the brain." Psychological review 65.6 (1958): 386.

[8] Altrichter Márta, Horváth Gábor, Pataki Béla, Strausz György, Takács Gábor, Valyon József : Neurális Hálózatok. 2006 Hungarian Edition Panem Könyvkiadó Kft., Budapest, 81-112.

[9] Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. Science, 313(5786), 504-507.

[10] Zeiler, M. D., Ranzato, M. A., Monga, R., Mao, M., Yang, K., Le, Q. V., ... & Hinton, G. E. (2013, May). On rectified linear units for speech processing. International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE. IEEE, 3517-3521.

[11] Glorot, X., Bordes, A., & Bengio, Y. (2011, April). Deep Sparse Rectifier Neural Networks. In Aistats (Vol. 15, No. 106, p. 275).

[12] Hecht-Nielsen, R. (1989, June). Theory of the backpropagation neural network. 1989. IJCNN., International Joint Conference on Neural Networks , IEEE, 593-605.

[13] LeCun, Y. A., Bottou, L., Orr, G. B., & Müller, K. R. (2012). Efficient backprop. In Neural networks: Tricks of the tradem Springer Berlin Heidelberg, 9-48.

[14] Sutskever, I., Martens, J., Dahl, G., & Hinton, G. (2013). On the importance of initialization and momentum in deep learning. In Proceedings of the 30th international conference on machine learning (ICML-13),1139-1147.

[15] Senior, A., Heigold, G., & Yang, K. (2013, May). An empirical study of learning rates in deep neural networks for speech recognition. In 2013 IEEE International Conference on Acoustics, Speech and Signal Processing (pp. 6724-6728). IEEE.

[16] Tieleman, T., & Hinton, G. (2012). Lecture 6.5-rmsprop. COURSERA: Neural networks for machine learning.

[17] Kingma, D., & Ba, J. (2014). Adam: A method for stochastic optimization.arXiv preprint arXiv:1412.6980.

[18] Prechelt, L. (1998). Automatic early stopping using cross validation: quantifying the criteria. Neural Networks, 11(4), 761-767.

[19] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research, 15(1), 1929-1958.

[20] Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. Journal of Machine Learning Research, 13(Feb), 281-305.

[21] Cortes, C., & Vapnik, V. (1995). Support-vector networks. Machine learning,20(3), 273-297.

[22] Camacho, Arturo. SWIPE: A sawtooth waveform inspired pitch estimator for speech and music. Phd. University of Florida, 2007.

[23] Csapó, T. G., Németh, G., & Cernak, M. (2015). Residual-Based Excitation with Continuous F0 Modeling in HMM-Based Speech Synthesis. In Statistical Language and Speech Processing (pp. 27-38). Springer International Publishing.

[24] Sutskever, I., Martens, J., Dahl, G., & Hinton, G. (2013). On the importance of initialization and momentum in deep learning. In Proceedings of the 30th international conference on machine learning (ICML-13),1139-1147.

[25] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Vanderplas, J. (2011). Scikit-learn: Machine learning in Python. The Journal of Machine Learning Research, 12, 2825-2830.

[26] Oliphant, T. E. (2007). Python for scientific computing. Computing in Science & Engineering, 9(3), 10-20.

[27]

[28] Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., ... & Bengio, Y. (2010, June). Theano: a CPU and GPU math expression compiler. In Proceedings of the Python for scientific computing conference (SciPy) (Vol. 4, p. 3).

[29] Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. Computing in science and engineering, 9(3), 90-95.

[30] Chollet, F. (2015). Keras. GitHub repository: https://github. com/fchollet/keras.

[31] Bergstra, J., Yamins, D., & Cox, D. D. (2013). Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms. In Proceedings of the 12th Python in Science Conference (pp. 13-20).

[32] Szaszák, G., & Beke, A. (2012). Exploiting prosody for syntactic analysis in automatic speech understanding. Journal of Language Modelling, (1), 143-172.

[33] Tóth, B. P., Kis, K. I., Szaszák, G., & Németh, G. (2016, August). Ensemble Deep Neural Network Based Waveform-Driven Stress Model for Speech Synthesis. In International Conference on Speech and Computer (pp. 271-278). Springer International Publishing.

[34] Zhang, G., Patuwo, B. E., & Hu, M. Y. (1998). Forecasting with artificial neural networks:: The state of the art. International journal of forecasting, 14(1), 35-62.

[35] LeCun, Yann; Bengio, Yoshua; Hinton, Geoffrey (2015). "Deep learning". Nature 521: 436–444.

[36] Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. Neural networks, 2(5), 359-366.

[37] Lieberman, P., & Michaels, S. B. (1962). Some aspects of fundamental frequency and envelope amplitude as related to the emotional content of speech. The Journal of the Acoustical Society of America, 34(7), 922-927.

[38] Zheng, F., Zhang, G., & Song, Z. (2001). Comparison of different implementations of MFCC. Journal of Computer Science and Technology,16(6), 582-589.

[39] Zheng, F., Song, Z., Li, L., Yu, W., Zheng, F., & Wu, W. (1998). The distance measure for line spectrum pairs applied to speech recognition. In ICSLP.

[40] Olaszy, G. (2013). Precíziós, párhuzamos magyar beszédadatbázis fejlesztése és szolgáltatásai [Development and services of a Hungarian precisely labeled and segmented, parallel speech database]. Beszédkutatás 2013 [Speech Res. 2013], 261-270.

[41] Stein, J. Y. (2000). A Computer Science Perspective. Wiley. 383-386.

Függelék

A felhasznált beszédhang készlet

(68 db beszédhang)

"a","a1","e","e1","i","i1","o","o1","o2","o3","u","u1","u2","u3","b","bb","c","cc","cs","cscs","d","dd","dz","dzdz","dzs","dzsdzs","f","ff","g","gg","gy","gygy","h","hh","j","jj","k","kk","l","ll","m","mm","n","nn","ny","nyny","p","pp","q","qq","r","rr","s","ss","sz","szsz","t","tt","ty","tyty","v","vv","zs","zszs","pau","ssil","esil","x"

A neurális hálózat kimeneti és bemeneti paramétereinek listája

Bináris bemenetek:

(beszédhang darabszám (itt 68) * karakterek száma (itt 5)) darab bináris bemenet

- p_1: A kiejtés szerinti átírásban az éppen aktuális hangkaraktert (p3) kettővel megelőző hangkaraktert jelöli.
- p_2: A kiejtés szerinti átírásban az éppen aktuális hangkaraktert (p3) megelőző hangkaraktert jelöli.
- p_3: Az éppen aktuális hangkaraktert jelöli, erre vonatkozik az összes többi azonos sorban lévő címke.
- p_4: Az aktuális hangkaraktert (p3) követő hangkaraktert jelöli.
- p_5: Az aktuális hangkaraktert kettővel követő hangkaraktert jelöli

Numerikus bemenetek:

(21 darab elemi paraméter és a speciális bemenetek)**:**

- p_6: A jelenlegi beszédhang szótagon belüli pozíciója, sorszáma.
- p_7: A jelenlegi beszédhang hátulról számított pozíciója a szótagon belül.
- *a2*: Azt jelöli, hogy az előző szótagon van-e mondathangsúly (0-5) (0: nincs hangsúly, 1-3: gyenge hangsúly, 4-5: erős hangsúly)
- *a3*: Az előző szótagban szereplő hangkarakterek száma.
- b2: Azt jelöli, hogy a jelenlegi szótagon van-e mondathangsúly (X vagy 0-2)
- b_3: A jelenlegi hangkarakterhez tartozó szótagban szereplő hangkarakterek száma.

- b_4: Az éppen aktuális hangkarakterhez tartozó szótag pozíciója, sorszáma a jelenlegi szóban.
- b_5: Az éppen aktuális hangkarakterhez tartozó szótag hátulról számított pozíciója, sorszáma a jelenlegi szóban.
- b_6: A jelenlegi szótag pozíciója, sorszáma az adott tagmondatban.
- b_7: A jelenlegi szótag hátulról számított pozíciója, sorszáma az adott tagmondatban.
- c2: Azt jelöli, hogy a következő szótagon van-e mondathangsúly (0-5) (0: nincs hangsúly, 1-3: gyenge hangsúly, 4-5: erős hangsúly)
- c3: A következő szótagban lévő hangkarakterek száma.
- d1: Az előző szóban szereplő szótagok száma.
- e1: A jelenlegi szóban szereplő szótagok száma.
- e_2: A jelenlegi szó tagmondaton belüli pozíciója, sorszáma.
- e_3: A jelenlegi szó tagmondaton belüli hátulról számított pozíciója, sorszáma.
- f_1: A következő szó szótagszáma.
- h_1: A jelenlegi tagmondat által tartalmazott szótagok száma.
- h_2: A jelenlegi tagmondat szavainak száma.
- h_3: A jelenlegi tagmondat mondatbeli pozíciója, sorszáma.
- h_4: A jelenlegi tagmondat hátrafelé számított mondatbeli pozíciója, sorszáma.
- j_1: A jelenlegi mondat által tartalmazott szótagok száma.
- j_2: A jelenlegi mondat által tartalmazott szavak száma.
- j_3: A jelenlegi mondat által tartalmazott tagmondatok száma.

Speciális bemenetként szerepelt az adott beszédhang hossza, illetve az adott f0 érték százalékos pozíciója az adott hangkarakteren belül.

Kimenetek:

(1 darab numerikus és 1 darab bináris, csak f0 tanítás esetén):

- *log(f_0)*: A becsült alapfrekvencia érték természetes alapú logaritmusa
- *zöngés/zöngétlen flag*: értéke 0, ha zöngétlen, 1, ha zöngés az adott beszédhang
- *25 db LSP paraméter:* (amennyiben ez is tanítva volt)

Adatbázis fájl részlet

(minta)

2681716 3227280 v^i-l+a1=g@1_2/A:x_3_2/B:x-0-2@2-3&3-7#0-0$1-0!x-x;1-6|a1/C:x+0+2/D:1/E:4@2+3/F:2/G:x_x/H:9=4!1=3/I:5_2/J:22+9-3/L:0&x$x!x#x+x;x;x/M:x!x!x+

SWIPE eredmény részlet

(minta)

0.2700000 129.6627

0.2750000 129.4287

0.2800000 129.4287

Neurális hálózat bemeneti adat

(egy darab minta adatvektor, az utolsó két elem a kimenet, az előtte lévők a bemenetek)

[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0.8703379261065693,1.148979,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,21,10,1,129.38810370370373,0,4.18979,0.098476,0.215879,0.37659,0.495928,0.613609,0.715972,0.841484,0.921349,1.057,1.19007,1.32619,1.4905,1.62821,1.77101,1.87374,1.98645,2.09991,2.20904,2.36258,2.48141,2.59542,2.73667,2.86412,2.98914]