Önálló laboratórium beszámoló

Távközlési és Médiainformatikai Tanszék

Készítette:	**Hegedüs Nóra**
Neptun-kód:	**BCLHKC**
Ágazat:	**Infókommunikáció**
E-mail cím:	**norika200208@gmail.com**
Konzulens(ek):	**Dr. Zainkó Csaba**
E-mail címe(ik):	**zainko@tmit.bme.hu**

Téma címe: Mélytanulás alapú beszédgenerálás

Feladat

Féléves feladatomként mélytanulás alapú beszédgenerálással foglalkoztam. Feladataim közé tartozott elsősorban a beszédgenerálás folyamatának megismerése, egy modell kipróbálása az LJ Speech hangadatbázissal, ezek után egy másik angol nyelvű hangadatbázis keresése, és ennek felhasználásával a modell betanítása több beszélőre is, majd különböző módosítások elvégzése és ezek tesztelése.

2023/2024. 2. félév

1. A laboratóriumi munka környezetének ismertetése, a munka előzményei és kiindulási állapota

1.1 Bevezető

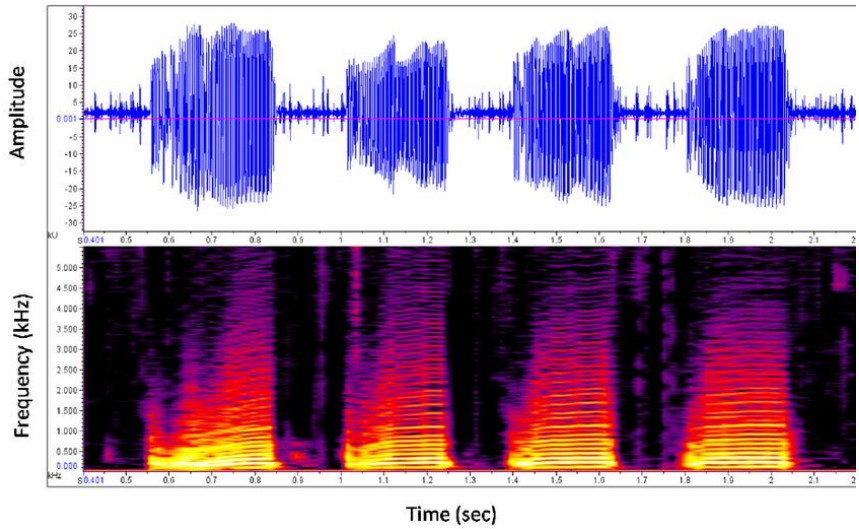
Azért esett a választásom erre a témára, mert a Témalaboratórium tárgy keretein belül már korábban foglalkoztam mesterséges intelligenciával, és megismerkedtem annak alapjaival, valamint a neurális hálózatok felépítésével. A képek osztályozása egy megfelelő kezdő projekt volt számomra, de szerettem volna nagyobb kihívást magamnak, és úgy éreztem, hogy a hangfeldolgozás ezt nyújthatja nekem. Mivel korábban még nem mélyedtem el a mesterséges intelligencia ezen ágában, ezért alaposan utánajártam a témának. A beszédszintézis egy olyan terület, ahol a számítógép szövegeket alakít hangzó szöveggé. A szintetizált hang általában emberi hanghoz hasonlít, gyakran pedig olyan hangot ad vissza, amelyet egy konkrét személy hangmintái alapján tanítottak be. Bár néhány új algoritmus már lehetővé teszi, hogy olyan hangokat hozzon létre, amelyek nem szerepeltek a tanítási adatai között, vagy teljesen új hangokat is képes reprodukálni.

1.2 Elméleti összefoglaló

Annak érdekében, hogy a leírt szövegből beszéd keletkezzen, ami végeredményben az emberi beszédhez hasonló lesz, ahhoz számos fontos lépésen kell végigmenni. Először is, a nyers szöveget beszéddé kell alakítani a normalizáció folyamatával. Ez a lépés magába foglalja a rövidítések és mozaikszavak kibontását (például Dr. → doctor), valamint a számok (tizedestörtek és római számok is), az égtájak és a webcímek átalakítását beszélt formába. Ez általában egy manuális folyamat a rengeteg szabály és reguláris kifejezés miatt, amit nagyon nehéz lenne automatizálni, illetve különböző nyelvek között nem lehet általánosítani. Erre jó példa a dátum, amit más sorrendben és teljesen máshogyan mondanak ki az angolok, mint mi, magyarok.

Ezt követi az úgynevezett G2P (Grapheme to Phoneme - grafémából fonémává) folyamat, ahol a szöveg legkisebb egységei (karakter/graféma) a beszélt nyelv legkisebb egységeire alakulnak át, ami a fonéma. A normalizációhoz hasonlóan ez is manuális folyamat. Néhány nyelv (például német és spanyol), fonetikus, ami azt jelenti, hogy a leírt karaktereik mindig ugyanúgy hangzanak, ezeknél a nyelveknél ez a folyamat felesleges. Azonban az angol nem egy fonetikus nyelv, mert a karakterek kiejtése attól függ, hogy milyen szóban vannak és ezenkívül pedig néhány fonéma több karakter által van reprezentálva. Ezenkívül az angol tartalmaz heteronímákat is, amik ugyanúgy betűzött szavak különböző jelentéssel vagy kiejtéssel (például a „read” szó kiejtése attól függ, hogy múlt vagy jelen időben használják-e). Két fonetikus ábécé van, amit a leggyakrabban használnak erre a célra, az IPA (International Phonetic Alphabet-nemzetközi fonetikus ábécé) és az ARPABET (Advanced Research Projects Agency - fejlett kutatási projektek ügynöksége). Ezek közötti különbség látható az „Arpabet to IPA” című oldalon [1].

Ezek után a spektrogram szintézis következik, ahol a fonémák spektrogrammá alakulnak, ami a beszélt hangot reprezentálja. A hangot általánosságban egy hosszú számsorozattal fejezik ki, amelyek az amplitúdót mutatják az idő függvényében. Azt, hogy milyen gyakran rögzítik a hangmintákat, mintavételezési sebességnek nevezik. Általában tizenhatezer Hertz, 22050 Hertz és 44100 Hertz szokott lenni, minél magasabb ez az érték, annál pontosabb és jobb minőségű lesz a hang.



1. ábra Hang reprezentálása amplitúdóval és frekvenciával [2]

Az (1) -es ábrán látható felső kép egy hangot ábrázol idő és amplitúdó dimenziókban, ami általában hasznos lehet felvételnél és hallgatásnál, azonban nem optimális feldolgozásra. Abban az esetben az alsó képen látható reprezentáció előnyösebb, ahol az idő és a hang frekvenciája van rögzítve. Ez a megközelítés részletesebb információt nyújt a hangról. Konkrétan, kisebb időintervallumonként (audio frame) Fast Fourier transzformáció segítségével kiszámoljuk minden ilyen egységre az energiáját és az amplitúdóját és a fázisát (bár ez utóbbit nem használjuk fel). Ezek után az eredeti frekvenciák mel-skálával vannak összekapcsolva, átlapolva egy háromszögű szűrővel, amiből megkapható a mel-spektrogram. A mel-skála használata azért előnyös, mert jobban illeszkedik az emberi hang felfogásához, míg a Hertz egy egyszerű fizikai mértékegység. Ez a folyamat gazdagabb információkat nyújt a hangokról, és segít a hangfeldolgozási feladatok pontosabb végrehajtásában. Azokon az előnyökön kívül, amiket eddig felsoroltam, van még néhány oka, amiért jobb a spektrogramot használni. Az egyik ok az, hogy a számításokban sokkal hatékonyabb, mint a nyers adatok feldolgozása. Ezzel a módszerrel nagyjából ötször kisebbé tudjuk tenni az adatainkat. Egy másik ok pedig az az, hogy egy nagyon hosszú egydimenziós szekvencia helyett sokkal rövidebb többdimenziós szekvenciát használunk.

Majd végezetül pedig a hangszintézissel fejeződik be az egész folyamat, ahol is a spektrogramból végeredményül hang lesz. Ezt más néven spektrogram inverziónak is hívják, mivel elméletben inverz Fourier transzformáció használatával rekonstruálható lenne a hang, azonban ez ennél egy kicsit komplikáltabb feladat. Az egyik probléma, hogy a mel-spektrogram nem tartalmazza a fázis információkat, a másik pedig, hogy a jósolt spektrogram nem tökéletes. Nagy valószínűséggel tartalmaz zajokat vagy más természetellenes jellemzőket. Habár a fázis információkat meg lehetne közelíteni a Griffin-Lim algoritmus segítségével, azonban ez még mindig csak egy közelítés, ami sajnos nem oldaná meg az átalakítás problémáját. Ehelyett egy úgynevezett vocoder-t (beszédszintetizátor) szoktak alkalmazni erre, ami egy külön modell, ami arra van betanítva, hogy hangot generáljon. Ez a modell meg tudja tanulni, hogy pontosabban végezze el a hang rekonstrukcióját. A leggyakoribban használt beszédszintetizátorok a HiFi-GAN, és a WaveGlow, amikről az alábbi weboldalakon lehet több információt megtudni [3] [4]. Illetve a beszédgenerálás folyamatairól és működéséről többet lehet olvasni a Text-to-Speech 101: The Ultimate Guide nevű weboldalon [5].

1.3 A munka állapota, készültségi foka a félév elején

Ahogyan azt már korábban is említettem a bevezetőben, az előző félév során a

Témalaboratórium tárgy keretein belül már foglalkoztam mélytanulással. Ebben az időszakban egy képosztályozó programot fejlesztettem, ahol teljesen az elejétől, az előkészítéstől egészen a tesztelésig, lépésenként kellett egy neurális hálózatot felépítenem. Ez a projekt segített, hogy jobban megértsem, hogy milyen elemekből áll egy neurális hálózat, és hogy milyen lépésekből áll egy tanítási folyamat. Ez megfelelő alapot adott az idei projektem elkezdéséhez.

2. Az elvégzett munka és eredmények ismertetése

2.1 A munkám ismertetése

Irodalomkutatás

A féléves munkámnak a fentebb lévő feladatleírásban meghatározott feladat megvalósítását tűztem ki magam elé célul és azt végül sikeresen meg is valósítottam. Első lépésként egy átfogó és alapos irodalomkutatást végeztem annak érdekében, hogy megérthessem a beszédgenerálás folyamatát. Különféle forrásokat tanulmányoztam, beleértve az NVIDIA oldalán található Fastpitch és HiFi-GAN modellek dokumentációit is. Ezt követően az NVIDIA által kínált TTS-hez (text-to-speech, szövegből beszéd) kapcsolódó gyakorló feladatai közül választottam ki a „NeMo TTS Primer" nevű tutorial-t [6], amit a Google Colab platformon futtattam. Ez a tutorial lépésről lépésre vezetett végig a szövegből hanggenerálás folyamatán, és lehetővé tette számomra, hogy megismerjem a különböző kiegészítőket és lehetőségeket a témában. Minden egységhez részletes leírás készült, valamint a legtöbb részt szemléletes ábrák és példa kódok kísértek.

A munkát egy távoli szerveren végeztem el, amelyre ssh (távoli bejelentkezés) segítségével csatlakoztam. Ezen a kapcsolaton keresztül helyben futtattam a szkripteket, illetve távolról Jupyter notebook-on dolgoztam. Egy új projektet létrehozva ezen a platformon is végrehajtottam a „Nemo TTS Primer" tutorial-t.

Miután megismerkedtem a különböző felületekkel, illetve kipróbáltam néhány gyakorlati feladatot, el tudtam kezdeni a tényleges munkámmal foglalkozni. Ezt a „DeepLearningExamples" nevű Github repository klónozásával kezdtem [7]. Ebben a repository-ban különböző mélytanulással kapcsolatos témákban (például osztályozás, ajánlórendszerek, előrejelzés és detektáció) találhatók példa kódok, amik alapot adnak egy-egy ilyen feladat elsajátításához. Ezek közül én a beszédszintézishez tartozó szkripteket használtam fel. A „scripts" mappán belül található „download_dataset.sh" szkript segítségével letöltöttem a feladathoz tartozó hangadatbázist, ami nagyjából tizenegyezer mondatot tartalmazott, amelyek mind stúdiókörnyezetben voltak rögzítve.

Betanítás LJ Speech adatbázissal

A sikeres előkészítést követően, amit a „prepare_dataset.sh" futtatásával oldottam meg, el tudtam kezdeni a hangokkal a modell betanítását a kód paramétereinek megfelelően. Ezek a paraméterek úgy voltak beállítva, hogy ezer epoch-ig (iteráció) menjen, azaz ezer alkalommal iteráljon végig az adatokon, illetve száz epoch-okonként mentse el egy-egy checkpoint-ként (ellenőrzési pont) a tanítás akkori állását. Mivel a teljes tanítási folyamat hosszú időt vett volna igénybe, nagyjából huszonöt órát, ezért a korábbi checkpoint-ok közül (kétszázas és a háromszázas) használtam fel párat a teszteléshez. A teszthez az „inference_example.sh" fájlban kellett a „FASTPITCH_LJ" paraméter értékét kicserélni az általam betanított modellre, majd számomra meglepő módon már a tanítás ilyen korai szakaszában is teljesen jó minőségű és érthető hangokat sikerült generálnom.

Adatbázis keresése

Mindezek után elkezdtem keresni egy másik hangadatbázist, amivel az eredeti LJ Speech adatbázist helyettesíthettem volna. A CMU oldalán találtam egy megfelelőt, ahonnan letöltöttem egy brit akcentusú férfi beszélő által rögzített hangokat, amik szintén stúdiókörnyezetben voltak rögzítve [8]. Körülbelül ezeregyszáz mondat szerepelt az adatbázisban és tartozott hozzá egy szöveges fájl, amiben le volt írva, hogy melyik hangfájlban mit mondanak. Azonban az adatok előkészítése során felmerült néhány technikai kihívás. A mondat-hangfájl összerendeléseket tartalmazó fájlban néhány mondat nem szerepelt, ami a hangfájlok között megtalálhatóak voltak, emiatt manuálisan át kellett néznem az egész fájlt, hogy ki tudjam javítani az elcsúszást,

illetve át kellett alakítanom olyan formátumúra, ahogyan az LJ Speech fájlja is kinézett, különben nem tudta feldolgozni a program. Az új hangokkal volt még egy technikai probléma, példának okáért különböző mintavételezési sebesség. Az új hangok tizenhatezer Hertzzel lettek felvéve, a másik viszont 22 050 Hertzzel. Ezt sikeresen kezeltem egy rövid kódrészlettel, amiben az „ffmpeg" parancs segítségével átkonvertáltam őket. Így miután lecseréltem a „prepare_dateset.sh" fájlban a dataset-path (adatbázishoz vezető útvonal) és wav-text-filelists (hangfájl-mondat fájllista) paramétereket, hogy az én adatbázisomra, illetve a szöveg fájlomra mutassanak, sikeresen lefuttattam az adatokat előkészítő kódot.

Tanítás két beszélővel és eredményei

Miután ezen lépésekkel végeztem, a következő célom az volt, hogy a modellt két beszélővel tanítsam be, és képes legyek ezek között váltani. Ehhez először át kellett alakítanom a mondatokat tartalmazó szöveg fájlokat, hogy a mondatok végére odakerüljenek a beszélők azonosítói. Az LJ Speech-nek a nullás, míg a másiknak egyes azonosítót állítottam be, így később könnyen tudtam rájuk hivatkozni a tesztelés során. Ezt követően összegyűjtöttem mindkét beszélő hangfájljait egy közös mappába, hogy hozzáférjen a „train.sh" szkript, ezenkívül pedig a beszélők számát beállító paramétert lecseréltem kettőre. Ennek a modellnek a tanítását hagytam ezer iterációig futni, majd mindkét beszélőre kétszáz checkpoint-onként lefuttattam az „inference_example.sh" szkriptet. Majd az összes legenerált hangot meghallgattam, hogy értékelni tudjam, hogy melyik milyen minőséget ért el. Annak érdekében, hogy minél sokszínűbb legyen a teszt, öt különböző mondatot generáltattam minden alkalommal, ezek között szerepeltek kettő-három szavas mondatok, illetve összetettek is, mivel mind a rövid, mind a hosszú mondat generálása kihívás lehet.

Az (1) -es táblázatban összefoglaltam, hogy miket figyeltem meg egy-egy hang meghallgatásánál. A tapasztalatom az volt, hogy az LJ Speech adatbázison tanított beszélő már az elejétől kezdve teljesen érthető és egészen jó minőségű hangokat generált. Ezzel szemben a CMU oldalról letöltött beszélő eleinte teljesen érthetetlen beszédet hozott létre, és ez a későbbi checkpoint-tal sem lett sokkal jobb minőségű. Azonban fontos itt megjegyezni, hogy olyan mondatokkal teszteltem, amik teljesen újak voltak számára, és előfordulhat, hogy a szavak egy részével egyáltalán nem is találkozott. Erre abból következtetek, hogy voltak teljesen érthető szavak, míg mások csak zajnak hallatszottak. Emiatt megpróbáltam tesztelni egy olyan mondattal is, ami teljes egészében szerepelt a tanító adathalmazában, ebben az esetben a minőség valóban javult.

Ezt a nagy minőségbeli különbséget a két beszélő között elsősorban az befolyásolta a legjobban, hogy nagyságrendbeli eltérés volt a tanító adathalmazok számosságában, mivel az egyik tízezer, míg a másik csak ezer mondattal találkozott. Egy tanítás végeredményének minősége nagyban függ attól, hogy mennyi adatot látott a tanítása során, és azok mennyire voltak sokszínűek, különbözőek, mennyire fedték le az összes lehetséges kimenetet. Az emberi beszéd generálása egy rendkívül nehéz és összetett feladat, illetve egy-egy emberi hangnak számos meghatározó tényezője van. Ezeket nem lehet általánosítani, mert minden emberé egyedi. Példának okáért egy-egy betű kiejtését sem lehet generalizálni, mivel a legtöbb nyelvben, amelyik nem fonetikus nyelv (egy-egy betűt minden alkalommal ugyanúgy ejtenek ki), attól is függhet, hogy milyen szóban vannak. Ezenkívül a kiejtést a beszélő akcentusa még tovább tudja formálni. Ahogyan egy képosztályozási feladatnál, ahol a modell pontosabban tudja jósolni azokat az osztályokat, amelyekből nagyobb a tanító adathalmaz, úgy a beszédszintézisnél is az első beszélő vált pontosabbá és érthetőbbé, mert sokkal több adattal lett tanítva.

Checkpoint	LJ Speech	CMU
200	A legrövidebb mondat nem hallatszódott, illetve néhány mondatnál lemaradt a mondat eleje, de nagyon tiszta és érthető a hang	A hosszú és rövid mondatok nagyon rosszak, de a többiből már több minden volt érthető
400	Kisebb hibáktól eltekintve egészen használható hang	Kicsit visszaesett a minőség, sok dolog megint nem érthető, váltakozó a hang minősége
600	Most viszont majdnem minden mondat eleje lemaradt	A hosszabb mondatokat nem tudja jól generálni, sok dolog ugyanúgy nem érthető, mert szinte csak zaj lesz belőle
800	Az előzőekhez hasonló, tartja a minőséget	Általában a mondat eleje jól kezdődik, de a második fele már nem jó, főleg a hosszabbaknál
1000	Nagyon tiszta és érthető, talán még érezhetőbb hangsúlyozás	Egyáltalán nem hasonlít emberi hangra, még mindig elég rossz minőség

1. táblázat A beszélők értékelése

Azt sem szabad elfelejteni, hogy a modell hiperparaméterei (ami egy olyan beállítás vagy paraméter, ami befolyásolja egy gépi tanulási modell viselkedését, de nem azonos az adathalmazokban szereplő adatokkal vagy a modell paramétereivel) teljesen az LJ Speech adatbázisára voltak optimalizálva. A táblázatban az LJ Speech-hez tartozó oszlopban sokszor olvasható, hogy a tesztmondatoknál viszonylag sokszor nem hallatszik a mondat eleje és vége. Az is előfordult, hogy a nagyon rövid mondatból szinte nem hallatszott semmi. Ezt nagyon érdekesnek találtam, ezért megvizsgáltam a hangokat, amikkel tanult a modell, és azt a felfedezést tettem, hogy a legtöbb mondat is hasonlóképpen volt felmondva, azaz előfordult, hogy a mondat eleje és vége le volt vágva.

A fájl manuális szerkesztése egyáltalán nem felhasználóbarát, ezért, hogy könnyebbé és dinamikusabbá tegyem a különböző tesztek végrehajtását, létrehoztam egy Jupyter notebook projektet, amit bash kernellel indítottam el. Elsőként „echo” paranccsal bemásoltam egy tetszőleges mondatot egy szövegfájlba, majd úgy futtattam le a hanggeneráló fájlt, hogy paraméterként megadtam neki egy szöveges fájlt, amiből a mondatokat kell neki generálnia (vagy az a fájl, amibe az általunk megadott mondatot másoltuk, vagy akár egy másik is megadható), a beszélő azonosítóját, illetve a checkpoint-ot is. Ezeket így könnyen egy helyen át lehet írni, viszont ez még mindig nem megfelelő, mivel bash-el a hangfájl visszaolvasása és lejátszása nem megoldható.

Tanítás hét beszélővel és eredményei

Folytatásképp hét különböző beszélővel szerettem volna betanítani a modellemet. Ehhez először letöltöttem az összes beszélő hangfájljait az adott forrásoldalról, ahonnan a brit akcentusú férfi beszélő hangját töltöttem le. Ezeknél a beszélőknél is át kellett konvertálni megfelelő formátumúra a szöveges fájlokat, amikben a hangfájl-mondat párosítások szerepeltek. Mivel mindegyik beszélőnek megegyezett a hangfájljainak nevei, így azokat is át kellett neveznem egy rövid Python kód segítségével, ahol minden hangfájl nevének végére beszúrtam a

beszélőjének azonosítóját.

Még egy feladat volt, mielőtt neki állhattam volna az előkészítésnek és tanításnak, a hanganyagok átkonvertálása a megfelelő mintavételezési rátára. A hangok előkészítése után a tanítási folyamatra összpontosítottam, amely során némi módosítást kellett végrehajtanom a „train.sh" fájlban. A tesztet most is Jupyter környezetben hajtottam végre, de most már nem bash-el, hanem Python környezetben végeztem el. Ehhez az egész „inference_example.sh" fájl tartalmát részletekben átmásoltam egy újabb Jupyter notebook fájlba. Ahhoz, hogy ezt megtehessem, át kellett írni a bash szkriptet Python nyelvre. Az előző teszthez hasonlóan itt is kigyűjtöttem egy helyre a fontos beállítandó változókat. A legelejére tettem egy listát, ahova bármennyi mondatot lehet írni, amivel tesztelni lehet a modellt, ezeket pedig utána beleírtam egy fájlba. Ahogyan azt a kétbeszélős modellem tesztelésénél is tettem, itt is az elejére tettem a beszélő azonosítóját, a checkpoint-ot amelyiket szeretném, ha be lenne töltve, illetve az útvonalat ahhoz a fájlhoz, amiben a mondatok el vannak mentve, amit igény szerint át lehet írni, ha egy előre megírt szöveges fájllal szükséges tesztelni.

Annak érdekében, hogy még dinamikusabb legyen a kód, és máshol se kelljen átírni dolgokat, azt a kódrészletet, ahol a megfelelő FastPitch modellt tölti be, ahol a kimeneti fájl nevét adom meg, és ott is, ahol a hangfájlt töltöm be (amit utána meg lehet hallgatni), azt átírtam olyanra, hogy a beszélő azonosítóját és a checkpoint-ot behelyettesítse az útvonalaik nevébe. Tesztelés céljából legeneráltattam mindegyik beszélőhöz a legrosszabb és legjobb változatát.

Beszélő	Checkpoint 100	Checkpoint 1000
0 – cmu_us_awb (skót férfi erős akcentussal)	Nagyon egyenetlen a beszéd, néhány szó csak zaj	Nem sokkal lett jobb, még mindig csak közepes a minősége
1 – cmu_us_bdl (brit férfi gyenge akcentussal)	Minden mondatnál lemarad az eleje és a vége, ami hallható, az egészen érthető, bár kicsit zajos, a legrövidebb mondatból szinte semmi nem hallható	A rövidebb mondatokkal még mindig vannak kisebb problémák, nem hallható már sok zaj
2 – cmu_us_clb (brit nő gyenge akcentussal)	Sokkal több hallatszik a mondatokból, minimálisan vágja le csak az elejét és a végét, egészen érthető, de szintén zajos	A mondatok eleje most már hallatszik, szinte nem hallani zajt
3 – cmu_us_jmk (kanadai férfi gyenge akcentussal)	Az egyes beszélőhöz hasonló, sok nem hallatszik az elejéből és végéből a mondatoknak, viszont sokkal zajosabb, mint az egyes	Sokkal jobb lett a minősége, minimális zaj hallható, a rövid mondat itt is probléma
4 – cmu_us_ksp (indiai angol férfi erős akcentussal)	Egész jó minőségű, szinte az egész mondat hallható és nem annyira zajos	Érthető és nem zajos
5 – cmu_us_rms (brit férfi gyenge akcentussal)	Elég gépiesnek hangzik, de a szavak érthetőek	Nagyon jól érthető, de még mindig gépies a hangzása
6 – cmu_us_slt (brit nő gyenge akcentussal)	Itt is sok lemarad az elejéből is és a végéből is, de azon kívül nagyon jól érthető	Nem zajos és jól érthető, de a rövid mondatok itt is rosszul hallhatóak

2. táblázat A hét beszélő értékelése

Az általam végzett elemzés során a (2) – es táblázatban összefoglaltam a generált hangokkal kapcsolatos megfigyeléseimet, melyeket a kétbeszélős teszt módszeréhez hasonlóan végeztem. Az egyik legfeltűnőbb tapasztalat az volt, hogy a különböző beszélők közötti különbségek ezúttal kevésbé voltak markánsak. Ráadásul, minden egyes hang minősége lényegesen javult.

Ennek jelentős része annak köszönhető, hogy ugyanazzal a mondatkészlettel történt a tanítás miden beszélő esetében, és nem volt különbség a tanító adatok mennyiségében, ami segített elkerülni az egyensúlyhiányt a tanítás során.

Azonban megfigyelhető volt, hogy a tanítás nagyon kezdeti szakaszában jóval rosszabb volt a hangok minősége, mint az LJ Speech esetében, és hosszabb időbe telt számukra a tanulás. Emellett a tanítás végére sem érte el azt a minőséget, amit az LJ Speech nyújtani tudott. Ennek oka az, hogy az ezer mondatból álló tanító adathalmaz viszonylag kevés adatot tartalmazott. Ennek következtében a generált hangok minősége alacsonyabb szinten maradt, mint amit az eredeti LJ Speech adatbázis biztosított. Ezenkívül észrevehető, hogy a legtöbb hang nem volt annyira tiszta, sok közülük zajosnak tűnt. Miután meghallgattam néhány tanító hangot, észrevettem, hogy azok is általánosságban zajosabbak voltak, ami magyarázatot adott arra, hogy miért lett ilyen a generált hangok minősége. A (2) - es táblázatban az is megfigyelhető, hogy szinte mindegyik beszélőnél az ezredik checkpoint-tal generált hang jobb minőségű volt, mint a századikkal generált. Ez arra utal, hogy nem lépett fel a túltanulás jelensége, ami azt jelenti, hogy az adott modell egy ponton túl már nem képes tovább tanulni, és rátanul a tanító adatok esetleges hibáira. Ennek következtében a modell a tesztelés során nehezebben tud általánosítani, és csak a tanító adatokhoz nagyon hasonló adattal fog tudni jól jósolni és jó eredményeket elérni.

Hang módosítása

Miután sikerült több beszélő hangját beépítenem a modellbe, a feladatomat a beszélők hangjának módosításával folytattam. Elsőként a pace (sebesség) paraméterrel kísérleteztem. Ez a paraméter lehetővé teszi, hogy beállítsam, hogy mennyire gyors legyen a generált hang. Ehhez a már meglévő Jupyter notebook fájlban hozzá kellett adnom az argumentumokhoz a pace paramétert is. Az első teszt során lecsökkentettem a sebesség értékét a felére, majd ezután kipróbáltam, hogy milyen lesz a végeredmény, ha a kétszeresére gyorsítom. A lassított mondat nem volt folyamatos, nagy szünetek voltak a szavak között és egy-egy szó hosszabban volt kiejtve. A gyorsított mondatnál pedig nem volt semmi szünet a szavak között, és nagyon gyors volt, szinte el volt hadarva, de még mindig érthető volt. Mindezek után kipróbáltam pár szélsőségesebb értéket is. Az ötszörös sebességű mondatból nem lehetett érteni semmit, olyan gyors volt. A 0.1-szeresére lecsökkentett sebességű mondat pedig nagyon szakadozott volt, egy-egy betű kimondása több másodpercig is eltartott.

Megpróbáltam hasonló módon a hang frekvenciájának a módosítását is, de ebben az esetben nem volt változás a hangban az eredetihez képest. Mivel ugyanúgy bemeneti paraméterként lehet megadni a frekvenciára vonatkozó paramétereket, ezért először nem értettem, hogy miért nem változott semmi a hangon. Azért, hogy megtaláljam, mi okozhatja a hibát, átnéztem a kód erre vonatkozó részeit, és azt vettem észre, hogy amennyiben a modellben a Torchscript (segít Pytorch kódból készített modellek mentésére, majd azt egy Python nyelvtől független platformra betölteni) értéke igaz, abban az esetben minden hangmagasság változtatásával kapcsolatos paraméter le van tiltva és nincs hatással a hang jóslásánál. Miután a paraméter értékét hamisra állítottam, már sikeresen tudtam változtatni a hang magasságát változtató paramétereket. Ilyen paraméter a shift (eltolás), amivel el lehet tolni a hang frekvenciáját egy megadott Hertz értékkel, ami lehet pozitív és negatív is. A pozitív számok magasabbá teszik a hangot, míg a negatívak mélyítik azt. Másik ilyen paraméter az amplify (erősít), amivel pedig meg lehet szorozni egy adott számmal a frekvencia értékét, ezzel a mélyebb és magasabb frekvenciák közötti különbséget meg tudja növelni, ettől nagyobbakat fog a hangmagasság ugrálni.

Beszélők hangjának keverése

Már az eddigi változtatások is hozzájárulnak ahhoz, hogy minimálisan más hangon szólaltassuk meg a modellt, mint amivel betanítottuk. Azonban van lehetőség arra is, hogy

egyszerre több beszélő hangját is felhasználjuk, és ezeket összekeverjük. Ahhoz, hogy ezt elérjük, nem elég a szkript fájlokban átírni valamit, hanem a fastpitch mappán belül lévő „model.py” forward függvényét kellett átalakítani úgy, hogy ötven-ötven százalékban adja hozzá a generált hanghoz a két beszélő hangját. Legeneráltattam pár hangot tesztelés céljából, és jól hallható, hogy ezek a hangok egyik eddigi beszélő hangjára se hasonlítanak. Ezzel sikeresen létrehoztam új hangokat.

Modell elmentése

Végső célom volt a betanított modellt kimenteni, majd utána azt a Colab környezetbe betölteni és felhasználni az ott használt FastPitch modell helyett. Ezekből a célokból eddig a modell elmentése valósult meg. Mivel a checkpoint fájl alapból minden információt tárol a modellről, és ezek nagy része nem szükségesek a másik környezetbe való átvitelhez, ezért lekértem belőle a szükséges információkat. A feladatot TorchScript segítségével oldottam meg, ami ahogyan azt már említettem, erre van kitalálva. Ezenkívül a modellt Nemo formátumú fájlként kellett elmenteni, amiben a modell súlyainak és egy konfigurációs fájlnak kell szerepelnie. A betöltött checkpoint szótár típusú, ezért a megfelelő kulcsok megadásával (súlyok – state_dict, konfiguráció – config) megkaptam azok értékeit, amik közül a súlyokat egy .pt kiterjesztésű, a konfigurációt pedig egy .json kiterjesztésű fájlként tudtam elmenteni. Ennek a modellnek a beimportálása másik környezetbe, illetve felhasználása a jövőbeli projektjeim közé tartozik.

2.2 Összefoglalás

A félév során a beszédgenerálás témakörével foglalkoztam, amivel azért érdemes foglalkozni, mert a mai világban már a mindennapi eszközeink nagy részének alapját szolgálja a beszédgenerálás, amik sokat tudnak segíteni az embereknek különböző területeken (például hangos bemondók). Ehhez a projekthez először kipróbáltam pár példát, ami a megértésben segített, majd azt felhasználva egy másik adatbázissal dolgoztam tovább. A munkám során az okozta számomra a legnagyobb nehézséget, hogy a hang generálás egy nehéz és összetett feladat, és teljesen új technológiákat kellett benne alkalmaznom, amiket eddig nem ismertem. Az is kisebb nehézséget okozott, hogy egy olyan kóddal kellett dolgoznom, amit nem én írtam, így sokat kellett tanulmányoznom, hogy megértsem, hogy melyik függvény mit csinál. Ez a projekt egy megfelelő kihívás volt számomra, és amit mindenképp szeretnék továbbfejleszteni a későbbiekben.

A feladatomat gyakorló feladatok megoldásával kezdtem, amit a Colab és a Jupyter notebook felületén is elvégeztem. Ezekkel sikeresen megismerkedtem a téma alapjaival, illetve a két platformmal is. Az LJ Speech adatbázist felhasználva elsőként egy beszélővel tanítottam be a modellemet. Az ezzel generált hangoknak nagyon jó minősége volt, mert nagyon sok és sokfélék voltak a tanító adatok, ráadásul stúdió környezetben voltak rögzítve. Ezután egy másik hang hozzáadásával két beszélőssé tettem a modellt. Itt megfigyelhető volt, hogy egymáshoz képest nagyon nagy különbség volt a két beszélő hangjának minőségében. Ebben szerepet játszott a tanító adatok minőségének és mennyiségének különbsége. A következő lépésben a LJ Speech adatbázist teljesen lecseréltem hét új hangra. Ezek mind ugyanazokkal a mondatokkal voltak betanítva és a mondatok mennyiségében sem volt különbség. Ennek eredményeként egymáshoz képest nem voltak nagy minőségbeli különbségek, egyedül az tett különbséget, hogy a rögzítés során mennyire lettek zajosak a felvételek. Mivel csak ezer mondattal voltak betanítva, így viszont érezhetően alacsonyabb minőséget értek el az LJ Speech-nél. Ezt tovább lehetne javítani több tanító adattal, a hiperparaméterek optimalizálásával, illetve meg lehetne próbálni, hogy képes-e tovább tanulni a modell, ha több iterációig hagyjuk futni. Ezután ezzel a betanított modellel dolgoztam tovább, és megpróbáltam különböző módosításokat végezni a

hangokon. Elsőként a hang sebességét, majd a frekvenciáját módosítottam, ezzel ugyanazzal a beszélővel különböző hangokat el tudtam érni. Legnagyobb eredményem volt, hogy két beszélő hangját össze tudtam keverni, aminek segítségével egy harmadik teljesen új hang jött létre. Ezt el lehetne végezni több beszélő hangjával is, sőt eltérő mértékben is lehetne őket keverni. Végül a betanított modellt Nemo formátumban elmentettem, amelyet későbbi feladataimhoz fel tudok majd használni.

Ezt a projektet szeretném a jövőben továbbfejleszteni és kibővíteni. A hangokat lehetne még érthetőbbé és kevésbé zajossá tenni. Lehetne érzelmek szerint különböző hangsúllyal mondani a mondatokat, illetve többnyelvűvé is át lehetne alakítani a modellt.

3. Irodalom, és csatlakozó dokumentumok jegyzéke

A tanulmányozott irodalom jegyzéke:

[1] *Arpabet to IPA,* https://docs.soapboxlabs.com/resources/linguistics/arpabet-to-ipa/, szerk: SoapBox Labs Resources, Utolsó letöltés: 2024. május 10.

[2] *Spectrograms and Oscillograms*, https://www.researchgate.net/figure/Spectrograms-and-Oscillograms-This-is-an-oscillogram-and-spectrogram-of-the-boatwhistle_fig2_267827408, szerk: Phillip S. Lobel, 2014. Utolsó letöltés: 2024. április 30.

[3] *Hifi-GAN,* https://paperswithcode.com/method/waveglow, szerk: Ryan Prenger, Rafael Valle, Bryan Catanzaro, 2020. Utolsó letöltés: 2024. május 13.

[4] *Waveglow,* https://paperswithcode.com/method/hifi-gan, szerk: Jungil Kong, Jaehyeon Kim, Jaekyoung Bae, 2018. október 30. Utolsó letöltés: 2024. május 13.

[5] *Text-to-Speech 101: The Ultimate Guide,* https://medium.com/neuralspace/text-to-speech-101-the-ultimate-guide-9a4b10e20fef, szerk: Felix Laumann, 2023. november 30., Utolsó letöltés: 2024. május 8.

[6] *NVIDIA, NeMo TTS Primer,* https://colab.research.google.com/github/NVIDIA/NeMo/blob/stable/tutorials/tts/NeMo_TTS_Primer.ipynb#scrollTo=Z5_6Esy-Cz52*,* szerk: NVIDIA, 2024. Utolsó letöltés: 2024. március 10.

[7] *DeepLearningExamples/PyTorch/SpeechSynthesis GitHub,* https://github.com/NVIDIA/DeepLearningExamples/tree/master/PyTorch/SpeechSynthesis*,* szerk: Adrian Lancucki, Krzysztof Kudrynski, 2023. Utolsó letöltés: 2024. március 18.

[8] *CMU ARCTIC Databases, Festival Speech Synthesis Systems* http://www.festvox.org/cmu_arctic/*.,* szerk: Alan W Black, 2017. Utolsó letöltés: 2024. március 25.