Témalabor beszámoló

Távközlési és Médiainformatikai Tanszék

Készítette:	**Kiss Tekla Krisztina**
Neptun-kód:	**ZMTZZH**
	kteklus@gmail.com
Ágazat:	**Villamosmérnök**
Vezető:	**Dr. Zainkó Csaba**
	zainko@tmit.bme.hu
Konzulens:	**Nagy Péter**
	nagyp@tmit.bme.hu

Téma címe:

Zaj csökkentés Deep Learning segítségével

Feladat

A hallgató feladata a témalabor feladat során egy olyan neurális hálózat kialakítása és betanítása, ami alkalmas annak bemenetén szolgáltatott hangjelen lényegkiemelés/háttérzaj szűrés megvalósítására.

2019/2020. 1. félév

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

1.1 Bevezető

Izgalmas projektnek képzeltem el azt, hogy hogyan lehet a háttérzajokat csökkenteni egy-egy beszélgetés során. Különböző tudományos cikkeket olvasva rájöttem, hogy ebben a témában még van hova fejlődnie a világnak. Bár van egy pár jó megoldás, nem mindegyik tudja jelen időben ezt a helyzetet helyesen lekezelni. Kíváncsi voltam, hogy egy Deep Learning-es projektként vajon hogyan lehet ezt megvalósítani.

Mivel a Deep Learning hatékony megoldást kínál a képi, hangi, nyelvi környezetben ezért ez a projekt is megvalósíthatónak tűnt.

1.2 Elméleti összefoglaló

Mindennapi életben egyre nagyobb mennyiségű és változatos formátumú adat keletkezik, például ipari, egészségügyi, vagy pénzügyi adatok. Iyen mennyiségű adatot emberi ésszel felfogni sem vagyunk képesek, így született meg a technika, amely a segítségünkre válhat. A gépek emberi segítséggel vagy anélkül képesek példa adatok alapján azokban mintákat és szabályszerűségket keresni, majd ezeket ismeretlen adatokra általánosítani és következtetéseket levonni belőle. Ezen törekvésünk, hogy ezek a gépi algoritmusok, minél tökéletesebbek legyenek az emberi szakértelem helyettesítésére, kiegészítésre irányulnak. Ezért szükségünk van a Deep Learning megismerésésre.

A Deep Learning a Machine Learning rendszer egy szűkebb része, amelyek neurális hálók összerakásával működnek. A mesterséges neurális hálókat elsősorban a biológiai agy inspirálta. Úgy is tekinthetünk ezekre a hálókra, mint matematikai modellekre, amelyeket az információ feldolgozására használunk. Olyan gépi tanulás, ahol a belső rétegek megfelelő összeállításából egy magasabb szintű eredményt hoz létre a nyers adatokból.

A tanulás elemi számítási egysége a mesterséges neuronok, ezeket rétegekbe rendezzük, a számításokat pedig mátrix műveletekkel szemléltethetjük. Az elnevezés a biológiából származik, de nem egyértelmű a megfeleltetés, csupán hasonlóságot takar az ingertovábbítás terén.

$$\varphi\,(\sum(w_i x_i + b))$$

x_i: egy bemeneti réteg

w_i: ehhez tartozó súly

b: az eltolás(bias)

φ: egy adott aktivációs függvény

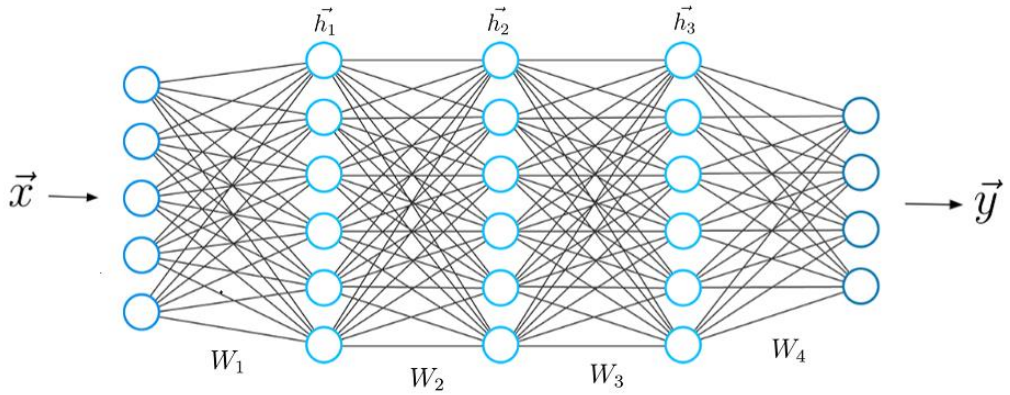
A mesteséges neuronokban a bemeneti értékek súlyozott összegét átvezetik egy aktivációs függvényen, majd a kapott eredménytől függően továbbítják azt.

A rétegek között megkülönböztetünk bemeneti réteget, rejtett rétegeket és kimeneti réteget.

Bemeneti rétegként azt az információt adjuk a hálózatnak, ami a rendelkezésünkre áll és amelynek a tulajdonságait, viselkedését szeretnénk megfigyelni, alakítani. Itt igazából annyi történik, hogy az adatokat megfelelő formátumra hozzuk, attól függően, hogy milyen típusú adatokat gyűjtöttünk, találtunk.

A rejtett rétegek feladata az információ feldolgozása, itt változtathatjuk ezek számát, típusát, megadhatunk aktivációs függvényeket. Azt is befolyásolhatjuk, hogy ezeknek egymáshoz milyen kapcsolatuk van. Ahogy fejlődik, ez a tanulási forma egyre változatosabb megoldások keletkeznek. Itt arra gondolok, hogy meghatározható az is, hogy egyes rétegek több korábbi réteg kimenetét kapják meg bemenetként vagy csupán az őket megelőzőét.

A kimeneti rétegből kaphatjuk meg az eredményünket. Természetesen itt is meghatározó, hogy mit adtunk a bemenetre és milyen feladatot állítottunk a hálózat elé. Azaz meghatározhatjuk, hogy hány kimeneti neuronunk legyen.



Ábra [1] Neurális hálók szerkezete: bemeneti neutronok, rejtett rétegek, és a kimeneti értékek.[6]

A feladatok megvalósításánál két fázist különböztetünk meg. A tanítási fázisban az ismert bemeneteket (x) és az egyes rétegek súlyait változtatjuk, úgy, hogy a kiválasztott hibafüggvénnyel minimalizáljuk a hibát. Ezt úgy kell érteni, hogy a rétegeken végig vezetjük a bemeneti értékeket, beszorozzuk a súlyokkal, hozzáadunk egy konstans értéket, ezzel kapunk egy becsült értéket, ezt pedig össze tudjuk hasonlítani az elvárt kimeneti értékkel (y). Célunk az, hogy az így keletkező x-y párokkal megadott függvényt jól megközelítsük úgy, hogy a modell lehető legjobb általánosító képességgel rendelkezzen. Minél több adatunk van, annál tovább taníthatjuk a hálónkat, így a predikciós fázisban az előtanított háló helyesebb működést eredményez. Ilyenkor olyan adatokat jutattunk a háló bemenetére, amik számára ismeretlenek. Az eredményből pedig következtetthetünk a hálózat fejlettségi színtjére. Ha nem vagyunk elégedettek, akkor visszatérünk a tanítási fázisba és korrigálhatjuk a rétegek számát, a súlyok értékét, vagy, hogy éppen milyen optimalizációs függvényt használjunk.

Szükséges, hogy a mérjük a modell általánosítóképességét, ezért meghatározunk egy olyan függvényt, ami számszerüen megadja, hogy a modell becsült kimenete mennyire különbözik az előre megadott mintától. Azaz mekkora a távolság közöttük, ezt nevezzük veszteségfüggvénynek.

Gradiens módszert használunk a hálózat tanítására. A hálózat rétegeit alkotó függvények differenciálhatók, így a hibafüggvény is differenciálható. Ebből már megállapíthatjuk egy adott függvény súlyai szerinti gradiensét, hogy merre kell elindulni a hálózat paraméterei mentén, ahhoz, hogy a hiba csökkenjen. A lépéseket iterálva megkaphatjuk a függvény egy lokális minimumát.

Optimalizációnak az olyan technikákat nevezzük, amelyeknél nem a súlyokból vonjuk ki a gradienst, hogy így csökkentsük a veszteségfüggvényt, hiszen ez sok esetben lassan képes biztosítani a konvergenciát, hanem gradiensereszkedés algoritmust különböző kiterjesztésekkel látjuk el. Ezzel végződik egy tanítási iteráció, és kezdődik a következő részminta képzés, hibaszámítás, visszaterjesztés, és gradienseresztés. Tanulókorszaknak nevezzük az adatok egy teljes végig járását. A veszteségfüggvény kimenetét nyilvántartjuk, és addig folytatjuk a tanítást, amig el nem érünk a minimumba.

1.3 A munka állapota, készültségi foka a félév elején

A félév elején még a tanszék különböző laborjait ismertük meg, majd személyes konzultációk következtek a kiválasztott laborban. Én a SmartLabot választottam, azon belül is a Deep Learning témakört. Rögtön az elején mindenki kiválasztotta a neki tetsző témát és szakmai irodalmak olvasásába, valamint a fejlesztő környezet felállításába kezdett. Szerencsére nagyon sok Open Source kódot és megoldásokat lehet találni az interneten és a könyvekben. Viszont az is belátható, hogy még rengeteg tanulás vár ránk, hogy a Deep Learning-ben szélesebb ismeretekre tegyünk szert. Motiváló közeget kaptunk az információk gyűjtésére, ugyanakkor ez csak a jéghegy csúcsa.

Abból indultam ki, mint a telefonok, hogy egyszerre több mikrofon van elhelyezve bennük, hogy több irányból szűrjék a hangokat, így képezve egy komplementer kimenetet. Tehát, míg van egy mikrofon a telefon felsőrészén és egy alul, ahol beszélünk bele, a felső kevesebb beszédhangot kap, tehát amit ott érzékel, a telefon azt nyugodtan kivonhatja a beszélőnél érzékelt hangból, így tisztítva meg a zajoktól.

2. Az elvégzett munka és eredmények ismertetése

1.1 Saját feladat megvalósítása

A félév elején szükséges volt a python nyelv elsajátítása, megismerkedés olyan könyvtárakkal amelyek lehetővé teszik a kódolás explicit leírását, mert a mátrix műveletek egy-egy függvény meghívásával elérhetőek számunkra. A TensorFlow, Keras, és PyTorch mind ilyen open source könyvtárak amelyeket Pythonon keresztül érhetünk el. Mindhárom a bevezetőben leírt tanítást segítik.

Az adatokat a The University of Edinburgh weboldaláról szereztem be, mert olyan hanganyaggal szerettem volna dolgozni, amin emberek beszélnek és van némi háttérzaj, amit kiszűrhetek. Ezek pár másodperces (3-4 s) hosszú wav fileokba lettek feltöltve. A gyenge értelmű stacionalítás elvethető, mivel a szórás változik. A wav file-okban inteket tárolunk, ezek 16 bites mono csatornájának 48 kHz a keret frekvenciája, ami azt jelenti, hogy 48000 mintánk keletkezik másodpercenként.

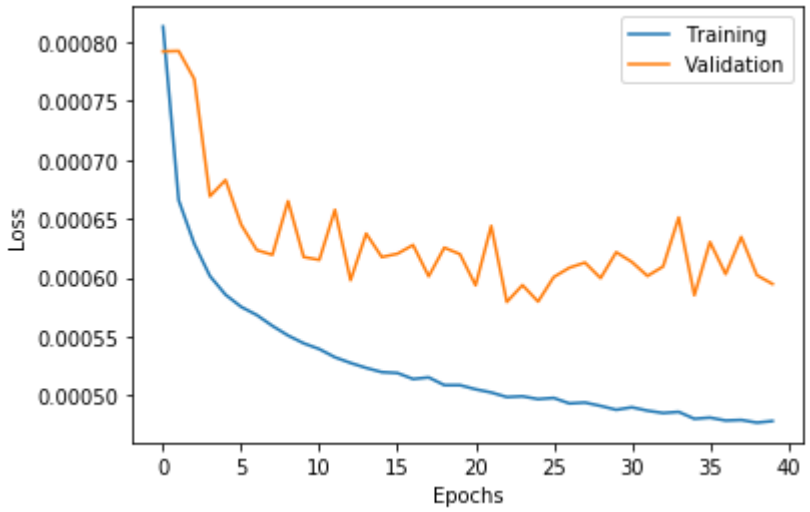
Az zajosadatok beolvasásánál egy ablakozást végeztem, ahol 100 mintát veszek és 50-esével csúsztatom az ablakokat a következő mintákra, ezeket a mintákat egybefűztem. A tiszta adatokat is hasonlóan ablakoztam, csak itt 25 minta után kezdtem, 50 mintát olvastam egyszerre és 50-sével futottam végig a mintákon. Ezt elvégeztem a teszt adatokon is, így keletkezett 111188 ablaknyi tanító és 29850 db teszt adat. Az adatok -16000 és 16000-es értékek között mozognak, ahhoz, hogy ezeket egy háló bemenetére tudjam kötni egy elég nagy konstans (32767) segítségével leosztottam egyesével, hogy -1 és 1 közötti értékeket vegyenek fel. Szükséges volt továbbá, hogy az adatokat int-ről floatra alakítani.

A mesterséges neurális hálóm deklarálásához a Keras könyvtárat hívtam segítségül. Itt ugyanis meg vannak írva a számomra fontos modellek. 5 rétegű modellt hoztam létre, ebből egy 200 neuront tartalmazó bemenetű réteg, 3 rejtett réteg mind 200 neuront tartalmaz és egy kimeneti réteg, ami 50 neuront tartalmaz. A rejtett rétegek aktiválásához 'relu'-t használtam, míg a kimenetnél 'linear'-t, ami azért fontos, hogy megmaradjanak a negatív és pozitív értékek az eredményben. A 'relu' bemenetére adott negatív értékeket levágja, míg a pozitívokat változatlanul hagyja. Ez egy teljesen kapcsolt réteg struktúra, mert előállítja a bemenetek és a súlyok lineáris kombinációját. A neurális hálóban az i-edik réteg teljes, azaz

fully-connected, ha úgy épül fel, hogy az i-1-edik réteg neuronjainak mindegyike össze van kötve az i-edik réteg minden neuronjával.

A háló struktúra után megadtam neki, hogy milyen optimalizátorral és milyen hiba függvénnyel javítson az értékeken. 'Adam' optimalizátort választottam, ez egy olyan lendület módszer, amely nyilvántartja a gradiensek mozgó átlagát és négyzetes mozgó átlagát ezek bizonyos kombinációjával csökkenti a súlyok értékét addig, míg a derivált érték eljut a minimumához. Minden egyes neuron kiszámításánál a hibaszámítást veszi alapul, ami kiszámolja visszafelé az adott réteg hibáját.

Miután az előző részben megadtuk milyen optimalizációt és hibaszámítást szeretnék, a modellünk segítségével betaníthatjuk az előkészített adatainkkal. Itt a model.fit() függvényt használtam, megadtam neki a training és testing mintákat és azt is, hogy hányszor szeretném, ha ezeken az adatokon tesztelne és milyen mintavételi aránnyal. A hányszornál az epoch-okat állítottam 40-re, arra, hogy hány mintát nézzen egyszerre a batch_size-t 32-re vettem.



Ábra[2] a tanulási folyamat: hibacsökkenése a training és a test adatokon

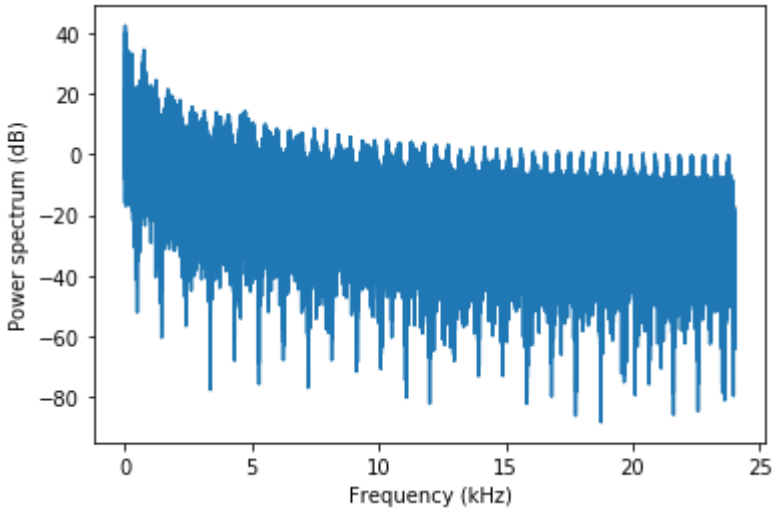
Bár a háló nem tanul tökéletesen, de a hiba mind a training, mind a test adatokon csökken.

A kimenetet a predict függvénnyel alkottam meg. Ezt kimentettem egy újabb wav file-ba, úgy, hogy a mintákat visszaszoroztam azzal a konstanssal, amit a legelején inicializáltam és osztottam el vele a bemenő adatokat, majd intre kerekítettem ezeket a számokat.

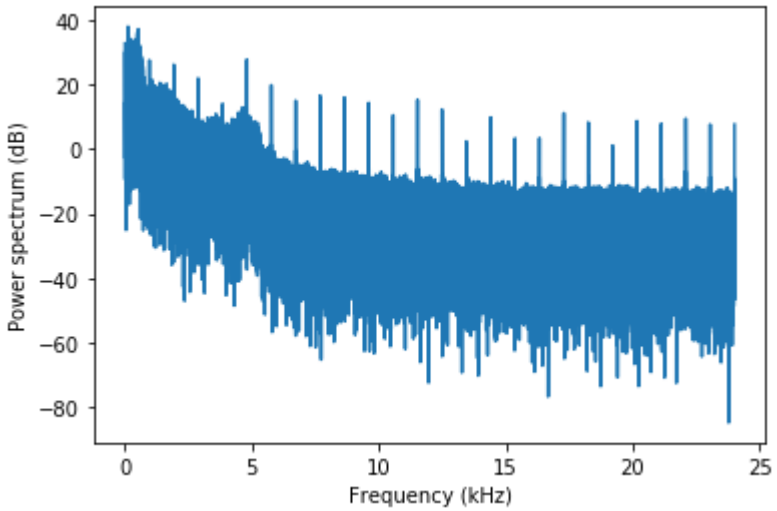
Meghallgatás után azt tapasztaltam, hogy bár valamelyest javult a minta, nem tudta az

összes zajt megszűrni, továbbá torzítás figyelhető meg a beszélő hangjában is.

Szeretném még ezzel a két ábrával szemléltetni, hogy mi is történt közben a mintákkal. Ezek a képek a minták spektrumait ábrázolják és láthatjuk, hogy a kimeneti hang spektrum képén már kevesebb zaj ül, azaz sikerült csökkentenem a zaj hatását.



Ábra[3] A bemeneti zajos adatok spektrum képe



Ábra[4] Az javított zajos minták spektrum képe

2.2 Összefoglalás

A félév során egészen tág képet kaphattam arról mi mindenre lehet felhasználni ezt a tudomány ágat. Megismerkedtem a Python struktúrájával, és a Deep Learning hálók megalkotásával. Ezek összehangolt használata olyan jövőbeli eredményeket hozhat a világ számára amit ma még el sem tudunk képzelni.

Elégedett vagyok az eredményemmel, mert ez az első saját neurális hálóm, élveztem a Keras layerek működésének megértését és a háló betanítását. Habár azt is tudom, hogy a jövőben ki lehetne probálni további struktúrákat, ilyen lehetne a GAN-ok világa, esetleg Wavenet, CNN vagy LSTM hálók összerakásával.

A GAN-oknál van egy úgynevezett generátorháló és egy diszkriminátor háló ezek egymás mellett párhuzamosan működnek és addig tanul a generátorháló, amig a diszkriminátor visszadobja az adatokat, azok hibája miatt. A diszkriminátor akkor fogadja el a mintákat helyesnek, amikor már nem tudja megkülönböztetni azoktól amit várunk tőlük. A saját példámban ezt úgy lehetne megvalósítani, hogy a generátorháló folyamatosan javítja a bemenő mintát azáltal, hogy csökkenti rajta a zajt, majd új mintákat állít elő és a diszkriminátor ellenőrzi ezeket, amig nem olyan tiszta, mint a zaj nélküli minták, addig nem fogadhatja el így a generátornak tovább kell próbálkoznia.

A Wavenet olyan metodus, amely elég összetett és sok időt vesz igénybe megérteni, de kifejezetten jól tud zajtcsökkenteni felvételeken.

Remélem ez kipróbálására is lesz még lehetőségem a következő félévekben.

3. Irodalom, és csatlakozó dokumentumok jegyzéke

A tanulmányozott irodalom jegyzéke:

1) Ivan Vasilev, Daniel Slater, Gianmario Spacagna, Peter Roelants, Valentino Zocca, *Python Deep Learning Second Edition*, Packt Publishing, 2019.
2) Michael Michelashvili, Lior Wolf, *Audio Denoising with Deep Network Prios,* Tel Aviv University, 2019. https://arxiv.org/pdf/1904.07612.pdf
3) https://devblogs.nvidia.com/nvidia-real-time-noise-suppression-deep-learning/
4) https://datashare.is.ed.ac.uk/handle/10283/2791
5) https://github.com/drethage/speech-denoising-wavenet
6) https://www.deeplearning-academy.com/p/ai-wiki-machine-learning-vs-deep-learning
7) https://hu.wikipedia.org/wiki/Mesters%C3%A9ges_neur%C3%A1lis_h%C3%A1l%C3%B3zat
8) https://keras.io/models/sequential/