Eötvös Loránd Tudományegyetem

Informatikai Kar

Numerikus Analízis

Tanszék

Zenei műfaj meghatározása neurális hálóval

Témavezető:
Dr. Szekeres Béla János
Adjunktus - PhD

Szerző:
Müller Alex Tibor
programtervező informatikus BSc

Szombathely, 2023

Tartalomjegyzék

1. fejezet

Bevezetés

A neurális hálózatok az utóbbi években a mesterséges intelligencia területén nagy előrelépést értek el. Az élet számos területén alkalmazható a gépi tanulás által nyújtott tudás. Az elmúlt egy évben több ezen a tudáson alapuló program és szolgáltatás jelent meg, melyek nagyban megváltoztathatják az emberek információ szerzési és számítógépezési szokásait. Az egyik ilyen terület, ahol neurális hálózat alkalmazható az a zeneipar.

1.1. Téma

Szakdolgozatom egy általunk választott zene műfaját meghatározó programmal foglalkozik. Ahhoz, hogy egy zene műfaját meghatározzuk, két előzetes lépés szükséges, ezért szóba kerül az audió fájlok feldolgozása és a neurális hálózat betanítása. Továbbá szerepelnek ezen folyamatok megértéséhez szükséges zenéhez, gépi tanuláshoz és Fourier-analízishoz kapcsolódó tudás. Az adatok feldolgozásánál szóba kerül a hang fájl felépítése, főbb típusai, Fourier-transzformációk, spectogramok és mel-frekvencia cepstrum együtthatók (mfcc) készítése, ezekből kinyerhető információk, valamint mi a szerepük a műfaj meghatározás során. A betanításnál ismertetem a neurális hálózat felépítését, feladatát és felhasználását a műfaj meghatározó programban.

2. fejezet

Felhasználói dokumentáció

A műfajt meghatározó program és a működéséhez szükséges audió fájl előfeldolgozó és a neurális háló tanítását végző programok során az alábbi Python 3 álltal biztosított könyvtárakat használtam: numpy, librosa, mutagen, matplotlib, tinytag, json, os, math, time, tensorflow, kaggle.api, zipfile, sklearn, pydub, pygame és pyqt5. Numpy különböző numerikus számításokhoz, míg a librosa az audió fájlok feldolgozásához nyújtott hatékony eszközöket. A programban használt konvolúciós neurális hálózat kialakításához Tensorflowt használtam. A főprogram felhasználói felületének elkészítéséhez a Pyqt5 biztosított széles eszköztárat, valamint a kiválasztott zene metaadatainak kinyerésében a mutagen és a tinytag segített.
Az audió fájlok előfeldolgozásánál a program a GTZAN Dataset-et [1] használja, amit mel-frekvencia cepstrum együtthatókká alakítva egy JSON fájlban tárol el, ami később a neurális hálózat bemenetét biztosítja.
A főprogramot, azaz a műfajmeghatározásért felelő egységet, zeneiparban dolgozó vagy zene iránt érdeklődő felhasználónak ajánlom. A főprogram működését befolyásoló részegységek (audió fájl feldolgozó program, neurális hálózat szerkezetéért és betanításáért felelős program) használatát csak azoknak a felhasználóknak javaslom, akik rendelkeznek python, gépi tanulás és zenei háttértudással.

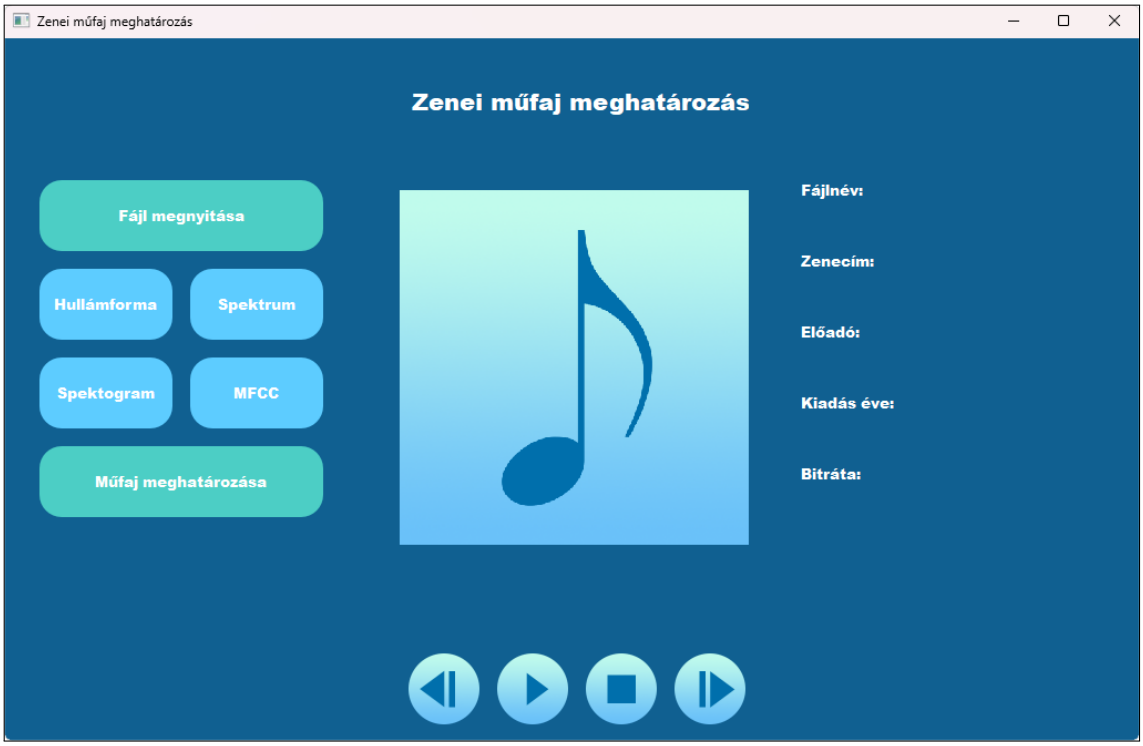
2.1. A műfaj meghatározó program

A műfaj meghatározó program négy fő funkcióval rendelkezik. Az első, hogy képes a felhasználó által kiválasztott hangfájlt elemezni, különböző típusú plotokon megjeleníteni. A második funkció egy beépített zenelejátszó, amivel meghallgatható

a betöltött audió fájl. A harmadik a kiválasztott hangfájlba kódolt metaadatokat jeleníti meg, ha rendelkezik ilyennel. A legfontosabb funkció pedig a negyedik, ami az adott zene műfaját fogja meghatározni.

2.1.1. Felhasználói felület

A felhasználói felület PyQt5 Python könyvtár használatával készült el, ami széles eszköztárat nyújtott egy modern felhasználóbarát felület kialakításához. A program elindításakor az alábbi felület fogadja a felhasználót:



2.1. ábra. A program elindításkor

Az ablakon belül vízszintes elrendezéssel jelenik meg a 4 különböző vezérlőegység (widget), ezáltal a program ablakának nagyítása után is mindegyik részegység szépen elkülönül. A felhasználói felület és a főprogram elkészítése során, igyekeztem úgy felépíteni a felületet, hogy egy átlagfelhasználó, egy zene vagy gépi tanulás iránt érdeklődő felhasználó is könnyedén használni tudja és megértse a műfajmeghatározás mögött lévő logikát. Bal oldalt találhatóak a fájl megnyitásáért, négy különböző plot megjelenítéséért és a kiválasztott zene műfajának meghatározásáért felelős gombok. Középen az adott audió fájl metadatájában tárolt borítókép jelenik meg, ha rendelkezik ilyennel. Jobb oldalt további metadatában található adatok jelennek meg, mint a zene címe, előadója, kiadási dátuma, bitrátája és a fájlnév

fájlkiterjesztéssel együtt. A borítókép alatt egy zenelejátszó található, amivel képes a felhasználó elindítani, szüneteltetni, előre tekerni, vissza tekerni valamint leállítani az adott hangfájlt. A felhasználói felületen szereplő gombok színe elkülönül a háttértől, valamint a kurzorral való rámutatás során elszíneződik, hogy ezzel megkönnyítse a felhasználót a döntésben, hogy melyik funkcióra kattintson rá.

2.1.2. Hangfájl betöltése és elfogadott fájlformátumok

A fájl megnyitása gombra kattintva, kiválasztható egy tetszőleges hangfájl a számítógépről. Megnyitás során .wav, .flac és .mp3 kiterjesztéssel rendelkező fájlokat lehet megnyitni. A három formátum között minőségbeli és tömörítésbeli különbségek vannak.
Az alábbi táblázat az egyik saját zenémen (MVLLER - Freedom) mutatja be a minőség és fájlméret közötti kapcsolatot. A zeneszám 44100 Hz-es mintavételezési frekvencián lett felvéve és 3 perc 40 másodperc hosszú.

Fájl formátum	**Bitráta**	**Fájlméret**
Wav 32 Bit float	2822 kb/s	74.0 MB
Wav 24 Bit int	2116 kb/s	55.5 MB
Wav 16 Bit int	1411 kb/s	37.0 MB
Flac 24 Bit int	1508 kb/s	39.5 MB
Flac 16 Bit int	825 kb/s	21.6 MB
Mp3	320 kb/s	8.39 MB
	256 kb/s	6.71 MB
	128 kb/s	3.35 MB
	96 kb/s	2.51 MB
	32 kb/s	859 KB

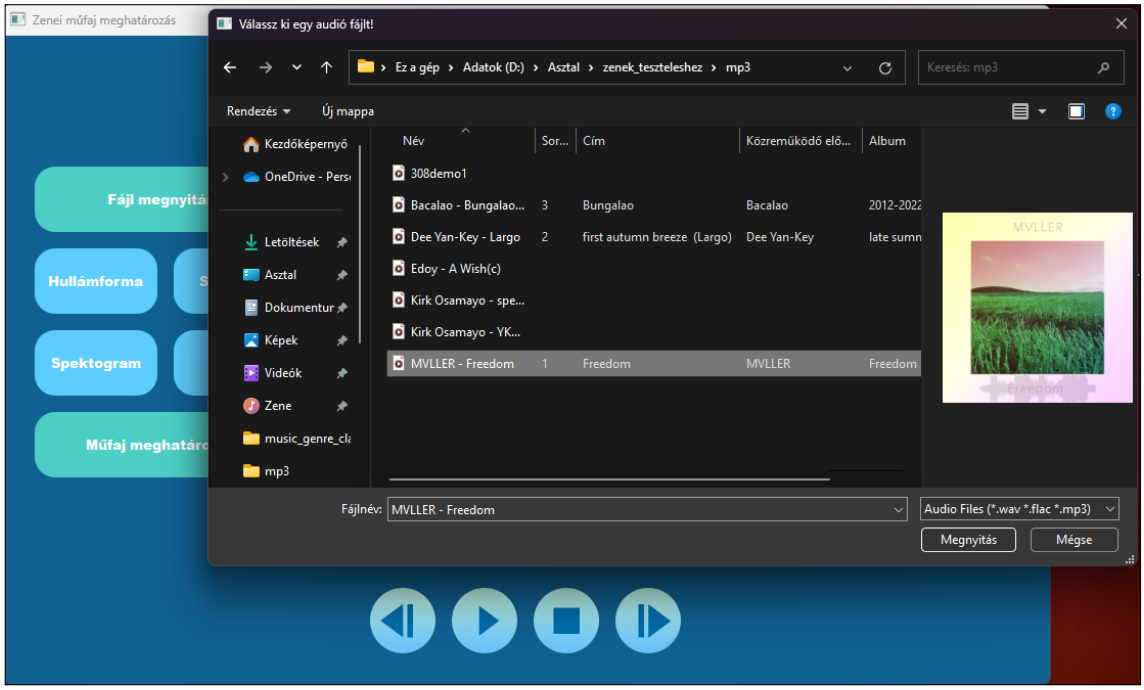
2.1. táblázat. Minőség és fájlméret közötti kapcsolat

A WAV vagy WAVE-ként emlegetett formátumot eredetileg a Microsoft és az IBM fejlesztette ki 1991-ben, és kompatibilis a legtöbb elterjedt operációs rendszerrel. Legtöbbször tömörítetlen, de képes tömörített hangot is tárolni. Fájl mérete nagy, viszont egy sokkal részletesebb hangfájlt hallgathatunk az .mp3 és a .flac-hez képest. A RIFF (Resource Interchange File Format) formátumra épül és az adatokat darabokban (chunkokban) tárolja, ami egy azonosítóból, adatból és a hosszából áll [2].
A FLAC (Free Lossless Audio Compression) célja a veszteségmentes hangtömörítés.

Kihasználja a hangminták közötti magas korrelációt és lineáris predikciót használ a minták számsorozattá alakításához. A FLAC tömörítési aránya 50-60% . Mivel tömörített formátum, ezért mérete kisebb, mint a .wav formátumé, és a hangfájlunk tömörítés során veszteségmentes marad [3].

Az mp3 az egyik leggyakrabban használt fájlformátum, mivel tömörítési aránya személyre szabható így lehetővé teszi a felhasználók számára, hogy döntsenek a fájlméret és a minőség között. Az mp3 már egy tömörített és veszteséges fájlformátum, viszont egy átlagfelhasználó zene hallgatási igényeit kielégíti. Tömörítési algoritmusát úgy tervezték, hogy csökkentse a hanganyag megjelenítéséhez szükséges adatmennyiséget úgy, hogy hű maradjon az eredeti veszteségmentes hangfelvételhez [4].

A fájl megnyitáshoz a program a PyQt5 QFileDialog könyvtárát használja. A gombra kattintás után a fájlkezelő megnyílik és a felhasználó kiválaszthatja a számára szimpatikus audió fájlt ami az előbb ismertetett három fájlformátum egyike. Megnyitás után betöltődik a hangfájl és a program többi funkciója ezzel a fájllal foglalkozik tovább.



2.2. ábra. Fájl megnyitása

2.1.3. Hangfájl megjelenítése különböző plotokon

A fájl megnyitást követően a felhasználó számára új funkciók érhetőek el. A kiválasztott hangfájlt négy különböző típusú ploton rajzolhatjuk ki: Hullámforma,

Spektrum, Spektogram és MFCC. Plotok megjelenítésével könnyebben megvizsgálható egyes zenék és zenei műfajok közötti különbségek valamint szemléltethető a Fourier transzformáció alkalmazása után elért változás. A hangfájlok különböző típusú kirajzolásához Librosa, Matplotlib valamint Pydub könyvtárakat használja a program.

Hullámforma kirajzolásához szükséges a megnyitott hangfájlunk elérési útja, mintavételezési frekvenciája (sample rate) és a hossza. Ezekre a paraméterekre a többi plot kirajzolásánál is szükség lesz. Ezen a ploton a hanghullámot egy idő, amplitúdó reprezentációban jeleníthetjük meg.

A zene frekvencia sprektrumának megjelenítését a Spektrum gombra kattintva érheti el a felhasználó. Ebben az esetben már egy Gyors Fourier transzformációt (FFT) alkalmaz a program librosa segítségével a betöltött hangfájlon. Ezen a ploton az audió fájlt hang intenzitás - frekvencia ploton rajzolódik ki. Az idővel kapcsolatos információkat elveszítjük a FFT során.

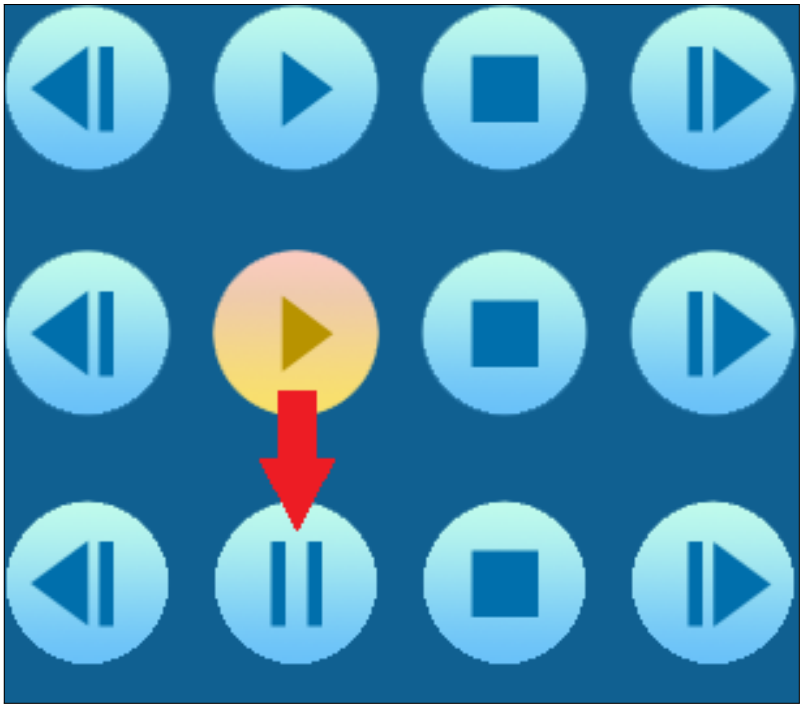
A Spektogram gombbal, a hangfájlt úgy jeleníthetjük meg, hogy mind az idő, frekvencia és hang erősséggel kapcsolatos információk megmaradnak. Ebben az esetben több FFT-t hajtunk végre, amit más néven Short-Time Fourier transzformációnak (STFT) neveznek.

Az MFCC gombbal az adott zene egy Mel frekvencia spektogrammon jeleníthető meg. Az emberi hallási folyamatot utánozza ezáltal gyakran használják hang felismeréshez és műfaj meghatározáshoz. A programban használt neurális hálózat is ebben a formában kapja meg az adathalmazunkban szereplő hangfájlokat, amiket a tanulási folyamat során fog használni.

2.1.4. Zenelejátszó

Program tervezése során egy olyan programot akartam létrehozni, amiben nem csak szemmel látható információkat tekinthetünk meg a kiválasztott hangról, hanem ezek meghallgatására is lehetőség nyíljon a felhasználó előtt. A lejátszóban négy egyedi készítésű ikonnal rendelkező gomb jelenik. A legfontosabb gombok a lejátszás/szüneteltetés és a leállítás gomb. Elindítás ikon segítségével meghallgatható a fájl kiválasztás során kiválasztott hangfájl, amit mégegyszer megnyomva szüneteltethetünk, valamint a leállítással ez leáll és a legközelebbi indításkor a lejátszás elölről indul. A lejátszóhoz a Pygame és Time könyvtárakat használtam, hogy a

hangfájlban történő előre és hátra pörgetés megfelelően működjön.



2.3. ábra. A zenelejátszó funkciógombjai

2.1.5. Hangfájlból kinyerhető metaadatok

Metaadatot, azaz az adatról szóló adatot, a különböző formátumú hangfájlok is tartalmazhatnak. Segítségével olyan információkat tudhatunk meg egy zenéről, hogy ki az előadója, mi a címe, kiadási éve, bitrátája és még sok más tulajdonságot. Ha a programban megnyitott fájl rendelkezik metaadat címkékkel, akkor ezek jobb oldalt, jelennek meg. Ha az adott hangfájl metaadatában lemezborító is található, akkor az középen a lejátszó felett jelenik meg. Mivel gyakran találkozunk olyan zenékkel, ahol hiányosan vannak a metaadat információk kitöltve, így a műfajra vonatkozó címke meghatározásában segítséget nyújthat ez a program. A metaadatok kinyeréséhez a program a Tinytag valamint a lemezborító megjeleítéshez a mutagen könyvtárakat használja.

2.4. ábra. Kiválasztott zene betöltés után, és metaadatainak megjelenítése

2.1.6. Műfaj meghatározása

Műfaj meghatározása gombra kattintva, a programunk meghatározza a kiválasztott hangfájl műfaját. Ilyenkor egy program egy konvolúciós neurális hálózat modellt tölt be, aminek elkészítésében fontos szerepe lesz az előfeldolgozásért valamint a modell tanításáért felelős programnak. A modell betöltése után a kiválasztott hangfájlunkat STFT segítségével MFCC-vé alakítja át, amit a betöltött modell feldolgoz és egy predikció során meghatározza a műfaját. Mivel ez egy olyan folyamat, ahol nagy számítások mennek végbe, így a programnak is kell egy kis töltési idő, amíg a műfajmeghatározás végbe megy. Az eredmény a műfaj meghatározás gomb alatt jelenik meg a predikció pontosságával együtt. Mivel a neurális hálózat egy olyan adathalmazon tanult, amiben 10 műfaj szerepel, mindegyikre 100 db hangminta, ezért a program az alábbi 10 műfajba tudja csoportosítani a hangfájlunkat: blues, klasszikus, country, disco, hiphop, jazz, metál, pop, reggae és rock.

2.2. Előfeldolgozó program

A főprogram valamint a konvolúciós neurális hálózat tanításához tartozó program működéséhez szükésges volt egy program, ami az adathalmazban található fájlokat

feldolgozza, átalakítja, majd lementi, hogy később a hálózat betanításánál bemenetként tudja őket használni. A program az os, math, json és librosa könyvtárakat használja. Ennek a programnak a használatát csak azoknak a felhasználóknak javaslom, akik megfelelő háttértudással rendelkeznek gépi tanulás, adatfeldolgozás és zene területén.

2.2.1. Dataset megismerése

Először meg kell ismernünk az adathalmazunk szerkezetét, felépítését, hogy tudjuk milyen fájlokkal dolgozunk. GTZAN Dataset [1] esetében 10 műfajra osztott mappát kapunk. Minden egyes műfajhoz tartozó mappa 100 db 30 másodperc hosszúságú hangmintát tartalmaz .wav formátumban 22050 Hz-es mintavételezési frekvenciával. Az adatok előfeldolgozásához szükség van az adathalmazra ezért, ha a program nem találja a megadott elérési úton, akkor letölti a Kaggle oldaláról, majd kicsomagolja a letöltött zip fájlt. Kicsomagolás után már használható is az előfoldolgozást végző feladat.

2.2.2. Adatok előfeldolgozása, átalakítása és lementése

Az adathalmaz megismerése után, elkezdődhet a benne lévő adatok feldolgozása. A program először végig megy az adathalmaz mindegyik mappáján. Ebben az esetben egy mappa egy műfajhoz tartozik. Az adathalmaz elérési útját a forráskódban kell definiálni. A program egy JSON fájlba fogja lementeni a feldolgozott adatokat, amit a neurális hálózat kap meg bemenetként a tanulási folyamathoz. A JSON fájlba először a műfajok elnevezését mentjük le a mappák elnevezése szerint. Ezután végig megy a program az adott mappa mindegyik hangmintáján, amikből több szegmenst készít a program és mel frekvencia cepstrális együtthatóként (MFCC) menti el őket. Mivel műfajonként 100 db hangminta kevésnek számít egy műfajmeghatározás vagy hasonló hanggal kapcsolatos probléma megoldásánál, ezért szükségünk van több szegmensre az adott mintából. A hangminták MFCC-vé alakítása Short Time Fourier transzformációval történik, amire a Librosa nevezetű Python könyvtár nyújt széles eszköztárat. Végül mindegyik MFCC-hez egy indexet mentünk le, ami alapján beazonosítható, hogy melyik műfajhoz tartoznak. Végül a program lementi a feldolgozott adatokat JSON fájlba, aminek a nevét és elérési útját szintén a forráskódban definiálhatjuk.

2.3. Neurális hálózat modell

A neurális hálózat tanulásáért felelős program használata azoknak a felhasználóknak ajánlott, akik rendelkeznek gépi tanuláshoz és neurális hálózatokhoz köthető tudással. A program célja, hogy betanítsa a műfaj meghatározás mögött álló modellt, amit a program a tanulási folyamat végén egy .h5 fájlba lement, amit, majd a főprogram fog felhasználni, amivel lehetséges egy tetszőleges zene műfaját meghatározni. A hálózat elkészítéséhez a program tensorflow-t használ. A programnak szüksége van még az előfeldolgozás során létrehozott adatokkal teli JSON fájlra, aminek az elérési útvonala a forráskódban van megadva. A JSON fájl megadása nélkül a program nem fut végig, mivel nem képes bemenettel szolgálni a hálózatnak.

3. fejezet

Fejlesztői dokumentáció

A műfajt meghatározó program és a működéséhez szükséges audió fájl előfeldolgozó és a neurális háló tanítását végző programok során az alábbi Python 3 álltal biztosított könyvtárakat használtam: numpy, librosa, mutagen, matplotlib, tinytag, json, os, math, time, tensorflow, kaggle.api, zipfile, sklearn, pydub, pygame és pyqt5.
Numpy különböző numerikus számításokhoz, míg a librosa az audió fájlok feldolgozásához nyújtott hatékony eszközöket. A programban használt konvolúciós neurális hálózat kialakításához Tensorflowt használtam. A főprogram felhasználói felületének elkészítéséhez a Pyqt5 biztosított széles eszköztárat, valamint a kiválasztott zene metaadatainak kinyerésében a mutagen és a tinytag segített.
Az audió fájlok előfeldolgozásánál a program a GTZAN Dataset-et [1] használja, amit mel-frekvencia cepstrum együtthatókká alakítva egy JSON fájlban tárol el, ami később a neurális hálózat bemenetét biztosítja.

3.1. Előfeldolgozás

Az adatok előfeldolgozása a neurális hálózatok egyik kulcsfontosságú lépése. A cél általában az adatok előkészítése olyan formába, amelyet a neurális hálózat bemeneteként tudunk használni. Szakdolgozatom során a GTZAN Dataset-ben [1] található .wav formátumú audió fájlokat fogom feldolgozni.

3.1.1. Hang

A hang egy rugalmas közegben mechanikai rezgéshullámként terjedő, fülünk által felfogható fizikai jelenség.

3.1.2. Hullámforma

A hanghullámot egy szinusz hullám segítségével lehet ábrázolni, amely egy sima, ismétlődő hullám. Megmutatja, hogy a hang amplitúdója (intenzitása) hogyan változik az idő múlásával. Azonban a hanghullámok többsége sokkal bonyolultabb, mint egy egyszerű szinusz hullám. Több szinusz hullámot tartalmaz különböző frekvenciákkal, amplitúdókkal és fázisokkal, amelyek összekapcsolódnak és egyedi hullámformát hoznak létre. A harmónikus rezgést, azaz egy egyszerű szinuszos hullámformát az alábbi formulával írhatunk le:

$$y(t) = A\sin(2\pi ft + \varphi)$$

3.1. ábra. Harmonikus rezgés

φ : a fázis azaz (phase)

3.1.3. Periódusidő

A periódusidő (T) egy időtartam ami egy teljes hullám oszcillációhoz (például az egyik maximumtól a következő maximumig) szükséges.

3.1.4. Frekvencia

A frekvencia (f) egy fizikai mennyiség, amely a hanghullámok ismétlődő ciklusainak számát méri másodpercenként, és mértékegysége a Hertz (Hz). Az emberek által hallható hangok általában a 20 Hz és 20 000 Hz közötti frekvenciákon belül helyezkednek el. Ami ezen intervallum alatt helyezkedik el azt infrahangnak, és ami felette azt ultrahangnak nevezzük. A hangmagasság és a frekvencia között szoros összefüggés van. A magasabb hangmagasságú hangoknak magasabb, míg a mélyebbeknek alacsonyabb a frekvenciája. A frekvencia a periódusidő reciproka.

$$f = \frac{1}{T}$$

3.2. ábra. Frekvencia képlete

3.1.5. Amplitúdó

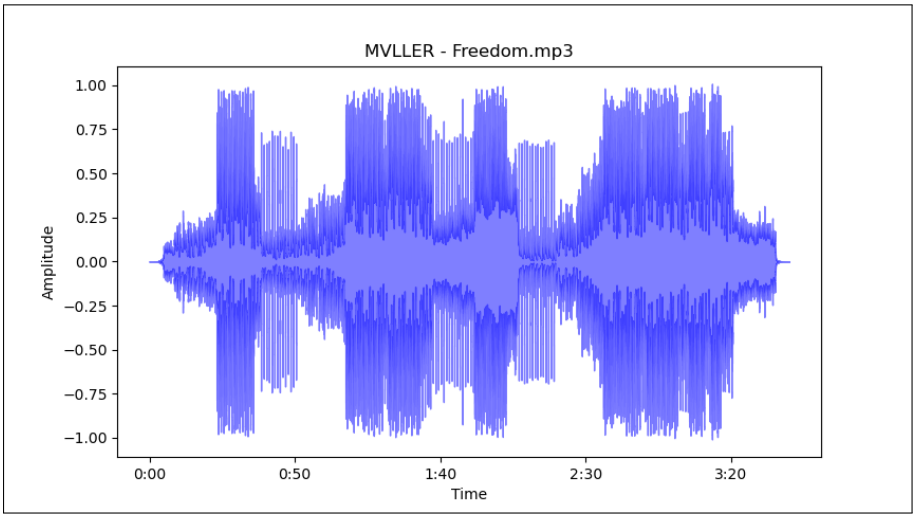
Az amplitúdó a hullámok intenzitását méri. Fontos szerepet játszik a hangok hangerejének meghatározásában. Azonos hangnyomás esetén a magasabb hangokat hangosabbnak halljuk, viszont körülbelül 4000 Hz fölött már egyre gyengébbnek.

3.1.6. Analóg - digitális konverzió (ADC)

Analóg - digitális konverziónak azt a folyamatot nevezzük, amely során az analóg jeleket, digitálissá alakítjuk át. Az analóg jelek folytonosak, míg a digitálisak diszkrétek. Az ADC pontossága a mintavételi sebességtől és a felbontástól függ. Először az analóg jelet egységes időközönként mintavételezi, majd ezeken az intervallumokon az amplitúdó kvantált értékét bináris kódként fejezi ki. A felbontását bit-ben adjuk meg. Minél nagyobb a bitmélység, annál több amplitúdó érték kerül rögzítésre.

3.1.7. Audió fájl hullámformája

Az alábbi ábrán látható, hogy ez egy jóval komplexebb hanghullám, mint a szinuszos harmonikus rezgésé.



3.3. ábra. Komplex hullámforma

3.1.8. Fourier transzformáció

A jelfeldolgozás egyik hasznos eszköze. A vizsgált folytonos jel számos tulajdonságát vizsgálhatjuk meg vele. Egy hanghullám különböző frekvenciájú és amplitúdójú periodikus függvények, mint a szinusz és koszinusz függvények összegeként előállítható. Egy komplex hangot az alábbi képlettel írhatunk le:

$$s = A_1 \sin(2\pi f_1 t + \varphi_1) + ... + A_n \sin(2\pi f_n t + \varphi_n)$$

3.4. ábra. Komplex hanghullám leírása képlettel

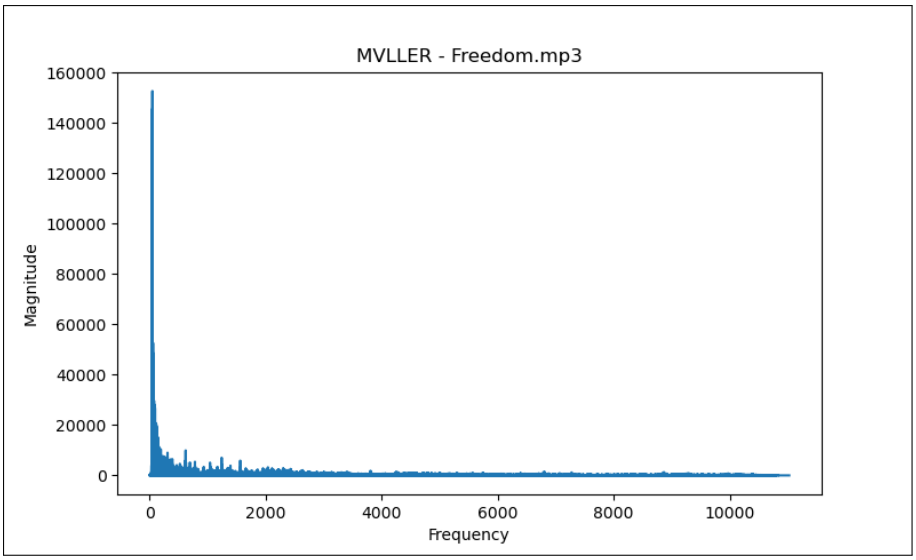
A transzformáció eredményeként a periodikus függvények, Fourier-sorba írhatóak. A definiált a tartományon kívül a függvény örökké ismétli önmagát. A Fourier-transzformáció integrálja a Fourier-sor határértéke [5].

3.1.9. Diszkrét Fourier transzformáció

A diszkrét fourier transzformáció a numerikus számítások és közelítések egyik rendkívül hasznos eszköze. Ez által diszkrét függvényeken és valós adatokon is alkalmazhatunk fourier transzformációt, viszont nagy méretű minták esetén nem a leghasznosabb. Erre nyújt megoldást a Gyors fourier transzformáció [5].

3.1.10. Fast Fourier transzformáció

Lehetővé teszi a nagy méretű diszkrét fourier transzformációk hatékony kiszámítását. Fast Fourier transzformáció (FFT) alkalmazása után, a hang intenzitás-frekvencia változás alapján reprezentálható, megkapjuk a teljes jel spektrumát. Ebben az esetben viszont az idővel kapcsolatos információkat elveszítjük.



3.5. ábra. A hang spektruma

3.1.11. Short-Time Fourier transzformáció

A Short-Time Fourier transzformáció (STFT) lehetővé teszi a jel idő- és frekvenciadoménbeli ábrázolását egyidejűleg. Segítségével úgynevezett spektogramokon is ábrázolhatjuk a vizsgált hangot. A vizsgált jel adott idejű időablakokra kerül feloszlásra, majd Fourier transzformáció alkalmazásával megkapjuk az adott idő intervallum frekvenciaspektrumát. Az időablakok között bizonyos nagyságú átfedés van. Ha a teljes jelen végbement a transzformáció, akkor a kapott információkból spektogram készíthető, ami az időablak funkció segítségével már időben is képes az adott jel intenzitását és frekvenciáját megmutatni. Ezt a transzformációt az alábbi képlettel írhatjuk le:

$$X_k(n) = \sum_{m=0}^{N-1n} w(m)x(n-m)e^{2\pi jmk/M}$$

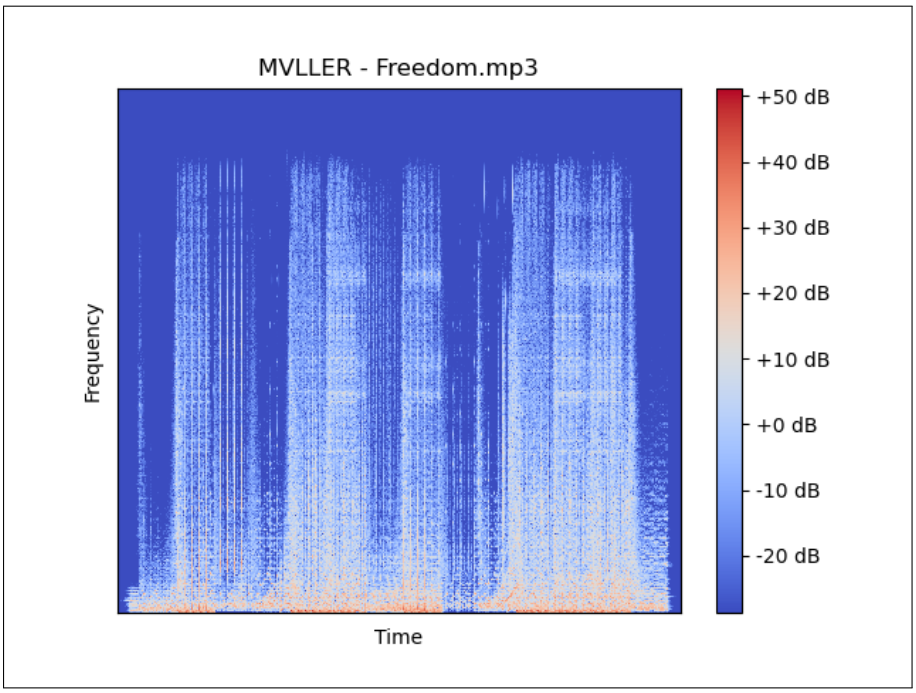
ahol
n: Idő index
k: Frekvencia index $0 \leq k < M$
N: Bemeneti pontok száma framenként
M: Transzformáció mérete ($M \geq N$)
w(m): Időablak funkció (Window function)

3.6. ábra. Short-Timw Fourier transzformáció

Ez a számítás fix időintervallumonként megy végbe amit "hop sizenak" vagy "frame ratenek" hívunk [6].

3.1.12. Spektogramok

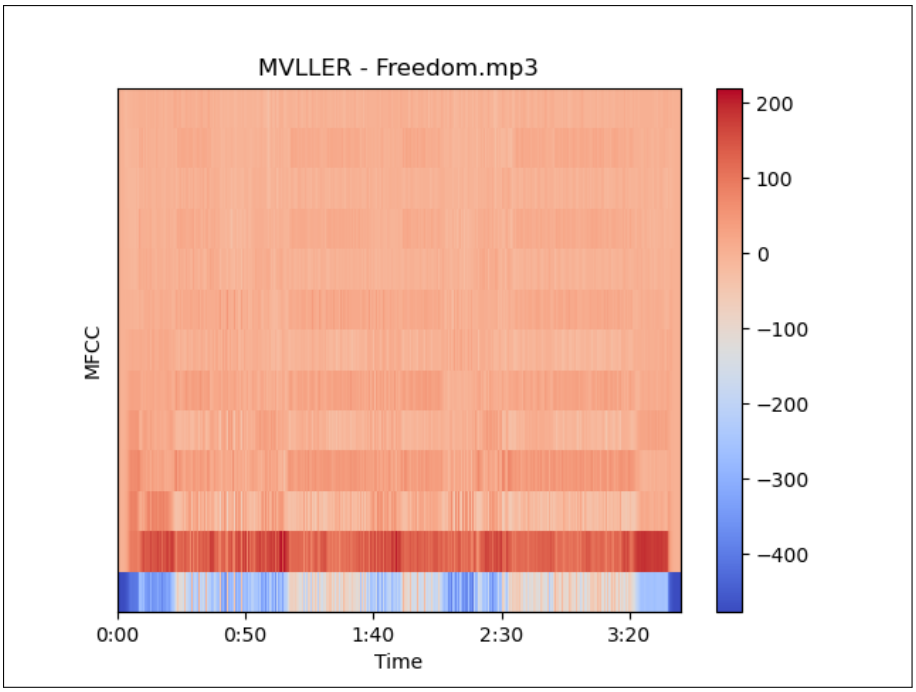
A spektogram olyan vizuális ábrázolások, amelyek megmutatják a jel frekvenciáinak időbeli változását. A STFT eredménye, Lehetővé teszik a frekvenciaváltozások dinamikus követését az idő függvényében, és segíthetnek azonosítani a jel tulajdonságait. A spektrogramokat általában színskála segítségével ábrázolják, ahol az erősebb frekvenciák világosabb, míg a gyengébbek sötétebb színnel jelennek meg.



3.7. ábra. Hang megjelenítése spektogramként

3.1.13. Mel-frekvencia cepstrum együtthatók

Az emberi hallási folyamatot utánozza, és a jel frekvencia tartományát olyan sávokra osztja fel, amelyek a hallási érzékenységünkhöz hasonlóak. Létrehozása során szintén időablakokra osztjuk a jelet, amiken Fourier transzformációt végzünk el és megkapjuk az adott idő szakasz spektrumát. A spektrumokat frekvenciasávokra osztjuk, amiket végül a mel skála szerint súlyozunk és egy adott nagyságú MFCC keretbe rendezünk. Az MFCC-k lehetővé teszik a hangjellemzők hatékonyabb kinyerését.



3.8. ábra. Hang átalakítva mel-frekvencia cepstrum együtthatóvá

3.1.14. Adathalmaz előfeldolgozása

Az adathalmaz előfeldolgozásáért felelős forráskódban először definiálni kell az elérési útvonalát. Mielőtt a program bármilyen adat feldolgozását elkezdené, először leellenőrzi, hogy a megadott elérési út valóban létezik-e. Ha a megadott elérési útvonalon nem találja az adathalmazt, akkor azt a kaggle.api segítségével először letölti, majd a letöltött zip fájlt kicsomagolja. Az adathalmaz mappáján belül mindegyik mappa egy-egy műfaj hangmintáit tartalmazza. Ha később a datasetet műfajainak bővítése szükséges lenne, akkor a műfajhoz tartozó hangmintákat egy külön a műfaj megnevezésével ellátott mappában kell tárolni. A program ezeken a mappákon megy végig és a benne lévő hangmintákat dolgozza fel. Az adathalmaz 10 műfajt és mindegyikhez 100 db hangmintát tartalmaz. Hangfelismerés, műfajfelismerés és hasonló hanggal kapcsolatos feladatok esetén ez nem számít még sok adatnak, ezért több szegmenst hozunk létre egy hangmintából, amit a num segments nevű változóval lehet megadni. Az adathalmaz előkészítési folyamatához, Valerio Velardo 19 részes Deep Learning For Audio With Python sorozata nyújtott segítséget [7].

3.1.15. Adatok átalakítása mel-frekvencia cepstrum együtthatókká

A hangminták mel-frekvencia cepstrum együtthatókká alakítása során, a függvénynek szüksége van a zenék mintavételezési frekvenciájára, ami az adathalmaz esetében 22050 Hz, FFT, mel-frekvencia együtthatók számára és az időablak függvény átfedésének nagyságára. Az alábbi librosa funkció végzi az átalakítást:

```
mfcc = librosa.feature.mfcc(y=signal[start_sample:finish_sample],
                                                    sr=sr,
                                                    n_fft=n_fft,
                                                    n_mfcc=n_mfcc,
                                                    hop_length=
                                                        hop_length)
```

3.1. forráskód. Hangminták átalakítása mel-frekvencia cepstrum együtthatókká

3.1.16. Feldolgozott adatok lementése JSON fájlba

Pontos eredmény elérésének érdekében fontos, hogy az adathalmazunk azonos hosszúságú zenéket dolgozzon fel. A program készítése során használt adathalmaz 30 másodperces mintákat tartalmaz. Ahhoz, hogy az átalakított mfcc vektorokat JSON fájlba lementse a program, először leellenőrzi azok hosszát, hogy megegyezik-e a várható mfcc vektorok számával szegmensenként. Ezt a várható értéket az alábbi módon számolhatjuk ki:

$$\text{mintavételezés hangmintánként} = \text{mintavételezési frekvencia} \cdot \text{hangminta hossza}$$

$$\text{mintavételezés száma szegmensenként} = \frac{\text{mintavételezés hangmintánként}}{\text{szegmensek száma}}$$

$$\text{várható mfcc vektorok száma} = \frac{\text{mintavételezés száma szegmensenként}}{\text{hop length nagysága}}$$

Végül létrejön a JSON fájl ami tartalmazni fogja az átalakított adatokat. Az indent paraméterrel formailag testreszabhatjuk a fájlban lévő értékek behúzását.

```
with open(json_path, "w") as fp:
    json.dump(data, fp, indent=4)
```

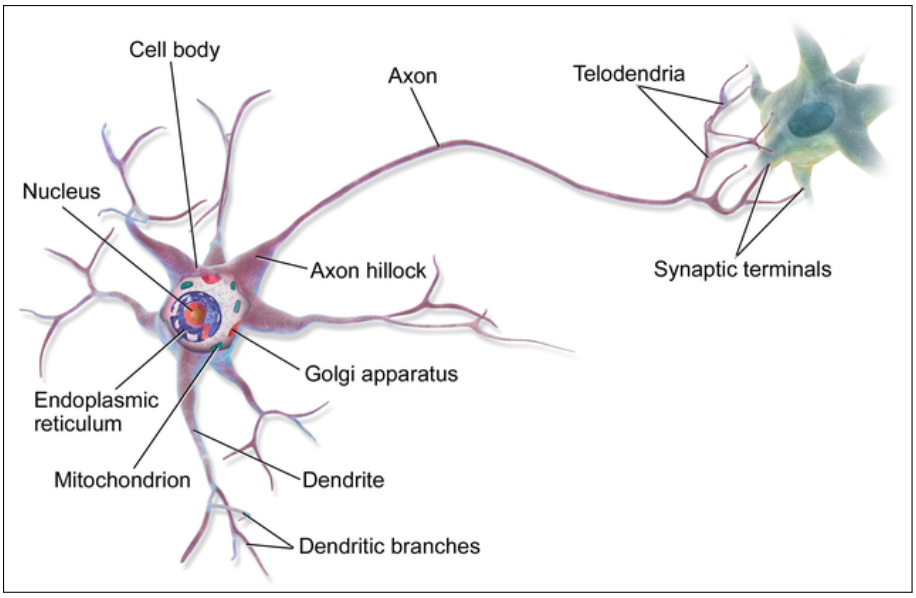
3.2. forráskód. JSON fájl létrehozása

3.2. Neurális hálózatok

A neurális hálózat egy olyan összetett rendszer, ami a biológiai idegrendszer működésének mintáját követi. Ezek az algoritmusok az adatokból tanulnak, azaz képesek megtanulni azokat az összefüggéseket, amelyek alapján később képesek új adatokat megjósolni vagy osztályozni. Nagyon sok területen felhasználhatóak, mint például kép- és hangfelismerésben, szövegértésben és a dolgozatom témájában, egy adott zene műfajának meghatározásában.

3.2.1. Biológiai neuron

Az emberi agy a legösszetettebb struktúra, aminek működésének megértése az egyik legizgalmasabb és legnehezebb kihivás a tudósok számára. A biológiai neuronok az emberi idegrendszer alapegységei, amelyek az információ feldolgozásáért és továbbításáért felelősek az agyban és a gerincvelőben. Az emberi agy körülbelül 10^{11} elektromosan aktív sejtet, úgynevezett neuront tartalmaz. Általában három fő részből állnak: dendrit, sejttest és az axon. A dendritek felelnek a neuronokra érkező bemeneti jelekért, míg az axon az idegi impulzusokat továbbítja, azaz kimenetként funkcionál. A neuronok közötti kommunikáció szinapszisokon keresztül történik, aminek a száma az emberi agyban meghaladja a 10^{14} darabszámot. Minden szinapszishoz tartozik egy súly, ami meghatározza az impulzus hatásának nagyságát [8].



3.9. ábra. Egy biológiai neuron felépítése [9]

3.2.2. Mesterséges neuron

A mesterséges neuron egy olyan matematikai modell, amely az idegrendszerben található biológiai neuronok működési elvét utánozza. Felépítésben is hasonlóak, mivel ebben az esetben is van egy bemenetünk, egy súlyozott összegző egység és az aktivációs függvény ami meghatározza, hogy milyen kimeneti jelet generál. A McCulloch-Pitts modell alapján a bemeneti értékünk x_i az i-dik bemenetünknél, amit először a hozzátartozó súly w_i értékével megszorzunk, majd összeadjuk az összes többi súlyozott bemeneti jellel. Ezt az alábbi formulával írhatjuk le:

$$a = \sum_{i=1}^{d} w_i x_i + w_0,$$

ahol az w_0 eltolási paramétert bias-nak nevezünk [8].

3.2.3. Aktivációs függvény

A neurális hálózatok során legtöbbször a szigmoid aktivációs függvénnyel találkozunk, ami egy S alakú görbe ként ábrázolható. Ezen kívül létezik küszöb, szakaszonként lineáris, tangens hiperbolikus függvény is. A szigmoid függvény az alábbi formulával írható le, ahol a a bemenetek és hozzátartozó súlyok szorzatának összege:

$$y = \frac{1}{1 + e^{-a}}$$

ReLu (Rectified Linear Unit) is egy gyakran használt aktivációs függvény a neurális hálók alkalmazása során. A ReLU függvény kimenete mindig a bemenet és nulla közötti maximumot veszi fel. Ha negatív a bemenetünk, akkor a kimenet nulla lesz, ha pozitív, akkor azonos a bemenettel [8].

3.2.4. Réteg

Egy neurális hálózat több rétegből áll, amiknek részegységei a neuronok. A hálózat legelső rétege az input réteg, ami a bemeneti adatokat fogadja. Ezek általában vektorok, tömbök. A rejtett réteg általában több rétegből áll. Feldolgozzák a bemenetet, és különböző számításokat végeznek el rajta. A neurális hálózat legvégén található réteg pedig a kimeneti réteg. Ebben a rétegben az aktivációs függvény által generált kimeneti jel a végleges eredményt jelenti.

3.2.5. A neurális hálózat tanítása és hiba visszaterjesztése (Backpropagation)

A hálózat tanítása (training) egy olyan folyamatot, amely során úgy finomítja a hálózat súlyait és a bias értékeit, hogy azok jobban illeszkedjenek a bemenő adatokhoz, és jobb eredményeket érjenek el a kimeneti adatokkal.
Felügyelt tanulás esetén ismerjük bemeneti adataink kimenetelét, ezáltal lehetőségünk van hibaszámításra. A neurális hálózat kimenetelét összehasonlíthatjuk a várható értékekkel, így tudunk hibát számolni. A tanítás során a legfőbb cél, hogy ezt a hibaértéket minimalizáljuk, ezáltal később pontosabb eredményeket kapunk. Hiba számításra az alábbi formulát alkalmazzuk:

$$E = \frac{1}{2}\sum_{q=1}^{n}(y(x^q; w) - t^q)^2,$$

ahol n az adatpontok száma,
q az adatpontok indexe $q = 1, ...n$
$y(x^q; w)$ a polinomiális függvény által kapott kimeneti értékek
és t^q a várt eredmény (target).

3.10. ábra. Hiba számítása, a hiba értékének minimalizálásához [8]

3.2.6. Gradiens módszer (Gradient Descent)

A veszteség minimalizálására a gradiens módszer egy hasznos optimalizációs algoritmus. Ezzel az iteratív módszerrel meghatározhatjuk a veszteség lehetséges lokális minimumát. Ezt úgy tudjuk elérni, hogy a gradiens irányával ellentétesen a lehetséges lokális minimum irányába teszünk egy lépést. Ezt konvergencia eléréséig folytatjuk. A konvergenciát a lépések nagysága is befolyásolja, ami a tanulási tényezőt (learning rate) is meghatároz. Alacsony tanulási tényező használatakor rendkívül hosszú ideig is eltarthat a tanulási folyamat, míg ellenben a nagy tanulási tényező használatakor a tanulás divergens is lehet.

3.2.7. Konvolúciós neurális hálózatok

Konvulóciós neurális hálózatokat többnyire képfeldolgozással kapcsolatos feladatokra használják. Mivel hangokat képesek vagyunk különböző spektogramok és melfrekvencia cepstrumokként ábrázolni, ezért gyakran használják zajszűréssel, hangfelismeréssel vagy szakdolgozatom témájára, azaz műfaj meghatározási feladatok

elvégzésére.

Az egyik legfontosabb különbség a többi mesterséges neurális hálózathoz képest, hogy a rétegek neuronjai három dimenzióba, magasság, szélesség és mélységbe szervezett neuronokból állnak, amiket kernelnek hívunk. A CNN-ek három fő rétegből állnak. Konvolúciós rétegek, pooling rétegek és teljesen összekapcsolt rétegekből.

A bemenet az adott kép pixeleinek értékét tárolja. Ahogy a nevében is benne van, a konvolúciós réteg létfontosságú szerepet játszik a CNN-ek működésében. A réteg tanulható kernelek használatán épül. Általában kis térbeli dimenzióba (magasság x szélesség x mélység) alkalmazzák őket, viszont végigléptetik az egész bemenet teljes mélységén. Ahogy végigmegyünk a bemeneten, kiragadunk a kernel méretével megegyező tenzort, majd minden egyes értékére kiszámoljuk a skalásis szorzatukat. Végül a kapott értéket a kimeneti tenzorban a megfelelő helyen rögzítjük. A konvolúciós rétegek kimeneti rétegét három hiperparaméterrel képesek vagyunk optimalizálni. A mélységgel, ami egy rgb kép esetén a 3 színcsatorna száma, a lépések nagyságával, amivel a kernelt léptetjük végig a bemeneten és egy úgy nevezett zero paddinggal, ami a bemenetünk köré egy 0 értékekkel teli keretet ad. Zero padding alkalmazásával megváltoztathatjuk a kimenet térbeli dimenzionalitását, aminek kiszámításához az alábbi képletet alkalmazhatjuk:

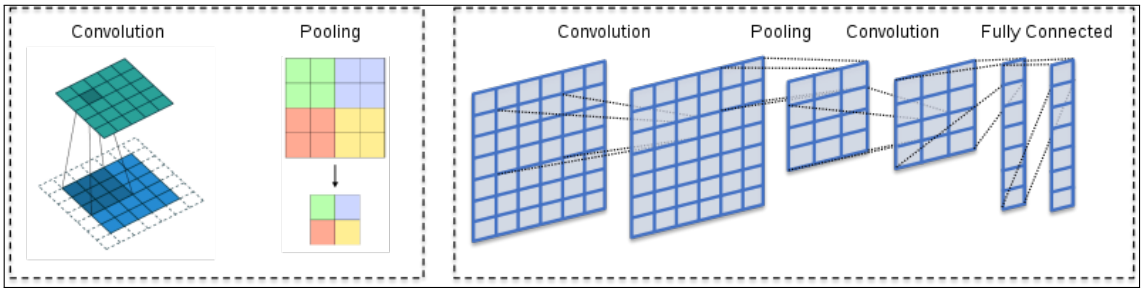
$$\frac{(V - R) + 2Z}{S + 1},$$

3.11. ábra. Kimenet térbeli dimenzionalitása zero padding alkalmazásával

ahol V a bemenet méretét (magasság x szélesség x mélység), R a receptív mező mérete, ami a bemenet azon része, amit az egyes neuronok látnak, Z a zero padding mérete és S a léptetések (stride) nagysága. Mivel egy konvolúciós rétegben több kernel is létezhet, ezért az adott réteg kimenetén kapott 2 dimenziós tömbök száma megegyezik a kernelek számával.

A pooling réteg célja a dimenzionalitás csökkentése, ezáltal csökkenti az egyes paraméterek számát, és a modell számításigényét. Két legismertebb típusa a max pooling és az avarage pooling. Max pooling során a bemenetből kiragadott részletből a legnagyobb értéket rögzítjük a kimenet megfelelő helyén. Avarage pooling esetén az értékek átlagát számoljuk ki. Ha pooling során 2x2-es dimenziójú kernelt alkalma-

zunk 2 nagyságú léptetéssel, akkor a negyedére csökkentjük a kimenetet miközben megőrizzük a mélységet [10].

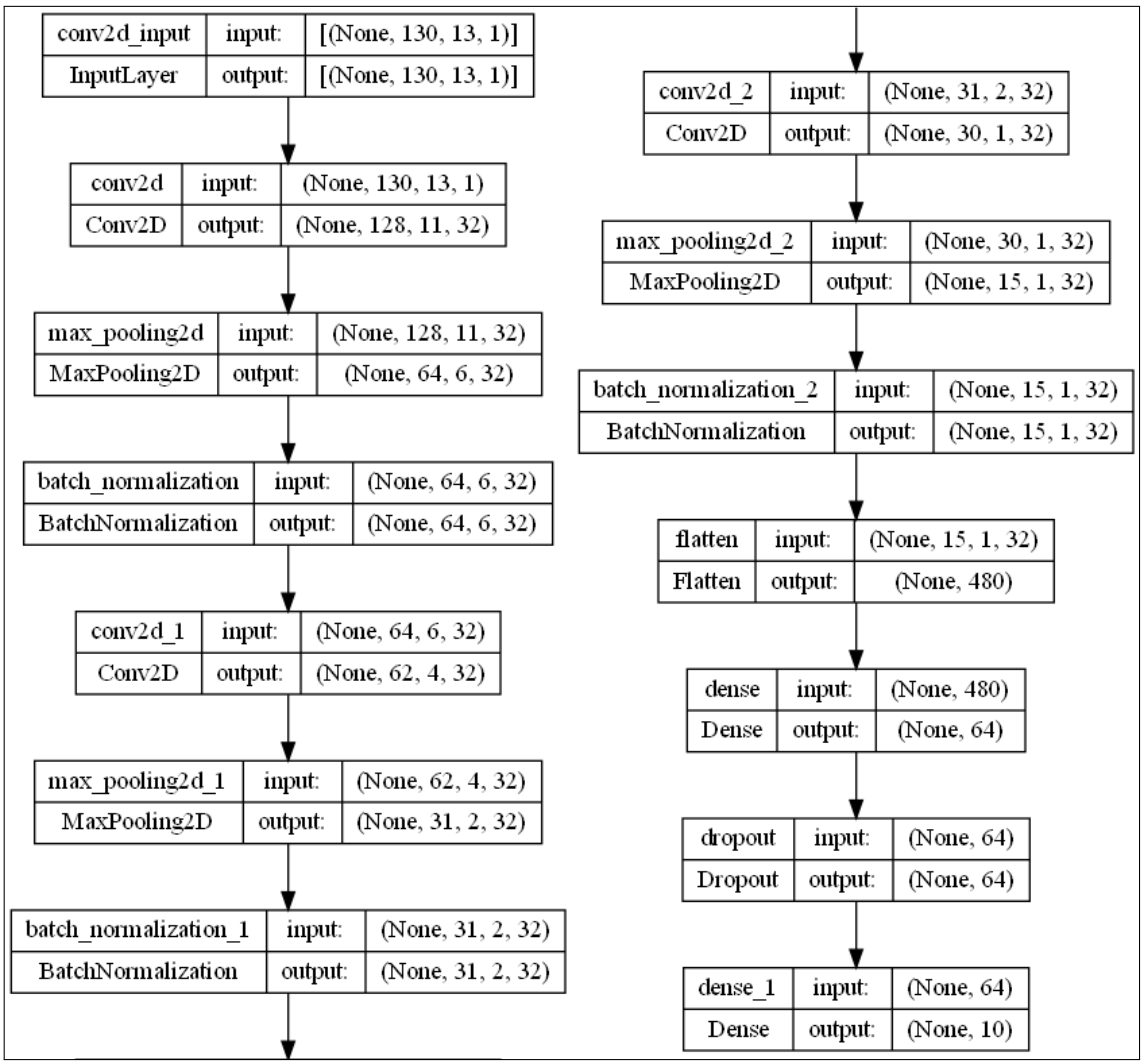


3.12. ábra. Konvolúciós és a pooling réteg [11]

3.2.8. Feldolgozott adathalmaz betöltése

A program működéséhez szükséges, hogy először az előfeldolgozás során lementett JSON fájlt sikeresen betöltse. A fájlba lementett MFCC-k lesznek a hálózat bemenetei és a hozzá tartozó indexek a kimenetelek, mivel a futtatás végén ez a program is képes predikciót hozni. A bemeneti és kimeneti adatokat először training, teszt és validációs részekre ossza a program. A training halmazon tanul a hálózat, a teszt halmaz értékei érintetlenül maradnak a tanulás során, a validációs egység értékei pedig a kiértékelésben segítenek. Ezután a bemeneti értékeket mindhárom egység esetében (teszt, training, validáció) 4 dimenziós tömbbé kell alakítani őket, ahol az értékek a következők lesznek: (minták száma, várható MFCC vektorok száma, MFCC-k száma, és a mélység). A neurális hálózat tanítási folyamatának kódban történő alkalmazásához, segítséget nyújtott Valerio Velardo 19 részes Deep Learning (For Audio) With Python sorozata [7].

3.2.9. Konvolúciós neurális hálózat modell felépítése



3.13. ábra. Konvolúciós neurális hálózat felépítése

A modell felépítése szekvenciális. A bemeneti réteg után három konvolúciós réteggel rendelkezik a hálózat, ahol mindegyiket egy max pooling és egy batch normalization réteg követi. Ezek után egy flatten réteg következik, ami a több dimenziós bemenetet, egy dimenziósra csökkenti. Ezt követően egy dense réteg, majd a túltanulás elkerülése miatt egy dropout réteg is szerepel. A kimeneti réteg egy dense réteg 10 lehetséges kimenettel, amik a műfajok lesznek. A kimenet aktivációs függvénye softmax, mivel a legvalószínűbb műfajt írja ki a program predikciónak. A modell során használt optimalizáló az Adam és a veszteség függvény a keresztentrópia. A tanulási folyamat során a teljes training halmazon végigmenő tanulás, azaz epochok nagyságát a forráskódban adható meg a model.fit függvény egyik paramétereként.

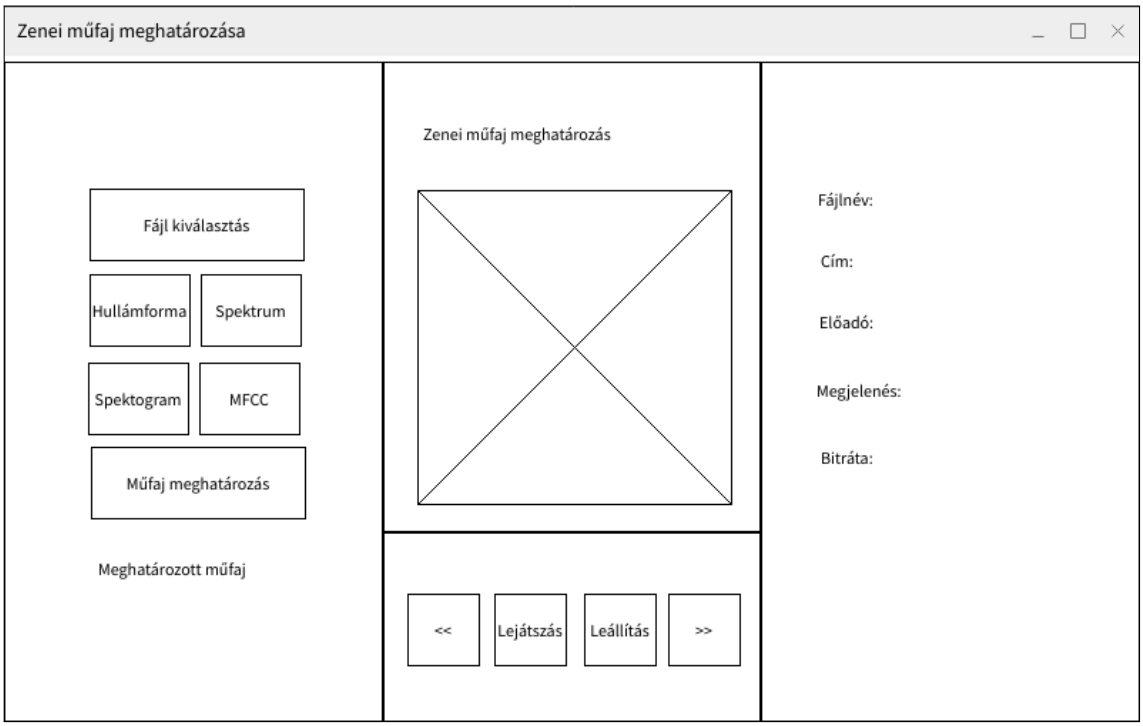
A tanulási folyamat befejezése után a modell elmentésre kerül egy .h5 fájlban és kiértékelődik a hálózat pontossága a teszt halmazon.

3.3. Műfaj meghatározó program

A műfaj meghatározó program, az a program, ami főként a felhasználóknak lett készítve. A három program közül egyedül ez rendelkezik grafikus felhasználói felülettel és ez, ami a legtöbb felhasználási lehetőséget és funkciót tartalmazza. A program futtatásához Python 3-as környezet szükséges és az alábbi csomagok: json, math, librosa, pydub, numpy, matplotlib, sklearn, tensorflow, time, pygame, PyQt5, mutagen, tinytag, sys és os.

3.3.1. Felhasználói felület

A felhasználói felület a PyQt5 Python csomaghoz tartozó QtDesignerrel készült. A felület 4 fő egységből épül fel. A végleges felhasználói felület elkészítése előtt drótvázterv készült az elképzelt kinézetről, ami tartalmazza a programba beépített funkciókat.



3.14. ábra. A műfaj meghatározó program felhasználói felületének drótvázterve

A bal oldalt elhelyezkedő rész, főleg QPushButton komponensekből áll, amikhez funkciók vannak hozzákötve. Itt érhetjük el a fájl kiválasztását, készíthetünk plotot belőle és meghatározhatjuk a műfaját, amit végül ki is ír. A program közepén elhelyezkedő terület két részre osztható. A felső rész a kiválasztott zene borítóképét jeleníti meg egy QLabelen QPixmap használatával. Borítókép csak abban az esetben jelenik meg, ha az adott zene metaadatában található borítóképpel kapcsolatos adat. A zenelejátszó egység a program alján helyezkedik el, amivel belehallgathat a felhasználó az általa kiválasztott hangfájlba. A gombok ebben az esetben is QPushButton komponenssel készültek, amik border-image-ként kaptak a funkciójukat jól szimbolizáló ikont. A program jobb oldalán a kiválasztott fájl fontosabb metaadatai jelennek meg, mint például a cím, előadó, megjelenés éve és a bitráta, valamint a fájlnév is kiírásra kerül. Ezek a komponensek QLabel-ként jelennek meg.

3.3.2. Fájl megnyitása

A fájlok megnyitását QFileDialog.getOpenFileName használatával éri el a program. Ezt a program egy filepath nevű változóban tárolja, hogy más funkciók használata során is fel lehessen használni. Ebben a függvényben adható meg, hogy milyen formátumú fájlokat engedjen megnyitni a program. A fájl kiválasztása, valamint útvonalának elérése az alábbi kódsorral lehetséges:

```
1 self.filepath = QFileDialog.getOpenFileName(self, 'Valassz ki egy
    audio fajlt!', '.', 'Audio Files (*.wav *.flac *.mp3)')
```

3.3. forráskód. Fájl kiválasztása és az elérési útvonal eltárolása

A kódsor lefutása után a program ellenőrzi, hogy történt-e fájl kiválasztás vagy sem. Ha történt kiválasztás, akkor kinyeri a fájl metaadatait és betölti őket, valamint a zenelejátszóba is betöltődik a hangfájl.

3.3.3. Metaadatok betöltése

Metaadatok hangfájlból való kinyerésére a TinyTag könyvtárat használtam. A program által támogatott három (.wav, .flac, .mp3) fájlformátum esetében sikeresen megszerezte a betöltött adat információit, ha az rendelkezett metaadatokkal. Az egyes metaadat címkéket a TinyTag.get() függvénnyel lehet megszerezni. Paraméternek a fájl elérési útvonalát meg kell adni, majd a megszerzendő címkéhez tartozó funkciót.

```
#Audio metadata tag-jeinek kinyerese
audio_metadata = TinyTag.get(self.filepath[0])
#Zene cim
songtitle = audio_metadata.title
#Eloado
songartist = audio_metadata.artist
#Megjelenes eve
songrelease = audio_metadata.year
#Bitrata
songbitrate = audio_metadata.bitrate
```

3.4. forráskód. Metaadat címkék megszerzése és megjelenítése

3.3.4. Zene borító betöltése

A zene borító betöltése egy PyQt5 QLabelen történik QPixmap használatával. QPixmap megkapja a kiválasztott zene metaadatában tárolt borítóképet, ha rendelkezik a hangfájl borítóval. Ha nem található borító, akkor a program indításakor is megjelenő alapértelmezett kép jelenik meg. Zeneborító megszerzésére a mutagen könyvtár id3, wave és flac címkékhez használatos csomagjait használtam.

3.3.5. Plot készítése a betöltött hangfájlból

A program hullámforma funkciójával, a kiválasztott hangfájl hullámformáját plotként lehet megjeleníteni. Librosa nevű könyvtár segítségével a hangfájlok könnyedén megjeleníthetőek egy matplotlib ploton. Először librosa betölti a kiválasztott fájlt és meg kell adni a mintavételezési frekvencia értékét, ami sr=None esetén a betöltött fájl mintavételezési frekvenciája, valamint szükség van a hangfájl hosszára. Végül librosa.display.waveshow funkcióval létrehozzuk a hullámformát, amit egy ploton megjelenítünk. Hullámforma megjelenítését az alábbi kód részlet valósítja meg:

```
def show_waveform(self):
    try:
        file = self.opened_file
        #Hanghullam hossza
        dur = mediainfo(file)["duration"]
        #Fajl megadasa librosanak
```

```
7         signal, sr = librosa.load(file, sr=None, duration=math.
              floor(float(dur)))
8         #Hullamforma letrehozasa
9         librosa.display.waveshow(signal, sr=sr, alpha=0.5, color='b
              ')
10        #Plot elkeszitese
11        plt.title(self.filename)
12        plt.xlabel("Time")
13        plt.ylabel("Amplitude")
14        plt.show()
15    except Exception as e:
16        print(e)
```

3.5. forráskód. Hullámforma megjelenítése Librosaval

A programban a frekvencia spektrum megjelenítésére is van lehetősége. Hullámforma megjeleítéshez hasonlóan ebben az esetben is librosanak meg kell adni a betöltött fájl elérését, mintavételezési frekvenciáját és hosszát. Egy hang spektrumának megjelenítéséhez FFT szükséges. Ezt a funkciót az alábbi kód hajtja végre:

```
1  def show_spectrum(self):
2      try:
3          file = self.opened_file
4          #Spektrum
5          dur = mediainfo(file)["duration"]
6          signal, sr = librosa.load(file, sr=22050, duration=math.
               floor(float(dur))) # sr(SampleRate) * T(duration)
7          #Fast Fourier Transzformacio
8          fft = np.fft.fft(signal)
9          #Hang intenzitas es frekvencia megadasa
10         magnitude = np.abs(fft)
11         frequency = np.linspace(0, sr, len(magnitude))  #Egyenletes
               tavolsagban levo szamok szama egy intervallumban
12         left_frequency = frequency[:int(len(frequency) / 2)]
13         left_magnitude = magnitude[:int(len(frequency) / 2)]
14
15         plt.plot(left_frequency, left_magnitude)
16         plt.title(self.filename)
17         plt.xlabel("Frequency")
18         plt.ylabel("Magnitude")
19         plt.show()
20     except Exception as e:
```

```
21          print(e)
```

3.6. forráskód. Frekvencia hang intenzitás spektrum megjelenítése Librosaval

Spektogram megjelenítése szintén lehetséges. Spektogram megjelenítése esetén szükség van a FFT-k számát (n fft), valamint az időablak léptetésének nagyságát (hop length) definiálni. A plot megjelenítése előtt még az amplitúdó értékeket dB-re átalakítjuk, hogy a spektogram könnyebben értelmezhető legyen.

```
1  def show_spectogram(self):
2      try:
3          file = self.opened_file
4          #Spektogram
5          dur = mediainfo(file)["duration"]
6          signal, sr = librosa.load(file, sr=None, duration=math.
               floor(float(dur))) # sr(SampleRate) * T(duration)
7          n_fft = 2048  #Mintak szama
8          hop_length = 512  #Eltolas jobbra
9
10         stft = librosa.core.stft(signal, hop_length=hop_length,
               n_fft=n_fft)
11         spectogram = np.abs(stft)
12
13         log_spectogram = librosa.amplitude_to_db(spectogram)  #
               Amplitudo atalakitasa dB-re
14
15         librosa.display.specshow(log_spectogram, sr=sr, hop_length=
               hop_length)
16
17         plt.title(self.filename)
18         plt.xlabel("Time")
19         plt.ylabel("Frequency")
20         plt.colorbar(format="%+ 2.0f dB")
21         plt.show()
22     except Exception as e:
23         print(e)
```

3.7. forráskód. Spektogram megjelenítése Librosaval

Úgy ahogy a spektogramok esetében, a mel-frekvencia cepstrum együtthatók (MFCC) megjelenítése során is szükség van a FFT-k számának és az időablak mozgatásának nagyságának megadására. MFCC készítés során a mel-frekvencia cepstrum

együtthatók számát is meg kell adni, ami a program esetében 13. Ezt az értéket hangfelismerés és osztályozási feladatokban 12-20 között ajánlott megadni.

```
def show_MFCC(self):
    try:
        file = self.opened_file
        #MFCC
        dur = mediainfo(file)["duration"]
        signal, sr = librosa.load(file, sr=None, duration=math.
            floor(float(dur))) # sr(SampleRate) * T(duration)
        n_fft = 2048  #Mintak szama
        hop_length = 512  #Eltolas jobbra

        MFCCs = librosa.feature.mfcc(y=signal, n_fft=n_fft,
            hop_length=hop_length,
                                     n_mfcc=13)

        librosa.display.specshow(MFCCs, sr=sr, hop_length=
            hop_length)

        plt.title(self.filename)
        plt.xlabel("Time")
        plt.ylabel("MFCC")
        plt.colorbar()
        plt.show()
    except Exception as e:
        print(e)
```

3.8. forráskód. Mel-frekvencia cepstrum együtthatók megjelenítése Librosaval

3.3.6. Zenelejátszó

Az audio fájlok lejátszására a pygame nyújtott széles eszköztárat. A zenelejátszó képes a hanganyagokat elindítani, szüneteltetni, folytatni, leállítani, előre pörgetni és hátra pörgetni. A felsorolt funkciók használata előtt inicializálni kell a lejétszót, ami az alábbi kódsorral történik:

```
pygame.mixer.init()
```

3.9. forráskód. Zenelejátszó inicializálása

Mivel elindítás és folytatás két külön függvény a könyvtárban, ezért mindig le kell ellenőrizni, hogy a megnyitott hangfájl már volt elindítva vagy még nem. Az audió fájl aktuális állapotát is ellenőrizni kell, az indítás, szüneteltetés, folytatás gomb megnyomásakor. A hangfájlban történő léptetés során az indítástól eltelt időt is folyamatosan számolni kell, így a léptető gombok megnyomásával előre vagy hátra 5 másodpercet tudunk a hanganyagban ugrani. A lejátszó gombjai PyQt5 QPushButton komponenssel készültek, és setStyleSheet funkcióval border-image -ként jelennek meg a lejátszó gombjai. A gombokhoz tartozó feladatok a clicked esemény bekövetkezésekor kerülnek meghívásra.

```
#Hangfajl betoltese
pygame.mixer.music.load()
#Hangfajl lejatszasa
pygame.mixer.music.play()
#Hangfajl szuneteltetese
pygame.mixer.music.pause()
#Hangfajl folytatasa
pygame.mixer.music.unpause()
#Hangfajl leallitasa
pygame.mixer.music.stop()
#Hangfajl allapota
pygame.mixer.music.get.busy()
#Hangfajl poziciojanak megadasa
pygame.mixer.music.set_pos()
```

3.10. forráskód. Pygame függvények

3.3.7. Műfaj meghatározása

Műfaj meghatározása gomb megnyomásakor, a megnyitott hangfájlunkat először mel-frekvencia cepstrum együtthatókká alakítja át a program, úgy, ahogy az előfeldolgozó programnál is történt. Mivel a háló által tanult adatokból is több szegmenst hoztunk létre, ezért a kiválasztott zenénket is megegyező hosszúsású szegmensekre kell darabolni, hogy a konvolúciós háló elfogadja bemenetnek és feldolgozás után pontos predikciót hozzon létre. Előfeldolgozáskor az adathalmaz 30 másodperces mintáit 10 szegmensre osztotta a program, ami szegmensenként 3 másodperc, ezért predikció előtt a betöltött hangfájlt is 3 másodperces szakaszokra kell vagdosni. Ezután betöltődik a neurális hálózat modell, amit egy .h5 fájlba mentett le a taní-

tást végző program. Az audió fájl részekre való darabolása után, mindegyik egységet tovább adjuk a betöltött modellnek és hoz rá egy predikciót, ami az adathalmazban szereplő 10 műfaj közül kerül meghatározásra. Ezeket a predikciókat egy tömbben eltároljuk és a leggyakrabban előfordult műfajt tekintjük az adott zene műfajának. A hangfájl szegmensekre osztása főként az olyan hosszabb fájloknál hasznos, ahol két műfaj keveredik az adott hanganyag során.

3.4. Tesztelés

3.4.1. Nem található az adathalmaz

Az előfeldolgozás megkezdéséhez szükséges, hogy a program könyvtárán belül elérhető legyen az adathalmaz. Abban az esetben, ha ezek az adatok nem találhatóak, akkor a program a kaggle.api segítségével letölti a Kaggle oldalán található GTZAN Dataset-et. [1] Ezután kicsomagolja a letöltött zip fájlt és kezdetét veszi a hangminták feldolgozása és lementése JSON fájlba.

3.4.2. Másik adathalmaz használata

Használható másik adathalmaz előfeldolgozáshoz, ha könyvtárának szerkezete megegyezik az eredetileg használt könyvtár szerkezetével. Ilyenkor az előfeldolgozás folyamat lefut, viszont a neurális hálózat tanítása után kapott modell pontatlan lehet. Ha az adathalmaz szerkezetileg eltérő, akkor a program nem fut le.

3.4.3. Tanulási folyamat előfeldolgozás nélkül

Ha a program az előfeldolgozás végén létrehozott JSON fájlt nem találja, akkor a program nem fut le, mert ebben találhatóak azok az értékek, amik a minél pontosabb predikció eléréséhez szükségesek. Adatok nélkül a hálózat nem tudja elindítani a tanulási folyamatot. Ebben az esetben a felhasználónak először az előfeldolgozásért felelős programot kell elindítania, majd ha az sikeresen létrehozta az adatokkal feltöltött JSON fájlt, csak utána lehetséges a tanulási folyamat elindítása és sikeres lefutása.

3.4.4. Műfaj meghatározás különböző minőségű hangminták esetén

Ezt a tesztet az adathalmaz jazz hangmintái közül 6-ot kiválasztva végeztem el (jazz.00000.wav, jazz.00005.wav, jazz.00010.wav, jazz.00015.wav, jazz.00020.wav, jazz.00025.wav). Mivel mindegyik minta 30 másodperc hosszú, ezért ezeket egy hangfájlba egyesítettem és az így kapott 3 perces hangmintán végeztem el a tesztet. Az így kapott egyesített audió fájl $6 \cdot 10$, azaz 60 db 3 másodperces szegmensre osztható. A következő táblázat ezen a 3 perces jazz hangfájlon végzett műfajmeghatározás eredményeit szemlélteti különböző minőségű audió formátumokban:

Fájl formátum	**Meghatározott műfaj**	**Szegmensek**
Wav 24 Bit int és FLAC 16 Bit int	jazz	50 db jazz 5 db klasszikus 2 db blues és country 1 db hiphop
Mp3 320, 256, 128 kb/s	jazz	40 db jazz 10 db klasszikus 5 db blues 3 db country 1 db hiphop és reggae
Mp3 96 kb/s	jazz	39 db jazz 11 db klasszikus 5 db blues 2 db country és reggae 1 db hiphop
Mp3 32 kb/s	blues	18 db blues 13 db reggae 8 db rock 6 db jazz, country, klasszikus 1 db hiphop, metál, pop
Telefonos hangfelvétel (Mp3 102 kb/s)	klasszikus	45 db klasszikus 8 db blues 5 db jazz 2 db country,

3.1. táblázat. Minőség és műfaj meghatározás közötti kapcsolat

A táblázat alapján elmondható, hogy rosszabb minőségű hangfájlok esetén pontatlan a műfaj meghatározás. Ezért tiszta és jó minőségű részletes audió fájlokkal javasolt a program használata.

3.4.5. Más fájlformátum betöltése

A program normális működés során csak wav, flac és mp3 fájlformátumokat enged megnyitni a felugró fájl kiválasztó ablakból. A kód tartalmaz egy olyan elágazást, ami figyeli a kiválasztott fájl kiterjesztését, ezért például egy png vagy jpeg fájl megnyitása során hibaüzenetet kap a felhasználó, hogy a megnyitott fájl formátum nem megfelelő. A zenelejátszó funkció is csak audió fájlokkal működik, ezért nem megfelelő formátum esetén nem játszik le semmit.

3.4.6. Műfaj meghatározás túl rövid hangminták esetén

Az előfeldolgozás során 3 másodperces szegmensekre osztódik mindegyik adathalmazban szereplő hangminta és ezeken végződik el a transzformáció, ami során mel-frekvencia cepstrum együtthatókká (MFCC) alakul. A konvolúciós háló tanulási folyamatához bemenetnek, ezeknet a 3 másodperces MFCC értékeket kapja meg. A bemenetnek formailag (130,13,1) nagyságú tömböket ad vissza, ahol 130 az MFCC vektorok száma szegmensenként, 13 együttható az MFCC-n belül és az 1-es érték a kernel mélysége. Mivel spektogramok és MFCC-k megjeleníthetőek egy színcsatorna különböző árnyalataival, ezért elegendő a kernel mélységet 1-nek megadni. Túl rövid 3 másodpercnél rövidebb hangfájlok esetén a program nem tudja a hálózat által elvárt formára hozni a bemenetet, ezért ilyenkor a fájl betöltődik, lejátszható és metaadatai megjelenítésre kerülnek, ha található a fájlban, viszont a műfaj meghatározás esetén egy új hangfájl betöltését kéri a program.

3.4.7. Hiányzó metaadatok

Sok hangfájlnál fordul elő, hogy egyes metaadat címkéi hiányoznak. Ilyenkor a program csak azokat a címkéket jeleníti meg, amiket megtalált a zene metaadatában és a többi mezőt üresen hagyja. Ehhez hasonlóan működik egy zene borítóképének megjelenítése is. Borítóképeknél viszont arra is figyelnie kell a programnak, hogy a különböző fájlformátumok metaadatában, hogy található meg az adott kép, mivel ezek másképp találhatóak meg egy WAV, FLAC és MP3 fájlban. Ha a program nem talál zene borítóval kapcsolatos információt, akkor a program indulásakor megjelenő borítót helyettesítő kép kerül megjelenítésre. A metaadatok hiánya a műfaj meghatározás pontosságát nem befolyásolja.

3.5. Továbbfejlesztési lehetőségek

A bemutatott programokat több téren is tovább lehetne fejleszteni. Első sorban az adathalmaz bővítésével, több műfaj, hangminta hozzáadásával egy sokkal pontosabb műfaj osztályozást kaphatna a felhasználó. A neurális hálózati modell szerkezetének módosításával, bővítésével is lehetne pontosítani a végső meghatározáson. Viszont fontos megjegyezni, hogy a tanulási folyamat fejlesztéséhez egy erősebb hardverrel rendelkező gép szükséges, mivel eléggé időigényes folyamat lehet lassabb gépeken. Érdemes a tensorflow grafikus kártyát (GPU-t) használó változatát használni, ha a fejlesztő grafikus kártyája támogatott ilyen feladatok elvégzésére. A műfaj meghatározó programot is lehetne bővíteni több hang anyagok elemzését segítő plot megjelenítésével, több neurális hálózat modellt használó hangszűrést, zene felismerést végző funkciókkal, akár a programot több eszközön, platformon is elérhetővé lehetne tenni, mint például okostelefonokon és weboldalon.

4. fejezet

Összegzés

A bemutatott programok jól szemléltetik, hogy neurális hálózatok felhasználásával zenei területen is számos problémára nyújthatnak megoldást. Mivel évről évre új műfajok jelennek meg, sok esetben két létező műfaj keveréke, ezért gyakran nehéz őket egy adott műfajba bekategorizálni. Ilyen esetekre nagy segítséget nyújthat a neurális hálózatokkal történő műfaj meghatározás, ahol a kiválasztott hangfájlt kisebb szegmensekre osztja és mindegyik szakaszra meghatároz egy műfajt. Amelyik műfajt a leggyakrabban felismeri az egész hangfájl során, akkor a modell azt tekinti a kiválasztott zene műfajának. Ahogy az emberi fül is hallás alapján tud egy hangot pontosan behatárolni, úgy a neurális hálózatnak is az emberi halláshoz hasonló formában kell továbbadni a meghatározni kívánt zenét. Ehhez nyújtanak nagy segítséget a hangfájlok különböző ábrázolásai, mint például a spektogramok, vagy a programom hangfájlok feldolgozásánál használt mel-frekvencia együtthatók. Az ilyen ábrázolásoknál nyújt nagy segítséget a Fourier transzformáció és annak különböző típusai. Mivel ezek az ábrák képként ábrázolhatóak, ezért a tanulási folyamatnál és műfaj meghatározásnál, gyakran képfeldolgozással kapcsolatos feladatoknál használt konvolúciós neurális hálózat is alkalmas műfaj szerinti osztályozásra. Természetesen a felhasznált transzformációk és neurális hálózat típus nem csak műfaj meghatározás területén lehetnek hasznosak, hanem bármilyen más hangszín alapján történő kategorizáló vagy szűrő problémára is megoldást nyújthatnak.

Köszönyetnyilvánítás

Szeretném megköszönni mindazoknak, akik támogattak a szakdolgozatom elkészítése során. Köszönettel tartozom témavezetőmnek, Dr. Szekeres Béla Jánosnak, aki egyetemi éveim során kitartó munkával és nagy szakértelemmel megtanította az analízis és gépi tanulás tudományát, valamint lehetővé tette, hogy szakdolgozatom témáján dolgozhassak. Továbbá szeretném megköszönni mindenkinek, aki az évek során az egyetemen oktatott és hasznos tudást adott át. Biztos vagyok benne, hogy az egyetemen tanult tudásnak hasznát fogom venni. Végül szeretném megköszönni családomnak és barátaimnak, akik támogattak és biztattak. Mégegyszer szeretném megköszönni mindenkinek, aki segített szakdolgozatom megírásában.

Tisztelettel,

Müller Alex Tibor

Forrásjegyzék

[1] Andrada. *GTZAN Dataset - Music Genre Classification*. https://www.kaggle.com/datasets/andradaolteanu/gtzan-dataset-music-genre-classification/versions/1. 2020.

[2] Simon Whibley és tsai. „WAV Format Preservation Assessment". *British Library: London, UK* (2016).

[3] Luthfi Firmansah és Erwin Budi Setiawan. „Data audio compression lossless FLAC format to lossy audio MP3 format with Huffman Shift Coding algorithm". *2016 4th International Conference on Information and Communication Technology (ICoICT)*. 2016, 1–5. old. DOI: 10.1109/ICoICT.2016.7571951.

[4] Mengyu Qiao, Andrew H Sung és Qingzhong Liu. „Revealing real quality of double compressed MP3 audio". *Proceedings of the 18th ACM international conference on Multimedia*. 2010, 1011–1014. old.

[5] J. Nathan Kutz Steven L. Brunton, szerk. *Data Driven Science Engineering, Machine Learning, Dynamical Systems, and Control*. Cambridge University Press, 2019. ISBN: 9781108422093.

[6] James A. Moorer. „A note on the implementation of audio processing by short-term fourier transform". (2017), 156–159. old. DOI: 10.1109/WASPAA.2017.8170014.

[7] Valerio Velardo musikalkemist. *DeepLearningForAudioWithPython*. https://github.com/musikalkemist/DeepLearningForAudioWithPython.git. 2020.

[8] Chris M. Bishop. „Neural networks and their applications". *Rev. Sci. Instrum.* 65.6 (1994), 1803–1832. old. URL: https://web.archive.org/web/20060903153555id_/http://www.stat.purdue.edu/~zdaye/Readings/Neural_Networks_and_Their_Applications.pdf.

[9] BruceBlaus Wikimedia Commons. *Multipolar Neuron*. 2013. URL: https://commons.wikimedia.org/wiki/File:Blausen_0657_MultipolarNeuron.png.

[10] Keiron O'Shea és Ryan Nash. „An Introduction to Convolutional Neural Networks". (2015). arXiv: 1511.08458 [cs.NE].

[11] Andreas Maier Wikimedia Commons. *ConvolutionAndPooling*. 2019. URL: https://commons.wikimedia.org/wiki/File:ConvolutionAndPooling.svg.

Ábrák jegyzéke

Táblázatok jegyzéke

Forráskódjegyzék