Eötvös Loránd Tudományegyetem

Informatikai Kar

Inormációs Rendszerek Tanszék

Automatikus zenei hangszerfelismerés többszólamú zenében mély neuronhálók segítségével

Témavezető:
Gombos Gergő
Adjunktus, PhD

Szerző:
Hamrák János
programtervező informatikus MSc

Budapest, 2020

EÖTVÖS LORÁND TUDOMÁNYEGYETEM

INFORMATIKAI KAR

DIPLOMAMUNKA TÉMABEJELENTŐ

Hallgató adatai:

Név: Hamrák János

Neptun kód: KLGCQ1

Képzési adatok:

Szak: programtervező informatikus, mesterképzés (MA/MSc)

Tagozat: Nappali

Belső témavezetővel rendelkezem

Témavezető neve: *Gombos Gergő*

munkahelyének neve: Eötvös Loránd Tudományegyetem, Információs Rendszerek Tanszék

munkahelyének címe: 1117 Budapest, Pázmány Péter sétány 1/C.

beosztás és iskolai végzettsége: Adjunktus, PhD

A diplomamunka címe: *Automatikus zenei hangszerfelismerés többszólamú zenében mély neuronhálók segítségével*

A diplomamunka témája:

(A témavezetővel konzultálva adja meg 1/2 - 1 oldal terjedelemben diplomamunka témájának leírását)

Diplomamunkám témája az automatikus zenei hangszerfelismerés többszólamú zenében mély neuronhálók segítségével. Egy alkalmas modell segítségével zenék sokaságát tudjuk felannotálni megfelelő címkékkel. Ezek pedig kereső- és ajánlórendszerek alapját képzik. Fontosnak tartom kiemelni, hogy polifonikus (többszólamú) környezetben foglalkozom a problémával. Korunk zenéjének túlnyomó része rendelkezik ezzel a tulajdonsággal és szeretnék minél jobban az életszerűségre törekedni. A megvalósítás érdekében első lépésként a zene területéhez kapcsolódó elérhető adathalmazok feltérképezésével, összevetésével foglalkozom. Az általam legalkalmasabbnak talált adathalmaz fog szolgálni munkám inputjaként. Főként az ezen adathalmazra épített hagyományos gépi tanulási, illetve mély neuronhálós modellek elemzése nyújt kiindulási pontot saját modellem elkészítéséhez. Diplomamunkám fő célja egy mély neuronhálós modell tervezése és fejlesztése. Ez a modell egy multi-class, multi-label osztályozást fog megvalósítani. Ez azt jelenti, hogy több osztályba (hangszerek) sorolhatjuk az inputokat (zenék), de emellett egy inputhoz (zene) tartozhat több címke (hangszer) is. A modell fejlesztése python nyelven történik, kiegészítve a mély neuronhálós modellek építésére szolgáló Keras magas szintű API-val.

Budapest, 2019.11.20.

Tartalomjegyzék

1. fejezet

Bevezetés

Napjainkban a zenéhez legkönnyebben digitális formában, a világhálón keresztül férhetünk hozzá. Néhány kattintással olyan zenei tartalomszolgáltatókat érhetünk el, melyek széleskörű adatbázissal rendelkeznek. A folyamatosan bővülő adatmennyiség ellenére ezeknek az adatbázisoknak átláthatónak és könnyen kezelhetőnek kell maradniuk, hogy a felhasználókat a kívánt módon tudják kiszolgálni. Ennek érdekében nap mint nap új megoldások születnek zenei információk automatikus kinyerése és feldolgozása céljából. Ezek teszik lehetővé a digitálisan tárolt zenék körében például az osztályozást vagy keresést.

A zenei információk kinyerésének tudományába (Music Information Retrieval, a továbbiakban: "MIR") tartozik a továbbiakban taglalt probléma, az automatikus hangszerfelismerés. Ez egy osztályozási feladat. Célja, hogy a meglévő digitális hanganyag alapján az adott zenéről megállapítsuk, hogy milyen hangszerek szólalnak meg benne. Ezt az információt több célra is fel tudjuk használni, például:

- Későbbi feldolgozásra, további MIR feladatok inputjaként
- Statisztikák készítésére
- Adatbázisban való keresés szűrőfeltételeként
- Egy ajánlórendszer részeként, ahol az aktuális zeneszámot követően például egy hangszerelésében hasonló számot szeretnék ajánlani a felhasználónak.

Az automatikus hangszerfelismerés feladatot több aspektusból lehet megközelíteni, például a bemeneti adatok jellege, reprezentációja, a megvalósított architektúra,

az osztályozás módszere, vagy az osztályok száma alapján. A dolgozatom keretein belül felkutatok néhány létező megoldást, majd ezekre alapozva prezentálom saját megközelítésemet és ennek eredményeit.

Az általam bemutatott megoldás egy multi-label osztályozást valósít meg mély neuronhálós rendszer segítségével többszólamú zenében. Pontosabban egy CNN architektúrát alkalmaztam, amelynek inputként polifónikus zenei reprezentációkat adtam meg. Minden zenéhez rendelkezésre álltak címkék arra vonatkozólag, hogy milyen hangszerek szólalnak meg benne. Outputként adott zenére hangszerenként egy bináris értéket vártam, amely arra utal, hogy az adott hangszer megszólal-e az adott zenében. Adott zenére ezen bináris értékekből álló vektor adja meg a zenét játszó hangszerek összességét.

1.1. Motiváció

Az ember kognitív képességei segítségével a zenében könnyedén fel tudja ismerni az egyes hangszereket. Ugyanez a feladat a számítógép számára azonban már sokkal kevésbé triviális. Ennek egyik oka, hogy egy hangszer megszólaltatásának digitális reprezentációja nagyon változatos lehet. Függ például a hangszíntől, hangmagasságtól, hangerőtől és előadásmódtól, de a felvétel minőségétől és az esetleges háttérzajtól is. További nehezítő körülmény a többszólamúság, amikor egy időben több hangszert is megszólaltatunk, ezzel összemosva az egyszólamú környezetben is sokváltozós képünket.

A MIR nagyban támaszkodik a mesterséges intelligenciára. A számítógépek számítási kapacitásának folyamatos növekedése és az elérhető adathalmazok gyarapodása által pedig egyre nagyobb figyelmet kap a mesterséges intelligencia egy kiemelten számításigényes részterülete: a mély tanulás. Ezt bizonyítja, hogy az évente megrendezésre kerülő ISMIR (International Society for Music Information Retrieval) konferencián 2010-ben még csak 2 ([1], [2]) mély tanulással kapcsolatos cikk jelent meg, de 2015-ben már 6, 2016-ban pedig már 16. [3]

A mély tanulás tehát egy ígéretes módszer lehet a MIR problémák megoldásában, ideértve az automatikus hangszerfelismerést is. Ezt kihasználva és megoldás életszerűségére törekedve döntöttem úgy, hogy elkezdek kísérletezni mély neuronhálókkal

többszólamú zenében. Célom volt találni egy tanításra alkalmas adathalmazt, azon pedig tervezni egy olyan mély neuronhálós rendszert, amely a jelenlegi megoldások pontosságát meghaladja.

1.2. A dolgozat felépítése

Dolgozatomban tehát az eddigi kapcsolódó kutatásokat, illetve saját munkám eredményét dolgozom fel. A következő alfejezetben felsorolom az általam relevánsnak tartott, a State-of-the-Arthoz vezető kutatásokat.

A második fejezetben betekintést adok a téma elméleti hátterébe. Először kifejtem a zenével kapcsolatos főbb fogalmakat, bemutatom fontosabb tulajdonságait, reprezentációit. Kitérek a MIR bemutatására is. Ezután bevezetem a gépi tanulás és a mély tanulás fogalmát.

A harmadik fejezet az adathalmazokról fog szólni. Itt előbb felsorolom az adathalmazok kiválasztásának szempontjait, majd minden felhasznált adathalmaznak ismertetem a főbb jellemzőit.

A negyedik fejezetben a módszertanról ejtek szót. Itt kifejtésre kerülnek az adatok előfeldolgozási módszerei, az általam bemutaott mély tanulási architektúrák, illetve ezek megvalósításai.

Az ötödik fejezetben részletezem az általam végzett kisérleteket és ezek eredményeit. Ezeket összevetem egymással, illetve a releváns State-of-the-Art kutatásokkal.

A hatodik fejezetben összegzem a leírtakat, valamint továbbgondolom a kutatásomat, felvázolok néhány ötletet annak jövőjéről.

1.3. Kapcsolódó munkák

Az automatikus hangszerfelismerés témában a korábbi kutatások túlnyomó része a monofónikus, azaz egyhangszeres zenékkel foglalkozik. Martin és Kim [4] mintafelismerési statisztikai technikája 1023 izolált hangjegy és 15 különböző hangszer között a hangszercsaládok felismerésében 90%-os, egyéni hangszerek felismerésében pedig 70%-os pontosságot produkált. Brown [5] a kepsztrális együtthatókat használta fel K-közép klaszterezési módszeréhez. Eronen és Klapuri [6] széleskörű, spektrális és

időbeli feature-halmaz segítségével - összesen 43 különböző feature felhasználásával - 81%-os hangszer és 95%-os hangszercsalád pontosságról számolt be. Deng [7] klasszikus zenei hangszerek tekintetében elemezte a különböző, gépi tanulási módszerekben használatos feature összeállításokat. Bhalke [8] tört Fourier-transzformáción alapuló MFCC feature-ök segítségével tervezett CPNN osztályozót mutatott be, amellyel hangszercsaládok tekintetében 96.5%-os, hangszerek tekintetében pedig 91.84%-os pontosságot ért el.

Többszólamú környezetbe való átültetéssel foglalkozott Burred tanulmánya [9], aki a többszólamúságot két kísérlettel közelítette meg. Először csak egy-egy hangjegyet kombinált össze többszólamú hangjeggyé. Itt két szimultán hangjegy esetén 73.15%-os, három hangjegyre 55.56%-os, négy hangjegy kombinációjára pedig 55.18%-os pontosságot sikerült elérni. Másik kísérletként hosszabb szekvenciákat kombináltak össze, ekkor két hang esetén 63.68%-os, három hang esetén pedig 56.40%-os pontosságot kaptak.

Eggink és Brown [10] a polifónikus zenékben a hiányzó adat elméletükkel próbálták feltárni az egyes hangszereket. Ennek lényege, hogy felderítették azon idő- és frekvenciabeli részeket a zenén belül, ahol szeparáltan egy hangszer tulajdonságait vélték felfedezni és ezt dolgozták fel. Erre a módszerre épített Giannoulis és Klapuri [11] kutatása is, és hasonló megközelítést alkalmazott Garcia [12] is.

Jiang [13] egy többlépcsős megoldást mutat be. Első lépésben a hangszercsaládot határozták meg, ezzel szűkítve a a lehetséges hangszerek halmazát és a változók számát. A pontos hangszer-meghatározás csak ezután következett.

Az előbbi kutatások többnyire hagyományos gépi tanulási megoldásokat alkalmaztak, amelyekhez maguk nyerték ki a különböző bemeneti feature-öket. Humphrey [14] írásában a mély tanulási architektúrákat ismerteti a MIR terület korszerű irányzataként. A témában gyakorlati segítségként szolgál Choi [15] írása, amiben konkrét adatreprezentációkat, mély neuronhálós rétegeket, és mély tanulási technikákat mutat be.

Li [16] a nyers hanganyagot inputként felhasználva egy konvolúciós mély neuronhálós rendszert mutatott be a polifónikus zenében való automatikus hangszerfelismerés kapcsán. Ezt a megoldást aztán összevetette hagyományos gépi tanulási módszerekkel is. A mély neuronhálós rendszer teljesített legjobban. 75.60%-os pon-

tossággal, 68.88%-os felidézéssel, 72.08 mikro F értékkel és 64.33 makro F értékkel. Han [17] szintén egy mély konvolúciós hálót használt, azonban az osztályozás szempontjából máshogyan járt el: a zenékben egy darab domináns hangszert keresett. Bemenetként a zenék spektogramját használta fel, 0.602-es mikro és 0.503-as makro F értéket ért el.

2. fejezet

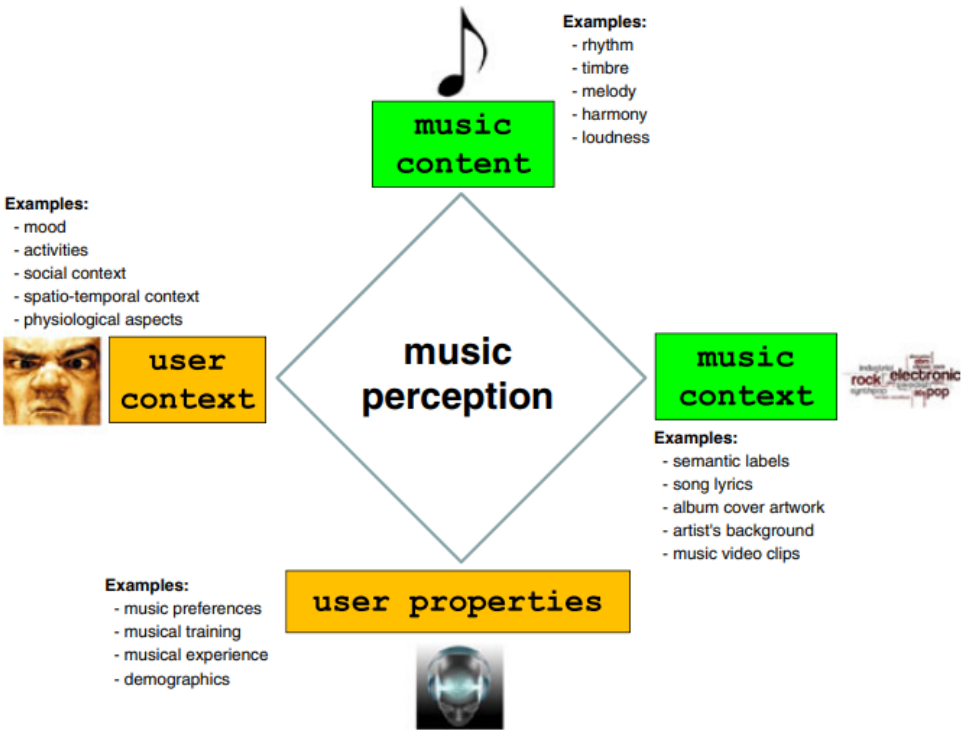
Elméleti háttér

Ebben a fejezetben a dolgozathoz kapcsolódó fogalmakat és elméleti alapokat mutatom be. Először magának a zenének a releváns tulajdonságairól ejtek szót. Ezután ismertetem a MIR kutatási területet, amelybe dolgozatom is tartozik. Majd végül a mesterséges intelligencián alapuló megoldásokról nyújtok elméleti bevezetőt, érintve a hagyományos gépi tanulás és a mély tanulás módszereit is.

2.1. Zene és reprezentációi

2.1.1. Zene fogalma, tulajdonságai

A zene egy meglehetősen összetett fogalom. Az ember számára a zene megjelenhet például hang formájában, leírhatjuk őket szimbólumok segítségével egy kottában, előfordulhat szöveges formában dalszövegként, képi formában egy albumborító, vagy egy zenész képében, illetve mozdulatokban egy zenei előadás keretében. A teljes zenei élményt ezek kombinációja nyújtja. A zene észlelését befolyásoló tényezőket Schedl [18] a következő kategóriákba sorolta: a zene tartalma (music content), a zene kontextusa (music context), a hallgató kontextusa (user context) és a hallgató tulajdonságai (user properties). [19]



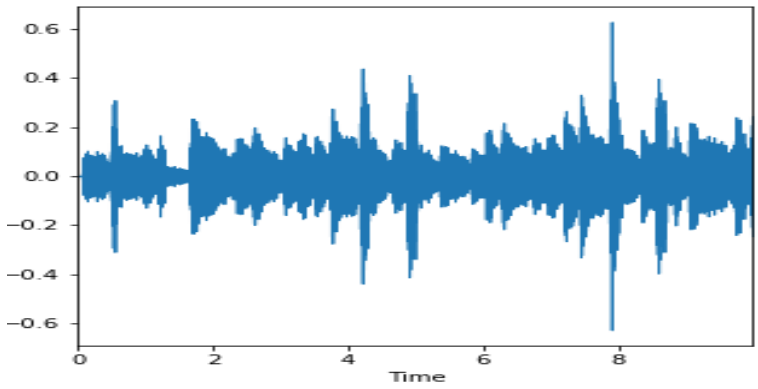
2.1. ábra. A zene észlelését meghatározó tényezők, forrás: [18]

A zenei tartalom fogalma utal azokra a tulajdonságokra, melyek a hangok fizikai jelként való leírása definiál. Ilyen például a ritmus, a hangszín, a dallam, a harmónia, a hangerő, vagy a dalszöveg. Ezzel szemben a zene kontextusa alatt azokat a tényezőket értjük, melyeket nem tudunk közvetlenül a zenéből kinyerni, mégis szorosan kapcsolódik hozzá. Ide tartozik például az előadó hírneve, az albumborító, a művész kulturális- vagy politikai háttértörténete, vagy a zeneéhez készített videoklip. Ami a hallgatóval kapcsolatos aspektusokat illeti, a hallgató kontextusa alatt értjük a dinamikus, gyorsan változó tényezőket. Ide sorolható a hallgató aktuális hangulata, tevékenysége, társadalmi helyzete, tér- és időbeli helye, illetve pszichológiai állapota. Ezzel ellentétben a hallgató tulajdonságai az állandó, vagy csak lassan változó jellemzőit takarja. Ilyen az egyén zenei ízlése, zeneelméleti képzettsége, demográfiai adatai, a hallgatott előadóval kapcsolatos véleménye, vagy a barátai zenei ízlése, véleménye. [19]

Dolgozatomban az észlelést meghatározó tulajdonságok közül a zenei tartalmat fogom felhasználni a hangszerek kinyerése érdekében. A zenei tartalom számítógépes felhasználásához pedig elengedhetetlen, hogy a hangokat megfelelően tudjuk számítógépen ábrázolni.

2.1.2. Hang reprezentációk

A hangok fizikai mivoltukban rezgésekként jelennek meg. A rezgéseket matematikailag olyan folytonos függvényekkel tudjuk leírni, melyek értelmezési tartománya az idő, értékkészlete pedig a nyugalmi állapothoz viszonyított pillanatnyi kitérés. Ilyen lehet például egy szinuszgörbe. Ahhoz, hogy a hangokat számítógépen tudjuk tárolni és feldolgozni, ezeket a függvényeket kell ábrázolnunk. Mivel azonban a számítógép számábrázolása véges, ezért a hangokat először digitalizálni kell. Ez azt jelenti, hogy a folytonos függvényeket diszkrét, azaz véges helyen vett és véges értékekkel rendelkező fügvényekké alakítjuk. Ez úgy történik, hogy ez eredeti függvényünkből megadott időközönként mintát veszünk, diszkrét értékre kerekítjük, és ezeket az értékeket összefűzzük. Az így kapott függvény lesz a hangnak az idő függvényében ábrázolt digitális reprezentációja.

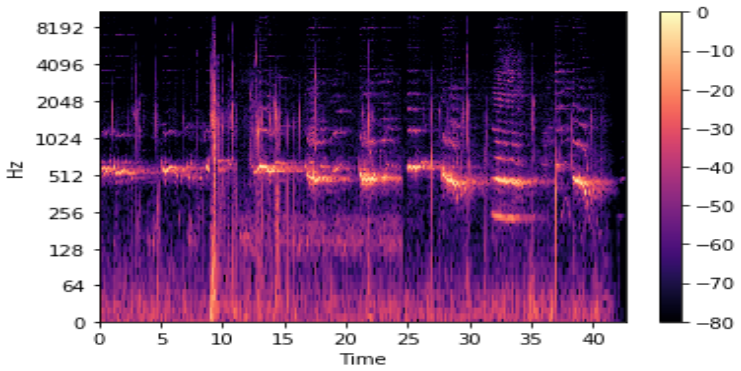


2.2. ábra. Tíz másodperces hanganyag hanghullám reprezentációja

A hangok idő függvényében ábrázolt digitális reprezentációja tehát egydimenziós, mivel egy függvénygörbének tekinthetjük. Ezt szokták hívni hullámforma reprezentációnak, illetve nyers hangnak (raw audio) is, ugyanis további reprezentációkká tudjuk transzformálni. A MIR területen megjelenő mély tanulási megoldások jelentős része ezen nyers hangábrázolás helyett inkább a kétdimenziós reprezentációkat alkalmazza bemenetként. Ezt azzal indokolják, hogy a nyers bemeneten való tanítás sikeréhez nagyobb adathalmaz szükséges, mint a kétdimenziós reprezentációkéhoz. [15]

Az említett kétdimenziós reprezentációk Fourier-transzformáción alapszanak. A Fourier-transzformáció nagyon leegyszerűsítve egy olyan függvény, amely bemenetként kap egy idő függvényében ábrázolt jelet és ezt felbontja frekvenciákra [20].

- **Ablakozott Fourier transzformált (STFT)**: Gábor Dénes magyar fizikus nevéhez kötődik. A bemeneti jelet egyenlő méretű időszeletekre (ablakokra) osztjuk, majd ezeken alkalmazzuk a Fourier-transzformációt. Ezáltal kapjuk meg a hang spektrumát (spectogram). [20], [15]



2.3. ábra. Spektrum [21]

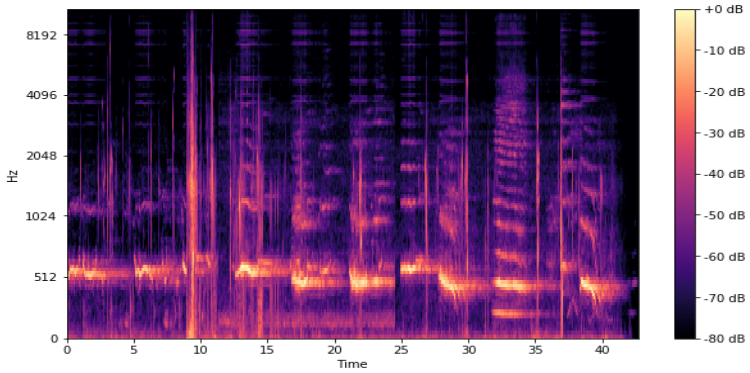
- **Mel-frekvenciára transzformált spektrum (Mel-spectogram)**: A mel-spectogram az előbb említett spektrum Mel skálára való transzformáltja. A Mel skálát egy nemlineáris függvény segítségével kapjuk a frekvenciaskálából. Célja, hogy a skála jobban reprezentálja az ember hallásának tartományait. Tehát az értékek közti különbség a Mel skálán megfeleltethető legyen annak, hogy az ember mennyire különböző magasságúnak hallja ezeket. A frekvencia skálán például sokkal nagyobbnak érezzük az 500Hz és 1000Hz közti különbséget, mint a 7500Hz és 8000Hz közti különbséget. A mel-frekvenicára való konverzió képlete a következő:

$$Mel(f) = 2595 * ln(1 + \frac{f}{700}) \tag{2.1}$$

Ahol Mel(f) az adott frekvenciaérték mel-skálán való értéke, f pedig az adott frekvencia érték. A képlet segítségével egymást átfedő frekvencia sávokat alakítunk ki, melyek a mel-skálát tekintve egyenlő távolságra helyezkednek el egymástól, majd a frekvenciasávok energia értékeit egyenként leképzzük mel-skálára. [20], [15]

Léteznek egyéb, hasonló skálák is, mint pl. Bark skála, vagy az hallás pszichológián alapúló ERB. Ezek MIR környezetben még nem kerültek összevetésre,

de beszédfelismerés környezetben nem mutatnak szignifikáns eltérést az egyes skálák eredményei. A Mel skála, illetve a mel-spectogram használata azonban egy gyakran használt és jól felhasználható reprezentációnak bizonyul MIR környezetben. [15]

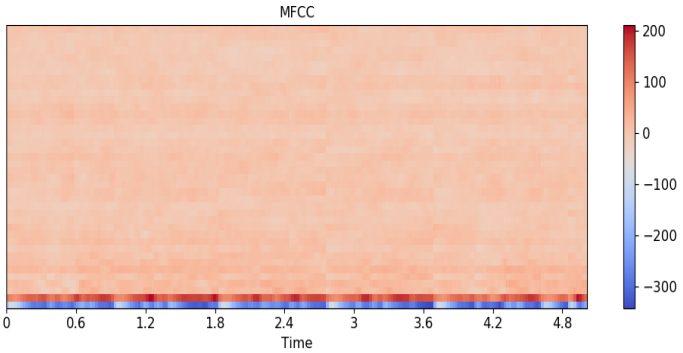


2.4. ábra. Melspectogram reprezentáció [21]

- **Mel-frekvencián vett kepsztrum együtthatók (MFCC)**: Az MFCC együtthatókat a melspectogramon való diszkrét koszinusz-transzformációval kapjuk meg. A képlet a következő:

$$C_i(m) = \sum_{n=0}^{M-1} S_i(n) cos[\pi m \frac{n - 0.5}{M}], 0 \leq m \leq M \tag{2.2}$$

Ahol M a frekvenciaszeletek száma, $S_i(n)$ az egyes sávokban kiolvasott energia értékek és $C_i(m)$ az i-edik MFCC együttható. Ezzel a mel-spectogramnál tömörebb reprezentációt kapunk. Ennek hátránya lehet az információvesztés, előnye pedig az apró zajok kiszűrése. [8]



2.5. ábra. MFCC reprezentáció [21]

2.2. Music Information Retrieval (MIR)

A bevezetőben már említettem a zenei információk kinyerését (music information retrieval - MIR). Ez egy interdiszciplináris kutatási terület, magában hordozza többek között a zeneelmélet, pszichoakusztika, pszichológia, informatika, jelfeldolgozás és gépi tanulás tudományágakat. Céljára jól utal az elnevezése, zenékből szeretnénk releváns információt kinyerni, és ezeket felhasználni [15]. A felhasználásra szerintem nagyon jó, életszerű példát ad Downie 2003-as cikkének [22] bevezetője, amelyet a következőképp fordíthatunk le:

"Képzeljünk el egy világot, ahol egyszerűen felénekelhetjük egy számítógépnek a dalrészletet, ami már reggeli óta a fejünkben jár. A gép elfogadja a hamis énekünket, kijavítja, és azonnal javaslatot tesz arra, hogy éppen a "Camptown Races" című számra gondoltunk. Miután mi belehallgatunk a gép által talált számos relevánsnak tartott MP3 fájl egyikébe, elfogadhatjuk a gép javaslatát. Ezután elégedetten elutasíthatjuk a felajánlást, hogy az összes további létező verzióját is felkutassa a dalnak, ide értve a nemrég megjelent olasz rap verziót, vagy a skótdudás duettre írt kottát." [22]

Figyeljük meg, hogy ez a hétköznapi eset mennyire össztett probléma. A következő feladatok jelennek meg:

- Az emberi éneklés, vagy dúdolás alapján hangfelismerés.
- Hang alapú lekérdezés egy zenei adatbázisban az előbbi bemenettel.
- Hangelemzés, feldolgozás, hogy a hamis hangokat ki tudjuk javítani, az esetleges háttérzajokat eltávolítsuk, illetve ha kell, a dallamból automatikusan kottát generáljunk.
- Hasonlóságon alapuló keresés zenék között, hogy megtaláljuk a kívánt dalt az adatbázisban.
- Zenei feldolgozások detektálása, hogy további verzióit is megtaláljuk egy adott dalnak.

MIR problémák definiálását több szempontból közelíthetjük meg. Choi cikke [15] két tengelyre osztja fel a problémateret: szubjektivitás és eldöntési időmérték. A

szubjektivitás tengelyen léteznek szubjektívebb feladatok, melyekre nincsenek egyértelmű válaszok. Ilyen lehet például a zene műfajának meghatározása. Objektívebb feladatoknak tekinthetjük azokat, melyek eredménye egyértelműen meghatárhozható, esetleg számszerűsíthető. Ide tartozik a hangszerfelismerés, vagy a tempó észlelés. [15]

A másik tengely, az eldöntési időmérték aszerint sorolja be a feladatokat, hogy mekkora időegységeken értelmezhető egy becslés. Ez egy relatív mérték. Például a dallamfelismerés eldöntési időmértékére azt mondhatjuk, hogy alacsony, mert egy felismert dallam jó eséllyel nem fedi le az egész zenét. Másik kifejezéssel azt mondhatjuk, hogy ez egy időben változó, azaz dinamikus tulajdonság. Ellenben a tempó általában állandó értékű az egész zenében, így a teljes zeneszámot fel tudjuk címkézni egy adott tempóval. Erre azt mondjuk, hogy eldöntési időmértéke relatív magas, azaz ez egy statikus tulajdonsága a zenének.[15]

A hangszerfelismerést tekinthetjük statikus, illetve dinamikus feladatnak is a probléma megközelítésének függvényében. Dinamikus, ha erős címkézést szeretnénk megvalósítani, tehát arra vagyunk kíváncsiak, hogy adott időpillanatban éppen milyen hangszerek szólalnak meg. Gyenge címkézés esetén viszont a feladat statikussá válik. Ebben az esetben az egész zenére vetítve szeretnénk címkéket kapni egyes hangszerek jelenlétével, vagy a többi hangszerrel szembeni dominanciájával kapcsolatban.

A MIR terület főbb részterületei és feladatai Schedl [19] cikkére alapozva a következők:

- Jellemző kinyerés (feature extraction)
  - hangszín leírás pl. [23], [24]
  - kotta- és dallamkinyerés pl. [25], [26], [27]
  - ütem lekövetés, tempó becslés pl. [28], [29], [30]
  - tonalitás becslés pl. [31], [32], [33], [34], [35], [36]
  - struktúrális analízis, szegmentáció pl. [37], [38], [39]
- Hasonlóságon alapuló feladatok
  - hasonlóság mérés pl. [40], [41], [42]

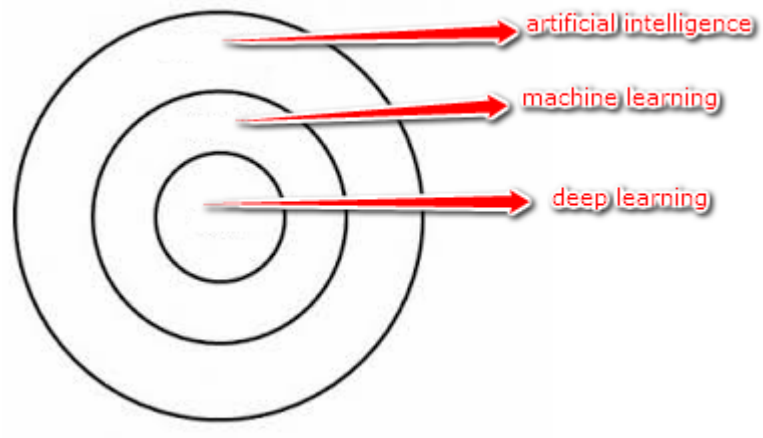
  - zenei feldolgozás felismerés pl. [43], [44]
  - dúdoláson alapuló lekérdezés pl. [45], [46], [47]

- Osztályozási feladatok
  - érzelem- és hangulatfelismerés pl. [48], [49]
  - műfaj szerinti osztályozás pl. [50], [51]
  - hangszerfelismerés pl. [52]
  - szerző / előadó / énekes felismerés pl. [53]
  - automatikus címkézés pl. [54], [55], [56]

- Alkalmazások
  - hanganyaghoz forrásazonosító készítés (fingerprinting) pl. [57], [58]
  - tartalom alapú lekérdezés pl. [59]
  - zene ajánlás pl. [60], [61], [62]
  - lejátszási lista generálás pl. [63], [64], [65], [66]
  - kottázás pl. [67], [68], [69]
  - dal/előadó sikeresség becslés pl. [70], [71], [72]
  - zene vizualizáció pl. [73], [74], [75], [76], [77]
  - felhasználói felületen való zenei böngészés pl. [78], [79], [80], [81], [82]
  - személyre szabott, alkalmazkodó rendszerek pl. [83], [84], [85], [86]

2.3. Mesterséges intelligencia

A mesterséges intelligencia egy általános fogalom az emberi gondolkodás számítógéppel való reprodukálására történő módszerekre. Ahogy arról a bevezetésben is szót ejtettem, a MIR tudományág gyakran használ mesterséges intelligencián alapuló megoldásokat. A továbbiakban két mesterséges intelligenciát megvalósító módszert mutatok be röviden: elsőként a gépi tanulást, majd a mély tanulást, amely a gépi tanulás egy ágazata és napjaink meghatározó trendje. [87]



2.6. ábra. Egyes fogalmak közti kapcsolat szemléltetve, forrás: [87]

2.3.1. Gépi tanulás (Machine Learning)

A gépi tanulás tehát a mesterséges intelligencia megvalósításának egy módszere. Lényege, hogy a bemeneti adatokon statisztikai alapú, optimalizáló algoritmusok segítségével releváns mintákat fedezzünk fel, illetve egy olyan modellt alkossunk, amivel előrejelzéseket tudunk tenni.

Tanulás fajtái

A machine learning technikákat megkülönböztethetjük egyrészt a tanítás módja alapján:

- **Felügyelt tanulás (Supervised learning)** - felügyelt tanulás esetén előre rendelkezésünkre állnak a kimeneti címkék, ezek segítségével tanítjuk a modellünket. Ide tartozik például a hangszerek felismerése előre adott címkék segítségével is.

- **Felügyelet nélküli tanulás (Unsupervised learning)** - felügyelet nélküli tanulásnál a modellünk hoz létre csoportokat vagy címkéket a bemeneti adatokpontok közti összefüggések alapján. Ilyen fürtök segíthetnek például egy webshop esetén bizonyos termékek kosárba tétele után hasonló termékeket ajánlani.

- **Megerősítő tanulás (Reinforcement learning)** - A megerősítő tanulás esetén az algoritmus kísérletezések eredményéből tanul. Az egyes műveletek

után visszajelzést kap, amely segít megállapítani, hogy a választott döntés helyes, semleges vagy helytelen volt-e. Például egy társasjáték bot. [88]

Feladatok jellege

A feladatok jellege alapján pedig a következőképp kategorizálhatjuk a machine learning megoldásokat:

- **Regresszió (Regression)** - regresszió alatt olyan függvényt értünk, amely bizonyos tényezők értéke alapján hivatott megjósolni egy folytonos értékű változót.
- **Osztályozás (Classification)** - az osztályozás hasonló, mint a regresszió, azonban itt a kimenet nem egy folytonos érték lesz, hanem egy kategória. A feladat itt tehát a kategóriák elhatárolása.
- **Klaszterezés (Clustering)** - a klaszterezés, vagy fürtözés esetén az adatpontok egymástól való távolsága alapján alakítunk ki egymással korreláló csoportokat. [89]

Algoritmusok

A hagyományos machine learning algoritmusok közé tartoznak az alábbiak:

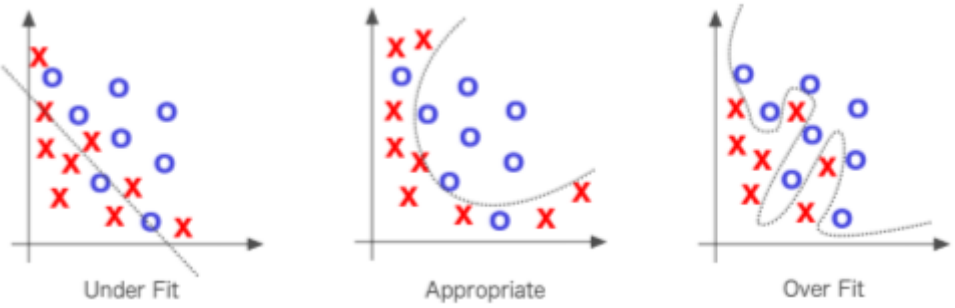
- **Lineáris regressziós algoritmusok (Linear regression)** - két változó közötti kapcsolatot mutatjuk meg azzal, hogy egy folytonos egyenes vonalat illesztünk az adatokra.
- **Logisztikus regressziós algoritmusok (Logistic regression)** - hasonló a lineáris regresszióhoz, azonban itt logisztikai görbét, azaz szigmoid függvényt illesztünk az adatpontokra.
- **Naív Bayes osztályozók (Classification)** - az esemény előfordulásának valószínűségét egy kapcsolódó esemény bekövetkezése alapján számoljuk ki.
- **Támogatási vektorgépek (Support-vector machines, vagy SVM)** - egy hipersíkot rajzolnak a két legközelebbi adatpont között. Ez marginalizálja az

osztályokat, és maximalizálja a közöttük lévő távolságot, hogy egyértelműbben meg lehessen különböztetni őket.

- **Döntési fák (Decision tree)** - az adatokat két, vagy több homogén halmazba osztják szét. Ha–akkor típusú szabályokat használnak arra, hogy az adatokat elkülönítsék az adatpontok közötti legjelentősebb különbségek alapján.
- **Véletlenszerű erdő (Random forest)** - döntési fákon alapulnak, de egy fa létrehozása helyett faerdőt hoznak létre, majd az erdőben lévő fákat véletlenszerűen rendezik el. Ezután összesítik a döntési fák különböző véletlenszerű formációinak szavazatait, és ez alapján határozzák meg a tesztobjektum végső osztályát.
- **K legközelebbi szomszéd (K nearest neighbors)** - minden adatpontot a hozzá legközelebb eső adatpontok alapján osztályozunk egy távolsági függvénnyel történő mérés alapján.
- **K közép fürtözés (K means clustering)** - K darab fürtbe csoportosítjuk az adatokat. Az egyes fürtökön belüli adatpontok homogének, más fürtök adatpontjaihoz képest pedig heterogének. [88]

Overfitting és underfitting

A gépi tanulás optimalizáló algoritmusának elsődleges célja, hogy modellünk megfelelő mértékben tudjon tanulni. A tanulás két elkerülendő, szélsőséges jelensége az alultanulás (underfitting) és a túltanulás (overfitting). Underfittingről beszélünk, amikor a modellünk nem tanulta meg a megfelelő mintákat, összefüggéseket az adatból. Az overfitting pedig ennek az ellentéte, amikor túlságosan az adatra specializálva tanítjuk a modellünket. Ennek eredményeként más adaton valószínűleg rosszul fog működni, nem fog tudni általánosítani.



2.7. ábra. Underfitting és overfitting fogalmak szemléltetve, forrás: [89]

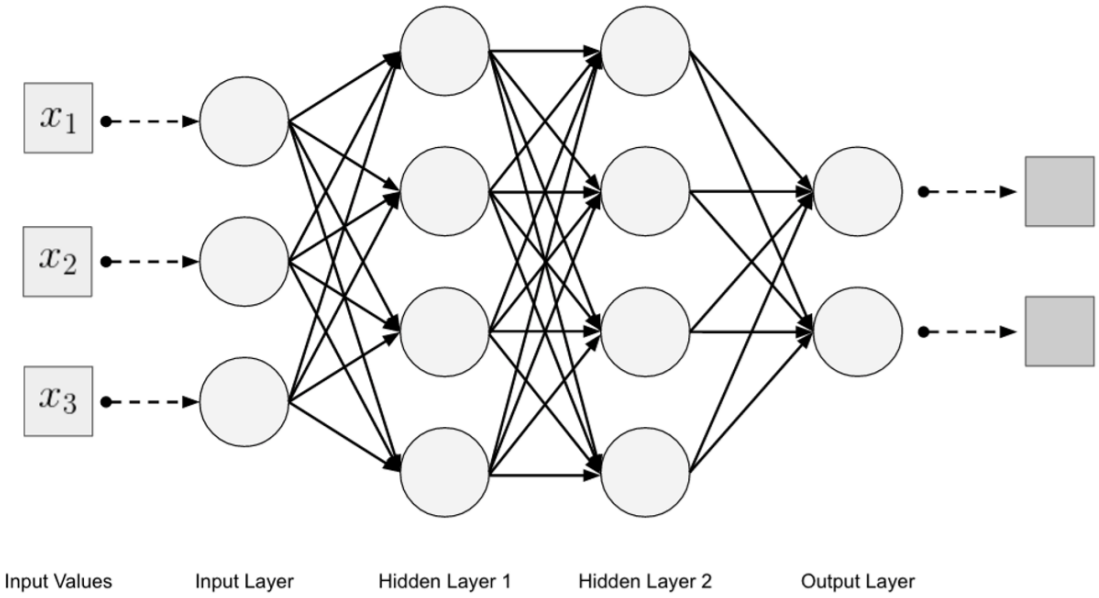
2.3.2. Mély tanulás (Deep Learning)

A mély tanulás a gépi tanulás napjainkban felkapott ágazata. Olyan gépi tanuló algoritmusokat értünk mély tanulás alatt, melynek rétegei a bemeneti adatok magasabb szintű absztrakcióinak kinyerésével hatékonyan képesek tetszőleges folyamatot modellezni. Ezek alatt a gyakorlatban olyan neurális hálózatokat értünk, melyek egy bemeneti, egy kimeneti és több rejtett rétegből állnak, illetve magas paraméterszámmal rendelkeznek. [90]

A hagyományos gépi tanulási modellekkel ellentétben a mély tanulási neurális hálók tanításához nem kell a bemeneten ismernünk minden releváns adatjellemzőt, mivel a magas szintű bemeneti adatból a hálózat saját maga fogja a rejtett rétegeken keresztül felismerni azokat. Ennek sikerességéhez viszont több bemeneti adattal és nagyobb számítási kapacitással kell rendelkeznünk, mint egy hagyományos machine learning modell esetén. [90]

Felépítés

A neurális hálók építőelemei a neuronok, melyek egy-egy skalár értéket reprezentálnak, így hordozzák az információt. Több neuron együttesen egy vektorként alkot egy réteget. Egy neurális háló egy bemeneti, egy kimeneti és több rejtett rétegből áll. A hálót alkotó rétegek száma határozza meg annak mélységét. [15]



2.8. ábra. Többrétegű előrecsatolt neurális hálózat felépítése, forrás: [89]

A bemeneti rétegen található az input adatunk (például egy kép), melynek egy-egy adatpontját (például egy pixelt a képen) egy-egy neuron fog ábrázolni. A bemeneti réteg neuronjai kapcsolatban állnak a következő, rejtett réteg neuronjaival. A kapcsolat típusa többféle lehet a neuronrétegek típusától függően. Ezekben a rejtett rétegekben ismeri fel a neuronhálónk a bemeneti adatból a számára releváns jellemzőket. A rejtett jelző onnan ered, hogy az itteni adat nem kerül nyilvánosságra, csupán a rákövetkező rétegnek fog bemenetként szolgálni. Végül a neurális háló utolsó rétege lesz a kimenet, itt kapjuk meg a modell előrejelzéseit az adott feladatra. [89]

Néhány példa különböző típusú neuronrétegekre:

- **Teljesen kapcsolt réteg (Fully-connected layer)** - ezen rétegek összes neuronja kapcsolatban áll az előző és a következő réteg összes neuronjával. Az rétegben található neuronok értékeinek és a kapcsolatok súlyának linerális kombinációját továbbítja a következő réteg neuronjai felé. Formalizálva:

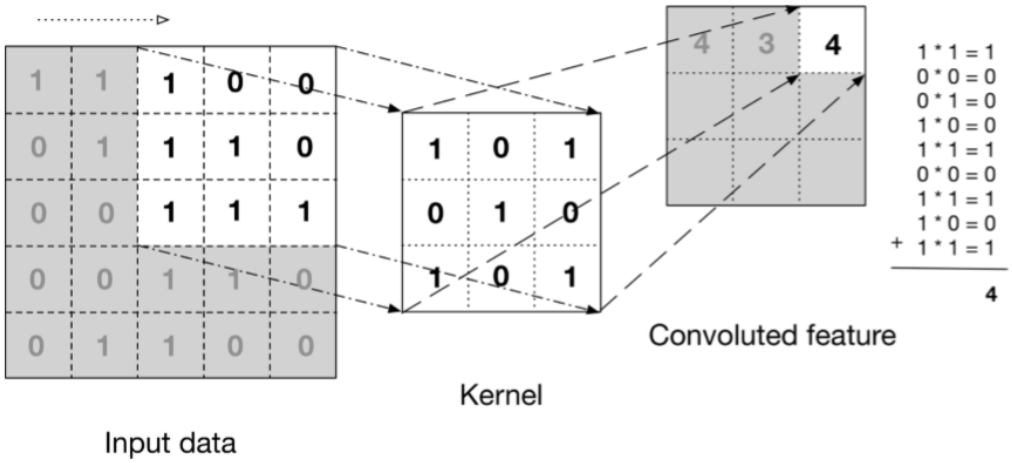
$$y = f(Wx + b) \tag{2.3}$$

, ahol x, b, y, W és $f()$ rendre a bemeneti neuron, az valós értéktől való eltérés értéke (bias), a kimeneti neuron, a súlymátrix és $f()$ egy nemlineáris aktivációs függvény.

- **Kovolúciós réteg (Convolution layer)** - lokális korrelációkat számol a neuronok és a konvolúció megadott magja (kernel) közt. Formalizálva:

$$y^j = f(\sum_{k=0}^{K-1} W^{jk} * x^k + b^j) \tag{2.4}$$

, ahol y^j, W^{jk}, x^k és b^j rendre kétdimenziósak. y^j a j-edik csatorna kimenete, x^k a k-adik csatorna bemenete, $*$ a konvolúciós operátor, W^{jk} a konvolúció magja a k-adik bemenethez és j-edik kimenethez és b^j a valós értéktől való eltérés.



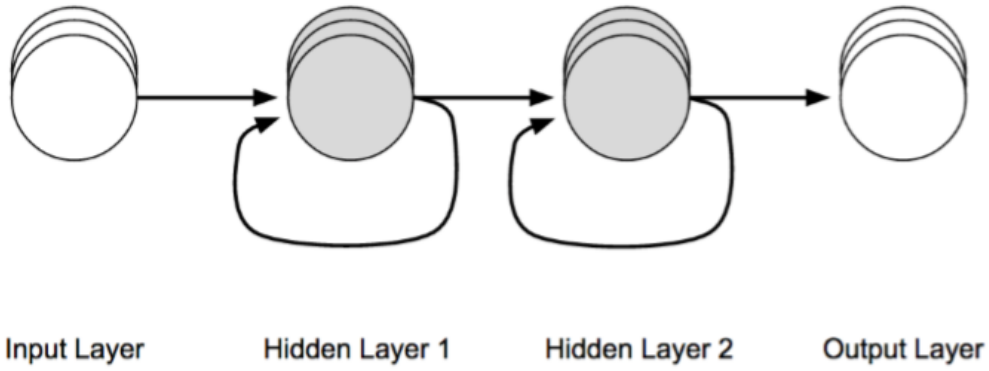
2.9. ábra. A konvolúció művelet, forrás: [89]

- **Összevonó réteg (Pooling layer)** - csökkenti a reprezentáció méretét, így kezelhetőbbé téve azt. Szomszédos értékeket von össze a maximumuk (MaxPooling), vagy átlaguk (AveragePooling) alapján.

- **Rekurrens réteg (Recurrent layer)** - a teljesen kapcsolt réteghez képest van egy olyan plusz tulajdonsága, hogy egy adott neuron megkapja a saját kimenetét is következő lépésben bemenetként. Formalizálva:

$$y_t = f_o ut(Vh_t)h_t = f_h(U_{x_t} + Wh_{t-1}) \tag{2.5}$$

, ahol h_t a háló egy rejtett vektora, amely t időpontra vonatkozó információt raktároz el, U, V, W pedig súlymátrixok. [15]



2.10. ábra. Rekurrens rétegek közti kapcsolatok, forrás: [89]

Aktivációs függvények

Az aktivációs függvényeket azért használjuk, hogy az egyes neuronok értékeit egy nemlineáris transzformáció alkalmazásával adjuk át a következő réteg számára. Néhány ezek közül:

- **Szigmoid** - más néven logisztikus függvény. [0,1] intervallumba transzformálja a kimenetet, ezáltal igaz-hamis, illetve probabilisztikus osztályozásra jól használható a kimeneti rétegen. Hátránya, hogy számítása költséges és szélsőségesen negatív vagy pozitív bemenet esetén eltüntetheti a gradienst.

$$sig(x) = \frac{1}{1 + e^{-x}} \tag{2.6}$$

- **Hiperbolikus tangens (tanh)** - [-1,1] intervallumba transzformálja a kimenetet, ezáltal nem csak pozitív értékeket alkalmazunk. Hátránya, hogy ennek a számítása is költséges és esetenként ez is eltüntetheti a gradienst. Kiszámítása:

$$tanh(x) = \frac{2}{1 + e^{-2x}} - 1 \tag{2.7}$$

- **Rectified Linear Unit (ReLU)** - a negatív bemeneteket nullára állítja, a pozitívakat változatlanul hagyja. Kiszámítása nem költséges: [89]

$$ReLU(x) = max(0, x) \tag{2.8}$$

Hibafüggvény (Loss Function)

A loss function egy olyan függvény, amely a neurális hálónk kimenetének valóságtól való eltérését adja meg. Ezt a függvényt úgy kell megválasztanunk, hogy a értékének minimalizálása jól reprezentálja a tanítás sikerességét. Különböző feladattípusokra különböző loss function-öket érdemes használni. Például:

- **Mean Squared Error** - regressziós feladatok esetén használatos:

$$L(W) = \frac{1}{2n}\sum_{i=1}^{n}(y^{(i)} - f(x^{(i)}; W))^2 \tag{2.9}$$

- **Binary Cross Entropy** - bináris osztályozási feladatok esetén használatos:

$$L(W) = \frac{1}{n}\sum_{i=1}^{n}(y^{(i)}log(-f(x(i); W)) + (1 - y^{(i)})log(1 - f(x(i); W)) \tag{2.10}$$

- **Categorical Cross Entropy** - multi-class osztályozási feladatok esetén használatos: [89]

$$L(W) = \sum_{i=1}^{n}(y^{(i)}log(-f(x(i); W)) \tag{2.11}$$

Optimalizáló algoritmusok

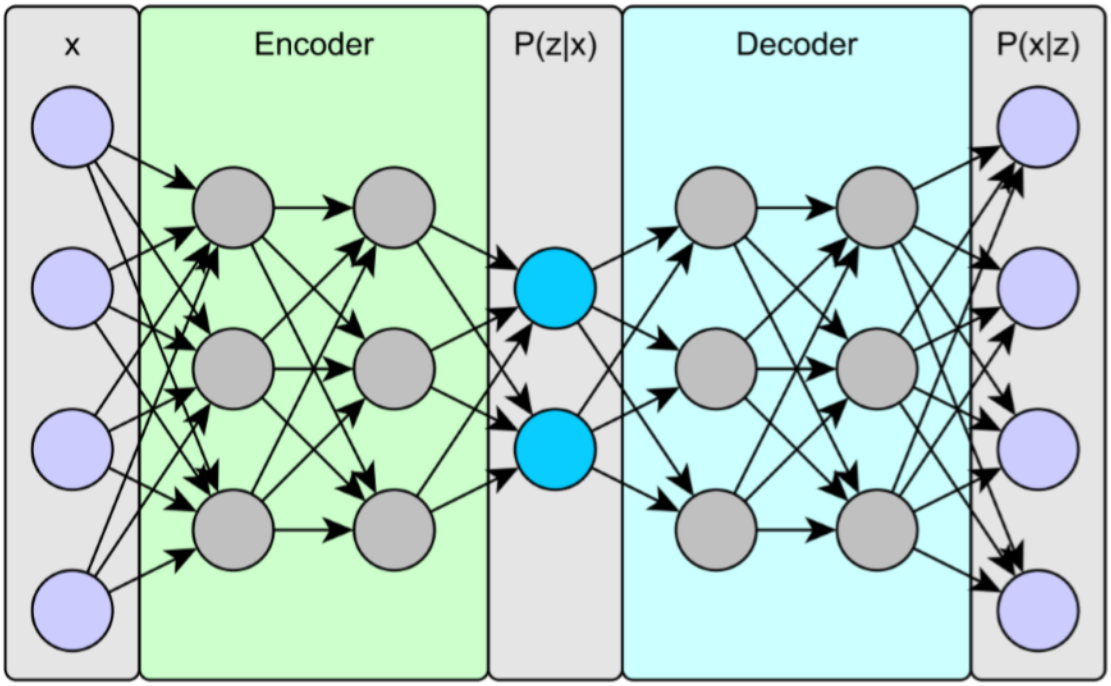
A loss function minimalizálása a súlyok igazításával gradiens módszerrel történik. Tehát a súlyok hibához való hozzájárulását kiszámítjuk a hibafüggvény gradienséből, majd ennek megfelelően változtatjuk a súlyokat. Azonban a gradiens csak lassan képes biztosítani a konvergenciát, ezért a gradiensereszkedés algoritmust különböző kiterjesztésekkel láthatjuk el. Ezeket a technikákat nevezzük optimalizálóknak. Néhány példa: gradiensereszkedés, Nesterov lendület, RMSProp, Adam. [89]

Architektúrák

A neurális hálózatok rétegei alapján több fajtájú architektúrát különböztethetünk meg. Néhány példa:

- **Deep Belief Network (DBN)** - felügyelet nélküli tanulást megvalósító architektúra. Restrictive Boltzmann gépeket használ a tanításhoz, majd előrecsatolt neurális hálót a finomhangoláshoz.

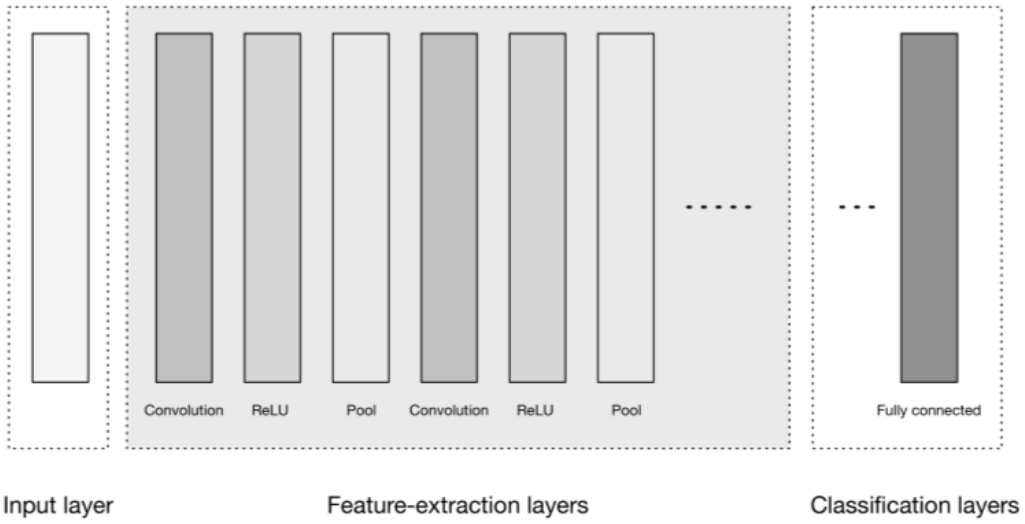
- **Variational Autoencoder (VAE)** - adathalmazok tömörített reprezentációinak megtanulására alkalmas architektúra. Gyakorlati felhasználása adathalmazok dimenzionalitásának csökkentése. Az encoder kimenete a bemeneti adat rekonstruált változata.



2.11. ábra. VAE háló architektúra, forrás: [89]

- **Generative Adversarial Network (GAN)** - az architektúra lényege, hogy két modellt tanítunk párhuzamosan. Az egyik képeket (vagy akár hangokat, szövegeket) generál, a másik pedig a bemenetről megpróbálja eldönteni, hogy az vajon valós, vagy az első modell által generált tartalom. A cél az, hogy az első modellünk olyan életszerű tartalmak generálására legyen képes, amivel képes megtéveszteni a másik modellt. Ezáltal

- **Konvolúciós Neurális Háló (CNN)** - az architektúra célja, hogy konvolúciók segítségével releváns alacsony szintű jellemzőket azonosítson be a bemeneti adatokon és ezeket használja fel osztályozáshoz. Az input után konvolúciós és pooling rétegek jellemzik, majd a hálót egy, vagy több teljesen kapcsolt réteggel zárjuk.



2.12. ábra. CNN architektúra, forrás: [89]

- **Rekurrens Neurális Háló (RNN)** - az architektúra rekurrens rétegeket használ, hogy korábbi időpontok értékeiből is tudjon tanulni, így például nagyon jól kezelhetőek vele szekvenciák. [89]

3. fejezet

Adathalmaz

Ebben a fejezetben a munkám kapcsán felkutatott és alkalmazott adathalmazokról lesz szó. Egy deep learning megoldás tervezésének első lépéseként érdemes egy alkalmas kiinduló adathalmazt kiválasztani. Ezt aztán a modell tanítására és tesztelésre használjuk.

3.1. Kiválasztási szempontok

Egy adott MIR probléma felügyelt tanulási módszerrel való megközelítésekor az adathalmazzal szemben alapvetően két nagyon fontos szempontot kell vigyelembe vennünk.

1. **Az adatok jellemzői:** a zenékre vonatkozó szerzői jogok miatt a kutatók általában nem tudnak hozzáférni "valós", mindennapi életben előforduló zenékből álló adathalmazhoz, így a fellelhető adathalmazokat általában mesterségesen, kutatási célra fejlesztik. Emiatt mérlegelnünk kell például a következő jellemzők figyelembe vételével:

    - Elérhető számunkra? Ingyenes?
    - A nyers hanganyag rendelkezésre áll, vagy csak valamilyen alternatív reprezentáció? Utóbbi esetén mi az pontosan?
    - Átestek e valamilyen előfeldolgozáson a minták?
    - Milyen minőségűek a felvételek? Zajosak?
    - Egyszólamúak, vagy többszólamúak a hangminták?

- Milyen hosszúak a minták? Egy hangjegy, fix hosszúságú (például 10 másodperc), vagy teljes zenék?
- Mennyire szerteágazó az adathalmaz? Az általunk vizsgálni kívánt, vagy egyéb fontos tulajdonsággal kapcsolatban van túlreprezentált, vagy alulreprezentált érték?

2. **A metaadatok jellemzői:** egy adott MIR probléma felügyelt tanulási módszerrel való megközelítéséhez fontos, hogy adathalmazunkhoz rendelkezésre álljanak a tanítani kívánt jellemzők metaadatként, például:

   - Hangszerek
   - Zene feldolgozásai
   - Tempók
   - Dallamok
   - Stílus besorolások
   - Egyéb címkék, metaadatok...

   Fontos, hogy ezek jellege is megfeleljen az igényeinknek. Például hogy az címkék egy adott időpontra vonatkoznak, vagy statikus egy bemenetre, vagyis hogy a címkézés erős, vagy gyenge.

   A metaadatok fogják meghatározni a lehetséges osztályozási feladat típusát is. Például ha bináris értékekkel rendelkezünk minden bemenet minden címkéjére, azzal egy multi-label osztályozást valósíthatunk meg. Míg ha az egyes bemenetekhez egy címke tartozik (és kettőnél több lehetséges címke közül választhatunk), az egy multi-class osztályozást fog jelenteni.

Egy remek gyűjtőoldalnak bizonyult a Nemzetközi MIR Közösség (International Society of Music Information Retrieval - ISMIR) weboldala. Számos adathalmazhoz biztosít hivatkozást, kulcsszavakban leírja a bennük található adatok és metaadatok jellegét és azt, hogy a hangfájlok csatolva vannak-e az adott adathalmazhoz. [91]

Kísérleteimhez első körben egy egyszerű, könnyen tanítható adathalmazt kerestem (ld. Philharmonia Orchestra). Majd miután ezen megbizonyosodtam a modell működéséről, kísérletet tettem egy reprezentatívabb adathalmazba való átültetésre (ld. OpenMIC).

3.2. Philharmonia Orchestra

Kutatásom első fázisában a Philharmonia Zenekar ingyenesen elérhető hangminta könyvtárát használtam fel. A könyvtárban egyszólamú mintákat találunk, melyeket a zenekar tagjai vettek fel és tettek közzé használatra. A minták a főkönyvtáron belül a bennük megszólaló hangszer nevével megegyező könytárban találhatóak. Ez biztosítja a hangszer címkézés metaadatot.

Egy minta az adott hangszerrel egy hangjegy lejátszását tartalmazza. Mivel a minták rövidek és tiszták, a hangszerek közti különbségek relatív könnyen felismerhetőek. Az adathalmazban összesen 20 hangszer szerepel. Az egyes hangszerek különböző mennyiségű mintával vannak jelen, hangszerenként körülbelül 100 és 1000 közti fájl érhető el, összesen 12 992 fájl.

3.3. OpenMIC

A többszólamúság bevezetését kutatásomban az OpenMIC [92] adathalmaz felhasználásával értem el. Ez szintén egy ingyenesen elérhető adathalmaz, melynek tartalma 20 000 többszólamú minta 20 különböző hangszert lefedve. A minták 10 másodperc hosszú kivonatok a Free Music Archive-on [93] elérhető zenékből. [92]

Az adathalmaz igen előnyös metaadatok tekintetében. Amellett, hogy az összes hanganyag elérhető .ogg formátumban, a hangszer címkék egy fájlban elérhetőek. Minden egyes mintához hangszereként definiált, hogy az adott hangszer megszólal-e a mintában. Az, hogy hangszerenként külön információnk van a mintákra vonatkozóan, lehetőséget ad arra, hogy multi-label osztályozást valósítsunk meg. [92]

A címkézéshez bárki hozzájárulhatott. Nyílt, tömeges folyamat volt. Hátránya, hogy nem minden minta összes hangszeréhez született egyértelmű címke. A gyakorlatban egy mintához hangszerenként két változó tartozik: egyik egy bináris maszk változó, amely arra utal, hogy van-e információnk az adott minta vonatkozásában az adott hangszer jelenlétéről. A másik változóban egy százalékérték található, amely az adott hangszer előfordulásának valószínűségét mondja meg a címkézést végző közösség ítéletei alapján. Ez természetesen csak akkor tekinthető relevánsnak, ha a maszk változóban a vonatkozó érték igaz. Ezáltal a címkézést rugalmasnak tekinthet-

jük, hiszen mi magunk határozhatunk meg egy küszöbértéket a hangszer jelenlétére vonatkozóan. [92]

3.3.1. VGGish

Az OpenMIC adathalmazhoz továbbá csatoltak a nyers hanganyag alternatívájaként egy "VGGish" elnevezésű reprezentációt is, mint felhasználható bemenetet. Ez a reprezentáció a nyers hanganyag feldolgozása egy VGG [94] architektúrán alapuló mély neuronhálón, amely a VGGish elnevezést kapta. Ez a neuronhálós modell egyfajta előfeldolgozást valósít meg számunkra. A belőle kapott reprezentáció 0.96 másodperces, egymást nem átfedő időablakonként egy 128 értéket tartalmazó leíró vektorból áll. Felhasználásával mi egy sekélyebb downstream, vagyis utána illesztett modell használatával érhetünk el kielégítő osztályozási eredményeket. [95]

4. fejezet

Módszertan

Ebben a fejezetben az alkalmazott módszertant mutatom be. Először ismertetem az alkalmazott előfeldolgozási módszereket. Ezután a tanító modellek architektúráját mutatom be. Végül pedig szót ejtek az ismertetett módszerek implementációjáról.

4.1. Bemeneti adatok, előfeldolgozás

Az OpenMIC adathalmazzal két formában kapjuk meg a bemeneti adatokat. Egyrészt elérhetőek a nyers hanganyagok .ogg formátumban. Ezen kívül rendelkezésre állt a VGGish nevű reprezentáció is, amelyről a korábbi 3.3.1 fejezetben írtam. A bemeneti adatokat tanítás előtt azonban néhány előfeldolgozási folyamaton vittem végig részben a megvalósíthatóság, részben pedig a performancianövelés érdekében. Ezekről írok a következő alfejezekben.

4.1.1. Alternatív reprezentációk kinyerése

A .ogg fájlokból beolvasott nyers audioval való tanítás sajnos nem volt lehetséges, mivel ez egy memória szempontjából igen költséges reprezentáció, amihez a hardver eszközöm kapacitása kevésnek bizonyult. Helyette a dolgozat korábbi fejezetében bemutatott melspectogram és MFCC reprezentációkat használtam fel. Mindkét reprezentációt a beolvasott nyers hullámforma reprezentációból nyertem ki az elméleti háttér fejezetben leírt módon.

4.1.2. Adathalmazok normalizálása

Az adatok normalizálása a tanulási folyamat gyorsítására szolgál. A normalizálást úgy érjük el, hogy az adatokat arányosan a 0 és 1 értékek közé transzformáljuk. Erre egy szokásos megoldás:

$$x(_i) := (x(_i) - x_{\min})(x_{\max} - x_{\min}) \tag{4.1}$$

A normalizált adathalmon a gradiens általában hamarabb csökkenthető, ezért a tanítási folyamat rövidebb lehet. [96]

4.1.3. Osztályok kiegyensúlyozása

Tanító modelleknél gyakori probléma, hogy a tanulni kívánt osztályok egy része alul van reprezentálva. Dolgozatom multi-label osztályozása esetén két osztályról beszélhetünk: igaz (jelen van az adott hangszer) és hamis (nincs jelen az adott hangszer). Mivel sok esetben a hamis értékek jelentős többségben voltak, - egyes hangszerek esetén az összes adat közel 80%-át is kitette - ezért a modell jó pontosságot tudott elérni csupán "hamis" predikciókkal.

Ezt elkerülendő, három megoldás jöhetett szóba:

- **Alulmintavételezés (Undersampling)** - a többségi osztályokból véletlenszerűen eltávolítunk annyi elemet, hogy az osztályok kiegyensúlyozottá váljanak.
- **Túlmintavételezés (Oversampling)** - a kisebbségi osztályok véletlenszerű elemeit duplikáljunk addig, amíg az osztályok kiegyensúlyozottá válnak.
- **Adat augmentáció (Data augmentation)** - hasonlóan az oversampling technikához a kisebbségi osztály elemeit bővítjük. Ebben az esetben a meglévő elemek transzformálásával mesterségesen hozunk létre új elemeket (pl. képeknél forgatás). [97]

Dolgozatom keretében az undersampling módszert alkalmaztam.

4.1.4. Tanító-, tesztelő- és validáló halmaz kialakítása

Tanító modellek alkalmazásakor fontos, hogy legalább kettő, de inkább három független adathalmazzal rendelkezzünk:

- **Tanító adathalmaz (Train Set)** - ebből a halmazból tanul és ezt a halmazt látja a modellünk.
- **Validáló adathalmaz (Validation Set)** - a tanítási lépések (epoch-ok) között ezen a tanító halmaztól független adathalmazon kiértékeljük modellünk működését. Ezáltal visszajelzést tudunk adni a tanítási folyamat felé és ha szükséges, változtathatunk a hiperparamétereken. Modellünk tehát ezt a halmazt időnként látja, de nem ebből tanul
- **Teszt adathalmaz (Test Set)** - a tanítási folyamat végén ezen a halmazon értékeljük ki modellünk teljesítményét. Modellünk ezt a halmazt a tanítás alatt egyáltalán nem látja és ezáltal nem is tud tanulni belőle. Ezzel a függetlenséggel biztosítjuk a kiértékelés torzítatlanságát. [98]

Bevett szokás, hogy e három halmazt körülbelül 60%-20%-20% arányban osztjuk fel a tanító halmaz javára. [98] Dolgozatom során én is ezt az elvet követtem.

4.2. Architektúra

A következő alfejezetekben a különböző tanító modell kísérletek architektúráját mutatom be.

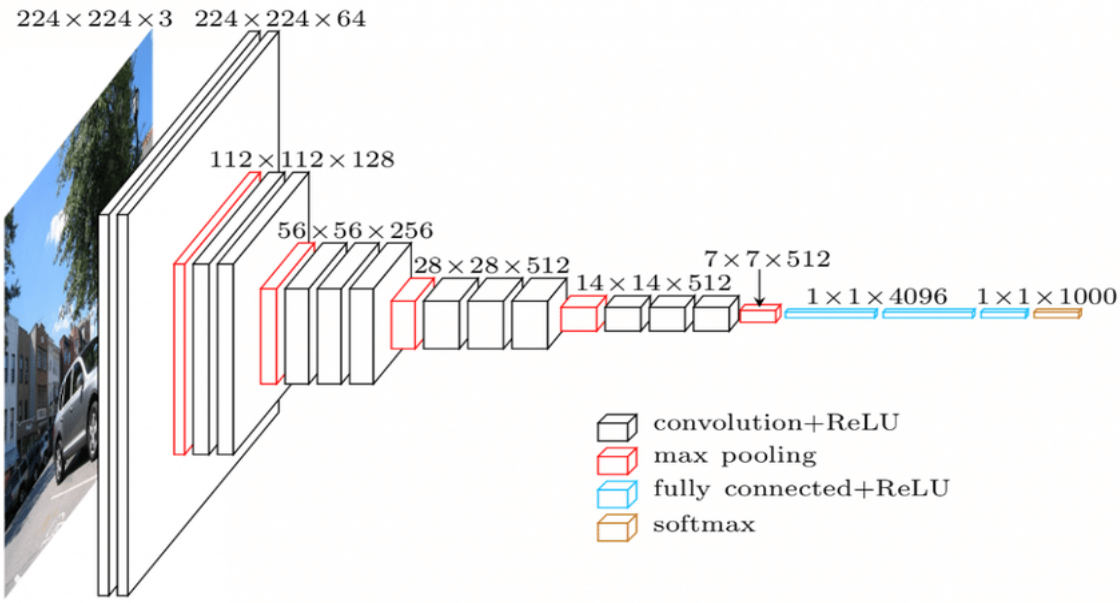
4.2.1. Hagyományos Machine Learning (Modeling Baseline)

Az OpenMIC dataset megalkotói bemutattak egy példa modellt, amely egy véletlen erdő osztályozó (Random Forest Classifier). [92] Ez egy hagyományos gépi tanulási algoritmus. Az osztályozó 100 eldöntési fából áll, a maximális mélysége 8. Ez szolgált munkám kiinduló pontjaként. A továbbiakban Modeling Baseline elnevezéssel hivatkozok rá.

A Modeling Baseline futtatását alapvetően a VGGish reprezentációra implementálták. Dolgozatom keretében megvalósítottam ugyanennek az osztályozónak a melspectogram, ileltve MFCC reprezentációkon való tanítását is. Ezáltal minden reprezentációra kaptam egy viszonyítási alapot.

4.2.2. Deep CNN

Az első mély tanulási megközelítésem a VGG-16 [94] architektúrához hasonló mély konvolúciós neuronháló megvalósítása. Ennek megfelelően Deep CNN néven fogok rá hivatkozni a továbbiakban. A VGG-16 egy 2014-ben bemutatott architektúra, amelyet 224x224 méretű RGB képek tanítására hoztak létre. Több tanulmány is azt állítja, hogy az ehhez hasonló mély konvolúciós hálók hangok tanítására is alkalmasak lehetnek (például [3]). Erre az állításra az intuíció pedig az lehet, hogy a hangok különböző kétdimenziós reprezentációi képek, melyekben a mély konvolúciós hálók mintákat, összefüggéseket tudnak keresni.



4.1. ábra. VGG-16 neurális háló architektúrája, forrás: [99]

Az architektúra a következő elemekből épül fel:

- Konvolúciós rétegek, ReLU aktivációs függvénnyel
- Max-pooling rétegek
- Opcionálisan Dropout rétegek
- Flattening réteg
- Fully-connected rétegek, ReLU aktivációs függvénnyel
- Fully-connected utolsó réteg, szigmoid aktivációs függvénnyel

A bemenetet először két konvolúciós rétegnek adjuk át. Ezt követi egy Max-pooling réteg, majd esetleg a nagyobb random-faktor érdekében egy Dropout réteg. Ezután ugyanilyen sorrendben még háromszor-négyszer (attól függően, milyen mély architektúrát szeretnénk megvalósítani) hozzáfűzzük ezeket a rétegeket az előző kimenetéhez. A konvolúciós rétegek ahogy zsugorodnak a Max-poolingok hatásaként, úgy egyre több szűrővel rendelkeznek. Következőnek egy Flattening réteg következik, amely után a Fully-connected rétegeket csatolhatjuk. Dropout rétegeket itt is alkalmazhatunk. Az utolsó Fully-connected layerhez szigmoid aktivációs függvény tartozik, hogy 0 és 1 közötti értékeket kapjunk a kimenetre. Ebből nyerjük ki a végleges igaz-hamis értékeket.

Erre az architektúrára csak a melspectogram reprezentációt lehetett ráilleszteni. A VGGish 10 és az MFCC 20 méretű dimenziója a konvolúciók és Max-poolingok következtében 0 alá redukálódott volna, ami nem lehetséges.

Réteg	**Melspec**
Input	1 × 128 × 430
Conv2D	64 × 128 × 430
Conv2D	64 × 128 × 430
MaxPool2D	64 × 63 × 214
Dropout	64 × 63 × 214
Conv2D	128 × 63 × 214
Conv2D	128 × 63 × 214
MaxPool2D	128 × 31 × 106
Dropout	128 × 31 × 106
Conv2D	256 × 31 × 106
Conv2D	256 × 31 × 106
MaxPool2D	256 × 15 × 52
Dropout	256 × 15 × 52
Conv2D	512 × 15 × 52
Conv2D	512 × 15 × 52
MaxPool2D	512 × 7 × 25
Flatten	89600
Fully-connected	4096
Fully-connected	2048
Fully-connected	1

4.1. táblázat. Deep CNN architektúra rétegei és az egyes rétegek outputjainak mérete Melspectogram reprezentáció esetén

4.2.3. Shallow CNN (VGGish Embedding Downstream CNN)

Kísérleteim során a Deep CNN után egy sekélyebb architektúrát is kialakítottam. Ezt a továbbiakban Shallow CNN néven fogom említeni. A Shallow CNN-t három okból hoztam létre:

- Amint azt az adathalmazok körében a 3.3.1 alfejezetben is említettem, a VGGish reprezentáció már előre tanítva van. Ezért erre nem lenne hatékony

egy Deep CNN mélységű architektúra alkalmazása.

- A melspectogram, illetve MFCC reprezentációk esetében a kísérletek arra utaltak, hogy túl kevés az adatunk egy ilyen mély architektúra alkalmazásához. Az eredmények azt mutatták, hogy a modell fölöslegesen próbál nagyon mély összefüggéseket keresni az adatok között. Ezek feltárásához az adott ezres nagyságrendű bemeneti mintaszám helyett legalább tízerzes, de inkább százezres kellene.

- Az MFCC és VGGish reprezentációk mérete nem megfelelő a megadott architektúra dimenzió redukciói miatt.

Maga a Shallow CNN a Deep CNN leegyszerűsített változata, tehát egy sekélyebb konvolúciós neuronháló megvalósítása. A rétegek típusa megegyezik a Deep CNN-ével, azonban számuk és méretük jelentősen kisebb (lásd 4.2 táblázat). Ennek eredményeképp jelentősen csökkent a tanítási idő, a modell teljesítőképessége pedig nőtt.

Réteg	**VGGish**	**MFCC**	**Melspec**
Input	1 × 10 × 128	1 × 20 × 430	1 × 128 × 430
Conv2D	64 × 10 × 128	64 × 20 × 430	64 × 128 × 430
Conv2D	64 × 10 × 128	64 × 20 × 430	64 × 128 × 430
MaxPool2D	64 × 4 × 63	64 × 9 × 214	64 × 63 × 214
Dropout	64 × 4 × 63	64 × 9 × 214	64 × 63 × 214
Conv2D	128 × 4 × 63	128 × 9 × 214	128 × 63 × 214
MaxPool2D	128 × 1 × 31	128 × 4 × 106	128 × 31 × 106
Dropout	128 × 1 × 31	128 × 4 × 106	128 × 31 × 106
Flatten	3968	54272	420608
Fully-connected	128	128	128
Fully-connected	1	1	1

4.2. táblázat. Shallow CNN architektúra rétegei és az egyes rétegek outputjainak mérete reprezentációtól függően

4.3. Hiperparaméterek

Az előfeldolgozási technikákon és az architektúra kialakításán kívül fontos szerepe volt a megfelelő hiperparaméterek megválasztásának is. Ezek többségét best-practice-ek alapján határoztam meg, majd kísérletezések során optimalizáltam. A különböző hiperparaméterek:

- **Epoch-ok száma + Early Stopping** - a tanulásban az epoch-ok száma jelenti azt, hogy a modell hány alkalommal megy végig a bemeneti tanítóhalmazon. Ennek 50 értéket adtam és early stopping technikával egészítettem ki. Az early stopping lényege, hogy minden epoch után egy kiválaszott érték alapján megvizsgáljuk, hogy a tanulás még hatékony-e, és ha nem, akkor leállítjuk. Én ezt a learning rate függvényében, magas learning rate esetén 4, alacsony esetén 7 epoch türelmi küszöbbel a validation loss csökkenésének vizsgálatára alapozva vezettem be.

- **Learning rate** - a tanulás mértékét határozza meg. Túl magas érték esetén előfordul, hogy a modellünk nem tud konvergálni az optimális megoldás felé, mert folyton átugorja azt. Túl alacsony érték esetén pedig a tanítás folyamata túl hosszú lehet, illetve előfordulhat, hogy bizonyos nem kielégítő lokális minimum gradiens érték irányába konvergálunk, amelyet épp át kellene ugranunk az optimális megoldás megtalálásához.

- **Dropout** - az architektúra egyes rétegei közé dropout rétegeket illesztettem, amelyek az adott rétegig beállított súlyok egy részét véletlenszerűen eldobják az overfitting elkerülése érdekében. Ezt bármely két réteg közé beilleszthetjük és bármilyen mértékű lehet. Én a súlyok 20%-ára állítottam be őket, az architektúrában feltüntetett pontokon.

- **Mini-batch size** - ennek az értéke adja meg, hogy egyidőben hány konkrét bemeneti adaton végezzük el a léptetés és visszacsatolás folyamatokat. Ez az érték bármi lehet egytől a bemeneti adataink számáig. Minél kisebb az érték, annál kevésbé memóriaigényes a tanítás és minél nagyobb, annál gyorsabb. Best-practice-ként alacsony kettő hatvány értékeket szokás használni. Én 64-nek választottam.

5. fejezet

Kísérletek, eredmények

Ebben a fejezetben a releváns mérőszámok ismertetése után kiértékelem kísérletem eredményeit. Összevetem a kiinduló modell és saját fejlesztéseim teljesítményét a különböző bemeneti reprezentációkon.

5.1. Mérőszámok

A modellek teljesítményét F értékek alapján vetettem össze, egyrészt hangszerenként külön-külön, másrészt a hangszerenkénti átlagot véve is. Az F értékhez a következő metrikák vezettek el:

- **Pontosság (Accuraccy)** - a modell helyes előrejelzéseinek száma osztva az összes előrejelzés számával. Ez már magában egy jó mérőszám tud lenni, azonban multi-label osztályozások esetén érdemes a helytelen előrejelzések fajtájával is kalkulálni. Ezek a valótlan igazak (false positives) és valótlan hamisak (false negatives).

		Actual	
		Positive	Negative
Predicted	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

5.1. ábra. Előrejelzések fajtái, forrás: [100]

- **Veszteség (Loss)** - a veszteségfüggvény eredménye. A modell predikcióinak a valóságtól való eltérését összeadva kapjuk meg. A modell célja tanuláskor ennek az értéknek a minimalizálása.
- **Precizitás (Precision)** - valós igaz predikációk száma osztva a valós és valótlan igazak számának összegével.
- **Felidézés (Recall)** - valós igaz predikációk száma osztva a valós igaz és valótlan hamis predikációk összegével.

$$\textbf{Precision} = \frac{\text{True Positive}}{\text{Actual Results}} \quad \text{or} \quad \frac{\text{True Positive}}{\text{True Positive + False Positive}}$$

$$\textbf{Recall} = \frac{\text{True Positive}}{\text{Predicted Results}} \quad \text{or} \quad \frac{\text{True Positive}}{\text{True Positive + False Negative}}$$

5.2. ábra. Precizitás és felidézés, forrás: [100]

- **F érték (F score)** - a precizitás és felidézés értékek harmónikus közepe. Ez lesz tehát a meghatározó mérőszám modelljeink összevetésénél. [100]

$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

5.3. ábra. F score képlete, forrás: [100]

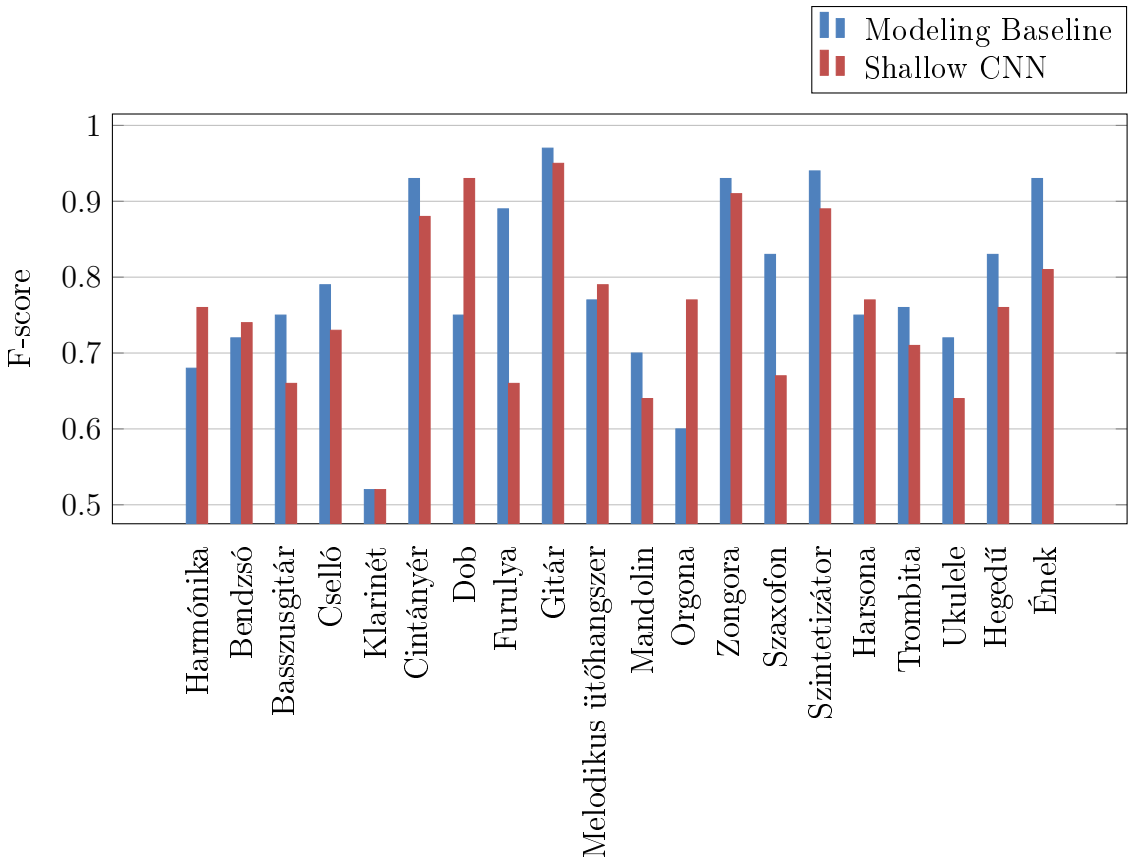
5.2. Eredmények

A végső eredmények csak az OpenMIC datasettel kapcsolatos kísérleteket tartalmazzák. Az eredmények összevetésére a hangszerenkénti F score-okat használtam fel. Ez modellenként 20 hangszer, ami 20 különböző értéket jelent. Ezért minden modell végső összevetési alapjának két összesítő értéket választottam: az F-score-ok átlagát és a 0,70 feletti F score-t meghaladó hangszerek számát. Utóbbira a továbbiakban jól tanított hangszerként fogok hivatkozni. Saját intuícióm vezetett arra a következtetésre, hogy 0,70-es F score felett már életszerűen használható értékről beszélhetünk.

A továbbiakban a Modeling Baseline és a Shallow CNN architektúrák különböző eredményeit mutatom be. Az eredményeket mindenhol többszöri futtatás maximálisan elért értékei reprezentálják.

5.2.1. VGGish kiindulás

Először a VGGish reprezentáció tekintetében tettem kísérleteket, mivel a Modeling Baseline eredeti implementációja ezt használta fel. Első lépésként ezt futtattam módosítás nélkül, majd lecseréltem a felhasznált RFC-t a saját CNN architektúrámra.



5.4. ábra. Modeling Baseline és Shallow CNN eredményei optimalizációk nélkül VGGish reprezentáción

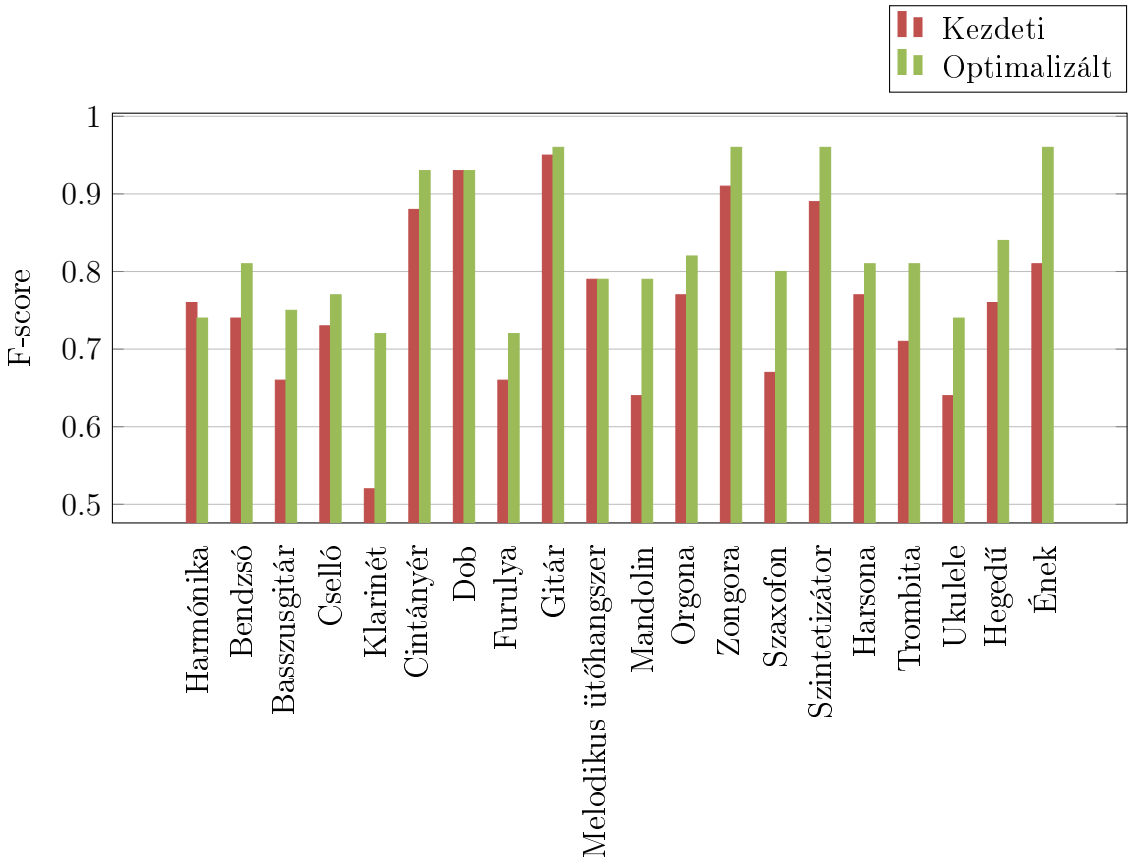
Az eredmények hangszerenként a 5.4 ábrán láthatóak. Az F-score-ok átlaga és a jól tanított hangszerek száma kicsivel magasabb volt a Modeling Baseline modellen, mint a Shallow CNN-en. Előbbi 0,79 átlag F score-t és 17 jól tanított hangszert, utóbbi 0,76 átlag F score-t és 14 jól tanított hangszert produkált.

Ami még a 5.4 diagramról leolvasható, hogy a hangszerek között vannak jobban és kevésbé taníthatóak. Például a klarinét és mandolin mindkét modellen relatív alacsony F-értéket hozott. Ezzel szemben a gitár és zongora tanítása mindkét modell esetében 0.9 feletti F-értéket eredményezett.

A hangszereket vizsgálva a modellek között megjelentek nagyon hasonló és különböző értékek is. A legangyobb különbséget a Shallow CNN javára a dob mutatta, itt 0,75 állt szemben 0,93 értékkel, ami 0,18 F-score eltérés. A Modeling Baseline javára pedig a furulyával kapcsolatos számok jelentették a legnagyobb, 0,23 F-score eltérést. Itt 0,66 és 0,89 voltak az értékek.

5.2.2. Optimalizáció

Következő lépésként implementáltam a különböző előfeldolgozási adatoptimalizáló technikákat, mint például undersampling és normalizálió. Ezután szintén a VGGish reprezentáción futtatva a modellek teljesítménye javulást ért el: mindkét modell F score átlaga egyaránt 0,83-ra növekedett a korábbi 0,79-ről és 0,76-ról. Ezen felül a jól tanított hangszerek száma mindkét modell esetén elérte a maximális 20-at a korábbi 17 és 14 helyett. Ez a ShallowCNN számára magasabb relatív javulást jelent, ennek hangszerenkénti összevetése a 5.5 ábrán látható.



5.5. ábra. Shallow CNN optimalizációk előtt és után VGGish reprezentáción

Így végül optimalizált adathalmazon bár hangszerenként vannak kisebb eltérések mindkét megoldás irányába, összesítve ugyanolyan eredményt képes produkálni egy hagyományos gépi tanulási algoritmus, mint egy mély tanulási modell. Legalábbis a kis méretű, előre tanított VGGish reprezentáción.

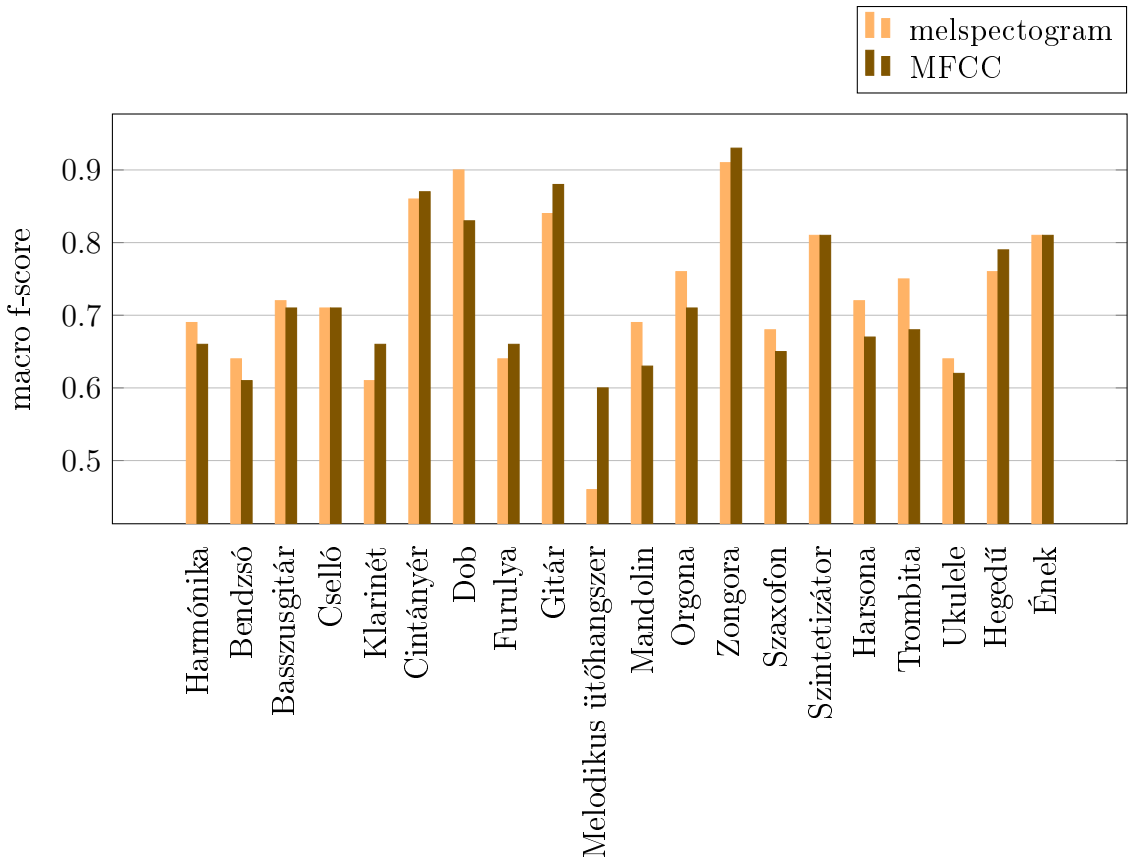
5.2.3. Alternatív reprezentációk

A VGGish reprezentáció után mindkét modellt futtattam melspectogram és MFCC reprezentációkat felhasználva. A melspectogram tekintetében végeztem az adatoptimalizálás előtt is kísérleteket. Már itt észrevehető volt: a nagyobb, komplexebb reprezentáción a mély tanulás jobb eredményt mutat. A Modeling Baseline 0,56-os F-score-t és 4 jól tanított hangszert, a Shallow CNN architektúra pedig 0,61-es F score-t és 5 jól tanított hangszert produkált.

Ám az adatoptimalizáció utáni értékek még jelentősebbek. Az optimalizáció kapcsán lényegében ugyanaz látszik, mint a VGGish esetén: mindkét modellen javulást eredményezett az optimalizálás, de a Shallow CNN-en szignifikánsabb a különbség. A végső értékek a Modeling Baseline tekintetében 0,65-ös F-score és 5 jól tanított hangszer. Ezzel szemben a Shallow CNN F-score-ja 0,73-ra, jól tanított hangszereinek száma 12-re nőtt. Ez a Modeling Baseline esetén plusz 0,09 F-score-t és 1 jól tanított hangszert jelent, míg a Shallow CNN esetén plusz 0,12 F-score-t és 7 jól tanított hangszert. Így tehát a Modeling Baseline és Shallow CNN melspectogramon vett végső eredményei közül a Shallow CNN-é magasabb 0,08 átlag F-score-ral és 7 jól tanított hangszerrel. A hangszerenkénti értékek a 5.6 ábráról olvashatóak le.

5.6. ábra. Modeling Baseline és Shallow CNN eredményei melspectogram reprezentáción adatoptimalizáció után

A melspectogram reprezentáció után az MFCC következett. Az MFCC reprezentáción mindkét modell a melspectogramon való futtatáshoz hasonló eredményt adott. A Modeling Baseline 0,61-es átlag F-score-t és 4 jól tanított hangszert ért el, ez 0,04 F-score-ral és 1 jól tanított hangszerrel alacsonyabb, mint melspectogram esetén. A Shallow CNN tekintetében pedig 0,72 átlag F-score és 8 jól tanított hangszer lett a végső eredmény. Ezek az értékek 0,01 F-score és 4 jól tanított hangszer különbséget mutatnak a melspectogram javára. A hangszerenkénti értékek a 5.7 ábráról olvashatóak le.



5.7. ábra. Shallow CNN végső eredményei melspectogram és MFCC reprezentáción

Melspectogram, illetve MFCC reprezentációk közti jelentős effektívségbeli különbség tehát nem jelenik meg az eredményekben. Ezt érdemes lenne tovább vizsgálni. Mélyebb architektúrára, illetve több bemeneti adatra lenne szükség a bizonyításához, vagy cáfolásához.

5.2.4. Összehasonlítás

A bemutatott eredményekből és a 5.1 összefoglaló táblázatból látszik, hogy melspectogram és MFCC bemeneti reprezentációk esetén az általam prezentált Deep Learning architektúra (Shallow CNN) jobb eredményeket produkál, mint egy hagyományos gépi tanulási algoritmus (Modeling Baseline).

	Átlagos F-score	**Jól tanított hangszerek száma**
Modeling Baseline VGGish kezdeti	0,79	17
Shallow CNN VGGish kezdeti	0,76	14
Modeling Baseline melspectogram kezdeti	0,56	4
Shallow CNN melspectogram kezdeti	0,61	5
Modeling Baseline MFCC kezdeti	Nincs adat	Nincs adat
Shallow CNN MFCC kezdeti	Nincs adat	Nincs adat
Modeling Baseline VGGish optimalizált	0,83	20
Shallow CNN VGGish optimalizált	0,83	20
Modeling Baseline melspectogram optimalizált	0,65	5
Shallow CNN melspectogram optimalizált	0,73	12
Modeling Baseline MFCC optimalizált	0,61	4
Shallow CNN MFCC optimalizált	0,72	8

5.1. táblázat. Kísérletek és eredmények összefoglalva

A reprezentációk tekintetében elmondható, hogy a melspectogram és MFCC jelentősen gyengébb eredményeket értek el, mint a VGGish. Ennek fő oka az adathalmaz méretéből fakad. Ugyanis a VGGish reprezentáció célzottan sekélyebb architektúrákra és kevesebb adatra lett megalkotva.

6. fejezet

Összegzés, kitekintés

Dolgozatom keretében megvizsgáltam a zenei információk kinyerése (MIR) tudományterület feladatait, megismerkedtem az automatikus hangszerfelismerés problémájával, ennek megközelítéseivel. Felkutattam különböző gépi tanulási és mély tanulási megoldásokat, tanító adathalmazokat.

Bemutattam egy CNN architektúrát, amellyel három különböző bemeneti reprezentáción kísérleteztem. A kísérletek konklúziójaként az látszik, hogy míg a már előre tanított kisebb dimenziójú reprezentáción nagyjából egyforma teljesítményt nyújt, a nagyobb felbontású reprezentációkon már jobban teljesít a bemutatott mély tanulási architektúra, mint egy hagyományos gépi tanulási megoldás.

A jövőben több módon is tovább lehetne optimalizálni a bemutatott modellt. A tapasztalataim alapján úgy gondolom a legnagyobb javulást az adathalmaz bővítése eredményezné. Erre azonban nem biztos, hogy van lehetőség, így én a következő továbbfejlesztéseket fontolnám meg:

- Undersampling helyett oversampling, vagy data augmentation technikát alkalmazni az adatok előfeldolgozásakor
- A jelenlegi CNN architektúrát tovább mélyíteni, vagy helyette RNN architektúrát implementálni
- Az early stopping technika mellé model checkpointokat is bevezetni.

Irodalomjegyzék

[1] Florian Eyben és tsai. "Universal Onset Detection with Bidirectional Long Short-Term Memory Neural Networks." *Proceedings of the 11th International Society for Music Information Retrieval Conference* (Utrecht, Netherlands). Utrecht, Netherlands: ISMIR, 2010. aug., 589–594. old. DOI: 10.5281/zenodo.1417131. URL: https://doi.org/10.5281/zenodo.1417131.

[2] Philippe Hamel és Douglas Eck. "Learning features from music audio with deep belief networks". *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR)*. Utrecht, The Netherlands, 2010. aug., 339–344. old.

[3] Keunwoo Choi és tsai. "A tutorial on deep learning for music information retrieval". *arXiv preprint arXiv:1709.04396* (2017).

[4] Keith Dana Martin és Youngmoo E. Kim. "Musical instrument identification: A pattern-recognition approach". 1998.

[5] Judith Brown. "Computer Identification of Musical Instruments Using Pattern Recognition With Cepstral Coefficients as Features". *The Journal of the Acoustical Society of America* 105 (1999. ápr.), 1933–41. old. DOI: 10.1121/1.426728.

[6] A. Eronen és A. Klapuri. "Musical instrument recognition using cepstral coefficients and temporal features". *2000 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.00CH37100)*. 2. köt. 2000, II753–II756 vol.2.

[7] Jeremiah Deng, Christian Simmermacher és Stephen Cranefield. "A Study on Feature Analysis for Musical Instrument Classification". *IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics : a publication of the*

IEEE Systems, Man, and Cybernetics Society 38 (2008. máj.), 429–38. old. DOI: 10.1109/TSMCB.2007.913394.

[8] Daulappa Bhalke, C. Rao és D. Bormane. "Automatic musical instrument classification using fractional fourier transform based- MFCC features and counter propagation neural network". *Journal of Intelligent Information Systems* 46 (2015. máj.). DOI: 10.1007/s10844-015-0360-9.

[9] Juan José Burred, Axel Roebel és Thomas Sikora. "Dynamic Spectral Envelope Modeling for Timbre Analysis of Musical Instrument Sounds". *Audio, Speech, and Language Processing, IEEE Transactions on* 18 (2010. ápr.), 663 –674. old. DOI: 10.1109/TASL.2009.2036300.

[10] J. Eggink és Guy Brown. "A missing feature approach to instrument identification in polyphonic music". 5. köt. 2003. nov., 49–. old. ISBN: 0-7803-7850-4. DOI: 10.1109/ICASSP.2003.1200029.

[11] Dimitrios Giannoulis és Anssi Klapuri. "Musical Instrument Recognition in Polyphonic Audio Using Missing Feature Approach". *Audio, Speech, and Language Processing, IEEE Transactions on* 21 (2013. szept.), 1805–1817. old. DOI: 10.1109/TASL.2013.2248720.

[12] Jayme Barbedo és George Tzanetakis. "Musical Instrument Classification Using Individual Partials". *Audio, Speech, and Language Processing, IEEE Transactions on* 19 (2011. febr.), 111 –122. old. DOI: 10.1109/TASL.2010.2045186.

[13] Wenxin Jiang és Zbigniew Ras. "Multi-label automatic indexing of music by cascade classifiers". *Web Intelligence and Agent Systems* 11 (2013. ápr.), 149–170. old. DOI: 10.3233/WIA-130268.

[14] Eric J. Humphrey, Juan Pablo Bello és Yann LeCun. "Moving Beyond Feature Design: Deep Architectures and Automatic Feature Learning in Music Informatics." *Proceedings of the 13th International Society for Music Information Retrieval Conference* (Porto, Portugal). Porto, Portugal: ISMIR, 2012. okt., 403–408. old. DOI: 10.5281/zenodo.1415726. URL: https://doi.org/10.5281/zenodo.1415726.

[15] Keunwoo Choi és tsai. "A Tutorial on Deep Learning for Music Information Retrieval". *CoRR* abs/1709.04396 (2017). arXiv: 1709.04396. URL: http://arxiv.org/abs/1709.04396.

[16] Peter Li, Jiyuan Qian és Tian Wang. "Automatic instrument recognition in polyphonic music using convolutional neural networks". *arXiv preprint arXiv:1511.05520* (2015).

[17] Yoonchang Han, Jaehun Kim és Kyogu Lee. "Deep convolutional neural networks for predominant instrument recognition in polyphonic music". *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 25.1 (2016), 208–221. old.

[18] Markus Schedl, Arthur Flexer és Julián Urbano. "The Neglected User in Music Information Retrieval Research". *J. Intell. Inf. Syst.* 41.3 (2013. dec.), 523–539. ISSN: 0925-9902. DOI: 10.1007/s10844-013-0247-6. URL: https://doi.org/10.1007/s10844-013-0247-6.

[19] Markus Schedl, Emilia Gómez és Julián Urbano. "Music Information Retrieval: Recent Developments and Applications". *Foundations and Trends in Information Retrieval* 8 (2014. jan.), 127–261. old. DOI: 10.1561/1500000042.

[20] Dalya Gartzman. *Getting to Know the Mel Spectrogram.* URL: https://towardsdatascience.com/getting-to-know-the-mel-spectrogram-31bca3e2d9d0. Felkeresve: 2020. 05. 09.

[21] Brian McFee és tsai. *librosa/librosa: 0.6.3.* 0.6.3. verzió. 2019. febr. DOI: 10.5281/zenodo.2564164. URL: https://doi.org/10.5281/zenodo.2564164.

[22] J. Stephen Downie. "Music information retrieval". *Annual Review of Information Science and Technology* 37.1 (2003), 295–340. old. DOI: 10.1002/aris.1440370108. eprint: https://asistdl.onlinelibrary.wiley.com/doi/pdf/10.1002/aris.1440370108. URL: https://asistdl.onlinelibrary.wiley.com/doi/abs/10.1002/aris.1440370108.

[23] G. Peeters és tsai. "The timbre toolbox: extracting audio descriptors from musical signals". *Journal of the Acoustical Society of America* 130.5 (2011),

2902–2916. old. DOI: 10.1121/1.3642604. URL: http://eprints.gla.ac.uk/68491/.

[24] Perfecto Herrera, Geoffroy Peeters és Shlomo Dubnov. "Automatic Classification of Musical Instrument Sounds". *Journal of New Music Research* 32 (2010. aug.). DOI: 10.1076/jnmr.32.1.3.16798.

[25] Anssi Klapuri és Manuel Davy. *Signal Processing Methods for Music Transcription*. 2006. jan. DOI: 10.1007/0-387-32845-9.

[26] Justin Salamon és Emilia Gómez. "Melody Extraction from Polyphonic Music Signals using Pitch Contour Characteristics". *IEEE Transactions on Audio, Speech and Language Processing* 20 (2012), 1759–1770. old. URL: http://hdl.handle.net/10230/42183.

[27] Walter B Hewlett és Eleanor Selfridge-Field. *Melodic similarity: Concepts, procedures, and applications, volume 11*. 1998.

[28] Juan Bello és tsai. "A Tutorial on Onset Detection in Music Signals". *Speech and Audio Processing, IEEE Transactions on* 13 (2005. okt.), 1035 –1047. old. DOI: 10.1109/TSA.2005.851998.

[29] F. Gouyon. *Computational Rhythm Description*. VDM Verlag, 2008. ISBN: 978-3836477697.

[30] Martin F. McKinney és Dirk Moelants. "Extracting the perceptual tempo from music". *ISMIR*. 2004.

[31] Gregory H. Wakefield. "Mathematical representation of joint time-chroma distributions". *Advanced Signal Processing Algorithms, Architectures, and Implementations IX*. Szerk. Franklin T. Luk. 3807. köt. International Society for Optics és Photonics. SPIE, 1999, 637 –645. old. DOI: 10.1117/12.367679. URL: https://doi.org/10.1117/12.367679.

[32] Elaine Chew. "Towards a mathematical model of tonality". 2000.

[33] Emilia Gómez. "Tonal Description of Polyphonic Audio for Music Content Processing". *INFORMS J. on Computing* 18.3 (2006. jan.), 294–304. ISSN: 1526-5528. DOI: 10.1287/ijoc.1040.0126. URL: https://doi.org/10.1287/ijoc.1040.0126.

[34] Hélène Papadopoulos és Geoffroy Peeters. "Large-Scale Study of Chord Estimation Algorithms Based on Chroma Representation and HMM". *2007 International Workshop on Content-Based Multimedia Indexing* (2007), 53–60. old.

[35] Laurent Oudre, Yves Grenier és Cédric Févotte. "Template-based Chord Recognition : Influence of the Chord Types." *Proceedings of the 10th International Society for Music Information Retrieval Conference* (Kobe, Japan). Kobe, Japan: ISMIR, 2009. okt., 153–158. old. DOI: 10.5281/zenodo.1414884. URL: https://doi.org/10.5281/zenodo.1414884.

[36] David Temperley. "What's Key for Key? The Krumhansl-Schmuckler Key-Finding Algorithm Reconsidered". *Music Perception: An Interdisciplinary Journal* 17.1 (1999), 65–100. old. DOI: 10.2307/40285812. URL: https://doi.org/10.2307\%2F40285812.

[37] Matthew L. Cooper és Jonathan Foote. "Automatic Music Summarization via Similarity Analysis". *ISMIR*. 2002.

[38] Geoffroy Peeters, Amaury La Burthe és Xavier Rodet. "Toward Automatic Music Audio Summary Generation from Signal Analysis". *In Proc. International Conference on Music Information Retrieval*. 2002, 94–100. old.

[39] Wei Chai. "Semantic Segmentation and Summarization of Music". *IEEE Signal Processing Magazine* (2006. jan.).

[40] Dmitry Bogdanov és tsai. "From Low-Level to High-Level: Comparative Study of Music Similarity Measures". 2009. jan., 453–458. old. DOI: 10.1109/ISM.2009.72.

[41] Michael A. Casey és tsai. *Content-Based Music Information Retrieval: Current Directions and Future Challenges*. 2008.

[42] Markus Schedl és tsai. "Exploring the Music Similarity Space on the Web". *ACM Trans. Inf. Syst.* 29.3 (2011. júl.). ISSN: 1046-8188. DOI: 10.1145/1993036.1993038. URL: https://doi.org/10.1145/1993036.1993038.

[43] Joan Serrà, Emilia Gómez és Perfecto Herrera. "Audio Cover Song Identification and Similarity: Background, Approaches, Evaluation, and Beyond". *Advances in Music Information Retrieval*. Szerk. Zbigniew W. Raś és Alicja A. Wieczorkowska. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, 307–332. old. ISBN: 978-3-642-11674-2. DOI: 10.1007/978-3-642-11674-2_14. URL: https://doi.org/10.1007/978-3-642-11674-2_14.

[44] Thierry Bertin-Mahieux és Daniel Ellis. "Large-scale cover song recognition using the 2D Fourier transform magnitude". (2012. jan.).

[45] Naoko Kosugi és tsai. "A practical query-by-humming system for a large music database". 2000. jan., 333–342. old. DOI: 10.1145/354384.354520.

[46] Justin Salamon, Joan Serrà és Emilia Gómez. "Tonal representations for music retrieval: From version identification to query-by-humming". *International Journal of Multimedia Information Retrieval, special issue on Hybrid Music Information Retrieval* 2 (2013. márc.), 45–58. old. DOI: 10.1007/s13735-012-0026-0.

[47] Roger Dannenberg és tsai. "A comparative evaluation of search techniques for query-by-humming using the MUSART testbed". *JASIST* 58 (2007. márc.), 687–701. old. DOI: 10.1002/asi.20532.

[48] yi-hsuan Yang és Homer Chen. "Machine Recognition of Music Emotion: A Review". *ACM Transactions on Intelligent Systems and Technology (TIST)* 3 (2012. máj.). DOI: 10.1145/2168752.2168754.

[49] Cyril Laurier és tsai. "Indexing music by mood: Design and integration of an automatic content-based annotator". *Multimedia Tools Appl.* 48 (2010. máj.), 161–184. old. DOI: 10.1007/s11042-009-0360-2.

[50] George Tzanetakis és Perry Cook. "Musical Genre Classification of Audio Signals". *IEEE Transactions on Speech and Audio Processing* 10 (2002. jan.), 293–302. old.

[51] Peter Knees, Elias Pampalk és Gerhard Widmer. "Artist Classification with Web-Based Data." 2004. jan.

[52] Perfecto Herrera, Anssi Klapuri és Manuel Davy. "Automatic Classification of Pitched Musical Instrument Sounds". 2006. jan., 163–200. old. DOI: 10.1007/0-387-32845-9_6.

[53] Youngmoo Kim és Brian Whitman. "Singer Identification in Popular Music Recordings Using Voice Coding Features". (2002. szept.).

[54] Mohamed Sordo. "Semantic annotation of music collections: A computational approach". Dissz. 2012. jan.

[55] Emanuele Coviello, Antoni Chan és Gert Lanckriet. "Time Series Models for Semantic Music Annotation". *Audio, Speech, and Language Processing, IEEE Transactions on* 19 (2011. aug.), 1343 –1359. old. DOI: 10.1109/TASL.2010.2090148.

[56] Riccardo Miotto és Nicola Orio. "A Probabilistic Model to Combine Tags and Acoustic Similarity for Music Retrieval". *ACM Transactions on Information Systems - TOIS* 30 (2012. máj.), 1–29. old. DOI: 10.1145/2180868.2180870.

[57] Xinxi Wang, David Rosenblum és Ye Wang. "Context-Aware Mobile Music Recommendation for Daily Activities". 2012. okt. DOI: 10.1145/2393347.2393368.

[58] Pedro Cano és tsai. "Audio Fingerprinting: Concepts And Applications". 2. köt. 2005. szept., 233–245. old. DOI: 10.1007/10966518_17.

[59] Michael Casey és tsai. "Content-Based Music Information Retrieval: Current Directions and Future Challenges". *Proceedings of the IEEE* 96 (2008. máj.), 668 –696. old. DOI: 10.1109/JPROC.2008.916370.

[60] Oscar Celma. *Music Recommendation and Discovery - The Long Tail, Long Fail, and Long Play in the Digital Music Space.* 2010. jan. ISBN: 978-3-642-13286-5. DOI: 10.1007/978-3-642-13287-2.

[61] Yuan Zhang és tsai. "Auralist: Introducing serendipity into music recommendation". 2012. febr., 13–22. old. DOI: 10.1145/2124295.2124300.

[62] Marius Kaminskas, Francesco Ricci és Markus Schedl. "Location-aware music recommendation using auto-tagging and hybrid matching". 2013. okt., 17–24. old. DOI: 10.1145/2507157.2507180.

[63] Tim Pohle és tsai. ""Reinventing the Wheel": A Novel Approach to Music Player Interfaces". *Multimedia, IEEE Transactions on* 9 (2007. máj.), 567 –575. old. DOI: 10.1109/TMM.2006.887991.

[64] Gordon Reynolds és tsai. "Towards a Personal Automatic Music Playlist Generation Algorithm: The Need for Contextual Information". (2007. jan.).

[65] Elias Pampalk, Tim Pohle és Gerhard Widmer. "Dynamic Playlist Generation Based on Skipping Behavior." 2005. jan., 634–637. old.

[66] J.-J Aucouturier és Francois Pachet. "Scaling up music playlist generation". 26. köt. 2002. febr., 105 –108 vol.1. ISBN: 0-7803-7304-9. DOI: 10.1109/ICME.2002.1035729.

[67] Dixon, Simon Lui és Gerhard Widmer. "MATCH: A Music Alignment Tool Chest". 2005. szept.

[68] Meinard Müller, Henning Mattes és Frank Kurth. "An Efficient Multiscale Approach to Audio Synchronization". 2006. okt.

[69] Bernhard Niedermayer és Gerhard Widmer. "A Multi-Pass Algorithm for Accurate Audio-to-Score Alignment". 2010. dec., 417–422. old.

[70] Markus Schedl és tsai. "What's Hot? Estimating Country-specific Artist Popularity." 2010. jan., 117–122. old.

[71] Francois Pachet és Pierre Roy. "Hit Song Science Is Not Yet a Science." 2008. jan., 355–360. old.

[72] Noam Koenigstein és Yuval Shavitt. "Song Ranking based on Piracy in Peer-to-Peer Networks." 2009. jan., 633–638. old.

[73] Meinard Müller és Nanzhu Jiang. "A scape plot representation for visualizing repetitive structures of music recordings". *Proceedings of the 13th International Society for Music Information Retrieval Conference, ISMIR 2012* (2012. jan.).

[74] Arpi Mardirossian és Elaine Chew. "Visualizing Music: Tonal Progressions and Distributions." 2007. jan., 189–194. old.

[75] Matthew Cooper és tsai. "Visualization in Audio-Based Music Information Retrieval". *Computer Music Journal* 30 (2006. jún.), 42–62. old. DOI: 10.1162/comj.2006.30.2.42.

[76] Jonathan Foote. "Visualizing Music and Audio using Self-Similarity". 1999. jan., 77–80. old. DOI: 10.1145/319463.319472.

[77] Emilia Gómez és Jordi Bonada. "Tonality visualization of polyphonic audio". (2005. jan.).

[78] Sebastian Stober és Andreas Nürnberger. "MusicGalaxy: A Multi-focus Zoomable Interface for Multi-facet Exploration of Music Collections". 6684. köt. 2010. jún., 273–302. old. DOI: 10.1007/978-3-642-23126-1_18.

[79] Stefan Leitich és Martin Topf. "Globe of Music - Music Library Visualization Using Geosom." 2007. jan., 167–170. old.

[80] Paul Lamere és Douglas Eck. "Using 3D Visualizations to Explore and Discover Music." 2007. jan., 173–174. old.

[81] Elias Pampalk és Masataka Goto. "MusicRainbow: A New User Interface to Discover Artists Using Audio-based Similarity and Web-based Labeling." 2006. jan., 367–370. old.

[82] Rebecca Stewart és Mark Sandler. "The amblr: A mobile spatial audio music browser". 2011. júl., 1–6. old. DOI: 10.1109/ICME.2011.6012203.

[83] Markus Schedl és Dominik Schnitzer. "Hybrid retrieval approaches to geospatial music recommendation". 2013. júl., 793–796. old. DOI: 10.1145/2484028.2484146.

[84] Sebastian Stober. "Adaptive Methods for User-Centered Organization of Music Collections". Dissz. 2011. nov.

[85] Marius Kaminskas, Francesco Ricci és Markus Schedl. "Location-aware music recommendation using auto-tagging and hybrid matching". 2013. okt., 17–24. old. DOI: 10.1145/2507157.2507180.

[86] Linas Baltrunas és tsai. "InCarMusic: Context-Aware Music Recommendations in a Car". 85. köt. 2011. aug., 89–100. old. DOI: 10.1007/978-3-642-23014-1_8.

[87] Dr. Michael J. Garbade. *Clearing the Confusion: AI vs Machine Learning vs Deep Learning Differences.* URL: https://towardsdatascience.com/clearing-the-confusion-ai-vs-machine-learning-vs-deep-learning-differences-fce69b21d5eb. Felkeresve: 2020. 05. 04.

[88] *Gépi tanulási algoritmusok.* URL: https://azure.microsoft.com/hu-hu/overview/machine-learning-algorithms/\#popular-algorithms. Felkeresve: 2020. 05. 16.

[89] Josh Patterson és Adam Gibson. *Deep Learning: A Practitioner's Approach.* Beijing: O'Reilly, 2017. ISBN: 978-1-4919-1425-0. URL: https://www.safaribooksonline.com/library/view/deep-learning/9781491924570/.

[90] *Mélyreható tanulás és gépi tanulás.* URL: https://docs.microsoft.com/hu-hu/azure/machine-learning/concept-deep-learning-vs-machine-learning. Felkeresve: 2020. 05. 26.

[91] ISMIR community. *ISMIR list of datasets.* URL: http://www.ismir.net/resources/datasets/. Felkeresve: 2020. 05. 10.

[92] Eric Humphrey, Simon Durand és Brian McFee. "OpenMIC-2018: An Open Data-set for Multiple Instrument Recognition." *ISMIR.* 2018, 438–444. old.

[93] Kirell Benzi és tsai. "FMA: A Dataset For Music Analysis". *CoRR* abs/1612.01840 (2016). arXiv: 1612.01840. URL: http://arxiv.org/abs/1612.01840.

[94] Karen Simonyan és Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". *arXiv preprint arXiv:1409.1556* (2014).

[95] *VGGish.* URL: https://github.com/tensorflow/models/tree/master/research/audioset/vggish. Felkeresve: 2020. 05. 14.

[96] Yann A. LeCun és tsai. "Efficient BackProp". *Neural Networks: Tricks of the Trade: Second Edition.* Szerk. Grégoire Montavon, Geneviève B. Orr és Klaus-Robert Müller. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, 9–48. old. ISBN: 978-3-642-35289-8. DOI: 10.1007/978-3-642-35289-8_3. URL: https://doi.org/10.1007/978-3-642-35289-8_3.

[97] Tara Boyle. *Dealing with Imbalanced Data.* URL: https://towardsdatascience.com/methods-for-dealing-with-imbalanced-data-5b761be45a18. Felkeresve: 2020. 05. 13.

[98] Tarang Shah. *About Train, Validation and Test Sets in Machine Learning.* URL: https://towardsdatascience.com/train-validation-and-test-sets-72cb40cba9e7. Felkeresve: 2020. 05. 13.

[99] Manolis Loukadakis, José Cano és Michael O'Boyle. "Accelerating Deep Neural Networks on Low Power Heterogeneous Architectures". 2018. jan.

[100] Christopher Riggio. *What's the deal with Accuracy, Precision, Recall and F1?* URL: https://towardsdatascience.com/whats-the-deal-with-accuracy-precision-recall-and-f1-f5d8b4db1021. Felkeresve: 2020. 05. 11.