XXI. reál- és humántudományi Erdélyi Tudományos Diákköri Konferencia (ETDK)
Kolozsvár, 2018. május 24–27.

Birdify

Madárhangfelismerés konvolúciós neurális háló segítségével

Szerzők:

Incze Ágnes
Babeş–Bolyai Tudományegyetem, Matematika és Informatika Kar, Informatika szak, III. év
Jancsó Henrietta-Bernadett
Babeş–Bolyai Tudományegyetem, Matematika és Informatika Kar, Informatika szak, III. év

Témavezetők:

Szilágyi Zoltán, szoftverfejlesztő,
Codespring
Farkas Attila, szoftverfejlesztő,
Codespring
Sulyok Csaba, doktorandusz
Babeş–Bolyai Tudományegyetem, Matematika és Informatika Kar

CODESPRING
OUR CODE YOUR SUCCESS

Kivonat

A konvolúciós neurális háló egy gépi tanulásban alkalmazott módszer, mely igen hatékony a képfeldolgozási problémák megoldásában. Jelen dolgozat egy osztályozási feladatatot taglaló rendszer leírása, amely egy madárhangról képes dönteni, hogy melyik fajhoz tartozik.

A feladat megoldására több megközelítést is tesztel a program. Az alkalmazott alapmodell a MobileNet nevű konvolúciós neurális háló, a korpuszt a xeno-canto oldalon található hangok egy része képezi, a háló bemenetei pedig az előfeldolgozott hangokból kinyert spektrogrammok. A változatok, amelyek alapján a háló tanul, a következők: a bemeneti osztályok számossága, az adatok előfeldolgozásában alkalmazott színskála típusa, illetve a hiperparaméterek, amelyek jellemzik a hálót.

A legjobb eredményekkel szolgáló hálót egy RESTful webszolgáltatás révén tesszük elérhetővé. Ezen keresztül kommunikál egy Android mobil alkalmazás.

Tartalomjegyzék

Bevezető

Az utóbbi években a számítógépes hangfelismerés nagy népszerűségnek örvend a mélytanulás iránt érdeklődők körében [18], [3]. Ezt a típusú osztályozást felhasználó széleskörű szakterületek közül a mi kutatásunk a madárhangfelismerésre összpontosít. Célközönségünk körébe tartoznak a természetkedvelők, illetve ornitológusok, akik számára szeretnénk megkönnyíteni és felgyorsítani a madárfajok megkülönböztetését egymástól a madarak hangjai alapján.

A jelenlegi kutatás előtanított hálókat [21] használ, ezáltal jelentős időt takarítva meg a kísérletek végrehajtása során, amelyek madárhangok felismerésére, osztályozására öszpontosulnak. Sok előretanított háló arra van betanítva, hogy általános vonásokat ismerjen fel a képekben. A hangok egy dimenzióban vannak reprezentálva, ezzel szemben a képek kétdimenziós jelek, ezért egy reprezentatív transzformációra van szükség a kompatibilás miatt. Erre szolgál a spektrogramm, ami egy vizuális reprezentációja a rövid idejű Fourier-transzformáció (Short Time Fourier Transform, röviden STFT)[1] amplitúdójának. Az STFT egyféle diszkrét Fourier-transzformáció (DFT), ami ahelyett, hogy csak egyszer végezné el a jel egészére a DFT-t, a jelet felosztja valamennyi átfedéssel rendelkező darabokra és ezekre külön elvégzi a DFT-t. Ebből kifolyólag az eredmény egy kétdimenziós reprezentációja lesz egy hangdarabnak, ahol az idő és a frekvencia felel meg a tengelyeknek. A spektrogram egy színtérkép segítségével ábrázolja képként az STFT értékeit, amely bemenetként szolgálhat egy képekre épített előre tanított háló számára.

Mi a kompakt és sebességorientált orientált MobileNetet[6] használjuk, mint előretanított háló. A kísérleteink során egy összehasonlítást végzünk az osztályok számáról és a színtérképről, amit a spektrogrammok előállítása során használunk.

Biztosítunk egy Android alkalmazást is, annak érdekében, hogy az elméletet gyakorlatba helyezzük. Az Android kliens a szerverrel kommunikál, amely a madár, hangja alapján történő, osztályzását végzi. A kliens és a szerver közti kapcsolat HTTP-n keresztül történik.

Kutatások folytak a közelmúltban az említett probléma megközelítése szempontjából. Például 2017-ben, Kahl et al. [8] 1500 faj osztályozásáról ír, nemcsak a leghangosabb madarat megkeresve, hanem az összes hallhatót osztályozza. A legjöbb eredményük 60% pontosság több madár osztályozásakor és 68% a legdominánsabb madár klasszifikálása esetén. Az adathalmazuk egy előre elkészített adathalmaz (BirdCLEF 2017), ami 36.496 hangfájlból áll, szintén a Xeno-cantoról gyűjtve, minden osztályban különböző mennyiségű hang van. Az általunk készített szkript a begyűjtés során minden osztályhoz ugyanannyi hangot tölt le. A megközelítésük nagyban hasonlít a miénkhez abban, hogy a hangokat spektrogrammokként reprezentálják, és az osztályozásra konvolúciós neurális hálót használnak. Ők azonban felépítették és betanították a saját hálójukat, míg mi egy előretanított CNN-t használunk, a MobileNetet.

Piczak [16] 2016-ban hasonló megközelítéssel 41,2%-os pontosságot ért el a több címkével ellátott hanganyagok osztályozása, illetve 52,9%-ot a felvételen levő domináns hang osztályozása esetén. Szintén CNN hálókkal dolgozott, melynek bemenetei mel-frekvencián vett cepstrumok voltak. Ezzel szemben mi általános spektrogrammokat használtunk. Piczak módszerében megjelenik egy maximális frekvencia határ a zajok szűrésére, amit mi nem alkalmaztunk az előfeldolgozás során.

Egy másik megközelítés 2016-ban Sprengel [19] publikációjában volt tárgyalva, amely megnyerte a BirdCLEF 2016-os Recognition Challenge-ét a 68,6%-os illetve 55,5%-os pontosságok elérésével az egy illetve a több címkével ellátott osztályozások esetén. Az általuk használt öt konvolúciós réteggel és egy sűrű réteggel rendelkező CNN háló bemeneteként spektrogrammokat adtak meg, amely előzőleg kivágta zajokat és csak a madárhangokat tartotta meg.

A bemutatott módszerek alapján a mi módszertanunk is a hangok képekkénti ábrázolásán alapszik, amelyeket egy CNN háló bemeneteként használunk fel. 2015-ben Piczak et al. [15] javaslata alapján a spektrogrammok használata a hangok CNN-el való osztályozásában előnyösebb a hangok mel-frekvencia cepstrum reprezentációjánál. Így mi is a felcímkézett spektrogrammokat tekintettünk a háló bemeneteként.

Jelen dolgozatban bemutatott projekt a Codespring mentorprogram és nyári gyakorlat alatt jött létre, ezért külön köszönetben részesítjük a mentorainkat, Farkas Attilát, Szilágyi Zoltánt és Sulyok Csabát, és egyetemi kollégáinkat, akik a csoportos projekt tantárgy keretein belül közreműködtek: Bóné Norbert-Máté, Golicza Lukács, Kapros Regina illetve Popoviciu Brigitta.

A dolgozat a következőket foglalja magába: egy elméleti áttekintést a konvolúciós neurális hálók, illetve az hangok előfeldolgozása terén, amely során tisztázzuk a módszerben kulcsfontosságú fogalmakat. Az elméleti áttekintés után bevezetjük az általunk alkalmazott módszertant, mely során részletesen tárgyaljuk a felhasznált korpusz felépítését, forrását és begyűjtésének lépéseit. Ezután részletezzük a korpuszon elvégzett előfeldolgozás folyamatát, majd bevezetjük az általunk használt konvolúciós neurális háló (a MobileNet) jellemzőit, felépítését. Tárgyaljuk a MobileNettel végzett tanítás folyamatát, ezzel lezárva a módszertani jellemzők bemutatását. Nem utolsó sorban a rendszer architektúrájáról lesz szó, azaz a szerver oldal funkcióit tárgyaljuk, illetve az ehhez kapcsolódó Android klienst mutatjuk be. Utolsó lépés gyanánt bemutatjuk a kísérleteket és az ezekből származó eredményeket, és részletezzük a következtetéseket, továbbfejlesztési lehetőségeket ajánlva a jobb eredmények érdekében.

1. Elméleti áttekintés

A következőkben tárgyalni fogjuk a konvolúciós neurális hálók elméleti hátterét, mely során tisztázunk olyan fogalmakat, mint például az osztályozás, háló tanítása, veszteségfüggvény, hálók felépítése. Ezek alapján fogjuk elemezni az általunk elvégzett kísérleteket, módszereket és eredményeket.

1.1. Neurális hálók és konvolucios neurális hálók

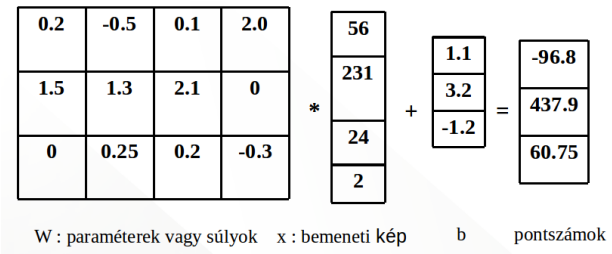
A jelfeldolgozásban hatékonynak bizonyuló mesterséges neurális hálók többek között beszédfelismerés, írott betű felismerése és arcfelismerés terén értek el különösen jó eredményeket. Ezek a típusú problémák bonyolultabb osztályozási feladatokként is tekinthetők és nagy mértékben foglalkoztatják a mélytanulás felől érdeklődőket. Az interneten számos információ megtalálható erről a témakörről, ezek közül a Stanford CS231n konvolucios neurális háló tematikájú kurzusa [12] kiemelkedő jó minősége és szerkezete révén. A következőkben a konvolucios neurális hálók szerkezetét és működését tárgyaljuk részletesebben.

A legismertebb osztályozási feladat a képfelismeréssel köthető össze. Két ilyen probléma a MNIST és a CIFAR-10 osztályozás. Ezek közül a MNIST hetvenezer kézzel írt, 0 és 9 közötti számjegy kötött méretű fekete-fehér képeit tartalmazza. A CIFAR-10 hatvanezer, szintén kötött méretű színes képet tartalmaz, tízezret minden osztály esetében, amelyek a következők: repülőgép, autó, madár, macska, őz, kutya, béka, ló, hajó, teherautó. A MNIST adathalmaz esetén az osztályozási feladat a következő: egy kézzel írott számjegyről készült kép alapján a gép tudja eldönteni, hogy melyik számjegyről van szó. Értelemszerűen a CIFAR-10 esetén hasonló a probléma, csak nem számjegyeket kell kategorizálnia, hanem a felsorolt tíz osztály bármelyikéről készült képet kell felismernie.

Több osztályozási módszerrel közelítették meg a fent említett problémákat, ezek közül a konvolúciós neurális hálók voltak a leghatékonyabbak. A konvolúciós neurális hálók a mesterséges neurális hálóknak egy továbbfejlesztett változata. A mesterséges neurális hálók működését a biológiában ismert neurális hálók viselkedését vették alapul.

A mesterséges neurális hálók megalkotása során a biológiából ismert neurális hálók viselkedését vették alapul. A tanítás eredménye egy modell lesz, amit ismeretlen adatokat minél pontosabban címkéz fel. Ezt a folyamatot tanítási folyamatnak szokták nevezni.

A fent említettekből kikövetkeztethető, hogy egy osztályozási feladat megoldásának kulcsfontosságú része a nagy mennyiségű felcímkézett adathalmaz. Felcímkézett adathalmazon olyan címke-adat párokat értünk mint például a CIFAR-10 esetén egy kép a felsorolt tíz elem egyikéről és a hozzá tartozó osztály megnevezése (illetve a neki megfelelő kód). Az adathalmaz mellett még két fontos függvény létezik, ezek az értékelő függvény és a veszteségfüggvény. Az



1. ábra. Lineáris klasszifikáció. W a súlymátrix, x a kép, b az eltolás. A képet kinyújtja egy vektor formájába, az eredményeknél a nagyobb pontszám nagyobb valószínűséget jelent.

értékelő függvény egy olyan függvény, amely egy ismeretlen adatot leképez egy n elemű pontérték tömbre, ahol n az osztályok száma. A veszteségfüggvény feladata, hogy a már említett felcímkézett adatok ismeretében egy számban kifejezze az értékelő függvény hibáját. A tanítási folyamat ilyen módon több iterációból tevődik össze. Egy iteráció a következő lépéseket foglalja magába: az adatok kiértékelését az értékelő függvény segítségével, az így kapott modell hibájának meghatározását az eredmények alapján a veszteségfüggvény segítségével, illetve modell frissítésének folyamatát. A modell frissítése akkor sikeres, ha a veszteségfüggvény által számított veszteség az iterációk során csökken. Ezt nevezik optimalizálásnak is. Ezt a típusú tanítási módszert a lineáris osztályozók alkalmazzák és ebből indulnak ki a neurális hálók is. A lineáris klasszifikáló matematikai alakja a következőképpen néz ki:

$$f(x) = W * x + b$$

A fenti képletben az ismeretlen x az osztályozandó elemet jelenti, a CIFAR-10 esetében egy képet. A bemeneti adat általában egyetlen vektor. Képek esetén a háromdimenziós RGB-kódolt mátrixot egyetlen vektorba nyújtják ki. A W és b pedig a paraméterek, amelyek eleinte véletlenszerűen vannak meghatározva, de az optimalizálás során változnak lehetőleg úgy, hogy a veszteség csökkenjen. A W paraméter egy súlymátrix, a b egy vektor, tehát a klasszifikálás szerepét egyszerű mátrix szorzás és összeadás fogja felvenni. A súlymátrix kulcsfontosságú szerepet játszik benne. Annyi sora van, ahány osztályra tanítjuk a hálót, és annyi oszlopa, ahány eleme van a bemeneti x adatnak. A szorzat és összeg végeredménye egy vektor, amely elemeinek a száma megegyezik az osztályok számával. Az eredményként kapott vektor a pontszámokat adja meg, hogy melyik osztályhoz milyen mértékben sorolható a bemeneti adat. Az 1 ábrán egy olyan lineáris klasszifikáló látható, mely három osztályt ismer. A példa kedvéért a bemeneti adat csak négy elemű vektor, ez a gyakorlatban sokkal nagyobb szokott lenni.

A veszteségfüggvény szerepe, hogy meghatározza egy modellről, hogy mennyire téves. A modell elemei, amelyek a tanulási folyamat befolyása alatt vannak, azok a már említett súlymátrixok, szűrők és más paraméterek.

A két legismertebb osztályozó, amelynek a kimenetéből veszteséget számolunk, az SVM (Multiclass Support Vector Machine) és a Softmax. A gyakorlatban a két függvény hason-

lóan jó eredményekkel szolgál, de eltérő okokból. Az SVM-et önmagában hibásnak tekintik, de hibáját úgynevezett regularizációs folyamattal ellensúlyozzák. A Softmax függvénynek egy N dimenziós vektor a bemeneti paramétere amiből létrehoz egy szintén N dimenziós vektort, amely értékei 0-1 intervallumban vannak és elemeinek összege 1. A Softmax eredményének egy interpretációja lenne a következő: a legmagasabb értékhez tartozó osztályba csoportosítható a legnagyobb valószínűséggel a bemeneti kép.

Az optimalizáció egy olyan folyamat, amely során a tanítandó háló paramétereit oly módon változtatjuk, hogy a veszteség minél kisebb legyen. Több módszer van, amely ezt a feladatot hivatott elvégezni, a legismertebb ezek közül a Gradient Descent. Alapja, hogy a hálóban elvégzett műveletek ismeretében minden tanulási iteráció végén „back propagation" folyamatot hajt végre és az így kapott gradienseket beszorozva egy tanulási ráta (lépés méret) hiperparaméterrel (step size, learning rate), kivonja a nekik megfelelő paraméterekből.

A nagy méretű adathalmazok esetén nem szokták az egész adathalmazra elvégezni az optimalizációs folyamatot a paraméter frissítés érdekében, mert túl sok számítást jelentene. Ezt a problémát kiküszöböli a Mini-batch Gradient Descent, amely ahelyett, hogy a teljes adathalmazt használná fel, mindig csak konstans méretű, random kiválasztott részét az adathalmaznak dolgozza fel. Gyakran a feldolgozott adatmennyiség egy iterációban 32, 64, 256 méretek közül az egyiket veszi fel. Más optimalizálók is adaptálták ezt a megközelítést.

A mesterséges neurális hálók felépítése bonyolultabb a lineáris klasszifikálónál. Több paraméterrel rendelkezik és működésükben fontos szerepet töltenek be az aktivációs függvények. Felépítése réteges, a különböző rétegek lineárisan kapcsolódnak egymáshoz. A szomszédos rétegek bemeneti és kimeneti adatok által kommunikálnak egymással. Egy példa kétrétegű neurális hálóra matematikailag felírva:

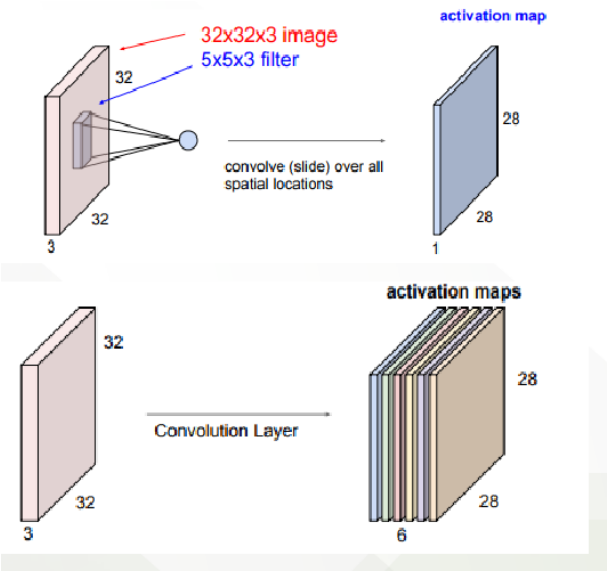
$$f(x) = W_2 * max(0, W_1 * x + b)$$

A $max(0, x)$ függvény egy aktivációs függvény, illetve a képletben felfedezhető mátrixszorzások lineáris klasszifikálókra emlékeztetnek. A képletben megjelenő aktivációs függvény a ReLU (Rectified Linear Units) [22].

A neurális hálóknál a mintafelismerés terén hatékonyabb vátozata a konvolúciós neurális hálók, melyek, habár több komputacionális erőforrást igényelnek, mégis előnyösebb modellt szolgáltatnak mint a hagyományos neurális hálók. A konvolúciós neurális hálók tipikus rétegei:

1. Konvolucios réteg

   Ez a típusú réteg szintén mátrixszorzásokon alapszik, viszont jelen esetben a bemeneti adatot nem vektor formájában tároljuk, hanem megtartja eredeti dimenzióit. Például a CIFAR-10 adathalmaz esetén a bemeneti adat (kép) dimenziója 32x32x3. A súlymátrixok



2. ábra. Konvolúciós réteg.

szerepét felveszik a filterek (szűrők), amelyek hosszúságban és szélességben kisebbek a bemeneti adatoknál, de mélységben megtartják a dimenziójukat. Egy szűrő méretére jó példa CIFAR-10 esetén a 2x2x3-as dimenziójú mátrix, vagy akár az 1x1x3-as is.

Ezen a rétegen belül lehet egy vagy több filter is. Minden filter a következő folyamatot hajtja végre: a bemeneti adat különböző részeihez illeszkedve elvégződik a mátrix szorzás, az eredmény elmentődik az illesztett helynek megfelelő pozícióra egy úgynevezett aktivációs térképen. Minden filterrel végzett konvolúció eredménye egy-egy aktivációs térkép. Ezt a folyamatot a 2 [1] ábrán lehet megfigyelni.

2. ReLU vagy más aktivációs függvény

   Az aktivációs függvény feladata, hogy eldöntse melyik neuron aktív és melyik inaktív. Neuronnak nevezik a neurális háló legkisebb, számításokat végző, elemét amely kimenete a skálázott és eltolt bemenet (azaz az osztályozás kimenete). Szintén a biológiából ismert neuronok működésén alapszik, viszont sokkal egyszerűbb a működése. A legismertebb aktivációs függvény a Sigmoid ($y = 1/(1 + e^{-}x)$), ma viszont csupán történelmi szempontból jelentős. Elterjedt a ReLU[22] használata ($y = max(0, x)$) mert nem követel meg bonyolult számítások elvégzését és gyakran nagyon jó eredményekkel szolgál.

3. Tömörítő réteg (Pooling layer)

   Leredukálja a bemeneti mátrix méretét szélességben és hosszúságban. Létezi átlagoló- illetve max-tömörítő réteg. A bemenetet meghatározott, azonos méretű részekre darabolja,

[1] Ábra forrása: https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/convolutional_neural_networks.html, utolsó megtekintés dátuma: 2018.04.22

és adott feltételek alapján, minden részből egyetlen elemet tart meg (max-pooling esetén a legnagyobb értéket) vagy új elemet hoz létre (average pooling esetén az elemek átlagát).

4. Teljesen összekötött réteg (FC, Fully Connected layer)

   A konvolúciós rétegtől eltérően a bemeneti adatot nem osztja fel, hanem egyetlen komponensnek tekinti. Megegyezik a neurális hálókban is alkalmazott lineáris klasszifikációval.

5. Batch Normalization Az Batch Normalization [7] réteg egy közismert megoldás a tanítási folyamat felgyorsítására. Az elméleti bevezetésben érintettük, hogy bevett szokás tanítás során az adathalmazt kötegekre (batch) bontani. Ez, bár kevesebb adaton kell elvégezni a nagy mennyiségű számításokat, mégis megköveteli, hogy a tanulási ráta hiperparaméter kisebb legyen, így az optimalizálás lassan csökken. Ezt a problémát oldja meg a Batch Normalization, mely során minden köteg normalizálódik. Ez nemcsak nagyobb tanulási rátát enged meg, de használatával a paraméterek inicializálása kevésbé lesz fontos.

Néhány neves konvolúciós neurális háló: AlexNet [9], GoogLeNet [20], Microsoft ResNet [5]. Az AlexNet Krizhevsky, Sutskever és Hinton fejlesztésével jött létre. Ez volt az első olyan háló, amely jelentősen jó eredményt ért el az ImageNet adathalmazra. A GoogLeNet egy 22 rétegű konvolúciós neurális háló, amely nem használ egyáltalán FC-réteget. A Microsotf ResNet 152 réteggel rendelkezik, és elért 3,6% hibaarányt, ami azért jelentős, mert az emberek esetében ez átlagosan 5-10%.

1.2. Kép- és hangfeldolgozás

A feldolgozás során az adatok, akár kép, akár hang esetében, digitális jelként foghatóak fel. A hangok esetében, a felvétel után a hangot számok sorozataként reprezentáljuk. A képeket hasonlóképpen, azonban ez kétdimenziós formában kerül tárolásra.

A hangokon, ahhoz, hogy képet – vagyis spektrogrammot – nyerhessünk belőlük, rövid idejű Fourier-transformációt (Short Time Fourier Transformation, röviden STFT) hajtunk végre, ezt valamilyen színskálán ábrázolva lesz belőle kép. A rövid idejű Fourier-transzformáció megértéséhez érdemes előbb a diszkrét Fourier -transzformációt taglalnunk.

A digitális jelek esetében diszkrét időről beszélünk, azaz az idő fogalma ebben a kontextusban két egymást követő mintavétel között eltelt időt jelenti. Ugyanígy diszkrét amplitúdóról beszélünk digitális jelek esetén, amit egész számokkal (integer) reprezentálunk, ami azért is fontos, mert kisebb helyen lehet tárolni, könyebb feldolgozni és tovább küldeni. A diszkrét idő fogalmát használva nem beszélhetünk valós számmal jellemezhető pontokról az idő-tengelyen, hanem egy index, egy egész szám, fogja jellemezni a mintavétel pillanatát (hányadik minta). A periódus pedig azt jelenti, hogy hány mintavétel (sample) után ismétlődik a minta (pattern). A

fizika világában a periódus azt jelenti, hogy mennyi idő telt el egy adott jelenség ismétlődéséig. Mértékegysége a másodperc. A frekvencia, ezzel szemben, azt méri, hogy egységnyi idő alatt hányszor ismétlődik az adott jelenség. Mértékegysége a Hertz (Hz), ami definíció szerint az s^{-1} mértékegységet jelenti. Legyen T_s másodpercben kifejezett idő két mintavétel között, a periódus M minta (valós világban $M * T_s$), az analóg vagy valós világbeli frekvencia képlete pedig: $f = \frac{1}{MT_s} Hz$. Továbbá F_s a mintavételek száma egy másodperc alatt, így $T_s = \frac{1}{F_s}$.

Példa: $F_s = 48000$, $T_s \approx 20.8\mu s$. Ha $M = 110$, akkor $f \approx 440Hz$.

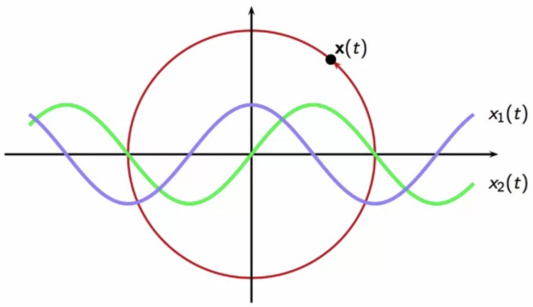
A Fourier-analízis célja[14], hogy bármilyen jel kifejezhető legyen szinuszoid komponensekből, vagyis szinuszokból és koszinuszokból, $x_1(t) = cos(ft)$ és $x_2(t) = sin(ft)$, ahogy ez az 3 [2] ábrán is látható. Ez azért hasznos, mert az Euler-képlet szerint:

$$e^{j\alpha} = \cos\alpha + j\sin\alpha$$

. A matematikailag véges hosszúságú jeleket veszünk (vektorok). A Fourier-analízis egy egyszerű báziscsere, ami felfogható egy perspektívacserének is, és a megváltozott perspektíva elárulhatja erről 4a [3] az azonosítatlan jelről, hogy milyen frekvenciák találhatóak benne, lásd 4b [4] ábrát.

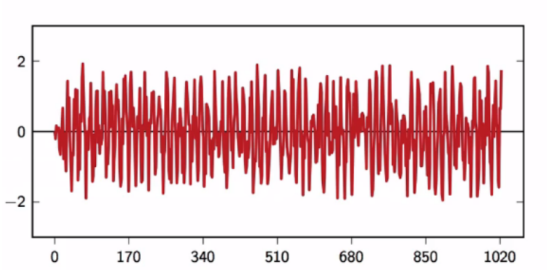
A diszkrét Fourier-transzformáció (DFT)[14] képlete, analízis:

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi}{N}nk}, k = 0, 1, ..., N-1$$

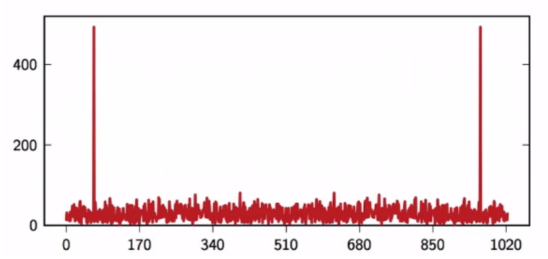


3. ábra. Egy komplex számrendszerben körülírt jel (x) szinuszoid (x_2) és koszinusoid (x_1) elemei egymásnak komplementerei: amplitúdójuk és frekvenciájuk közös, míg 90 fokos fáziseltolással rendelkeznek.

[2] Ábra forrása: https://bit.ly/2qRmrf6, utolsó megtekintés dátuma: 2018.04.22
[3] Ábra forrása: https://bit.ly/2qT07Sd , utolsó megtekintés dátuma: 2018.04.22
[4] Ábra forrása: https://bit.ly/2qT07Sd , utolsó megtekintés dátuma: 2018.04.22



(a) A jel DFT előtt, a vízszintes tengely az idő.



(b) A jel DFT után, a vízszintes tengely a frekvencia.

4. ábra. Egy azonosítatlan jelre alkalmazott Fourier-transzformáció

N pontból álló jel a frekvencia doméniumban. Szintézis képlete:

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{j\frac{2\pi}{N}nk}, n = 0, 1, ..., N-1$$

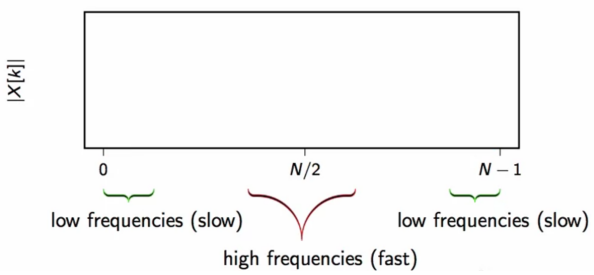
N pontból álló jel az idő doméniumban. Ugyanezek vektornotációval, analízis: $X_k = \langle w^{(k)}, x \rangle$, szintézis:

$$x = \frac{1}{N} \sum_{k=0}^{N-1} X_k w^{(k)}$$

Egy DFT kirajzolt változatának interpretációja során érdemes megjegyezni a következőket: az ábra 0 és N-1 közti frekvencia együtthatókat tartalmaz, 0-tól N/2-ig tartalmazza a π-nél kisebb frekvenciát (óramutatóval ellentétes mozgású), N/2-től N-1-ig pedig a π-nél nagyobb frekvenciákat (óramutatóval megegyező mozgás).

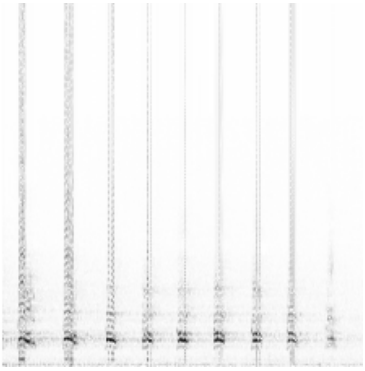
Ugyanakkor az 5 [5] ábrán látható, hogy az ábra szimmetrikus, a szimmetria tengely pedig az N/2-nél található, ez azt is jelenti, hogy az ábra két szélén találhatóak az alacsony frekvenciák és az N/2 körül találhatóak a magas frekvenciák. Ahhoz, hogy meg lehessen feleltetni egy frekvenciát a vízszintes tengelynek, elégséges, ha tudjuk a rendszer órajelét, T_s, ebből kiszámolható a frekvencia, mert ismerjük a következőket:

1. leggyorsabb pozitív frekvencia $\omega = \pi$



5. ábra. DFT eredmény interpetációja. Mivel az ábra szimmetrikus N/2 körül, az N/2-től az N-1-ig tartó részek figyelmen kivül hagyhatóak.

[5] Ábra forrása: https://bit.ly/2HllQN8, utolsó megtekintés dátuma: 2018.04.22



6. ábra. Madárhang spektrogrammja. A vízszintes az idősík, a függőleges a frekvencia. Az ismétlődő minták az időtengely mentén a madár csiripelő hangjának mintáit tükrözik.

2. az $\omega = \pi$ szinuszoidnak 2 minta kell hogy egy teljes körbefordulást tegyen
3. minták közti idő: $T_s = \frac{1}{F_s}$ másodperc
4. valós analóg periódusa a leggyorsabb szinuszoidnak: $2T_s$ másodperc
5. valós analóg frekvencia a leggyorsabb szinuszoidnak: $\frac{F_s}{2}$ Hz

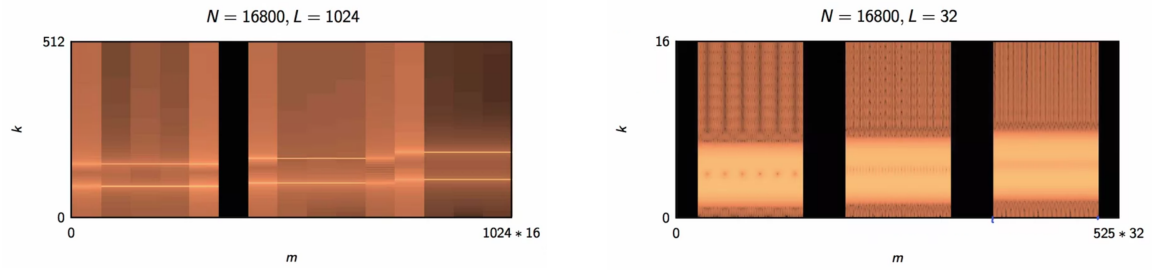
A rövid idejű Fourier-transzformáció (angolul Short Time Fourier Transform, röviden STFT) ötlete, hogy L darab kisebb részre ossza a jelet, és ezekre külön-külön elvégzi a DFT-t.

$$X[m;k] = \sum_{n=0}^{L-1} x[m+n]e^{-j\frac{2\pi}{L}nk}$$

A fenti képlet alapján, az STFT-nek a kimenete nem egydimenziós, hanem kétdimenziós mátrixként fogható fel, ugyanakkor komplex számok fognak szerepelni ebben a mátrixban. A komplex számok valós része adja meg az amplitúdót és az imaginárius rész a fázist (phase). A darabok mérete egy választott ablakmérettől függ (window size), ez lesz a mátrix egyik mérete. Egy másik paraméter az átfedés (overlap), azt határozza meg, hogy mennyi legyen az előző és a mostani ablak közti átfedés, amire végrehajtjuk az DFT-t. Például, ha az ablakméret 224 és az átfedés 100, akkor a második ablakunk tartalmazná az első ablak utolsó 100 tagját, ebből az is következik, hogy az eltolás az ablakméret és az átfedés különbségével egyezik meg. Az ablak típusát is meg lehet mondani, nem alkalmaznak négyszög ablakot ilyen célra, mert a különböző részek közti átmenetek túl durvák lennének. Ismertebb ablakfüggvények[2]: háromszög ablak, Hanning-ablak (Hann), Hamming-ablak, Blackman-ablak. A mi esetünkben Hamming-ablakot használunk.

A spektrogramm az STFT-kimenet komplex számainak valós részének, vagyis az amplitúdónak egy grafikus ábrázolása.

A vízszintes tengely az időt ábrázolja, a függőleges a frekvenciát, és a 6 ábrán a szürke pedig az energia mennyiségét. Az energia mennyiségét általában színtérképekkel (colormap)



(a) Nagy ablakméret.

(b) Kis ablakméret.

7. ábra. Egy hang spektrogrammja két különböző ablakmérettel.

ábrázolják, jelen esetben ez egy szürke színtérkép. A tengelyekhez a következőképpen rendelhetünk mértékegységeket: ha a rendszeróra $F_s = \frac{1}{T_s}$, akkor a legnyagyobb frekvencia $F_s/2$ Hz, frekvencia rezolúció F_s/L Hz és az időszeletek hossza LT_s másodperc.

Az ablakméret megválasztásakor nem ajánlott extrém értékeket választani, mert ha túl nagy az ablakméret, lásd 7a [6] ábra, akkor frekvencia szintjén ugyan pontos információval fog szolgálni a spektrogramm, de az idő síkjában nagyon pontatlan lesz, elveszhetnek a rövid impulzusok. Fordított esetben, túl rövid ablakméret esetén, a frekvencia lesz pontatlan és az idő pontos, 7b [7] ábra. Ez a jelenség azért figyelhető meg, mert az egyes részek területe, aminek az STFT kimenetének egy értéke felel meg, függetlenül attól, hogy mekkora az ablakméret, állandó marad. Az idő rezolúció $\Delta t = L$, a frekvencia rezolúció $\Delta f = \frac{2\pi}{L}$, a terület pedig mindig $\Delta t \Delta f = 2\pi$.

[6] Ábra forrása: https://www.coursera.org/learn/dsp/lecture/mXc4H/3-4-b-the-spectrogram, utolsó megtekintés dátuma: 2018.04.22

[7] Ábra forrása: https://www.coursera.org/learn/dsp/lecture/mXc4H/3-4-b-the-spectrogram, utolsó megtekintés dátuma: 2018.04.22

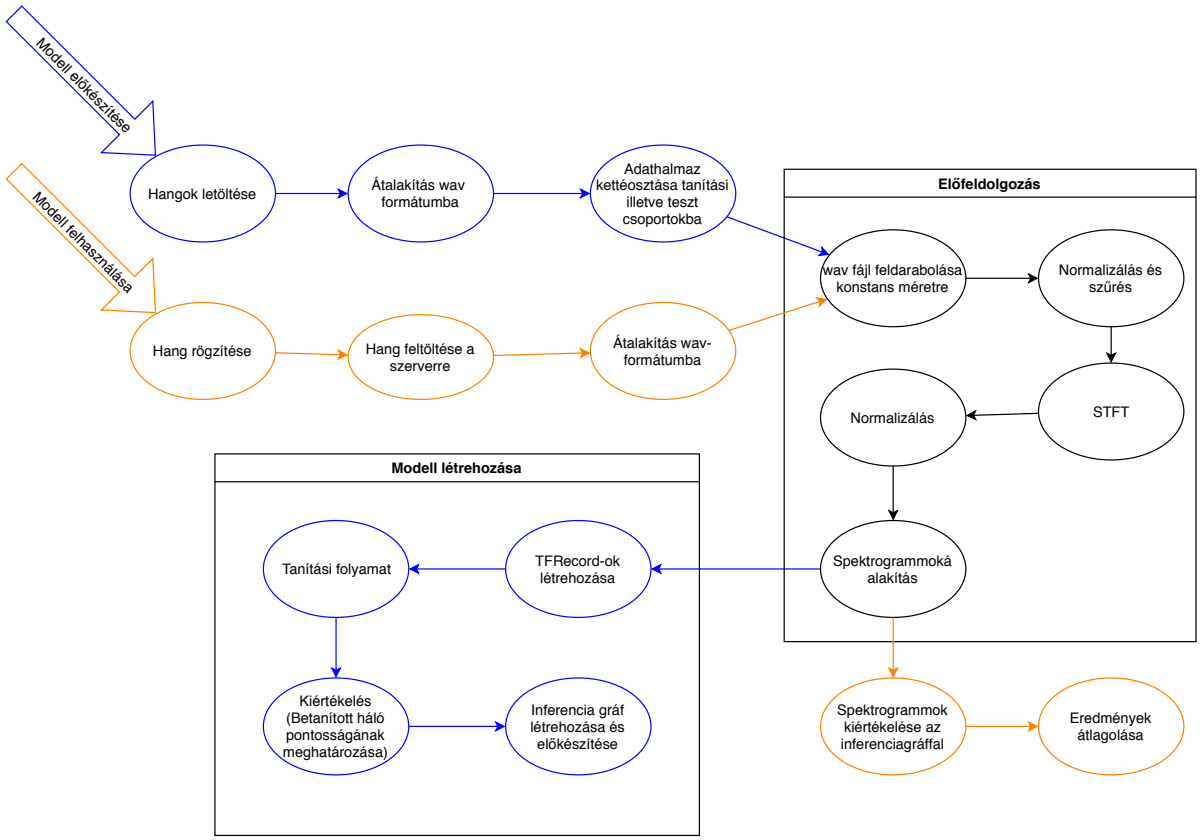
2. Módszertan

Két fontosabb lépéssorozatra van szükség a hangfelismerés megvalósításához: a neurális háló betanítása, illetve ennek kiértékelése és közzététele. Az 8. ábrán látható az a folyamat, amely során létrejön egy modell, amely alkalmas madárhangok osztályozására, illetve az a folyamat, amely során az említett modellt alkalmazzuk egyetlen felvett madárhangra.

A kékkel jelölt lépések a modell előkészítésére vonatkoznak, a narancssárgával jelöltek, pedig a modell felhasználására.

A modell előkészítése a következő folyamatokat foglalja magába: a háló tanításához szükséges adathalmazt begyűjti, azaz a Xeno-canto oldalról letölti a madárhangokat. A kezdetben MP3 kiterjesztésű hanganyagok WAV-formátummá alakítja. A WAV fájlokat két halmazba csoportosítja, annak érdekében, hogy megkülönböztessük a tanítási adatokat a teszt adatoktól. Ezt a folyamatot a 2.2 alfejezet tárgyalja.

Ezután következik az előfeldolgozás folyamata. Az előfeldolgozás külön a tanítási és külön a tesztadatokra is végrehajtódik. A WAV kiterjesztésű fájlokat előbb egyenlő hosszúságú csonkokra szegmentáljuk, az így keletkezett konstans méretű részeken normalizálást, aztán küszöbérték alapján szűrést végzünk el, így elvetve a halk részeket. A megmaradt hangokon rövid



8. ábra. Modell előkészítésének és felhasználásának a munkamenete. Az előkészítés (kék) és a felhasználás (narancssárga) során az előfeldolgozás lépései megegyeznek, ezért került egy ábrára a két munkamenet.

idejű Fourier-transzformáció (Short-time Fourier transform) kerül végrehajtásra, képpé alakítás előlépéseként normalizáljuk az adatot, majd létrejönnek a spektrogrammok, ezzel bezárul az előfeldolgozás folyamata (2.3) és kezdődik a modell létrehozásának folyamata. A kapott spektrogrammokat felcímkézve a nekik megfelelő madárfajok kódjával, *TFRecord* fájlokban tároljuk. A TFRecordokat felhasználva tanul a háló, melynek kiértékelése után eredményül egy inferenciagráfot kapunk. (2.5)

A modell felhasználásakor egyetlen hangról szeretnénk eldönteni hogy milyen madárfajhoz tartozik. Első lépésben a felhasználó rögzíti a hangot és feltölti ezt a szerverre, amely átalakítja WAV formátummá és végrehajtódik rajta az előfeldolgozás, melynek kimenetele 0, 1 vagy több spektrogramm lesz a küszöbértékes szűrés szerint. Amennyiben kellően erős jelet tartalmaz az eredeti hang, a feldolgozáson átesett spektrogrammok továbblépnek az inferenciagráf bemeneteiként. Az eredményeket összesítő átlagolás után a szerver egy HTTP-válaszban küldi el a végeredményt a felhasználónak. (3.1)

2.1. Korpusz

A Xeno-canto egy madárhanggyűjtő portál, lásd 9 [8] ábra, ahová bárki feltölthet a világ bármelyik tájáról hangokat, amiket fel tud címkézni, milyen génusz, faj, alfaj, hely, típus, minőség (A és E között, ahol A a legjobb minőségnek felel meg) és még néhány szempont alapján, illetve – ha nem tud a madárról semmit – megjelölheti rejtélyként, ebben az esetben a többi felhasználó segíthet a hang és esetleges leírás alapján azonosítani. A hangokat bárki letöltheti ingyenesen. Jelenleg majdnem 400.000 felvétel, közel 10.000 faj található az oldalon, amely több mint 4.000 feltöltőnek köszönhető, ez több mint 6.000 órányi madárhangot foglal magába.

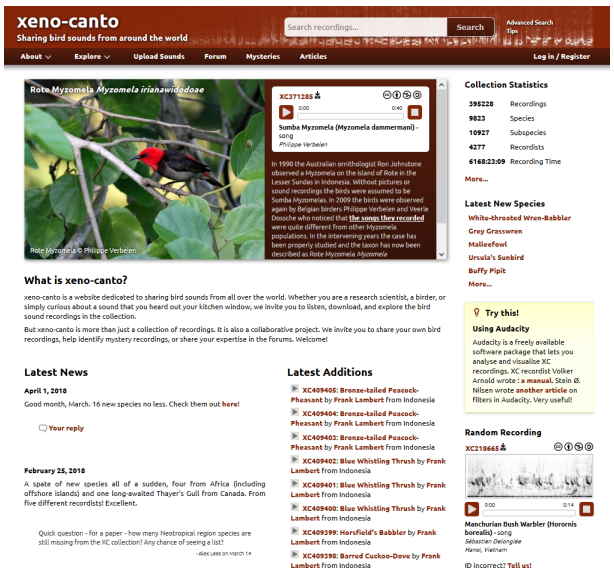
A korpusz forrásweboldala számos címkét nyújt minden hangmintának, pl. génusz, faj és alfaj. A kísérleteinkben használt osztályokat egy-egy génusz és faj kombinációja határoz meg.

A fajokon belül levő madarak hangja továbbá osztható még két csoportra: hívásra és énekre[13]. A hívás rövid, legtöbbször riasztásra szolgál, a hímek ezzel jelzik egymásnak a területüket, míg az ének hosszabb terjedelmű, a fajok körülbelül egyharmada képes énekelni, ezen fajok nagyrészében a hímek énekelnek a nőstények magukhoz való csalogatására. Az énekekben megkülönböztethetőek mondatok, szótagok és hangok is. Az általunk készített osztályokban nincsenek megkülönböztetve az énekek és a hívások.

A modell betanításához nagy mennyiségű adatra van szükség, melyet egy konfigurálható letöltő szkript gyűjt be a Xeno-canto oldalról felhasználva az oldal által szolgáltatott RESTful API-t.

A Xeno-canto oldal érdeklődő felhasználók rendelkezésére bocsát egy API-t [9], amelyre kül-

[8] Ábra forrása: `https://www.xeno-canto.org/`, utolsó megtekintés dátuma: 2018.04.21
[9] `http://www.xeno-canto.org/api/2/recordings`, utolsó megtekintés dátuma: 2018.04.21



9. ábra. A Xeno-canto oldala.

dött REST-kérésekre egy JSON formátumú válaszokat térít vissza. Ezen válaszok tartalmazzák a releváns hangokról szóló információt más hasznos mezők kíséretében, például találatok száma, fajok száma, oldalak száma, jelenlegi oldal, stb. Bizonyos címkék, például ország és hangminőség, alapján szűrni lehet. További információ ezen a linken található [10].

2.2. Korpusz letöltése

A 2.1 említetteknek megfelelően a háló betanításához szükséges nagy terjedelmű homogén adathalmaz begyűjtését egy letöltő szkript segítségével automatizáljuk.

A könnyű felhasználhatóság érdekében a szkript futtatásakor különböző opciók beállításával lehet meghatározni a letöltéshez szükséges információkat. A letöltés során több különböző lehetőség közül választhatunk:

- Megadhatunk egy szöveges állományt, amely tartalmazza a letöltendő madárfajokat (`Birdifile`)
- Generáltathatunk egy szöveges állományt, amely tartalmazni fogja az általunk megadott számú, legtöbb felvétellel rendelkező európai madárfaj nevét
- Nem határozunk meg semmilyen bemenetet madárfajok számával vagy nevével kapcsolatban, és letölti az összes európai madárfajhoz tartozó felvételt

Lehetőségünk van meghatározni a hangfájlok letöltési célkönyvtárát, ahova a hanganyagokat és az ezekből létrejövő spektrogrammokat tárolja. Emellett meg lehet adni a célállományt, ahol a háló bemenetének megfelelő típusára konvertált előfeldolgozott adathalmazt tárolja. Ezek a TFRecord fájlok, melyeket a 2.4 alfejezetben részletesebben tárgyalunk.

[10] `https://www.xeno-canto.org/article/153`, utolsó megtekintés dátuma: 2018.04.21

Jelen dolgozatban tárgyalt tanításhoz a korpuszt a második letöltési stratégia alapján hoztuk létre. A továbbiakban a használt letöltési módszer lépéseit részletesebben tárgyaljuk.

Első lépésben bemeneti adatként meghatározzuk a tanítani kívánt háló osztályainak a számát, azaz, hány madárfajhoz tartozó madárhangot szeretnénk letölteni. Bemeneti adatként meg kell határozni a létrehozandó (`Birdifile`) elérési útvonalát és két célállományt: egyet a hanganyagoknak és spektrogrammoknak, egyet a letöltés végtermékének, a TFRecordok számára.

Második lépésben a program lekéri a Xeno-canto oldalról az összes európai, legjobb minőségűként felcímkézett felvétel listáját, megjegyezve a madárfajok nevét, és a hozzájuk tartozó hanganyagok számát. Ez a szám szerint csökkenő sorrendbe rendezi a madárfajokat, és az első, bemeneti adatként megadott számú madárneveket kiírja a program által létrehozott (`Birdifile`)-ba.

A `Birdifile` létrehozása után a program lekéri a szöveges állományban levő madárfajokhoz tartozó felvételeket, újra megszámolja, hogy melyik madárfajhoz hány felvétel tartozik, megjegyzi az így kapott számok közül a legkisebbet annak érdekében, hogy letöltéskor minden fajhoz azonos számú adat tartozzon. Minden madárfajhoz véletlenszerűen kiválasztott felvételeket tölt le. A Birdifile-ban felsorolt összes madárfajról létrehoz egy-egy mappát és a neki megfelelő mappába menti el a madárfajhoz tartozó felvételeket.

A Xeno-canto oldalon MP3 kiterjesztésű hanganyagok vannak, amelyeket letöltésük után WAV kiterjesztésűre alakítja a program, majd az MP3 kiterjesztésű fájlt kitörli. A hangokat ezek után két csoportba osztjuk: 80%-a a hangoknak tanítási (más néven tréning) adat lesz, a maradék 20% pedig teszt adat. Ezzel zárul a letöltési folyamat és kezdődik az előfeldolgozás.

2.3. Hangok előfeldolgozása

A letöltés folyamán az MP3-as fájlokat 44100 mintavételezésű WAV-formátumú hangokká konvertáljuk. Ezután hajtódik végre rajta az előfeldolgozás.

Első lépésként a hangfeldolgozó algoritmus minden hang értékét normalizálja, utána három másodperces részekre darabolja. Ha kisebb a rész hossza, akkor azt a részt elveti. Az egyes részek hossza konfigurálható az algoritmusban. Minden rész kiértékelődik, hogy van-e benne fontos információ, nem túl halk-e a hang vagy nem csak zajt tartalmaz-e. Ezt egy küszöbérték segítségével teszi, amelyhez hasonlítja az illető rész hangerősségét és ha ez alá esik, akkor az adott részt elveti. Egyes részek hangerősségét négyzetes közép (root mean square) módszerrel határozza meg.

A következő lépés az STFT végrehajtása. Mivel a háló 224x224-es bemenet igényel a használt ablakmérete 448 egységnyi. Az ablakméret a képek méretének duplája (részletek a 1.2 fejezetben). Az algoritmus Hamming-ablakot használ, az átfedés (overlap) 224, az ablakméret

fele, ez is állítható, az STFT végrehajtásakor csak az amplitúdót kéri le, a fázis (phase) információjára nincs szükség a spektrogramm ábrázolásánál. Azonban ábrázolás előtt még szükséges normalizálni az adatokat, a felhasznált könyvtár (PIL) megkötése, hogy 0 és 1 közti értékek lehetnek csak. A normalizált STFT-kimenet decimáláson esik át, ez az adathalmaz mérete miatt fontos, hogy az 224x224-es mátrix legyen. Így minden érték egy pixelnek felel meg, ezután az adathalmazt felerősítjük egy gamma értékkel, és az ezután rátett színtérkép (colormap) határozza meg az értéknek megfelelő színt, és ezt kimenti PNG-formátumba, konfigurálható helyre, elérési útvonalra. A színskáláknál két különböző térképet lehet beállítani a függvényből, ezek jet vagy greyscale lehetnek, mindkettő a matplotlibbe beépített színtérkép. A greyscale-nek a 0 felel meg, a jetnek pedig bármilyen más szám.

2.4. Az előre betanított háló: TF slim és MobileNet

Hálónk tanítása és kiértékelése a Tensorflow [17] gépi tanulást elősegítő keretrendszerben íródtak.

A Tensorflow támogatja a nagy számításigényű és párhuzamosítható algoritmusok végrehajtását, lehetőséget adva a felhasználónak, hogy CPU mellett GPU, TPU platformon működtesse. Elterjedt a gépi tanulás és mélytanulás szakterületeken belüli használata.

A Tensorflow keretrendszerre épített számos magas szintű API közül a TF-Slimet [11] használjuk. A TF-Slim komplex hálók létrehozására, tanítására és teljesítményének kiértékelésére könnyen használható környezetet biztosít. Olyan függvényeket tartalmaz, amelyek elősegítik a hálók tanítását, ismert adathalmazok beszerzését (CIFAR-10, MNIST, ImageNet), TFRecordok írását, olvasását, inferenciagráfok létrehozását, alkalmazását. Ezek mellett még más népszerű konvolúciós neurális hálókkal való munkát is lehetővé teszi, mint például az Inception, ResNet, VGG, MobileNet. Lehetőségünk van előre betanított TensorFlow állományokat letölteni ezekhez a modellekhez, amelyek „ckpt” kiterjesztésűek (checkpointok). Előtanított hálók hatékonynak bizonyulnak gyakorlatban idő, illetve erőforrás takarékosság szempontjából.[21]

Az általunk választott előretanított háló a MobileNet. Több változata közül a legnagyobb pontosságot (70.9%) elérő változat 224x224 méretű, színes (RGB) képeket fogad el bemenetként. A hálót, mivel mobil applikációra volt tervezve, nem annyira a pontossága miatt, inkább gyors válaszideje miatt választottuk.

Architektúrájának fő rétege a mélységben elválasztható konvolúciós réteg (Depthwise Separable Convolution). Ez a réteg a hagyományos konvolúciós réteget két másik konvolúciós rétegre bontja fel: egy mélységi konvolúcióra, amely a MobileNet esetén egyetlen filtert alkalmaz minden bemenetre és egy 1x1-es pontonkénti konvolúcióra, amelyet a mélységi rétegre alkalmaz. Ez a folyamat úgy interpretálható, hogy szűrés és a „felgöngyölítés”, amelyet a ha-

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

10. ábra. MobileNets architektúrája.

gyományos esetben egy réteg, a konvolúciós réteg, végzi, itt két réteg hajtja végre. Több lépésben végzi el a feladatot, viszont egyszerűbb számításokat kell elvégeznie, ami a számítógép erőforrásfelhasználás szempontjából előnyös.

A MobileNet 28 réteget tartalmaz. Az első réteg egy hagyományos konvolúciós réteg, amelyet követnek a mélységi konvolúciós és a pontonkénti konvolúciós rétegek. Minden konvolúciós réteget egy Batch Normalization és egy ReLU aktivációs függvény követi. Az utolsó három réteg az átagoló tömörítés (average pooling), teljesen összekötött réteg (FC layer) amit egy Softmax kiértékelő függvény követ. Az architektúra megtekinthető a 10 [11] ábrán. [6]

2.5. A háló tanítása

A háló tanítása a tréning adatok által történik, és a folyamat eredménye egy modell, amelyet tesztadatokkal validálunk. A tanítási és teszt adatok a letöltési folyamat és előfeldolgozási folyamat eredménye. A program külön átalakítja a tanítási és teszt adatokat TFRecordokká. Annak érdekében, hogy azonos fajhoz tartozó spektrogrammok ne csoportosuljanak, átalakítás előtt az adatokat megkeverjük, különös tekintettel arra, hogy a spektrogrammokhoz tartozó osztályok (madárfajok) a képekkel párhuzamosan keveredjenek, így megtartva a helyes kép-címke (spektrogramm-madárfaj) kapcsolatot.

A TFRecord a Tensorflow által használt fájltípus. Ez a formátum olyan információkat foglal magába a bemeneti adatokról, mint például a spektrogramm, a hozzátartozó osztály kódja (az-

[11] Ábra forrása: https://joshua19881228.github.io/2017-07-19-MobileNet/, utolsó megtekintés dátuma: 2018.04.23

az melyik madárfajhoz tartozik), kép mérete, illetve kódolási és dekódolási információk. Egy TFRecord fájlba több ilyen adatstruktúra kerülhet. A program lehetőséget ad meghatározni, hogy hány képet (a hozzá tartozó információkkal) tároljon egy TFRecordba. Használata nem kötelező, de előnyös és elterjedt a Tensorflow keretrendszert használók körében, mert könnyen kezelhető.

A TFRecord állományok a tanítási és tesztadatokból való elkészülése jelképezi a végső előfeldolgozási lépést, amely után a korpusz készen áll a hálóval való interakcióra.

A képosztályozást elvégző algoritmus kódját és a MobileNet hálót a TF-Slim szolgáltatja, így a tanítást egy függvényhívással indítjuk el, amelynek paraméterként megadjuk a beállításokat.

Tanításaink során a 4 fejezetben tárgyalt paraméterek változatlanok maradnak, függetlenül attól, hogy hány osztályra tanítunk be, vagy milyen színtérképet alkalmazunk a spektrogrammok vizualizálására. A háló kiértékelésére szintén van lehetőség egy egyszerű függvényhívás keretén belül. A függvényhívás eredményeként megkapjuk a pontosságot, amellyel a háló felismeri és megkülönbözteti az osztályokat.

A tanítás kimenetelei úgynevezett checkpointok lesznek – ckpt kiterjesztésű fájlok, amelyek a háló állapotáról tárolnak információkat. Ezeket a checkpointokat két módon lehet felhasználni: tovább lehet tanítani a hálót azonos vagy különböző beállításokkal valamint felhasználható állapontba hozni a hálót inferenciagráfok által.

Inferenciagráfok létrehozására és kezelésére szintén létezik példaimplementáció a TF-Slim csomagban. Erre épülnek a Birdify gráfját elkészítő és felhasználó szkriptek.

3. A rendszer architektúrája

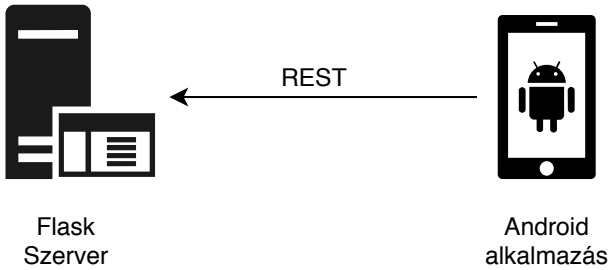
Jelen fejezet bemutatja, hogy a már betanított hálót hogyan használjuk fel, illetve hogyan lesz elérhető a felhasználó számára. Ennek megvalósításához egy klasszikus kliens-szerver architektúrát használtunk.

3.1. RESTful szerver

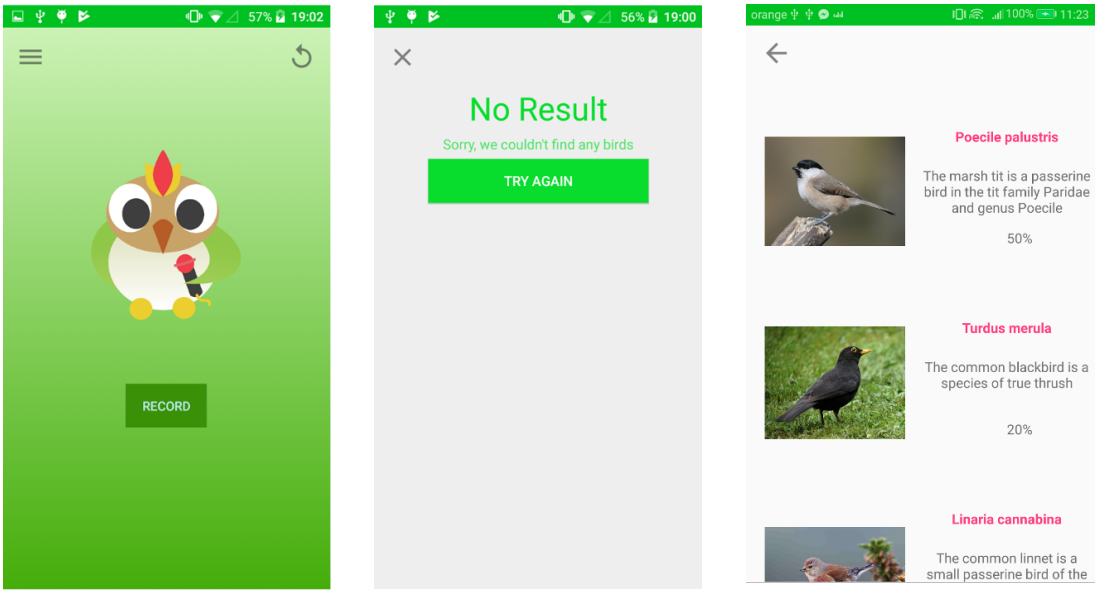
A betanított modellt reprezentáló referenciagráfot egy RESTful API-n keresztül tesszük elérhetővé az érdeklődök számára. Ahogyan a 11. ábrán is látható, az alkalmazás egy kliens-szerver architektúrán alapszik. Szerverünk a Pythonban írt Flask[4] keretrendszert alkalmazza HTTP kérések kiszolgálására. Ezen kérések betartják a REST (Representational State Transfer) szabványokat.

A szerverre kiértékelés céljából az `/api/upload/records` URL használatával lehetséges felvételt feltölteni. A továbbiakban tárgyalni fogjuk a szerver működését.

Feltöltés esetén első lépésként a szerver ellenőrzi, hogy sikeres volt-e. Amennyiben sikeres, ellenőrzi, hogy a kiterjesztés megfelel-e a követelményeknek. Jelenleg a szerverünk MP3 és MP4 kiterjesztésű audio fájlokat tud kezelni. A különböző támadások kivédése érdekében biztonságossá alakítja a fájl elérési útvonalát, majd elmenti a fájlt a szerveren található `Recordings` mappába. Átalakítja a hangfelvételt WAV kiterjesztésűvé, annak érdekében, hogy el tudja végezni az előfeldolgozást, amelyet a tanítási és tesztelési adatok esetén is tárgyaltunk. Az előfeldolgozás végeredménye 0, 1 vagy több PNG kiterjesztésű spektrogrammot ábrázoló kép. Ha nem készült egyetlen spektrogramm sem, akkor a feltöltött hanganyag túl halk volt, és egy ennek megfelelő hibaüzenetet küld válaszként a szerver. Ellenkező esetben minden spektrogrammra kiértékeli a betanított modellt. Minden képre kapunk egy-egy listát, amelynek annyi eleme van, ahány osztályra betanítottuk a hálót. Mivel a háló utolsó rétege Softmax kiértékelő, a listák elemei 0 és 1 közötti értékek, és összegük 1. Elvetjük azokat a listákat, amelyek nem tartalmaznak egyetlen a küszöbértéknél nagyobb elemet sem. A megmaradt listákat átlagoljuk. Így egyetlen listát kapunk eredményül, amelynek az osztályokkal megegyező számosságú elemünk



11. ábra. Kommunikációs diagramm. Az Android alkalmazás REST kéréseket küld a Flask szerver felé.



(a) A gomb amelyet lenyomva elkezdi felvenni a hangot.

(b) Az eredménynézet 0 találat esetén.

(c) Az eredménynézet több találat esetén.

12. ábra. A mobilalkalmazás főbb nézetei.

lesz.

Az URL-ben argumentumként a felhasználó meghatározhat két paramétert: `top` és `threshold` (küszöbérték). Ezeknek alapértelmezett értékük 3 és 0.5. A szerver visszatéríti az első `top` számú olyan madár adatait (ID, név és elért pontszám), amelynek pontszáma nagyobb vagy egyenlő a `threshold` értékével.

3.2. Android kliensalkalmazás

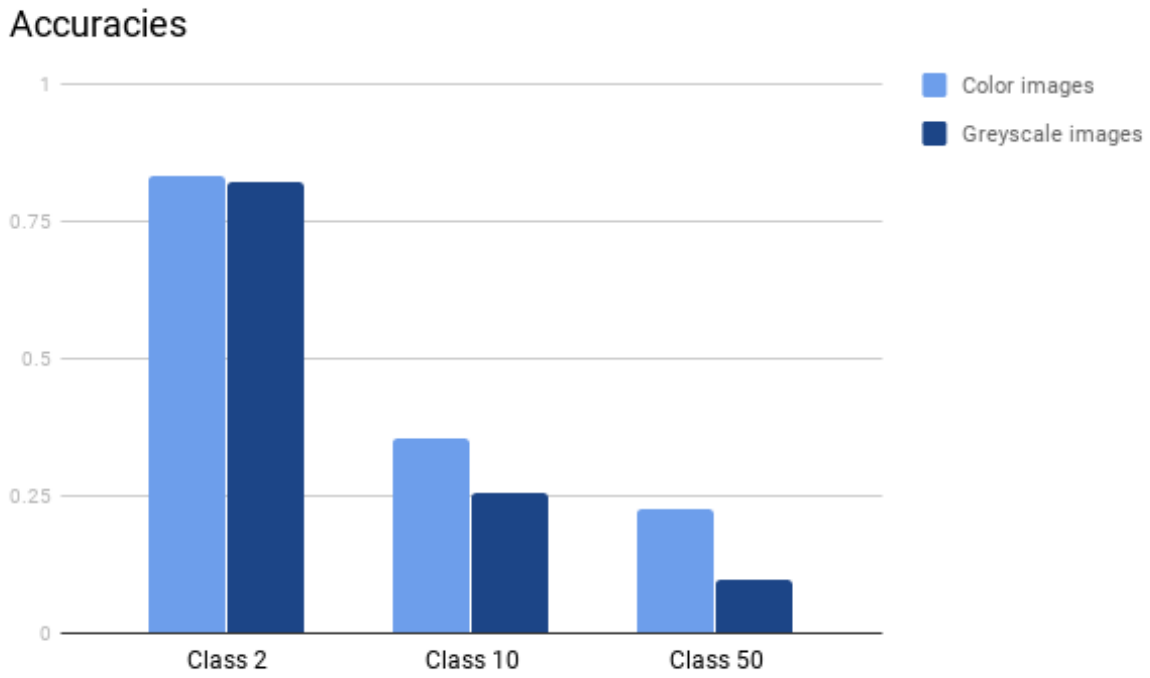
A szerver mellett elkészült egy Android kliensalkalmazás is. Ennek felhasználói felületének központi eleme egy mikrofonnal ábrázolt madárral együtt megjelenő gomb (12a. ábra), amelynek lenyomására a telefon elkezd egy legfennebb tíz másodperces hangot felvenni. A hang felvétele után ez elmentődik MP4-es formátumba, továbbküldődik a szerverre, illetve a szerver válaszára vár. Egy bizonyos időn belül, ha nem érkezik válasz, illetve ha üres tömb érkezik vissza válasz gyanánt, akkor az jelenítődik meg, hogy nincsen találat (12b. ábra). Válasz esetén kiírja az első három madarárfajt, amire legnagyobb az egyezés (12c. ábra). A lista nagyságának értéke változtatható, illetve lehetőségünk van egy küszöbérték megadására, amely szerint a rendszer kiszűri a megjelenített eredménylistából az ennél kisebb pontot szerzett madarakat.

4. Kísérletek és eredmények

A kísérletek során több konfigurációt is elemeztünk. Elsősorban megnéztük, hogy mennyire változik a háló eredményessége az osztályok számától függően, illetve tanulmányoztuk a spektrogrammok ábrázolási színskáláinak jelentőségét a tanítási folyamatban. A következő osztály számosságokra tanítottunk hálót: 2, 10 és 50, hogy megfigyelhessük az osztályozási pontosság csökkenését az osztályok számának növekedésével. A spektrogrammok létrehozása folyamán két színskála típust vettünk figyelembe: a fekete-fehér illetve RGB színtérkép, másnéven Jet színskála. A Jet a spektrogrammokban kéktől-pirosig történő reprezentáció a hangok erősségét jelöli, ahol kék a halk, piros a hangos. Ez a típusú reprezentáció jó az emberi szemnek, de nem biztos hogy jó a gépnek, ezért alkalmazzuk a fekete-fehér ábrázolást is.

A tanítás végén nemcsak checkpoint-okat kapunk, hanem a tanítási folyamatot leíró fájlokat is, amelyeket fel tud használni a Tensorboard. A Tensorboard egy vizualizációs eszköz, amely a Tensorflowban szerkesztett hálókról vizuális információt ad. Tensorboard segítségével egyszerű követni a veszteségfüggvény csökkenését a tanítási folyamatok során.

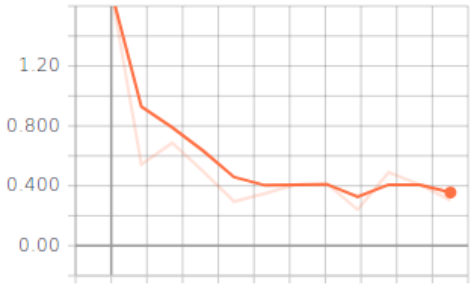
Kísérleteink során hat lényeges tanítási folyamatot végeztünk el, az ezek során megfigyelető veszteségcsökkenést a 14 ábrán lehet megtekinteni. Minden tanítás tízezer lépést hajtott végre. A lépéseket a vízszintes tengelyen, míg a veszteségek értékeit a függőleges tengelyen vannak ábrázolva.



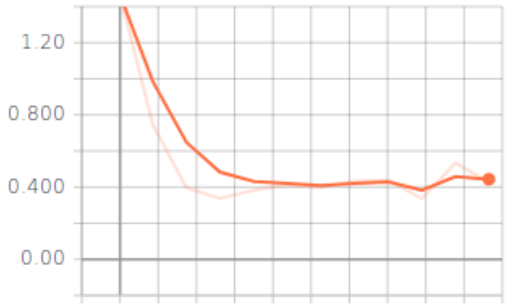
13. ábra. MobileNet háló által elért eredmények az osztályok számának és spektrogramm színtérképének függvényében.

Az elvárásnak megfelelően a háló kiértékelési pontossága (accuracy) az osztályok számának növekedésével csökken. A tanítások során használt paraméterek (hyperparameters), mint például a lépések száma, tanulás sebessége, változatlanok. A tanítási adatot 32-es kötegekre bontottuk illetve a tanítási ráta 0,0001 és a súlycsökkenés paraméter 0,00004. Mivel a letöltési fázis során véletlenszerűen töltjük le a hangokat, így a bemeneti adathalmaz változatos, viszont számszerinti különbségek elenyészően kicsik.

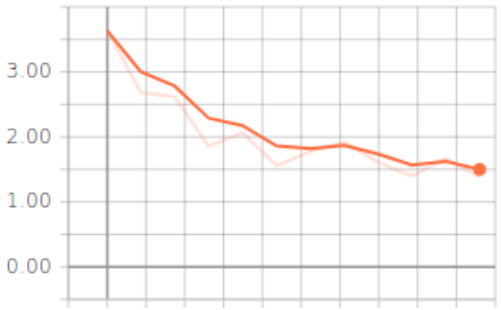
A kísérletek által elért pontosságok az 13 diagrammal ábrázoljuk. A spektrogrammok színtérképeit figyelembe véve a Jet színtérképpel átlagban 0,079-el nagyobb kiértékelő pontosságot értek el, tehát 8%-al jobb eredményt értünk el RGB színskála esetén. Ez a jelenség azzal is magyarázható, hogy a MobileNet hálónk színes képekre voltak előre betanítva, a mi tanításaink



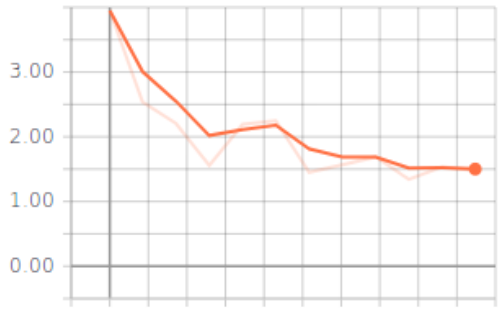
(a) Két osztály, színes színtérkép-spektrogramm.



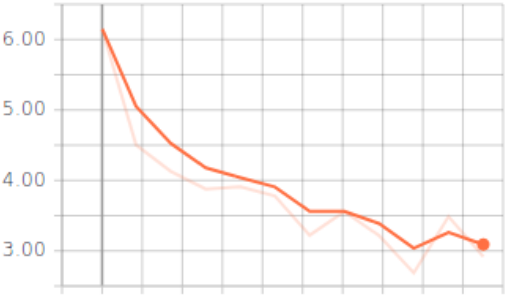
(b) Két osztály, fekete-fehér színtérkép-spektrogramm.



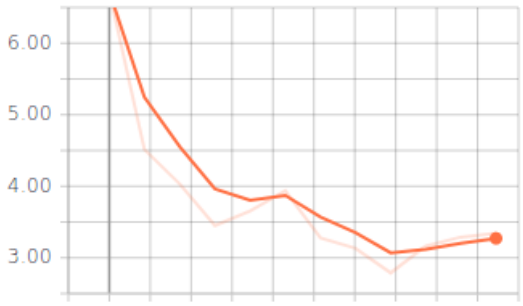
(c) Tíz osztály, színes színtérkép-spektrogramm.



(d) Tíz osztály, fekete-fehér színtérkép-spektrogramm.



(e) Ötven osztály, színes színtérkép-spektrogramm.



(f) Ötven osztály, fekete-fehér színtérkép-spektrogramm.

14. ábra. Veszteségek csökkenésének diagrammjai a különböző tanítások során.

erre a betanított hálóra alapoznak. Így a fekete-fehér színskála előnyei nem tudtak érvényesülni, mivel a háló alacsony szinten a színeket is „kereste” a tanítási folyamat során.A 13 diagrammon megfigyelhető, hogy minél több osztályra tanítottuk a hálót, annál nagyobb mértékben növekedett az RGB színskála alapján tanított háló pontossága a fekete-fehér színskálával szemben.

Következtetések és továbbfejlesztési lehetőségek

Munkánk során arra következtettünk, hogy az RGB spektrogrammok hatásosabbak, az előre betanított MobileNet háló esetén. Ezt magyarázza a hálónak az alsó rétegei színes képekkel való betanítottsága. Emellett még kiemelendő az a jelenség, amelyet a színes illetve a fekete-fehér spektrogrammok alapján betanított hálók eredményeit összehasonlítva lehet megfigyelni az osztályok nagyságának függvényében. Az osztályok számának növekedtével, az eltérés a különböző típusú spektrogrammok eredményei között egyre jelentősebb, mégpedig olyan módon, hogy az RGB színskála jobb eredményekkel szolgál.

Jobb eredmények elérésen érdekében a következő javításokat lehet figyelembe venni: egy robosztusabb előretanított kovolúciós neurális háló használata, mint például a ResNet. Annak érdekében, hogy megfelelő módon hasonlíthassuk össze a fekete-fehér illetve a színes spektrogrammok hatását a tanításban, a tanítások elvégezhetők nem előre betanított hálóval. Az előfeldolgozási folyamatba a zajszűrés bevezetése előnyös lehet ahhoz, hogy azokat a hangokat, amelyeket nem madarak adnak ki, szűrjük ki, illetve különböző gamma értékek kipróbálásával megnövelhető a hálónak jelentős információk kiemelésére. Egy másik, madárfelismerés eredményességét növelő, módszer lenne a hang- és képfelismerés kombinált alkalmazása egy multi-attention CNN háló által. [10]

Hivatkozások

[1] J. Allen. „Short term spectral analysis, synthesis, and modification by discrete Fourier transform”. *IEEE Transactions on Acoustics, Speech, and Signal Processing* 25. évfolyam, 3. szám (1977. jún.), 235–238. oldal. ISSN: 0096-3518. DOI: 10.1109/TASSP.1977.1162950.

[2] Kelemen András. *Digitális jelfeldolgozás*. URL: https://www.tankonyvtar.hu/hu/tartalom/tamop412A/2011-0013_kelemen_digitalis_jelfeldolgozas/441_ablakozs.html (utolsó elérés dátuma: 2018. ápr. 23.)

[3] Victor Bisot et al. „Leveraging deep neural networks with nonnegative representations for improved environmental sound classification”. *IEEE International Workshop on Machine Learning for Signal Processing MLSP*. Tokyo, Japan, 2017. szept. URL: https://hal.archives-ouvertes.fr/hal-01576857.

[4] *Flask Documentation*. URL: flask.pocoo.org/docs/0.12/ (utolsó elérés dátuma: 2018. ápr. 22.)

[5] Kaiming He et al. „Deep Residual Learning for Image Recognition”. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), 770–778. oldal.

[6] Andrew G. Howard et al. „MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications”. *CoRR* abs/1704.04861. évfolyam, (2017).

[7] Sergey Ioffe és Christian Szegedy. „Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.” *ICML*. Szerkesztette Francis R. Bach és David M. Blei. 37. évfolyam, JMLR Workshop and Conference Proceedings. JMLR.org, 2015, 448–456. oldal. URL: http://dblp.uni-trier.de/db/conf/icml/icml2015.html#IoffeS15.

[8] Stefan Kahl et al. „Large-Scale Bird Sound Classification using Convolutional Neural Networks”. *CLEF*. 2017.

[9] Alex Krizhevsky, Ilya Sutskever, és Geoffrey E. Hinton. „Imagenet classification with deep convolutional neural networks”. *Advances in Neural Information Processing Systems*, 2012. oldal.

[10] *Learning Multi-Attention Convolutional Neural Network for Fine-Grained Image Recognition*. URL: https://www.microsoft.com/en-us/research/publication/learning-multi-attention-convolutional-neural-network-fine-grained-image-recognition/ (utolsó elérés dátuma: 2018. ápr. 23.)

[11] Sergio Guadarrama Nathan Silberman. *TensorFlow-Slim image classification model library*. 2018. URL: https://github.com/tensorflow/models/tree/master/research/slim (utolsó elérés dátuma: 2018. ápr. 16.)

[12] *Official syllabus of the Stanford University online course CS231n: Convolutional Neural Networks for Visual Recognition*. 2015. URL: http://cs231n.stanford.edu/2015/syllabus (utolsó elérés dátuma: 2018. ápr. 16.)

[13] *Ornithology The Science of Birds: Songs and Calls*. URL: https://ornithology.com/ornithology-lectures/songs-calls/ (utolsó elérés dátuma: 2018. ápr. 21.)

[14] Martin Vetterli Paolo Prandoni. *Digital Signal Processing Coursera course*. URL: https://www.coursera.org/learn/dsp (utolsó elérés dátuma: 2018. ápr. 23.)

[15] Karol J. Piczak. „Environmental sound classification with convolutional neural networks". *2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP)* (2015), 1–6. oldal.

[16] Karol J. Piczak. „Recognizing Bird Species in Audio Recordings using Deep Convolutional Neural Networks". *CLEF*. 2016.

[17] *Programming Guide for Tensorflow*. URL: https://www.tensorflow.org/programmers_guide/ (utolsó elérés dátuma: 2018. ápr. 16.)

[18] Justin Salamon és Juan Pablo Bello. „Deep Convolutional Neural Networks and Data Augmentation for Environmental Sound Classification". *IEEE Signal Processing Letters* 24. évfolyam, (2017), 279–283. oldal.

[19] Elias Sprengel et al. „Audio Based Bird Species Identification using Deep Learning Techniques". *CLEF*. 2016.

[20] Christian Szegedy et al. „Going Deeper with Convolutions". *Computer Vision and Pattern Recognition (CVPR)*. 2015. URL: http://arxiv.org/abs/1409.4842.

[21] *Transfer learning and The art of using Pre-trained Models in Deep Learning*. URL: https://www.analyticsvidhya.com/blog/2017/06/transfer-learning-the-art-of-fine-tuning-a-pre-trained-model/ (utolsó elérés dátuma: 2018. ápr. 21.)

[22] Avinash Sharma V. *Official syllabus of the Stanford University online course CS231n: Convolutional Neural Networks for Visual Recognition*. 2017. URL: https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0 (utolsó elérés dátuma: 2018. ápr. 16.)