Mélytanulás tantárgy házi feladat (2023 őszi félév)

Végső beadási dokumentációja

Választott projekt: BirdCLEF 2023

Csapatnév: DeepBirding

A csapat tagjai:

Név	Neptun-kód	Email-cím
Bihari Bence	IVXWF8	bence.bihari99@gmail.com
Hegyi Lehel	GSZLZ7	hegyilehel@gmail.com
Turi Máté*	OPVP7J	-

***Turi Máté csapattagunk sajnos nem tudta folytatni velünk a közös munkát a Milestone 1 után, így ezt követően nélküle folytattuk tovább a projektet**

1. Bevezetés

1.1 A BirdCLEF 2023 versenyről

A BirdCLEF 2023 gépi tanulási kódversenyt [1] a Kaggle nevezetű gépi tanulás és adattudomány platformon hirdette meg a NATURAL STATE kenyai természetvédelmi szervezet, melyen a résztvevők 2023. március 07. és május 25. között küldhették be megoldásaikat. A kódverseny célja, hogy a versenyzőknek egy olyan megoldást kellett benyújtaniuk a felületen, mellyel képesek voltak 264 kelet-afrikai madárfaj azonosítására a megadott madarak hangjai alapján. A BirdCLEF 2023 értékelési metrikája a padded cmAP volt, amely a scikit-learn könyvtárban is implementált makro-átlagolt átlagos pontossági pontszám származéka.

1.2 Kitűzött feladatunk

A tantárgy keretében a feladatunk ezen versenyen való késői részvétele volt, melynek regisztrációjára a Kaggle felület lehetőséget is adott, a csapattagok a verseny elfogadásával az adathalmazhoz hozzáférési jogot kaptak, valamint a felületre megoldás beadására való lehetőséget is kaptunk.

1.3 Az adathalmaz bemutatása

A munkák bemutatásának elengedhetetlen része, hogy a BirdCLEF 2023 adathalmazt vázlatosan bemutassuk. Az adathalmaz a következő fájlokat/könyvtárakat tartalmazta:

- **train_audio/**: ez könyvtár a modell tanításához szükséges hanganyag fájlokat tartalmazza. A hanganyagok .ogg kiterjesztésben, valamint 32.000 KHz felbontásban vannak, valamint az egyes madárfajok azonosítói szerinti mappában vannak rendezve, ezzel megkönnyítve az adatok betöltését. Az adathalmaz pontosan 16753 darab hangfájl példányt tartalmaz.
- **train_metadata.csv**: a metaadatokat tartalmazó .csv fájl, melyben megtalálhatók az egyes hangfájlhoz tartozó annotációk, többek között a felvételen található madár fajának azonosítója, fájl neve, valamint a felvétel készítésének koordinátája és a felvétel készítője. Egy hangfájl példányhoz 12 darab metaadat tartozik.
- **test_soundscapes/**: ez a könyvtár a kész megoldás tesztelésénél lesz releváns, ezt a mappát fogja feltölteni teszt audio adatokkal, és ezen adatok mentén fog kiértékelődni a megoldás.
- **sample_submission.csv** egy példa beküldési .csv fájl, amely a formalitást egyértelműsíti.
- **eBird_Taxonomy_v2021.csv**: .csv kiterjesztésű fájl, amely megadja, hogy egyes madárfajok között milyen típusú relációk fedezhetők fel.

1.4 State of the Art megoldás

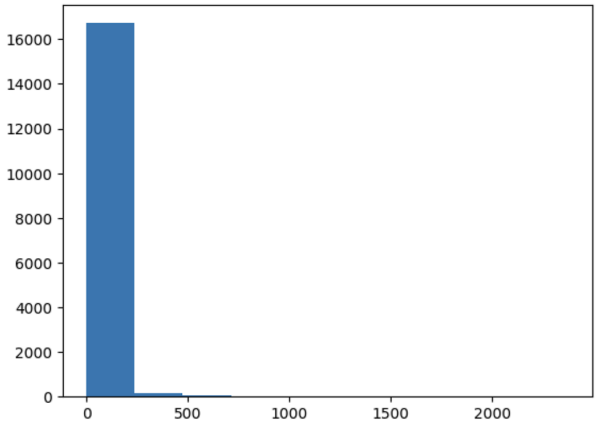
Mivel a Kaggle-n található BirdCLEF 2023 verseny már május 25.-i dátummal véget ért, így volt lehetőségünk betekintést nyerni a legjobb pontosságot elérő megoldások leírásainak közé. A ranglistán lévő versenyzők megoldásaikhoz opcionálisan egy vázlatos kiírást tettek közzé, melyben leírták, hogy milyen mélytanulási technikák bevetése segítette őket előre a pontosság növeléséhez, továbbá azt is leírták, hogy melyek nem segítettek őket előre.

2. Munkánk bemutatása

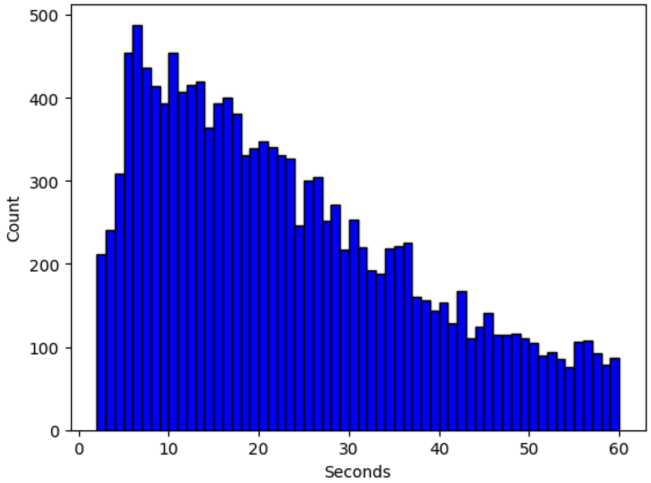
A csapatunk a legjobb eredményt elérő megoldások feltérképezésével kezdte meg munkáját, mellyel sok tanulságot le lehetett vonni a versennyel kapcsolatban. Többek között arról azt tudhattuk meg, hogy melyik úton érdemes elindulni a csapatunk saját megoldásával, valamint hogy mely technikák bevetésére érdemes nagyobb hangsúlyt fektetnünk, továbbá hogy melyekre nem érdemes nagy hangsúlyt fektetni.
A megoldásunkat a Kaggle felületen készítettük el, mely egy Python Jupyter notebook-os virtuális környezetet szolgáltat. A Kaggle biztosít a virtuális környezethez a CPU, memória és tárhelyen kívül GPU eszközt is, melynek heti használati kvótája 30 óra, mely sajnos esetenként kellemetlenül ért minket a tanítás során az erőforrás elérhetőségének korlátozottsága miatt.
Mind az adatok előkészítéséhez, mind a modell tanításához és teszteléséhez a PyTorch Lightning [2] által biztosított eszközöket használtuk fel, mellyel könnyen egységbe lehetett zárni a modellhez és adathoz, valamint a tanítást végrehajtó egységhez tartozó struktúrákat. A modell prédikációinak tesztelésére pedig egy Gradio által szolgáltatott felhasználói felületet készítettünk.

2.1 Az adathalmaz előfeldolgozása

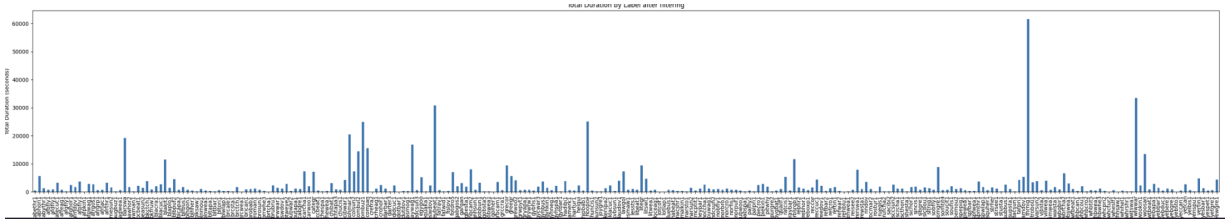
Az adathalmaz előfeldolgozását a data_processing.ipynb Jupyter notebookban valósítottuk meg. A BirdCLEF 2023 adathalmaz eléréséhez két lehetőségünk van: vagy a Kaggle notebookon keresztül lehetőségünk van inputként hozzáadni az adathalmazt, vagy lokálisan is el tudjuk érni Kaggle API kulccsal. Az utóbbi módszerről találhatunk példát a data_processing.ipynb notebook-ban. A notebookban továbbá betöltjük az adathalmaz metaadatait és audio adatait, és azokból statisztikát állítunk elő.



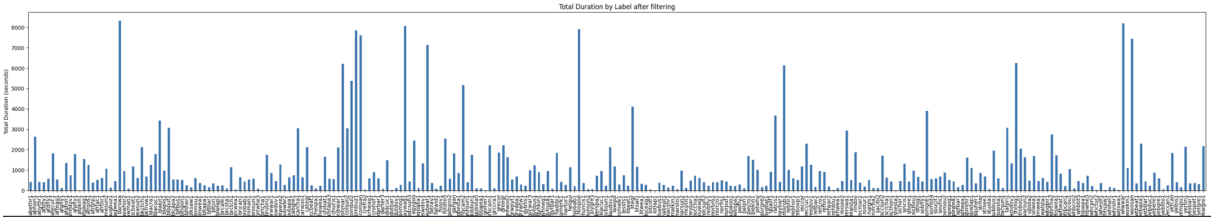
Az előbb látott diagram vízszintes tengelyén az audio fájl hossza másodpercben, a függőleges tengelyen az abban a hossz terjedelemben lévő audio példányok száma található. Felfedezhetjük, hogy az audio adatok túlnyomó többsége 225 másodperc alatt van, de találhatunk olyan hangfájl példányt is az adathalmazban melynek több mint 2374 másodperc a hossza.
Ezeket a kiugróan magas értékeket érdemes a teljesítmény érdekében eltávolítani az adathalmazból, ezt a hossz maximálisan 60 másodpercre határoztuk meg. Továbbá érdemes eltávolítani azokat az audio példányokat is, melyek túl kevés információt hordoznak ahhoz, hogy megtartsuk őket az adathalmazban, ezt a határszámot 2 másodperc alatt határoztuk meg. A következő ábrán a szűrt adathalmaz példányainak hossza található. Látható, hogy egy sokkal egységesebb adathalmazt kaptunk a módszer felhasználása után.



Következő lépésként megnéztük az adathalmaz osztályaira vetített hanghossznyi eloszlást, ami sajnálatos módon nem mutatott teljesen kiegyensúlyozott eloszlást. Mivel 264 osztályunk van, így az ábrát meglehetősen nehéz kiolvasni, azonban látszik, hogy egy-két osztályhoz rengeteg mennyiségű hanghossz "jut", míg vannak olyanok, melyek alig pár másodperc hanganyagot "kapnak".



Ezen okból kifolyólag ilyen téren is egy adatszűrést végzünk el, melyben a túl nagy hanganyag hosszal rendelkező osztályok hanganyag hosszát lecsökkentjük az adathalmaz kiegyensúlyozásának érdekében. A következő ábrán látható a szűrés okozta különbség az adathalmaz osztályain.

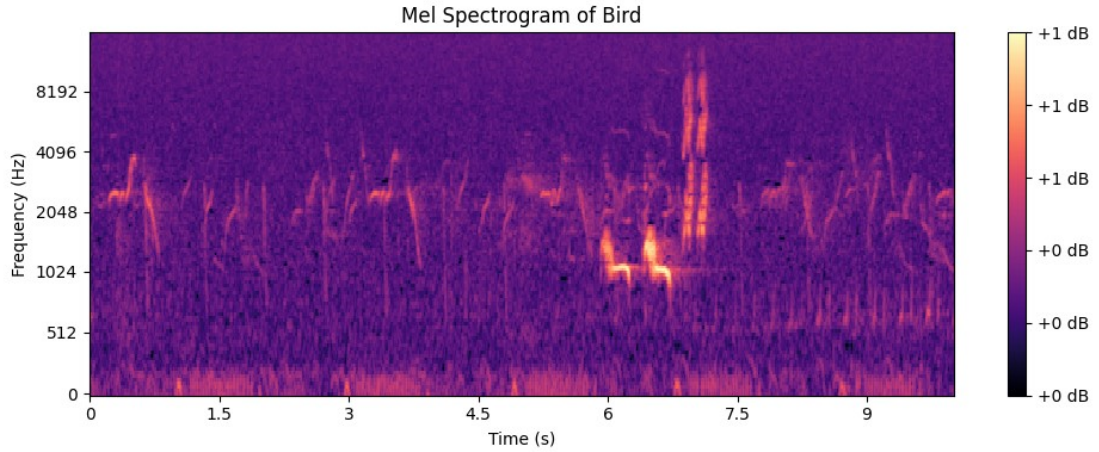


A megszűrt adathalmaz metaadatait filtered_metadata.csv fájlként mentettük el, mely az adathalmaz feldolgozásának fázisában bemenetként fog szolgálni.

2.2 Adathalmaz feldolgozása a modell bemenetére

Az adathalmaz feldolgozását a training.ipynb Jupyter notebookben valósítottuk meg. A feladat megvalósítása során azt a döntést hoztunk, hogy a bemeneti audio fájlokat átalakítjuk Mel Spectrogramokra a jobb pontosabb predikciók eléréséhez. Ezt a módszert gyakran használják a Mel Spectrogramokat audióval kapcsolatos gépi tanulás feladatok megvalósítására.[3] A Mel Spectrogramról azt kell tudni, hogy a Mel skálát használja a frekvencia helyett és a decibel skálát az amplitúdó helyett. A Mel skála az emberi hallást reprezentáló logaritmikus skála. A Mel Spectrogramokkal való tanítás olyan mintha képeket használnánk tanításhoz, ezért is volt az eredet tervünk a klasszifikációhoz CNN modellt használni. Mivel a feladat madár hang osztályozás volt a feladat ezért a Mel Spectrogramok használata mellett döntöttünk. Az eredeti audió fájlok sample rate-je 32Khz volt, rövid kutatás után eldöntöttük, hogy a sample ratet lecsökkentjük 12Khz-re, mivel nem volt olyan madár a datasetben, amelynek van ennél magasabb frekvencián hangja. Meg az emberi hallás is 20Khz-ig terjed. Ezután normalizáltuk a Mel Spectrogramok decibel értékeit, meg alkalmaztunk transzformációkat és augmentációkat. Egyforma hosszra alakítjuk a spectrogramokat, gaussian zajt és

time-shiftet is adtunk a bemeneti spectrogramokhoz, valamint egy esőt szimuláló hanganyagot is hosszáadunk. Ezen feldolgozási és augmentációs lépések után kapott Mel Spectrogram:



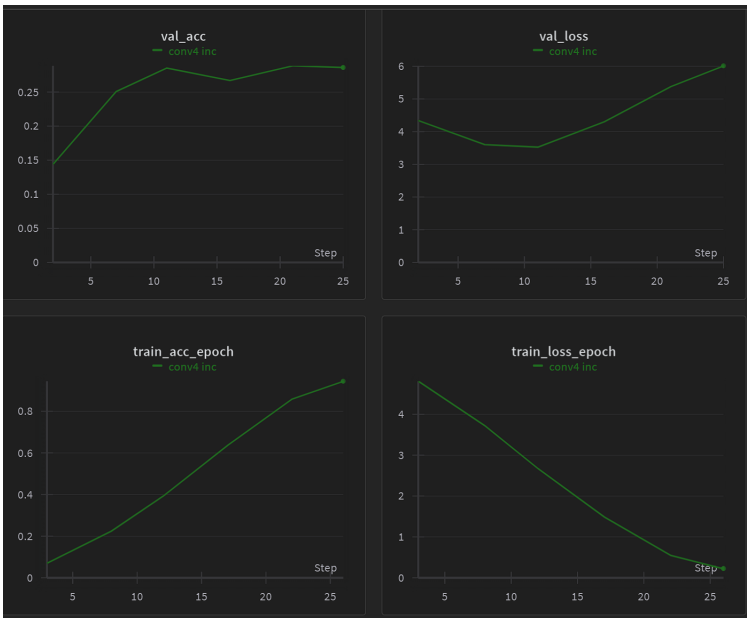
2.2.1 Az adatok feldolgozásának pipeline-ja

- Audio fájlok és a meta adatok kinyerése és párosítása az adathalmaz osztályba
- Audio augmentációk alkalmazása
- Audio transzformációk alkalmazása
- Hanghullám Mel Spectrogram-má alakítása
- Mel Spectrogram transzformációk alkalmazása

2.3.1 Baseline modell és tanítása

A kiinduló baseline modellünk definiálását és tanítását a training.ipynb notebookben valósítottuk meg. A modellt egy általunk egyénileg definiált konvolúciós hálót választottunk, melyet az *OurCustomModel* osztályba implementáltunk. A konvolúciós modell elsőként három konvolúciós réteget, valamint két fully connected réteget tartalmazott. Az első konvolúciós réteg esetében külön ügyelnünk kellett arra, hogy a bemeneti csatornák számát egy értékre állítsuk, mivel a bemeneten lévő Mel Spectogram-okat monokróm képekként értelmeztük. Továbbá figyelembe kellett vennünk, hogy az utolsó fully connected réteg kimenete a mi általunk elvárt osztályok tartományába legyen, tehát az esetünkben ez a 264, amely a madárfajok one-hot enkódolt osztályának mennyiségét jelöli.
A modell tanításának egységét az *AudioClassifier* osztály implementációjával valósítottuk meg, benne realizáltuk a PyTorch Lightning-ből jövő LightningModule interfészt, így benne a *forward* metódus, valamint a *trainin_step* és a *validation_step* és a *configure_optimizers* metódusok találhatóak meg.

A modell tanításának egybefogó egységét a *train* metódusban valósítottuk meg, hol különböző tanítási metodikákat is bevettünk a modell tanulásának javítása érdekében. A hiperparamétereket a Weights and Biases által adott sweep lehetőséggel automatikusan optimalizáltuk a bayes módszer szerint. A tanítás monitorozásához külön létrehoztunk egy *deepbirding* nevű projektet a Weights and Biases platformon belül, hogy a csapat össze tagja ezen projekten belül dolgozzon a félév során. Többek között a baseline modellünk *learning rate, optimizer, batch size, weight decaying és convolutional filter width* paramétereit optimalizáltuk, melyek értékeinek skáláját a tanítások során manuálisan is igazítottuk. A baseline modellünk komplexitását inkrementálisan is bővítettük a tanítások során. Először három kovolúciós réteggel tanítottunk, majd úgy láttuk, hogy egy 4. konvolúciós réteg bevezetése is szükségesség vált, ezzel magasabb pontosságot tudtunk elérni. A következő ábrákon látható, hogy a bővített baseline modell tanítása során hogyan alakultak a tanítási és validációs loss és accuracy értékei az epochok során.

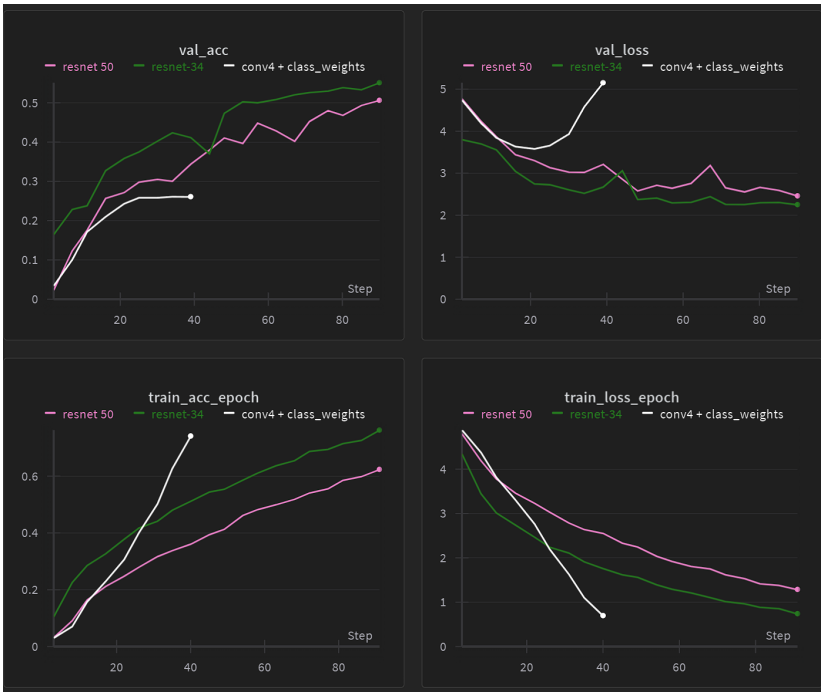


Az ábráról leolvasható továbbá, hogy a modell tanulása során az overfitting jelenségét produkálja. Az overfitting azt jelenti, hogy a modell túlságosan alkalmazkodik a tanító adatokhoz, és nem általánosít jól az új, nem látott adatokra. Az ábrán látszik, hogy a training loss folyamatosan csökken, azonban a validation loss egy idő után növekedésnek indul el.

2.3.2 Alkalmazott training metodikák

Az overfitting és a tanítás teljesítményének növelése érdekében különböző tanítási módszereket alkalmaztunk a modell tanításának során. Implementáltuk az early stopping mechanizmust, mellyel 3 epoch-ig türelemmel várjuk, hogy a hálózat a pontosságban növekedést produkáljon, különben leállítjuk a tanítást. Ez azért is egy hasznos mechanizmus, mivel a Kaggle környezetben korlátozott a GPU erőforrás ideje, így nem vesztegetünk időt azon modell tanításával, mely már valószínűleg overfitt-ben “szenved”.
A következő módszer a dropout technika bevezetése volt, mely során a modell architektúrájában definiálunk dropout pontokat, melyek során a neurális hálózat lefagyasztja a súlyainak egy részét, így az általánosításban vélhetően jobb eredményt fogunk elérni. Sajnálattal tapasztaltuk, hogy a modell a magas dropout érték bevezetése után tanulása során többször is instabillá vált, valamint alacsony értékben nem eredményezett további javulást a pontosság és a loss terén. Az overfitting jelenség kiküszöbölésére és a pontosság növelése érdekében továbbá próbálkoztunk azzal a módszerrel is, hogy az eredetileg 5 másodperces darabokban érkező bemeneti hanganyag hosszát 10 mp-re növeljük, azonban ez sem hozott jelentős előrelépést a tanítás során.
A versenyre benyújtott vezető megoldások közül felfigyeltünk egy technikára [4], melyet a kiegyensúlyozatlan adathalmazokkal való tanítás során szokás felhasználni, mégpedig az osztály szerinti súlyozás. Ezzel a módszerrel úgy próbáljuk kompenzálni a kiegyensúlyozatlan adathalmazon tanuló modellt, hogy az adathalmazban kisebb előfordulással rendelkező osztályokhoz nagyobb súlyt társítunk, így elősegítve, hogy a modell ezen osztályokhoz nagyobb hangsúlyt fektessen a tanulás során. Ezen módszer felhasználása sem segített azonban nagy előrelépést eredményezni.
A nagy előrelépést azonban baseline modellünkön adatok augmentációja jelentette, melyet már az adatok feldolgozásánál részleteztünk. Az augmentációk segítségével mind az overfitting jelenségét, mind a modell pontosságát javítani tudtuk, mellyel a 0.28 validációs pontosság helyett 0.354 pontosságot sikerült elérnünk.
A további pontosság növeléséhez úgy gondoltuk, hogy az inkrementált komplexitású baseline modellünktől meg kell válnunk, így a választásunk egy igen népszerű konvolúciós klasszifikációs modellre, a ResNet modell családra [5], azon belül is ResNet-34-re esett előtanított súlyokkal. Ezen modellel 0.5366 pontosságot sikerült elérnünk a validációs adathalmazon, mely messze felülmúlta a baseline modellünket. Ezután még egy modell kipróbálásra került sor, mely a ResNet-50 névre hallgat, szintén pretrained súlyokkal, azonban

ezen modell nem teljesítette túl a 34-es elődjét. A következő ábrán láthatjuk az egyes modellek összehasonlítását a tanítás során.



2.3.3 További tanítási módszerek

A tanítás idejét meggyorsíthatjuk, ha a tanítás során párhuzamossá tesszük a adat batch-eken való tanulást, ezt a PyTorch Lightning környezetben könnyen meg is lehet valósítani, sajnos azonban, mivel a Kaggle virtuális környezetben futtattuk a tanítást, sokszor kaptunk kivételeket a párhuzamosítás során, így végül ezt az ötletünket elvetettük.

A tanítás során szintén még egy módszert szándékunkba volt beiktatni, mégpedig a cosine annealing learning módszerét, melyben a learning rate értékét egy koszinusos függvény segítségével egy bizonyos idő után újra növeljük, azonban ez a tanuló modellben instabilitást okozott, így ezen ötletünket is végül elvetettük.

A tantárgy egyik gyakorlata során megismerhettünk egy módszerrel mellyel a modellen elvégzett *torch.compile* metódussal a tanítás során időbeli hatékonyság növelését fedezhettük fel. Sajnos azonban ez a metódus legalább CUDA 7.0 verziószámot követel, a Kaggle felületen a GPU viszont csak CUDA 6.0-ás verziója érhető el, így ezen ötletet is végül elvetettük.

3. Konténerizáció

A feladatunk jellegéből és a környezetéből adódóan a konténerizáció egy kezdetleges megoldást dolgoztunk ki, mely abban merült ki, hogy létrehoztunk egy *requirement.txt* fájlt, melyben a pipeline futtatásához az összes felhasználandó könyvtárat, és azok verzióját tüntettük fel. Létrehoztunk továbbá egy *Dockerfile* nevű fájlt, mely segítségével már létre lehet hozni egy reprodukálható környezet a projekt számára.

4. Összefoglalás, megoldás kiértékelése

A modellünk és munkánk kiértékelésére két módszert is választottunk, az egyik a teszt adathalmazon különböző metrikák alapján elért eredmények kimenete.

Az első módszert implementációját az evaluation.ipynb notebook-ban találhatjuk meg. A teszthez a training fázisa során statikusan kimentett test_indicies fájl betöltése szükséges, mivel a teszt adathalmazon mindenképpen nem látott mintákkal szeretnénk kiértékelni a modell teljesítményét. A legjobb modell súlyait a WandB felületről töltjük le, majd a példányosított modell súlyait a tanult súlyokra cseréljük ki.

A tanulás során legjobban teljesítő modellünket ezután teszteljük a teszt adathalmazon, általa elért pontosság a teszt adathalmazon a következő ábrán látható eredményt adta:

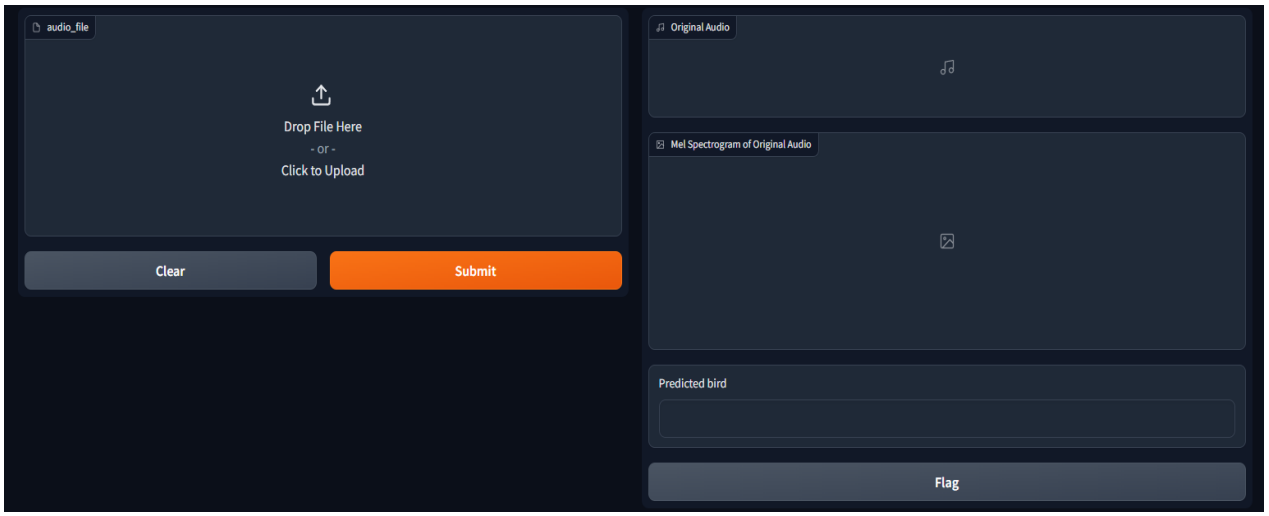
Test metric	DataLoader 0
test_acc	0.5402201414108276
test_loss	2.431685447692871

Látható hogy a legjobb modellünk a teszt adathalmazon pontosság terén 0.5402 eredményt ért el, ami azt mutatja, hogy a modell valóban megtanulta a mintákat és az összefüggéseket a tanítási adathalmazon. Ugyanezen modell 0.398-as recall, továbbá 0.3197-es F1-score értéket ért el. Az eredményünk véleményünk szerint egy meglehetősen jó eredmény, ha azt vesszük alapul, hogy a modellünknek 264 osztály közül kellett multinomális osztályozást elvégeznie, egy 100%-os pontosságú klasszifikációs találatra ~0.00378 volt a valószínűség, míg egy bináris osztályozásnál ezzel szemben 0.5 ugyanennek az esélye. Természetesen a pontosság tovább növelésére még ez után is van lehetőség, újabb tanítási metodikák bevezetése, model ensemble-vel való kiértékelés valószínűsíthetően növelné a teszt adathalmazon nyújtott pontosságot.

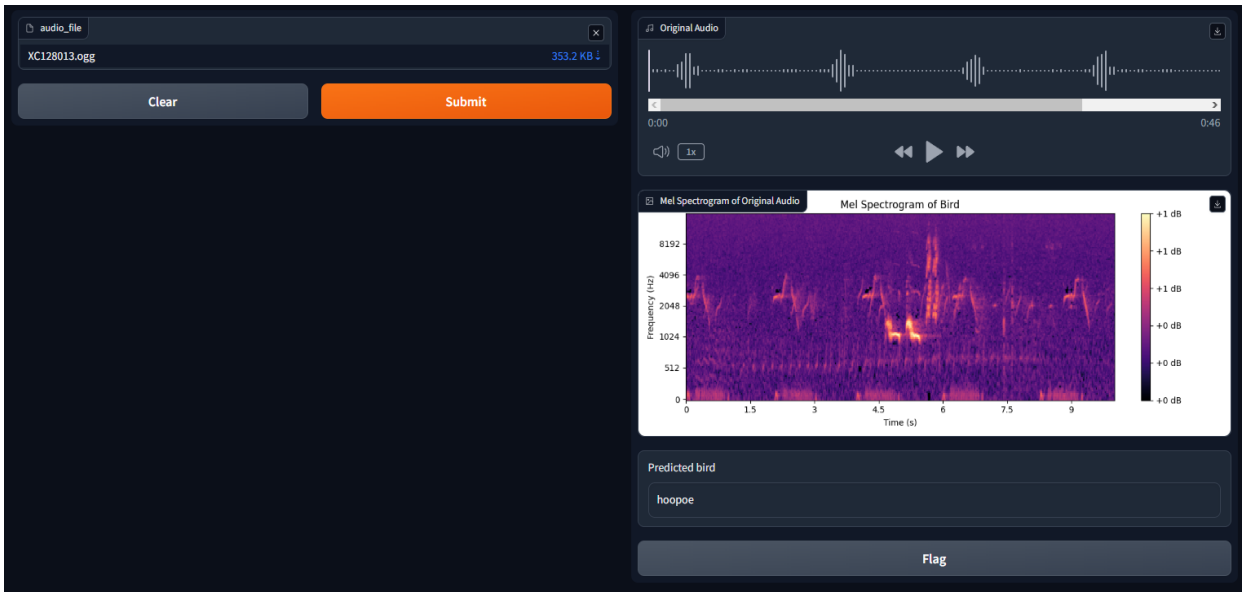
Ezen kiértékelési módszerhez készítettünk továbbá egy konfúziós mátrixot is, amelyen viszonylag jól látható, hogy mennyi osztályt tévesztett a modellünk a teszt adathalmazon. Annak ellenére, hogy 264 classunk van a konfúziós mátrix hasznos, mivel a cél, hogy minden elem a mátrix főátlójába essen, ezt egy hőtérkép módszerrel ábrázoltuk. A konfúziós mátrix eredeti felbontásában megtekinthető a projekt results könytárán belül.

4.1 UI Komponens az interaktív kiértékeléshez

A másik módszerünk egy interaktív kiértékelési módszer, melyben a felhasználó egy UI felületen, amit a Gradio-val valósítottunk meg az eval_interactive.ipynb Jupyter notebook-ban. Ezen a felületen keresztül egy madár hang audiófelvétel anyagot tud betölteni a rendszerbe a felhasználó, majd ezen az audió fájlon a Wandb-ből letöltött legjobban teljesítő modellünk egy predikciót hajt végre, és kimenetben a felismert madár fajának nevét írja ki számunkra. Továbbá a felhasználó a UI felületen akár a hang anyagot le is tudja játszani, valamint a hangfájl a modellben való átalakítására is betekintést tud nyerni, tehát a rendszer szemlélteti az éppen betöltött audio fájl Mel Spectrogram-má alakított reprezentációját. A felhasználó felület megvalósításához a Gradiót használtuk fel.
Egy példa fájlal a UI komponens működése:



A fenti képen látható, hogy a felhasználónak lehetősége van kiválasztani egy audió fájlt, amit utána feltölt a notebookba a “Submit” gomb lenyomásával.



Miután a felhasználó rákattint a “Submit” gombra a modellünk ad egy predictiont, ami a “Predicted Bird” textbox-ban látható. Ezen kívül még kirajzolja a tanítás során használt Mel Spectrogramot és lehetősge van lejátszani akár az audiót is és össze is vetheti a Mel Spectrogrammal. Az adatok nehéz elérhetősége miatt a GitHub repositoryba egy mappába feltöltöttünk pár audió fájlt a UI felület teszteléséhez, amelyeket betöltve egyszerűen ki lehet próbálni a komponenst és a modell pontosságát. A teszteléshez a fájlok a “*DeepBirding_test_audio*” nevű mappában találhatóak, hogy a tesztelés teljes körű legyen ezért azt, hogy melyik audió fájl melyik madárhoz tartozik ezt egy “*original_bird_to_filename_mapping.txt*” fájl tartalmazza. Így könnyen validálható, hogy tényleg jó a predikció.

Felhasznált irodalom

[1] BirdCLEF 2023 competition on Kaggle, Elérhető: https://www.kaggle.com/competitions/birdclef-2023

[2] PyTorch Lightning, Elérhető: https://lightning.ai/

[3] “Audio Deep Learning Made Simple (Part 2): Why Mel Spectrograms perform better”, Elérhető: https://towardsdatascience.com/audio-deep-learning-made-simple-part-2-why-mel-spectrograms-perform-better-aad889a93505

[4] “1st place solution: Correct Data is All You Need”, Elérhető: https://www.kaggle.com/competitions/birdclef-2023/discussion/412808

[5] “Deep Residual Learning for Image Recognition”, Elérhető: https://arxiv.org/abs/1512.03385