TEHNIČKA DOKUMENTACIJA

Audio Classification

LUMEN Data Science

2023

Tim **Tamburaši:**

Tin Josip Čurik

Marko Haralović

SADRŽAJ DOKUMENTACIJE

1. SAŽETAK
2. UVOD
3. OPIS RJEŠENJA

3.1.ARHITEKTURA MODELA

3.2. PSEUDOKOD

3.3. FUNKCIONALNA SPECIFIKACIJA

3.4. PROTOK INFORMACIJA

4.ORGANIZACIJA IZVORNOG KODA

5. NASTAVAK RADA

6. INSTALACIJSKI PREDUVJETI 7. API

8. OPIS KORIŠTENJA ZA KORISNIKA

APPENDIX – LISTA AKRONIMA

1.SAŽETAK

U našem slučaju zadatak je bio klasifikacija polifonijskih audio zapisa. Dan nam je IRMAS dataset koji u sebi sadrži sljedećih 11 klasa instrumenata: cel, cla, sax, voi, vio, tru, gac, gel, org, flu i pia. Na istome datasetu obavljen je trening modela i validacija istog. Model koji smo koristili bio je konvolucijski model neuronskih mreža kojemu smo kao input slali različite metrike zvuka koje smo izvlačili iz audio zapisa. Te metrrike su redom : mel spektrogrami, mel-frekvencijski cepstralni koeficijenti te spektralni kontrast. Dataset nije bio poslan kao sirov podatak niti je izvlačenje prethodnih metrika obavljeno na originalnim duljina dataseta, odnosno audio datoteka; svaki audio je zapis po posebnim konstantama podijeljen u segmente od jedne sekunde, generirani su podatci od interesa, dodani u numpy polje, generirane su labele podataka na temelju klasa pripadnosti te su slane kao input CNN-u. Nadalje, koristi se metoda hot encodinga čime se model izravno uči pogađati višeinstrumentalne audio zapise. Za učitavanje i obradu audio zapisa koristi se librosa library , izrada CNN modela je u tensorflow libraryu te je naposlijetku, po razvitku modlea, isti spojen na docker kontenjer tensorflow servinga, nekoliko modela su spojeni na siti te je razvijena web stranica na kojoj je omogućen prijenos aduio filea, slanje istog, čime se poziva model koji obrađuje audio zapis te generira predickije, koje se zatim šalju nazad na web poslužitelj te se iste ispisuju na web stranici. Razvijene su metode optimizacije tresholda za predikciju instrumenta, kao i odvojeni model koji služi za predikciju broja instrumenata u svakom audio zapisu. Dodatno, podaci IRMAS-a su augmentani kako bi se postigla robusnost koda/modela te kako bi model mogao učiti na raznim, više instrumentalnim kombinacijama instrumenata koje nisu sadržane u samome IRMAS datasetu. Naposlijetku, model koji je razvijen na IRMAS-u postiže state of the art rezultate.

KLJUČNE RIJEČI: audio procesurianje, polifonija, konvolucijske neuronske mreže, povećanje podataka, IRMAS dataset, librosa, tensorflow, tensorflow serving, docker kontenjeri.

2. UVOD

2.1. LIBROSA BIBLIOTEKA

U našem smo projektu koristili python library Librosa, koja je Python biblioteka za analizu zvuka i glazbe, kao takva se često koristi u raznim zadacima vezanim uz glazbu i neku generalnu obradu zvuka . Librosa nudi razne funkcije za učitavanje, pretprocesurianje i ekstrakciju značajki iz audio signala, kao što su mel spektrogrami, MFCC (MEL-frequency cepstral coefficients), FFT ( brza Fourierova transofrmacija), kromagrami,spektralni kontrasti, BPM (beats per minute) i mnogi drugi. Naravno, uz prethodno navedene funckije, odnsosno značajke koje smo izvlačili i koristili, govorimo i o tome kako nam Librosa nudi funckije za obradu zvuka kao što je promjena visine tona, vremensko sužavanje i rastezanje zvuka, broj uzoraka zvuka, redukcije šuma i drugi, što nam je koristilo prilikom „čišćenja“ podataka , odnosno audio zapisa te obradu istih, koje smo zatim kao numpy polja lagano mogli slati u model te ga trenirati na taj način.

2.2. ZNAČAJKE ZVUKA

2.2.1. MEL SPEKTROGRAM

Mel spektrogram je vizualni prikaz kratkotrajnog spektra snage zvuka , pri čemu se frekvencije skaliraju u Mel skali. Mel skala je perceptualna skala frekvencija koja je dizajnirana kako bi bolje odgovarala ljudskoj percepciji zvuka. Mel je spektrogram uobičajeno korišten za analizu govora i zvuka , posebice za prepoznavanje instrumenata i za žanrovsku klasifikaciju.

2.2.2. KROMAGRAM

Kromagram je vizualni prikaz različitih visina tonova u glazbenom signalu, raspoređenih u 12 binova koji predstavljaju 12 polutonova u jednoj oktavi. Ovaj prikaz koristi se za analizu harmonijskih struktura glazbe i prepoznavanje akorda, tonaliteta i ključeva. Kromagram može pomoći u identifikaciji glazbenog sadržaja i analizi sličnosti između različitih glazbenih djela.

2.2.3. MFCCs (Mel-frekvencijski cepstralni koeficijenti):

MFCCs su značajke koje se široko koriste u obradi govora, prepoznavanju govornika i prepoznavanju glazbe. Ovi koeficijenti predstavljaju kratak opis spektralne omotnice zvuka. MFCCs se dobivaju transformacijom spektra snage u Mel domenu, a zatim se koristi diskretna kosinusna transformacija kako bi se dobila reducirana reprezentacija spektra. MFCCs mogu karakterizirati različite aspekte zvuka, kao što su boja, visina tona i intenzitet.

2.2.4.SPEKTRALNI KONTRAST

Spektralni kontrast je mjera razlike u energiji između vrhunaca (visokoenergetskih frekvencija) i dolina (niskoenergetskih frekvencija) u spektru snage zvuka. Spektralni kontrast se koristi za razlikovanje različitih zvukova, posebno za prepoznavanje instrumenata i glazbenih žanrova. Ova značajka može pomoći u analizi teksture zvuka i karakterizaciji različitih vrsta zvukova.

2.3. KONVOLUCIJKA NEURONSKA MREŽA

CNN-ovi su posebna vrsta dubokih neuronskih mreža inspirirana biološkim procesima vizualnog korteksa. Glavna značajka konvolucijskih mreža su konvolucijski slojevi koji omogućuju mreži da prepoznaje lokalne značajke i njihove prostorne veze u podacima. To ih čini izvrsnima za obradu slika i zvuka, gdje postoji prirodna prostorna struktura. Tijekom treninga, CNN uči hijerarhiju značajki, pri čemu niži slojevi detektiraju jednostavnije značajke, a viši slojevi kombiniraju te značajke u složenije obrasce. U obradi zvuka, CNN-ovi su često korišteni za klasifikaciju žanrova, klasifikaciju instrumenata, prepoznavanje govora, i drugo.

2.4. GITHUB REPOZITORIJ

Kreiran je javni github repozitorij <https://github.com/MarkoHaralovic> preko kojega se obavljao prijenos podataka, koda, promjena i ostalog. Ova je verzija novija verzija prethodno privatnog repozitorija, koji je očišćen nepotrebnih podataka i kodova te koji sadrži „čistu“ verziju rješenja i koda. U README.md fileu nalazi se opis koda/projekta, a za pokretanje rješenja , odnosno API-ja, potrebno je pročitati datoteku user\_manual.txt u kojemu je opisan rad s kodom, pokretanje API-ja , odnosno web stranice, pokretanje docker kontenjera te ostale nužne instalacije za ostvarenje istoga.

2.5. IRMAS dataset

IRMAS (Instrument Recognition in Musical Audio Signals) dataset je zbirka glazbenih isječaka dizajnirana posebno za prepoznavanje glazbenih instrumenata. Dataset je razvijen za evaluaciju automatskog prepoznavanja monofonih glazbenih instrumenata u polifoničkim zvučnim zapisima. Glavni cilj ovog skupa podataka je pružiti istraživačima resurs za testiranje i poboljšanje algoritama prepoznavanja instrumenata.

2.5.1. IRMAS trening skup

Sadrži 6705 zvučnih isječaka podijeljenih u jedanaest datoteka,gdje svaka datoteka ima zvučne isječke koji predstavljaju jedan od 11 različitih glazbenih instrumenata. Instrumenti uključuju klavir, gitaru, flautu, klarinet, saksofon, trubu, violinu, violončelo, kontrabas, akustičnu bas-gitaru i električnu bas-gitaru. Zvučni isječci su kratki (3 do 5 sekundi) i mogu sadržavati jedan ili više instrumenata koji sviraju istovremeno. Naravno, iako ovo jest trening set, moram napomenuti kako naš kod nije treniran na ovim podacima, kako je arhitektura rješenja upravo takva da se ovi zapisi koji varijabilno traju nekoliko sekundi podijele na segmente od po jedne sekunde te kao takvi predprocesuiraju i šalju u model.

2.5.2. IRMAS validacijski skup

Ovaj skup podataka sadrži 2874 zvučnih isječaka s informacijama o prisutnim instrumentima u obliku tekstualnih datoteka. Instrumenti u ovom skupu podataka su isti kao u trening skupu, ali udio pojedinog instrumenta u zbirki je drugačiji. Način obrade isti je kao i u trening skupu: poslani audio zapis se segmentira i obradi, svaki obrađeni komadić zapisa šalje se modelu koji zatim vraća klasu tog segmenta te tako za svaki zaseban dio, a zatim se agreacijskom funckijom dobio optimalni rezultat, odnosno predviđanje za audio file. Jasno, u ovome se segmentiranju pretpostavlja većinska prisutnost svakog instrumenta te se tom pretpostavkom dolazi do predviđanja.

2.6. PROŠIRENJE DATASETA

2.6.1. OPENMIC DATASET

OPENMIC (Open Music Instrument Classification) dataset je zbirka glazbenih isječaka namijenjenih za prepoznavanje glazbenih instrumenata. Ovaj dataset je rezultat kolaborativnog projekta koji uključuje Freenode #music s ciljem razvoja otvorenih izvora podataka za istraživanje glazbene analitike i strojnog učenja.

Dataset sadrži više od 20.000 zvučnih isječaka duljine 3 sekunde iz različitih glazbenih žanrova i stilova. Svaki isječak ima oznaku koja odražava prisutnost ili odsutnost 20 različitih glazbenih instrumenata, uključujući gitaru, klavir, bubnjeve, saksofon, trubu, violinu i druge. To znači da OPENMIC dataset ima veći broj instrumenata u usporedbi s IRMAS datasetom.

Unatoč velikoj količini podataka, označavanje instrumenata u OPENMIC datasetu možda nije savršeno. Označavanje podataka temelji se na kombinaciji strojnog učenja i ljudskih procjena, što može dovesti do netočnosti u oznakama. Strojni modeli možda nisu u potpunosti točni u prepoznavanju instrumenata, a ljudske procjene također mogu biti subjektivne ili neprecizne.

Osim toga, način na koji su podaci organizirani i označeni u OPENMIC datasetu možda nije idealan za proširenje IRMAS dataset-a. Kao što je već spomenuto, OPENMIC koristi binarne oznake za prisutnost ili odsutnost instrumenata, što znači da za svaki instrument postoji zasebna oznaka. U IRMAS datasetu, instrumenata ima manje, a svaki isječak ima jedinstvenu oznaku koja predstavlja instrument ili kombinaciju instrumenata. Različiti pristupi označavanju mogu otežati spajanje ili proširenje oba dataseta.

Tako , OPENMIC dataset pruža veliku količinu podataka i širi spektar instrumenata u usporedbi s IRMAS datasetom, ali zbog netočnosti u označavanju i razlike u organizaciji podataka te dodavanjem njegovih podataka u model, uvidjeli smo kako ovakvo proširenje nema smisla te se gubi na vrijednosti podataka koje imamo, a sami je model neprecizniji.

2.6.2 DATA AUGMENTING

Ova metoda proširenja podataka je tehnika koja se koristi u području strojnog učenja, posebice u dubokom učenju, s ciljem povećanja količine podataka za treniranje. Data augmenting uključuje generiranje novih podataka na temelju postojećih podataka pomoću različitih tehnika, kao što su rotacija, zumiranje, šum i promjene boje za slike ili dodavanje šuma, promjena visine tona i brzine zvuka za zvuk. Cilj proširenja podataka je poboljšanje performansi modela i smanjenje problema prenaučenosti. Naravno, u našemu slučaju dodaje se na robusnosti kodu i predikcijama, kako se sada model mogao učiti na zapisima koji sadrže i do 5 ili 6 instrumenata u sebi, što je daleko više negoli je iti jedan zapis IRMAS-a sadržavao. Poznate su tehnike nasumičnog miksanja, pitch.sync miksanja,tempo-sync miksanja te miksanja po sličnosti žanra. U našemu slučaju, govorili smo o kombinaciji nasumičnog miksanja, u procesu gdje jednostavno izvučene podatke konkatenirano. Opis metode slijedi u sekciji opisa koda.

2.6.3. TENSORFLOW SERVING

TensorFlow Serving je open-source softver za posluživanje modela strojnog učenja putem mreže. Omogućuje razvijanje i upotrebu skalabilnih, fleksibilnih i brzih sustava za posluživanje modela koji se mogu koristiti u produkcijskim okruženjima.

TensorFlow Serving može raditi s različitim vrstama modela, uključujući duboke neuronske mreže te oogućuje jednostavno dodavanje novih modela, ažuriranje i brisanje modela, uz minimalan prekid rada sustava. TensorFlow Serving također pruža mogućnost upravljanja verzijama modela i različitim strategijama posluživanja modela,što je bio upravo razlok zašto je pokrenut Deocker kontenjer tfserving preko kojega su spojeni različiti ponajbolji CNN modeli koje smo razvijali te preko koga su vršena i spajanja na web API.

2.6.4 DOCKER (KONTENJERI)

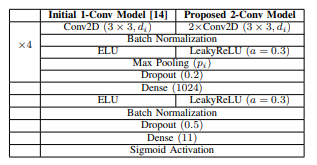
Docker je platforma za upravljanje kontejnerima, što su lagane i samodostatne izolirane omotnice za izvođenje aplikacija i njihovih ovisnosti. Kontejneri su slični virtualnim strojevima, ali umjesto da emuliraju cijeli operacijski sustav, dijele kernel domaćina i izvršavaju se izolirano od ostatka sustava. Svaki kontejner ima svoj set datoteka i biblioteka koji su potrebni za izvršavanje aplikacije, čineći ih vrlo fleksibilnim za korištenje i prenosive između različitih računalnih okruženja.

Docker kontejneri se mogu izgraditi na temelju definicija poznatih kao Dockerfile, koje specificiraju koje će se datoteke i biblioteke koristiti za aplikaciju. Nakon što se Dockerfile definira, moguće je generirati Docker kontejnere za izvršavanje aplikacija. Kontejneri se mogu pokrenuti na bilo kojem računalu s Dockerom instaliranim, bez obzira na operacijski sustav i ostale instalirane programe, čime nam je omogućena prenosivost, učinkovitost i skalabilnost.

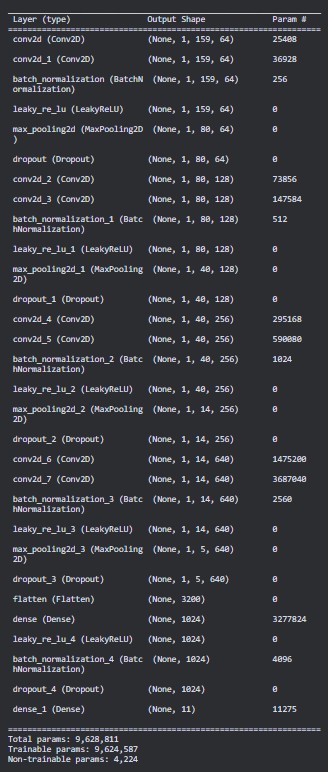
3.OPIS RJEŠENJA

Koristeći biblioteku librosa, učitaj audio zapis. Provuci audio zapis kroz skriptu za predprocesuiranje te izvuci podatke u obliku numpy polja te korespondentnih hot encoded labela (hot encoding je metoda kojemu se u polju duljine n, gdje n označava broj klasa, u našem slučaju 11, postavi 1 na pozicijski predefinirana mjesta ovisno o tome je li određena klasa pristuna u zapisu, odnosno 0 ako nije) , pošalji podatke u razvijeni konvolucijski model, optimiziraj hiperparametre modela (broj slojeva, broje neurona po slojevima, input shape podataka, learning rate, optimizer, metrika validacije, early stopping metrika, implementacija vlastitog callbacka, itd.), treniraj model, spremi ga u obliku h5 filea. Nakon što je model gotov, validiraj ga te spoji na tensorflow serving docker kontenjer, uz interne metode push i request razvij Flusk app.py , razvij HTML/CSS/JS stranicu te poveži s Flusk aplikacijom te pokreni lokalnim serverom, omogući upload audio zapisa, pošalji isti modelu, obradi i vrati predikcije, ispiši predikcije.

* 1. ARHITEKTURA MODELA



Model se sastoji od četiri bloka koji se sastoje od dva Conv2D sloja, BatchNormalization, LeakyReLU, MaxPooling2D i Dropout slojeva. Broj filtera u Conv2D slojevima raste s dubinom mreže, a veličina pool\_size u MaxPooling2D slojevima se mijenja nakon prvog bloka. Nakon četiri bloka, mreža koristi Flatten sloj kako bi pretvorila višedimenzionalne značajke u 1D tenzor, koji se zatim koristi kao ulaz za Dense sloj s 1024 jedinica. Mreža koristi LeakyReLU aktivaciju s alpha = 0.3 kako bi se spriječile mrtve ReLU jedinice, a BatchNormalization i Dropout slojevi dodani su za regulaciju mreže i sprječavanje prenaučenosti. Na kraju, model ima Dense sloj s 11 jedinica koji koristi sigmoidnu aktivacijsku funkciju kako bi se predvidjela vjerojatnost prisutnosti svakog od 11 instrumenata u ulaznom podatku.



3.2.PSEUDOKOD

3.2.1. PSEUDOKOD ZA PREDPROCESURIANJE

1. Uvezi potrebne biblioteke (os, numpy, librosa, tqdm)

2. Definiraj konstante (SAMPLE\_RATE, BLOCK\_SIZE, HOP\_SIZE, MEL\_BANDS, DURATION)

3. Definiraj funkciju parse\_label s ulazom label\_path

a. Otvori datoteku label\_path za čitanje

b. Pročitaj sadržaj datoteke i ukloni praznine s početka i kraja

c. Razdvoji sadržaj po tabulatoru i vrati rezultirajući popis instrumenata

4. Definiraj funkciju preprocess\_audio s ulazom audio\_path

a. Učitaj audio datoteku na audio\_path s određenom brzinom uzorkovanja i u mono formatu

b. Pretvori stereo audio u mono

c. Normaliziraj amplitudu zvuka

d. Podijeli audio u jednosekundne intervale

e. Za svaki audio segment izračunaj njegov kratkotrajni Fourierov transform i mel spektrogram

f. Pretvori mel spektrogram u decibelske jedinice i dodaj na popis

g. Vrati popis log mel spektrograma

5. Definiraj funkciju process\_audio\_files s ulazom input\_dir

a. Inicijaliziraj prazne liste X i y

b. Za svaki korijen, direktorij i datoteku u input\_dir

i. Dohvati popis audio datoteka i datoteka s oznakama u trenutnom direktoriju

ii. Za svaku audio datoteku pronađi njezinu odgovarajuću datoteku s oznakama

iii. Ako je pronađena datoteka s oznakama, analiziraj datoteku s oznakama i prethodno obradi audio datoteku

1. Dodaj rezultirajuće log mel spektrograme u X i oznaku u y

iv. Ako datoteka s oznakama nije pronađena, ispiši poruku o pogrešci

c. Pretvori liste X i y u numpy nizove i vrati ih

6. Glavni program

a. Pozovi funkciju process\_audio\_files s putanjom ulaznog direktorija

b. Spremi rezultirajuće X i y nizove na disk

3.2.2.PSEUDOKOD ZA AUGMENTACIJU

1. Uvezi potrebne biblioteke (os, numpy, librosa, tqdm, random, re)

2. Definiraj konstante (SAMPLE\_RATE, BLOCK\_SIZE, HOP\_SIZE, MEL\_BANDS, DURATION, class\_names)

3. Definiraj funkciju spectral\_contrast s ulazom audio\_path

4. Definiraj funkciju chromagram s ulazom audio\_path

5. Definiraj funkciju parse\_instruments s ulazom file\_name

6. Definiraj funkciju check\_unique\_instruments s ulazima file1 i file2

7. Definiraj funkciju parse\_label s ulazom label\_path

8. Definiraj funkciju mel\_spectrogram s ulazom audio\_path

9. Definiraj funkciju one\_hot\_encode s ulazima labels i class\_names

10. Definiraj funkciju process\_audio\_files s ulazima input\_dir i num\_iterations

a. Inicijaliziraj prazne liste X i y

b. Uz pomoć for petlje i funkcije tqdm obradi num\_iterations iteracija

i. Izaberi nasumične poddirektorije

ii. Pronađi i obradi audio datoteke iz tih poddirektorija

1. Izračunaj mel spektrogram, kromagram i spektralni kontrast

2. Spremi rezultate u X

3. Dohvati i kodiraj instrumente za svaku datoteku

4. Spremi rezultate u y

c. Pretvori liste X i y u numpy nizove i vrati ih

11. Glavni program

a. Pozovi funkciju process\_audio\_files s putanjom ulaznog direktorija

b. Spremi rezultirajuće X, y i z nizove na disk

3.2.3. PSEUDOKOD ZA OCJENU KVALITETE MODELA

1. Uvezi potrebne biblioteke (keras, numpy, matplotlib, librosa, tensorflow\_addons)

2. Učitaj X i y iz numpy datoteka

3. Učitaj model iz datoteke

4. Inicijaliziraj varijable tp (true positives), fp (false positives) i fn (false negatives)

5. Za svaki i-ti element X s korakom 10:

a. Preoblikuj X[i] dodavanjem dimenzije i konvertiranjem u float32

b. Inicijaliziraj niz non\_binary\_label s nulama

c. Za svaki j-ti element X[i]:

i. Izračunaj predikciju modela za X[i][j] i zbroji s non\_binary\_label

d. Normaliziraj non\_binary\_label

e. Inicijaliziraj pred\_binary\_label s nulama

f. Za svaki j-ti element non\_binary\_label:

i. Postavi pred\_binary\_label[j] na 1 ako non\_binary\_label[j] >= 0.5

ii. Ažuriraj tp, fp i fn ovisno o pred\_binary\_label[j] i y[i][j]

g. Izračunaj i ispiši F1 mjera (F1 = 2 \* P \* R / (P + R), gdje P = tp / (tp + fp), R = tp / (tp + fn))

3.2.4. PSEUDKOD ZA TOČNOST PREDIKCIJE MODELA

1. Uvezi potrebne biblioteke (numpy, librosa, os, glob, keras, scipy, tqdm, tensorflow\_addons, json)

2. Definiraj konstante (SAMPLE\_RATE, BLOCK\_SIZE, HOP\_SIZE, MEL\_BANDS, DURATION)

3. Definiraj funkciju preprocess\_audio(audio\_path):

a. Učitaj audio podatke iz audio\_path

b. Konvertiraj audio u monokanalni zvuk

c. Normaliziraj audio

d. Segmentiraj audio u intervale od jedne sekunde

e. Izračunaj log mel-spektrogram za svaki segment

f. Vrati listu log mel-spektrograma

4. Definiraj funkciju validation\_accuracy(file\_path\_txt, predictions, threshold=0.5):

a. Učitaj ground truth iz datoteke file\_path\_txt

b. Usporedi predictions s ground truth i izračunaj točnost

c. Vrati točnost

5. Definiraj funkciju process\_audio\_files(input\_dir):

a. Inicijaliziraj X kao praznu listu

b. Za svaku datoteku u input\_dir:

i. Ako je datoteka audio format, izračunaj log mel-spektrogram

ii. Dodaj log mel-spektrogram u X

c. Vrati X kao numpy niz

6. Definiraj funkciju predict\_windows(model, log\_mel\_spectrogram):

a. Vrati predikciju modela za log\_mel\_spectrogram

7. Definiraj funkciju aggregate\_predictions(predictions):

a. Sumiraj predictions

b. Vrati normalizirane sume

8. Definiraj funkciju test\_model(model, test\_data\_path, threshold=0.5):

a. Inicijaliziraj rezultate i točnosti

b. Definiraj listu instrumenata

c. Za svaku audio datoteku u test\_data\_path:

i. Izračunaj log mel-spektrogram

ii. Izračunaj predikcije za sve prozore

iii. Agregiraj predikcije

iv. Stvori rječnik instrumenata s vrijednostima

v. Izračunaj točnost predikcije i spremi ju u listu točnosti

vi. Ispiši predikcije i točnost za svaku datoteku

d. Izračunaj srednju točnost

e. Ispiši srednju točnost

f. Vrati rezultate

9. Učitaj model iz datoteke

10. Postavi test\_data\_path

11. Pokreni testiranje modela na testnim podacima

12. Spremi rezultate u JSON datoteku

3.2.5. PSEDUKOD ZA APLIKACIJU U FLASK-U

1. Uvezi potrebne biblioteke (requests, flask, os, tempfile, werkzeug, numpy, librosa, tensorflow.keras, scipy, tqdm, glob, json)

2. Definiraj konstante (SAMPLE\_RATE, BLOCK\_SIZE, HOP\_SIZE, MEL\_BANDS, DURATION)

3. Definiraj funkciju preprocess\_audio(audio\_path):

a. Učitaj audio podatke iz audio\_path

b. Konvertiraj audio u monokanalni zvuk

c. Normaliziraj audio

d. Segmentiraj audio u intervale od jedne sekunde

e. Izračunaj log mel-spektrogram za svaki segment

f. Vrati listu log mel-spektrograma

4. Definiraj funkciju predict\_windows(model, log\_mel\_spectrogram):

a. Dodaj dimenziju kanala ulaznim podacima

b. Pošalji zahtjev na TensorFlow serving API

c. Ako je uspješno, vrati predikciju, inače vrati None

5. Definiraj funkciju aggregate\_predictions(predictions):

a. Sumiraj predictions

b. Vrati normalizirane sume

6. Učitaj model

7. Inicijaliziraj Flask aplikaciju

8. Definiraj rute za Flask aplikaciju:

a. index() - prikaži početni HTML template

b. predict() - prihvati zahtjev za predikciju

i. Provjeri postojanje audio datoteke u zahtjevu

ii. Spremi audio datoteku u privremeni direktorij

iii. Preprocesiraj audio datoteku

iv. Obriši privremenu audio datoteku

v. Za svaki log mel-spektrogram, izračunaj predikcije

vi. Ako neuspješno, vrati grešku

vii. Agregiraj predikcije

viii. Odredi prag i pripremi rječnik instrumenata

ix. Postavi odgovarajuće instrumente u rječniku na 1

x. Vrati rječnik instrumenata kao JSON

9. Pokreni Flask aplikaciju

3.3.FUNKCIONALNA SPECIFIKACIJA

Model ima sljedeće značajke:

-Klasifikacija polifonije u zvučnim zapisima s vrhunskim rezultatima

-Podrška za različite arhitekture modela (hanCNN, gruraniCNN, kratimenosCNN)

-Normalizacija značajki ulaznih podataka

-Redovito spremanje modela tijekom treniranja

-Korištenje optimizatora Adam za brzo i učinkovito treniranje

-Korištenje metrike F1 za evaluaciju performansi modela

-Tehnike regulacije kao što su Dropout i BatchNormalization za bolju generalizaciju

-Mogućnost nastavka treniranja s prethodno spremljenim modelom

Krajnji korisnik može izvesti sljedeće zadatke sa sustavom:

-Trenirati model za klasifikaciju polifonije na zvučnim zapisima

-Prilagoditi arhitekturu modela i hiperparametre prema potrebama

-Učitati prethodno spremljeni model i nastaviti s treniranjem

-Evaluirati performanse modela na validacijskom skupu podataka

-Koristiti model za klasifikaciju polifonije na novim zvučnim zapisima

Osim toga, model podržava sljedeće programerske sučelja:

-Keras API za definiranje modela, kompilaciju, treniranje i evaluaciju

-TensorFlow Addons za dodatne metrike i optimizacije

-NumPy i scikit-learn za manipulaciju i pripremu podataka

-Docker slika TensorFlow Serving-a za jednostavno pokretanje i upravljanje modelom

-Lokalni server koji omogućuje pokretanje aplikacije putem app.py

-Web sučelje (index.html) za interakciju s korisnicima i omogućavanje slanja audio datoteka za analizu

-Mogućnost postavljanja audio datoteka putem web sučelja, nakon čega model analizira datoteku i vraća predikcije instrumenata

3.4. PROTOK INFORMACIJA

A picture containing text, diagram, plan, technical drawing

Description automatically generated

4.ORGANIZACIJA IZVORNOG KODA

5.NASTAVAK RADA

Trenutni modeli postižu state of the art rezultate pa je prilično teško komentirati koja bismo poboljšanja predlagali u tome smislu. Jasno, postoje razne metode koje bismo mogli implementirati i probati na stvarnome slučaju. Počevši od audio procesuiranja, postoje različite metode za segmentiranje audio zapisa, čišćenje šuma, oplemenjivanje podataka, ispravno izvlačenje podataka, pravilno proširenje seta augmentiranjem (koje jasno može biti po drugačijim metrikama te prateći drugačije kombiniranje od našega), proširenje seta drugim mogućim datasetom, moguća izrada vlastititog dataseta ukoliko je vlastoručno miksanje instrumenata reprezentativno u odnosu na stvarnu polifoniju ( ne smije se isključiti boja zvuka i umijeće skladanja). Jasno, jedna je od mogućnosti i transfer learning , odnosno preuzimanje gotovih modela koji bi se prilagodili našim podacima. Tu se jasno govori o IMAGENET-u ,densenet, inception i resnet-u (slično radu i pripadajućem git repozitoriju s linka: https://arxiv.org/abs/2007.11154). Jasno, sama konfiguracija mreže, kao i veličina dataseta limitirani su bili s našim tehnićkim specifikacijama laptopa. Jedan dio bio je pokrenut na CPU-u, ali trening mreže te procesuiranje je bilo odrađeno na GPU-u, no memorijski je problem bio prisutan, što bi se u daljnem radu moglo korigirati pa tako možebitno bolji rezultat bi mogao uslijediti ukoliko bi se pokrio takoreći „tehnički nedostatak“ koji je bio prisutan u ovome slučaju.

6.INSTALACIJSKI PREDUVJETI

Prije početka instalacije i pokretanja koda, morate imati sljedeće alate i biblioteke:

-Docker

-Visual Studio Code

-Pip 23.1.2 (Python Package Manager)

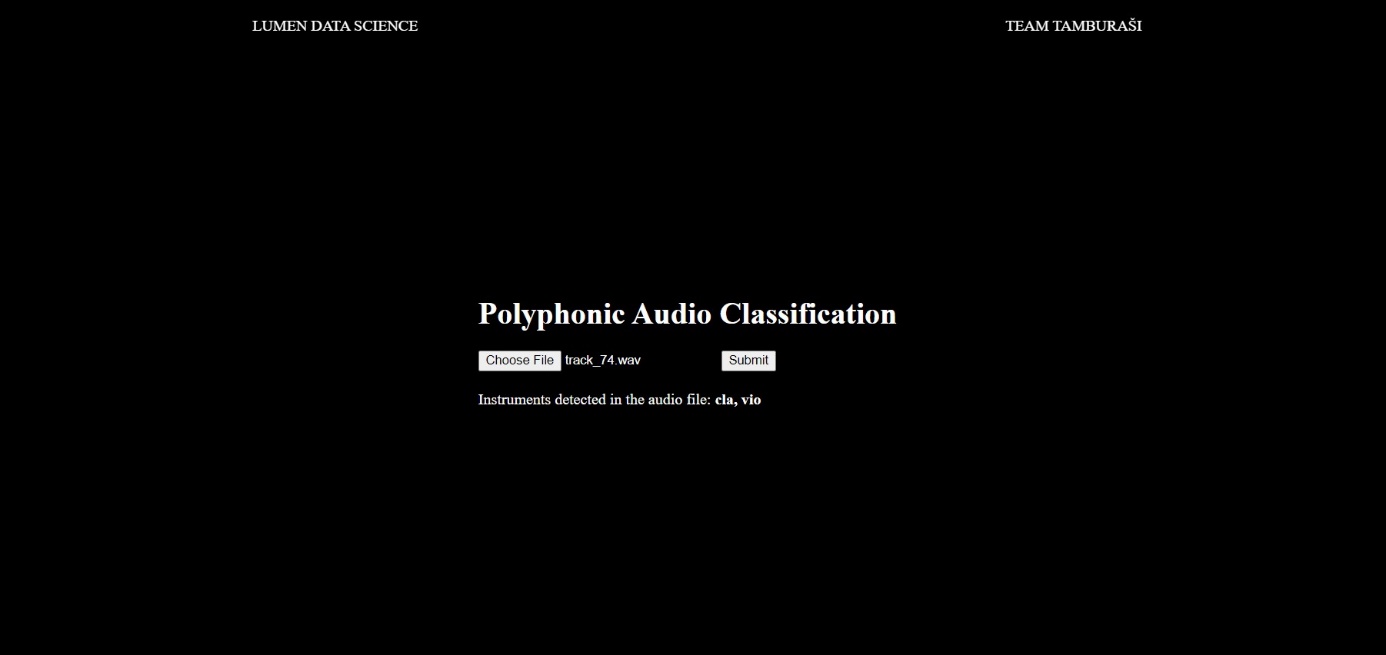
-Python 3.10.5

-Git (git bash)

7. API

Api je razvijen po prethodno definiranom pseudokodu. U git repozitoriju nalazi se u mapi API, gdje se nalazi nekoliko dijelova:

-mapa model s podmapama 1,2… gdje se nalaze različiti modeli koji se mogu mountat i spojiti s docker slikom tensorflow servinga, što označava sljedeće: ukoliko želim spojiti model 1 (ili bilo koji) te pomoću njega odraditi predviđanja, upravo preko ove mape, u kojoj se nalaze mape assets, variables te datoteka saved\_model.pb, što je u biti model, spojim se na docker kontenjer tfserving te na taj način , prilikom pokretanja Flask aplikacija, dohvaćam taj model te pomoću njega predviđam prisutnost instrumenata u audio zapisu. Aplikacija je razvijena u Flasku te se pokreće u direktoriju API. Prilikom pokretanja, podiže se lokalni server na kojemu je moguće obaviti prijenos audio zapisa, koji se klikom na gumb submit prenosi i obrađuje, šalje modelu te model predviđa instrumente. Set tih predikcija vraća se te ispisuje na stranicu, koja se nalazi u API mapi u podmapi templates pod nazivom indeks.html. Sve slike ili ostali sadržaj koji bih htio imati na stranici mora se nalaziti u mapi static. Cijela je aplikacija „zamotana“ u Dockerfile radi portabilnosti te se taj Dockerfile nalazi u mapi API, zajedno i s requirements.txt u kojemu su specificirane nužne verzije biblioteka za pokretanje koda. Izgled stranice vidljiv je prema slici:



8. OPIS KORIŠTENJA ZA KORISNIKA

Ovaj korisnički priručnik pruža detaljne upute za instalaciju, pokretanje koda i korištenje aplikacije za klasifikaciju polifonije na zvučnim zapisima. Aplikacija je testirana na laptopu Lenovo IdeaPad Gaming 3 15ARH05 82EY00LUMH s NVIDIA GeForce GTX 1650 Ti video karticom, Intel Core i7 procesorom i 16 GB RAM-a.

Ako već nemate Docker instaliran, posjetite Docker službenu web stranicu i slijedite upute za preuzimanje i instalaciju.

Preuzmite Git repozitorij aplikacije posjetom https://github.com/MarkoHaralovic/AudioClassification i klikom na "Code" te "Download ZIP" ili kroz terminal pomoću git naredbe:

git clone <https://github.com/MarkoHaralovic/AudioClassification>

Otvorite terminal u direktoriju AudioClassification i instalirajte potrebne Python pakete naredbom:

pip install -r requirements.txt

Pokrenite TensorFlow Serving Docker sliku naredbom:

docker run -p 8500:8500 -p 8501:8501 --name=tfserving --mount type=bind,source=C:\\Polyphonic\_audio\_classification\\API\\model,target=/models/2 -e MODEL\_NAME=2 -t tensorflow/serving

(ukoliko se prilikom pokretanja javi error: docker: Error response from daemon: Conflict. The container name "/tfserving" is already in use by container, što je mouće prilikom 2. pokretanja, potrebnan je sljedeći niz naredaba:

docker stop tfserving

docker rm tfserving

docker run -p 8500:8500 -p 8501:8501 --name=tfserving --mount type=bind,source=C:\\Polyphonic\_audio\_classification\\API\\model,target=/models/2 -e MODEL\_NAME=2 -t tensorflow/serving

ili alternativno,moguće je pokrenuti novi kontenjer:

docker run -p 8500:8500 -p 8501:8501 --name=tfserving2 --mount type=bind,source=C:\\Polyphonic\_audio\_classification\\API\\model,target=/models/2 -e MODEL\_NAME=2 -t tensorflow/serving)

U direktoriju AudioClassification\API , pokrenite app.py naredbom:

python app.py

Aplikacija će se pokrenuti na lokalnom serveru. Otvorite web preglednik i posjetite http://localhost:8080 da biste pristupili web sučelju aplikacije.

Kada otvorite web sučelje aplikacije, naći ćete opciju za postavljanje audio datoteke.

Kliknite na "Odaberi datoteku" ili povucite audio datoteku koju želite analizirati na označeno područje.

Nakon što odaberete audio datoteku, aplikacija će analizirati datoteku i vratiti predikcije instrumenata.

Rezultati predikcije prikazuju se u obliku grafičkog prikaza koji prikazuje vjerojatnost prisutnosti svakog instrumenta u zvučnom zapisu.