TEHNIČKA DOKUMENTACIJA
Audio Classification
LUMEN Data Science
2023

Tim **Tamburaši:**

Tin Josip Čurik

Marko Haralović

Sadržaj

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 metrike su redom : mel spektrogrami, mel-frekvencijski cepstralni koeficijenti, kromagrami 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 više instrumentalne audio zapise. Za učitavanje i obradu audio zapisa koristi se librosa library , izrada CNN modela je u tensorflow libraryu te je naposlijetku, po razvitku modela, isti spojen na docker kontenjer tensorflow servinga, odnosno nekoliko modela je spojeno na isti te je razvijena web stranica na kojoj je omogućen prijenos audio datoteke, te je omogućeno slanje audio zapisa , čime se poziva model koji obrađuje audio zapis te generira predikcije instrumenta, 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 augmentirani 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 na IRMAS validacijskom datasetu.

KLJUČNE RIJEČI: audio procesuiranje, polifonija, konvolucijske neuronske mreže, povećanje podataka, IRMAS dataset, librosa, tensorflow, tensorflow serving, docker kontenjeri , mel spektrogrami, značajke zvuka.

2. UVOD

2.1. LIBROSA BIBLIOTEKA

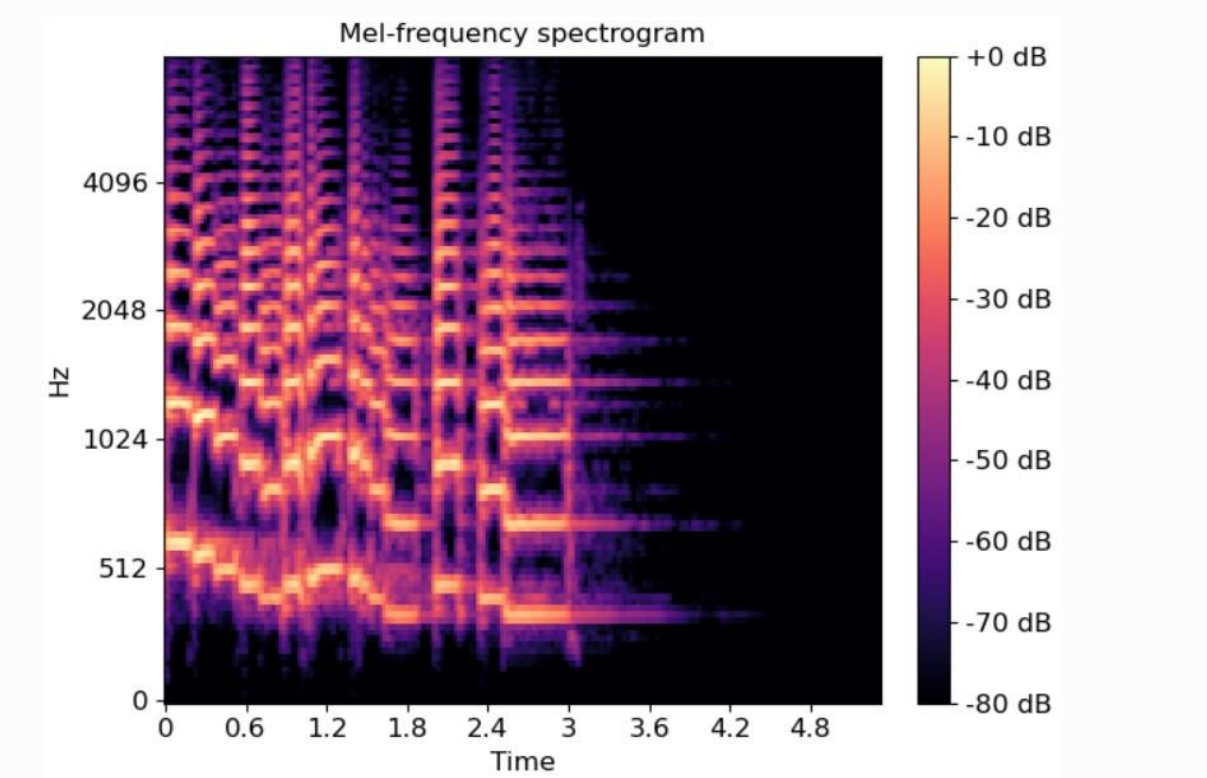
U našem smo projektu koristili Python library Librosa, koja je Python biblioteka za analizu zvuka i glazbe, kao takva se često koristi u raznim zadacima vezanim uz glazbu i neku generalnu obradu zvuka. Librosa nudi razne funkcije za učitavanje, pretprocesiranje i ekstrakciju značajki iz audio signala, 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 funkcije, odnosno značajke koje smo izvlačili i koristili, govorimo i o tome kako nam Librosa nudi funkcije 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.

Dokumentacija biblioteke na ovome je linku: https://librosa.org/doc/latest/index.html.

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.

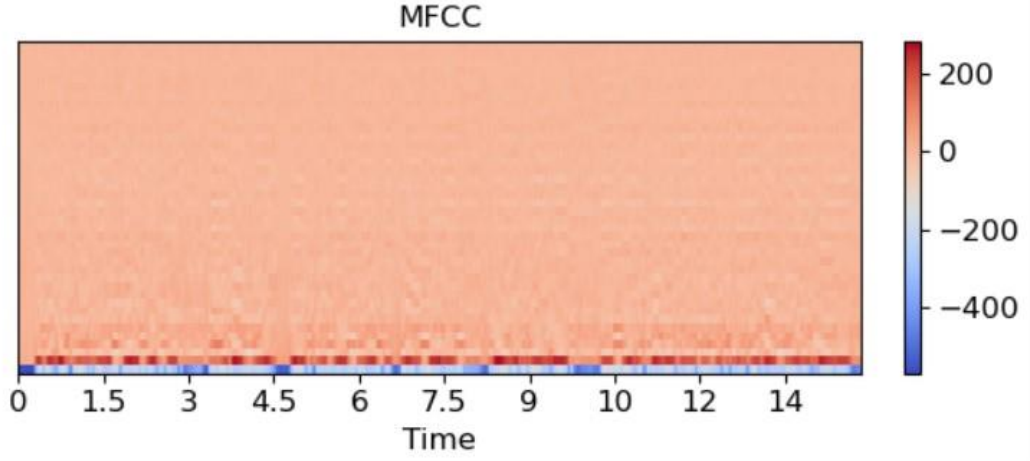


2.2.2. KROMAGRAM

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.

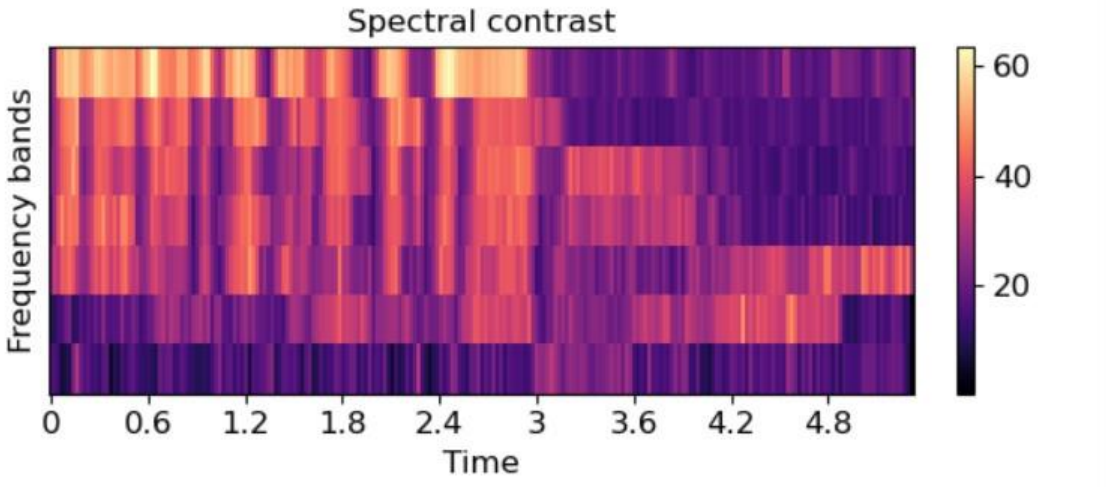
2.2.3. MFCCs (Mel-frekvencijski cepstralni koeficijenti):

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. Podroban je opis dan u samome radu priloženom ovome projektu. Za potrebe treninga CNN-a koristio se laptop s grafičkom procesnom jedinicom (GPU-om).

2.4. GITHUB REPOZITORIJ

Kreiran je javni git repozitorij https://github.com/MarkoHaralovic/AudioClassification 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.

Svaki push na glavnu granu obavljen je preko git basha, a za velike datoteke korišten je Git LFS.

Git LFS dostupan je na linu_: https://git-lfs.github.com

Provjera instalacije slijedi naredbom: **git lfs version**.

Svaki put kada se želi pratiti datoteka veličine koja premašuje klasičan git push, treba se upisati konfiguracijska naredba: **git lft install** te potom **git lfs init.**

Nepraćena datoteka može se pratiti naredbom**: git lfs track „datoteka|folder“.**

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 pretprocesiraju 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 zapis. 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. Set podataka nalazi se na sljedećoj poveznici: https://github.com/cosmir/openmic-2018.

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 model nije precizniji.

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 našem slučaju dodaje se na robusnosti koda i predikcijama, kako se sada model mogao učiti na zapisima koji sadrže i do 5 ili 6 instrumenata u sebi, što je daleko više nego što je ikoji zapis IRMAS-a sadržavao. Poznate su tehnike nasumičnog miksanja, pitch.sync miksanja,tempo-sync miksanja te miksanja po sličnosti žanra. U našem slučaju, govorili smo o kombinaciji nasumičnog miksanja, u procesu gdje jednostavno izvučene podatke konkateniramo. 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.

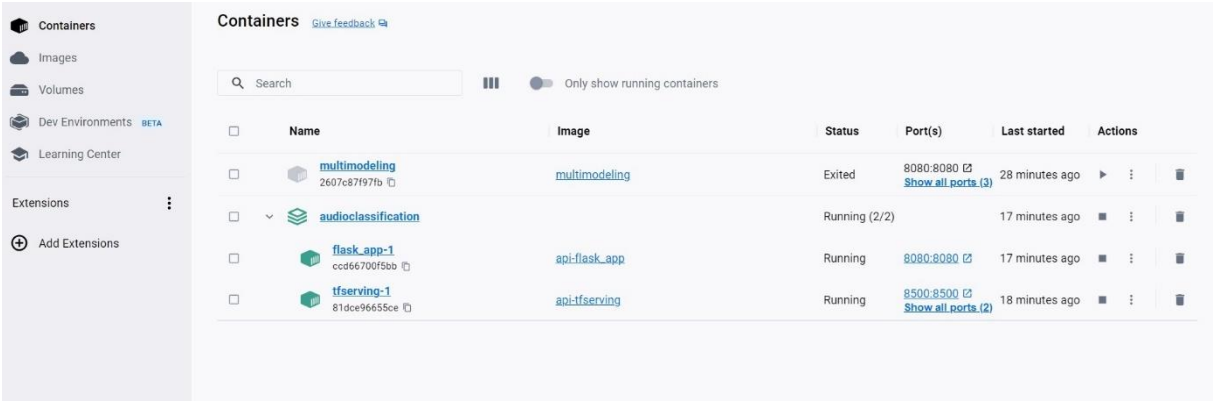
TensorFlow Serving može raditi s različitim vrstama modela, uključujući duboke neuronske mreže te omogućuje jednostavno dodavanje novih modela, ažuriranje i brisanje modela, uz minimalan prekid rada sustava. TensorFlow Serving također pruža mogućnost upravljanja verzijama modela i različitim strategijama posluživanja modela, što je bio upravo razlog zašto je pokrenut Deocker kontenjer tfserving preko kojega su spojeni različiti ponajbolji CNN modeli koje smo razvijali te preko kojega su vršena i spajanja na web API. Dokumentacija vezana za isti nalazi se na poveznici: https://www.tensorflow.org/tfx/guide/serving

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. Slika docker kontenjera korištenih u našem kodu:



3.OPIS RJEŠENJA

3.1 ARHITEKTURA MODELA

Koristeći biblioteku Librosa, učitaj audio zapis. Provuci audio zapis kroz skriptu za predprocesiranje 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. Slika modela dana je u nastavku:

	Initial 1-Conv Model [14]	**Proposed 2-Conv Model**
$\times 4$	Conv2D $(3 \times 3, d_i)$	$2\times$Conv2D $(3 \times 3, d_i)$
	Batch Normalization	
	ELU	LeakyReLU $(a = 0.3)$
	Max Pooling (p_i)	
	Dropout (0.2)	
	Dense (1024)	
	ELU	LeakyReLU $(a = 0.3)$
	Batch Normalization	
	Dropout (0.5)	
	Dense (11)	
	Sigmoid Activation	

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. Izgled modela po slojevima pa broj parametara vidljiv je na slici:

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 1, 159, 64)	25408
conv2d_1 (Conv2D)	(None, 1, 159, 64)	36928
batch_normalization (BatchNormalization)	(None, 1, 159, 64)	256
leaky_re_lu (LeakyReLU)	(None, 1, 159, 64)	0
max_pooling2d (MaxPooling2D)	(None, 1, 80, 64)	0
dropout (Dropout)	(None, 1, 80, 64)	0
conv2d_2 (Conv2D)	(None, 1, 80, 128)	73856
conv2d_3 (Conv2D)	(None, 1, 80, 128)	147584
batch_normalization_1 (BatchNormalization)	(None, 1, 80, 128)	512
leaky_re_lu_1 (LeakyReLU)	(None, 1, 80, 128)	0
max_pooling2d_1 (MaxPooling2D)	(None, 1, 40, 128)	0
dropout_1 (Dropout)	(None, 1, 40, 128)	0
conv2d_4 (Conv2D)	(None, 1, 40, 256)	295168
conv2d_5 (Conv2D)	(None, 1, 40, 256)	590080
batch_normalization_2 (BatchNormalization)	(None, 1, 40, 256)	1024
leaky_re_lu_2 (LeakyReLU)	(None, 1, 40, 256)	0
max_pooling2d_2 (MaxPooling2D)	(None, 1, 14, 256)	0
dropout_2 (Dropout)	(None, 1, 14, 256)	0
conv2d_6 (Conv2D)	(None, 1, 14, 640)	1475200
conv2d_7 (Conv2D)	(None, 1, 14, 640)	3687040
batch_normalization_3 (BatchNormalization)	(None, 1, 14, 640)	2560
leaky_re_lu_3 (LeakyReLU)	(None, 1, 14, 640)	0
max_pooling2d_3 (MaxPooling2D)	(None, 1, 5, 640)	0
dropout_3 (Dropout)	(None, 1, 5, 640)	0
flatten (Flatten)	(None, 3200)	0
dense (Dense)	(None, 1024)	3277824
leaky_re_lu_4 (LeakyReLU)	(None, 1024)	0
batch_normalization_4 (BatchNormalization)	(None, 1024)	4096
dropout_4 (Dropout)	(None, 1024)	0
dense_1 (Dense)	(None, 11)	11275

Total params: 9,628,811
Trainable params: 9,624,587
Non-trainable params: 4,224

3.2.PSEUDOKOD

3.2.1. PSEUDOKOD ZA PREDPROCESURIANJE

- Uvezi potrebne biblioteke (os, numpy, librosa, tqdm)
- Definiraj konstante (SAMPLE_RATE, BLOCK_SIZE, HOP_SIZE, MEL_BANDS, DURATION)
- Definiraj funkciju parse_label s ulazom label_path
    - Otvori datoteku label_path za čitanje
    - Pročitaj sadržaj datoteke i ukloni praznine s početka i kraja
    - Razdvoji sadržaj po tabulatoru i vrati rezultirajući popis instrumenata
- Definiraj funkciju preprocess_audio s ulazom audio_path
    - Učitaj audio datoteku na audio_path s određenom brzinom uzorkovanja i u mono formatu
    - Pretvori stereo audio u mono
    - Normaliziraj amplitudu zvuka
    - Podijeli audio u jednosekundne intervale
    - Za svaki audio segment izračunaj njegov kratkotrajni Fourierov transform i mel spektrogram
    - Pretvori mel spektrogram u decibelske jedinice i dodaj na popis
    - Vrati popis log mel spektrograma
- Definiraj funkciju process_audio_files s ulazom input_dir
    - Inicijaliziraj prazne liste X i y
    - Za svaki korijen, direktorij i datoteku u input_dir
- 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
    - Dodaj rezultirajuće log mel spektrograme u X i oznaku u y
- iv. Ako datoteka s oznakama nije pronađena, ispiši poruku o pogrešci
    - Pretvori liste X i y u numpy nizove i vrati ih
- Glavni program
    - Pozovi funkciju process_audio_files s putanjom ulaznog direktorija
    - Spremi rezultirajuće X i y nizove na disk

3.2.2.PSEUDOKOD ZA AUGMENTACIJU

- Uvezi potrebne biblioteke (os, numpy, librosa, tqdm, random, re)
- Definiraj konstante (SAMPLE_RATE, BLOCK_SIZE, HOP_SIZE, MEL_BANDS, DURATION, class_names)
- Definiraj funkciju spectral_contrast s ulazom audio_path
- Definiraj funkciju chromagram s ulazom audio_path
- Definiraj funkciju parse_instruments s ulazom file_name
- Definiraj funkciju check_unique_instruments s ulazima file1 i file2
- Definiraj funkciju parse_label s ulazom label_path

- Definiraj funkciju mel_spectrogram s ulazom audio_path
- Definiraj funkciju one_hot_encode s ulazima labels i class_names
- Definiraj funkciju process_audio_files s ulazima input_dir i num_iterations
    - Inicijaliziraj prazne liste X i y
    - Uz pomoć for petlje i funkcije tqdm obradi num_iterations iteracija
- Izaberi nasumične poddirektorije
- ii. Pronađi i obradi audio datoteke iz tih poddirektorija
    - Izračunaj mel spektrogram, kromagram i spektralni kontrast
    - Spremi rezultate u X
    - Dohvati i kodiraj instrumente za svaku datoteku
    - Spremi rezultate u y
    - Pretvori liste X i y u numpy nizove i vrati ih
- Glavni program
    - Pozovi funkciju process_audio_files s putanjom ulaznog direktorija
    - Spremi rezultirajuće X, y i z nizove na disk

3.2.3. PSEUDOKOD ZA OCJENU KVALITETE MODELA

- Uvezi potrebne biblioteke (keras, numpy, matplotlib, librosa, tensorflow_addons)
- Učitaj X i y iz numpy datoteka
- Učitaj model iz datoteke
- Inicijaliziraj varijable tp (true positives), fp (false positives) i fn (false negatives)
- Za svaki i-ti element X s korakom 10:
    - Preoblikuj X[i] dodavanjem dimenzije i konvertiranjem u float32
    - Inicijaliziraj niz non_binary_label s nulama
    - Za svaki j-ti element X[i]:
- Izračunaj predikciju modela za X[i][j] i zbroji s non_binary_label
    - Normaliziraj non_binary_label
    - Inicijaliziraj pred_binary_label s nulama
    - Za svaki j-ti element non_binary_label:
- 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]
    - 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

- Uvezi potrebne biblioteke (numpy, librosa, os, glob, keras, scipy, tqdm, tensorflow_addons, json)
- Definiraj konstante (SAMPLE_RATE, BLOCK_SIZE, HOP_SIZE, MEL_BANDS, DURATION)
- Definiraj funkciju preprocess_audio(audio_path):
    - Učitaj audio podatke iz audio_path
    - Konvertiraj audio u monokanalni zvuk

    - Normaliziraj audio
    - Segmentiraj audio u intervale od jedne sekunde
    - Izračunaj log mel-spektrogram za svaki segment
    - Vrati listu log mel-spektrograma

- Definiraj funkciju validation_accuracy(file_path_txt, predictions, threshold=0.5):
    - Učitaj ground truth iz datoteke file_path_txt
    - Usporedi predictions s ground truth i izračunaj točnost
    - Vrati točnost
- Definiraj funkciju process_audio_files(input_dir):
    - Inicijaliziraj X kao praznu listu
    - Za svaku datoteku u input_dir:
- Ako je datoteka audio format, izračunaj log mel-spektrogram
- ii. Dodaj log mel-spektrogram u X
    - Vrati X kao numpy niz
- Definiraj funkciju predict_windows(model, log_mel_spectrogram):
    - Vrati predikciju modela za log_mel_spectrogram
- Definiraj funkciju aggregate_predictions(predictions):
    - Sumiraj predictions
    - Vrati normalizirane sume
- Definiraj funkciju test_model(model, test_data_path, threshold=0.5):
    - Inicijaliziraj rezultate i točnosti
    - Definiraj listu instrumenata
    - Za svaku audio datoteku u test_data_path:
- 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
    - Izračunaj srednju točnost
    - Ispiši srednju točnost
    - Vrati rezultate
- Učitaj model iz datoteke
- Postavi test_data_path
- Pokreni testiranje modela na testnim podacima
- Spremi rezultate u JSON datoteku

3.2.5. PSEDUKOD ZA APLIKACIJU U FLASK-U

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

- Definiraj konstante (SAMPLE_RATE, BLOCK_SIZE, HOP_SIZE, MEL_BANDS, DURATION)
- Definiraj funkciju preprocess_audio(audio_path):
    - Učitaj audio podatke iz audio_path
    - Konvertiraj audio u monokanalni zvuk
    - Normaliziraj audio
    - Segmentiraj audio u intervale od jedne sekunde
    - Izračunaj log mel-spektrogram za svaki segment
    - Vrati listu log mel-spektrograma
- Definiraj funkciju predict_windows(model, log_mel_spectrogram):
    - Dodaj dimenziju kanala ulaznim podacima
    - Pošalji zahtjev na TensorFlow serving API
    - Ako je uspješno, vrati predikciju, inače vrati None
- Definiraj funkciju aggregate_predictions(predictions):
    - Sumiraj predictions
    - Vrati normalizirane sume
- Učitaj model
- Inicijaliziraj Flask aplikaciju
- Definiraj rute za Flask aplikaciju:
    - index() - prikaži početni HTML template
    - predict() - prihvati zahtjev za predikciju
- 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
- 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:

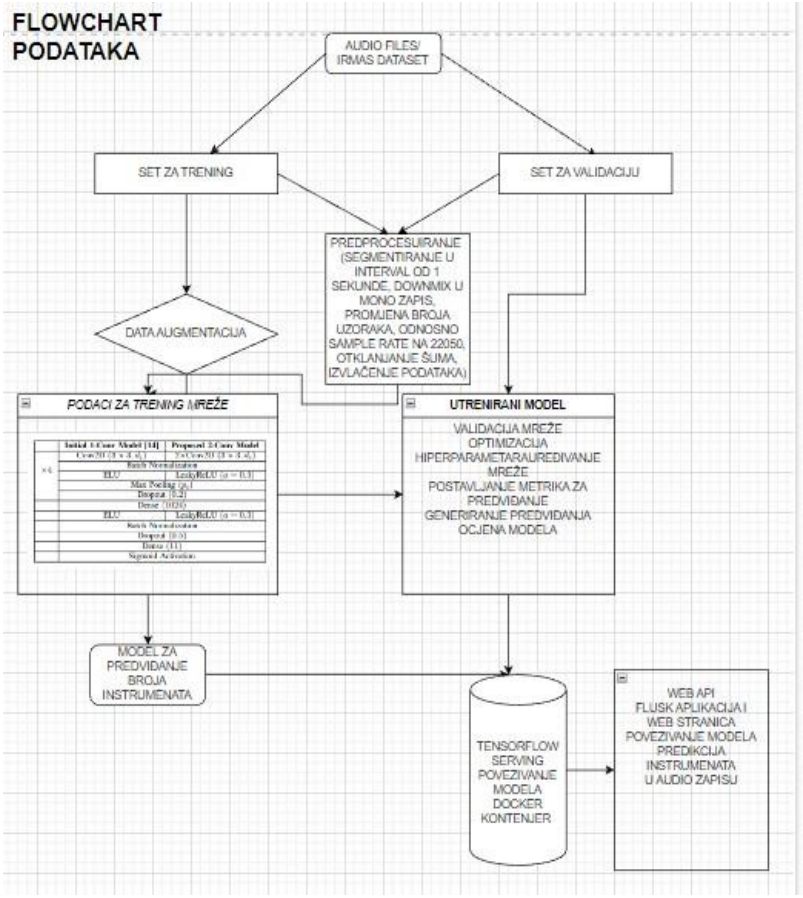
- Trenirati model za klasifikaciju polifonije na zvučnim zapisima
- Prilagoditi arhitekturu modela i hiperparametre prema potrebama
- Učitati prethodno spremljeni model i nastaviti s treniranjem
- Evaluirati performanse modela na validacijskom skupu podataka
- Koristiti model za klasifikaciju polifonije 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

Na sljedećoj slici prikazan je protok informacija, u našem slučaju audio zapisa iz seta podataka IRMAS-a.



4.ORGANIZACIJA IZVORNOG KODA

Cijeli je kod dostupan na javnom repozitoriju na gitu:
https://github.com/MarkoHaralovic/AudioClassification

Struktura repozitorija jest:

i. Mapa API – sadrži folder model s 3 različita modela. Ovdje su uključene samo potpuno trenirane verzije modela konvertirane u pb file. Prvi je model 78kratimenos.h5 , model treniran na mel spektrogramima, bez audio segmentacije. Drugi je model augpoly78.py koji je treniran na segmentiranim podacima te koji je robusniji model, pogodan za predikciju audio zapisa s većim brojem instrumenata. Ta su dva modela nositelji projekta te se na njima mogu testirati predikcije. Treći je model sačuvan kao prvotni model koji služi za predikciju predominantnog instrumenta u zapisu, ali nije pogodan za pogađanje više klasa. Taj modle, mel_spec_irmas_singleton.h5 ne služi za testiranje.
ii. Mapa IMAGES -sadrži najbitnije slike za projekt
iii. Mapa audio_segment sadrži kodove za audio segmentaciju audio zapisa.
iv. Mapa audio_preprocessing sadrži kodove za pred procesuiranje audio zapisa, sadrži nekoliko kodova, ovisno o namjeri i tipu željenih podataka za izvući iz slike.
v. Mapa h5_models sadrži h5 format modela (tri prethodno opisana)
vi. Mapa models koja sadrži kodove za građenje i treniranje konvolucijskih neuronskih mreža.
vii. Mapa npy_data koja sadrži informacije o podacima izvučenima iz slike u X npy varijabli te pripadajuće labele u y varijabli.
viii. Mapa testing sadrži funckije za testiranje postojećih modela na testnom primjeru. Pomoću njih generiraju se predikcije te se spremaju u JSON file, uz pridržavanje hot encodinga.
ix. Mapa validation_json_files sadrži json file u kojemu su spremljene predikcije modela.

5.NASTAVAK RADA

Trenutni modeli postižu state-of-the-art rezultate, stoga je prilično teško komentirati koja bi poboljšanja predložili u tom smislu. Jasno, postoje razne metode koje bismo mogli implementirati i isprobati na stvarnom slučaju. Počevši od audio procesuiranja, postoje različite metode za segmentiranje audio zapisa, čišćenje šuma, poboljšavanje podataka, pravilno izvlačenje podataka, pravilno proširenje seta augmentiranjem (koje jasno može biti procijenjeno po drugačijim metrikama i pratiti drugačije kombiniranje od našeg), proširenje seta drugim mogućim datasetima, izrada vlastitog dataseta ako je vlastoručno miješanje instrumenata reprezentativno u odnosu na stvarnu polifoniju (ne smije se isključiti boja zvuka i umijeće skladanja). Jedna od mogućnosti je i transfer learning, odnosno preuzimanje gotovih modela koji bi se prilagodili našim podacima. Tu se jasno govori o IMAGENET-u, densenet-u, inception-u 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, bili su limitirani našim tehničkim specifikacijama laptopa. Jedan dio je bio pokrenut na CPU-u, ali trening mreže i procesuiranje je bilo odrađeno na GPU-u. No, memorijski problem je bio prisutan, što bi se u daljnjem radu moglo korigirati. Tako bi mogao uslijediti još bolji rezultat ako bi se riješio takoreći "tehnički nedostatak" koji je bio prisutan u ovom 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)
- Flask==2.3.2
- numpy==1.23.5
- librosa==0.9.2
- requests==2.28.1
- tensorflow==2.8.0
- Werkzeug==2.3.3
- scipy==1.10.0
- tqdm==4.64.1
- protobuf==3.20.0

7. API

Api je razvijen prema prethodno definiranom pseudokodu, a nalazi se u git repozitoriju u mapi API. Ova mapa sadrži nekoliko dijelova:

- Mapa model s podmapama 1, 2 te 3 gdje se nalaze različiti modeli koji se mogu mountati i spojiti s Docker slikom TensorFlow Servinga. To znači da se model 1 (ili bilo koji drugi) može spojiti preko ove mape, u kojoj se nalaze mape assets, variables te datoteka saved_model.pb, koja je u biti sam model. Na ovaj način se prilikom pokretanja Flask aplikacije može dohvatiti taj model te koristiti za predviđanja prisutnosti instrumenata u audio zapisu.
- Aplikacija je razvijena u Flasku i pokreće se u direktoriju API. Prilikom pokretanja, podiže se lokalni server na kojemu je moguće prenijeti audio zapis klikom na gumb "submit". Audio zapis se obrađuje, šalje se modelu te model predviđa instrumente. Set predikcija vraća se i ispisuje na stranicu koja se nalazi u API mapi u podmapi templates pod nazivom indeks.html.
- Sve slike ili ostali sadržaj koji se želi imati na stranici moraju biti smješteni u mapi static.
- Cijela aplikacija "zamotana" je u Dockerfile radi portabilnosti, a taj Dockerfile nalazi se u mapi API, zajedno s requirements.txt datotekom u kojoj su specificirane nužne verzije biblioteka za pokretanje koda.

Pokretanje dockerfilea kreće s komandom:

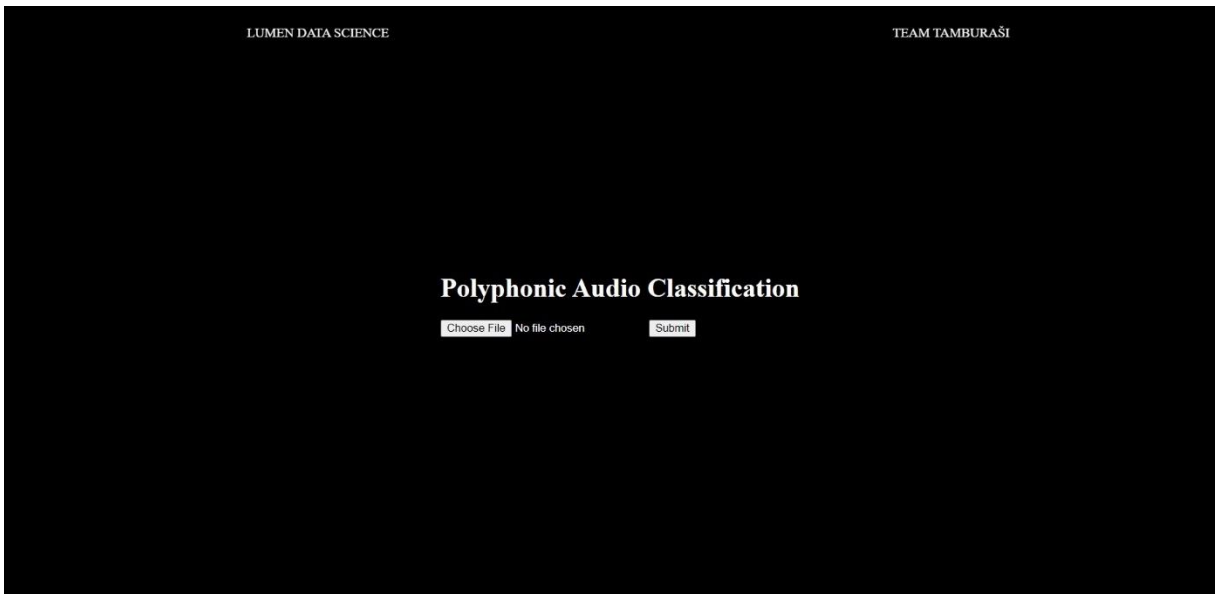
- docker build -t multimodeling .

gdje se pokreće izrada kontenjera po Dockerfile specificiranom u API folderu.

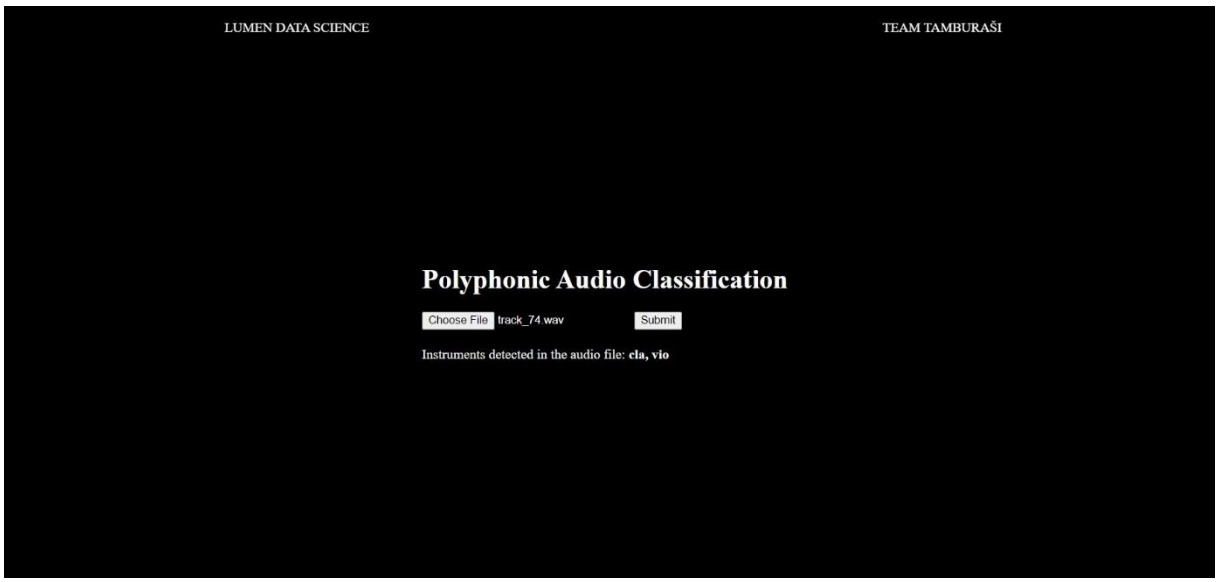
Kontenjer se pokreće naredbom:

- docker run -p 8500:8500 -p 8501:8501 -p 8080:8080 --name multimodeling -it multimodeling

Izgled početne stranice prikazan je na slici:



A nakon submita filea, poziva modela metodom tf servinga request, obrade audio zapisa , generiranja predikcija modela te povrata vrijednosti predikcija metodom push, iste se ispisuju na ekranu na sljedeći način:



8. OPIS KORIŠTENJA ZA KORISNIKA

Korisnički priručnik za instalaciju i uporabu aplikacije za klasifikaciju polifonije na zvučnim zapisima dostavlja detaljne upute. 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.

Sljedeći su koraci potrebni za instalaciju i pokretanje aplikacije:

1. **Instalirajte Docker ako ga već nemate na vašem računalu. Posjetite službenu web stranicu za preuzimanje i instalaciju.**
2. **Preuzmite Git repozitorij aplikacije posjetom https://github.com/MarkoHaralovic/AudioClassification i kliknite na "Code" te "Download ZIP", ili otvorite terminal i upotrijebite git naredbu:**
   - git clone https://github.com/MarkoHaralovic/AudioClassification
3. **U terminalu otvorite direktorij AudioClassification i instalirajte potrebne Python pakete naredbom:**
   pip install -r requirements.txt
4. **Pokrenite TensorFlow Serving Docker sliku naredbom:**
   - docker build -t multimodeling .
5. **Ukoliko se prilikom pokretanja javi error: docker: Error response from daemon: Conflict. The container name "/tfserving" is already in use by container, što je moguće prilikom 2. pokretanja, potreban je sljedeći niz naredaba:**
   - docker stop tfserving
   - docker rm tfserving
   - docker build -t multimodeling .
6. **ili alternativno, moguće je pokrenuti novi kontenjer:**

   - docker run -p 8500:8500 -p 8501:8501 -p 8080:8080 --name multimodeling -it multimodeling

   što može biti zamjenjeno naredbom koja će poslužiti sva tri modela:
   - docker run -p 8500:8500 -p 8501:8501 --name=tfserving_multi --mount type=bind,source=C:\AudioClassification\API\model\1,target=/models/model1 --mount type=bind,source=C:\AudioClassification\API\model\2,target=/models/model2 –mount type=bind,source=C:\AudioClassification\API\model\3,target=/models/model3 -e MODEL_NAME=model1,model2,model3 -t tensorflow/serving

Ukoliko se javlja greška, preko sljedeće konfiguracijske datoteke:

API > model_config_file.conf

```
model_config_list {
  config {
    name: "model1"
    base_path: "/models/model1"
    model_platform: "tensorflow"
    model_version_policy {
      specific {
        versions: 1
      }
    }
  }
  config {
    name: "model2"
    base_path: "/models/model2"
    model_platform: "tensorflow"
    model_version_policy {
      specific {
        versions: 1
      }
    }
  }
  config {
    name: "model3"
    base_path: "/models/model3"
    model_platform: "tensorflow"
    model_version_policy {
      specific {
        versions: 1
      }
    }
  }
}
```

problem se razrješava pozivom naredbe:

- docker run -p 8500:8500 -p 8501:8501 --name=tfserving_multi --mount type=bind,source=C:/AudioClassification/API/model/1,target=/models/model1 --mount type=bind,source=C:/AudioClassification/API/model/2,target=/models/model2 --mount type=bind,source=C:/AudioClassification/API/model/3,target=/models/model3 --mount type=bind,source=C:/AudioClassification/API/config,target=/models/config -e MODEL_NAME=model1,model2,model3 -t tensorflow/serving --model_config_file=/models/config/model_config_file.conf

7. **U direktoriju AudioClassification\API, pokrenite app.py naredbom:**
    - python app.py

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

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

10. Po postavljanju audio zapisa, obavlja se predprocesiranje, dohvaća se model, šalju se obrađeni podaci, generiraju predikcije te se iste ispisuju na zaslon.