Prepoznavanje instumenata u zvučnim datotekama

Travanj 2023.

Sadržaj

1 Uvod

1.1 Sažetak

U ovom radu rješavat ćemo problem prepoznavanja instrumenata u zvučnom signalu. Glavni cilj ovog rada je istražiti problem te vidjeti koji pristup rješavanju daje najbolje rezultate i zašto. Sporedni cilj je naučiti nešto o obradi podataka, obradi zvuka i neuronskim mrežama.

Prvo smo pregledali moguće transformacije zvuka - Fourierovu transformaciju, spektograme, spektralno središte i slično. Zatim smo isprobali jednostavnije metode klasfikacije: *k-najbližih susjeda*, *spectral centroid* i *linearnu regresiju*.

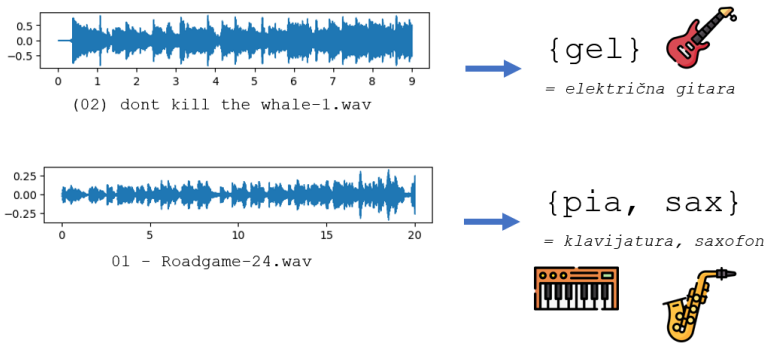
Kasnije pokazujemo kako su duboke neuronske mreže učinkovite u ovakvim zadacima. Isprobali smo različite početne transformacije zvuka, augmentacije i arhitekture samog modela. Model smo dovršili korištenjem klizećih prozora.

Na kraju predstavljamo aplikaciju koja koristi prethodno izrađeni model. Napravljena po uzoru na *Shazam* (poznatu aplikaciju koja prepoznaje naziv pjesme po jako kratkom uzorku), tako da klikom na gumb počinje snimati zvuk i prepoznaje instrumente koji su snimljeni.

Uz rad smo pripremili i Jupyter bilježnicu koja se nalazi u repozitoriju te sadrži cijeli postupak izrade modela te njegovog treniranja.

1.2 Zadatak

Ovogodišnji zadatak na Lumenu je prepoznavanje instrumenata iz zvučnih datoteka. Svaku .wav datoteku potrebno je učitati, obraditi te vratiti popis instrumenata koji se u njoj pojavljuje. Mogući instrumenti unaprijed su zadani: cel, cla, flu, gac, gel, org, pia, sax, tru, vio, voi.



Slika 1: Grafički prikaz zadatka

Problem možemo podijeliti na dva dijela:

- **Monofonija** je detektiranje točno jednog instrumenta u datoteci,
- **Polifonija** je detektiranje svih instrumenta koji se pojavljuju u datoteci.

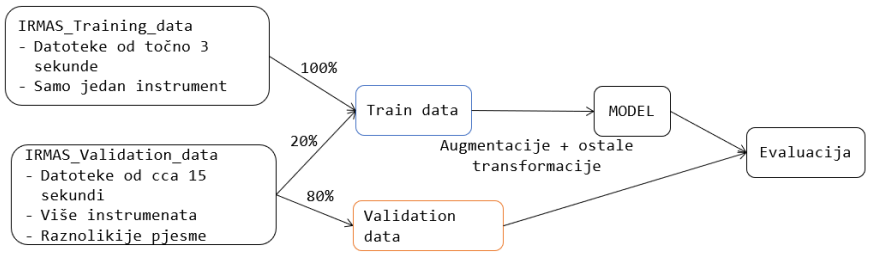
Problem monofonije je mnogo jednostavniji pa ćemo s njime i krenuti. Isprobat ćemo različite pristupe, a zaključke primijeniti na problemu polifonije.

1.3 Dataset

Dataset koji ćemo koristiti je IRMAS dataset[1]. Podijeljen je u dva dijela: *IRMAS_ training* i *IRMAS_ validation*. U prvome su datoteke dugačke točno tri sekunde i u njima se pojavljuje točno jedan dominantan instrument. U drugome su datoteke dugačke između 5 i 20 sekundi te se u njima može pojavljivati od jedan do pet instrumenata. Tu su datoteke raznovrsnije i sličnije stvarnoj primjeni.

Prvi skup podataka dobar je za problem monofonije pa ćemo ga u tu svrhu na početku i koristiti. Nasumično ćemo ga podijeliti na dva podskupa - skup za treniranje i skup za validaciju.

Za problem polifonije ćemo također podijeliti skup podataka na dva dijela. Cijeli *IRMAS_ training* i dio *IRMAS_ validation* (oko 20%) će nam biti skup za treniranje, a ostatak *IRMAS_ validation* će nam biti skup za testiranje.



Slika 2: Dataset

Ovo radimo zbog toga što je *IRMAS_ validation* raznovrsniji te se isplati trenirati i na njemu. Bitno za napomenuti je da 20% podataka nije *slučajno* odabrano, već je uzeto prvih 20% podataka. Kako se u datasetu od svake pjesme može pronaći više klipova (čak i više od 10), ovime dio njih neće završiti u train skupu, a dio u test skupu, već će svi klipovi iste pjesme završiti u istome skupu. Accuracy mjerimo isključivo na skupu za testiranje!

1.4 Slični radovi

Kako bismo bolje znali na koju preciznost trebamo ciljati, promotrili smo slične radove. Većina radova koje smo pronašli rješavaju samo problem monofonije:

- Prvi rad[3] dolazio je do 84.34% preciznosti već u prve tri epohe treniranja. Imao je otprilike jednak broj datoteka i instrumenata kao naš dataset. U radu je korišteno duboko učenje (*fastai* biblioteka), ali nisu bile primjenjene augmentacije ulaznih podataka.

- Drugi rad[2] također je baziran na dubokom učenju (*fastai*). Dolazio je čak do 87.5% preciznosti (*accuracy*) koristeći arhitekturu *densenet121*. S druge strane, koristeći *resnet18*, preciznost je bila 81.25%. Ovdje je broj instrumenata bio 8, ali broj datoteka znatno manji, oko 200.

- Treći rad[6] je iz 2019. te umjesto direktne primjene modela dubokog učenja, prvo računaju glavne značajke svake datoteke. One ovise o frekvencijama koje se pojavljuju u datoteci. Tako kreiraju vektor od 13 značajki te na te vektore primjenjuju različite metode strojnog učenja: *duboko učenje*, *random forests*, *Logistic regression*... Dobivaju accuracy od 79% što se pokazao među najboljim rezultatima za takav pristup. Za dataset su također koristili IRMAS dataset.

Iako su datasetovi i pristupi različiti, daju nam osjećaj o težini problema. Čini nam se da je za problem monofonije pristojan accuracy od 80% te da primjenom dubokog učenja s time ne bismo trebali imati problema.

No što je sa prepoznavanjem svih instrumenata koji se pojavljuju na datoteci? Definicija *preciznosti* se sada mijenja te koristimo metriku Hamming score. Stoga bismo trebali prilagoditi svoja očekivanja.

1. U *IRMAS_ Validation* skupu se u prosjeku javlja $avg = 1.78$ instrumenata po datoteci. Stoga model koji uvijek tvrdi da se ne pojavljuje nijedan instrument bi imao preciznost od 83.83%.

2. Pretpostavimo sada da model monofonije možemo jednostavno *preseliti* na polifoniju. Ako model nastavi pogađati samo jedan instrument te u $ac = 85\%$ slučajeva pogodi, a u 15% promaši, njegova preciznost bila bi:

$$ac \times (1 - (avg - 1)/11) + (1 - ac) \times (1 - (avg + 1)/11) = 90.19\%$$

3. Možda se model uz neke modifikacije može i bolje preseliti na problem polifonije. Recimo da ovaj put u $ac = 85\%$ slučajeva pogodi sve instrumente koji se pojavljuju, a u 15% ih sve promaši, tada bi imamo preciznost:

$$ac + (1 - ac) \times (1 - (avg + 1.7)/11) = 95.15\%$$

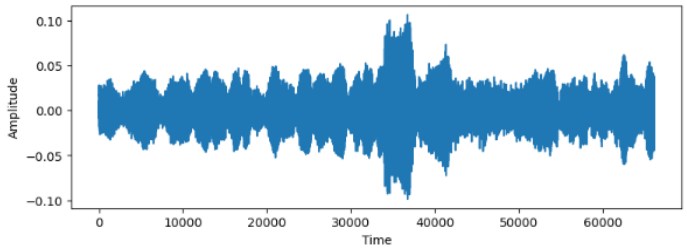
Naravno da je problem polifonije daleko teži te nije jednostavno samo preseliti model. Sami podatci su kompleksniji te bi mogli zbuniti ovakav model. Stoga bi razumno bilo ciljati na neku preciznost između prve i druge, minimalno 84%, a 90% bi bilo izvrsno.

2 Analiza i obrada podataka

Prije samog kreiranja modela potrebno je pomno promotriti skup podataka s kojim radimo te iz njih izvući korisne statistike koje bi nam kasnije mogle pomoći u kreiranju modela. Osim što neke statistike možemo koristiti direktno kao ulaz u model, druge nam mogu pomoći u otkrivanju novih pristupa prema samom stvaranju modela.

2.1 Što je to zvuk?

Osnovni tip podataka koji promatramo je zvuk, preciznije, .wav datoteka. Takva reprezentacija zvuka ima dvije dimenzije: vrijeme i amplitudu (Slika 3). No, takva reprezentacija zvuka nam baš ne daje previše vidljivih informacija. Nama je cilj pronaći neke frekvencijske karakteristike instrumenata pomoću kojih bismo mogli lakše razaznati koji je točno instrument u pitanju.

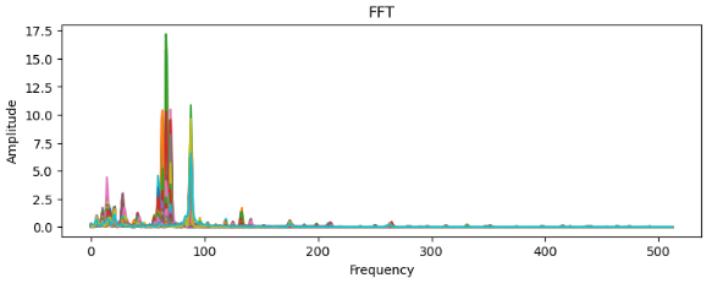


Slika 3: Audio zapis

Zvuk je zapravo titranje molekula zraka, a brzina tog titranja određuje frekvenciju zvučnog signala. Naše je uho oblikovano da izmjeri pritisak zraka (tlak) na ušnoj opni te tako razaznaje određene frekvencije. Uzmimo za primjer sviranje pete žice na gitari. To je 'A' nota i proizvodi 110 Hz. Našem uhu će ovakav zvučni signal pobuditi opnu 110 puta u sekundi i ono će to percipirati kao notu 'A'. Ali u stvarnosti, zbog prirode instrumenta, gitara neće proizvesti samo jedan signal od 110 Hz, već mnogo njih, sa različitim frekvencijama, a naše uho će detektirati njihov zbroj. Neke od tih frekvencija će biti izraženije od drugih, a za ovaj specifičan slučaj će najjače biti izražena frekvencija od 110 Hz. Raspored i jačinu izraženosti tih sekundarnih frekvencija (harmonika) naše uho doživljava kao 'boju' (engl. timbre) instrumenta. Ti harmonici će nam jako dobro doći za detekciju instrumenata u našem modelu. S obzirom da nas ne zanima cijela snimka, već samo boja instrumenta na snimci, bilo bi dobro kada bi postojao neki alat s kojim možemo jednostavno preći iz vremenske domene u frekvencijsku.

2.2 Brza Fourierova transformacija

Srećom, takav alat postoji i zove se Fourierova transformacija. Nećemo ulaziti u to kako funkcioniraju Fourierove transformacije jer to zahtjeva opsežno znanje matematike koja nije predmet ovog rada. No ukratko, to je način pretvaranja valnog oblika u frekvencijski spektar i obrnuto. Za računalne potrebe uglavnom se koristi algoritam brze Fourierove transformacije (FFT) koja uz neke optimizacije ubrzava taj proces. Nakon provedene transformacije možemo lako vidjeti osnovni signal te izražene harmonike (Slika 4).

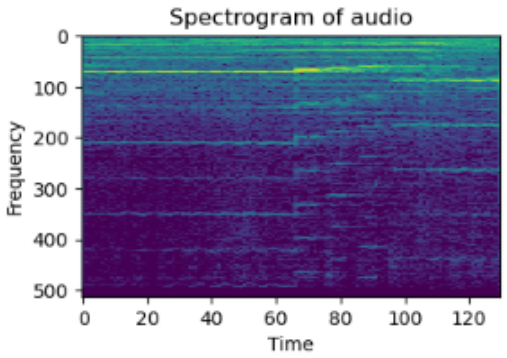


Slika 4: Fourierova transformacija slike

No uz ovakvu transformaciju ne prikazujemo informacije o vremenu unutar originalnog signala. S obzirom da ćemo se kasnije susretati sa glazbom koja ne pušta sve instrumente odjednom, već različiti instrumenti sviraju u različito vrijeme, a neki sviraju istovremeno, bilo bi odlično kada bi mogli prikazati frekvencijski spektar u ovisnosti o vremenu.

2.3 Spektogrami

Spektogram je upravo to, vizualna reprezentacija frekvencijskog spektra signala u ovisnosti o vremenu. Postoji puno programskih alata koji stvaraju spektogram iz originalnog zvuka (koji također koriste FFT za svoje kalkulacije). Ovakva reprezentacija zvuka bi trebala biti najbolji odabir za naš model, no isprobat ćemo i ostale pristupe i uvjeriti se da je tako.



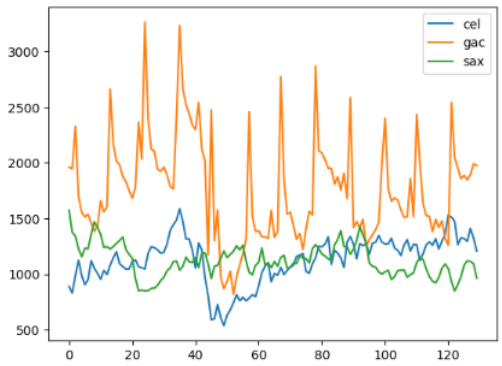
Slika 5: Spektogram

2.4 Spektralno središte i širina pojasa

Još jedan zanimljiv podatak koji se može izvaditi iz originalnog signala jest njegovo spektralno središte. Ono pokazuje gdje se nalazi centar mase spektra te snažno korelira s našim dojmom svjetline zvuka. Izračunava se kao težinska srednja vrijednost frekvencija pristunih u signalu koje se dobiju koristeći Fourierove transformacije.

$$\text{Centroid} = \frac{\sum_{n=0}^{N-1} f(n) \cdot X(n)}{\sum_{n=0}^{N-1} X(n)}, \tag{1}$$

gdje x(n) predstavlja amplitudu signala, a f(n) predstavlja srednju frekvenciju. Na slici 6 mogu se vidjeti spektralna središta tijekom vremena za tri različita instrumenta: violončelo, akustičnu gitaru i saksofon.



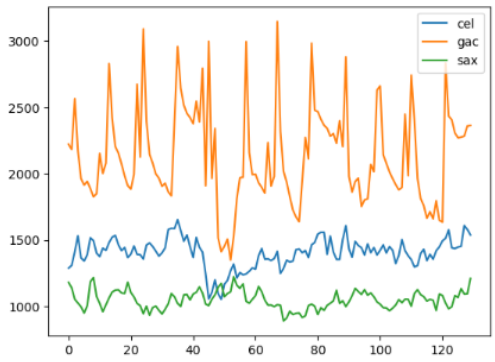
Slika 6: Spektralno središte

Spektralna širina pojasa je izdvedena iz spektralnog pojasa. Iako njihovi grafovi mogu izgledati dosta slično (Slika 6 i slika 7), zapravo opisuju dvije

različite stvari. Spektralna širina pojasa je razlika između donje i gornje granične frekvencije unutar signala. Recimo da se signal sastoji od 4 frekvencije: 1000Hz, 700Hz, 300Hz, i 100Hz, onda je spektralna širina pojasa:

$$1000 - 100 = 900Hz \tag{2}$$

Spektralna širina pojasa usko je povezana uz našu percepciju boje zvuka.



Slika 7: Spektralna širina pojasa

3 Jednostavni modeli

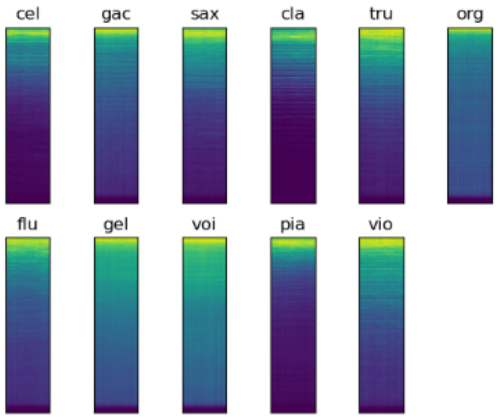
S obzirom na to da rješavamo problem klasifikacije, trebalo bi prvo isprobati neke od starijih metoda rješavanja tog problema pa ćemo razmotriti tri različita jednostavna modela te ih usporediti. Također ćemo gledati kako osnovne obrade signala iz prijašnjeg poglavlja utječu na točnost klasifikacije. Napomenut ćemo da se sljedeći modeli isprobavaju samo na problemu monofonije.

3.1 Model prosjeka

Jedna od prvih ideja koja bi nam mogla pasti napamet kada klasificiramo instrumente je da uzmemo sve zvučne zapise nekog instrumenta, zbrojimo ih i podijelimo s brojem zvučnih zapisa. Time smo dobili neki prosječni primjer zvučnog zapisa za taj instrument. Isto bismo napravili za sve ostale instrumente. Kada bismo htjeli raspoznati instrument na novom zvučnom zapisu, izračunali bismo njegovu udaljenost do prosjeka svakog instrumenta i odabrali onaj s najmanjom udaljenošću.

Ovakav model postiže točnost od 13% što je neznatno bolje od modela koji bi pogađao nasumično. No već kada koristimo spektograme umjesto originalnog signala, dobivamo gotovo trostruko veću preciznost, 34%.

Sljedeća slika 8 prikazuje kako izgleda prosječni spektogram za sve instrumente. Razlike između nekih instrumenata su lako primjetive.



Slika 8: Težine uprosječnog modela

3.2 K-najbližih susjeda

Problem prethodne metode je da uprosječavanjem snimaka zapravo gubimo podatke jer različite snimke istog instrumenta mogu znatno drugačije zvučati. Tako će određena snimka violončela možda više sličiti prosječnoj snimci klarineta.

Jedna nadogradnja mogla bi biti da u model spremamo sve snimke instrumenata iz skupa podataka za treniranje i ako želimo predvidjeti koji instrument se nalazi na novoj snimci, usporedit ćemo tu snimku sa svim zvučnim zapisima koje smo spremili u model. Metodom K-najbližih susjeda uzet ćemo K zapisa kojima je razlika između neviđene snimke i tog zapisa najmanja. Ovisno o tome koji od instrumenata je najučestaliji u tih K zapisa, taj instrument će biti predviđanje modela.

Korištenjem spektograma i uz ovakav pristup znatno poboljšavamo točnost modela. Točnost je dosegla čak 70%, a zanimljivo je primijetiti i ovisnost točnosti modela o drugačijem parametru K. Model najtočnije radi uz $K = 1$, a sve lošije za veće K-ove. Ovo je suprotno očekivanju jer ovakva metoda obično radi loše za jako niske ili ekstremno visoke K-ove i obično je potrebno isprobavati za koji parametar K metoda najbolje funkcionira. Ovakvo ponašanje metode K-najbližih susjeda moglo bi se objasniti određenim karakteristikama unutar samog skupa podataka (za ovaj dio testiranja, koristili smo manji podskup *IRMAS_training* skupa podataka) koji često ima više snimki istih instrumenata koje zvuče gotovo identično. Pa čak i ako podijelimo skup podataka na skup za treniranje i skup za testiranje, oba skupa će imati neke snimke koje su međusobno jako slične.

3.3 Logistička regresija

Univarijatna logistička regresija rješava problem binarne klasifikacije za zadan skup nezavisnih varijabli (primjerice vjerojatnost je li određena osoba glasala ili nije). Služi se raznim optimizacijskim postupcima da minimizira pogrešku pri klasificiranju na zadanom skupu podataka.

Glavna promjena kod multivarijatne logističke regresije je u broju izlaza, što nam omogućuje višerazrednu klasifikaciju. Naravno, postoji tu još znatnih razlika kao različite aktivacijske funkcije, ali to smo detaljnije objasnili u tehničkoj dokumentaciji.

Kako univarijantna regresija nije primjenjiva na naš problem, mi smo koristili multivarijantnu logističku regresiju. Metoda je davala neznatno lošije rezultate od metode K-najbližih susjeda s čak 69% preciznosti, što je i dalje vrlo dobar rezultat.

3.4 Zaključak

Sljedeća tablica 9 prikazuje točnost različitih kombinacija modela i upotrijebljenih transformacija ulaznih podataka.

	Average model	K-neighbours	Logistic regression
X original	0.13	0.17	0.17
X fft	0.28	0.63	0.53
X spec	0.34	0.71	0.69
X centroid	0.19	0.2	0.15
X bandwidth	0.22	0.27	0.16

Slika 9: Preciznost jednostavnih modela

Korištenje originalnog audio zapisa očekivano daje najlošije rezultate, a najučinkovitija metoda ispostavila se metoda K-najbližih susjeda u kombinaciji sa spektogramima.

Korištenjem centroida ne dobivamo puno bolji rezultat od korištenja originalnog signala, što ima smisla jer centroid opisuje svjetlinu zvuka, a svjetlina zvuka nije nužno toliko povezana s određenim instrumentom već s tom specifičnom snimkom.

Malo bolje rezultate dobivamo kada koristimo spektralnu širinu pojasa kao ulazne podatke jer ona opisuje boju zvuka koja je puno uže povezana uz pojedinačni instrument.

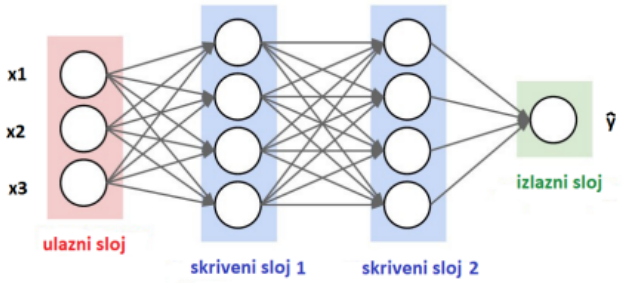
Iako se korištenjem spektograma dobivaju najbolji rezulati od svih transformacija ulaznih podataka, potrebno je naglasiti da se ovim metodama klasifikacije ne postiže potpuna prednost sačuvanja vremenske domene kod spektograma (u usporedbi s čistim izlazom iz Fourierove transformacije koja ni nema vremensku os). Potpuna prednost spektograma izrazit će se korištenjem konvolucija, što ćemo objasniti u sljedećem poglavlju.

4 Zašto konvolucijske mreže?

U ovom poglavlju objasnit ćemo razliku između običnih neuronskih mreža i onih koje koriste konvolucijske slojeve. Pokazat ćemo zašto su konvolucijski slojevi bitni u našem modelu. Podrazumijevat će se da čitatelj zna osnove neuronskih mreža i perceptrona jer bi njihovo objašnjavanje previše oduzelo od temeljnog predmeta ovog rada (stvaranje modela za klasifikaciju instrumenta).

4.1 Obične neuronske mreže

Ovakve neuronske mreže sastoje se od ulaznog sloja, jednog ili više skrivenih slojeva, te izlaznog sloja. Glavna mana ovakvog modela, kao i modela isprobanih u poglavlju 3 jest što ne čuvaju ovisnost koordinatnih osi na ulaznim podacima. Što znači da nešto kao spektogram, koji je dvodimenzionalna matrica, ovakvi modeli efektivno doživljavaju kao vektor. Ovaj problem se obično rješava dodavanjem konvolucijskih slojeva.

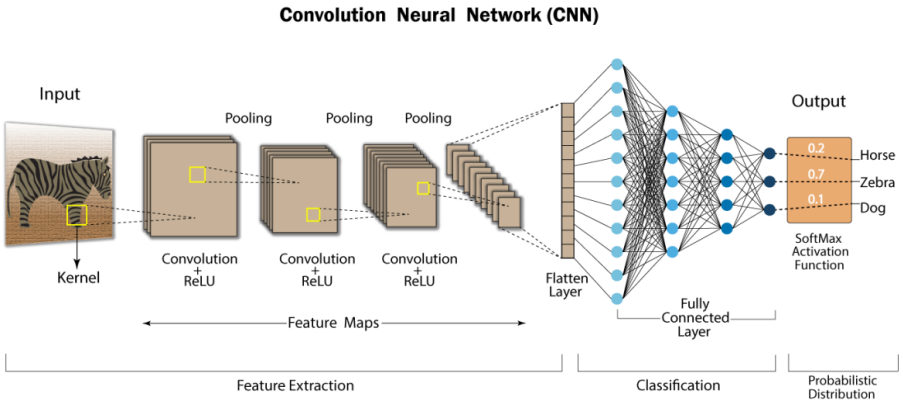


Slika 10: Neuronska mreža sa potpuno povezanim slojevima

Iako običnu neuronsku mrežu nismo isprobali, uvjereni smo da bi davala loše rezultate, neznatno bolje od onih u modela isprobanih u poglavlju 3, a za problem polifonije bila bi praktički neprimjenjiva.

4.2 Konvolucijske mreže

Kod konvolucijskih mreža ulazni podaci prvo prolaze kroz nekoliko konvolucijskih slojeva kombiniranih sa *Max Pooling* slojevima. Ovaj dio arhitekture mreže zovemo izvlačenje značajki (engl. feature extraction). Izlaz iz prethodnih slojeva se zatim sažima u vektor koji prolazi kroz nekoliko potpuno povezanih slojeva koji zapravo obavljaju klasifikaciju na izvučenim značajkama. Slika 11 prikazuje objašnjeni postupak.



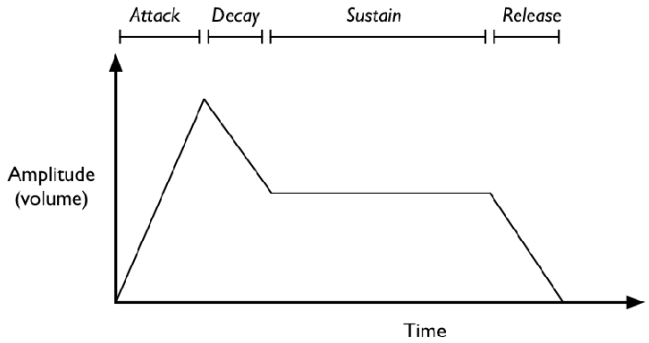
Slika 11: Konvolucijska neuronska mreža

4.3 Izvlačenje značajki

Ovdje ćemo detaljnije proći kroz proces izvlačenja značajki unutar konvolucijskih slojeva i koje značajke se nadamo da će naš model izvući u svrhu temeljnog problema.

Konvolucijske mreže se uglavnom primjenjuju na probleme klasifikacije slika, stoga će naš proces izvlačenja značajki izgledati malo drugačije nego što je uobičajeno. Kod slika, model se obično nauči izvlačenju značajki kao što su oštri rubovi, vertikalne i horizontalne linije i slično.

Spomenuli smo boju zvuka kao jednu od temeljnih značajki po kojoj bi model mogao zaključiti o kojem instrumentu je riječ. Ako pogledamo još jednom sliku 4, boja zvuka okarakterizirana je upravo odnosom između primarnih i sekundarnih harmonika. Pretpostavka je da će konvolucijska mreža moći izvući ovakvu značajku iz spektograma i time okarakterizirati trenutnu snimku. Potpuno povezani slojevi će kasnije pomoću te i ostalih značajki napraviti klasifikaciju.



Slika 12: Ovisnost glasnoće zvuka o vremenu

Još jedna glavna razlika između instrumenata jest *attack time*. To se ukratko može objasniti kao vrijeme koje instrumentu treba da nakon početka sviranja note, ta nota dosegne maksimalnu glasnoću. Na primjer, kada odsviramo notu na gitari, gotovo instantno se čuje odsvirana nota, a onda polako zvuk postaje sve tiši, dok na violini od početka povlačenja gudala treba malo dulje dok ta nota dosegne najveću glasnoću. Gitara ima kraći *attack time* od violine.

Ova značajka je izravno ovisna i o vremenskoj domeni i o frekvencijskoj. Zato je bitno koristiti spektrograme u kombinaciji sa konvolucijskim slojevima jer konvolucijskim slojevima čuvamo ovisnost domena ulaznih varijabla.

Slično kao i *attack time*, postoji i *release time*. To je vrijeme koje je potrebno da nakon završetka sviranja note, taj zvuk potpuno iščezne. Obično će instrumenti sa većim tijelima imati puno veći *release time* od manjih instrumenata.

Ovo su neke od značajki koje bismo mi očekivali da će model izvući, no modelove stvarne značajke po kojima će kasnije klasificirati instrumente obično budu vrlo apstraktne i drugačije od onog što bismo očekivali. Unatoč tome, model će svejedno pomoću njih naučiti raspoznavati instrumente.

4.4 Kreiranje modela

Nakon proučavanja podataka i isprobavanja nekih metoda za rješenje problema otkrili smo dvije ključne karakteristike koje bi uspješan model za klasifikaciju instrumenata trebao imati:

- Korištenje spektrograma,
- Konvolucijske slojeve.

4.4.1 Preneseno učenje

Strojno učenje je drastično napredovalo kroz posljednje desetljeće. Popularna metoda kod razvoja novih modela jest preneseno učenje (engl. transfer learning). Ova metoda fokusira se na primjenu znanja stečenog tijekom rješavanja jednog zadatka na sličan zadatak. Često znatno ubrzava vrijeme treniranja u dubokim mrežama i povećava preciznost modela.

Kako bismo primjenili ovu metodu trebamo uzeti već postojeći model koji je treniran na jako puno podataka i onda ga dotrenirati na našim podacima. Stvara se jedan problem, da ne postoji baš javno dostupnih modela koji su trenirani na glazbi, najviše je onih modela koji su trenirani za klasifikaciju slika. Ali mi ćemo transformirati ulazne podatke u spektrograme, a oni se mogu interpretirati kao slike. Sada će pronalaženje baznog modela biti lakše. Primjerice javno dostupni *resnet* modeli trenirani su već na puno slika. Iako možda nisu trenirani na spektrogramima, jako su dobri u prepoznavanju značajki i njihovo korištenje ubrzat će proces treniranja.

4.4.2 Ulazi različitih dimenzija

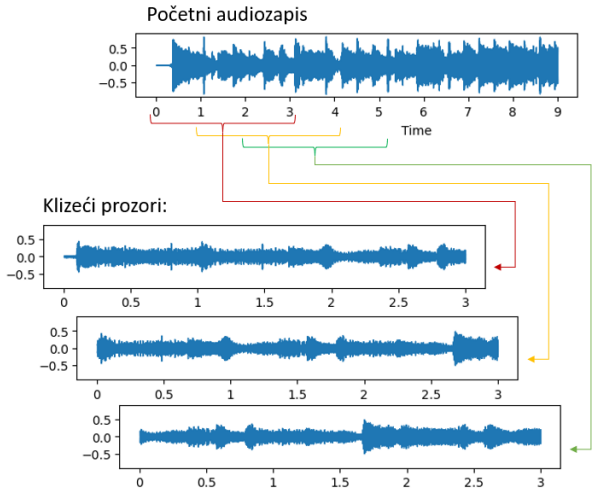
Jedan poznati problem kod strojnog učenja je kako riješiti ulaze različitih dimenzija. Računalo ne može u istu mrežu prvi put primiti sliku veličine 50×50,

a drugi put 100×100. Kod nas se problem javlja u vremenskoj domeni. Spektrogrami mogu uvijek imati isti raspon frekvencija, ali nekad će pjesma biti dulja, a nekad kraća. Taj problem se rješava na različite načine, a mi smo promotrili sljedeće tri metode:

1. **Skaliranje ulaza:** spektogram promatramo kao sliku i koristimo već postojeće algoritme za skaliranje slike (*squish* ili *crop*).

2. ***Zero padding*:** ideja je da odredimo maksimalnu veličinu vremenske osi (recimo 20 sekundi) i ako je neka pjesma kraća od toga, produljimo ju tako da traje 20 sekundi. Iako bi se model naučio da u praznim dijelovima snimke nema instrumenata, vrijeme treniranja bi bilo znatno dulje jer: 1) dimenzije ulaza su ekstremno velike i 2) ne koristimo efikasno procesorsku snagu jer će računalo raditi beskorisne operacije nad praznim dijelovima pjesme za koje znamo da nemaju instrumente.

3. **Klizni prozor:** modelu je ulaz uvijek fiksne veličine, recimo tri sekunde. Ako je pjesma dulja od toga, radvojit ćemo ju na isječke od tri sekunde (s razmacima od jedne sekunde). Svaki isječak ćemo provući kroz model i konačni rezultat će biti unija rezultata za svaki isječak.



Slika 13: Klizeći prozor

Metoda za koju smo se na kraju odlučili jest metoda kliznog prozora s veličinom prozora od tri sekunde. Pritom ako je pjesma kraća od toga koristimo *zero padding*.

Za metodu skaliranja ulaza (spektograma, ne originalnog signala) mislimo da ćemo izgubiti previše informacija jer je skaliranje s većih dimenzija na manje *lossy*, što znači da se neke informacije nepovratno izgube. Preciznije, gube se

informacije na vremenskoj domeni koje smatramo da su i dalje bitne za bolje modele.

No s obzirom na prirodu konvolucijskih mreža, moguće je da bi i ova metoda davala dobre rezultate uz ispravne augmentacije. Već se pokazalo da dobro radi na problemu klasifikacije slika gdje su često dimenzije detektiranih objekata različite.

5 Optimizacija modela i hiperparametara

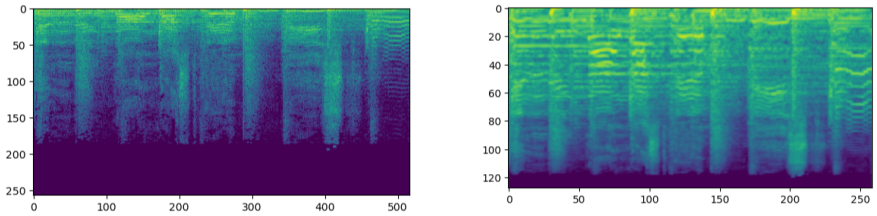
Dvije vrlo bitne stvari u treniranju svakog modela su sljedeće:

- **Ulazni podaci**: Što sve ulazi u model? Koje transformacije su najpogodnije za naš zadatak i mogu modelu olakšati učenje? Koje su augmentacije najbolje da se model navikne i na potpuno nove scenarije?
- **Izlazni podaci**: Koji treshold ćemo odabrati i kako ćemo ga primijeniti na model? Hoćemo li koristiti *vreće modela* i uprosječiti rezultate?

To su stvari na koje ćemo se najviše fokusirati u ovom poglavlju. Isprobat ćemo je li bolji spektogram ili melspektogram kao ulaz, koje su augmentacije najbolje te kako napraviti najbolji klizeći prozor.

5.1 Usporedba spektograma i melspektograma

Što je uopće melspektogram? Ukratko, to je isto što i spektogram uz to da frekvencije pretvori u **mel** skalu. Mel skala je skoro logaritamska te je sličnija ljudskom načinu percepcije zvuka. Pogledajmo razlike na slici:



Spektogram Melspektogram

Slika 14

Ideja korištenja melspektograma u modelu je da sažme bitne informacije i dio nebitnih informacija (visoke, neuptorebljene frekvencije) izbaci iz jednadžbe. Sljedeća tablica prikazuje treniranje modela na spektogramu i melspektogramu usporedno.

epoch	train_loss	valid_loss	acc	time
0	0.781949	0.442999	0.851125	02:19

epoch	train_loss	valid_loss	acc	time
0	0.508215	0.384117	0.860590	02:26
1	0.446415	0.378075	0.865505	02:27
2	0.415693	0.342985	0.870948	02:24
3	0.401005	0.346475	0.869973	02:27

Treniranje sa spektogramom

epoch	train_loss	valid_loss	acc	time
0	0.778834	0.420277	0.855228	05:46

epoch	train_loss	valid_loss	acc	time
0	0.518117	0.359464	0.864368	07:02
1	0.452723	0.338555	0.871598	07:06
2	0.414402	0.343960	0.868876	06:56
3	0.401446	0.334669	0.872492	07:04

Treniranje s melspektogramom

Slika 15

Ne samo da preciznost modela s melspektogramom nije puno bolja, već je model drastično sporiji. U konačnici, melspektogram i dalje sadrži iste informacije kao i spektogram, samo su drugačije skalirani. Zbog same prirode neuronskih mreža, nije uopće začuđujuće da rezultati nisu puno bolji. Neuronska mreža vrlo brzo može naučiti novo skaliranje ako je ono potrebno te ovakva transformacija ne olakšava previše posao. Zbog svega navedenog dalje ćemo koristiti **samo spektograme**.

5.2 Augmentacije podataka

Mnoge augmentacije koje smo primjenjivali zapravo su inspirirane nekim augmentacijama za problem klasifikacije slika: pomak slike → translacija, šum na slici → šum, crop → tišina. Osim toga, čine se vrlo prirodne za analizu zvuka, a njihovu učinkovitost ćemo mjeriti kasnije.

1. **Translacija**

   Translacija pomiče početak datoteke na malo kasniji trenutak. Možemo ju zamisliti kao rotaciju bitova u binarnom zapisu. Možda je najbitnija od svih augmentacija jer uopće ne uništava podatke, a opet pomaže modelu da se nauči prepoznavati značajke u svakom trenutku zvučnog signala.

2. **Tišina**

   Dijelove datoteke ćemo ugasiti tako da model nauči prepoznavati samo iz dijelova zvuka.

3. **Dodavanje šuma**

   Na pjesmu ćemo dodati nasumične brojeve prigodne amplitude tako da se stvori otpornost na greške u podacima.

4. **Mijenjanje visine zvuka**

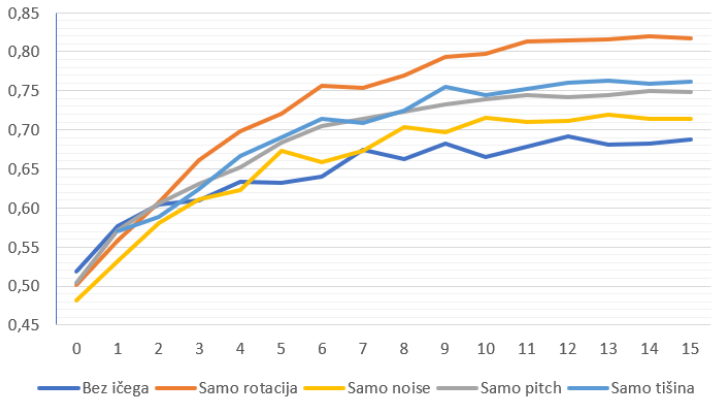
   Koristeći alate iz biblioteke *librosa*, pomaknut ćemo visinu zvuka datoteke za nasumičan broj.

5. **Spajanje više datoteka**

   Ova augmentacija vrlo je bitna za prepoznavanje polifonskih datoteka. U skupu podataka za treniranje nalaze se samo pojedinačni instrumenti te kako bismo simulirali pravi ulazni podatak, zbrojit ćemo više nasumičnih datoteka. Svaku od njih dodatno mijenjamo drugim augmentacijama.

Testirat ćemo ih prvo na problemu za monofoniju jer je treniranje brže i jasnije je što se događa. Sve augmentacije primjenjujemo na zvuku, prije nego iz njega napravimo spektogramu.

5.2.1 Usporedbe pojedinačnih augmentacija



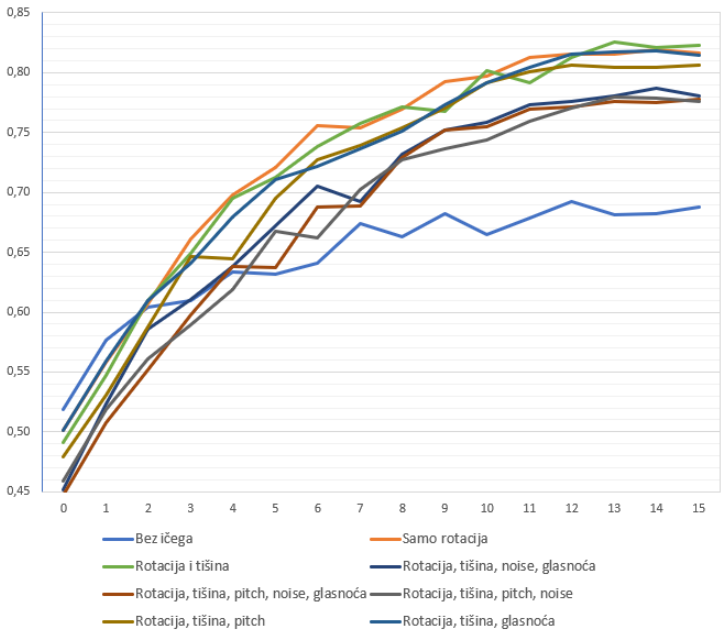
Slika 16: Jedna augmentacija

Kao što smo i pretpostavili, najučinkovitija augmentacija jest translacija [1], no svaka donosi pozitivne rezultate. Bitno za primijetiti je da je u prvih nekoliko epoha ne-augmentiranje najbolji izbor (primjerice šum pretiče ne-augmentiranje tek u 4. epohi). To se događa jer augmentiranje često napravi *lošije*[2] podatke nego one s kojima smo krenuli, no svaki put ih izmijeni pa je mreža bolje pripremljena na potpuno nove podatke.

[1] Na grafu je translacija označena kao rotacija

[2] Lošije u smislu da više nisu tako čiste, već su distorzirane. Najbolji primjer je augmentacija šuma koja može potpuno uništiti zvuk ako je pretjerano primjenja

5.2.2 Korištenje više augmentacija

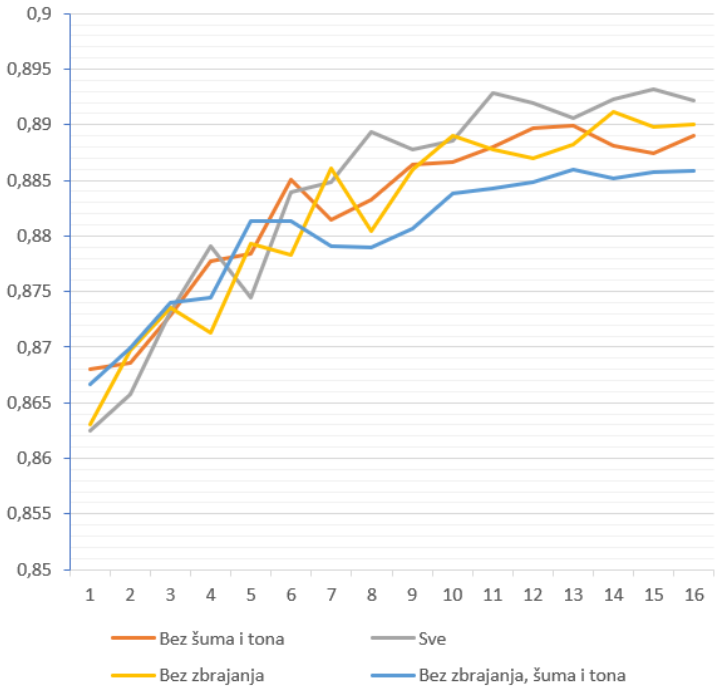


Slika 17: Više augmentacija

Naši rezultati pokazuju suprotno. Očigledno metoda papar i sol nije dobila ime samo zato što su hrani potrebni začini nego i po tome što se može prezačiniti. Razlike nisu velike, ali promjena tona i dodavanje šuma svakako smanjuje preciznost modela. Pravo je pitanje prenose li se ovi zaključci i na problem polifonije gdje je raznolikost podataka puno bitnija?

5.2.3 Polifonija

Sljedeći graf prikazuje preciznost modela za polifoniju u kojem primjenjujemo različite kombinacije augmentacija. Pokušali smo isključiti one augmentacije koje su se u prošlom poglavlju pokazale suvišnim te pokazati učinak metode zbrajanja više datoteka.



Slika 18: Augmentiranje u problemu polifonije

Kako smo analizirali na početku, u polifoniji su razlike od samo 1% vrlo značajne. Najbolju preciznost postiže model sa svim augmentacijama te se vidi da je zbrajanje datoteka dosta pridonijelo većoj preciznosti.

Trebalo bi napomenuti da se sve augmentacije ne događaju uvijek nego u određenom postotku slučaja, oko 50%. U početku smo testirali s većim postotcima, ali takvi rezultati su bili lošiji.

5.3 Pravi šum

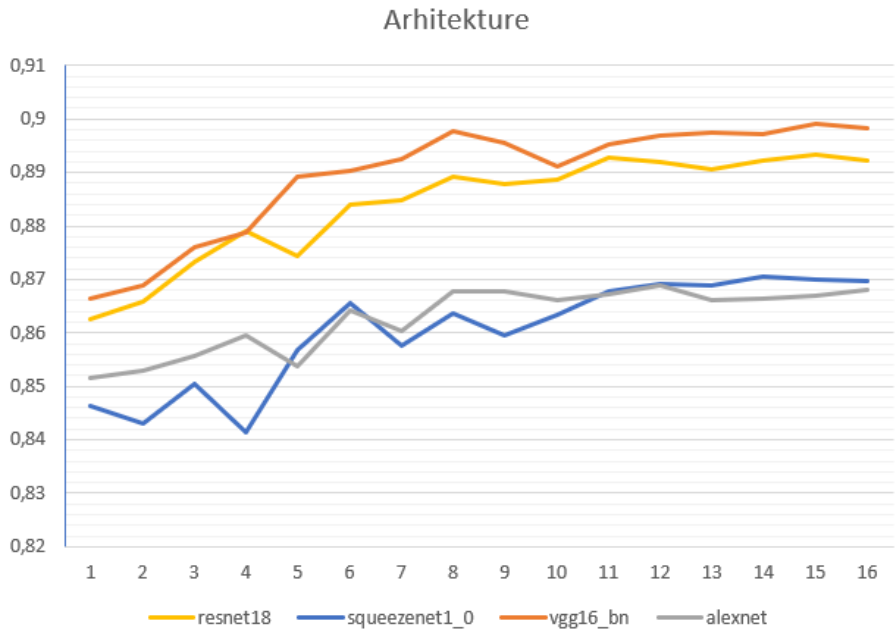
Pokušali smo proširiti skup podataka za treniranje sa zvukovima šuma. S interneta smo preuzeli oko 50 datoteka nasumičnih zvukova (valova, lavova, bušilice, grmljavine itd.) Osim toga smo digitalno kreirali oko 50 instrumentalnih datoteka koje nemaju nijedan instrument koji bi se trebao prepoznavati, već koristi neke druge. Trenirali smo model na tako proširenom datasetu.

Svrha ovog proširenja je naučiti model da postoje zvukovi koji možda nisu instrumenti i tako isključiti mogućnost lažne sigurnosti.

Nažalost, metoda nije pridonijela boljom preciznošću, već ju je samo pokvarila. Smatramo da je to zbog velike razlike u zvukovima prirode i instrumenata te već vrlo niskog broja lažno pozitivnih. Glavni problem modela su ipak lažno negativni (kada model ne misli da svira nijedan instrument).

5.4 Različite arhitekture

Sada ćemo testirati koja arhitektura najbolje funkcionira. Pokušali smo koristiti one lako dostupne iz biblioteke *fastai*.

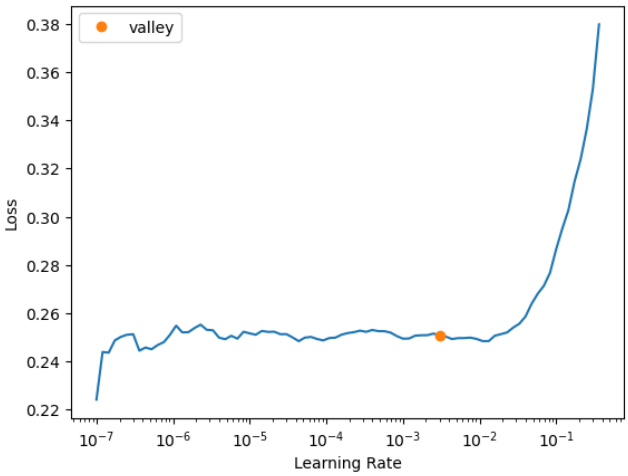


Slika 19: Arhitekture

Arhitektura *Vgg16* je postigla najbolje rezultate. No rezultati su došli uz cijenu jer je vrijeme treniranja prosječno 4:27 po epohi (dok je za ostale oko 2:13). Zbog toga ćemo nastaviti koristiti Resnet18.

5.5 Learning rate

Jedan od najbitnijih hiperparametara je *learning rate*. To je brzina kojom se model spušta u smjeru gradijenta (smjer najvećeg pada funkcije gubitka). Nije dobro kada je ta vrijednost prevelika niti premala. Stoga smo u ovom trenutku iskoristili metodu iz biblioteke *fastai* koja pronalazi najbolji learning rate za model. Primjer grafa ovisnosti learning ratea i pada loss funkcije prikazan je na sljedećoj slici:



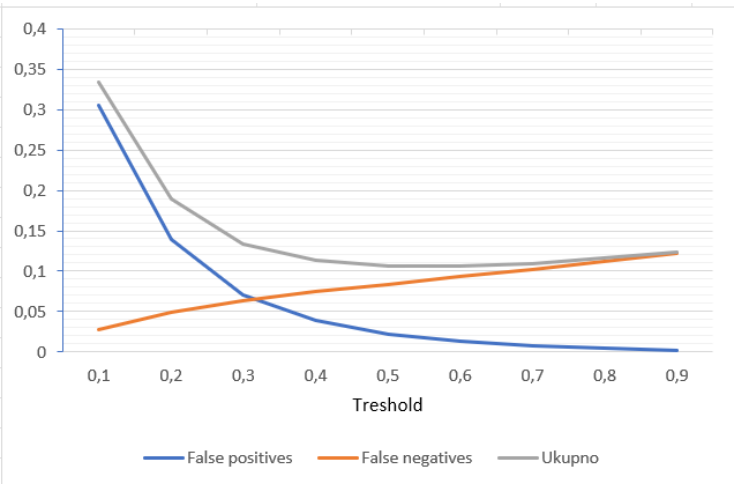
Slika 20: Learning rate

Dobar learning rate ovisi o vremenu. Model koji je već dobro treniran trebat će manji learning rate, dok će početni model trebati veći. Korištenjem ispravnog learning ratea, preciznost modela povećali smo s 89.2% na 89.6%.

5.5.1 Optimizacija praga za klasifikaciju

Prag za klasifikaciju je vrijednost na kojoj model *odsijeca* predviđanja. Njom izravno možemo utjecati na omjer lažno pozitivnih i lažno negativnih.

Osim što je samo po sebi vrlo bitno, trebamo imati na umu da kasnije želimo implementirati klizeći prozor. Ako će trenutni model imati grešku (pogotovo lažno pozitivne), ona će se dodatno akumulirati u klizećem prozoru.



Slika 21: Testiranje s različitim tresholdima

Optimalan prag za ukupnu grešku je između 0.5 i 0.6, ali kako ćemo u konačnici koristiti klizeće prozore, vjerojatno će biti bolje odabrati veći prag. Veći prag će možda imati malo veću grešku, ali i manje lažno pozitivnih, što znači da kada model napravi predviđanje, ono će vrlo vjerojatno biti dobro. To i želimo, jer ćemo kasnije na jednoj datoteci od 20 sekundi imati 18 nagađanja i ne bi bilo dobro da među njima ima puno lažno pozitivnih.

5.6 Analiza uspješnosti

Sljedeća tablica pokazuje postotak lažno pozitivnih (LP) i lažno negativnih (LN) u cijelom validacijskom datasetu za svaki instrument. Dodatno, za svaki instrument smo izračunali ukupnu grešku i podijelili ju s brojem pojavljivanja da dobijemo relativnu grešku:

$$R_{inst} = \frac{(LP_{inst} + LN_{inst})}{broj_{inst}}$$

.

Inst	LP	LN	Pojavljivanja	Relativna gr.
cel	0.313	3.619	3.8	103
cla	0	2.770	2.77	100
flu	0	6.077	6.43	95
gac	0.268	14.343	17.43	84
gel	0.402	29.491	34.45	87
org	1.385	11.215	13.23	95
pia	0.581	24.620	32.44	78
sax	0.268	5.853	9.2	67
tru	0.938	3.083	4.6	87
vio	0.089	6.345	7.15	90
voi	0.629	15.147	37.67	42

Tablica 1: Relativna greška po instrumentu

Iako nam se model činio vrlo uspješnim, ovakvi izračuni pokazuju koliko im je dorade potrebno. U svakom slučaju, smatramo da će se situacija barem malo popraviti jednom kada napravimo klizeći prozor.

Osim toga mislimo da bi trebalo više pažnje posvetiti svakom instrumentu zasebno. Možda bi bolji pristup bio napraviti jedan model za svaki instrument.

Među datotekama, najveću grešku model je imao na atmosferičnoj *modernoj* glazbi: *gyorgy ligeti- atmospheres*. Smatramo da je greška razumna, jer takva glazba zaista zvuči kaotično.

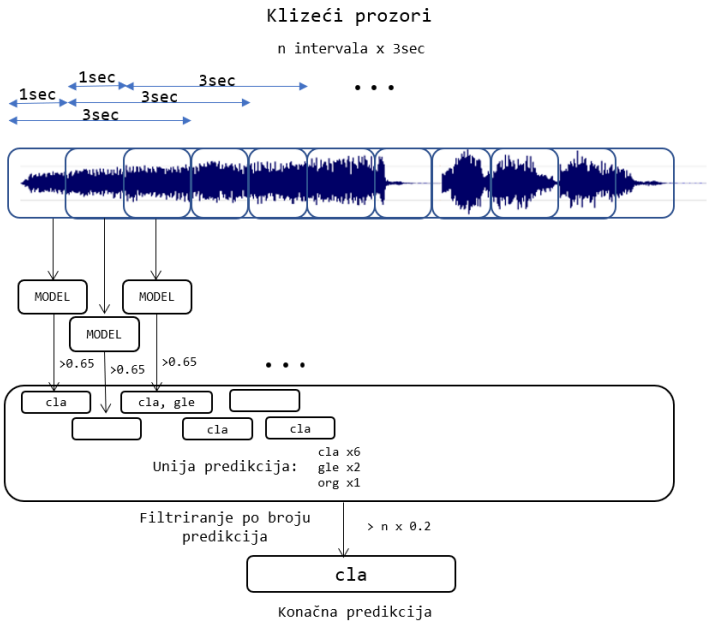
Također, glavni problem modela su lažno negativni. Vjerujemo da većina tog problema dolazi iz podataka. U skupu podataka *IRMAS_ Training_ Data* zapisani su samo dominantni instrumenti, a ne svi koji se pojavljuju. Zato se ponekad dogode situacije kada model možda točno pogodi sve instrumente koji se pojavljuju, ali za to bude kažnjen. Popravkom ove greške, vjerujemo da bi se preciznost modela **drastično** povećala. Greška se može riješiti na više načina:

- ispravnjem zapisa u *IRMAS_ Training_ Data*,
- eliminacijom pozadinskog zvuka. Ipak je jedan instrument dominantan i moguće je da postoje alati s kojim bi se ostatak zvuka mogao dobro eliminirati,
- korištenjem drugog skupa podataka za treniranje. Primjerice, dodatno podijeliti *IRMAS_ Validation_ Data* te iz njega uzeti 80% za treniranje, umjesto naših 20%.

5.7 Klizeći prozor

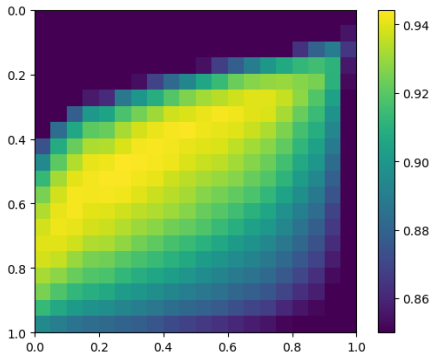
Jedino je preostalo optimizirati pristup sa klizećim prozorom. Model za svake tri sekunde, s pomakom od jedne sekunde, radi predviđanja. Ako je predviđanje za instrument veće od prvog praga $prag_1$ (eng. *treshold*), tada povećavamo brojač za taj instrument u datoteci za jedan. Ponovimo taj postupak za svaki prozor. Ako je na kraju brojač za instrument veći od $ukupanbrojprozora \times prag_2$, onda

kažemo da se u toj datoteci pojavljuje taj instrument. Grafički je postupak prikazan na sljedećoj slici za $prag_1 = 0.65$ i $prag_2 = 0.2$.



Slika 22: Klizeći prozor

Sada trebamo ta dva praga optimizirati. Optimizirat ćemo ih na polifonskom dijelu skupa za treniranje (20% iz *IRMAS_ Validation_ Data*), jer su jedino tamo pjesme dulje od tri sekunde. Sljedeći graf prikazuje ovisnost preciznosti na skupu po kojem optimiziramo i oba praga.



Slika 23: Različiti pragovi za klizeći prozor

Najbolji rezultat postiže se tamo gdje je $prag_1 = 0.46$, a $prag_2 = 0.26$.

Tada je preciznost na validacijskom skupu 89.779%. To je samo malo bolje od prethodnog rezultata 89.601%. Iznenadio nas je vrlo nizak $prag_1$ i mislimo da njega možemo dodatno popraviti.

Odlučili smo sada optimizirati oba praga po cijelom skupu za treniranje (zajedno s monofonskim datotekama). Dobivamo kombinaciju $prag_1 = 0.56$ i $prag_2 = 0.11$, s preciznošću od 90.043% pri validaciji.

No, najbolji rezultat dobivamo kombinacijom ta dva pristupa. Kada uzmemo optimalni $prag_2 = 0.26$ iz prvog pristupa i optimalni $prag_1 = 0.56$ iz drugog pristupa, dobivamo preciznost od čak 90.620%!

Na temelju analize u prošlom poglavlju, zaključili smo da možda vrijedi i optimizirati $prag_1$ za svaki instrument zasebno (jer su razlike lažno pozitivnih i negativnih vrlo velike među instrumentima), dok $prag_2$ držimo fiksiranim. Ovaj pristup ne daje toliko dobre rezultate jer dolazi do overfittanja. Preciznost na validacijskom skupu je samo 90.389%.

6 Aplikacija

Kao dodatak originalnom zadatku, napravili smo i aplikaciju. Aplikacija se sastoji od dva dijela, *backend-rest* poslužitelja i klijentske web aplikacije.

6.1 API

Ovo je dio aplikacije koji učitava model i uvijek biti spreman primiti *http* poziv koji sadrži zvučnu datoteku. Nakon toga šalje odgovor u obliku JSON-a koji sadrži popis svih instrumenata s bitnim vrijednostima nalazi li se taj instrument na snimci ili ne:

{ " cel " : 0 , " cla " : 1 , " flu " : 0 , " gac " : 0 , " gel " : 0 , " org " : 0 , " pia " : 0 , " sax " : 0 , " tru " : 1 , " vio " : 1 , " voi " : 0 }

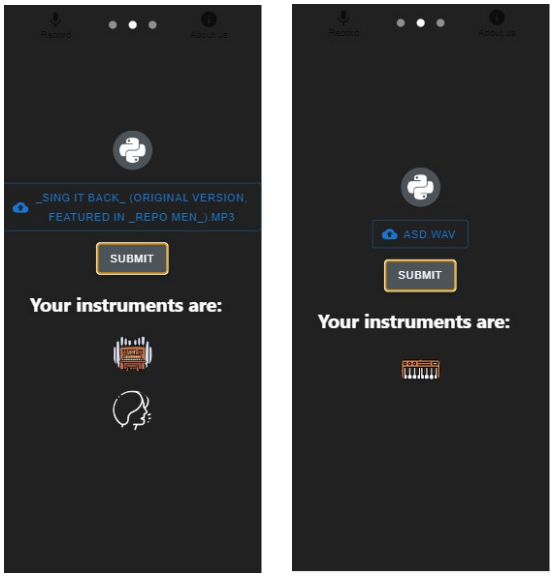
Budući da smo model razvijali u Python okruženju, najjednostavnije je bilo da i ovaj dio programa bude pisan u Pythonu. Python pruža biblioteku Django koju smo koristili da bismo lako napravili *backend-rest* poslužitelj.

6.2 Aplikacija

Osim *backend-rest* poslužitelja napravili smo i klijentsku web aplikaciju koja zapravo pokazuje kako bi se naš problem mogao primijeniti u stvarnome svijetu. Aplikacija je namijenjena za znatiženje slušatelje glazbe koje zanima koji se instrumenti nalaze na nekoj snimci, a možda nisu dovoljno glazbeno nadareni da to sami otkriju. Aplikacija se sastoji od tri stranice:

1. Glavna stranica
2. "Snimaj"
3. O nama

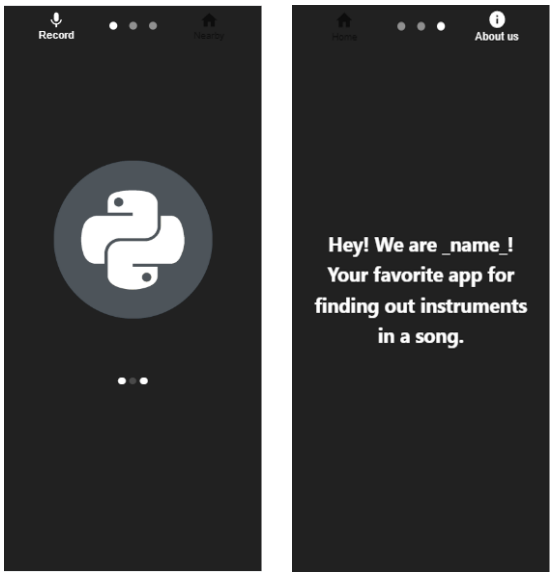
Na glavnoj stranici korisnik može odabrati zvučnu datoteku s uređaja i poslati ju. Pjesma se šalje poslužitelju koji ju obrađuje i vraća odgovor s popisom instrumenata. Odsvirani instrumenti se prikazuju kao ikonice.



Orgulje i glas Klavir

Slika 24: Glavna stranica

Kako bi aplikacija bila malo zanimljivija dodali smo stranicu gdje korisnik može uživo snimati zvuk (ako ima pristup mikrofonu) i snimljeni zvuk će se potom poslati na poslužitelj. Odgovor o snimljenim instrumentima se ponaša isto kao i na glavnoj stranici.



Stranica "Snimaj" "O nama"

Slika 25

Ako se poslužitelj ne može dohvatiti, poslana je ne-zvučna datoteka ili se dogodi druga greška na strani poslužitelja, klijent će to prikazati.

Nakon puno pokušaja, zbog tehničkih poteškoća, ovaj dio aplikacije i dalje ne radi najbolje. Web preglednici su nedavno ukinuli podršku za većinu tipova audio datoteka, a oni tipovi koji su podržani na web-u, pak nisu podržani u Python biblioteci koju koristimo.

Na trećoj stranici piše nešto o autorima aplikacije i projekta.

7 Moguća proširenja i poboljšanja

Model je vrlo jednostavno proširiti tako da radi na novim instrumentima. Potrebno bi bilo samo prilagoditi posljednji sloj i učitati već istrenirani model. Osim toga trebalo bi puno novih datoteka, ali zahvaljujući augmentaciji zbrajanja datoteka, dotreniravanje bilo bi puno jednostavnije. Ako broj instrumenata bude ogroman, trebalo bi jedino povećati veličinu hrpe (eng. *batch size*). Tada bi bili potrebni i značajniji računarni resursi.

Osim toga, preciznost modela možemo unaprijediti na sljedeće načine:

- **Bolji skup podataka**

  Kao što je rečeno u poglavlju 5.6, najbolje bi bilo napraviti bolji skup podataka ili možda drugačije podijeliti *IRMAS_ Validation_ Data*. Trenutni model ima najveći problem s lažno negativnim te smatramo da je to zbog

toga što su u *IRMAS_ Training_ Data*) označeni samo dominantni instrumenti, a ne svi.

- **Reccurent neural network**

  Povratne neuronske mreže su mreže koje dopuštaju da veze među vrhovima budu ciklusi i tako dopuštaju da trenutna vrijednost u vrhu utječe na sljedeće vrijednosti u tom vrhu. Vrlo su dobre za zadatke gdje se treba očuvati, primjerice, vremenska dimenzija, kad su uzastopni događaji usko povezani. Već su korištene u sličnim zadacima poput prepoznavanja raspoloženja pjesama[5] te mislimo da bi bile izvrsne u ovom zadatku.

- **Odvojeni modeli za svaki instrument**

  Kao što smo vidjeli u analizi greške, uspješnost modela se uvelike razlikuje po instrumentima. Vokali su vrlo uspješni, dok za violončelo model **uvijek** griješi. Stoga bi bilo dobro napraviti binarni model koji prepoznaje samo jedan instrument. U tom slučaju za različite instrumente mogli bismo koristiti i različite pristupe koji njima odgovaraju. Odvojeni modeli bi zauzeli puno memorije, ali dodavanje ne bi narušili prethodne modele. Također ne bi se zahtjevali značajni računarni resursi za treniranje novog modela, no njihova evaluacija bila bi sporija.

- **Korištenje informacije o pojavljivanju bubnjeva**

  Korištenje dodatnih informacija o datoteci (a koje nisu bitne za konačnu klasifikaciju) mogu pomoći treniranju mreže. Primjerice imamo hrpu fotografija biljaka riže. Neke prikazuju bolesnu biljku, a neke zdravu. Model treba raspoznati bolest od koje biljka boluje. Pokazalo se da ako model usput nauči raspoznavati vrstu riže, popravlja mu se i sposobnost raspoznavanja bolesti. Stoga smo u model mogli dodati zadatak prepoznavanja bubnjeva ili žanra (što je već označeno u *IRMAS_ Training_ Data*). Možda bi se tako povećala i njegova preciznost u prepoznavanju instrumenata.

8 Primjene

U ovom poglavlju analizirat ćemo gdje se ovakvi modeli za prepoznavanje instrumenata mogu koristiti:

- **Vađenje metapodataka o glazbi**

  Ovaj i slični modeli se mogu koristiti za vađenje metapodataka koji se kasnije mogu koristiti za razne stvari (objašnjeno u idućim točkama). Sličnu tehnologiju koristi aplikacija *Pinterest* koja iz slike stvara hashtag-ove pomoću kojih radi bolja pretraživanja. Trenutni metapodaci koje model može pronaći jest popis instrumenata, ali lako se može napraviti sličan model koji vadi druge podatke kao žanr glazbe, raspoloženje glazbe i slično. Također, broj instrumenata koji model klasficira se lako može proširiti.

- **Klasifikacija glazbe**

  Popis instrumenata i ostali metapodaci se mogu koristiti za klasifikaciju glazbe u žanrove. Žanr glazbe se kasnije može koristiti za bolje preporuke glazbe korisnicima u *song streaming* servisima kao što su **Spotify** ili *YouTube*.

- **Povijest glazbe**

  Možda neočekivana, ali vrlo dobra primjena bila bi u povijesti žanrova. Ovim alatom bilo bi puno lakše prepoznati prve pojave određenih zvukova u glazbi. Tako bismo mogli točno pronaći prvu *hip hop* pjesmu, *drill* pjesmu, *trap*, *house* i pjesme ostalih žanrova.

- ***Content based filtering***

  Jedan od glavnih problema preporučanja sadržaja korisniku je skupljanje korisnih podataka o datotekama. Automatsko prepoznavanje instrumenata moglo bi tome pomoći. Validna je pretpostavka da su instrumentali ljudima vrlo bitni. Primjerice osobe koje slušaju rock će prije voljeti električnu gitaru i bubnjeve nego violinu. Prednost koju pruža *content based filtering* je da ne ovisi o podacima drugih korisnika, nego može odmah pomoću metamodataka raditi preporuke na potpuno novim datotekama.

9 Izvori

[1] J. J. Bosch, J. Janer, F. Fuhrmann, and P. Herrera. "IRMAS: a dataset for instrument recognition in musical audio signals". In: (2012).

[2] Mohamed ElSalmi. "Instrument-Classificiation". In: (2020).

[3] J. Hartquist. "Audio Classification using FastAI and On-the-Fly Frequency Transforms". In: (2018).

[4] J. Howard. *Practical Deep Learning for Coders with fastai and PyTorch.*

[5] M. Malik, S. Adavanne, K. Drossos, T. Virtanen, D. Ticha, and R. Jarina. "Stacked Convolutional and Recurrent Neural Networks for Music Emotion Recognition". In: (2017).

[6] K. Racharla, V. Kumar, C. B. Jayant, A. Khairkar, and P. Harish. "Predominant Musical Instrument Classification based on Spectral Features". In: (2019).