



Data Capybara

Lumen Data Science 2023. Projektna dokumentacija

9.5.2023.

Sadržaj

1 Uvod	3
2 Analiza zvuka	4
2.1 IRMAS podatkovni skup	4
2.1.1 Brzina uzorkovanja i dubina bita	6
2.2 Fourierov red i transformacija	7
2.2.1 Diskretna Fourierova transformacija (DFT)	9
2.2.2 Short-Time Fourier Transform (STFT)	9
2.3 Značajke zvuka	11
2.3.1 Spektrogram	12
2.3.2 Mel-spektrogram	12
2.3.3 MFCCs (<i>Mel frequency cepstral coefficients</i>)	14
2.3.4 Spektralni centroid	15
2.3.5 Stopa prelaska nule (<i>Zero-Crossing Rate</i>)	16
2.3.6 Ostale značajke zvuka	17
3 Pretprocesiranje	18
3.1 Priprema podatkovnog skupa	18
3.2 Više različitih značajki zvuka	19
3.2.1 MFCC	19
3.2.2 Spektralni centroid	20
3.2.3 Stopa prelaska nule	21
3.3 Mel-spektrogram	22
3.4 Augmentacija	24
3.4.1 Dodavanje šuma	24
3.4.2 Promjena tona(<i>pitch shifting</i>)	25
3.4.3 Promjena vremena(<i>time stretching</i>)	26
3.4.4 Miješanje audio zapisa različitih instrumenata	27
3.5 Konačan odabir pristupa	27
4 Model	28
4.1 Neuronske mreže	28
4.2 Konvolucijske neuronske mreže	29
4.3 Tijek razvoja modela	30
4.3.1 Početni modeli	30
4.3.2 ResNet	31
4.3.3 ConvNeXt	32

4.3.4	DenseNet	33
4.4	Trening	34
4.4.1	Hiperparametri	34
4.4.2	Optimizator i funkcija gubitka	35
5	Rezultati	36
5.1	Model	36
5.2	Instrumenti	37
6	Zaključak	39

1. Uvod

U današnje vrijeme, razvoj tehnologije pronašao je mnogobrojne primjene u području glazbe i zvuka. Jedna od tih primjena je stvaranje modela za klasifikaciju instrumenata. Takav model može biti koristan u mnogim situacijama, kao što je automatsko prepoznavanje glazbenih instrumenata u snimljenim pjesmama, što može pomoći u organiziranju glazbenih baza podataka ili pretraživanju pjesama prema određenim instrumentima. Također, model za klasifikaciju instrumenata može biti koristan u stvaranju novih glazbenih djela, transkripciji glazbe za potrebe podučavanja, kao i u raznim drugim primjenama u glazbenoj industriji. Cilj ovog projekta je razviti model za klasifikaciju instrumenata koji će biti precizan i učinkovit u prepoznavanju različitih glazbenih instrumenata u audio zapisima.

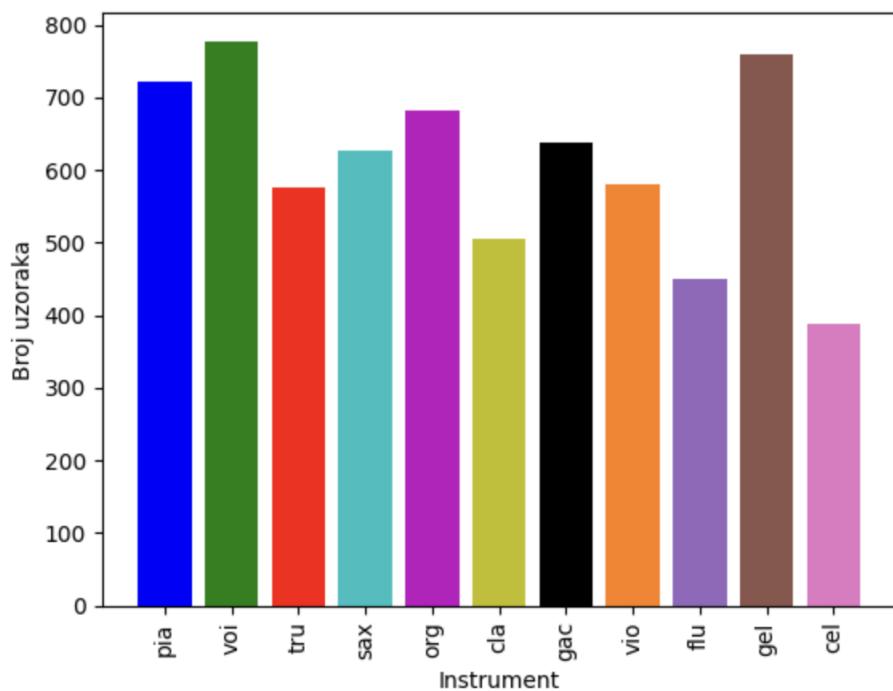
2. Analiza zvuka

2.1 IRMAS podatkovni skup

IRMAS(*Instrument recognition in musical audio signals*) podatkovni je skup koji uključuje audio snimke s napomenom o prisutnim instrumentima. IRMAS je namijenjen za treniranje i testiranje modela za prepoznavanje instrumenata u određenim audio zapisima. Razmatrani instrumenti i njihove oznake su: violončelo (cel), klarinet (cla), flauta (flu), akustična gitara (gac), električna gitara (gel), orgulje (org), klavir (pia), saksofon (sax), truba (tru), violin (vio) i ljudski pjevački glas (voi). Podatkovni skup sastoji se od skupa podataka za treniranje te skupa podataka za validaciju, odnosno testiranje.

Podatkovni skup namijenjen za treniranje sastoji se od 6705 audio zapisa u 16-bitnom stereo .wav formatu uzorkovanih na 44,1 kHz, duljine zapisa 3 sekunde.

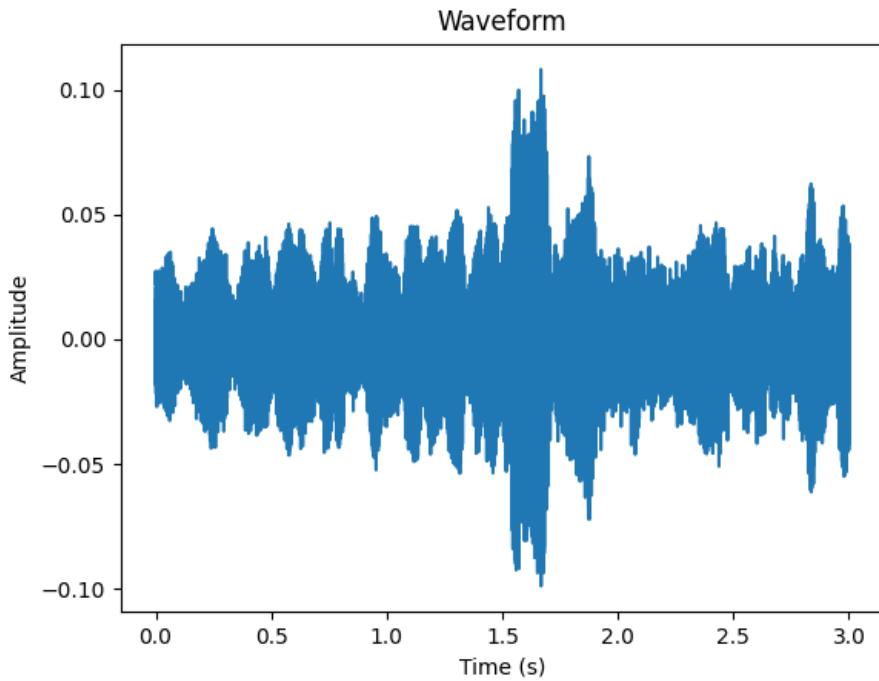
Validacijski podaci sadrže 2874 datoteka koje su također u 16-bitnom stereo .wav formatu uzorkovanih na 44,1 kHz. Za razliku od podataka namijenjenih za treniranje, validacijski podaci nisu fiksne duljine već im duljina varira između 5 i 20 sekundi te se u istom audio zapisu može pojaviti više različitih instrumenata.



Slika 2.1: Distribucija instrumenata u trening podacima

Možemo primijetiti da skup podataka namijenjen za trening nije u potpunosti uravnotežen. Iz priloženog grafikona vidimo da za neke instrumente kao što su električna gitara ili ljudski glas imamo skoro duplo više uzoraka nego za instrumente kao što su violončelo ili flauta.

Skup podataka za trening sastavljen je isključivo u svrhu treniranja modela za prepoznavanje instrumenata te zbog toga ne sadrži stršeće vrijednosti te nije potrebno dodatno čistiti podatke.

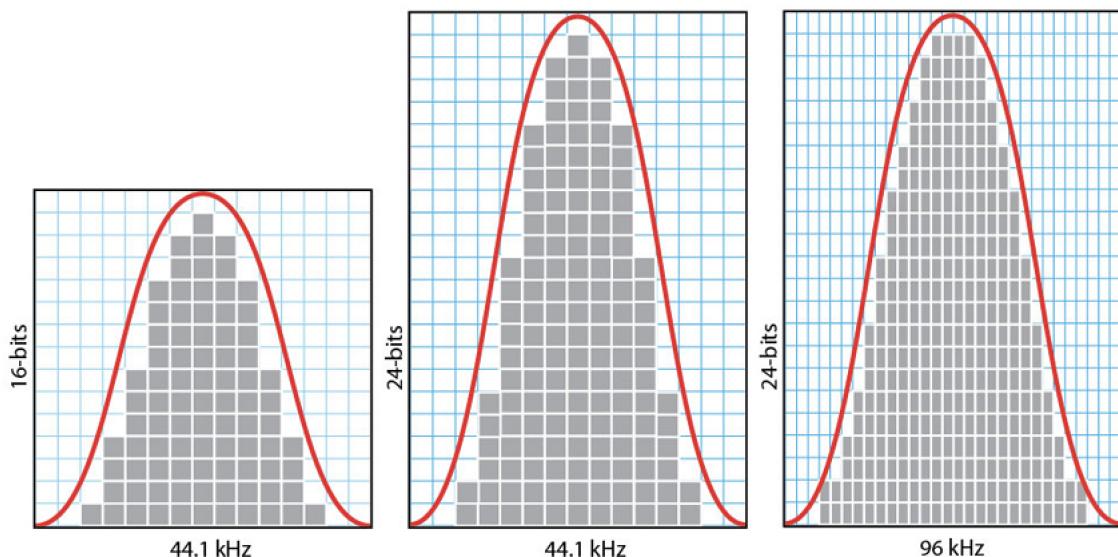


Slika 2.2: Prikaz waveform datoteke (violončelo)

Prvo ćemo se upoznati s nekim osnovnim svojstvima zvuka te ćemo proći tehnike za analizu zvuka koje će nam kasnije biti potrebne za izdvajanje i razumijevanje raznih značajki zvuka.

2.1.1 Brzina uzorkovanja i dubina bita

Digitalni zvuk prikaz je zvuka snimljenog ili pretvorenog u digitalni signal. Tijekom procesa analogno-digitalne pretvorbe (ADC), uzimaju se amplitude analognog zvučnog vala pri određenoj brzini uzorkovanja i dubini bita, te zatim se pretvaraju u podatke koje računalni softver može pročitati. Glavna razlika između zvuka i digitalnog zvuka je u tome što je digitalni zvuk niz vrijednosti amplitute koje se koriste za rekonstrukciju izvornog analognog zvučnog vala, dok je analogni zvuk kontinuirani signal s beskonačnim vrijednostima amplitute u bilo kojem trenutku u vremenu. Digitalni zvuk je kao "spajanje točaka", dok je analogni zvuk puna izvorna slika. Kao što smo već spomenuli, audio zapisi IRMAS podatkovnog skupa sastoje se od 16-bitnih stereo datoteka uzorkovanih na 44,1 kHz. Brzina uzorkovanja je broj uzoraka u sekundi koji se uzima kako bi se stvorio diskretni digitalni signal. Uzimanje previše uzoraka može stvoriti nepotrebno velike datoteke zvuka, dok uzimanje premalo uzoraka može dovesti do gubitka informacija i kvalitete zvuka. Dubina audio bita određuje broj mogućih vrijednosti amplitute koje možemo snimiti za svaki audio uzorak. Što je dubina bita veća, to se više vrijednosti amplitute po uzorku uzima za ponovno stvaranje izvornog audio signala.



Slika 2.3: Usporedba brzina uzorkovanja i dubina bita

2.2 Fourierov red i transformacija

Kako bi mogli razumjeti razne značajke zvuka te metode dobivanja istih, nužno je razumjeti Fourierove transformacije. Fourierov red je glavna ideja Fourierove analize, koja pokazuje da se svaka periodička funkcija može zapisati kao suma sinusa i kosinusa različitih amplituda, faza i frekvencija, te se prikazuje na sljedeći način:

$$x(t) = \frac{A_0}{2} + \sum_{n=1}^{\infty} (A_n \sin(n\omega_0 t) + B_n \cos(n\omega_0 t)) \quad (2.1)$$

gdje je:

- T_0 – osnovni period
- A_0 – istosmjerna komponenta, služi za translaciju funkcije duž y-osi
- A_n, B_n – Fourierovi koeficijenti reda n
- ω_0 – kutna frekvencija

$x(t)$ predstavlja signal u vremenskoj domeni.

Fourierovi koeficijenti u prethodnoj jednadžbi ovise o valnom obliku signala $x(t)$, iz toga slijede izrazi:

$$A_0 = \frac{2}{T_0} \int_0^{T_0} x(t) dt \quad (2.2)$$

$$A_n = \frac{2}{T_0} \int_0^{T_0} x(t) \sin(n\omega_0 t) dt \quad (2.3)$$

$$B_n = \frac{2}{T_0} \int_0^{T_0} x(t) \cos(n\omega_0 t) dt \quad (2.4)$$

gdje je:

- T_0 - osnovni period.

Analiza aperiodičkih signala zahtijeva drukčiji pristup, stoga je razvijena Fourierova transformacija, gdje je korištena metoda tretiranja aperiodičkog signala kao periodičkog. Iz toga proizlazi da osnovni period T_0 aperiodičkog signala teži u beskonačnost. Aperiodički signali nisu sastavljeni od diskretnih spektralnih komponenti, već imaju kontinuirani frekvencijski spektar s frekvencijski ovisnom spektralnom gustoćom.

Spektar kontinuiranih aperiodičkih signala definira se sljedećim jednadžbama Fourierove transformacije: Jednadžba analize:

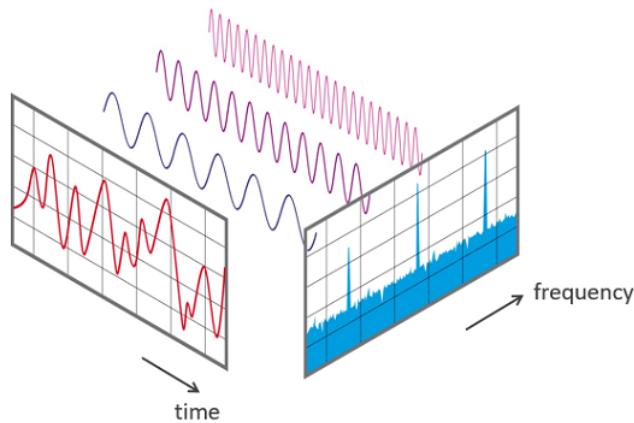
$$X(f) = \int_{-\infty}^{\infty} x(t) e^{-i2\pi f t} dt \quad (2.5)$$

Jednadžba sinteze:

$$x(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(f) e^{i2\pi f t} df \quad (2.6)$$

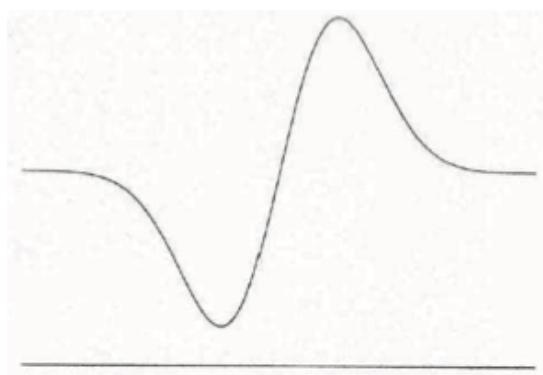
gdje je:

- $x(t)$ - signal u vremenskoj domeni
- $X(f)$ - signal u frekvencijskoj domeni
- f - frekvencija
- t - vrijeme
- ω - kutna frekvencija

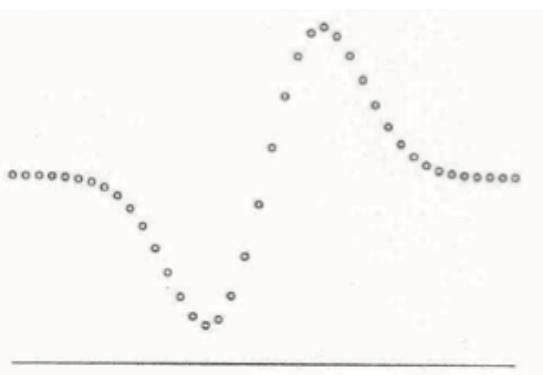


Slika 2.4: Fourierove transformacije

Fourierov red i Fourierova transformacija igraju važnu ulogu prilikom analize neprekinutog signala, dakako, u mnogim je primjenama signal diskretan skup podataka, kao na primjer signal koji nastaje od nosača zvuka. Zbog toga nam je potrebna diskretna verzija Fourierove transformacije, kako bismo analizirali i takve signale.



(a) Neprekinuti signal



(b) Diskretni signal

Pomoću već navedenih tehnika Fourierovih transformacija dobit ćemo informacije o frekvenčkim komponentama signala u prosjeku za cijelo vrijeme, dok signali poput glazbe i govora karakteriziraju načine na koje se frekvencijske komponente mijenjaju tijekom vremena. Upravo zbog te činjenice razvijena je tehnika STFT(*Short-time Fourier transform*) koju ćemo također promotriti u nastavku.

2.2.1 Diskretna Fourierova transformacija (DFT)

Diskretna Fourierova transformacija je diskretna verzija Fourierove transformacije, koja se koristi za obradu diskretnih signala. DFT se može definirati kao transformacija koja uzima niz N kompleksnih brojeva x_0, x_1, \dots, x_{N-1} , i stvara drugi niz N kompleksnih brojeva X_0, X_1, \dots, X_{N-1} , gdje je svaki X_k kombinacija svih x_n , pomnoženih s odgovarajućim kompleksnim faktorima. Preciznije, DFT se može izraziti sljedećim formulama:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi nk/N}, \quad k = 0, 1, 2, \dots, N-1 \quad (2.7)$$

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k e^{i2\pi nk/N}, \quad n = 0, 1, 2, \dots, N-1 \quad (2.8)$$

gdje je:

- x_n - n-ti uzorak signala u vremenskoj domeni
- X_k - k-ti uzorak signala u frekvencijskoj domeni
- N - duljina ulaznog niza, odnosno broj uzoraka u ulaznom signalu
- $e^{i\theta} = \cos \theta + i \sin \theta$ - kompleksni broj sa kutom θ

DFT se može primijeniti na bilo koji diskretni signal, bilo da je periodičan ili neperiodičan. Međutim, postoje određene pretpostavke koje se moraju zadovoljiti kako bi se DFT mogao primijeniti na signal. Signal mora biti ograničen na konačno razdoblje i mora biti diskretiziran, što znači da su uzorci signala dostupni samo u diskretnim vremenskim intervalima.

2.2.2 Short-Time Fourier Transform (STFT)

Short-time Fourier transform (STFT) je vrsta Fourierove transformacije koja se primjenjuje na vremenski varijabilne signale i koja pruža informacije o frekvencijskoj strukturi signala u kratkim vremenskim intervalima. STFT se dobiva primjenom Fourierove transformacije na mali prozor širine T_w signala $x(t)$ koji se pomakne za fiksni vremenski korak ΔT . Prozor se obično naziva i funkcijom prozora, a često se koriste funkcije s ugrađenim svojstvima poput Hamming prozora.

$$STFT(t, f) = \int_{-\infty}^{\infty} x(\tau) w(\tau - t) e^{-j2\pi f \tau} d\tau \quad (2.9)$$

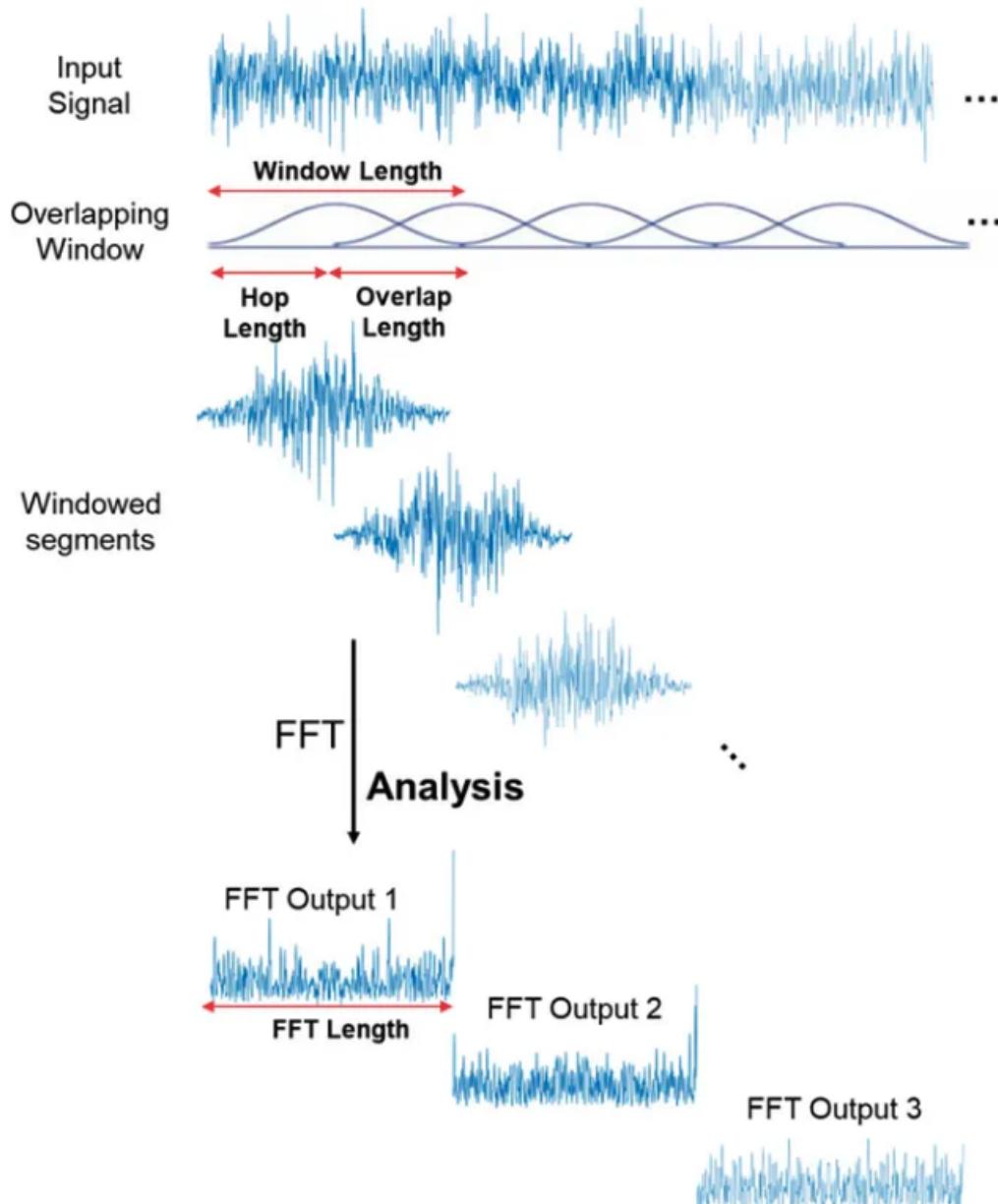
gdje je:

- $w(t)$ - funkcija prozora

- $x(t)$ - ulazni signal
- f - frekvencija
- t - vremenski pomak

Rezultat STFT-a je funkcija vremena i frekvencije koja predstavlja snagu ili amplitudu signala za svaku frekvenciju i svaki vremenski pomak. STFT se često koristi u analizi govora, obradi slike, glazbi i drugim područjima.

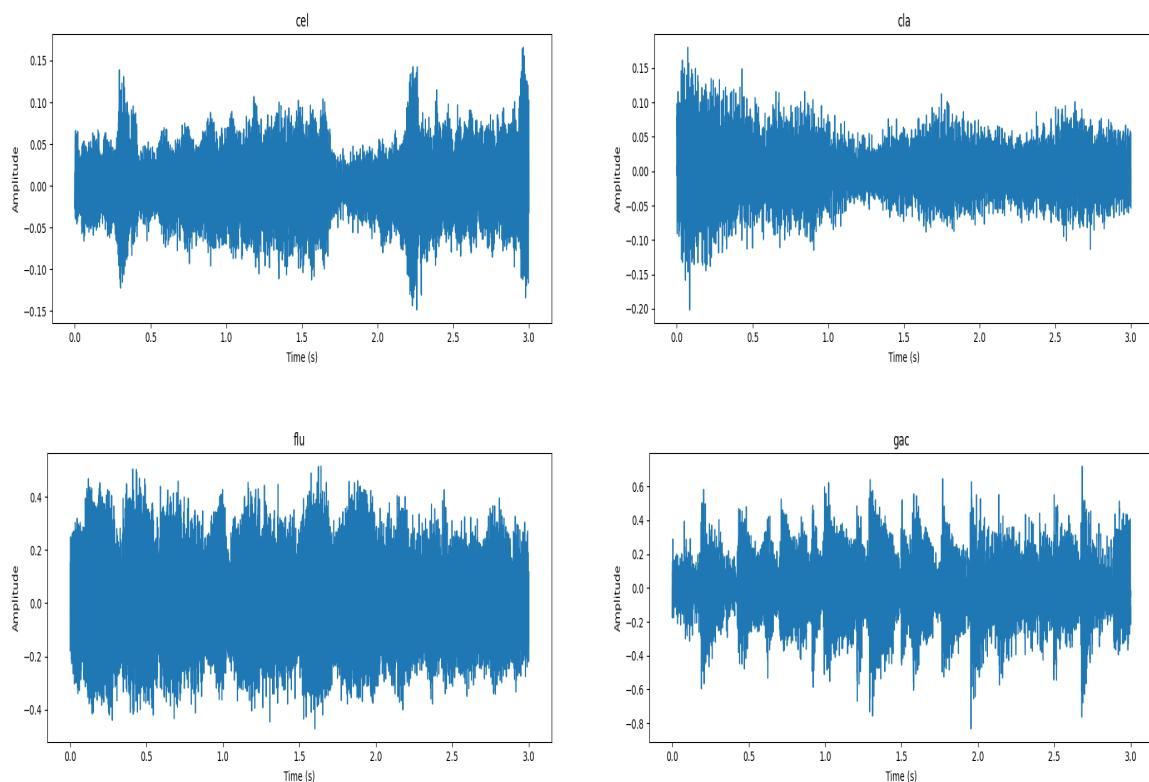
Možemo primijetiti kako će odabir parametara kao što su širina vremenskog prozora te vremenski korak biti od velikog značaja kod procesiranja danih signala.



Slika 2.6: STFT

2.3 Značajke zvuka

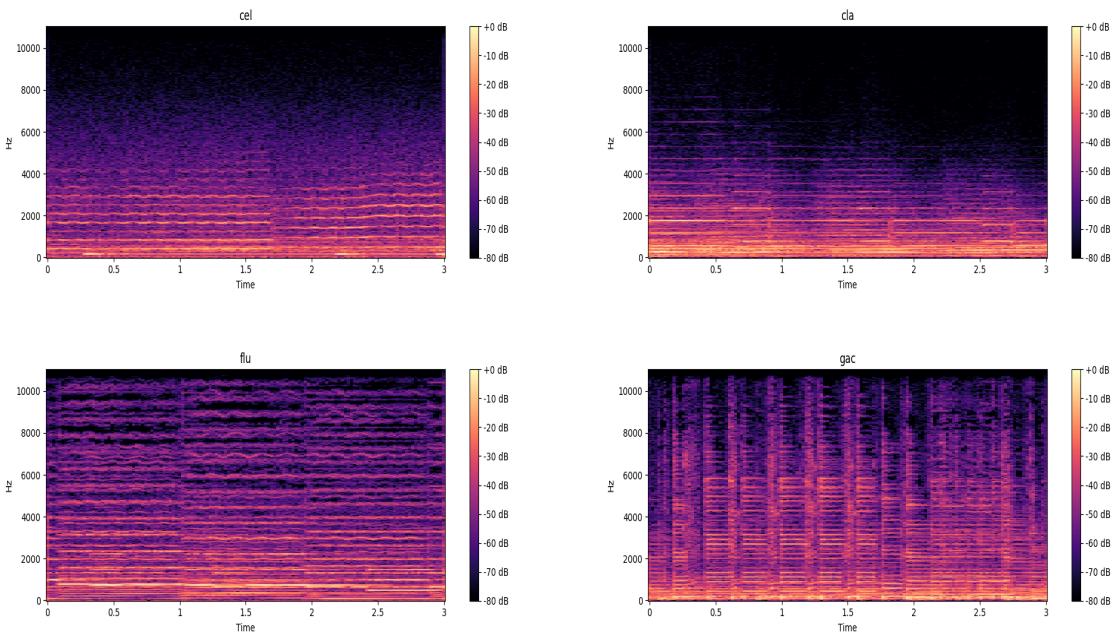
Kako bismo mogli klasificirati različite instrumente prvo moramo razumjeti koje su to karakteristike različitih zvučnih zapisa. Kod analize zvuka upravo ovaj proces izdvajanja različitih značajki zvuka od velike je važnosti. Tijekom procesa izdvajanja specifičnih značajki zvučnih signala cilj nam je smanjiti ulazni skup podataka tako da izvučemo podatke koji će nam dati što veću količinu informacija. Za svaku od sljedećih značajki zvuka koju ćemo navesti dati ćemo nekoliko primjera za različite instrumente te različite audio zapise. Kako bi jasnije i bolje primijetili razlike između određenih metoda analize zvuka koristiti ćemo ova 4 audio zapisa:



Slika 2.7: Waveforme izvornih oblika za različite instrumente

2.3.1 Spektrogram

Spektrogram možemo zamisliti kao vizualni prikaz zvuka, odnosno kao sliku zvuka. Spektar zvuka ili spektralne komponente zvuka naziv je za pojedinačne frekvencije koje čine potpuni signal. Spektrogram je način da se vizualno prikaže glasnoća ili amplituda signala koja varira tijekom vremena na različitim frekvencijama. Spektrogram iz zvučnog signala računamo tako da signal podijelimo na dijelove, zatim na svakom dijelu provodimo STFT. Potrebno je uzeti absolutnu vrijednost rezultata STFT-a kako bi se dobila magnituda signala u frekvencijskoj domeni. Kako bi dobili bolji prikaz zvuka dobivene rezultate pretvaramo u decibele (dB). Finalno, posložit ćemo rezultate svakog dijela početnog signala nad kojim smo radili navedene operacije kao i u početnom audio zapisu.

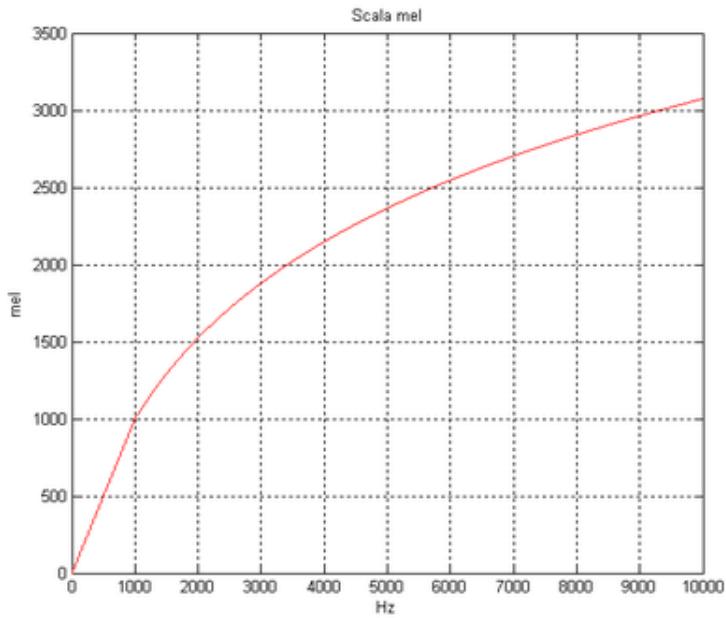


Slika 2.8: Spektrogrami različitih instrumenata

Uočavamo da svakako postoje razlike u spektrogramima između različitih instrumenata što će svakako biti korisno za naš zadatak raspoznavanje različitih instrumenata u audio zapisu. Veličina prozora koju smo koristili za dobivanje ovog prikaza je 1024 (window size), a veličina skoka (hop length) je 512.

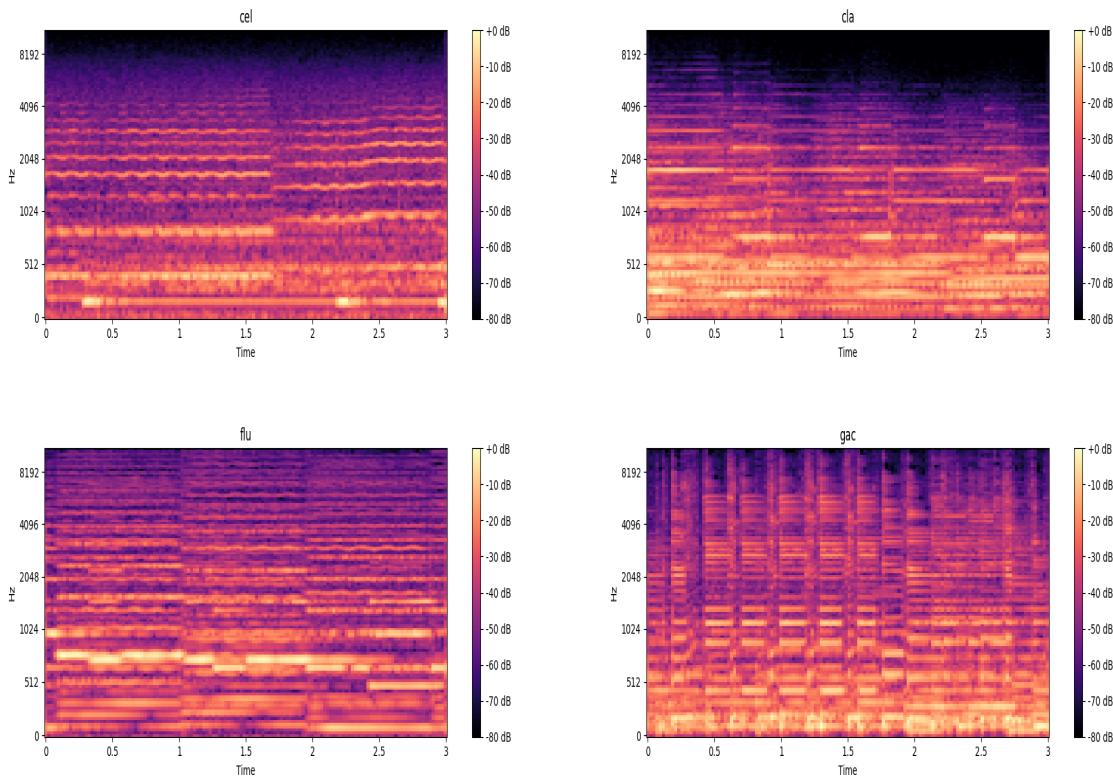
2.3.2 Mel-spektrogram

Način na koji čujemo frekvencije u zvuku poznat je kao "visina". To je subjektivni dojam učestalosti. Dakle, visok zvuk ima višu frekvenciju od niskog zvuka. Ljudi ne percipiraju frekvencije linearno te smo osjetljiviji na razlike između nižih frekvencija nego viših frekvencija. Ako kao primjer uzmem parove frekvencija od 100 i 200 Hz te frekvencija 1100 i 1200 Hz, iako je u oba slučaja razlika između danih parova 100 Hz, prvi par zvučat će "dalje" od drugog. Možemo zaključiti da mi ne čujemo zvuk na linearnoj skali, već na logaritamskoj. Upravo je to ideja mel skale. Mel skala je skala koja se koristi za opisivanje frekvencijskog spektra zvuka na način koji je više sličan percepciji ljudskog uha te se obično koristi u kombinaciji s logaritamskom transformacijom.



Slika 2.9: Mel skala i hertz(Hz) skala

Mel-spektrogram je ništa drugo, nego spektrogram gdje se frekvencije pretvaraju u vrijednosti prema mel skali. Prikazat ćemo mel-spektrograme za iste audio zapise za koje smo prikazali i spektrograme te ćemo svakako moći primijetiti da postoji razlika između ta dva prikaza. Broj mel filtera koje smo koristili za ovaj prikaz je 128 dok su svi drugi parametri ostali jednaki.



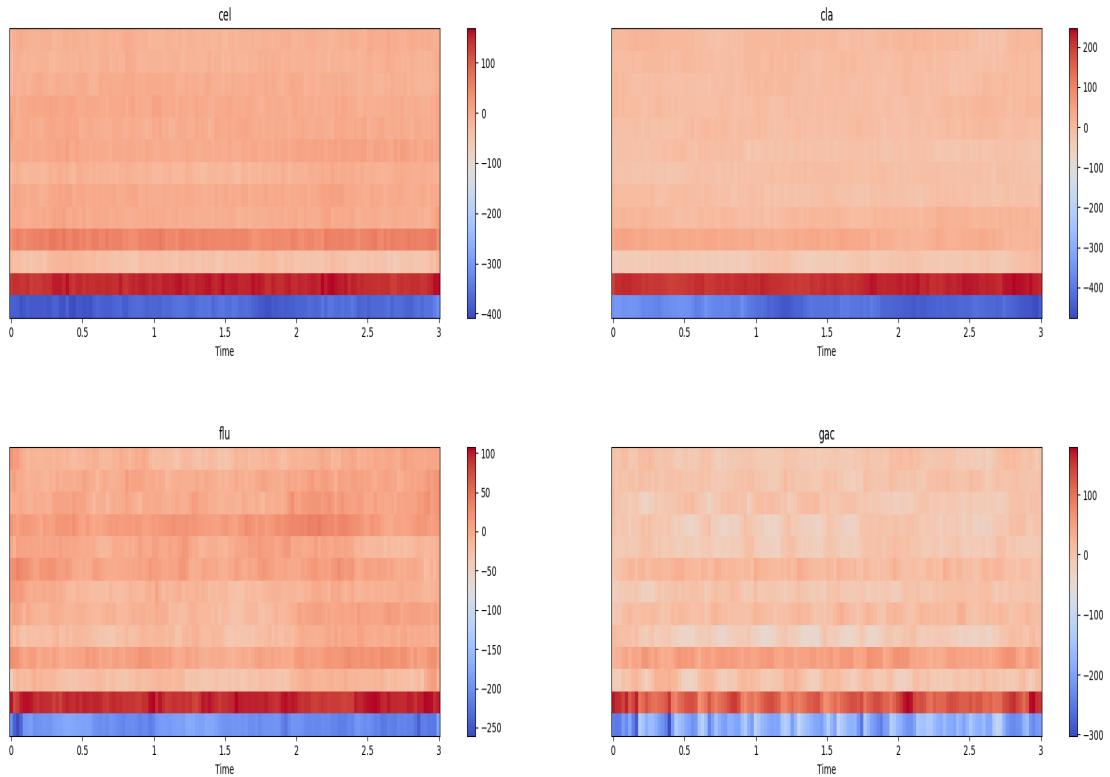
Slika 2.10: Mel-spektrogrami različitih instrumenata

2.3.3 MFCCs (*Mel frequency cepstral coefficients*)

Mel-spektrogram i MFCC-ovi usko su povezani, no ipak postoje neke bitne razlike. Ukratko, Mel-spektrogram je vremensko-frekvencijski prikaz signala, gdje se frekvencijska ljestvica prema mel skali, dok je MFCC daljnji korak obrade koji uključuje uzimanje logaritma Mel-spektrograma te se zatim nad tim podacima provodi diskretna kosinusova transformacija (*Discrete Cosine Transform*):

$$C(n) = \sum_{m=1}^M E(m) \cos\left(\frac{\pi(m-0.5)n}{M}\right), \quad n = 1, 2, \dots, p \quad (2.10)$$

gdje je p potrebnii broj MFCC koeficijenata. DCT eliminira relativnost u vektoru svojstava i minimizira kvadrat prosjeka greške. Kada je p velikog iznosa, MFCC teži u nulu.



Slika 2.11: MFCCs za različite instrumente

Prikazani MFCC-ovi odgovaraju audio zapisima koje smo prikazivali i u prethodnim primjerima. Broj koeficijenata koje smo koristili u ovom primjeru iznosi 13, a ostali parametri su ostali isti kao i kod primjera za izračunavanje mel-spektrograma.

2.3.4 Spektralni centroid

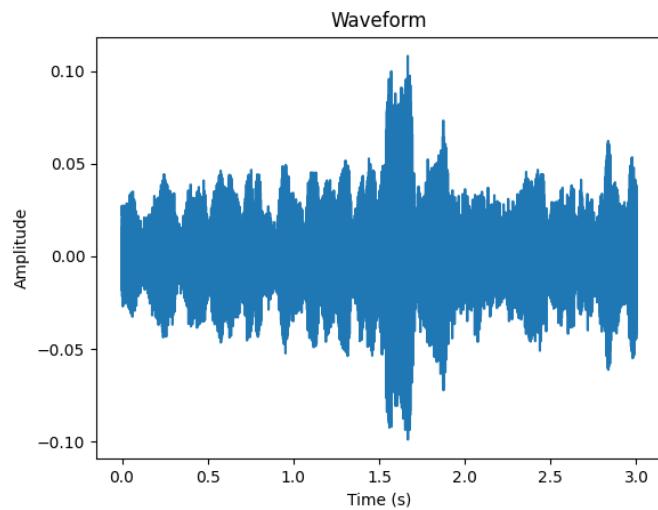
Kao što naziv sugerira, spektralni centroid je mjesto središta mase spektra. Budući da su audio datoteke digitalni signali i spektralni centroid je mjera koja može biti korisna u karakterizaciji spektra signala audio datoteke. Na nekim mjestima to se može smatrati medijanom spektra, ali postoji razlika između mjerjenja spektralnog centroida i medijana spektra. Spektralni centroid je poput ponderiranog medijana, a medijan spektra je sličan srednjoj vrijednosti. Oba mjere središnju tendenciju signala. U nekim slučajevima oba pokazuju slične rezultate. Značajka ovog svojstva je da više vrijednosti odgovaraju višim frekvencijama zvuka.

Spektralni centroid računamo na sljedeći način:

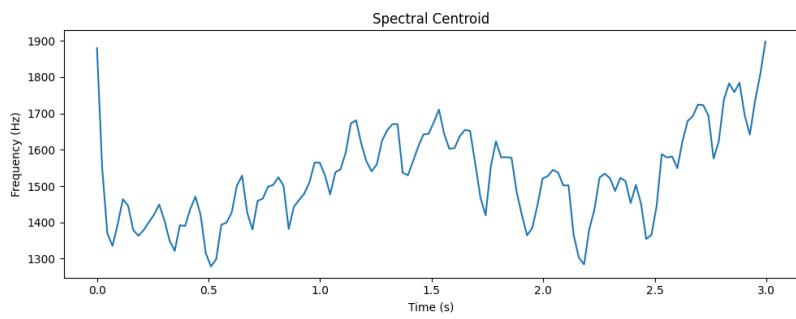
$$\text{Centroid} = \frac{\sum_{n=0}^{N-1} f(n)x(n)}{\sum_{n=0}^{N-1} x(n)} \quad (2.11)$$

gdje je:

- n - broj spremnika (*bins*)
- $x(n)$ - težinska vrijednost frekvencije
- $f(n)$ - središnja frekvencija spremnika



(a) Waveform (violončelo)



(b) Spektralni centroid (violončelo)

2.3.5 Stopa prelaska nule (*Zero-Crossing Rate*)

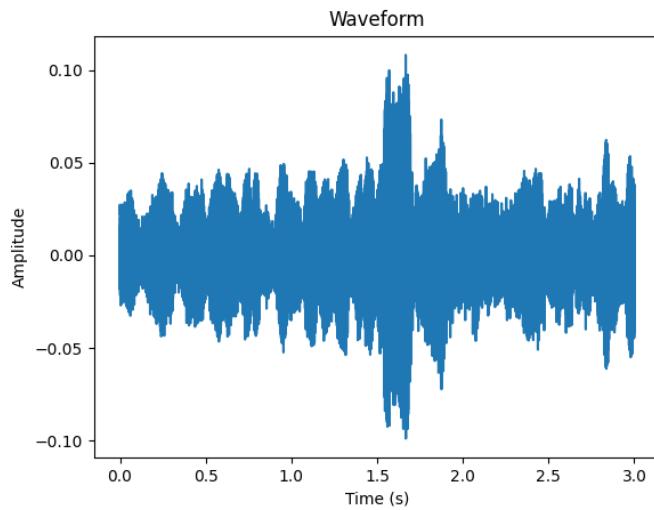
Stopa prijelaza nule u signalu, od određenog okvira u audio signalu, je mjera promjene predznaka vrijednosti signala tijekom trajanja tog okvira signala. Drugim riječima, to je broj prijelaza vrijednosti signala iz pozitivne u negativnu vrijednost ili negativne u pozitivnu, podijeljen s duljinom okvira. ZCR se može protumačiti kao mjera šuma u signalu. Npr. u slučaju povećanja šuma raste i vrijednost signala. ZCR se računa prema formuli:

$$zcr = \frac{1}{N-1} \sum_{n=1}^{N-1} |\operatorname{sgn}(x(n)) - \operatorname{sgn}(x(n-1))| \quad (2.12)$$

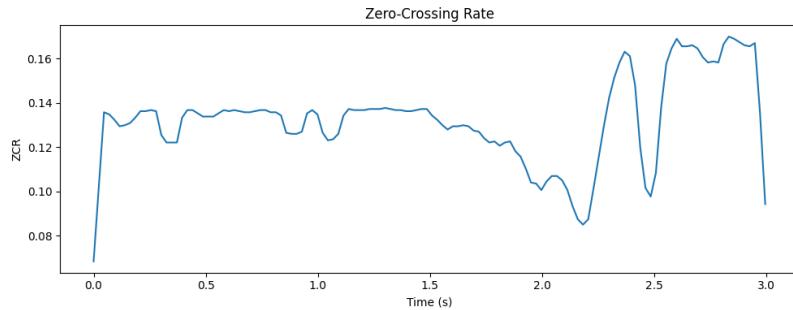
gdje je:

- N : duljina signala (okvira)
- $x(n)$: audio signal u vremenu n

$$\bullet \operatorname{sgn}(x_i(n)) = \begin{cases} 1 & x_i(n) \geq 0 \\ -1 & x_i(n) < 0 \end{cases}$$



(a) Waveform (violončelo)



(b) Stopa prelaska nule (violončelo)

2.3.6 Ostale značajke zvuka

Prethodno smo prošli kroz nama najbitnije i najznačajnije značajke zvuka. Osim navedenih, postoji još nekoliko važnih značajki zvuka koje nismo nabrojali jer ih nismo koristili u izradi rješenja.

Neke od njih su:

- Chromagram
- Spektralni kontrast (Spectral Contrast)
- Spektralna širina pojasa (Spectral Bandwidth)
- Spektralno opadanje (Spectral RollOff)

3. Pretpresiranje

Pretpresiranje podataka je jedan od ključnih koraka u procesu analize podataka. U ovom koraku, obrađujemo sirove podatke kako bismo ih pripremili za daljnju analizu. Cilj pretpresiranja je očistiti podatke od pogrešaka, nedostajućih vrijednosti i nepotrebnih informacija te ih transformirati u oblik koji je pogodan za daljnju analizu te kasnije za treniranje odabranog modela. Pretpresiranje je jedan od ključnih procesa tijekom analize podataka i stvaranja pogodnog modela za kasnije treniranje nad tim podacima.

Svaki od navedenih koraka je važan za osiguravanje kvalitete podataka i stvaranje preciznih i pouzdanih rezultata analize. U prethodnom poglavlju nabrojili smo te objasnili neke od najvažnijih značajki zvuka te ćemo sada izdvojiti one koje su nama bile najkorisnije u zadatku klasifikacije instrumenata u danom audio zapisu.

U ovom poglavlju, detaljno ćemo opisati korake i metode koje smo koristili u našem projektu. Također ćemo razmotriti i prokomentirati neke od izazova s kojima smo se susreli tijekom izrade rješenja.

3.1 Priprema podatkovnog skupa

Prvi korak u pretpresiranju podatkovnog skupa je učitavanje danih podataka. Tijekom učitavanja audio zapisu ponovno ga uzorkujemo na 22 050 Hz s originalnih 44 100 Hz, razlog tome je što time nećemo izgubiti puno informacije iz zadalog zapisa, a smanjiti ćemo samo vrijeme procesiranja. Zbog istog razloga ćemo stereo zvuk pretvoriti u mono.

Zatim smo sve datoteke u podatkovnom skupu za trening podijelili na 3 dijela po 1 sekundi. U slučaju validacijskog podatkovnog skupa koristili smo također dijelove datoteke od duljine 1 sekundu. Ako datoteka nije mogla biti podijeljena na jednake dijelove duljine 1 sekunde, koristili smo tehniku dodavanja nula na kraj zadnjeg dijela kako bi uspostavili željenu duljinu isječka (*padding*). Kroz postupak unaprjeđivanja rezultata isprobali smo razne duljine kao što su još i 3, 2, 1.5, 0.5 sekundi te nam je upravo navedena podjela audio zapisa na dijelove od 1 sekunde dala najbolje rezultate. Nakon što podijelimo audio zapise na dijelove krećemo u izvlačenje željenih značajki iz istih tih dijelova.

3.2 Više različitih značajki zvuka

Prva metoda kojoj smo pristupili bila je izvlačenje sljedećih značajki zvuka:

- MFCC
- Spekralni centroid
- Stopa prelaska nule

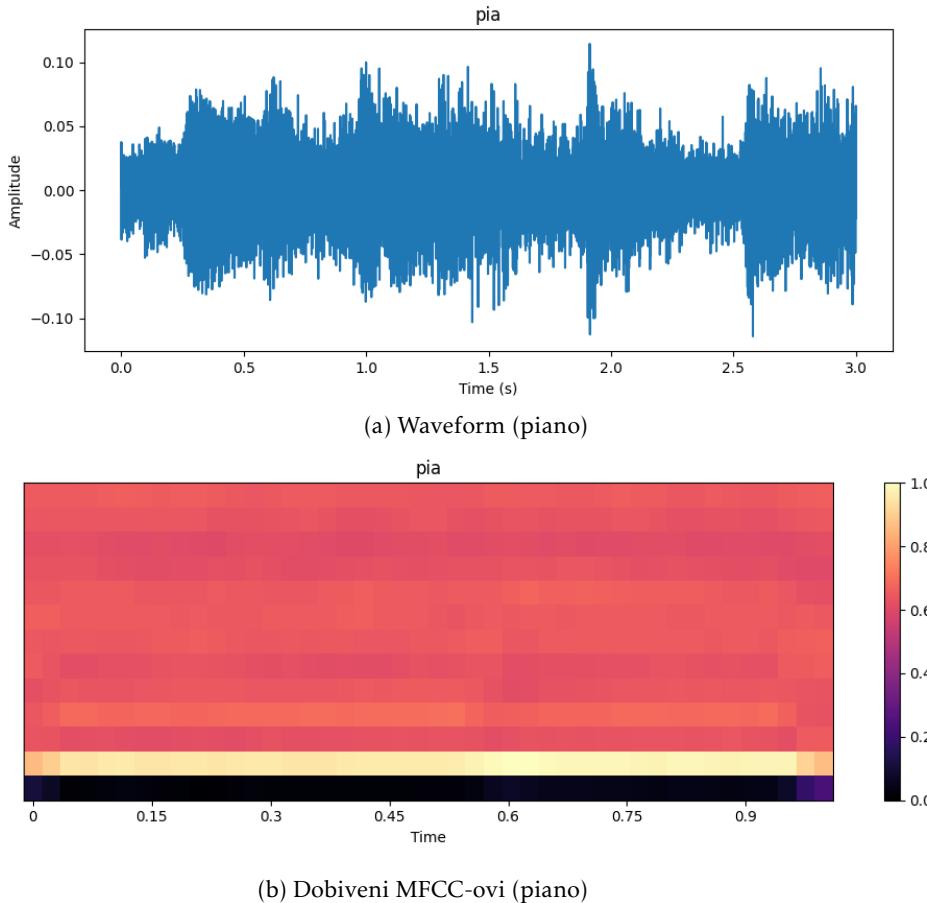
Zatim smo te značajke koristili u kombinaciji za treniranje modela.

3.2.1 MFCC

Kao što smo ranije spomenuli kako bi dobili MFCC-ove prvo provodimo STFT nad zadanim signalom. Parametri koje je trebalo podesiti bili su veličina prozora (*window size*) i veličina skoka (*hop length*). I ovdje je bilo potrebno isprobati različite kombinacije kako bi se dobili najbolji rezultati za naš specifičan zadatak. Nakon ispitivanja različitih kombinacija odlučili smo koristiti veličinu prozora od 1024 te veličinu skoka od 512. Ostale parametre STFT-a ostavili smo po default-u iz python paketa librosa.

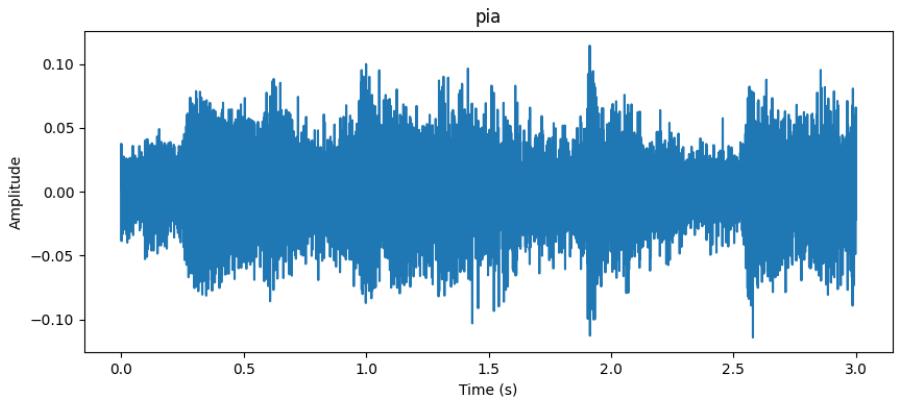
Nakon provođenja STFT-a nad audio zapisom računamo MFCC-ove. Broj koeficijenata koje smo računali postavili smo na 13, a broj mel band-ova postavili smo na 128, ostale vrijednosti ostavili smo kao što su default zadane u librosi. Odabir broja mel band-ova ovisi o karakteristikama audio signala i specifičnom zadatku. Povećanje vrijednosti može rezultirati prikazom zvučnog spektra u višoj razlučivosti, ali također može uvesti veću računsку složenost. Dobivene MFCC-ove zatim smo normalizirali tako što smo svaki MFCC iz liste oduzeli s minimalne vrijednosti iz tog skupa, te bi zatim te vrijednosti podijelili s razlikom između maksimalne i minimalne vrijednosti MFCC-ova. Time smo dobili novi niz gdje je maksimalna vrijednost 1, minimalna 0, a sve ostale vrijednosti su između 0 i 1. Svrha normalizacije je skalirati podatke na zajednički raspon kako bi se izbjegla bilo kakva pristranost prema većim ili manjim vrijednostima.

Nakon navedenih postupaka dobit ćemo sljedeće:

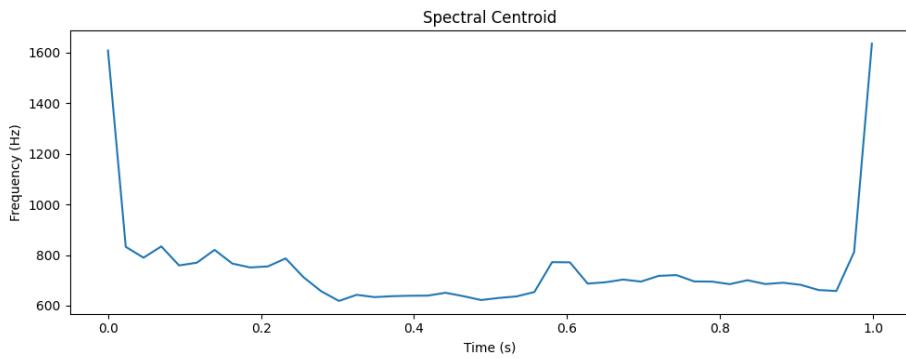


3.2.2 Spektralni centroid

Kao što smo napomenuli ranije, cilj ovog pristupa preprocesiranju podataka je kombinacija više različitih značajki zvuka što znači da je sljedeći korak dobivanje spektralnog centroida iz svakog isječka. Parametri STFT-a bili su isto kao i slučaju računanja MFCC-ova, odnosno veličina prozora od 1024 te veličina skoka od 512, ostali parametri bili su default-ni parametri python paketa librosa-e. Dobivene vrijednosti spektralnih centroida zatim smo normalizirali kao i MFCC vrijednosti te smo dobili sljedeće rezultate:



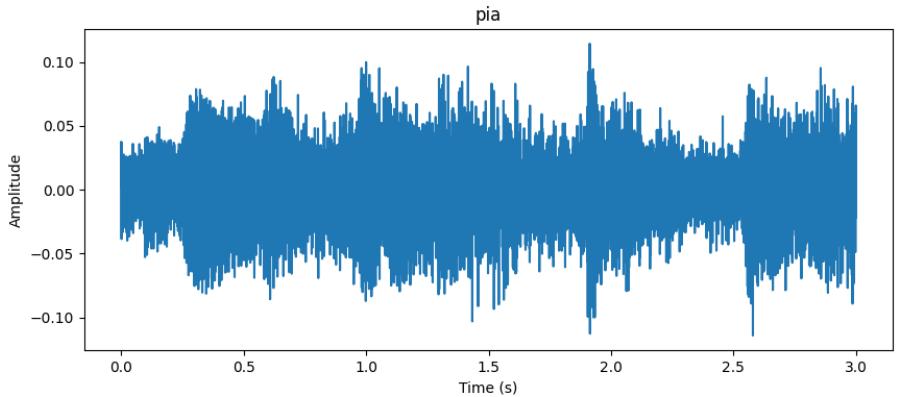
(a) Waveform (piano)



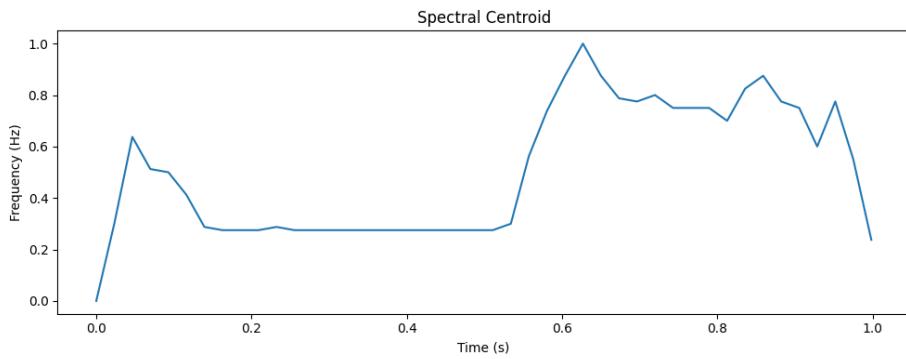
(b) Dobiveni spektralni centroid (piano)

3.2.3 Stopa prelaska nule

U kombinaciju MFCC-ova i spektralnog centroma još smo nadodali i stopu prelaska preko nule. Parametre kod računanja kod ovog pristupa ostavili smo po default-u python paketa librosa-e. Ove rezultate smo također zatim normalizirali kao i u prethodnim primjerima kako bi svakoj dobivenoj značajki pridijelili jednak značenje. Ovo je primjer rezultat koje smo dobili:



(a) Waveform (piano)

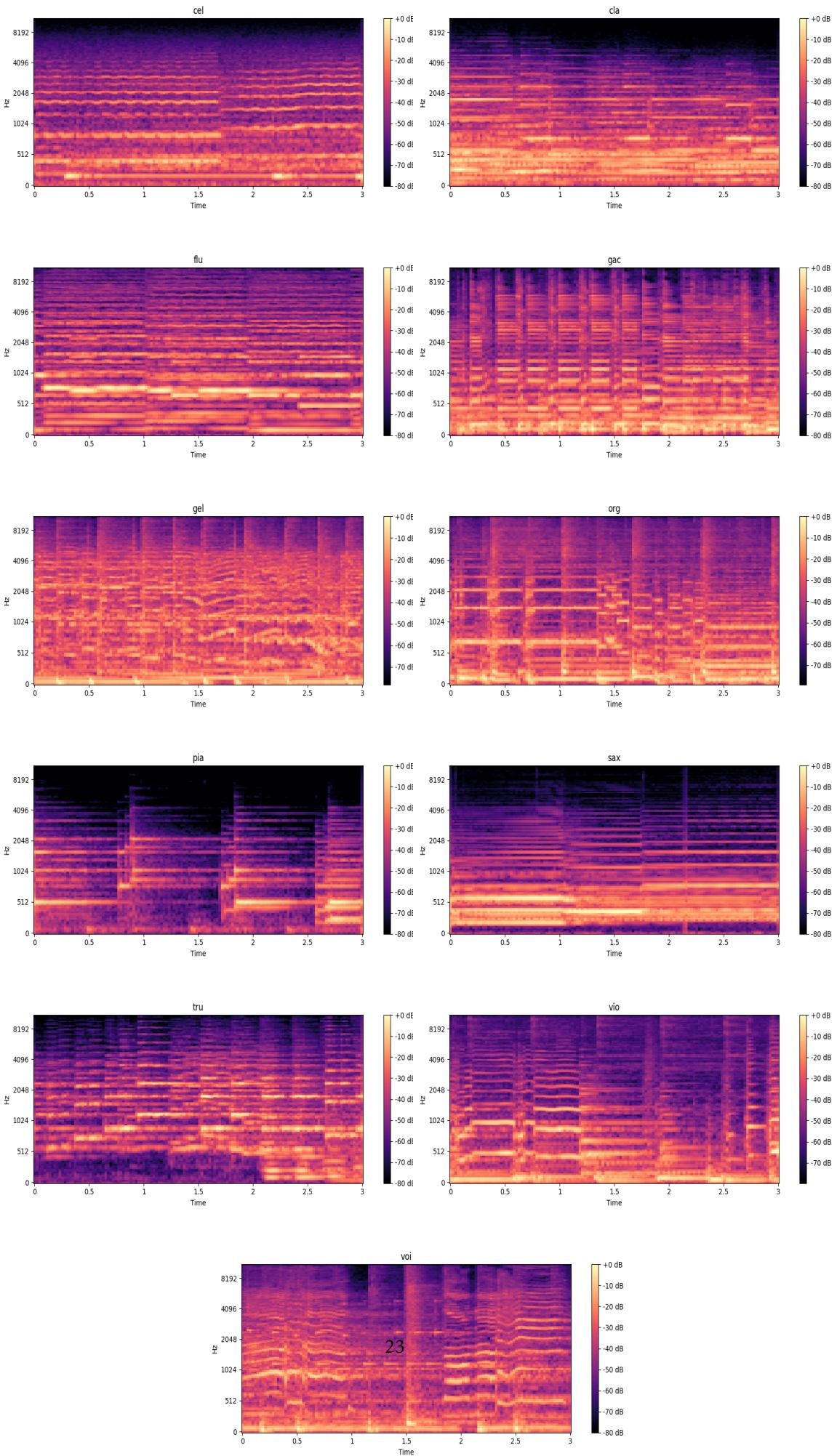


(b) Dobiveni ZCR (piano)

Na kraju smo sve dobivene rezultate kombinirali te smo na tome trenirali modele.

3.3 Mel-spektrogram

Drugi pristup pretprocesiranju podataka bio je izračun mel-spektrograma nad isjećima audio zapisa. I kod ovog pristupa bilo je potrebno podesiti parametre za STFT. Nakon isprobavanja raznih kombinacija, držali smo za veličinu prozora od 1024, veličinu skoka od 512 te broj mel-ova 128. S tim parametrima računali smo mel-spektrogram te bi dobivene rezultate skaliramo u decibele(dB) kako bi dobili bolji prikaz zvuka. Dobili smo sljedeće prikaze za zadanih 11 instrumenata:



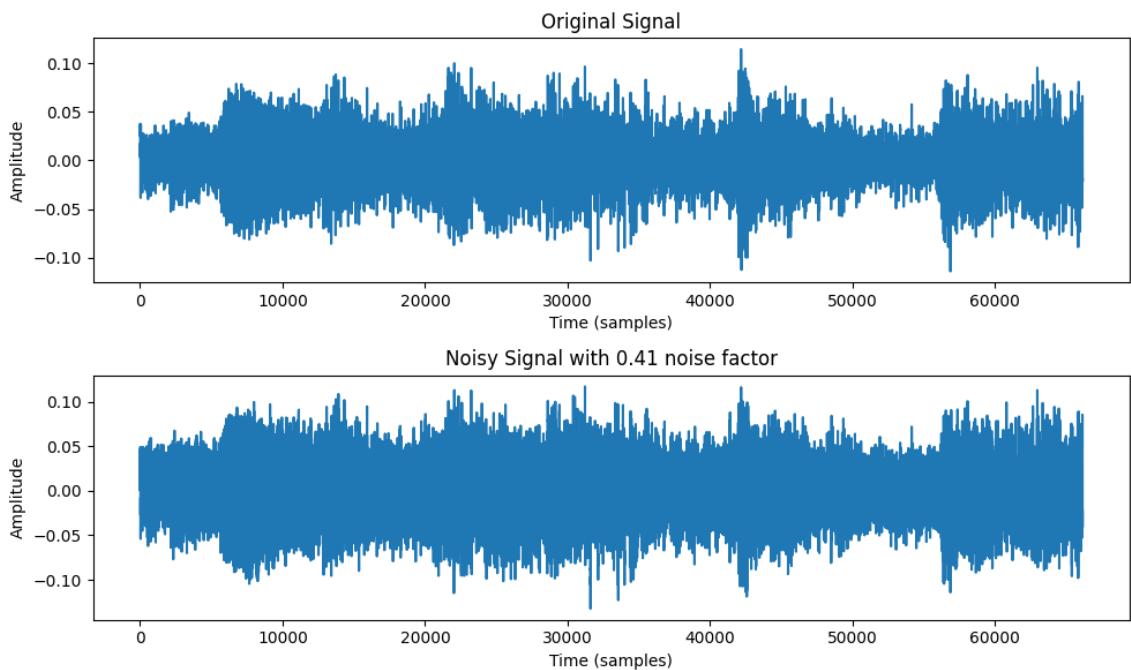
3.4 Augmentacija

Augmentacija podataka podrazumijeva umjetno generiranje novih primjera podataka iz postojećeg skupa podataka, primjenom različitih transformacija. Nedostatak podataka čest je problem u području znanosti o podacima te upravo pomoću augmentacije pokušavamo smanjiti taj problem. Za generiranje novih podataka, možemo primijeniti dodavanje buke, promjenu vremena i promjenu tona. Kako bi izbjegli overfitting ne koristimo uvijek iste parametre, već ih na neki način "slučajno" odabiremo.

3.4.1 Dodavanje šuma

Kako bi dodali šum zvuku jednostavno ćemo našem početnom signalu dodati neke vrijednosti. Kod dodavanja šuma potrebno je pripaziti na količinu dodanog šuma. Dodavanje previše šuma u početni signal moglo bi rezultirati narušivanjem kvalitete audio signala, što može otežati prepoznavanje značajki koje model koristi za klasifikaciju.

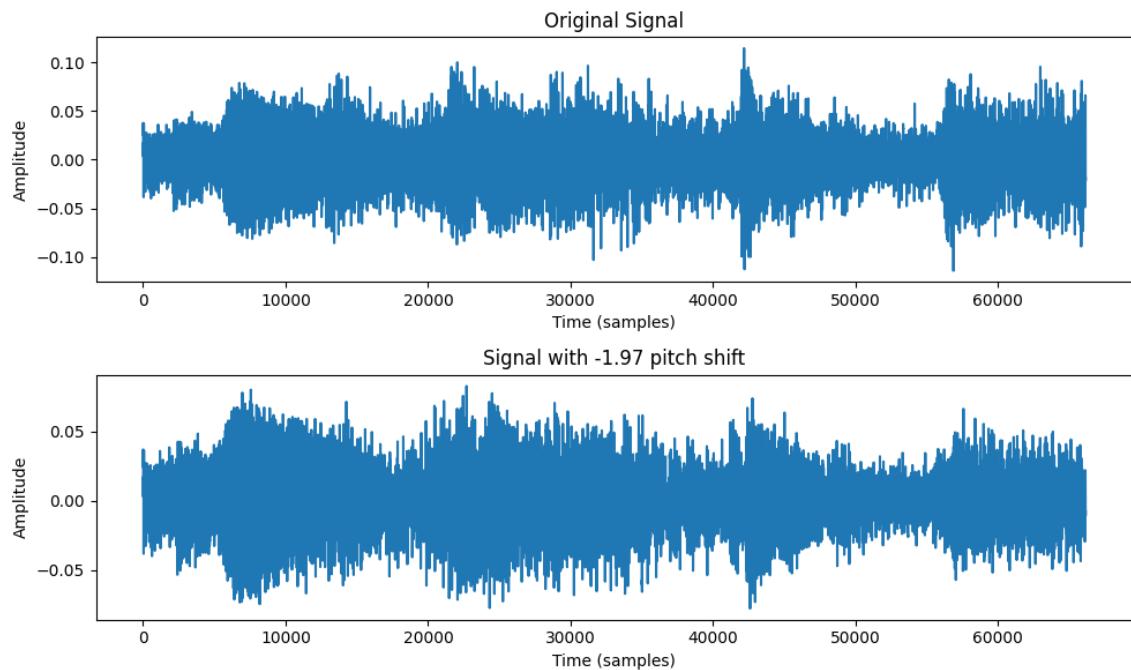
Šum smo dodavali tako da smo na početni signal dodali slučajnu vrijednost iz normalne distribucije sa medijanom 0 te standardnom devijacijom 1 pomnoženom sa faktorom šuma. Kao faktor šuma bi izabrali vrijednost iz uniformne distribucije između npr. 0.01 i 0.02. U ovom primjeru staviti ćemo veću količinu šuma radi lakšeg razlikovanja između originalnog signala i onog sa šumom.



Slika 3.4: Augmentacija dodavanjem šuma

3.4.2 Promjena tona(*pitch shifting*)

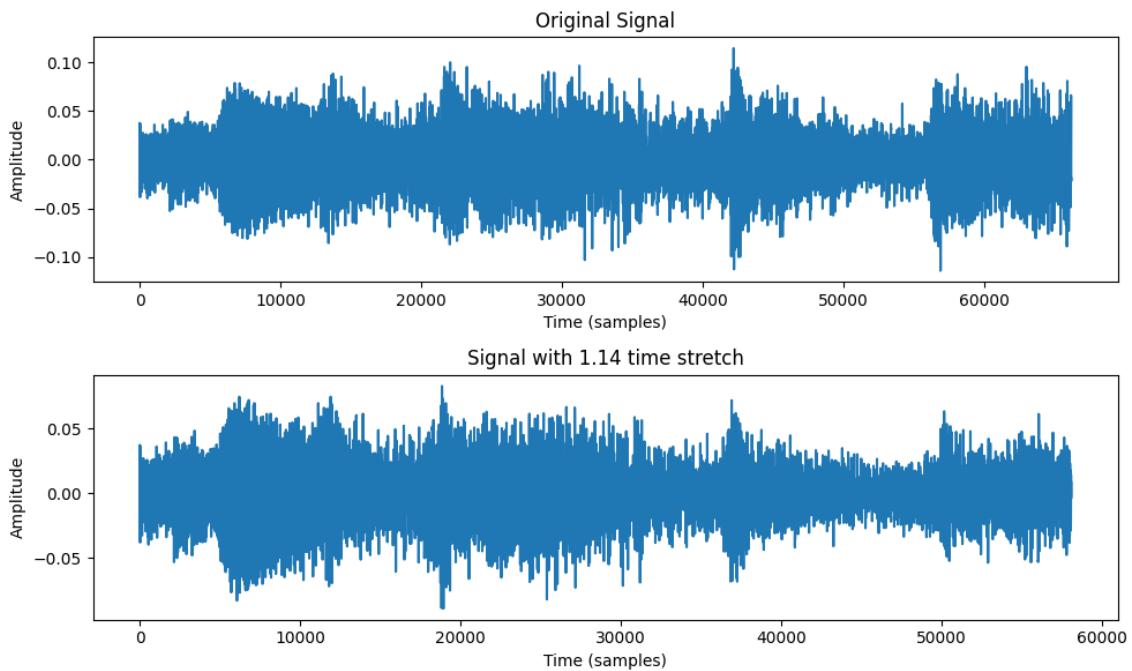
Promjena tona je tehnika u obradi zvuka koja se koristi za promjenu frekvencije tona, a time i njegove visine. Promjeni tona radili smo pomoću python paketa librosa-e. Kao broj polotonova za koje želimo pomaknuti tonalitet zvučnog zapisa koristimo broj iz uniformne distribucije između npr. -3 i 3. Ukoliko vrijednost nije tipa integer, librosa će tu vrijednost pridružiti najbljižem intergeru. Ukoliko je dobivena vrijednost negativna dogodit će se spuštanje tona, a u suprotnom ton će se podići.



Slika 3.5: Augmentacija promjenom tona

3.4.3 Promjena vremena(*time stretching*)

Promjena vremena je tehnika obrade zvuka koja mijenja trajanje audio zapisa bez utjecaja na visinu zvuka. Važno je napomenuti da promjena vremena može dovesti do gubitka kvalitete zvuka, posebno ako se signal ubrzava ili usporava za veliki faktor. Također, ako se primjenjuje na glasovne signale, promjena vremena može utjecati na razumljivost govora. Stoga je važno pažljivo odabrati faktor skaliranja kako bi se izbjegao gubitak kvalitete i razumljivosti signala. Ovu vrstu augmentacije također smo provodili pomoću python paketa librosa. Vrijednost pomoću koje ćemo provoditi promjenu vremena bit će vrijednost dobivena iz uniforme distribucije između npr. 0.8 i 1.2. Ukoliko je vrijednost 0.8 to znači da će se audio zapis usporiti za 20%, a ukoliko je dobivena vrijednost od 1.2 to znači da će audio zapis biti ubrzan za 20%.



Slika 3.6: Augmentacija promjenom vremena

3.4.4 Miješanje audio zapisa različitih instrumenata

Iako ovu metodu augmentacije na kraju nismo koristili, svejedno smo je dobro razmotrili. Iz razloga što je zadatak modela da prepozna više različitih instrumenata unutar istog audio zapisa, ideja iza ove vrste augmentacije je da u podatkovni skup za trening modela dodamo i audio zapise koji sadrže više od jednog instrumenta. Izazov ovakvog pristupa augmentaciji je upravo odabir načina na koji ćemo mijesati audio zapise. Ukoliko će naš model biti korišten za validaciju nad glazbom, mogao bi biti krivi pristup samo spajanje audio zapisa jer u glazbi svaki instrument ima svoju ulogu te oni zajedno čine jednu uređenu cjelinu. Ukoliko je i slučaj da će model raditi klasifikacija na temelju audio zapisa u kojem instrumenti ne poštuju nikakav red, zbog velikog broja instrumenata i trening podataka bilo bi vremenski i prostorno zahtjevno odraditi takav oblik augmentacije.

3.5 Konačan odabir pristupa

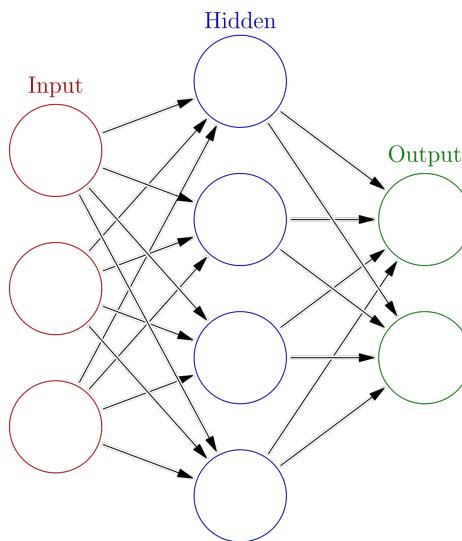
Nakon isprobavanja različitih kombinacija i vrsta značajki odlučili smo se za pristup sa navedenim mel-spektrogramom. Također, odlučili smo provesti augmentaciju nad danim podatkovnim skupom za trening modela čime smo osigurali dodatne podatke za trening modela, što je kasnije i rezultiralo pospješenju rezultata. Isprobali smo razne vrijednosti augmentacija te njihove kombinacije kako bi dobili najbolje rezultate. Bilo je važno odabrati vrijednosti augmentacije koje neće previše promijeniti početni audio signal, no isto tako ni one koje neće imati nikakav utjecaj na isti. Najbolje rezultate dobili smo primjenom augmentacije promjene vremena te promjene tona. Vrijednosti za promjenu tona dobili smo iz uniformne razdiobe između -3 i 3, a vrijednosti za promjenu vremena između 0.8 i 1.2. U našem slučaju, dodavanje šuma nije pozitivno utjecalo na rezultate modela.

4. Model

Izradi modela pristupili smo inkrementalno, isprobavajući nove stvari dok nismo dobili zadovoljavajuće rezultate. Prije nego što smo krenuli u izradu modela, istražili smo koji način je state-of-the-art za klasifikaciju nad audio skupom podataka. Zaključili smo da je najbolje krenuti u smjeru modela koji se inače primjenjuju u području računalnog vida, a to je konvolucijska neuronska mreža.

4.1 Neuronske mreže

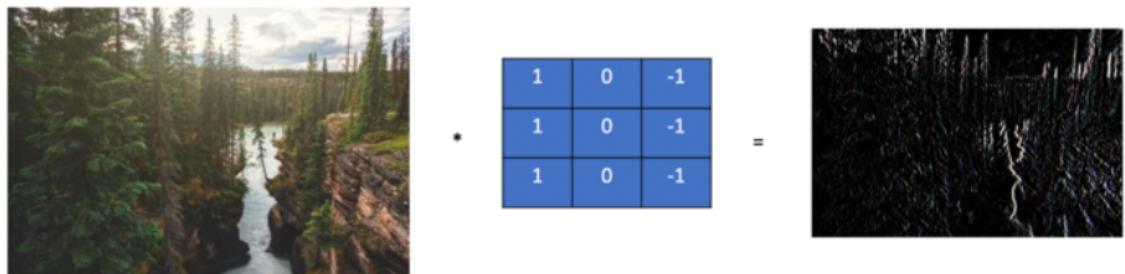
Neuronska mreža (neural network, skraćeno NN) je složeni matematički model koji simulira rad biološke neuronske mreže tj. mozga i pokušava učiti putem primjera. To je vrsta strojnog učenja koja se sastoji od umjetnih neurona povezanih u slojevima koji zajedno izvode računske operacije kako bi se došlo do zaključaka. Slojevi neuronske mreže klasificiraju se u 3 vrste, ulazni sloj (input layer), skriveni sloj (hidden layer) i izlazni sloj (output layer). Skrivenih slojeva može biti više te ih najčešće i ima više. Svaki neuron prima ulazne podatke i izvodi matematičke operacije na tim podacima, čime se generira izlazni podatak koji se šalje sljedećem sloju neurona. Tako se informacije obrađuju i prenose kroz više slojeva neurona, sve dok se ne dobije konačni izlaz. Na izlazu iz slojeva se na vrijednosti koje su dali neuroni primjenjuje neka aktivacijska funkcija (neki primjeri su sigmoidalna funkcija, RELU, tangens hiperbolni, softmax itd.) kako bi se u mrežu uvela nelinearnost.



Slika 4.1: Primjer male neuronske mreže

4.2 Konvolucijske neuronske mreže

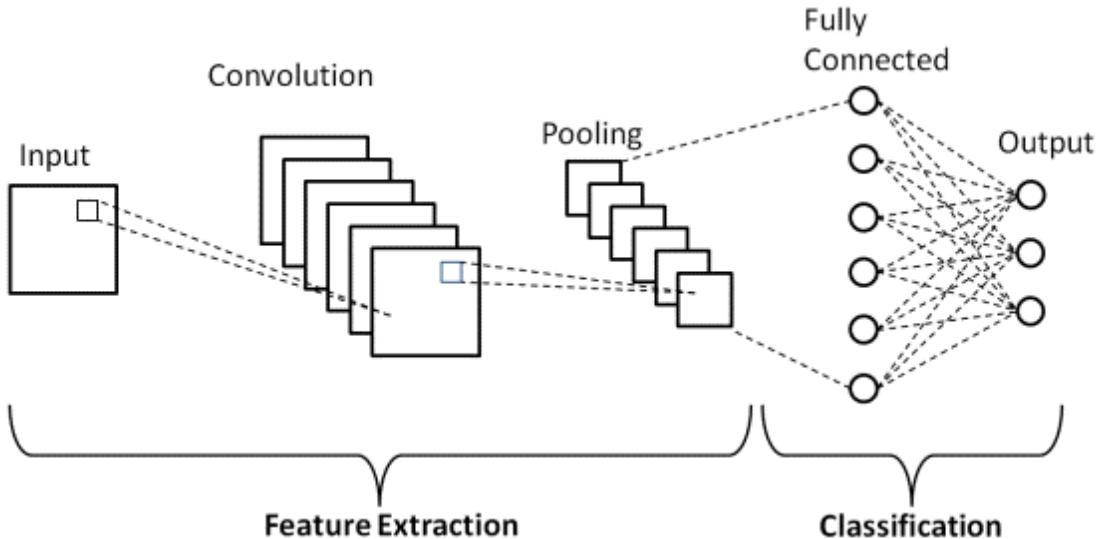
Konvolucijska neuronska mreža (CNN) je popularna vrsta neuronskih mreža koja se često koristi u analizi slike, prepoznavanju uzoraka i obradi prirodnog jezika. Temelji se na ideji da se učenje parametara modela provodi lokalno na manjim dijelovima slike, što mu omogućuje prepoznavanje uzoraka na različitim mjestima u slici. CNN koristi konvolucijske slojeve za filtriranje ulazne slike kroz niz filtera koji se primjenjuju na manje dijelove slike kako bi se izdvojile važne značajke. Primjer filtera i njihove upotrebe na slici možemo vidjeti na slici 4.2, gdje je filter jedan 3×3 kernel za vertikalni edge detection.



Slika 4.2: Primjer konvolucije s edge-detection kernelom

Sloj konvolucije se sastoji od više kanala, a unutar svakog kanala se primjenjuje filter (neki kernel). Cilj različitih filtera je izvući različite značajke (features) iz podataka. Zasluge za to da se već prije napomenuto dobro shvaćanje da se značajke može pojaviti bilo gdje u prostoru slike možemo pridijeliti tome da se filteri na jednak način primjenjuju na svaki piksel u slici. Nakon toga slijedi sloj za uzorkovanje (pooling) koji smanjuje dimenzije izdvojenih značajki. Ova dva sloja se obično ponavljaju nekoliko puta u mreži kako bi se postigla sve viša razina apstrakcije. Nakon konvolucijskih slojeva i slojeva za uzorkovanje, slijedi potpuno povezani (fully connected) sloj koji povezuje sve dobivene značajke u jedan vektor karakteristika. Nakon što se dobije vektor karakteristika iz potpuno povezanog sloja slijedi izlazni sloj koji koristi aktivacijsku funkciju kako bi se dobio konačni izlaz u obliku vjerojatnosti za svaku klasu kojoj ulazni podatci mogu pripadati.

Parametri koje trening mreže pokušava odrediti kako bi performanse modela bile što bolje uključuju vrijednosti u filterima (što zapravo određuje koje značajke se određuju za ulazne podatke), kao i vrijednosti poveznica u potpuno povezanim slojevima koje pridaju težine navedenim značajkama.



Slika 4.3: Prikaz generalne arhitekture konvolucijske neuronske mreže

Uz navedene slojeve, kako bi se spriječilo prenaučavanje (overfitting) mreže, često se koriste i različiti slojevi za normalizaciju i regularizaciju. Na primjer, sloj normalizacije po grupama (batch normalization) se može primijeniti kako bi se normalizirale izlazne vrijednosti svakog sloja prije nego što se prosljede u sljedeći sloj. Također, slojevi za regularizaciju poput dropout sloja mogu se primijeniti kako bi se spriječilo prenaučavanje mreže.

4.3 Tijek razvoja modela

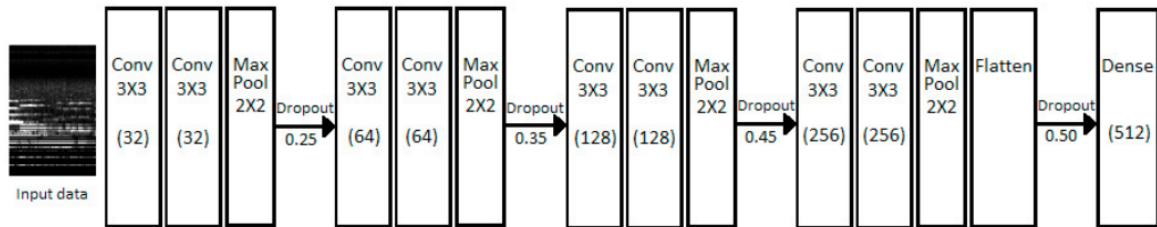
Budući da do samog kraja natjecanja nije bilo objavljeno po kojoj metriči će se evaluirati modeli, mi smo kao baseline metriku koristili micro F1-score koji daje jednaku važnost svakom instrumentu i uobičajena je metrika za slične modele.

$$F_1 = 2 \frac{precision * recall}{precision + recall} \quad (4.1)$$

Također smo nakon nekog vremena odlučili i pratiti kako se modelima kreće postotak potpuno točnih predikcija (exact matches percentage).

4.3.1 Početni modeli

Početni modeli koje smo isprobali bili su naši vlastiti modeli te neki inspirirani objavljenim rado-vima koji su se bavili sličnim izazovima, ponekad i na istom skupu podataka. Primjer jedne od početnih arhitektura je prikazan na slici 4.4. Na kraju svih modela je 11 konačnih neurona nad kojim smo primijenili sigmoid za aktivaciju i koji daju konačnu vjerojatnost da je neki od instrumenata prepoznat.



Slika 4.4: Primjer prvih mreža

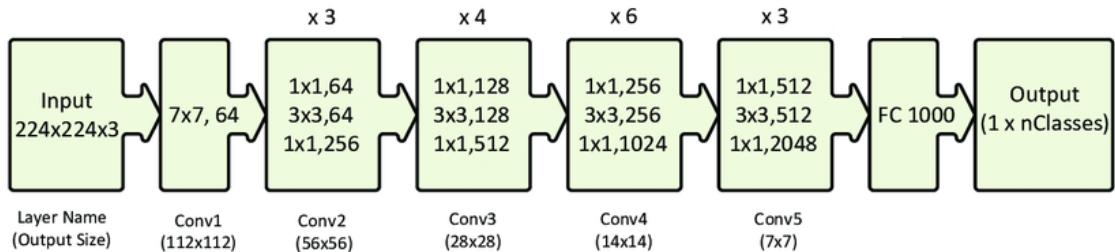
Budući da rezultati nisu bili zadovoljavajući, a nismo imali načina doći do još podataka za training modela, odlučili smo dodati augmentirati podatke te ih dodati u training dataset. Primijenjene augmentacije bile su pitch-shifting, time-stretching i dodavanje normalnog šuma. Suprotno očekivanjima, rezultati su bili lošiji nego s modelima koji nisu imali pristup augmentiranim podacima. Nakon puno isprobanih modela zaključili smo da bi bilo dobro pokušati promijeniti pristup. Odlučili smo koristiti već gotove, state-of-the-art arhitekture za klasifikaciju slika. Počeli smo s ResNet (slika 4.5) arhitekturom, i postigli smo poboljšanja, ali kako je to već stara arhitektura nastavili smo testiranja. Sljedeći izbori su nam bili DenseNet (slika 4.7) i ConvNeXt (slika 4.6) kao jednu od najnovijih arhitektura, ali rezultati su bili usporedivi DenseNetu koji je manji pa smo konačno odabrali DenseNet. Iako nam na početku augmentacija podataka nije pomogla već odmogla, ipak smo ju testirali i s DenseNetom te smo postigli značajno poboljšanje (koristili smo pitch-shifting, time-stretching i dodavanje normalnog šuma).

Točne performanse konačnog modela prikazane su u poglavljju *Rezultati*.

4.3.2 ResNet

ResNet je revolucionarna arhitektura iz 2015. ResNet je bio revolucionaran zato što je uveo koncept preskočenih veza (skip connections), koji je riješio problem degradacije performansi u veoma dubokim neuronskim mrežama. Prije ResNeta, uobičajena praksa u projektiranju dubokih neuronskih mreža bila je da se jednostavno stapaju sloj na sloj kako bi se povećao kapacitet modela. Međutim, kako bi se broj slojeva povećavao, preciznost modela bi često počela opadati. Ovaj fenomen se naziva problem degradacije, i bio je glavna prepreka u razvoju veoma dubokih neuronskih mreža.

Preskočene veze ResNeta su riješile ovaj problem omogućavajući informacijama da prolaze direktno između nesusjednih slojeva. To je omogućilo da se mreža uči ne samo na lokalnoj razini, već i na globalnoj razini. Tako, ResNet je mogao da obučava veoma duboke neuronske mreže bez gubitka performansi i postigne najbolje rezultate na nekoliko benchmark zadataka, kao što su klasifikacija slika, detekcija objekata i segmentacija.



Slika 4.5: Primjer ResNet50 mreže slične našoj

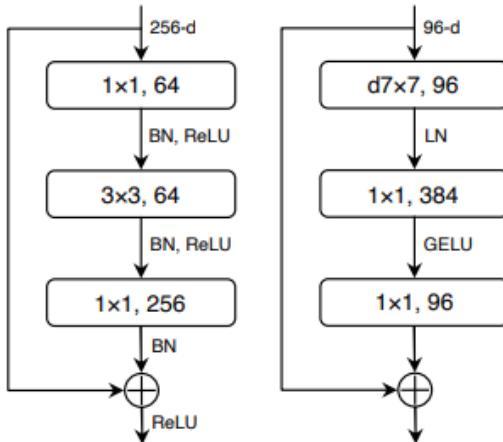
Uspjeh ResNeta je pokazao da problem degradacije nije samo pitanje dubine mreže, već temeljno pitanje načina na koji su neuronske mreže projektirane. Njegovo rješenje s preskočenim vezama otvorilo je nove mogućnosti za projektiranje dubokih i moćnih neuronskih mreža, koje su od tada postale temelj modernog strojnog učenja.

Također smo probali napraviti model koji kombinira 2 ResNet modela s 2D konvolucijom mel-spektrograma i 1D konvolucijom običnog waveforma s potpuno povezanim slojevima na kraju, ali nismo dobili bolje rezultate.

4.3.3 ConvNeXt

ConvNeXt je nova arhitektura iz 2022. godine, nastala je unaprjeđivanjem i moderniziranjem više različitih dijelova ResNet arhitekture. Prvo je promijenjen način treniranja (optimizacijske strategije i povezani hiperparametri). Zatim je promijenjen broj resnet blokova u svakoj fazi s [3,4,6,3] (kao što se vidi na slici 4.5) na [3,3,9,3]. ConvNeXt je isto tako promijenio matične ćelije iz ResNeta u sloj s "patchify" strategijom koji radi nepreklapajuću konvoluciju. Također su povećali efikasnost mreže tako da može imati više parametara. Ono što je dodatno poboljšalo ConvNeXt je korištenje većih veličina kernela (7x7 u odnosu na 3x3), promjena aktivacijskih funkcija iz ReLU u GELU i micanje nekih normalizacijskih slojeva.

ResNet Block ConvNeXt Block

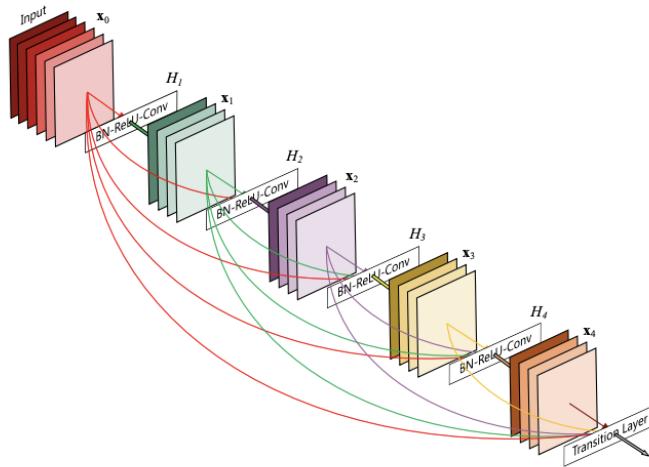


Slika 4.6: Razlike u blokovima

4.3.4 DenseNet

DenseNet (engl. Dense Convolutional Network) je arhitektura neuronskih mreža koja se koristi za klasifikaciju slika. DenseNet arhitektura predstavlja poboljšanje u odnosu na tradicionalne arhitekture konvolucijskih neuronskih mreža. Jedna od glavnih prednosti DenseNet-a je njegova sposobnost da efikasnije koristi informacije iz svih slojeva mreže. Ova arhitektura koristi blokove nazvane "Dense blocks" u kojima se svaki sloj povezuje sa svim prethodnim slojevima unutar istog bloka.

To znači da se informacije koje se obrađuju u ranijim slojevima mreže mogu direktno koristiti u kasnijim slojevima, što omogućuje učenje složenijih značajki slika i bolju generalizaciju. Druga prednost DenseNet-a je njegova efikasnost u učenju. U tradicionalnim arhitekturama neuronskih mreža, slojevi su dodani jedan po jedan kako bi se povećala dubina mreže, što može dovesti do problema u učenju zbog nestajućeg gradijenta (vanishing gradient). Međutim, u DenseNet arhitekturi, svaki sloj dobiva ulaz iz svih prethodnih slojeva, što pomaže u održavanju stabilnosti gradijenta i omogućuje brže učenje. DenseNet također ima manje parametara od drugih arhitektura neuronskih mreža, što ga čini manje zahtjevnim za treniranje i omogućava prilagodbu novim skupovima podataka. Također se pokazalo da DenseNet ima bolju performansu od drugih arhitektura neuronskih mreža na različitim zadacima klasifikacije slika.



Slika 4.7: Primjer DenseNet arhitekture

Ukratko, DenseNet donosi nekoliko prednosti u odnosu na tradicionalne arhitekture neuronskih mreža, kao što su bolje korištenje informacija iz svih slojeva mreže, efikasnije učenje i manje zahtjevno treniranje. Te prednosti čine DenseNet vrlo popularnom arhitekturom u području klasifikacije slika i drugih sličnih problema u strojnog učenju.

4.4 Trening

4.4.1 Hiperparametri

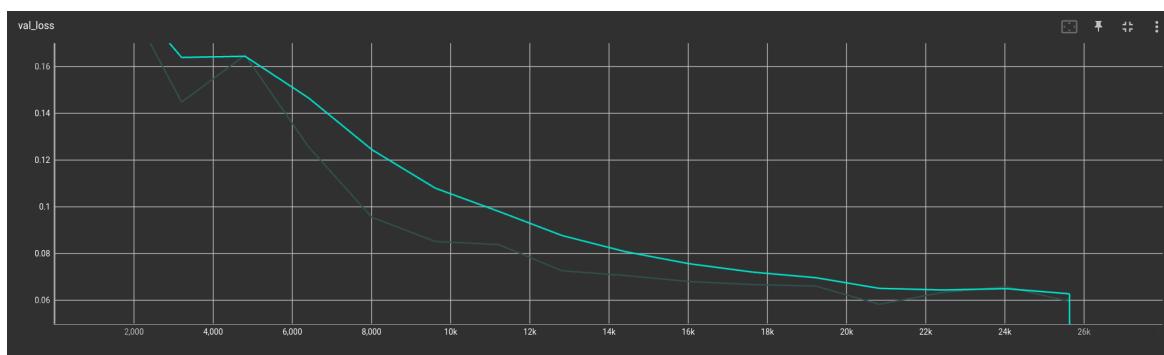
Hiperparametri (hyperparameters) su parametri koji se unaprijed određuju i podešavaju ručno prije postupka učenja neuronskih mreža u strojnog učenju. Za razliku od parametara koji se uče tijekom postupka učenja, poput težina između neurona, hiperparametri utječu na način na koji se postupak učenja odvija i na izvedbu mreže. Postoji mnogo različitih hiperparametara koji se mogu podešavati, uključujući veličinu batch-a, stopu učenja (learning rate), broj epoha učenja, broj skrivenih slojeva u mreži, broj neurona u slojevima i drugo. Pravilno podešavanje ovih hiperparametara ključno je za postizanje visokih performansi mreže u različitim zadacima strojnog učenja.

Testirali smo razne kombinacije hiperparametara, uključujući batch size, stopu učenja i broj epoha i zaključili da je optimalno uzeti batch size od 32 (za manje veličine rezultati su bili malo bolji, ali su epoge treniranja trajala predugo) i stopu učenja 0.001. Za broj epoha koristili smo rano zaustavljanje (early stopping). Early stopping tehnika se sastoji u praćenju funkcije gubitka na validacijskom skupu tijekom postupka učenja i prekidu postupka kada se funkcija gubitka na validacijskom skupu više ne smanjuje, nego počne rasti. To znači da se postupak učenja prekida prije nego što dođe do prenaučenja. Na ovaj način se sprečava mreža da se prilagodi detaljima u podacima za učenje koji nisu relevantni, što dovodi do bolje generalizacije mreže na novim podacima. Testirali smo early stopping s strpljenjem 2, 3 i 5 te su nam rezultati bili najbolji s 3. Aktivacijske funkcije koje smo koristili su RELU u slojevima prije zadnjeg, a u zadnjem sloju smo probali softmax i sigmoid aktivaciju, ali je sigmoid imao najbolje rezultate.

4.4.2 Optimizator i funkcija gubitka

Optimizator je algoritam koji se koristi za ažuriranje parametara mreže u svakom koraku učenja s ciljem minimiziranja funkcije gubitka. Postoji mnogo različitih optimizatora koji se koriste u treningu neuronskih mreža, poput stohastičkog gradijentnog spusta (SGD), AdaGrad, Adam i mnogih drugih. Odlučili smo se za Adam optimizator (Adaptive moment estimation) jer može brzo konvergirati u minimum te je pogodan za rad s velikim modelima.

Funkcija gubitka (eng. loss function) mjeri razliku između predikcija modela i stvarnih vrijednosti. Cilj funkcije gubitka je minimizirati tu razliku tijekom procesa učenja. Postoji veliki broj različitih funkcija gubitka koje se koriste ovisno o vrsti zadatka, poput regresije ili klasifikacije. Neke od najčešće korištenih funkcija gubitka su srednja kvadratna pogreška (MSE - mean squared error), binary cross entropy (BCE) i categorical cross entropy (CCE). Kako smo nad audio datotekama trebali raditi multi-label klasifikaciju, odlučili smo se za BCE funkciju gubitka koja se inače koristi za takve probleme.



Slika 4.8: Funkcija gubitka nad validacijskim skupom tijekom treniranja DenseNet121 modela

Konačni model trenirao se 16 epoha, s obzirom na *batch size* 32 i količinu podataka svaka epoha imala je 1602 koraka što nas dovodi do oko 25600 ukupnog broja koraka, ponašanje funkcije gubitka po koracima možete vidjeti na slici 4.8. Kao što možete vidjeti na slici 4.8 u zadnjih 5000 koraka (3 epohe) funkcija gubitka se prestala smanjivati te koristimo rano zaustavljanje kako bi spriječili overfitting.

5. Rezultati

5.1 Model

Konačni model koji smo koristili za klasifikaciju i aplikaciju bio je DenseNet121 arhitektura iz *torchvision* biblioteke koja kao ulaz prima *numpy array* log-mel spektrograma s oblikom (1, 128, 44), tj. audio podijeljen na dijelove od jedne sekunde te iz njega izvučen normalizirani log-mel spektrogram sa parametrima: *hop length*=512, *n-fft*=1024, *n-mels*=128. Konačnu klasifikaciju dobili smo na način da smo spojili tri operacije. Prvo smo primijenili ReLU aktivacijsku funkciju na izlaznom tensoru DenseNet121 arhitekture. Nakon toga smo primijenili adaptivni globalni sloj sažimanja *nn.functional.adaptive-avg-pool2d* kako bismo smanjili dimenzije ulaznog tensora na veličinu (1, 1). Zatim smo ravnali dobivene vrijednosti u jednodimenzionalni tensor. Na kraju, koristili smo potpuno povezanu (eng. fully connected) klasifikacijsku mrežu te sigmoidnu aktivacijsku funkciju kako bismo dobili vjerojatnost za svaki instrument te smo konačnu klasifikaciju vršili sa pragom 0.99. Visoki prag od 0.99 davao je najbolje rezultate nad IRMAS validacijskim skupom, također smatramo da bi ovaj model u produkciji dao više vrijednosti u zadacima poput organiziranja i anotiranja glazbenih podataka ako ima veći prag tj. ako radi manje netočnih pozitivnih klasifikacija (eng. *false positives*). S ovim modelom nad IRMAS validacijskom skupu postigli smo sljedeće rezultate:

Metric	Value
F1-Score	0.62
Hamming Score	0.89
Precision	0.66
Recall	0.58
Total files	2874

Tablica 5.1: Konačne metrike modela nad IRMAS validacijskim skupom

Kao što možete vidjeti u tablici 5.1 dobiveni rezultati prilično su dobri u usporedbi sa sličnim radovima na istom podatkovnom skupu. Koristili smo F1-score kao vodilju za evaluaciju našeg modela. Ostali modeli koje smo isprobali postizali su F1-score između 0.56 i 0.6 sve do ovog rezultata.

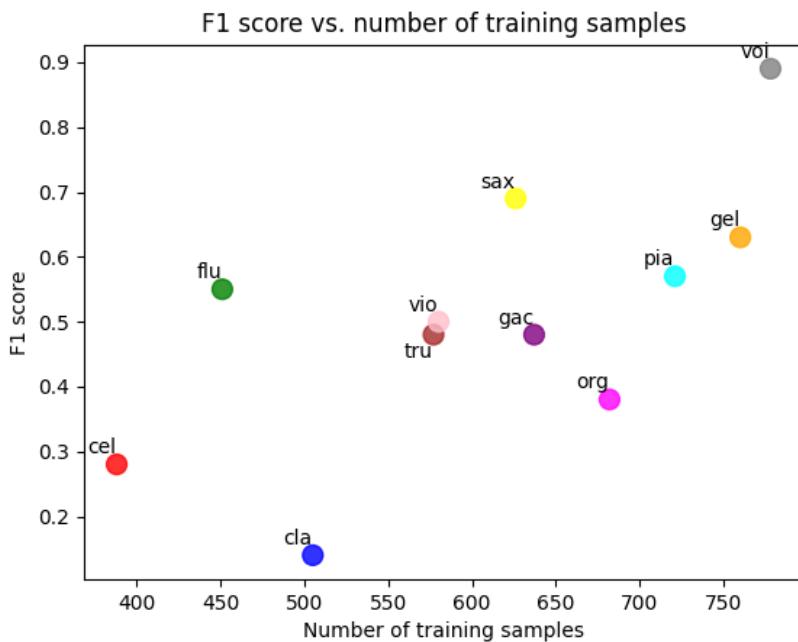
5.2 Instrumenti

Naravno nije dovoljno provjeriti samo metrike cjelokupnog modela. U problemima sa višestrukim oznakama važno je provjeriti metrike svake oznake posebno, u našem slučaju su to instrumenti: violončelo, klarinet, flauta, akustična gitara, električna gitara, orgulje, klavir, saksofon, truba, violina i glas. Na primjer, primijetili smo da je model manje precizan u klasifikaciji klarineta i violončela, možemo provjeriti je li model dobro naučio karakteristike tih instrumenata i možda ih više nglasiti u trening skupu podataka. Također možemo isprobati različite hiperparametre modela ili različite načine obrade ulaznih podataka kako bismo poboljšali preciznost klasifikacije za te oznake. Konačne metrike po instrumentima možete vidjeti u sljedećoj tablici:

Instrument	F1	Hamming Score	Precision	Recall	Occurrences
Violončelo	0.28	0.92	0.22	0.41	111
Klarinet	0.14	0.95	0.11	0.19	62
Flauta	0.55	0.94	0.48	0.64	163
Akustična gitara	0.48	0.86	0.76	0.35	535
Električna gitara	0.63	0.79	0.74	0.54	942
Orgulje	0.38	0.88	0.53	0.30	361
Klavir	0.57	0.77	0.79	0.45	995
Saksofon	0.69	0.92	0.59	0.84	326
Truba	0.48	0.92	0.38	0.64	167
Violina	0.5	0.92	0.46	0.56	211
Glas	0.89	0.92	0.88	0.91	1044

Tablica 5.2: Konačne metrike modela po svakom instrumentu nad IRMAS validacijskim skupom

Kao što možete vidjeti iz tablice 5.2 analiza metrika svakog instrumenta posebno daje nam puno više informacija o uspješnosti modela nego metrike cjelokupnog modela. Vidimo da model radi dosta loše za instrumente poput: violončela, klarineta, akustične gitare, orgulja te trube, sa F1-score vrijednosti ispod 0.5. Slučajevi poput klarineta i violončela možda možemo objasniti malim brojem pojavljivanja u validacijskom skupu, ali ovaj problem bi trebalo analizirati dublje.



Slika 5.1: Odnos broja trening uzoraka i F1-score vrijednosti modela za svaki instrument

Na slici 5.1 možete vidjeti odnos F1-scorea i broja primjera za svaki instrument u trening skupu podataka. Testiranjem Pearsonovog koeficijenta korelacije ($r = 0.6589$, p -vrijednost = 0.0274), zaključili smo s razinom signifikantnosti 0.05 da je naslućena povezanost F1-scorea i broja primjera stvarna. Nakon ovog zaključka pokušali smo raditi više augmentiranih primjera za određene instrumente poput: violončela, klarineta i flaute, s nadom da će to povećati F1-score navednih instrumenata. Nažalost, to nije uspjelo i smatramo da bi se trebale dodatno istražiti značajke svakog instrumenta te značajke instrumenata u njihovim kombinacijama kako bi se dobila nova saznanja te implementirala dodatna poboljšanja.

6. Zaključak

U ovom projektu primijenili smo konvolucijske neuronske mreže (CNN) za klasifikaciju instrumenata na temelju audio signala. Nakon obrade podataka, gdje smo iz zvuka izvukli mel spektrogramne te augmentiranjem postojećih generirali dodatne primjere, uspješno smo primijenili tehnikе računalnog vida tako da smo izgradili CNN model za prepoznavanje 11 različitih instrumenata.

Korištenjem gotovih arhitektura i različitih hiperparametara poput stope učenja, veličine batch-a, funkcija aktivacije, optimizatora i funkcije gubitka, uspjeli smo postići visoku točnost klasifikacije na validacijskom skupu. Na IRMAS Validation skupu podataka postigli smo Hamming score od 0.89 i makro F1-score od 0.62.

Uz dodatno vrijeme za rad na projektu, neke stvari bi se mogle doraditi kako bi konačan proizvod bio bolji. Na primjer, web aplikacija koja servira model može se deployati kako bi model bio javno dostupan. Moglo bi se provesti detaljnije istraživanje zbog čega model više griješi kod nekih instrumenata te to i popraviti. Također, moglo bi se testirati dodatne metode augmentiranja podataka kao i različitih arhitektura za poboljšanje modela općenito. Ideja koju nismo uspjeli isprobati, a mislimo da bi pomogla jest korištenje nekog generativnog modela (npr. WaveGAN - <https://github.com/chrisdonahue/wavegan>) za stvaranje više primjera.