Detektiranje anomalije u zvučnom zapisu korištenjem Autoenkodera baziranog na CNN-u

Projekt iz predmeta Neuronske mreže

Toni Ivanković
FER
Zagreb, Hrvatska
toni.ivankovic@fer.hr

Rea Pintar
FER
Zagreb, Hrvatska
rea.pintar@fer.hr

Lucija Marija Stanušić
FER
Zagreb, Hrvatska
lucija-marija.stanusic@fer.hr

Kristijan Verović
FER
Zagreb, Hrvatska
kristijan.verovic@fer.hr

Bjanka Vrljić
FER
Zagreb, Hrvatska
bjanka.vrljic@fer.hr

I. Opis zadatka

Tema našeg projekta je detekcija anomalije u zvučnom zapisu pomoću autoenkodera. Autoenkoderi se sastoje od dvije neuronske mreže koje zovemo enkoder i dekoder. Zadaća enkodera je uzeti podatak i sažeti ga u manju dimenziju od samog ulaznog podatka (eng. compression), a dekoder zatim pokušava od sažete verzije rekonstruirati originalan podatak, odnosno ono što je ušlo u enkoder. Usporedivši originalan podatak i izlaz dekodera možemo izračunati tzv. grešku u rekonstrukciji (eng. reconstruction error).

Naš skup podataka sastoji se od zvučnih zapisa nekoliko različitih igrački (igračka-automobil, igračka-transporter, itd.) trajanja oko 10 sekundi, a postoje dvije vrste zapisa u skupu – oni s anomalijom, i oni bez. Zvučni zapisi s anomalijom su snimljeni za igračke koje imaju tvorničku ili drukčiju grešku te stoga proizvode zvuk s anomalijom. Analogno tome, zvučni zapisi bez anomalije (normalni zapisi) pripadaju ispravnim igračkama. Naš zadatak jest koristiti autoenkodere za detekciju anomalije na sljedeći način – naš model ćemo trenirati nad skupom isključivo normalnih zvučnih zapisa (onih bez anomalije), dok ćemo ga testirati nad zvučnim zapisima s anomalijom i bez anomalije (za njih nemamo preddefiniranu oznaku, već samo znanje da se skup za testiranje sastoji od zvukova i s anomalijom i bez). Iako ovaj pristup zvuči apstraktno, glavna ideja jest da će u kod testiranja zvučni zapisi s anomalijom rezultirati većim reconstruction errorom te ćemo usporedbom ulaza i izlaza autoenkodera uvidjeti da se radilo o zvuku s anomalijom. Dodatno, bavit ćemo se obradom zvučnog zapisa, s obzirom na to da ulaz u autoenkoder neće biti sam zvučni zapis, već slikovni zapis zvan spektrogram, te je upravo spektrogram ono što će autoenkoder pokušati rekonstruirati. Naš autoenkoder baziran je na konvolucijskoj neuronskoj mreži (skraćeno *CNN*) čija arhitektura je ona dostupna na GitHub repozitoriju kao jedno od mogućih rješenja ovog zadatka. Zbog manjka resursa (autoenkoderi su računski poprilično skupi te je njihovo treniranje dosta dugačko vremenski) isprobali smo samo nekoliko kombinacija hiperparametara za ovu arhitekturu.

II. Obrada zvučnog zapisa

Jedan od zahtjeva ovog zadatka jest upravo obrada zvučnog zapisa. Kako su sami .wav zvučni zapisi preveliki za obradu i rekonstrukciju pomoću autoenkodera, prebacit ćemo se u frekvencijsku domenu i koristiti spektrograme - slikovne reprezentacije zvučnog zapisa, na kojima ćemo zapravo samo iz slike moći vidjeti radi li se o zvuku s anomalijom ili bez, kao što je prikazano na slikama Fig. 1 i Fig. 2. Na slikama je prikazan i valni oblik (eng. *waveform*) zvučnog zapisa, iz kojeg ne možemo samo iz izgleda previše toga razlučiti.

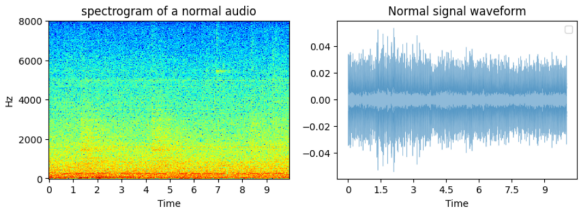

Fig. 1. Spektrogram zvuka ispravne igračke

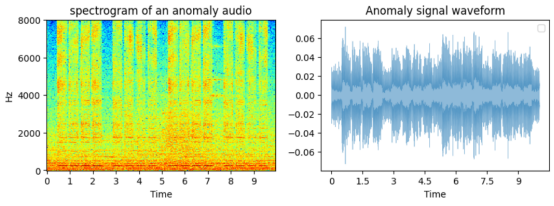

Fig. 2. Spektrogram zvuka neispravne igračke

Vide se očigledne deformacije na spektrogramu zvučnog zapisa kod igračke koja ima grešku, za razliku od spektrograma ispravne igračke koji je ujednačen. Prema ovome ćemo raspoznavati naše dvije klase zvukova, oni s anomalijom i oni bez. Konkretno ćemo koristiti logaritam spektrograma, s obzirom na to da je log-transformacija često korištena radi smanjenja složenosti jer je monotona funkcija. Log-spektrogram normalnog i abnormalnog zvuka prikazani su na Fig. 3. ispod.

Log-spektrogram kao prikazan na dva primjera ispod će biti ulaz autoenkodera, odnosno ono što će autoenkoder pokušati replicirati. Glavna premisa kod korištenja autenkodera za detekciju anomalija (eng. *anomaly detection*) jest upravo da će pogreška pri rekonstrukciji biti veća za primjere koji sadrže anomaliju nego li za ispravne primjere, s obzirom na to da se u fazi treniranja koriste samo ispravni primjeri.

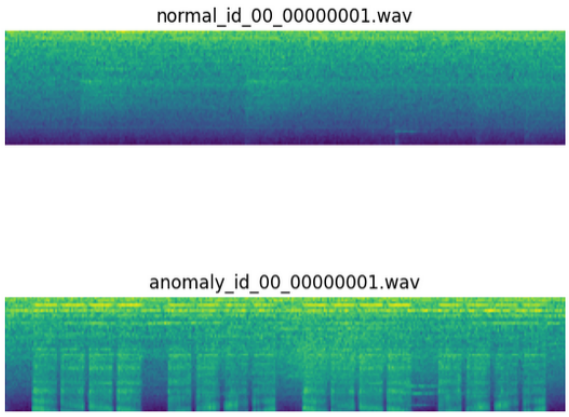

Fig. 3. Log-spektrogram normalnog i abnormalnog zvuka

Konačno, navedeni log-spektrogram transformiramo u tip podatka *vector array* prije treniranja kako bi ga autoenkoder mogao lakše obraditi, te je izlaz autoenkodera upravo još jedan *vector array* koji ćemo transformirati u slikovni prikaz log-spektrograma. Dodatni koraci predobrade podataka (eng. *pre-processing*) uključuju manipuliranje dimenzijama navedenih slika, kako bi svi ulazi u autoenkoder bili jednake veličine i sličnih značajki, što je očigledno u kodu u roditeljskoj klasi *DatasetCNN* te djeci te klase.

III. Skup podataka i pristup analizi

Skup podataka, detaljniji opis projekta (*challenge*) i skupa podataka dostupan je na DCASE stranici zadatka te je sličan zadatak objavljen na natjecanju na platformi Kaggle.

Skup podataka koji koristimo sastoji se od dijelova Toy-ADMOS skupa i skupa podataka MIMII koji sadrže normalne/anomalne zvučne zapise šest vrsta igračaka/stvarnih strojeva - Igračka-automobil i igračka-transportna traka (Toy-ADMOS) te ventil, pumpa, ventilator te klizna šina (MIMII). Svaki zapis je duljine oko 10 sekundi te uključuje radni zvuk određenog stroja i okolnu buku. Nadalje, naši podaci se dalje odjeljuju u tri odvojena skupa podataka - Development dataset, Dodatni training set te Evaluation set, a svaki služi određenoj svrsi. Naime, Development koristimo za prvu fazu treniranja modela, a on se sastoji od normalnih i abnormalnih zvučnih zapisa sedam igračaka-strojeva. Normalni zapisi se koriste za treniranje autoenkodera, dok se abnormalni zapisi koriste za provjeru treniranog modela. Kao što je spomenuto, jedna od mogućih svrha korištenja autoenkodera je detekcija anomalija, te se upravo na ovaj način dolazi do jednog takvog autoenkodera - njega ćemo trenirati na normalnim zvukovima te vidjeti kako se ponaša kad mu damo abnormalne primjere. Nakon što je model treniran isključivo na normalnim primjerima te mu damo log-spektrogram normalnog zvuka koji će on pokušati rekonstruirati, ta rekonstrukcija će sličiti onoj na Fig. 4. ispod.

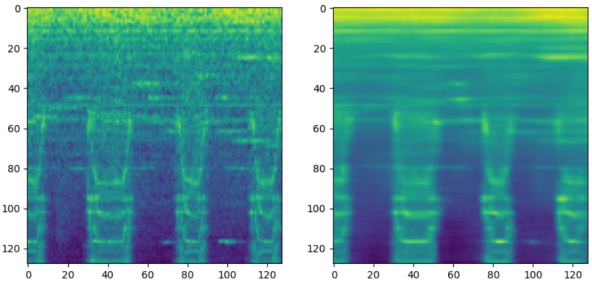

Fig. 4. Log-spektrogram zvučnog zapisa s anomalijom (lijevo - original i desno - rekonstrukcija autoenkodera)

Nakon treniranja modela na skupu za treniranje Development dataseta (samo normalni zvučni zapisi), analizirat ćemo distribuciju *anomaly scoreova* (pogrešaka pri rekonstrukciji) i ustanoviti prag za *anomaly score* na temelju kojega će naš model donositi odluku o klasifikaciji.

Dva glavna skupa podataka koji nas zanimaju su Development dataset i Evaluation dataset. Development dataset sastoji se od skupa za treniranje i skupa za testiranje, te isto vrijedi za Evaluation dataset. Skup za treniranje Development dataseta sastoji se isključivo od normalnih zvučnih zapisa, a skup za testiranje od normalnih i abnormalnih (s anomalijom) zvučnih zapisa te imamo točnu informaciju o kakvom zvuku se radi na temelju naziva datoteke zvučnog zapisa (*normal...*.wav* i *anomaly...*.wav*). Evaluation dataset je vrlo sličan, no razlika je što nemamo nikakvu informaciju o stvarnoj klasi zvučnih zapisa iz skupa za testiranje.

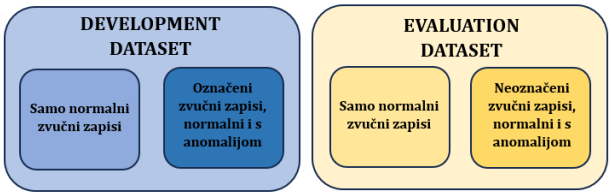

Fig. 5. Shematski prikaz skupova podataka

Također postoji dodatan skup podataka (*Additional dataset for training*) koji je naknadno objavljen nakon što je sam DCASE zadatak objavljen, te je vrlo sličan samom Development datasetu po broju zvučnih zapisa te udjelu abnormalnih zapisa. U ovom projektu nismo koristili ovaj dodatni skup podatka, odnosno nismo implementirali drugu fazu treniranja s ovim skupom zbog ograničenih resursa - kao što smo naveli, treniranje autoenkodera je vremenski i memorijski poprilično zahtjevno. Nakon treniranja modela na Development skupu,

naš model smo još jednom trenirali na skupu za treniranje Evaluation dataseta (koji se sastoji isključivo od normalnih zvukova) te zatim naš model testirali na skupu za testiranje koji se sastoji od neoznačenih primjera.

Kao rezultat prikazat ćemo neke spektrograme neoznačenih primjera iz skupa za testiranje i njihovu klasifikaciju prema našem modelu (s obzirom na to da je skup podataka relativno velik) gdje ćemo vizualno moći razlučiti radi li se o zvuku s anomalijom ili bez, te usporediti tu informaciju s klasom koju je naš model predvidio. S obzirom na to da Evaluation dataset broji više od 700 zvučnih zapisa, odlučili smo analizirati samo 10 zvučnih zapisa iz svake predviđene klase. Odlučili smo se za ovaj prikaz rezultata jer originalan DCASE Challenge nema objavljeno službeno rješenje, niti ono postoji, s obzirom na prirodu zadatka - klasifikacija ispravnosti rada igračke samo na temelju zvuka. U ovom slučaju ni ljudska presuda ni najbolji model dubokog učenja ne može dati točnu klasifikaciju za svaki primjer jer nam jednostavno trebaju druge informacije i drugi mediji za donošenje točne odluke. Na temelju nasumičnih primjera, uvidom u njihov spektrogram te pregledom klasifikacije koju je naš model donio na temelju praga ustanovit ćemo je li naš model ispravno klasificirao pojedini primjer.

Dodatno, detaljan shematski prikaz sva tri skupa podataka dostupan je na Figuri 2 na stranici zadatka DCASE.

IV. Arhitektura mreže

Arhitektura konvolucijske neuronske mreže (CNN) temelji se na korištenju konvolucijskih slojeva za obradu prostorno-strukturnih podataka poput slika. Konvolucijski slojevi služe kao izlučivači značajki (engl. *feature extraction*). Početni slojevi imaju ulogu izdvajanja jednostavnijih značajki, dok se dublji slojevi koriste za izdvajanje složenijih značajki. Stoga se izlazi ovih slojeva nazivaju mapama značajki. Svaki naredni sloj koristi mape značajki prethodnog sloja kako bi prepoznao značajke više razine.

Kod autoenkodera, dvije glavne komponente - enkoder i dekoder su simetrične kako bi se osiguralo učinkovito i vjerno rekonstruiranje ulaza. U slučaju primjene konvolucijskih autoenkodera, enkoder i dekoder dijele simetrične arhitekture, koristeći konvolucijske slojeve za izdvajanje i obnovu značajki.

Arhitektura našeg enkodera uključuje ulazni sloj, četiri konvolucijska sloja i jedan potpuno povezani sloj. Dekoder ima strukturu koja obuhvaća potpuno povezani sloj i četiri transponirana konvolucijska sloja. Figura 6 sadrži vizualni prikaz arhitekture.

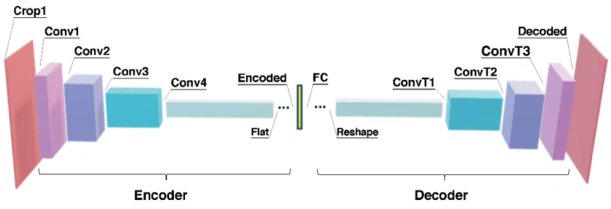

Fig. 6. Arhitektura konvolucijskog autoenkodera

Autoenkoderi su snažan alat s različitim primjenama, a ovisno o specifičnostima arhitekture, neki od njihovih glavnih zadataka uključuju redukciju dimenzionalnosti, generiranje podataka te kompresiju slika i zvuka.

V. Postavljanje praga (Thresholding)

Već smo naveli način na koji treniramo model u III. poglavlju, te ćemo u ovom dijelu detaljnije objasniti proces određivanja praga, odnosno *thresholding*.

Kada treniranom modelu damo skup za testiranje Development dataseta, podsjetimo se da ondje znamo stvarnu klasifikaciju svakog zvučnog zapisa, a model će za svaki zapis izračunati *anomaly score*. Uzevši u obzir informacije koje imamo, odnosno stvarnu klasu i anomaly score, pokušat ćemo utvrdi prag vrijednosti anomaly scorea na temelju kojega ćemo donijeti klasifikaciju, prema formuli:

$$f(x) = \begin{cases} 1 \text{ (anomalija)} & \text{ako } x \geq \text{prag} \\ 0 \text{ (normalan zvuk)} & \text{inače} \end{cases}$$

Pri čemu je x upravo *anomaly score* kojeg je naš autoenkoder izračunao. Kada vizualiziramo *anomaly scoreove* i stvarne klase na skupu za testiranje, dobit ćemo sljedeći graf (nalik *scatterplotu*, y dimenzija samo označava broj primjera, odnosno dummy varijablu).

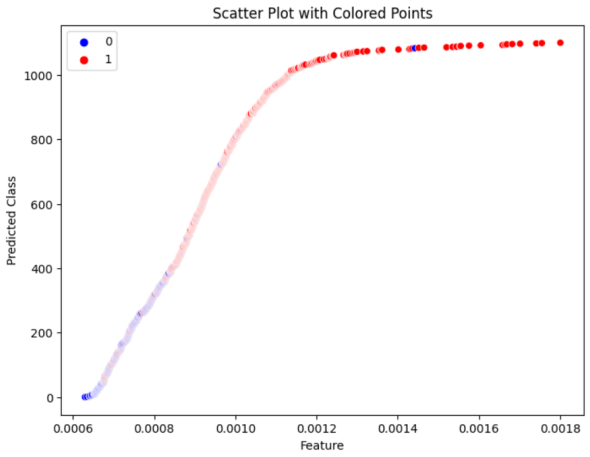

Fig. 7. Ovisnost klase (anomalija/bez anomalije) i *anomaly scorea*

Vidimo iz grafa da su točkice lijevog repa, odnosno primjeri koji poprimaju manje vrijednosti *anomaly scorea* obojeni većim dijelom u plavu boju (klasa 0 - normalan zvuk), a oko vrijednosti 0.0010 vidimo pretežno crvene točkice (klasa 1 - anomalija) što znači da bi se potencijalni prag mogao nalaziti negdje oko vrijednosti 0.0010. Ako uzmemo određeni odsječak ovog grafa gdje se najviše očituje prijelaz s pretežno normalnih zvukova (područje s puno plavih točkica) na pretežno abnormalne zvukove (crvene točkice), konkretno graf sa svim primjerima čiji je anomaly score između [0.0008, 0.0009] te također [0.00082, 0.00087] dobili bi grafove poput prikazanih ispod.

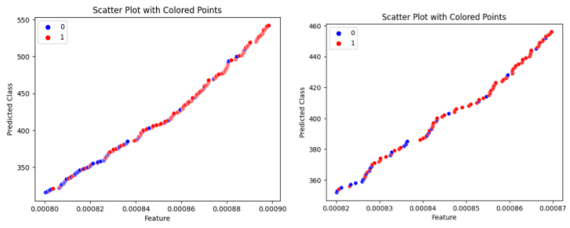

Fig. 8. Odsječci prethodnog grafa

Na temelju desnog odsječka, odnosno graf primjera koji poprimaju vrijednosti *anomaly scorea* između [0.00082, 0.00087] izabrat ćemo prag kao medijan upravo tog podskupa primjera, a to je *threshold* = 0.000847. Dakle, možemo upotpuniti našu raniju funkciju praga na način:

$$f(\text{anomaly sc.} = x) = \begin{cases} 1 \text{ (anomalija)} & \text{ako } x \geq 0.000847 \\ 0 \text{ (normalan zvuk)} & \text{inače} \end{cases}$$

VI. Testiranje na Evaluation datasetu i rezultati

Treniranje i testiranje na Development datasetu te određivanje praga možemo gledati kao prvu, a treniranje i testiranje na Evaluation datasetu kao drugu fazu treniranja. U početku postupak nije puno drukčiji od prethodne faze te autoenkoder treniramo na skupu za treniranje Evaluation dataseta, zatim mu damo skup za testiranje kako bi on izračunao *anomaly score-ove* za svaki primjer iz skupa podataka. Zatim primjenjujemo prag (thresholding) kako bismo ustanovili kojoj klasi zvuk pripada na temelju *anomaly scorea*. Na temelju 10 nasumičnih primjera iz obje previđene klase na temelju praga (sveukupno 20 primjera) testirali smo naš model, te se čini pogledom na spektrogram kao i preslušavanjem samih snimaka da je naš model na temelju *anomaly scorea* i praga kojeg smo odredili ispravno klasificirao svih 20 primjera. Naravno, nemamo informaciju o stvarnoj klasi tih primjera kako ti podaci nisu javno dostupni, no klasifikacije, zajedno s izgledom spektrograma i samim zvučnim zapisom su dosljedne onome što smo vidjeli u Development datasetu, gdje smo znali koja je stvarna klasa svakog primjera. U nastavku su prikazani po dva primjera iz svake klase, no u potpunom kodu (Jupyter bilježnici) su ispisani svih 20 nasumičnih primjera.

Na slikama ispod vidimo kako klase dodijeljene spektrogramima na temelju praga uistinu odgovaraju klasi koju bismo mi dodijelili vizualnom analizom spektrograma, točnije dva spektrograma koja su klasificirana kao normalni zvučni zapisi uistinu izgledaju kao spektrogrami normalnih zvučnih zapisa iz Development dataseta (ujednačenih su, nemaju deformacije odnosno vertikalne pruge na spektrogramu), a oni čija je predviđena klasa zvuk s anomalijom imaju deformacije upravo kakve su imali spektrogrami abnormalnih zvukova u Development datasetu.

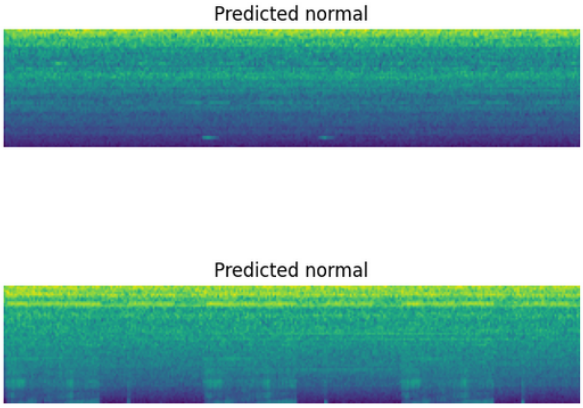

Fig. 9. Dva zvučna zapisa koje je naš model na temelju praga *anomaly scorea* klasificirao kao normalne zvučne zapise

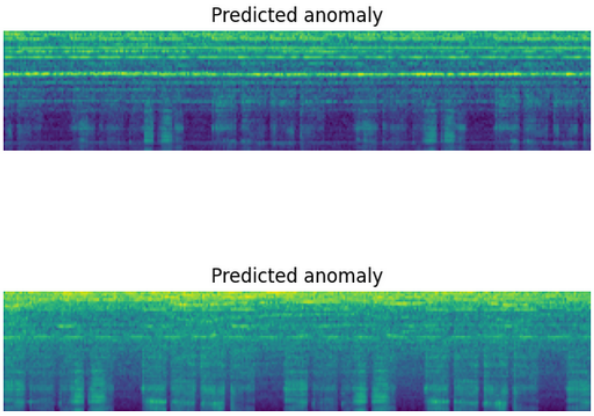

Fig. 10. Dva zvučna zapisa koje je naš model na temelju praga *anomaly scorea* klasificirao kao zvučne zapise s anomalijom

VII. Zaključak

Glavni cilj ovog projekta bila je primjena neuronskih mreža na stvarnom zadatku, odnosno korištenje neuronskih mreža kako bismo riješili neki problem. Mi smo odabrali fokusirati se na konvolucijske neuronske mreže. Konvolucijske neuronske mreže (CNN) su vrsta neuronskih mreža posebno dizajnirana za obradu prostorno-strukturnih podataka poput slika, koristeći konvolucijske slojeve za izdvajanje i prepoznavanje lokalnih značajki u ulaznim podacima, što ih čini idealnima za ovaj zadatak jer je ulaz upravo slika (spektrogram). One su često korištene u zadacima računalnog vida zbog njihove sposobnosti automatskog izlučivanja hijerarhijskih značajki iz slika. Odlučili smo ih iskoristiti za model zvan Autoenkoder, model baziran na neuronskim mrežama koji uči reprezentaciju podataka komprimiranjem ulaznih informacija u nižu dimenziju zvanu latentni prostor (ovaj posao obavlja enkoder), a zatim rekonstruira originalne podatke iz te sažete reprezentacije (posao dekodera). Ova mreža se koristi za redukciju dimenzionalnosti, generiranje podataka i detekciju anomalija, gdje se greška u rekonstrukciji koristi kao indikator prisutnosti anoma-

lije, što je upravo način na koji smo mi koristili autoenkoder u ovom projektu.

Da ponovimo, u ovom radu istražili smo detekciju anomalija u zvučnim zapisima pomoću konvolucijskog autoenkodera. Zadatak je originalno objavljen na stranici DCASE čija je poveznica dostupna u uvodu. Pristupili smo zadatku kroz dvije faze treniranja, prvo na Development datasetu, gdje smo model trenirali isključivo na normalnim zvukovima te testirali na skupu normalnih i abnormalnih zvukova te odredili klasifikacijski prag, te drugo na Evaluation datasetu, gdje smo dodatno trenirali model na normalnim zvucima.

Kroz analizu *anomaly scoreova* na skupu za testiranje Development dataseta, odredili smo prag na temelju kojega donosimo klasifikaciju izlaza autoenkodera. Navedeni pristup testiran je na Evaluation datasetu, gdje smo uspješno klasificirali neoznačene primjere u normalne i zvučne zapise s anomalijom.

Iako nemamo informaciju o stvarnoj klasi svakog primjera iz Evaluation dataseta, vizualna analiza spektrograma i zvučnih zapisa sugerira da je model dosljedan u klasifikaciji. Prikazali smo rezultate na 20 nasumično odabranih primjera (10 primjera koje je model klasificirao kao normalan zvuk te 10 koje je model klasificirao kao zvuk s anomalijom), gdje smo za svaki primjer usporedili stvarni izgled spektrograma s klasifikacijom temeljenom na pragu *anomaly scorea*. Sav korišten kod dostupan je u Jupyter bilježnici koja je dio ovog projekta. Svi korišteni podaci (3 navedena dataseta) javno su dostupni na stranici zadatka.

Za kraj želimo napraviti osvrt na tehničke poteškoće pri korištenju *PyTorcha* te autoenkodera. Za početak, općepoznato je da su autoenkoderi računski poprilično skupi te uistinu, za pokretanje i izvođenje cijele naše bilježnice potrebno je oko sat vremena, potencijalno više ako se radi o uređaju sa slabijim GPU-om ili uređaju s manjim brojem GPU-a. Dodatno, za izvođenje ovog projekta ili bilo kakvog koda u kojemu implementiramo neuronske mreže, tenzore i slično neophodno je korištenje radnog okvira (eng. *framework*) *PyTorch*. Na uređajima koji imaju novije verzije *Pythona* (primjetno na verzijama 3.9 i iznad) dolazi do nekompatibilnosti s korištenjem *PyTorcha*, ali i drugih *Python* biblioteka poput *librosa* koju smo koristili za generiranje spektrograma. *PyTorch* i razne nužne biblioteke za pokretanje ovog koda nisu dostupne na novijim verzijama *Pythona*. Iz ovog razloga naš kod smo uređivali i pokretali isključivo u *Kaggleovom* editoru koji koristi verziju *Pythona* 3.7.6 i ne stvara nikakve poteškoće s *PyTorch* okvirom niti bilo kojom od biblioteka koje smo koristili. Ako korisnik želi pokrenuti kod lokalno, potrebno je uvjeriti se da uređaj ili virtualni stroj koji se koristi ima verziju *Pythona* 3.7.6 ili nižu kako ne bi došlo do nekompatibilnosti, kao i lokalno preuzeti sve korištene skupove podataka i promijeniti *Path* (put) do skupova podataka, jer u našem kodu podaci se povlače direktno iz *Kaggleovog* repozitorija.

References

[1] Kawaguchi, Y., Imoto, K., Koizumi, Y., Harada, N., Niizumi, D., Dohi, K., ... & Endo, T. (2021). Description and discussion on DCASE 2021 challenge task 2: Unsupervised anomalous sound detection for machine condition monitoring under domain shifted conditions. arXiv preprint arXiv:2106.04492.

[2] Jalali, A., Schindler, A., & Haslhofer, B. (2020). DCASE CHALLENGE 2020: Unsupervised anomalous sound detection of machinery with deep autoencoders.