Klasifikacija fragmenata datoteka nad otvorenim skupom razreda

Petan, Laura

Master's thesis / Diplomski rad

2024

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Electrical Engineering and Computing / Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:168:312108

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2025-04-01**

Repository / Repozitorij:

FER Repository - University of Zagreb Faculty of Electrical Engineering and Computing repozitory

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 520

KLASIFIKACIJA FRAGMENATA DATOTEKA NAD OTVORENIM SKUPOM RAZREDA

Laura Petan

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 520

KLASIFIKACIJA FRAGMENATA DATOTEKA NAD OTVORENIM SKUPOM RAZREDA

Laura Petan

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

DIPLOMSKI ZADATAK br. 520

Pristupnica:	**Laura Petan (0036523368)**
Studij:	Računarstvo
Profil:	Računarska znanost
Mentor:	doc. dr. sc. Juraj Petrović

Zadatak: **Klasifikacija fragmenata datoteka nad otvorenim skupom razreda**

Opis zadatka:

Klasifikacija fragmenata datoteka važan je korak u postupcima oporavka podataka sa tvrdih diskova. Zbog složenosti tog postupka u njegovom rješavanju tipično se koriste tehnike strojnog učenja. U okviru diplomskog rada potrebno je istražiti i dokumentirati različite pristupe za klasifikaciju fragmenata datoteka nad otvorenim skupom razreda. Za istražene pristupe potrebno je ispitati njihovu učinkovitost kada se primijene na problem klasifikacije fragmenata datoteka iz FiFTy skupa podataka.

Rok za predaju rada: 28. lipnja 2024.

Zahvaljujem se mentoru doc. dr. sc. Jurju Petroviću na vodstvu kroz diplomski rad i studij.

Hvala mama i tata za sve ostalo.

Sadržaj

Uvod

Klasifikacija fragmenata datoteka je relevantan problem za područje računalne forenzike. Podrazumijeva prepoznavanje vrste datoteke na osnovi jednog fragmenta, bez da su poznate kontekstualne informacije koje su inače zapisane u datotečnom sustavu. Datotečni sustav je zadužen za pohranu podataka na disk te čuva adrese memorijskih lokacija. Događa se da se sustav pokvari ili podaci oštete te je potrebno oporaviti datoteke, klasifikacija fragmenata je prvi korak u rekonstrukciji čitave datoteke. Taj postupak je vrlo složen pa se obično koriste tehnike strojnog učenja.

U posljednjih nekoliko godina, duboko učenje je dostiglo i čak nadmašilo ljudske performanse u mnogim zadacima klasifikacije. Ove metode obično rade sa zatvorenim skupom razreda, gdje se pretpostavlja da su sve klase poznate tijekom treniranja. Međutim, u stvarnim primjenama, poznavanje svih klasa je nerealno, nepoznati razredi često se pojavljuju tijekom testiranja. Stoga su ove visoke metrike sa zatvorenim skupovima razreda u puno slučajeva nepouzdane.

Otvoreni skup razreda označava prisustvo razreda pri testiranju koji nisu bili prisutni pri treniranju mreže. Uveden je dodatni izazov u strojno učenje jer je potrebno odrediti kada je mreža dovoljno sigurna u odluku da prepoznaje određeni razred, odnosno kada je vjerojatnost pripadanja dovoljno mala da zaključi da ne pripada u do sad viđene razrede. Strojno učenje uz otvoreni skup razreda bliže modelira problem iz stvarnog svijeta. Gotovo nikad ne možemo predvidjeti sve moguće ulaze u model nakon treniranja pa se javlja potreba za rukovanjem ulazima koji se razlikuju od onih u skupu za učenje. U tom slučaju korisnija nam je informacija da je prezentiran novi razred nego što bi bila netočna klasifikacija u neki od poznatih razreda.

U sklopu ovog rada istraženi su različiti modeli i značajke za rješavanje opisanog problema. Modeli su međusobno uspoređeni te je ocijenjen doprinos novih značajki, diskretnih Fourierovih i kosinusnih koeficijenata, u odnosu na histograme. Odabrana je najuspješnija kombinacija modela i značajki te je primijenjena na otvoreni skup razreda. Ispitane su četiri tehnike klasifikacije na otvorenom skupu razreda: MSP, MLS, RPL i ARPL.

U prvom poglavlju opisana je problematika klasifikacije fragmenata datoteka. U drugom poglavlju pobliže su opisane karakteristike neuronskih mreža relevantne za ovaj rad. Treće

poglavlje opisuje značaj otvorenog skupa razreda. U četvrtom poglavlju dan je pregled transformacija podataka iz matematičke perspektive. Implementacijski detalji su opisani u petom poglavlju – odabrane značajke i modeli, korištene metrike za ocjenjivanje mreža, prilagodbe za udaljeno izvođenje programa. Posljednje poglavlje prezentira i komentira dobivene rezultate.

1. Problem klasifikacije fragmenata datoteka

Računalna forenzika je grana računalne znanosti koja se bavi pronalaženjem, prikupljanjem i obradom digitalnih dokaza u računalima i drugim medijima za pohranu. Važan dio tog procesa je dohvat podataka s diska. Datoteke se na disku pohranjuju u blokovima, grupama bajtova, tipično veličine 512 B ili 4096 B. To znači da se datoteka neće u cijelosti pohraniti na jednu memorijsku lokaciju, već će biti fragmentirana, tj. razbijena na manje dijelove i raspršena po disku. Datotečni sustav bilježi metapodatke poput veličine, tipa i lokacije fragmenata. Ako su ti metapodaci oštećeni, klasifikacija fragmenata datoteka postaje ključna za povrat podataka.

Klasifikacija ili identifikacija fragmenata podrazumijeva klasifikaciju niza bajtova u jedan od unaprijed definiranih tipova datoteka, bez dodatnih informacija iz datotečnog sustava. Ovaj postupak je značajan čak i ako je nemoguće oporaviti čitavu datoteku. U nekim slučajevima dovoljno je znati samo tip datoteke, a ne i njen sadržaj. Primjerice, prilikom analize sadržaja skinutog s interneta korisno je znati je li umetnut neki tip datoteke koji može naštetiti računalu.

Skup fragmenata za ovaj rad preuzet je iz rada FiFTy [1]. Okupljeno je najčešćih 75 razreda datoteka: jpg, arw, cr2, dng, gpr, nef, nrw, orf, pef, raf, rw2, 3fr, tiff, heic, bmp, gif, png, ai, eps, psd, mov, mp4, 3gp, avi, mkv, ogv, webm, apk, jar, msi, dmg, 7z, bz2, deb, gz, pkg, rar, rpm, xz, zip, exe, mach-o, elf, dll, doc, docx, key, ppt, pptx, xls, xlsx, djvu, epub, mobi, pdf, md, rtf, txt, tex, json, html, xml, log, csv, aiff, flac, m4a, mp3, ogg, wav, wma, pcap, ttf, dwg, sqlite. Za svaki razred dostupno je 102,000 primjera u dvije veličine bloka – 512 B i 4096 B, ukupno 7,500,000 primjera. Fragmenti su uzrokovani iz javno dostupnih datoteka, a kada to nije bilo dostatno korišteni su alati za pretvorbu tipa podataka. Na taj način je osigurana jednaka distribucija svih razreda.

2. Klasifikacija neuronskim mrežama

Strojno učenje se može svesti na problem minimizacije funkcije pogreške (engl. *loss function*). Neuronske mreže uče diskriminativnu funkciju na skupu podataka za učenje. Funkcija cilja će se u svakoj iteraciji sve više prilagođavati granici stvarne distribucije razreda tako da smanjuje pogrešku, a zadovoljava hiperparametre modela.

Hiperparametre određujemo prije samog procesa treniranja te su oni sastavni dio modela. Uključuju stupanj nelinearnosti modela – broj neurona u svakom sloju, broj i vrsta slojeva; stopu učenja, odnosno optimizator koji njome upravlja; broj epoha te broj i veličinu grupe (engl. *batch*). Kako ne bismo prenaučili mrežu na specifičnim primjerima, potrebno je odvojiti poseban skup primjera za provjeru (engl. *validation set*) nad kojim ćemo optimizirati hiperparametre.

Parametri modela se mijenjaju postupkom učenja. Kod neuronskih mreža to su težine za svaki neuron. Pronalaženjem optimalnih parametara, pod pretpostavkom da su hiperparametri dobro zadani, mreža je naučila rješavati zadani problem.

Postupak učenja je iterativan, tj. prvo odabiremo inicijalne hiperparametre i pokrećemo treniranje mreže na skupu za učenje. Kada je funkcija pogreške konvergirala, zaključujemo da su pronađeni optimalni parametri te pokrećemo ocjenjivanje mreže na skupu za validaciju. Zabilježimo rezultat te promijenimo hiperparametre. Cilj je pronaći hiperparametre koji daju najmanji gubitak uz optimalne parametre. Takav model dodatno treba ocijeniti na zasebnom skupu za testiranje kako bismo se uvjerili da mreža dobro klasificira do sad neviđene primjere.

Problem ovog rada spada u višeklasnu klasifikaciju što znači da su, za razliku od binarne klasifikacije, prisutna više od dva razreda. U ovom slučaju u posljednjem sloju želimo dobiti izlaz koji možemo protumačiti u kontekstu pripadnosti svakom razredu. Koristi se funkcija softmax opisana u poglavlju 2.2.2. Funkcija svodi izlaze mreže na raspon [0, 1] koji označava vjerojatnost pripadanja pojedinom razredu.

2.1. Umjetne i konvolucijske neuronske mreže

Umjetne neuronske mreže su vrlo moćan algoritam strojnog učenja. Sastoje se od višestrukih potpuno povezanih slojeva čiji su parametri potpuno prilagodljivi. Svaki sloj može imati, i uobičajeno ima, vrlo velik broj neurona, više nego razreda. Tolika složenost mreže lako dovodi do prenaučavanja, odnosno model se potpuno prilagodi primjerima na skupu za učenje čak i kada skup sadrži šum ili nekonzistentnosti u označavanju. Kako bi spriječili prenaučavanje mreže, između potpuno povezanih slojeva možemo konstruirati slojeve ispadanja (engl. *dropout layer*). Ispadanje znači da se nasumični neuroni u prethodnom sloju isključe, tj. težina im se postavi na nulu. Koliko neurona se isključuje je regulirano stopom ispadanja koja je svojstvo svakog takvog sloja. Umjetne neuronske mreže postižu jako dobre rezultate na jednodimenzionalnim skupovima podataka, npr. ulaz u model je rečenica teksta, a model ju treba klasificirati u jednu od emocija koju izražava [23].

Konvolucijske neuronske mreže su nešto kompliciranije arhitekture jer sadržavaju konvolucijske slojeve. Konvolucijski slojevi razlažu ulazni signal na više kanala, odnosno dodaju dimenziju. Preko svakog kanala prolazi filter i izvršava operaciju konvolucije. Uz konvoluciju koristi se sažimanje koje reduciraju dimenzije podataka unutar kanala tako da primjeni odabranu operaciju na više ulaza, a rezultat je jedna točka. Ovi slojevi se koriste u kombinaciji s potpuno povezanim slojevima. Za pretvorbu svih kanala natrag u originalnu dimenzionalnost koristi se operacija izravnavanja. CNN-ovi su prigodni za klasifikaciju višedimenzionalnih skupova podataka, npr. prepoznavanje prometnih znakova iz slike, ali imaju široku primjenu i za jednodimenzionalne skupove [23].

Konstruiranje neuronskih mreža iz nule je dugotrajan i kompliciran proces. Za mnoge probleme su dostupni pouzdani modeli koje je moguće preuzeti i fino podesiti (engl. *fine-tuning*). Modeli u ovom radu rađeni su po uzoru na modele korištene u radu FiFTy [1], ali je arhitektura pojednostavljena.

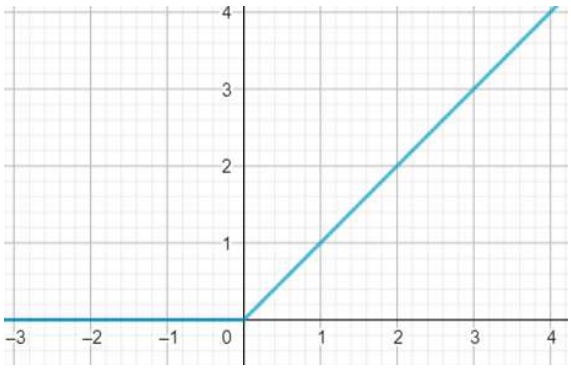
2.2. Funkcija aktivacije

Funkcija aktivacije uvodi nelinearnost u model. Bez nje bi izlaz svakog sloja bio linearna kombinacija značajki na ulazu pa ne bi bilo potrebe za konstruiranjem čitave neuronske mreže, već bismo cijeli model mogli opisati jednim slojem. Nadalje, uz nelinearnu funkciju aktivacije mreža može naučiti klasificirati i kada razredi nisu linearno odvojivi. Naravno,

primjere uvijek možemo savršeno klasificirati ako ih preslikamo u prostor gdje je broj dimenzija jednak broju značajki. Pokazalo se da ovakav pristup daje izrazito loše rezultate jer skup za učenje nikada nije iscrpan te podaci gotovo uvijek sadržavaju šum. Pri problemu klasifikacije nad otvorenim skupom razreda ovaj pristup bi bio dodatno neuspješan jer bi bilo vrlo teško odrediti koliko je mreža sigurna u klasifikaciju. Uz odabir nelinearne funkcije aktivacije nema potrebe za preslikavanjem primjera u prostor previsoke dimenzionalnosti. U nastavku su pobliže opisane dvije funkcije koje su korištene u ovom radu.

2.2.1. Funkcija ReLU

Populalrni izbor funkcije aktivacije u skrivenim slojevima mreže je funkcija zglobnice (ReLU, eng. *rectified linear unit*). Funkcija, prikazana grafom na slici (Slika 2.1), ponaša se kao linearna funkcija za pozitivne ulaze, dok ostale ulaze svodi na nulu što možemo zapisati formulom (1).



Slika 2.1 Graf funkcije zglobnice.

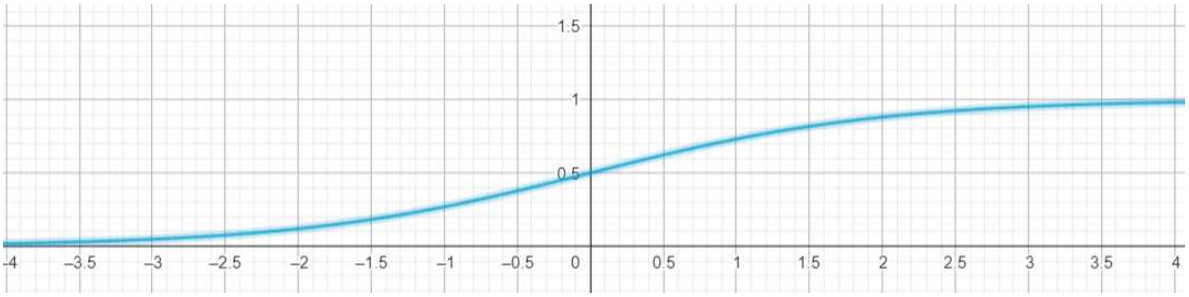
$$f(x) = \begin{cases} x, x > 0 \\ 0, \text{inače} \end{cases} \quad (1)$$

Prednost ReLU-a je što je derivabilna svugdje osim u nuli gdje možemo ručno postaviti iznos derivacije. Za ulaze veće od nule funkcija linearno raste pa joj je iznos gradijenta konstantan, ne smanjuje se. Ova dva svojstva su ključna za optimizacijski algoritam opisan u poglavlju 2.4. Nadalje, budući da ulaze manje od nule priteže na nulu, ReLU osigurava da će mnogi neuroni biti isključeni, odnosno model će biti rijedak (engl. *sparse model*).

2.2.2. Funkcija softmax

U izlaznom sloju želimo dobiti rezultat u rasponu [0, 1] koji ćemo protumačiti kao vjerojatnost da primjer pripada pojedinom razredu. Nažalost, izlaz ReLU-a nema vjerojatnosnu interpretaciju.

Sigmoidalna funkcija je prikazana na grafu u nastavku (Slika 2.2) te je možemo opisati formulom (2). Vidimo da su asimptote sigmoide 0 za negativne ulaze i 1 za pozitivne. Ova funkcija sama po sebi nije primjerena za višeklasnu klasifikaciju.



Slika 2.2 Graf funkcije sigmoide.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2}$$

Prilagođena sigmoidalna funkcija koja na izlazu daje vektor zove se softmax. Funkcija softmax dana je formulom (3) te radi na prethodno opisani način, samo što se sada količnik izračunava za svaki razred zasebno.

$$\text{softmax}(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}} \tag{3}$$

2.3. Funkcija gubitka

Funkcija gubitka opisuje koliko se predviđena i stvarna vrijednost razlikuju. Strojno učenje radi na principu minimizacije funkcije gubitka.

2.3.1. Unakrsna entropija

Gubitak unakrsne entropije (engl. *cross-entropy loss*) se koristi kod klasifikacije, specifično kada je u zadnji sloj uključena funkcija softmax. Prigodan je i za binarnu i za multiklasnu klasifikaciju. Formula (4) za izračun unakrsne entropije za N primjera i K razreda je dana

u nastavku. Gubitak je jednak sumi svih oznaka primjera $y_{i,j}$ pomnoženog s logaritmom vjerojatnosti klasifikacije $p_{i,j}$.

$$\text{loss} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{K}\left(y_{i,j} \cdot \log(p_{i,j})\right)$$

(4)

Pri svakom prolazu odabiremo jedan razred j koji će biti pozitivan, ako primjer i pripada tom razredu oznaka $y_{i,j} = 1$ inače je $y_{i,j} = 0$. Vidimo da gubitku u svakoj iteraciji doprinose samo oni primjeri koji pripadaju promatranom razredu, inače množimo nulom i gubitak je nula. Ukoliko je vjerojatnost $p_{i,j}$ da takav primjer pripada u pravu klasu jednaka 1, gubitak je ponovno 0 jer je $\log(1) = 0$. Ukoliko mreža pridaje manju vjerojatnost toj klasi, pojavljuje se gubitak. Možemo primijetiti da gubitak za $p_{i,j} = 0$ nije definiran zbog logaritamske funkcije. Međutim, to ne predstavlja problem jer softmax funkcija nikad ne doseže nulu.

2.3.2. Srednja kvadratna pogreška

Srednja kvadratna pogreška (MSE, engl. *Mean Squared Error*) je funkcija gubitka prikladna za problem regresije, što je potrebno pri učenju anti-prototipa razreda u algoritmu RPL. MSE za N razreda je dana formulom (5). Budući da radimo regresiju, y_i više nije oznaka razreda već je realni broj. Predviđena vrijednost je y_i^{pred}, a prava vrijednost y_i^{true}.

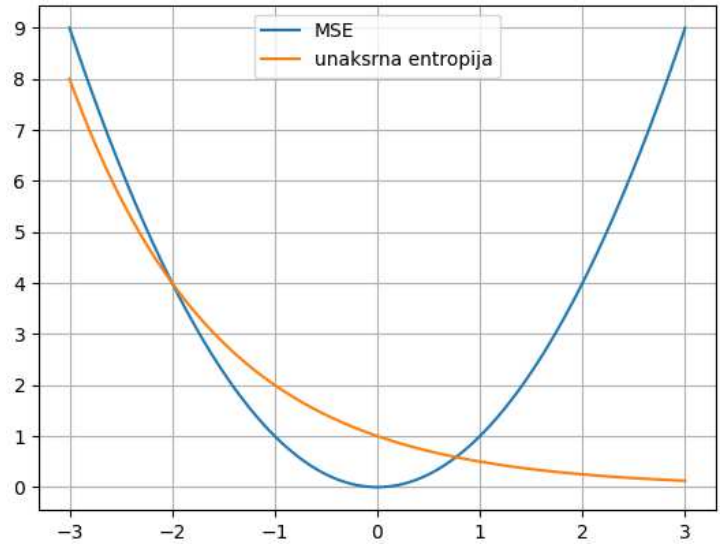
$$\text{loss} = \frac{1}{N}\sum_{i=1}^{N}\left(y_i^{\text{pred}} - y_i^{\text{true}}\right)^2$$

(5)

MSE kažnjava sve vrijednosti koje se razlikuju od istinske proporcionalno razlici istinske i predviđene vrijednosti. Isto kao i unakrsna entropija, odozdo je ograničena nulom, dok odozgo nije ograničena. No, kod ove funkcije gubitak gotovo nikada neće iznositi 0 jer će se predviđena i istinska vrijednost vrlo rijetko u potpunosti poklopiti. To znači da nećemo moći dobiti rijetke modele bez obzira što će težine nekih neurona biti zanemarivo malene. Važno svojstvo obaju funkcija je derivabilnost što je ključno za postupak optimizacije.

Na grafu (Slika 2.3) je prikazana usporedba funkcija gubitka u slučaju binarne klasifikacije odnosno regresije. Vidimo da MSE jednako kažnjava primjere s obje strane granice – koji

nisu 0. Ovo ima smisla za regresiju kojoj je cilj približiti se specifičnoj vrijednosti. Unakrsna entropija kažnjava netočno klasificirane primjere, one manje od 0, puno više nego točne. To je ponašanje koje želimo za klasifikaciju.



Slika 2.3 Prikaz dvije funkcije gubitaka: srednja kvadratna pogreška (plava boja) i unakrsna entropija (narančasta boja).

2.4. Postupak optimizacije

Optimizacijski algoritam u neuronskim mrežama je zaslužan za ažuriranje težina prilikom propagacije pogreške unazad (engl. *backpropagation*). Najčešće korišten algoritam je gradijentni spust. Kao što ime sugerira, algoritam se „spušta“ po funkciji pogreške u smjeru obrnutom od gradijenta. Na taj način pronalazi parametre koji minimiziraju grešku. Za ovaj postupak važno je da je funkcija pogreške derivabilna kako bismo mogli odrediti gradijent. Koliko daleko se algoritam spusti u smjeru gradijenta, odnosno koliko promijeni parametre mreže, određeno je stopom učenja. Bitno je da ne zadamo preveliku stopu učenja jer se može dogoditi da algoritam preskoči globalni minimum. Premala stopa učenja može značiti da pogreška neće stići konvergirati prije nego iscrpimo sve primjere.

U ovom radu se koristi grupni gradijentni spust (engl. *batch gradient descent*) što znači da se algoritmu prezentira `BATCH_SIZE` primjera prije nego što se ažuriraju težine. Na temelju tih primjera se izračuna prosječni gradijent te se tek onda nastavlja s propagacijom unazad. Prednost ovog pristupa je što doseže minimum puno brže od klasičnog gradijentnog spusta gdje se težine ažuriraju tek nakon prezentiranja čitavog skupa podataka. Također, nije toliko varijabilan kao stohastički gradijentni spust koji ažurira težine nakon svakog primjera. Zbog

regulacije stope učenja odabran je optimizator Adam (engl. *Adaptive Moment Estimator*) [10]. Adam radi na principu gradijentnog spusta, no regulira stopu učenja za svaki parametar zasebno. Dodatno, stopa je varijabilna i mijenja ju tijekom cijelog postupka učenja.

3. Otvoreni skup razreda

U okruženju otvorenog skupa razreda model mora ne samo razlikovati klase iz treniranja, već i naznačiti ako ulaz dolazi iz klase s kojom se još nije susreo. Iako ovaj scenarij bolje modelira problem iz pravog svijeta, još uvijek nije široko zastupljen u istraživanjima, koja se i dalje većinom fokusiraju na zatvoreni skup razreda. Postoje mnogi predloženi algoritmi kao i metrike za ocjenjivanje ovih modela, no većina njih mnoge stvari ostavlja otvorenima i ovisi o implementaciji. Prepoznavanje nad otvorenim skupom razreda (OSR, engl. *Open Set Recognition*) je relativno nov problem, prvi puta definiran 2013. godine [15]. Međutim, uočavamo sličnosti s nekim već postojećim područjima – detekcija anomalija, detekcija stršećih vrijednosti, itd. Pri konstruiranju modela za OSR moguće je koristiti neke tehnike iz navedenih područja.

Za ovaj rad preuzeti su skupovi podataka iz tri izvora. Prvo istraživanje [13] okuplja različite slikovne datoteke, drugo istraživanje [14] audio datoteke te posljednje istraživanje [12] okuplja video datoteke. Iz ta tri skupa podataka izdvojeni su razredi datoteka koji nisu bili prisutni tijekom treniranja, to su: bpg, flif, jpeg, jpf, jxr, webp, aac, adpcm, alaw, amr, awb, cvsd, g726, g729, gsm, ilbc, opus, pcm, speex, ulaw, asf, flw, rmvb. Fragmenti iz sva tri rada su prikupljeni u blokovima veličine 1024 B. Budući da su fragmenti iz zatvorenog skupa dostupni u veličini 512 B i 4096 B, nove datoteke je potrebno prilagoditi. Iz svakog fragmenta od 1024 B načinjena su dva veličine 512 B. Nije moguće sastaviti veći fragment iz četiri manja pa je istraživanje otvorenog skupa učinjeno samo na blokovima od 512 B. Iz originalnih datoteka dobiveno je ukupno 240.480 primjera koji su obrađeni na isti način kao i primjeri iz poznatog skupa razreda. U ovom slučaju nemamo jednaku distribuciju razreda, no kod otvorenog skupa to nije bitno jer se svi primjeri tretiraju kao jedna nepoznata klasa. Iz navedenih datoteka uočavamo mnoge koje imaju sličnosti s onima iz zatvorenog skupa. Možemo očekivati da će modeli imati većih poteškoća u razlikovanju tih specifičnih razreda, dok će datoteke koje nisu slikovni, audio- ni videozapisi biti točnije klasificirane.

U nastavku poglavlja opisane su četiri tehnike koje su predmet ovog rada.

3.1. MSP

Maksimalna softmax vjerojatnost (MSP, engl. *Maximum Softmax Probability*) je prvi ikad isprobani pristup klasifikaciji nad otvorenim skupom razreda [15]. Intuitivno slijedi iz

arhitekture neuronskih mreža. Nakon izlaznog sloja i aktivacijske funkcije softmax dodajemo funkciju praga. Ako je najveća vjerojatnost pripadnosti nekom razredu veća od praga, zaključujemo da primjer doista pripada tom razredu. U suprotnom, mreža nije dovoljno sigurna u klasifikaciju te zaključujemo da primjer pripada nepoznatom razredu.

Za MSP, kao i za sve tehnike koje ovise o pragu, ne možemo pouzdano tvrditi da će ocjene na testnom skupu odgovarati stvarnim performansama. To je zato što udio nepoznatih razreda na testnom skupu utječe na optimalni prag. Ukoliko ima puno manje nepoznatih primjera, ima smisla odabrati viši prag sigurnosti. Tada će manje primjera biti klasificirano kao nepoznato, no budući da ih je značajno manje nego poznatih, metrike se neće puno promijeniti. Vrijedi i obratno, što više nepoznatih primjera dovedemo za vrijeme testiranja, to je bolje odabrati niži prag. Ne možemo unaprijed znati distribuciju razreda koje će mreža vidjeti za svog rada pa proizvoljno odabrani prag predstavlja problem.

U radu [22] predlaže se odabir praga prema formuli (6). Prag ovisi o broju nepoznatih razreda U i poznatih razreda K. Pri rješavanju stvarnih problema klasifikacije ne možemo znati koliko ćemo imati nepoznatih razreda, inače ne bili nepoznati. Zato je svejedno potrebno prag odrediti eksperimentalno.

$$prag = 1 - \sqrt{\frac{U}{K+U}} \tag{6}$$

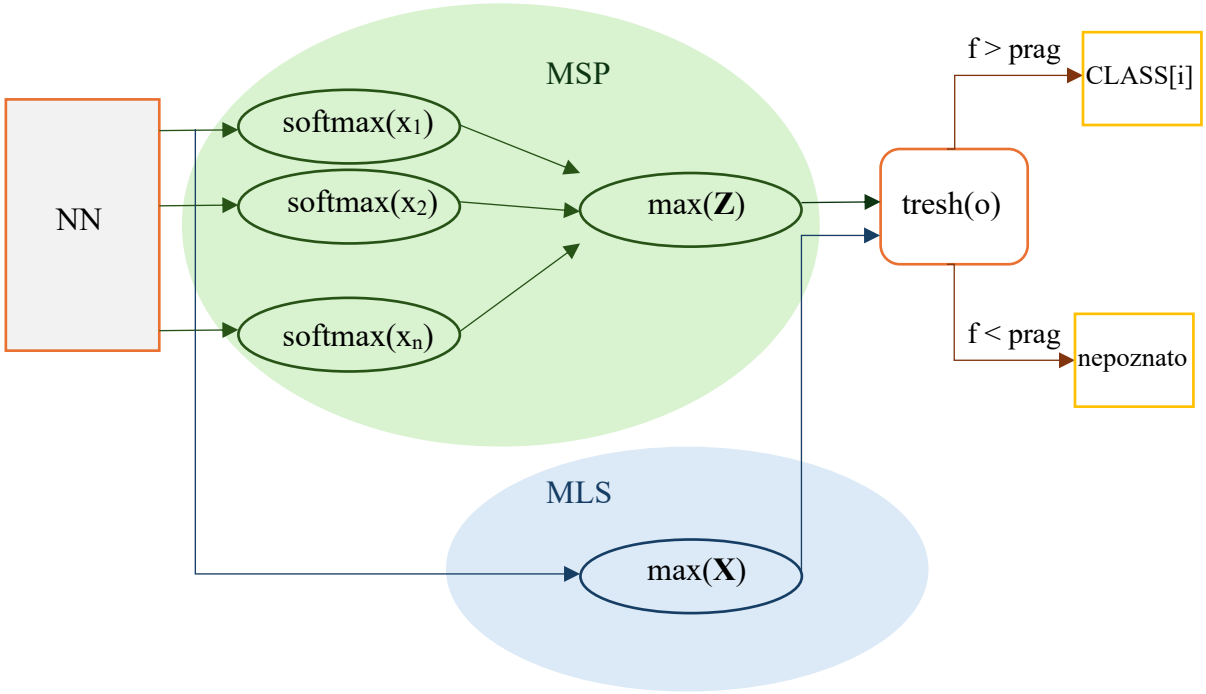
3.2. MLS

Poboljšanje u odnosu na MSP je odlučivanje prema maksimalnom logit rezultatu (MLS, engl. *maximum logit score*). Logit je izlaz iz neurona posljednjeg sloja prije nego što je doveden na aktivacijsku funkciju, u ovom slučaju softmax. U ovom trenutku vrijednosti nisu normalizirane. To znači da se kreću u puno većem intervalu i puno ih je lakše međusobno razlikovati. Sada možemo lakše i preciznije odrediti prag za sigurnost klasifikacije. Problem pri određivanju praga je što logit vrijednosti nisu ograničene. Koristimo skup za validaciju kako bismo odredili približan raspon vrijednosti izlaza. Prag se može postaviti na vrijednost na određenom percentilu. Drugi način za određivanje praga je korištenje jedne od funkcija koje služe za obradu signala. Otsuova metoda se koristi pri pretvaranju slika u boji u crno-

bijele slike. Metoda automatski pronalazi vrijednost u nizu bajtova koja minimizira varijancu unutar klasa.

Iako postiže bolje rezultate od MSP-a ova tehnika i dalje ovisi o proizvoljnom pragu pa nije pouzdana. No, u slučaju da nam je poznat udio novih razreda pri testiranju, MLS je dobar izbor.

Na slici (Slika 3.1) je uspoređen način rada MLS-a i MSP-a.

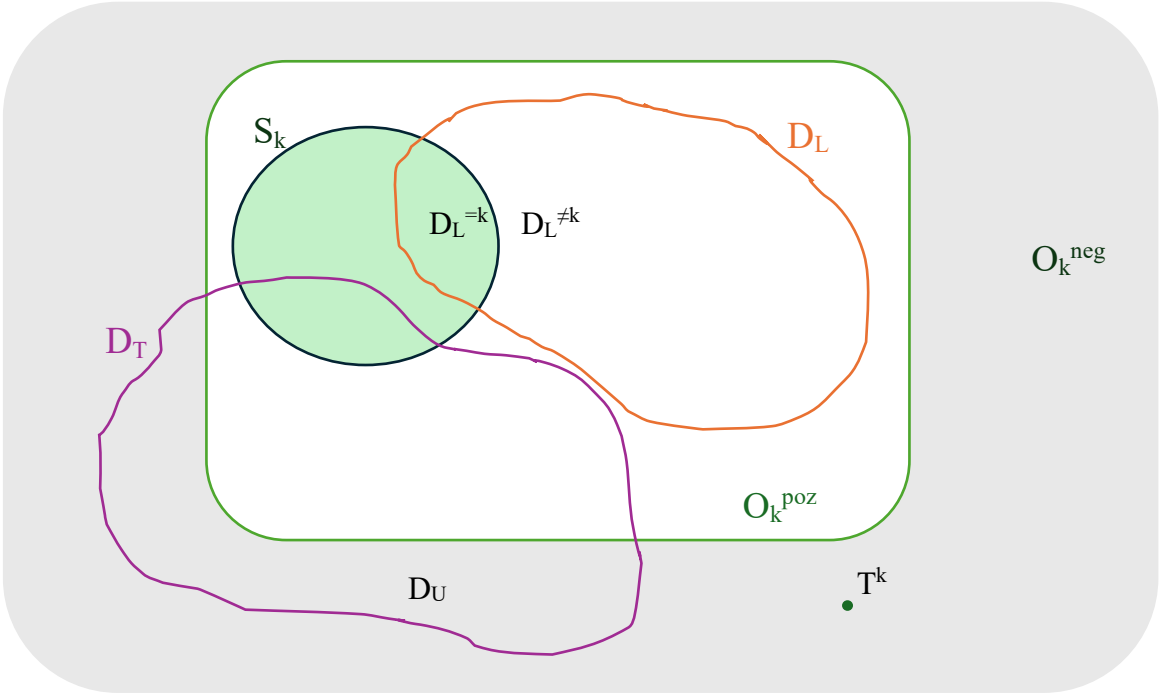


Slika 3.1 Uspoređene su dvije tehnike učenja na otvorenom skupu podataka – MSP (gore, zelena boja) i MLS (dolje, plava boja). Izlazi iz neuronske mreže NN se u slučaju MSP-a prvo dovode na funkciju softmax, a zatim se ocjenjuje sigurnost u klasifikaciju pomoću funkcije praga. Logits-score preskače taj korak, maksimalni izlaz iz mreže dovodi direktno na funkciju praga tresh().

3.3. RPL

Učenje recipročnih točaka (RPL, engl. *Reciprocal Point Learning*) inovira rješavanje problema otvorenog skupa razreda. Svaki primjer preslikan je iz ulaznog prostora u prostor značajki funkcijom $\Phi(x)$. Cijeli taj prostor predstavlja beskonačan skup svih razreda. Jedan razred bit će ograničen podskup prostora, svi primjeri koji pripadaju razredu k nalaze se unutar podskupa S_k. Nadalje definiramo skupove za učenje D_L i treniranje D_T. Iz perspektive klase k cijeli prostor koji ne uključuje skup S_k je otvoreni prostor O_k. O_k dijelimo na pozitivni

dio – ograničeni podskup s poznatim razredima O_k^{poz} i negativni dio – beskonačan podskup s nepoznatim razredima O_k^{neg}. Na sličan način iz skupa D_T izdvajamo podskup D_U koji sadrži primjere iz nepoznatih klasa prisutne pri treniranju. Skup za učenje dijelimo na podskup koji sadrži primjere iz klase k, $D_L^{=k}$, i disjunktni skup $D_L^{\neq k}$. Odnosi imenovanih skupova prikazani su na slici (Slika 3.2).



Slika 3.2 Prikazan je otvoreni prostor značajki O_k.

Poanta RPL-a nije da nauči kada primjer pripada već kada primjer ne pripada niti jednom poznatom razredu. Uvodimo recipročne točke, točke koje predstavljaju „anti-razred“. Potrebno je pronaći točku T_k takvu da je svaki primjer iz skupa S_k udaljeniji od nje nego svaki primjer iz skupova D_U i $D_L^{\neq k}$. Formalno to možemo izraziti formulom (7). Funkcija ζ vraća skup udaljenosti predane točke od svih točaka iz predanog skupa.

$$max\left(\zeta\left(D_L^{\neq k} \cup D_U, T^k\right)\right) \leq min\left(\zeta\left(D_L^{=k}, T^k\right)\right) \quad (7)$$

Što je veća udaljenost primjera od T_k, to je vjerojatnije da primjer pripada klasi k. Vjerojatnost pripadnosti možemo dobiti direktno uvrštavanjem euklidske udaljenosti

primjera od točke u softmax funkciju. Uvodimo hiperparametar γ koji određuje koliko tvrdo se udaljenost pretvara u vjerojatnost.

$$p(y = k|x, \Phi, T^k) \; = \; \text{softmax}(\gamma \cdot d_e(\Phi(x), T^k))$$

(8)

Još je potrebno odrediti margine oko točke T^k. Oblik podskupa „anti-razreda“ je hipersfera kako bismo osigurali jednaku maksimalnu udaljenost u svim smjerovima. Stoga, dovoljno je pronaći promjer r. Promjer definiramo prema već poznatom izrazu za udaljenost. Vrijedi formula (9).

$$\max\left(\zeta\left(D_L^{\neq k} \cup D_U, T^k\right)\right) \leq r^k$$

(9)

Sada možemo izraziti funkciju pogreške. Pogreška na otvorenom skupu jednaka je razlici udaljenosti primjera od recipročne točke i promjera (10). Ukoliko je ta razlika negativna, pogreška iznosi 0.

$$R_o(x; \theta, T^k, r^k) = \max(d_e(\Phi(x), T^k) - \; r^k, 0)$$

(10)

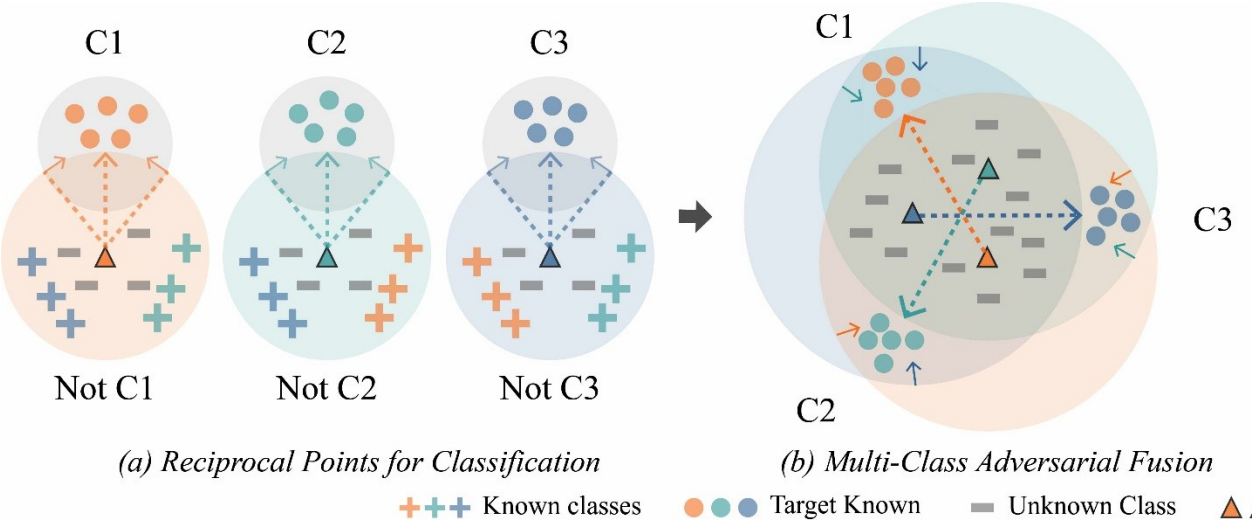
Iz svega navedenog slijedi da se primjeri iz $D_L^{\neq k}$, kada je minimiziran gubitak, nalaze unutar margine, dok su primjeri $D_L^{=k}$ izvan nje. Namještanjem hiperparametara moguće je dozvoliti mekšu ili tvrđu marginu, odnosno možemo propustiti neke primjere unutra ili van koji tamo ne pripadaju.

Možemo primijetiti da ovime nisu pokriveni primjeri iz skupa za testiranje koji pripadaju poznatim razredima, tj. primjeri iz zatvorenog skupa. To je zato što RPL kombinira dvije funkcije pogreške. Prva funkcija pogreške zadužena je za pravilni pronalazak recipročnih točaka i ugađanje margina. Druga funkcija pogreške je zadužena je za pravilnu klasifikaciju poznatih primjera, obično unakrsna entropija. Za multiklasnu klasifikaciju funkcija pogreške je suma pogešaka za svih N klasa (11). Hiperparametrom α možemo regulirati koliku važnost pridajemo pogrešci nad otvorenim skupom.

$$R = \sum_{k=1}^{N} R_{\text{cross-entropy}} + \alpha \cdot \sum_{k=1}^{N} R_O$$

(11)

Posljedica ovog postupka je da su podskupovi poznatih klasa pritisnuti u hipersferu uz rubove prostora, dok se recipročne točke nalaze između njih, no na drugoj strani prostora. Na slici (Slika 3.3) vidimo taj efekt za dvodimenzionalni prostor.



Slika 3.3 Poznate klase (kružići) su potisnute na rubove prostora u kružnicu. Recipročne točke (trokutići) se nalaze unutar njih, no na drugoj strani prostora. Slika je preuzeta iz rada [11].

3.4. ARPL

Učenje suprotstavljenih recipročnih točaka (ARPL, eng. *Adverserial Reciprocal Point Learning*) radi na istom principu kao RPL. Razlikuju se u tome kako izračunavaju udaljenost od recipročne točke. Dok RPL jednostavno mjeri euklidsku udaljenost, ARPL kombinira euklidsku i kosinusnu udaljenost. Za transformirani primjer $\Phi(x)$ u m-dimenzionalnom prostoru značajki i recipročnu točku klase k P^k vrijedi formula (12). Ukupna udaljenost d izražena je kao razlika euklidske d_e i kosinusne d_c udaljenosti, odnosno druge normalne forme i vektorskog umnoška.

$$d\left(\Phi(x), P^k\right) = d_e\left(\Phi(x), P^k\right) - d_c\left(\Phi(x), P^k\right)$$

$$d\left(\Phi(x), P^k\right) = \frac{1}{m}\left\|\Phi(x) - P^k\right\|_2^2 - \Phi(x) \cdot P^k \qquad (12)$$

Ovakvo računanje udaljenosti vrijedi samo za pronalazak recipročnih točaka. Prilikom izračuna margina primjenjuje se samo euklidska udaljenost kako bi se zahvatilo više primjera. Zbog promjene u računanju udaljenosti, neznatno se mijenja funkcija pogreške.

Prikazana je metoda `forward()` unutar razreda `ARPLLoss`. Metoda se poziva prije svakog ažuriranja težina u modelu. Vidimo da se prvo izračunavaju dvije suprotstavljene udaljenosti. Vrijednosti se oduzmu te se izračunava gubitak klasifikacije nad poznatim razredima. Zatim se računa MSE udaljenosti primjera od centra klase u koju je svrstan. Računa se gubitak nad otvorenim prostorom. Konačna greška je suma grešaka unakrsne entropije i greške na otvorenom skupu pomnoženom regulacijskim faktorom.

```
def forward(self, x, y, labels=None):
    dist_dot_p = self.Dist(x, center=self.points, metric='dot')
    dist_l2_p = self.Dist(x, center=self.points, metric='l2')
    logits = dist_l2_p - dist_dot_p

    loss = F.cross_entropy(logits, labels)

    center_batch = self.points[labels, :]
    dis_known = (x - center_batch).pow(2).mean(1)
    target = torch.ones(dis_known.size())
    loss_r = self.margin_loss(self.radius, _dis_known, target)

    loss = loss + self.weight_pl * loss_r

    return logits, loss
```

Implementacija za ARPL i RPL gubitke preuzeta je iz rada [11].

4. Transformacije podataka

4.1. Diskretna Fourierova transformacija

Fourierova transformacija omogućuje razlaganje vremenski kontinuiranog signala na njegove frekvencijske komponente. Ovo svojstvo je korisno kada je potrebno ukloniti šum iz prikupljenih podataka, kako bi se olakšala analiza signala i istaknule bitne značajke. Međutim, u praksi se često susrećemo sa signalima koji su diskretni skupovi podataka. U takvim slučajevima, koristi se diskretna Fourierova transformacija (DFT). DFT se može koristi i kada signal ne predstavlja vremenski niz, već za bilo koji periodični signal. Primjenu nalazi u spektralnoj analizi [10], filtriranju signala [11], raspoznavanju uzoraka [12] i drugim područjima. Definira se formulom (13). Rezultat je vektor $\langle X_k \rangle$ s onoliko članova koliko ima točaka u ulaznom signalu. Svaki član X_k u frekvencijskoj domeni ovisi o svih N ulaza x_n.

$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-i \cdot 2\pi \cdot \frac{k}{N} \cdot n}$$

(13)

U slučaju ovog rada raspolažemo ulaznim podacima od 512, odnosno 4096 točaka, stoga će DFT rezultirati s 512 ili 4096 koeficijenata.

Bitno svojstvo Fourierove transformacije je simetričnost u frekvencijskoj domeni. Apsolutne vrijednosti transformiranog signala su simetrične oko nulte frekvencije (14). To je ujedno i koeficijent maksimalnog apsolutnog iznosa, budući da nema imaginarni dio.

$$X_0 = \sum_{n=0}^{N-1} x_n$$

(14)

Zahvaljujući ovom svojstvu dovoljno je pohraniti polovicu koeficijenata plus centralnu frekvenciju, dakle za 512 ili 4096 ulaznih točaka pohranit će se 257 ili 2049 koeficijenata. Ovime smo uštedjeli na memorijskim zahtjevima programa, ali i potencijalno smanjili kompleksnost mreže koja će učiti na ovim podatcima.

Osim X_0 svi ostali koeficijenti su kompleksni brojevi, pretvoreni su u polarni oblik i spremljeni kao faza i radijus. Kada tražimo maksimalne koeficijente dovoljno je pretražiti listu radijusa.

Računanje diskretne Fourierove transformacije gotovo uvijek se obavlja računalno, što zahtijeva efikasan algoritam opisan u radu [9]. Brza Fourierova transformacija (engl. *Fast Fourier transform*, FFT) je algoritam koji značajno smanjuje složenost izračuna s $O(N^2)$ na $O(N \cdot \log N)$. Algoritam radi tako da sumu koeficijenata konstantno dijeli na dva jednaka dijela – parne i neparne koeficijente. Zato FFT najbolje performanse postiže kada je broj točaka potencija broja dva. U novije vrijeme sve implementacije algoritma podržavaju i rad s drugim brojevima točaka, no u tom slučaju algoritam nije toliko efikasan. Budući da FFT većinom radi matrične operacije linearno smanjenje signala eksponencijalno smanjuje kompleksnost operacija i vrijeme izračuna.

4.2. Kosinusna transformacija

Kosinusna transformacija slična je Fourierovoj po primjenama i nekim svojstvima. Također se koristi za razlaganje diskretnih signala na njihove frekvencijske komponente, ali s nekoliko ključnih razlika. DCT se definira formulom (15) gdje je k redni broj koeficijenata koji računamo. Kao i kod DFT-a transformira se signal $\langle x \rangle$ uzorkovan u N točaka.

$$X_k = \sum_{n=0}^{N-1} x_n \cdot \cos\left(\frac{\pi}{N} \cdot \left(n + \frac{1}{2}\right) \cdot k\right) \quad (15)$$

DCT će rezultirati s onoliko koeficijenata u frekvencijskoj domeni koliko ih ima u vremenskoj. Ova transformacija nema svojstvo simetričnosti pa će biti potrebno pohraniti sve rezultantne frekvencije. Međutim, možemo primijetiti da je kodomena ovaj puta skup realnih brojeva pa neće biti potrebno pohranjivati po dva podatka za svaki koeficijent. Dakle, DFT i DCT su otprilike jednako računalno i memorijski zahtjevne. Nadalje, postoji algoritam za izračunavanje brze diskretne transformacije, FCT (engl. *Fast Cosine Transform*) istih karakteristika kao i algoritam FFT.

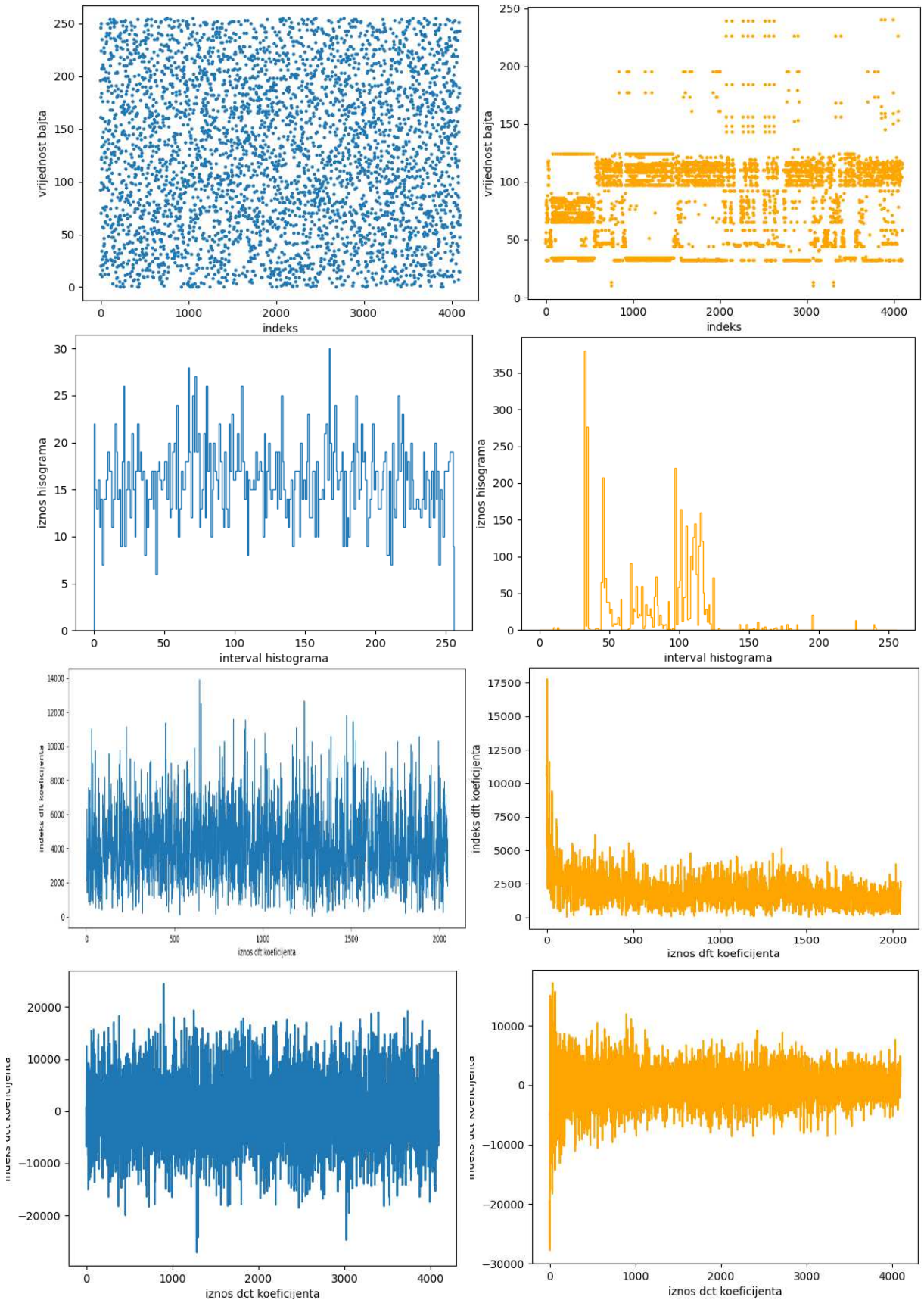
Prednost DCT-a je da koncentrira energiju signala u nekoliko prvih koeficijenata. To znači da će većina korisnih informacija biti sadržana u manjem broju većih koeficijenata što olakšava treniranje mreža. Ipak, pri transformaciji se gubi veća količina informacija nego

kod DFT-a koji radi s kompleksnim brojevima. Zbog ovih svojstva možemo očekivati bolje rezultate uz korištenje DCT koeficijenata kao značajke na manjem skupu podataka, dok će na zahtjevnijem skupu podataka biti potrebno prikazati utančanosti u razlikama razreda pa će bolji odabir biti DFT.

4.3. Histogrami

Histogrami su popularan izbor značajke općenito u području strojnog učenja, ali i specifično pri klasifikaciji fragmenata [19], [20]. Svaki podaci su ASCII znakovi pa svaki bajt možemo pretvoriti u cijeli broj u rasponu 0-255. Kada se blok podataka pretvori u histogram, pretvori se u listu s 256 članova. Član na indeksu *i* iznosi onoliko koliko je bajtova bilo jednako *i* u bloku sirovih podataka.

Na slici (Slika 4.1) je dana usporedba dva bloka podataka iz različitih datoteka. Prikazani su sirovi podatci te transformirani blokovi. Odabrani su blokovi iz .mp4 (stupac lijevo) i .csv (stupac desno) datoteke, ovaj par ekstenzija je odabran jer se razredi vidljivo razlikuju. U prvom redu prikazana je ASCII vrijednost svih 4096 bajtova. U ovakvim podatcima, posebice u .mp4 datoteci, teško je uočiti ikakve korisne informacije čime možemo opravdati potrebu za transformiranjem podataka. U drugom retku prikazani su histogrami, sada su vidljivije bitne značajke i istaknute razlike u razredima. Treći redak prikazuje realni dio DFT koeficijenata izuzev nulte frekvencije jer je prevelika za prikaz. Isto kao kod histograma, lakše možemo uočiti značajke koje sadržavaju više informacija, prema magnitudi stupca. U posljednjem redu prikazani su DCT koeficijenti ponovno bez prvog člana. Vidimo da su po magnitudi vrlo slični Fourierovima. Navedeno ilustrira na jednom primjeru da je lakše zaključivati o razredu na osnovi transformiranih nego sirovih podataka.



Slika 4.1 Prikazani su blokovi podataka veličine 4096 B iz .mp4 (lijevo) i .csv (desno) datoteka. Prvi redak prikazuje sirove podatke, a u sljedećim redovima su prikazane transformacije histogrami (drugi red), DFT (treći red), DCT (četvrti red).

5. Implementacija

5.1. Pretprocesiranje podataka

5.1.1. Izdvojene značajke

Nad originalnim skupom podataka izvršene su transformacije opisane u poglavlju 4. Svaki blok podataka pretvoren je u histograme, diskretne Fourierove koeficijente i diskretne kosinusne koeficijente.

Broj histograma je konstantan i jednak veličini bloka u bajtovima – 512 B ili 4096 B. Ova značajka se koristila kao samostalni ulaz te je služila kao referentna točka za ocjenjivanje doprinosa drugih značajki.

Nadalje, izdvojeni su diskretni Fourierovi koeficijenti u polarnoj formi. Odabran je maksimalan broj relevantnih koeficijenata kako bi se izbjeglo uvođenje šuma te smanjili resursi za treniranje mreža. Optimalan broj maksimalnih koeficijenata ovisi o specifičnom modelu te je prilagođen za svaki. Osim iznosa radijusa i kuta, za svaki maksimalni koeficijent izdvojen je i njegov indeks prema položaju u listi svih koeficijenata.

Na sličan način izdvojeni su diskretni kosinusni koeficijenti. Iz liste svih koeficijenata izdvojeni su iznos i indeks nekoliko maksimalnih.

Blok od 4096 B transformiran je u sljedeće značajke:

- 256 histograma
- 2049 DFT koeficijenata
- 4096 DCT koeficijenata

Blok od 512 B transformiran je u sljedeće značajke:

- 256 histograma
- 257 DFT koeficijenata
- 512 DCT koeficijenata

U tablici (Tablica 5.1) u nastavku prikazano je memorijsko zauzeće svake značajke. Histogrami i indeksi su pozitivni cijeli brojevi, zato se za njihovu pohranu koristi tip

`integer` koji zauzima 4 bajta. Ostale značajke su predznačeni realni brojevi te se pohranjuju kao tip `float` i zauzimaju 8 bajtova memorije.

	veličina bloka	
značajka	512 B	4096 B
histogram	$256 \cdot 4\ \text{B} \approx 1\ \text{kB}$	$256 \cdot 4\ \text{B} \approx 1\ \text{kB}$
svi DFT koeficijenti	$2 \cdot 257 \cdot 8\ \text{B} \approx 4\ \text{kB}$	$2 \cdot 2049 \cdot 8\ \text{B} \approx 32\ \text{kB}$
maksimalni $N_{DFT} = 10$ koeficijenti i indeksi	$2 \cdot 10 \cdot 8\ \text{B} + 10 \cdot 4\text{B} = 0.2\ \text{kB}$	$2 \cdot 10 \cdot 8\ \text{B} + 10 \cdot 4\text{B} = 0.2\ \text{kB}$
svi DCT koeficijenti	$512 \cdot 8\ \text{B} \approx 4\text{kB}$	$4096 \cdot 8\ \text{B} \approx 32\ \text{kB}$
maksimalni $N_{DCT} = 10$ koeficijenti i indeksi	$10 \cdot 8\ \text{B} = 0.08\ \text{kB}$	$10 \cdot 8\ \text{B} \approx 0.08\ \text{kB}$

Tablica 5.1 Prikaz koliko mjesta za pohranu koristi svaka korištena značajka. Radi ilustracije, odabran je proizvoljni maksimalni broj DFT i DCT koeficijenata – $N_{DFT} = 10$ i $N_{DCT} = 10$.

5.1.2. Skaliranje

Izdvojene značajke se izrazito razlikuju po rasponima u kojima se kreću. Štoviše, indeksi i histogrami su prirodni brojevi ograničene magnitude, a koeficijenti realni brojevi koji mogu poprimiti vrlo velike pozitivne i negativne vrijednosti. Iz takvih podataka mreža će vrlo teško učiti jer će veći doprinos težinama uvijek dati značajke većeg iznosa što u ovom slučaju nema smisla. Zato je provedeno skaliranje podataka. Postoji više metoda koje je moguće koristiti u ovu svrhu, a u ovom radu odabrana je metoda normalizacije podataka.

Normalizacija podataka podrazumijeva svođenje svih značajki na interval $[0, 1]$ ovisno o maksimalnoj i minimalnoj vrijednosti svake značajke zasebno. U sklopu Pythonove knjižnice `sklearn` postoji implementacija u obliku objekta `MinMaxScaler`. Objekt, detaljno opisan u dokumentaciji [2], skalira podatke prema formuli (16). Jedna vrijednost u listi značajki se skalira na način da se od nje oduzme minimum cijele liste te se dobiveni broj podijeli rasponom značajke. Raspon je jednak razlici najveće i najmanje vrijednosti u listi.

$$x' = \frac{x - \min(X)}{\max(X) - \min(X)}$$

(16)

Iz navedenoga je jasno da je pri normalizaciji podataka potrebno poznavati najveću i najmanju moguću vrijednost značajke. U suprotnom, potrebno je procijeniti ekstreme iz liste značajki koju koristimo za treniranje modela.

Histogrami poprimaju predefinirane diskretne vrijednosti opisane u poglavlju 4.3. Ako promatramo veličinu bloka od 4096 B to su prirodni brojevi 0-4096, dok su za blok veličine 512 B to prirodni brojevi 0-512. Stoga, ovu značajku skaliramo dijeljenjem s brojem bajtova u bloku. Slično, poznate su nam moguće vrijednosti indeksa Fourierovih i kosinusnih koeficijenata. Indeksi maksimalnih koeficijenata mogu poprimiti diskretne vrijednosti od 1 do ukupnog broja koeficijenata koje vrati transformacija. Za DFT to je polovica veličine bloka uvećana za 1, a za DCT veličina bloka. Ako koristimo numeriranje od nule (engl. *zero-based numbering*), notaciju koja se primjenjuje u Pythonu, pri skaliranju indeksa dovoljno ih je podijeliti s maksimalnom vrijednošću. Nadalje, poznate su nam i granice za iznos faze Fourierovih koeficijenata: $[-\pi, \pi]$. Ove brojeve uvrštavamo u formulu te dobivamo skaliranu značajku.

Za radijus koeficijenata DFT-ja te koeficijente DCT-ja ekstremi nam nisu unaprijed poznati. Zato je, prije transformacije podataka, napravljen prolaz kroz podatke za treniranje te su pronađene minimalna i maksimalna vrijednost. Vrijednosti su zabilježene i pohranjene u datoteku te je svaki puta prije treniranja ili testiranja `MinMaxScaler` njima inicijaliziran. Moguće je da se u podacima za treniranje nalazi vrijednost koja je izvan raspona procijenjenog na skupu za testiranje. U tom slučaju transformirana vrijednost će biti izvan traženog intervala $[0, 1]$ i testiranje će nastaviti bez greške.

U slučaju normalizacije Fourierovih koeficijenata isti su prvo sortirani prema iznosu radijusa. Zatim je zabilježena minimalna i maksimalna vrijednost radijusa te faze za svaki indeks koeficijenta, od 0 do 4096 ili od 0 do 512. Prilikom učitavanja podataka za treniranje, odabrano je nekoliko maksimalnih koeficijenata te je svaki skaliran zasebno. Sličan postupak proveden je za kosinusne koeficijente gdje su oni sortirani prema apsolutnoj vrijednosti.

U tablici (Tablica 5.2 Prikaz ekstremnih vrijednosti svake značajke te faktora kojim je skaliran) je dan popis značajki te minimum, maksimum i faktor skaliranja za svaku.

Tablica 5.2 Prikaz ekstremnih vrijednosti svake značajke te faktora kojim je skalirana

značajka	minimum	maksimum	faktor skaliranja
histogram	0	4096	$\frac{1}{4096}$
DFT indeks	0	2048	$\frac{1}{2048}$
DFT faza	-π	π	$\frac{\varphi}{2\pi}+\frac{1}{2}$
DFT radijus	0	procijenjen	$\frac{1}{\max(R)}$
DCT indeks	0	4095	$\frac{1}{4095}$
DCT koeficijent	procijenjen	procijenjen	formula (16)

Osim normalizacije, popularna priprema podataka je i standardizacija. Standardizacija se oslanja na srednju vrijednost i devijaciju podataka te je korisna kada možemo pretpostaviti Gaussovu distribuciju. U Pythonu postoji implementacija u objektu `StandardScaler` koji na podatak primjenjuje formulu (17). Od svake pojedinačne vrijednosti se oduzima srednja vrijednost te se dijeli s varijancom. Na ovaj način značajke su skalirane tako da im je srednja vrijednost 0, a devijacija jednaka 1.

$$x' = \frac{x - \mu}{\sigma} \tag{17}$$

Nedostatak normalizacije je što nije otporna na stršeće vrijednosti. Može se dogoditi da se prave vrijednosti skaliraju na puno manji interval u prisustvu vrijednosti koja značajno odskače, pri čemu će ta stršeća vrijednost imati najveći utjecaj na mrežu. Dodatno, za određene značajke u ovom radu – indeksi koeficijenata – nema smisla pretpostavljati normalnu distribuciju. Uz to, za indekse, histograme i faze su nam poznate minimalna i maksimalna vrijednost, dok bi devijaciju morali procjenjivati. Iz navedenih razloga puno je efikasnije i egzaktnije napraviti normalizaciju podataka te je u radu primjenjivan `MinMaxScaler`.

5.2. Pohrana i učitavanje podataka

Cijeli skup podataka je nakon transformacija prevelik da ga se učita u radnu memoriju. Potrebno ga je razlomiti u više datoteka. U svaku datoteku zapisano je 750 primjera, po 10 iz svakog razreda. Datoteke su zatim podijeljene na one za treniranje, testiranje i validaciju. Na taj način osigurana je jednolika distribucija razreda u svakom stadiju. Konstruiran je generator koji prima listu indeksa te prema njima odabire datoteke i iz njih vraća tražene značajke i oznake.

Da bi se pokrenulo treniranje korištena je sljedeća `keras` funkcija:

```
history = model.fit(loader_train, epochs=int(FILE_COUNT *
TRAIN_PERCENT / FILES_IN_EPOCH * MAX_EPOCH_COUNT),
steps_per_epoch=FILES_IN_EPOCH, verbose=2,
validation_data=loader_val, validation_steps=VAL_STEPS,
validation_freq=VAL_FREQ, callbacks=[early_stop],
initial_epoch=0)
```

Kao prvi argument je proslijeđen opisani generator podataka, zatim su odabrani parametri za epohe, validacijske podatke i uvjet zaustavljanja. Konstanta `FILES_IN_EPOCH` zapravo određuje koliko će datoteka biti pročitano u jednoj grupi (engl. *batch*). Tijekom cijelog rada konstanta je postavljena na 10, dakle u jednoj grupi ima 10 datoteka po 750 primjera, odnosno 7.500 primjera jednake distribucije razreda. Usprkos imenu, a zbog ograničenosti API-ja za treniranje, argument `epochs` određuje ukupan broj grupa, dok je broj epoha određen konstantom `MAX_EPOCH_COUNT`. U ovom slučaju na kraju svake grupe se poduzimaju akcije i za završetak epohe – ažuriraju se težine, vrši se validacija, izvršavaju se funkcije predane u `callbacks` argumentu. Na kraju čitave epohe, nakon što su iscrpljene sve datoteke za treniranje, lista indeksa se permutira kako bi se osigurao drugačiji redoslijed dovođenja primjera na ulaz mreže. Potrebno je balansirati argumente `epochs` i `steps_per_epochs`, odnosno broj grupa i broj primjera u grupi. Uobičajeno, ovi argumenti se postavljaju na potenciju broja 2, no kako bi se osigurala jednaka zastupljenost klasa, broj primjera u grupi je postavljen na višekratnik od 75.

Podaci za validaciju učitavaju se na isti način, pomoću generatora `loader_val` kojem su predani drugi indeksi datoteka. Validacija se ne događa nakon svake serije, već onoliko često koliko je definirano konstantom `VAL_FREQ`, a u svakom koraku validacije je pročitano `VAL_STEPS` datoteka. Konstanta `VAL_STEPS` iznosi 10 pa možemo govoriti o serijama

validacije (engl. *validation batches*) veličine 7.500 primjera. Učestalost validacije postavljena je na 10 kako bi se postignuo kompromis između brzine izvođenja i vjerodostojnosti rezultata. U tom slučaju validacija se vrši nakon što mreža vidi 75.000 primjera.

5.3. Udaljeno izvođenje

Rad s neuronskim mrežama zahtjeva puno više resursa nego što dozvoljavaju osobna računala. Budući da u svakoj iteraciji program obavlja velik broj nelinearnih matematičkih operacija, potrebna je velika procesorska snaga. Manipulacija velikog broja primjera s puno značajki podrazumijeva veliku količinu radne memorije, dok veliki skup podataka zahtjeva velike mogućnosti pohrane. Dakle, potreban je snažan procesor ili grafička kartica, dovoljno RAM-a za učitavanje i rad s dijelom skupa podataka te dovoljno memorije za pohranu čitavog skupa podataka. U tu svrhu korišten je udaljeni klaster računala Supek.

Računalni klaster je umrežena skupina nezavisnih računala s ciljem usklađenog ponašanja. Povezivanjem računala u klaster postižu se daleko bolje performanse nego korištenjem pojedinačnog računala. Računalni klaster Supek je na raspolaganju znanstvenoj i akademskoj zajednici Hrvatske sa svrhom omogućavanja stručnjacima da sudjeluju u znanstvenim projektima. Sastoji se od dva pristupna poslužitelja – CPU i GPU koji omogućuju pristup ekvivalentnim radnim poslužiteljima. Na radnim poslužiteljima se izvršava korisnički kod i pohranjuju potrebni podatci. Postoji 52 CPU radna poslužitelja, svaki od njih sadrži po dva procesora sa 64 jezgre te 256 GB radne memorije. GPU radnih poslužitelja ima 20, sastoje se od jednog procesora sa 64 jezgre, četiri grafička procesora s 40 GB RAM-a te 512 GB radne memorije ukupno. Za potrebe ovog rada korišteni su GPU poslužitelji što je uvelike ubrzalo izvođenje programa.

Da bi se na klasteru Supek pokrenulo izvođenje nekog programa, potrebno je napisati skriptu koja koristi jezik za opisivanje poslova SGE (engl. *Sun Grid Engine*). U skripti se određuje koliko resursa zahtijevamo. Resursi podrazumijevaju količinu radne memorije, vrijeme izvođenja, broj jezgara i mnoge druge parametre opisane u SGE dokumentaciji [broj]. Svaki korisnički program se pokreće u svojem kontejneru, odnosno okruženju iz kojeg nema pristup resursima koji u njega nisu instalirani. Zato je prije pokretanju programa potrebno osigurati kontejner u koji će biti instalirani vanjski paketi i knjižnice koje program koristi.

Skripta () koja pokreće treniranje jednog modela iz rada je dana u nastavku. Sve linije koje započinju znakovnim nizom `#PBS` označavaju parametre SGE. Vidimo da je projekt nazvan `train_nets` i stavljen u red za izvođenje na GPU poslužitelju. Očekivano vrijeme izvođenja je 120 h te se nakon toga program prekida. Zauzima se jedan čvor, 8 procesora, jedan grafički procesor te 10 GB RAM-a. U sljedećim redovima dana je putanja do direktorija u koji se ispisuju greške, odnosno izlaz programa. Zadnji parametri su mail adresa na koju se šalju obavijesti o stanju programa te uvjeti u kojima želimo primati obavijesti. Odabrani uvjeti su a, b i e što označava početak i kraj izvođenja te slučaj da je došlo do greške. Sljedeće, učitavaju se paketi potrebni za izvođenje programa, u ovom slučaju to je paket tensorflow. Na kraju se poziva skripta koja kreira kontejner na potrebnom čvoru i u njemu pokreće program.

```
#!/bin/bash

#PBS -P train_nets
#PBS -q gpu
#PBS -l walltime=120:00:00
#PBS -l select=1:ncpus=8:ngpus=1:mem=10GB
#PBS -e /lustre/home/jpetrovi/file_fragments_classification
#PBS -o /lustre/home/jpetrovi/file_fragments_classification
#PBS -M laurapetan2@gmail.com
#PBS -m abe

# pozovi modul
module load scientific/tensorflow/2.10.1-ngc

# potjeraj skriptu
run-singlenode.sh
/lustre/home/jpetrovi/file_fragments_classification/trainer_a
nn1.py
```

Uz ove specifikacije treniranje mreža trajalo je nekoliko dana, ovisno o modelu i odabranim značajkama. U tablici (Tablica 5.3) je dan pregled potrebnog vremena i radne memorije za treniranje nekih modela za blokove veličine 4096 B kroz jednu epohu. Radi ilustracije odabrane su kombinacije koji su zahtijevale najviše resursa. Jasno je da treniranje ne bi bilo moguće izvršiti na osobnom računalu.

Tablica 5.3 U tablici su prikazani memorijski i vremenski zahtjevi za treniranje navedenih modela. Uzeti su podaci za treniranje nad blokovima podataka veličine 4096 B. Značajka *hist* označava histograme, *DFT* i *DCT* diskretne Fourierove i kosinusne koeficijente. U zadnjem redu navedeni su podaci za pretprocesiranje podataka, tj. transformaciju svih primjera iz skupa podataka u značajke.

model	značajke	vrijeme / h	RAM / GB
ANN2	hist	9	25
CNN	hist	10	38
ANN2	hist + DFT + DCT	11	28
CNN	hist + DFT + DCT	10	38
-	izdvajanje značajki iz sirovih podataka	210	330

5.4. Modeli

Ukupno su ispitana tri modela, dva modela umjetnih neuronskih mreža – *ANN1* i *ANN2* te jedan model konvolucijskih neuronskih mreža – *CNN*. Broj čvorova ulaznog čvora za sve modele je jednak i ovisi o broju značajki koje su aktivne. Izlazni sloj u svakom slučaju ima 75 neurona jer mreže uče klasificirati 75 poznatih razreda. Za aktivacijsku funkciju u skrivenim slojevima odabrana je funkcija zglobnice, dok je u izlaznom sloju odabrana funkcija softmax. Odabir je argumentiran u poglavlju 2.2.

1. Model *ANN1* je jednostavnije arhitekture, opisan je programskim isječkom u nastavku. Model se sastoji od ulaznog i četiri potpuno povezana sloja s redom 100, 64, 32 i 100 neurona. Zadnji sloj je izlazni te sadrži `CLASS_COUNT=75` neurona.

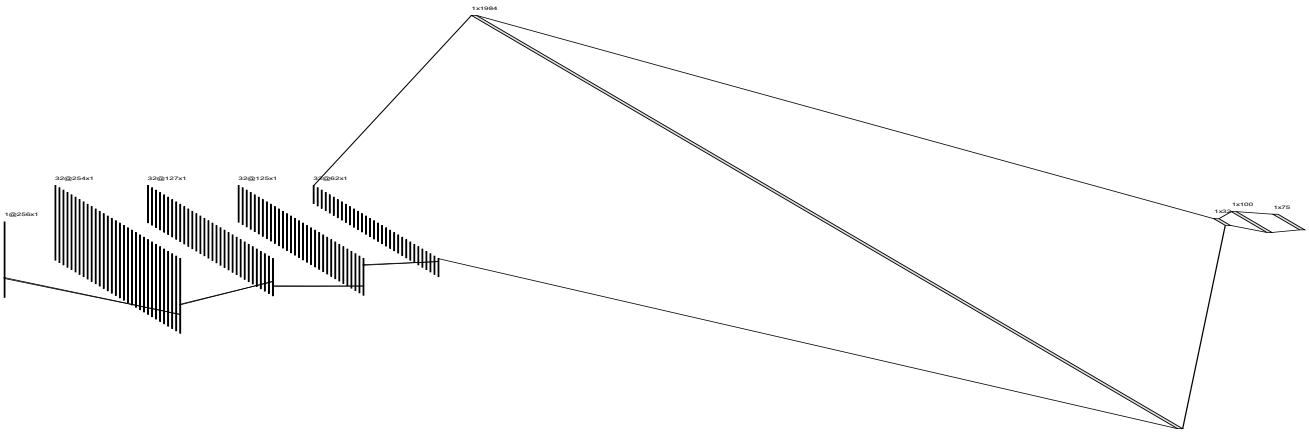
```
ann_model_small = keras.Sequential([
      layers.Dense(100, activation="relu", input_shape=(
            input_nodes,)),
      layers.Dense(64, activation="relu"),
      layers.Dense(32, activation="relu"),
      layers.Dense(100, activation="relu"),
      layers.Dense(CLASS_COUNT, activation="softmax"),
])
```

Prilikom konstrukcije modela `keras` knjižnicom dovoljno je eksplicitno zadati dimenzije ulaza samo u prvi sloj mreže. U slučaju da mreža uči klasificirati pomoću histograma i maksimalnih 10 DCT koeficijenata, argument `input_nodes` iznosit će $256 + 10 \cdot 2 = 276$. Ovaj broj je potrebno omotati u n-torku koja ima onoliko članova koliko ulazni podaci imaju dimenzija, u ovom slučaju podaci su jednodimenzionalni.

2. Model *ANN2* je nešto kompliciranije arhitekture, prikazan je isječkom koda u nastavku. Budući da je ovaj model složeniji, uvodi se sloj ispadanja (engl. *Dropout layer*). Mreža je naizmjenično građena od potpuno povezanih slojeva i slojeva ispadanja, a broj neurona kao i stopa ispadanja variraju. Uobičajeni odabir broja neurona je potencija broja 2, stoga su konstruirani slojevi s 2^{10} i 2^{11} neurona.

```
ann_model_big = keras.Sequential([
layers.Dense(1024, input_shape=(input_nodes,),
      activation="relu"),
      layers.Dropout(0.1),
      layers.Dense(1024, activation="relu"),
      layers.Dropout(0.5),
      layers.Dense(2048, activation="relu"),
      layers.Dropout(0.2),
      layers.Dense(1024, activation="relu"),
      layers.Dropout(0.5),
      layers.Dense(CLASS_COUNT, activation='softmax'),
])
```

3. Model *CNN* prikazan je na slici (Slika 5.1). Konvolucija se vrši jezgrom veličine 3 kroz 32 kanala korakom 1 bez nadopunjavanja. Sažimanje maksimumom (engl. *Max Pooling*) koristi jezgru veličine 2 kroz sve kanale. Na ulazni sloj primjenjuje se operacija konvolucije, zatim sažimanje maksimumom, ponovno konvolucija pa sažimanje te se izlaz izravna (engl. *Flatten*). Izravnati sloj se sastoji od 1984 neurona, a nastavlja se na još dva potpuno povezana sloja s 32 i 100 neurona.



Slika 5.1 Slika prikazuje arhitekturu modela CNN. S lijeva na desno prikazane su operacije konvolucija, sažimanje maksimumom, konvolucija, sažimanje maksimumom, izravnanje, dva potpuno povezana sloja te izlazni sloj. Skica je nacrtana uz pomoć alata NN-SVG [9].

5.5. Metrike

Korištene su četiri metrike za ocjenjivanje modela nad zatvorenim skupom: točnost (engl. *accuracy*), preciznost (engl. *precision*), odziv (engl. *recall*) i F1 ocjena (engl. *F1-score*). Sve metrike iskazuju odnose između četiri tipa rezultata. Rezultat klasifikacije jednog primjera može biti:

- TP ili istinski pozitivan (engl. *true positive*).
- FP ili lažno pozitivan (engl. *false positive*).
- TN ili istinski negativan (engl. *true negative*).
- FN ili lažno negativan (engl. *false negative*).

Rezultat svrstavamo u jednu od četiri kategorije ovisno o tome poklapaju li se stvarni i predviđeni razred prema tablici zabune (Tablica 5.4).

Tablica 5.4 Tablica zabune (engl. *confusion table*) daje pregled četiri kategorije u koju svrstavamo rezultate klasfikacije. Dijagonala je istaknuta jer su to ispravno klasificirani primjeri.

		y_predviđeni	
		1	0
y_stvarni	1	TP	FN
	0	FP	TN

Na temelju tih rezultata računaju se metrike točnost (18), preciznost (19), odziv (20) i F1-ocjena (21).

$$\mathrm{acc} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}} \tag{18}$$

$$\mathrm{prec} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{19}$$

$$\mathrm{rec} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{20}$$

$$\mathrm{F1} = 2 \cdot \frac{\mathrm{prec} \cdot \mathrm{rec}}{\mathrm{prec} + \mathrm{rec}} \tag{21}$$

Kod višeklasne klasifikacije konstruiramo tablicu onoliko puta koliko ima razreda. U svakoj iteraciji jedan razred odabiremo kao pozitivan, a svi ostali su negativni. Nakon tog koraka, razlikujemo mikro i makro način pronalaska ukupnih metrika. Mikro način prvo objedinjuje sve tablice u jednu jednostavno zbrajajući iznose na odgovarajućim mjestima. Zatim iz združene tablice konfuzije računa metrike. Makro način računa metriku za svaku klasu zasebno te prosjek tih vrijednosti. Na taj način svakoj klasi je pridana jednaka važnost, bez obzira na broj primjera. Makro mjere koristimo kada je osigurana jednaka distribucija klasa, kao u ovom radu.

Tablicu zabune možemo proširiti u matricu (engl. *confusion matrix*) koja će prikazivati sve klase. Na y-osi su ponovno stvarne oznake, a na x-osi predviđene oznake. Umjesto pozitivne i negativne klase sada imamo popis svih klasa. Što više primjera ima na dijagonali matrice, to ćemo imati bolje rezultate. TP, FP, FN i TN primjere više ne možemo direktno iščitati iz matrice jer ovise o promatranoj klasi. Obično normaliziramo stupce tako da se broj primjera za svaku predviđenu oznaka sumira na 1. U tablici (Tablica 5.5) je dan primjer matrice konfuzije za tri razreda.

Tablica 5.5 Primjer matrice konfuzije za klasifkaciju nad tri razreda. Dijagonala odgovara TP i TN primjerima dok su zasivljeni primjeri FP ili FN.

		y_predviđeni		
		C1	C2	C3
y_stvarni	C1	0.9	0.1	0.1
	C2	0.1	0.8	0.1
	C3	0	0.1	0.8

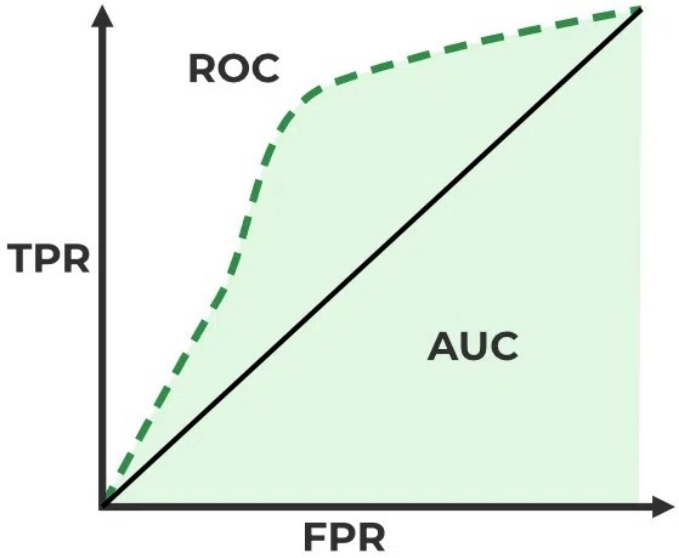
5.5.1. AUROC

U literaturi se klasifikacija nad otvorenim skupom razreda često ocjenjuje površinom ispod ROC krivulje pa je ta metrika dodatno korištena za ocjenjivanje ovog problema. Da bismo definirali AUROC (engl. *Area Under Receiver Operating Curve*) prvo trebamo definirati pomoćnu metriku FPR (engl. *False Positive Rate*).

FPR je metrika analogna odzivu koji se još naziva TPR (engl. *True Positive Rate*). Ako odziv mjeri udio istinski pozitivnih primjera u svim koji su trebali biti proglašeni pozitivnima, onda FPR mjeri udio lažno pozitivnih u svima koji su trebali biti proglašeni negativnima (22). Specifičnost je mjera komplementarna TPR-u. Mjeri koliko istinski negativnih primjera ima u svima koji su trebali biti proglašeni negativnima .

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{TN} + \mathrm{FP}} \tag{22}$$

Krivulja ROC prikazuje odnos TPR-a i FPR-a kroz različite pragove klasifikacije. Ako snizimo prag, više primjera bit će klasificirano kao pozitivno pa se povećavaju FP i TP. Vrijedi i obrnuto. Želimo dobiti što veći TPR uz što manji FPR. AUROC mjeri površinu ispod ROC krivulje. Poželjan je što veći AUROC iznos. Na slici (Slika 5.2) možemo vidjeti prikaz ROC i AUROC, tj. AUC metrika. Puna linija $x = y$ predstavlja nasumični klasifikator, onaj koji s jednakom vjerojatnošću predviđa svaku klasu u svakom koraku. Ukoliko je AUROC većeg iznosa od površine ispod te linije, tj. 0.5, znači da je konstruirani model bolji od nasumičnog pogađanja.



Slika 5.2 Prikaz ROC krivulje i AUC površine ispod krivulje. Slika je preuzeta sa stranice [24].

6. Rezultati i diskusija

6.1. Zatvoreni skup razreda

U prvom dijelu rada ocijenjena su tri modela na zatvorenom skupu podataka kroz jednu epohu. Isprobane su 4 kombinacije značajki: samo histogrami, histogrami i maksimalni DFT koeficijenti, histogrami i maksimalni DCT koeficijenti te histogrami i maksimalni DFT i DCT koeficijenti. Korišten je stalan broj koeficijenata. Za fragmente od 4096 B uzeto je 30 DFT i 50 DCT koeficijenata, a za fragmente od 512 B je uzeto 20 DFT i 30 DCT koeficijenata. Razlika u broju koeficijenata opravdana je time da su DFT koeficijenti više informativni od DCT-a pa ih treba izdvojiti manje kako se model ne bi prenaučio. Također, fragmenti od 512 B imaju više informacija koncentriranih u manjem broju koeficijenata jer čine kraći ulazni niz. Rezultati su dani tablicom (Tablica 6.1).

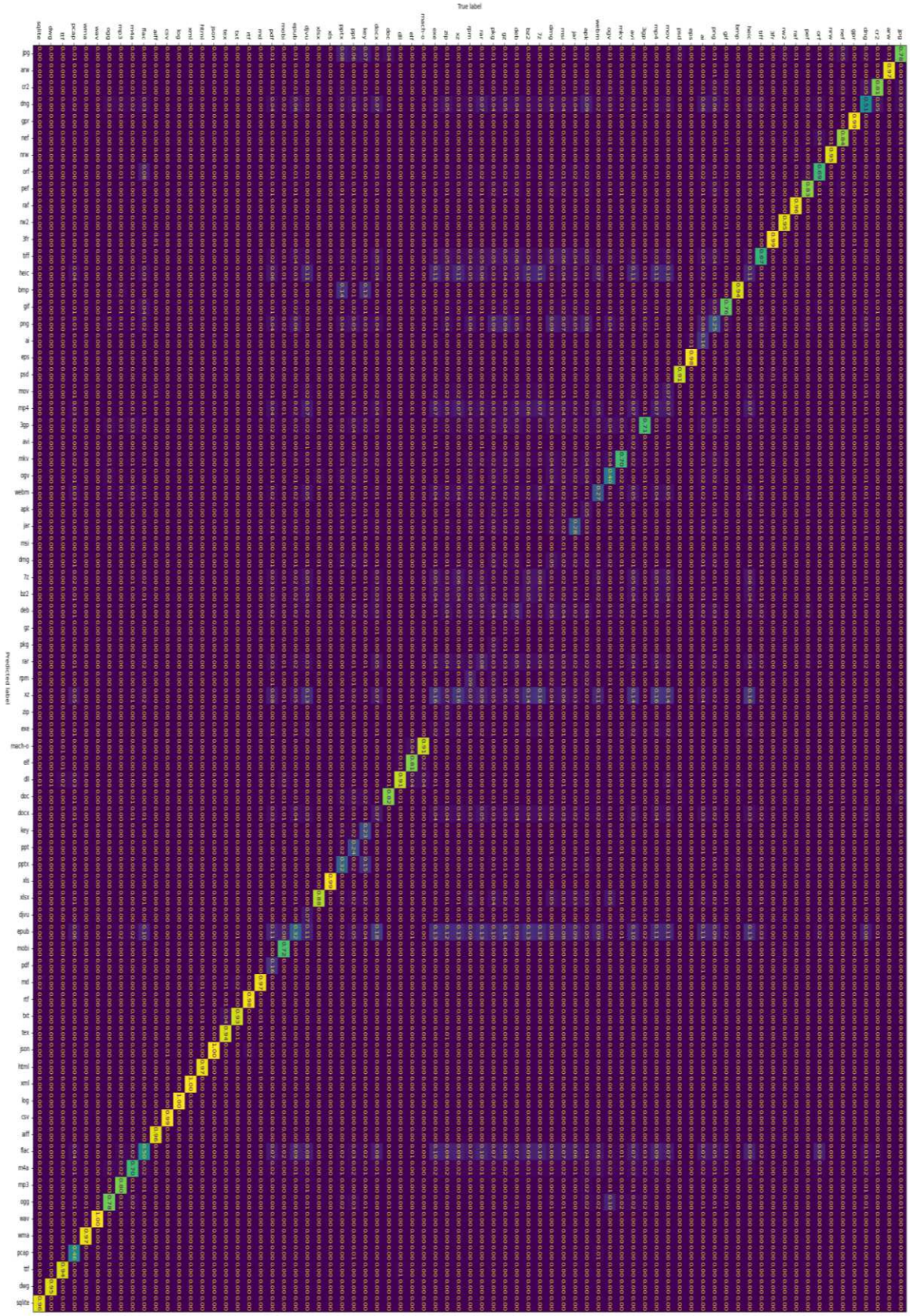
Najbolje rezultate postiže model ANN2 u kombinaciji s histogramima, točnost od 0,6814 i F1 ocjenu od 0,6774. Na slici (Slika 6.1) je dana matrica zabune za model ANN2 na fragmentima od 512 B radi usporedbe s otvorenim skupom. Uočavamo da se za umjetne mreže ne postiže nikakvo poboljšanje korištenjem dodatnih značajki. Model CNN postiže poboljšanje uvođenjem značajki DFT i DFT + DCT na fragmentima od 4096 B. Jasno je da svi modeli postižu bolje rezultate na većoj, informativnijoj veličini bloka. Općenito je najbolji model ANN2 zatim ANN1 te CNN.

Sljedeće je bilo potrebno odrediti može li neki broj koeficijenata poboljšati rezultate modela s histogramima. Za sve modele i obje veličine fragmenta, DCT i DFT histogrami su poprimali vrijednosti 10, 20, 30, 50 i 100. Na grafu (Slika 6.2) je prikazana točnost modela u ovisnosti o broju koeficijenata u odnosu na histograme.

Najveće poboljšanje možemo uočiti kod CNN-a. Konvolucijske mreže generalno imaju kompliciranju arhitekturu pa imaju koristi od više informacija. Isto vrijedi za veće fragmente, na skupu od 4096 B možemo imati više značajki prije nego što uvedemo šum ili koreliranost.

Tablica 6.1 Ocjene svih modela i značajki kroz jednu epohu na zatvorenom skupu razreda.

			metrika			
veličina bloka	model	značajke	točnost	preciznost	odziv	F1
4096B		histogrami	0,5678	0,5671	0,5678	0,5513
		hist + DFT	0,5227	0,5294	0,5227	0,5058
		hist + DCT	0,5149	0,5242	0,5149	0,4977
	ANN1	hist + DFT + DCT	0,5105	0,5171	0,5105	0,4924
		histogrami	0,6814	0,7105	0,6814	0,6774
		hist + DFT	0,6475	0,6698	0,6475	0,6390
		hist + DCT	0,6233	0,6555	0,6233	0,6168
	ANN2	hist + DFT + DCT	0,6102	0,6425	0,6102	0,6005
		histogrami	0,5551	0,5491	0,5551	0,5374
		hist + DFT	0,5713	0,5798	0,5713	0,5564
		hist + DCT	0,5286	0,5349	0,5286	0,5089
	CNN	hist + DFT + DCT	0,5851	0,5815	0,5851	0,5693
512B		histogrami	0,5039	0,5001	0,5039	0,4825
		hist + DFT	0,4594	0,4681	0,4599	0,4436
		hist + DCT	0,4495	0,4498	0,4495	0,4289
	ANN1	hist + DFT + DCT	0,4445	0,4837	0,4445	0,4273
		histogrami	0,5803	0,6051	0,5803	0,5664
		hist + DFT	0,5171	0,5524	0,5171	0,5017
		hist + DCT	0,5118	0,5427	0,5118	0,4986
	ANN2	hist + DFT + DCT	0,5062	0,5374	0,5062	0,4896
		histogrami	0,4753	0,4769	0,4753	0,4493
		hist + DFT	0,4564	0,4649	0,4564	0,4373
		hist + DCT	0,4872	0,4837	0,4872	0,4621
	CNN	hist + DFT + DCT	0,4532	0,4507	0,4532	0,4336



Slika 6.1 Matrica zabune za model ANN2 na blokovima od 512 B uz histograme na zatvorenom skupu.

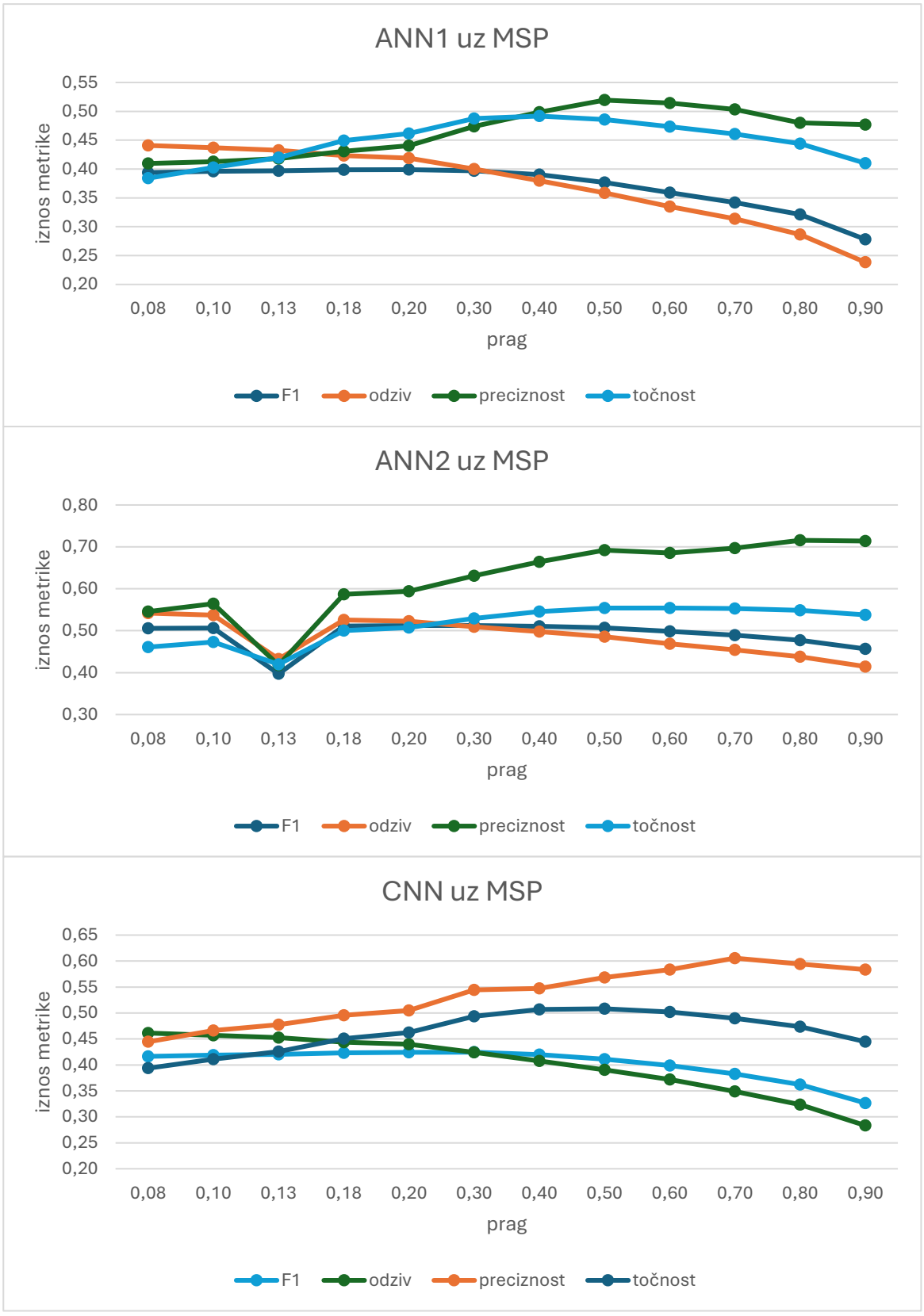
Slika 6.2 Prikaz kako broj DCT i DFT koeficijenata utječe na točnost modela.

6.2. Otvoreni skup razreda

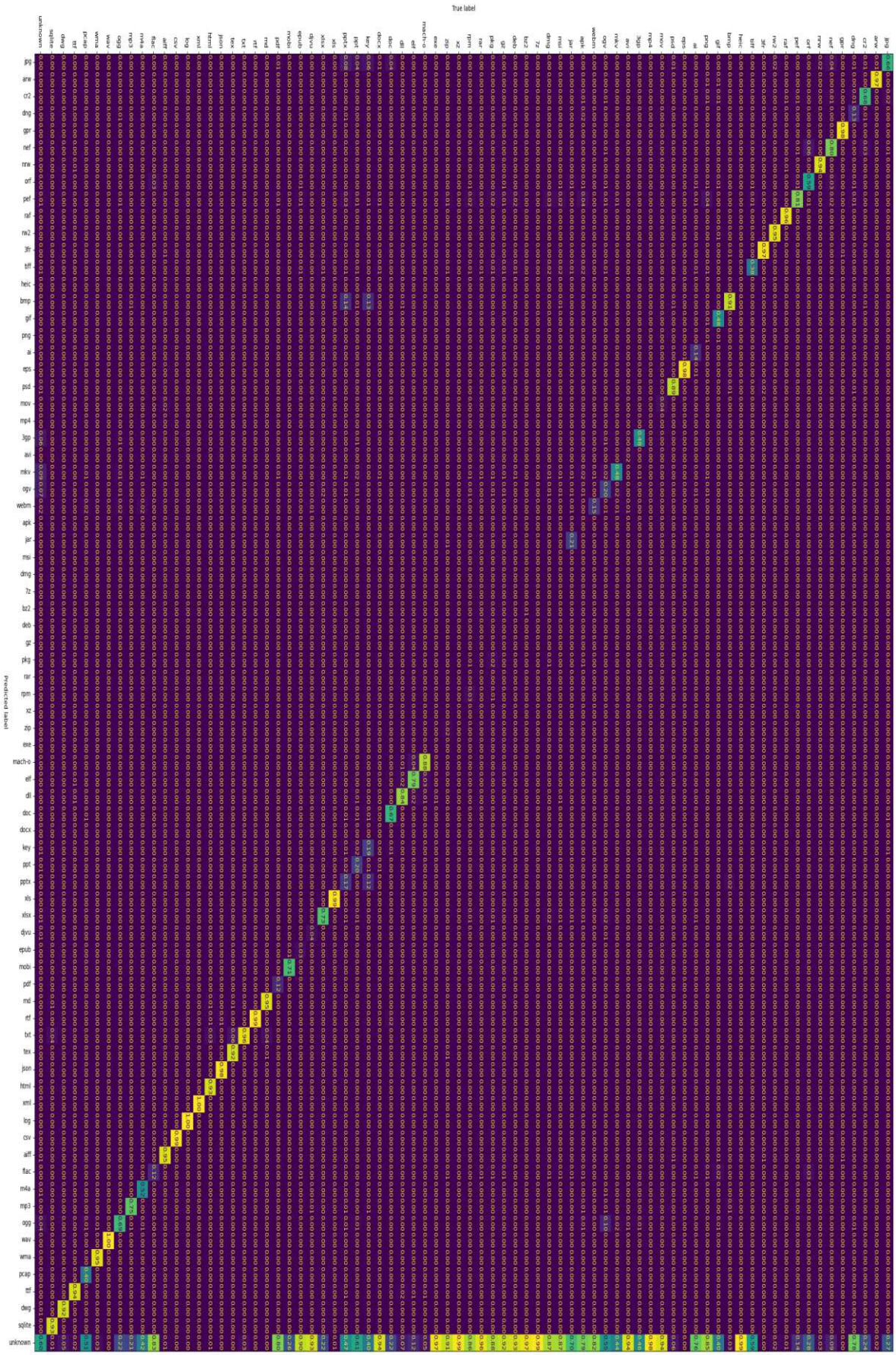
Na otvorenom skupu podataka kao značajka su u svim slučajevima korišteni samo histogrami te veličina fragmenta od 512 B.

6.2.1. MSP

Za metodu maksimalne softmax vjerojatnosti ispitane su metrike uz variranje praga. Rezultati su prikazani na slici (Slika 6.3). Vidimo da je F1 vrijednost za prag manji od 50% za sve modele poprilično konstantna. Između tih pragova biramo onaj koji daje najvišu točnost. Za ANN1 to je 0,4918 uz prag od 40%, za ANN2 točnost je 0,5541 uz prag od 50%, za CNN točnost je 0,5067 uz prag od 50%. Dana je matrica zabune za model najveće točnosti – ANN2 i prag 50% na slici (Slika 6.4). Vidljiva je jaka dijagonala koja predstavlja točno klasificirane primjere. Možemo primijetiti da su datoteke koje sadrže kompresirani sadržaj ponovno slabije klasificirane, dijagonala je tamnija na srednjem dijelu. Zadnji redak i stupac su nepoznata klasa. Uz prag od 50% model je puno poznatih primjera proglasio nepoznatima što se očituje kao svjetliji zadnji stupac.



Slika 6.3 Prikaz metrika ovisno o pragu za tehniku MSP na otvorenom skupu.



Slika 6.4 Matrica zabune za model ANN2 i prag 50% uz MSP.

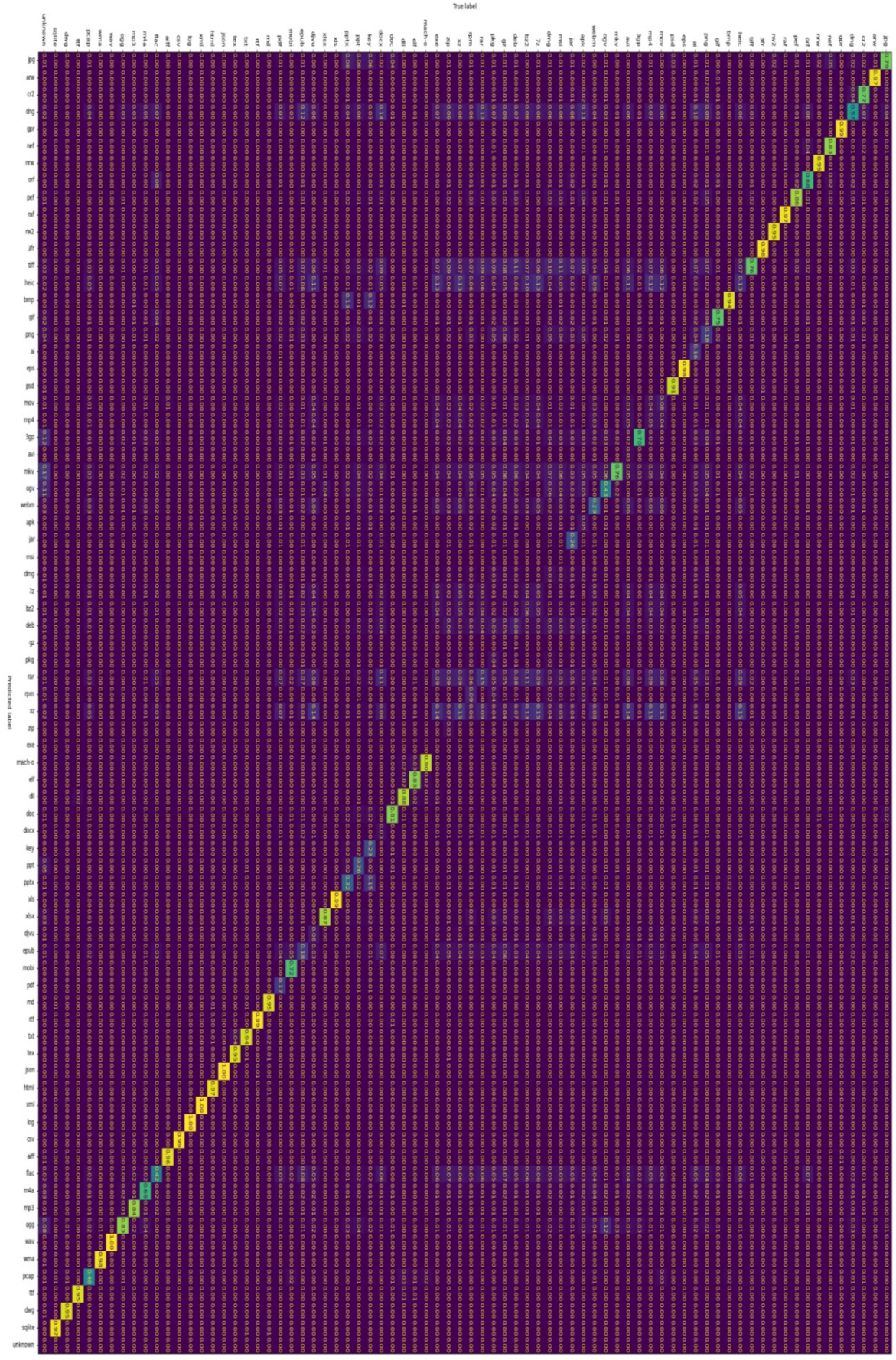
6.2.2. MLS

Rezultati MLS-a su dani u tablici (Tablica 6.2) za percentil validacijskog skupa 0.5. MLS tehnika nije bila uspješna, bez obzira na prag mreže nisu naučile prepoznavati nepoznat razred. U matrici zabune za model ANN2 i prag =-5,0864 (Slika 6.5) možemo vidjeti posljednji stupac koji predstavlja nepoznatu klasu. Stupac je gotovo potpuno mračan što znači da gotovo niti jedan primjer nije bio uvršten u nepoznat razred. Isto vrijedi za sve modele i pragove.

Tablica 6.2 Prikaz rezultata tehnike MLS za sve modele za prag koji odgovara percentilu 0.5.

model	**prag**	**točnost**	**preciznost**	**odziv**	**F1**
ANN1	1,2371	0,379933	0,430572	0,492795	0,435751
ANN2	-5,0864	0,441801	0,517919	0,573042	0,515514
CNN	0,7635	0,374522	0,418973	0,485777	0,433483

Neuspješnost možemo objasniti razlikama u izlazu mreže. Logit vrijednost nije ograničena pa je jako teško prepoznati koje vrijednosti predstavljaju sigurnu klasifikaciju a koje nesigurnu. Iako je prag odabran na temelju skupa za validaciju, ne mora značiti da će biti pogodan za skup za testiranje jer čak i mala razlika u ulazima istih razreda može rezultirati drastično različitim izlazima. Relativno visoki rezultati F1 metrike ostvareni su isključivo zbog točne klasifikacije poznatih razreda.



Slika 6.5 Matrica zabune za model ANN2 uz tehniku MLS.

6.2.3. RPL i ARPL

Algoritmi RPL i ARPL su ocijenjeni metrikama AUROC i točnošću, po uzoru na literaturu [11]. Uočavamo visoke rezultate AUROC-a dok je točnost značajno niža. Iako je AUROC inače pouzdanija metrika jer prolazi kroz sve pragove, razlika u ocjenama je od većeg značaja i vjerojatno ukazuje na nepravilnost u radu. Visoka AUROC vrijednost, a niska preciznost na balansiranom skupu podataka obično znači da klasifikator postiže dobre rezultate na pozitivnoj klasi, a loše na negativnoj.

Tablica 6.3 Prikaz točnosti i AUROC metrike za sve modele uz algoritme RPL i ARPL.

algoritam	model	AUROC	točnost
RPL	ANN1	0,9466	0,5644
	ANN2	0,9518	0,5774
	CNN	0,9296	0,4918
ARPL	ANN1	0,9436	0,6084
	ANN2	0,9518	0,6084
	CNN	0,9250	0,5084

6.3. Usporedba

U tablici (Tablica 6.4) su uspoređene sve tehnike učenja na fragmentima 512 B. Odabrani su hiperparametri – broj koeficijenata, prag, epohe – koji daju najveću točnost. Zelenom bojom označene su sve pozitivne razlike, dakle slučajevi kada je točnost veća od točnosti referentnog modela.

Promatrajući značajke na zatvorenom skupu razreda, vidimo da uvođenje DCT i DFT koeficijenata utječe na točnost neznačajno. Najveće poboljšanje točnosti je postignuto za model CNN i 10 DCT koeficijenata, točnost iznosi 0,5135 što je za 3.8% bolje od točnosti nad histogramima. No, i dalje je lošije od najboljeg modela – ANN2 nad histogramima postiže točnost od 0,5803. Ovo možemo protumačiti kompleksnošću modela, odnosno veličinom skupa podataka. Uvođenjem novih značajki na manjim fragmentima od 512 B lako dolazimo do prenaučenosti. Iz tog razloga, na otvorenom skupu podataka ispitivani su samo histogrami.

Algoritam MSP pogoršava rezultate 1-2% kod umjetnih mreža. Zanimljivo je da se kod CNN-a očituje povećanje točnosti 3,28%. Ovo svjedoči tome da su algoritmi koji se oslanjaju na prag nepouzdani. Mreža još uvijek ne zna klasificirati primjere koje je pogrešno klasificirala na zatvorenom skupu, no ako dobro namjestimo prag, većina primjera iz otvorenog skupa bit će svrstana u nepoznat razred. Budući da na testnom skupu podataka ima otprilike jednako poznatih i nepoznatih primjera, to će značajno poboljšati točnost. Međutim, ne možemo znati pravi udio nepoznatih razreda, dakle ne možemo unaprijed odrediti optimalni prag, a čak i ako ga pronađemo ne postoji garancija da se u budućnosti taj udio neće promijeniti. Zato ocijene MSP-a nisu toliko značajne u određivanju korisnosti ovih algoritama.

Sljedeća ispitana tehnika je MLS. MLS također ovisi o pragu, ali on se određuje prema skupu za validaciju pa je nešto pouzdaniji. Ovaj algoritam univerzalno daje najlošije rezultate, i prema točnosti, i prema broju identificiranih nepoznatih razreda. Razlog tomu može biti to da ne radi s normaliziranim vrijednostima, nego neograničenim logitima. Ako skup za validaciju ne predstavlja testni skup dovoljno dobro, rezultati će biti vrlo loši. U ovom slučaju imamo puno nepoznatih razreda i primjera u testnom skupu. MLS ima potencijala za dobrim rezultatima na otvorenom skupu, u mnogim radovima nadmašuje MSP. No, krucijalno je izvući dobar prag.

Algoritmi recipročnih točaka u većini slučajeva daju bolju točnost nego na zatvorenom skupu. Ovaj rezultat ne iznenađuje jer je promijenjen način na koji modeli klasificiraju. Uvedena je nova funkcija gubitka koja uvodi elemente klasteriranja. Za vjerodostojnu usporedbu ovih algoritama sa zatvorenim skupom podataka trebalo bi ponoviti treniranje mreža s novom funkcijom gubitka. Daleko najbolju točnost od 0,6084 postižu modeli ANN1 i ANN2 uz algoritam ARPL. To čini poboljšanje od 10,45% u odnosu na zatvoreni skup za ANN1. ARPL konzistentno rezultira boljim metrikama od RPL-a. Ovo ponašanje je očekivano jer ARPL predstavlja nadogradnju na stariji algoritam.

Tablica 6.4 Prikazan je iznos točnosti za sve modele i algoritme te razlika u točnosti u odnosu na referentni model.

model	skup razreda	algoritam/značajke	točnost	$\Delta_{\text{točnost}}$
	zatvoreni	hist	0,5039	-
	zatvoreni	hist+10DFT	0,5195	0,0156
	zatvoreni	hist+10DCT	0,5208	0,0169
	otvoreni	MSP, α=0.4	0,4918	-0,0121
	otvoreni	MLS, α=0.5	0,3799	-0,1240
	otvoreni	RPL	0,5644	0,0605
ANN1	otvoreni	ARPL	0,6084	0,1045
	zatvoreni	hist	0,5803	-
	zatvoreni	hist+10DFT	0,5689	-0,0114
	zatvoreni	hist+10DCT	0,5689	-0,0114
	otvoreni	MSP, α=0.5	0,5541	-0,0263
	otvoreni	MLS, α=0.5	0,4418	-0,1385
	otvoreni	RPL	0,5774	-0,0029
ANN2	otvoreni	ARPL	0,6084	0,0280
	zatvoreni	hist	0,4753	-
	zatvoreni	hist+10DFT	0,5123	0,0370
	zatvoreni	hist+10DCT	0,5135	0,0382
	otvoreni	MSP, α=0.5	0,5081	0,0328
	otvoreni	MLS, α=0.5	0,3745	-0,1008
	otvoreni	RPL	0,4918	0,0165
CNN	otvoreni	ARPL	0,5084	0,0331

Otvoreni skup razreda nije puno istražen u sklopu klasifikacije fragmenata pa ne postoji puno referentnih točaka s kojima bi se mogli usporediti rezultati. U radu [19] je konstruiran *ByteRCNN*, model rekurentnih konvolucijskih mreža. Klasifikacija je rađena na istom FiFTy skupu podataka na fragmentima od 512 B. Prosječna točnost ByteRCNN-a iznosi 68.5%, a FiFTy modela 58.1%. Za usporedbu, najveću točnost u ovom radu postiže ANN2_ARPL i iznosi 60,8%.

Zaključak

Cilj rada bio je ispitati različite tehnike učenja na otvorenom skupu razreda, pronaći najbolju tehniku i usporediti te rezultate s rezultatima modela na zatvorenom skupu. U prvom dijelu rada tri modela neuronskih mreža su ocijenjena na zatvorenom skupu. Tražila se najbolja kombinacija značajki među histogramima, Fourierovim i kosinusnim koeficijentima. Zatim su dva ANN i jedan CNN model u kombinaciji s histogramima ocjenjivani na otvorenom skupu podataka. Isprobane su četiri tehnike: MSP, logits score, RPL i ARPL.

Rezultati rada su obećavajući, ukazuju na iskoristivost algoritama RPL i ARPL. Uvođenjem nepoznatih razreda performanse modela se ne pogoršavaju značajno, a u nekim slučajevima točnost je veća. Rezultati su značajni za područje računalne forenzike jer je čest slučaj da se alati za klasifikaciju fragmenata susreću s nepoznatim datotekama bez da to prepoznaju. Korištenjem nekih od ovih algoritama ti alati bi se mogli poboljšati.

U budućem radu valjalo bi isprobati još tehnika učenja na otvorenom skupu podataka ili isprobati rješavanje nekim srodnim načinom, primjerice detekcijom podataka izvan distribucije (OoD, engl. *Out of Distribution*). Dodatno, istraživanje bi se moglo ponoviti s drugačijim modelima i značajkama s fokusom na konvolucijske neuronske mreže. Svakako bi trebalo izvršiti još istraživanja u području klasifikacije fragmenata datoteka nad otvorenim skupom kako bi se uspostavila referentna točka.

Literatura

[1] Mittal, G., Korus, P., Memon, N. *FiFTy: Large-Scale File Fragment Type Identification Using Convolutional Neural Networks*, IEEE Transactions on Information Forensics and Security, br. 16 (2021), str. 28-41.

[2] Pedregosa, F. i suradnici., *Scikit-learn: Machine Learning in Python*, JMLR 12, str. 2825-2830, 2011.

[3] Špoljar, J., *Arhitektura Supeka.* Poveznica: https://wiki.srce.hr/display/NR/Arhitektura+Supeka; pristupljeno: 2024.

[4] Sun Microsystems, *Sun Grid Engine Reference*. Poveznica: https://docs.oracle.com/cd/E19279-01/820-3257-12/n1ge.html; pristupljeno: veljača 2024.

[5] Cooley , J., Tukey, J. *An algorithm for the machine computation of the complex Fourier series*, Mathematics of Computation, br. 19, str. 297-301, 1965.

[6] Allen, J. B., Rabiner, L. R. *A unified approach to short-time Fourier analysis and synthesis*, Proceedings of the IEEE, br. 65, br. 11, str. 1558-1564, 1977.

[7] Smith, J. O. *Introduction to Digital Filters with Audio Applications*, 2007.

[8] Liu, W., Chen, B., Swartz, R. (2013). *Investigation of Time Series Representations and Similarity Measures for Structural Damage Pattern Recognition*, The Scientific World Journal, 2013.

[9] Lenail, A. *NN-SVG*. Poveznica: https://alexlenail.me/NN-SVG/LeNet.html; pristupljeno: svibanj 2024.

[10] *Optimizers Adam*. Poveznica: https://keras.io/api/optimizers/adam/; pristupljeno: 2024.

[11] Chen, G., Peng, P., Wang, X., Tian, Y. *Adversarial Reciprocal Points Learning for Open Set Recognition*, IEEE Transactions on Pattern Analysis and Machine Intelligence, (2021.), str. 1

[12] Sadeghi, N., Fahiminia, M., Teimouri, M. *Dataset for file fragment classification of video file formats*. BMC Res Notes 13, 213 (2020).

[13] Fakouri, R., Teimouri, M. *Dataset for file fragment classification of image file formats*, BMC Res Notes 12, 774 (2019).

[14] Khodadadi, A., Teimouri, M. *Dataset for file fragment classification of audio file formats*, BMC Res Notes 12, 819 (2019).

[15] Scheirer, W. J., Rocha, A., Sapkota, A., Boult, T. E. *Towards open set recognition*, IEEE TPAMI (2013).

[16] Bendale, A., Boult, T. E. *Towards Open Set Deep Networks*, arXiv, 2015.

[17] Wang, Z., Xu, Q., Yang, Z., He, Y., Cao, X., Huang, Q. *OpenAUC: Towards AUC-Oriented Open-Set Recognition*, 2023.

[18] Vaze, S., Han, K., Vedaldi, A., Zisserman, A. *Open-Set Recognition: A Good Closed-Set Classifier is All You Need?*, ICLR, 2022.

[19] Skračić, K., Petrović, J., Pale, P. *ByteRCNN: Enhancing File Fragment Type Identification with Recurrent and Convolutional Neural Networks*, IEEE, 2017.

[20] Skračić, K., Petrović, J., Pale, P. *Classification of Low- and High-Entropy File Fragments Using Randomness Measures and Discrete Fourier Transform Coefficients*, Vietnam Journal of Computer Science, Vol. 10, br. 4 (2023), str. 433–462.

[21] Chen, G., Qiao, L., Shi, Y., Peng, P., Li, J., Huang, T., Pu, S., Tian, Y. *Learning Open Set Network with Discriminative Reciprocal Points*, arXiv, 2020.

[22] Neal, L., Olson, M., Fern, X., Wong, W.-K., Li, F. *Open Set Learning with Counterfactual Images*, 2020.

[23] Hamdi, E., Rady, S., Aref, M. *A Convolutional Neural Network Model for Emotion Detection from Tweets*, Proceedings of the International Conference on Advanced Intelligent Systems and Informatics (2018).

[24] *AUC ROC Curve in Machine Learning* (2024). Poveznica: https://www.geeksforgeeks.org/auc-roc-curve/; pristupljeno: lipanj 2024.

Sažetak

Naslov

Klasifikacija fragmenata datoteka nad otvorenim skupom razreda

Sažetak

Problem klasifikacije fragmenata datoteka jedan je od ključnih problema kojim se bavi računalna forenzika. Zbog složenosti problema koriste se tehnike strojnog učenja. U pravim forenzičkim slučajevima često nailazimo na razrede datoteka na kojima mreže nisu učene. U ovom radu proučavaju se načini klasifikacije na otvorenom skupu razreda. Opisana su i implementirana četiri algoritma. Algoritmi su uspoređeni s rezultatima na zatvorenom skupu podataka kako bi se ocijenila njihova uporabljivost.

Ključne riječi

fragmenti datoteka, otvoreni skup razreda, klasifikacija, računalna forenzika, neuronske mreže

Summary

Title

Open-set File Fragment Classification

Summary

The problem of file fragment classification is one of the key issues addressed by computer forensics. Due to the complexity of the problem, machine learning techniques are used. In real forensic cases, we often encounter file classes on which the networks have not been trained. This paper studies methods for classification in an open set of classes. Four algorithms are described and implemented. The algorithms are compared with results on a closed data set to evaluate their usability.

Keywords

file fragments, open-set, classification, computer forensics, neural networks

Skraćenice

ANN	*Artifficial Neural Network*	umjetna neuronska mreža
API	*Application Programming Interface*	aplikacijsko programsko sučelje
ARPL	*Adverserial Reciprocal Point Learning*	suprotstavljeno učenje recipročne točke
AUROC	*Area Under ROC*	površina ispod ROC krivulje
CNN	*Convolutional Neural Network*	konvolucijska neuronska mreža
DCT	*Discrete Cosine Transform*	diskretna kosinusna transformacija
DFT	*Discrete Fourier Transform*	diskretna Fourierova transformacija
FCT	*Fast Cosine Transform*	brza kosinusna transformacija
FFT	*Fast Fourier Transform*	brza Fourierova transformacija
MLS	*Maximum Logit Score*	maksimalna logit vrijednost
MSE	*Mean Squared Error*	srednja kvadratna pogreška
MSP	*Maximum Softmax Probability*	maksimalna vjerojatnost funkcije Softmax
OoD	*Out of Distribution*	izvan distribucije
OSR	*Open Set Recognition*	prepoznavanje na otvorenom skupu razreda
ReLU	*Rectified Linear Unit*	funkcija zglobnice
ROC	*Reciever Operating Curve*	ROC krivulja
RPL	R*eciprocal Point Learning*	učenje recipročne točke
SGE	*Sun Grid Engine*	softver *Sun Grid Engine*