Klasifikacija ribe primjenom metoda strojnog učenja i strojnog vida

Cepernić, Marijo

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Marijo Cepernić

Zagreb, 2020.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Mentor:

Prof. dr. sc. Bojan Jerbić, dipl. ing.

Student:

Marijo Cepernić

Zagreb, 2020.

Izjavljujem da sam ovaj rad izradio samostalno koristeći znanja stečena tijekom studija i navedenu literaturu.

Zahvaljujem se svom mentoru, profesoru Bojanu Jerbiću kao i asistentu Filipu Šuligoju koji su mi svojim iznimnim znanjem pretočenim u upute i smjernice pomogli kod realizacije Rada. Također se zahvaljujem Ivanu Škariću bez čije nesebične pomoći ovaj Rad ne bi bio moguć.

Posebnu zahvalu upućujem obitelji i prijateljima na pruženoj potpori i razumijevanju.

Marijo Cepernić

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za završne ispite studija strojarstva za smjerove:
proizvodno inženjerstvo, računalno inženjerstvo, industrijsko inženjerstvo i menadžment, inženjerstvo materijala i mehatronika i robotika

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum	Prilog
Klasa:	
Ur.broj:	

ZAVRŠNI ZADATAK

Student: Marijo Cepernić Mat. br.: 0035210435

Naslov rada na hrvatskom jeziku: **Klasifikacija ribe primjenom metoda strojnog učenja i strojnog vida**

Naslov rada na engleskom jeziku: **Classification of fish using machine learning and machine vision methods**

Opis zadatka:

Klasifikacija spada u kategoriju nadziranog strojnog učenja gdje su osigurani ulazni podatci, najčešće prikupljeni putem strojnog vida ili fotografija. U strojnom učenju, s naglaskom na primjenu strojnog vida, klasifikacija je problem identificiranja određene kategorije objekta, koristeći slike objekata čija je pripadnost kategorijama poznata.

Klasifikacija životinjskih vrsta i podvrsta je specifičan problem kod primjene metoda strojnog vida zbog nesagledive genetske raznolikosti koja značajno usložava problem klasifikacije. Stoga se predlaže tražiti fleksibilnost rješenja primjenom metoda strojnog učenja. Zadatak ovog završnog rada je primijeniti metode strojnog učenja i strojnog vida za potrebe uspješne klasifikacije ribe (srdela i inćuna) koja se nalaze na pokretnoj traci u pogonu za preradu i pakiranje.

Potrebno je:

- Prikupiti set ulaznih podataka, odnosno slika srdela i inćuna iz pogona za preradu i pakiranje ribe.
- Primjenom algoritama strojnog vida automatizirati segmentaciju ribe u odnosu na pokretnu traku i druge ribe.
- Pripremiti ulazne setove podataka za strojno učenje i za testiranje rješenja.
- Proučiti mogućnosti primjene metoda strojnog učenja za potrebe klasifikacije, poput stabla odluka, neuronskih mreža i sl.
- Primijeniti odabranu metodu strojnog učenja za problem klasifikacije i analizirati dobivene rezultate.

Zadatak zadan:
28. studenog 2019.

Datum predaje rada:
1. rok: 21. veljače 2020.
2. rok (izvanredni): 1. srpnja 2020.
3. rok: 17. rujna 2020.

Predviđeni datumi obrane:
1. rok: 24.2. – 28.2.2020.

Zadatak zadao:
Prof. dr. sc. Bojan Jerbić

Predsjednik Povjerenstva:
Prof. dr. sc. Branko Bauer

SADRŽAJ

POPIS SLIKA

POPIS OZNAKA

N_p	- Broj piksela
m	- Širina slike
n	- Visina slike
N_c	- Broj boja
G_x	- Gradijent u x smjeru
G_y	- Gradijent u y smjeru
G	- Gradijent
φ	- Kut gradijenta
$d(H_1, H_2)$	- Razlika histograma
$H_{1,2,k}$	- Histogram
$H_{1,2}(i)$	- *i*-ti član u histogramu
$\bar{H}_{1,2,k}$	- Prosjek histograma
$\omega_1, \omega_2, \omega_3$	- Hrpe
$d(p, q)$	- Udaljenost u *n*-dimenzionalnim prostorima
p_i	- *i*-ta osobina *p*-tog člana
q_i	- *i*-ta osobina *q*-tog člana
$f(net)$	- Izlaz iz mreže
W_i	- *i*-ta težina
x_i	- *i*-ti ulaz u mrežu
$s(t)$	- Aktivacijska funkcija
$f(x_i, w)$	- Funkcija cilja
W	- Matrica težina
h	- Korak gradijenta
L_i	- Zavisna funkcija gubitaka
L	- Nezavisna funkcija gubitaka
β	- Hiperparametar kod regularizacije
$R(W)$	- Metoda regularizacije
K	- Broj kernela
S	- Korak kernela
F	- Dimenzija kernela
A	- Hiperparametar kod dizajniranja konvolucijskih mreža
P	- Količina nula dodanih u okvir slika
μ_β	- Prosjek hrpe
$б_\beta$	- Razlika *i*-tog člana i prosjeka hrpe
N_{kon}	- Konačni broj slika
$N_{poč}$	- Početni broj slika

SAŽETAK

U ovom Radu razmatra se primjena strojnog učenja i računalnog vida na problem klasifikacije riba. Prolazi se kroz prikupljanje baze podataka, procesiranje slika, ekstrakciju osobina i klasifikaciju. Prikupljanje baze podataka izvršava se s maksimalnom pozornosti na njenu konzistentnost i definiranost. Procesiranje slika u obzir uzima razne varijacije, šumove i neočekivane objekte koji su mogući. Klasifikacija se izvodi primjenom nekoliko metoda čiji se dizajn i parametri potpuno prilagođavaju zadatku. Kroz proces se za referencu uzima eventualna primjena sustava u industrijskoj okolini, pa se zbog toga pazi na robusnost i brzinu algoritama.

Ključne riječi: Strojno učenje, Računalni vid, Procesiranje slika, Baza podataka, Klasifikacija riba

SUMMARY

This paper discusses the application of machine learning and computer vision in the problem of fish classification. It goes through database collection, image processing, feature extraction, and classification. Database collection is performed with utmost care for its consistency and definiteness. Image processing takes into account the various variations, noises and unexpected objects that are possible. The classification is performed using several methods whose design and parameters are completely adapted to the task. Throughout the process, reference is made to the possible application of the system in an industrial environment, so the robustness and speed of the algorithms is taken into account.

Key words: Machine Learning, Computer Vision, Image Processing, Database, Fish Classification

1. UVOD

Činjenica je da su riba i riblji proizvodi neizbježan čimbenik u svakoj uravnoteženoj prehrani. Riba ne samo da obiluje OMEGA-3 masnim kiselinama, već je i bogat izvor proteina i bioaktivnih peptida što ju okarakterizira kao jednu od najzdravijih prehrambenih namirnica. Danas se smatra da prosječan čovjek na svjetskoj razini konzumira 23.7 *kg* ribe godišnje, dok je ta brojka npr. u Portugalu daleko viša: 55.3 *kg* [1].

Svjetska konzumacija ribe se u posljednjih 50 godina više nego udvostručila što zbog povećanja broja stanovništva, što zbog podizanja razine svijesti o zdravoj prehrani ali i zbog prerade ribljih proizvoda na način koji omogućuje konzumaciju bilo kad i bilo gdje po relativno pristupačnoj cijeni.

Republika Hrvatska ima razvijen lanac ulova, prerade i izvoza ribe i njenih proizvoda pa se tako Hrvatski proizvodi pronalaze na policama trgovina svih sedam kontinenata [2]. Jadransko more je vrlo zavučeni dio Mediterana, koji je dalje izoliran od oceana, pa zbog toga u Jadranu ne dolazi do snažnih struja, kombinirajući to s činjenicom da se radi o vrlo plitkom moru zaključak se nameće da se radi o vrlo osjetljivom ekosustavu. Zbog zagrijavanja mora u posljednjih nekoliko desetljeća pojavljuje se fenomen da hrvatske koćarice gotovo nikad ne ulove jednu te istu vrst sitne plave ribe već se u istom ulovu najčešće pojavljuju i srdele i inćuni u raznim omjerima. Velika količina vremena i truda potrebna je da se ručno odvoje te dvije vrste ribe. Trud i vrijeme znače dvije stvari – manja produktivnost i veći troškovi, a to se dalje reflektira i na cijenu proizvoda.

	Ulov i uzgoj (proizvodnja), t *Catches and production, t*		
	2016.		2017.
	ukupno *Total*	ukupno *Total*	ulov *Catches*
Ukupno	85 028	83 318	69 476
Ribe	81 441	79 834	66 974
Plava riba	68 339	65 335	63 173
Srdela	53 909	48 420	48 420
Inćun	8 125	10 883	10 883

Slika 1. Isječak iz tablice u dokumentu [3]

Pobliže promatrajući Sliku 1 vidi se da se količina ulovljenih inćuna u Republici povećala s 8 125t na 10 833t dok se istovremeno količina srdele smanjila sa 53 909t na 48 420t. Upravo u ovim brojkama leži veliki problem hrvatskog ribarstva a to je da se sve više inćuna lovi na lovištima srdele.

Kao što je rečeno, u (čestom) slučaju kad koćarica ne ulovi isključivo jednu vrst sitne plave ribe, ulovljena masa se dovodi u tvornicu za preradu riba gdje je potrebno realizirati sustav klasifikacije pomiješanih riba. Iako je vrlo teško prognozirati u kojem omjeru vrsta riba dolazi ulovljena masa, ona nikad ne sadrži veće od 30 – 40 % inćuna jer je u suprotnom ista neiskoristiva.

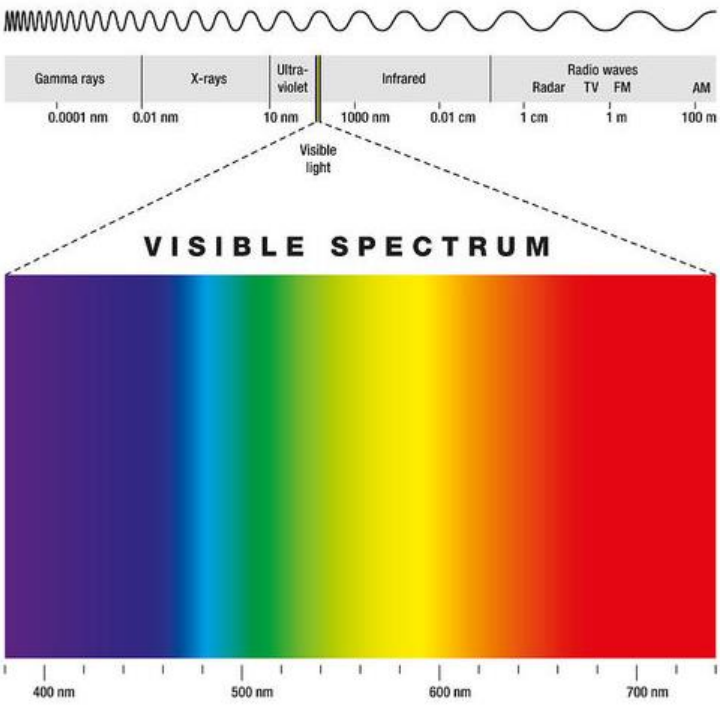
U ovom radu postavljaju se temelji sustava u kojem ribe dolaze razbacane na traci, potrebno ih je segmentirati, izbaciti nepotrebne šumove u slici, eliminirati slučajeve preklapanja i sl. Nakon navedenog pretprocesiranja slika pristupa se intezivnoj obradi s ciljem uspješne kvalifikacije u slučaju stabla odluke. U slučaju konvencionalnog strojnog učenja slike ili određene osobine na slikama se pripremaju kao adekvatan ulaz u neuronske mreže konstruirane upravo za ovu namjenu. Za daljnju preradu potrebna je isključivo jedna vrst a to su srdele. U ovom radu se ne razmatra fizička realizacija separacije, već je fokus isključivo na primjeni sustava računalnog vida čiji bi rezultati klasifikacije bili zadovoljavajući.

Baza podataka, odnosno slika, čije je prikupljanje dio ovog rada je temeljni izvor svih informacija, te se ostali izvori (volumen, masa i sl.) informacija o ribi neće razmatrati.

Da bi se to postiglo koristi se programski jezik Python sa svojim raznim bibliotekama za obradu slika kao što je OpenCV i za strojno učenje čiji su predstavnici Keras i TensorFlow. Osim navedenih biblioteka koriste se i popratne, a opet neizbježne, od kojih su neke NumPy, SkLearn, Imutils i MatPlotLib. Nadalje slike, kao jednini nosioc informacija u računalnom vidu, su temelj ovog rada i nad njima su izvršavane razne operacije s ciljem obrade i/ili izvlačenja korisnih informacija.

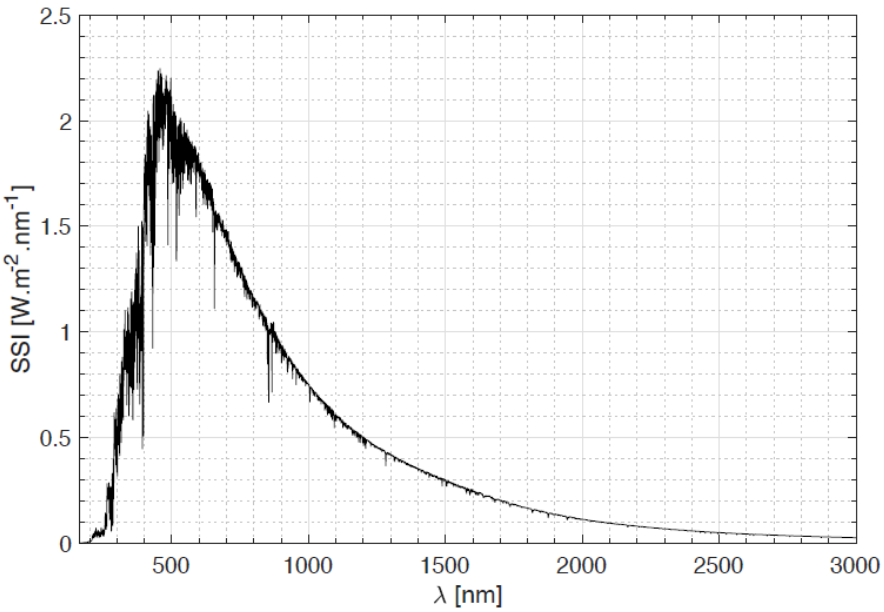
1.1. Slike

Poznato je da čovjek objekte u svojoj okolini vidi preko svjetlosti koju oni reflektiraju. Krećući od pretpostavke da je svjetlost koja pada na objekt sastavljena od svih valnih duljina, neke će objekt apsorbirati a neke reflektirati. Koje će se reflektirati ovisi o svojstvima tankog vanjskog sloja objekta. Ta svojstva se nazivaju boja. Tako će npr. crveni objekt apsorbirati sve valne duljine svjetlosti osim one koju karakterizira crvena boja (približno 0.7 μm). Reflektirana svjetlost se prolaskom kroz leću ljudskog oka lomi te se stvara rotirana slika razmatranog objekta u oku. Osim vidljivog dijela svjetlosti (slika 2) koju čovjek i većina životinjskog svijeta vidi postoji puno širi spektar one nama nevidljive. Tu su možda najpoznatije infracrvene i ultraljubičaste valne duljine svjetlosti.



Slika 2. Vidljiva svjetlost, Izvor [4]

Nije slučajnost da upravo ovaj raspon valnih dužina vide ljudi i životinje. Razlog tomu je što Sunce pri vidljivim valnim duljinama ima najintenzivnije zračenje (slika 3), čemu se očito evolucija priviknula.

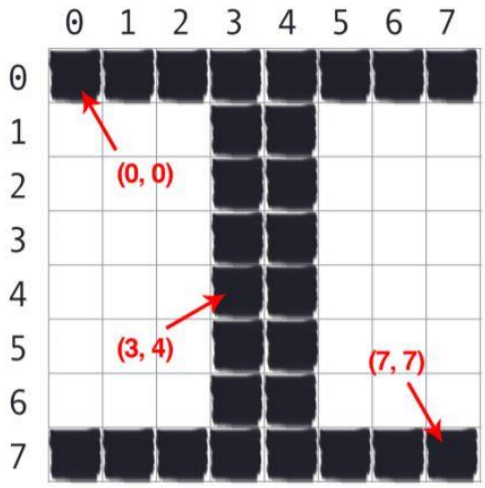


Slika 3. Ovisnost intenziteta zračenja Sunca o valnim duljinama

1.1.1. Osnove strojnog vida

1.1.1.1. Piksel i koordinatni sustav slika

Usprkos savršenom evolucijskom rješenju najkompleksnijeg osjetila, slika je u strojnom vidu potpuno drugačije realizirana – pomoću piksela. Piksel je osnovna jedinica svake slike koji opisuje koliki je intenzitet boje na tom mjestu i ne postoji finiji (manji) detalj slike od piksela! Ako se sastavni blokovi slike (pikseli) poslože na odgovarajući način dobije se slika. Najčešće se pikseli slažu tako da tvore pravokutnu sliku. Numeracija položaja piksela u slici vrlo je slična kartezijevom koordinatnom sustavu uz neke modifikacije.



Slika 4. Koordinate pikesla u slici, Izvor[5]

Iz slike se primjećuje osnovna razlika notacije položaja piksela od položaja točke u kartezijevom koordinatnom sustavu.

Broj piksela na slici se jednostavno izračunava po jednadžbi (1).

$$N_p = (m + 1) * (n + 1) \tag{1.1}$$

Gdje su *m* i *n* koordinate piksela u donjem desnom rubu slike.

Jednadžba (1.1) prilagođena je koordinatnom sustavu slika kojem je inicijalna vrijednost nula pa se zbog toga dodaje vrijednost nula stupcima i recima. Ova notacija nije uobičajena u nekim ostalim programskim jezicima poput Matlab-a pa je potrebna dodatna pozornost ukoliko se mijenja ili upoznaje jezik. Broj piksela je najčešće izražen u mega pikselima (Mp).

1.1.1.2. Prostori boja

1.1.1.2.1. Prostor sivih tonova

Kako je već rečeno piksel je osnovni nosioc informacije u slici, njegove koordinate su fiksne i besmisleno ih je perturbirati osim u iznimnim situacijama (zrcaljenje i sl.). Stoga mora postojati još jedna varijabla u pikselu koja je ključna za sliku. Ta varijabla se naziva intenzitet boje. Nju je najbolje opisati u najjednostavnijem i najosnovnijem prostoru boja a to je prostor sivih tonova (eng. grayscale).

U prostoru sivih tonova svaki piksel nosi jednu vrijednost koja je u binarnom zapisu najčešće 8-bitna, dakle u dekadskom poprima vrijednost od 0 do 255. Navedena vrijednost je analogna intenzitetu crne (sive) boje u svakom pikselu, odnosno na svakom mjestu na slici.

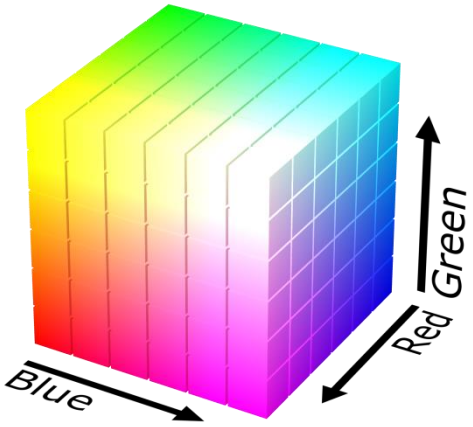
Slika 5. Ovisnost boje piksela o vrijednosti sivog tona

Na slici se vidi koju boju poprima piksel s kojom vrijednosti (interpolacija između rubnih vrijednosti je linearna). Može se reći da piksel s višom vrijednosti varijable sivog tona reflektira više svjetlosti, pa je tako piksel vrijednosti 0 potpuno crn dok je onaj s vrijednosti 255 bijel.

Zaključuje se da prostor sivih tonova ne nosi informaciju o boji već samo o intenzitetu refleksije. Zbog toga ovaj prostor boja zauzima minimalno memorije u računalu, ali s druge strane isto toliko informacija o pikselu pa onda i slici.

1.1.1.2.2. RGB prostor boja

Iako prostor sivih tonova nije nezastupljen u praksi, dosta češće se koristi Crveno-Zeleno-Plavi prostor boja poznatiji pod nazivom RGB (eng. Red Green Blue). Ovaj prostor boja omogućuje pogled na slike upravo onakav na kakvog je čovjek naviknuo. U slučaju RGB slike osim svjetline raspoznaju se i boje koje se generiraju različitim kombinacijama osnovnih boja, a to su Crvena, Zelena i Plava, upravo po čemu je i prostor dobio naziv.



Slika 6. RGB prostor boja

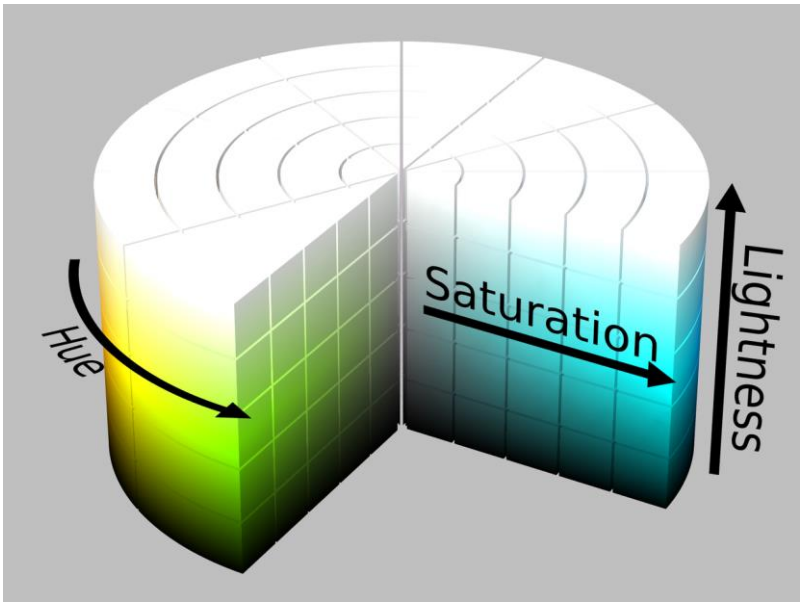
Na slici 6 vrlo jasno se vidi generiranje čitavog spektra boja na osnovu kombinacije intenziteta samo tri osnovne. Piksel pod ovim prostorom boja sad nema samo jednu (8-bitnu) varijablu, već su potrebne tri takve varijable (svaka za svoj kanal boje). Analogno slici 5, svaka od tri varijable predstavlja zastupljenost boje na koju se odnosi; crvene, zelene odnosno plave. Intuitivno se nameće pitanje koliko boja se može dobiti ovim pristupom?! Ako se broj boja označi sa N_c jednostavni izraz (1.2) daje vrijednost od približno 16 milijuna, što zasigurno nije malo.

$$N_c = 256^3 = 16\,777\,216 \quad (1.2)$$

Gdje je 256 broj koji može poprimiti svaki kanal a predstavlja intenzitet (zastupljenost) boje tog kanala na određenom pikselu.

RGB prostor boja je najzastupljeniji u praksi a osobito kod prikazivanja slika jer se jednostavno provodi zbrajanje svake boje.

1.1.1.2.3. HLS prostor boja



Slika 7. HLS prostor boja

Na slici vidi se da je veličina Hue najbitnija jer njen iznos određuje boju. U OpenCv biblioteci Hue vrijednost može biti cjelobrojna od 0 do 180, dok su Saturation i Lightness u rasponu od 0 do 255. Ovim prostorom boja dobiva se nekoliko prednosti. Jedna od njih je što se može odrediti vrlo uzak spektar boja koji ne ovisi o osvjetljenju ili intenzitetu i koji se može iznjedriti. Taj proces je i dio ovog Rada koji se primjenjuje kod eliminacije podloge. Naime, takvo nešto ne bi bilo izvedivo u RGB prostoru boja jer su u tom slučaju boje generirane adicijom tri osnovne boje i eliminacija bilo koje utječe na cijelu sliku.

1.2. Programski jezik Python

Prvi od preduvjeta koji se mora ispuniti je, između ostalog, odabrati programski jezik koji će osigurati kvalitetno okruženje za obradu slika i informacija, te za prikladno manipuliranje njima. Za taj zadatak je odabran Programski jezik Python. Ovaj programski jezik stvoren od strane Guido van Rossum-a godine 1991-e dopušta programiranje u objektnom, strukturnom i aspektnom stilu. Osim realizacije rješenja ovog problema u Python-u ono je moguće i u npr. Matlab-u ili pak u C++-u. Ipak Python-ova jednostavnost, preglednost i sve veća zajednica korisnika bili su presudni u odabiru upravo tog okruženja.

1.3. Korištene biblioteke

Sve biblioteke korištene u radu su pod „open-source“ licencom i najčešće održavane od strane volontera ili neprofitnih organizacija. Unošenje potrebnih paketa u okruženje (eng. environment) izvedeno je alatom zvanim Anaconda. Taj alat osim što stvara okruženje i instalira sve željene biblioteke, osigurava da se i popratne, možda ne na prvu potrebne, biblioteke instaliranju a bez kojih primarne ne mogu funkcionirati. Kako npr Tensorflow ili Keras koriste čitav spektar popratnih biblioteka ova osobina uvelike pomaže u kvalitetnom definiranju okruženja.

1.3.1. OpenCV

Inicijalno pušten u uporabu 2000-te, tek je svoju prvu ozbiljniju verziju dobio godine 2006-te OpenCv (eng. Open source Computer Vision) danas je najpoznatiji i najšire korišteni paket za sve funkcije obrade slika i videa. Moguća je i realizacija većine funkcija nad videom uživo, što je ultimativni cilj čije temelje polaže ovaj rad. OpenCV ima široku lepezu mogućnosti i funkcija. Neke od ponuđenih su korištene i u ovom Radu kao npr.:

- Učitavanja i spremanja slike
- Prikazivanja slike
- Promjena veličine slike
- Rotacija slike
- Zrcaljenje
- Primjena konvolucijskih filtara na slici (zamagljivanje, izoštravanje i sl.)

- Transformacija između različitih prostora boja (notacija) slika
- Pronalaženje kontura
- Uklanjanje podloge
- Odrezivanje dijela slike
- Pronalaženje maski (eng. mask)
- Logičke operacije nad maskama
- Pronalaženje histograma
- Usporedba histograma
- Pronalaženje orijentacije zatvorene konture u slici
- Izračunavanje dimenzija i površine
- Eliminacija pojedinih kanala boja
- Promjene osvjetljenja na slici

Svaka od navedenih funkcija ali i ostale korištene biti će detaljnije objašnjene kako općenito tako i kroz primjenu u radu.

2. PROCESIRANJE SLIKA

U ovom poglavlju opisane su odabrane funkcije za procesiranje slika koje se najintenzivnije koriste u ovom Radu. One su:

- Zamućivanje
- Izoštravanje
- Pronalaženje ruba
- Pravokutna međa minimalne površine
- Histogrami
- Usporedba histograma

2.1. Zamućivanje i izoštravanje slika

Zamućivanje i izoštravanje slika provodi se na sličan način s različitim filtrima (kernelima). Kernel je manja matrica (najčešće 5 x 5 ili 3 x 3) neparnih i, najčešće, jednakih dimenzija. Filter ima tzv. pivot a to je član u središtu matrice i upravo da bi on postojao mora biti matrica neparnih dimenzija. Matrica kernela se množi član po član sa slikom i nakon toga sve vrijednosti se sumiraju. Ne smije se dopustiti da sumirana vrijednost bude viša od 255 pa se s toga matrica mora podijeliti s pripadajućim brojem

$$K = \frac{1}{\texttt{ksize.width*ksize.height}} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 & 1 \\ 1 & 1 & 1 & \cdots & 1 & 1 \\ \cdots & & & & & \\ 1 & 1 & 1 & \cdots & 1 & 1 \end{bmatrix}$$

Slika 8. Kernel za zamućivanje

Sam proces provodi se tako da se pivot matrice postavi na prvi piksel slike i izvrši množenje član po član. Naravno, u ovoj poziciji nekoliko je članova koji nemaju svoj par pa se prije na sliku dodao tzv. okvir koji sadrži nule da bi množenje bilo definirano. Nakon množenja sve

vrijednosti se zbroje i postave na mjesto pivota. Tim činom kernel se pomiče za korak *S* koji je često 1 i ponavlja se radnja sve dok se ne prođu svi pikseli i svi kanali na slici.

Izoštravanje se provodi na isti način s razlikom što kernel više nije LPF (eng. Low Pass Filter) već je riječ o HPF (eng. High Pass Filter). Čest primjer kernela za izoštravanje je 3 x 3 matrica sa negativnim jedinicama po rubu i brojem 9 na mjestu pivota.

2.2. Pronalaženje ruba

Kod pronalaženja ruba, slika se prvo provodi kroz proces zamućivanja da bi se bilo kakav šum eliminirao (LPF). Šum može raditi prevelike oscilacije kod pronalaženja gradijenta.

Zamućenoj slici se sad primjenjuje npr. Sobel filter i u smjeru x i u smjeru y. Postoji niz različitih verzija filtara koji su prikladni za ovu fazu.

+2	+2	+4	+2	+2
+1	+1	+2	+1	+1
0	0	0	0	0
-1	-1	-2	-1	-1
-2	-2	-4	-2	-2

Gx

+2	+1	0	-1	-2
+2	+1	0	-1	-2
+4	+2	0	-2	-4
+2	+1	0	-1	-2
+2	+1	0	-1	-2

Gy

Slika 9. Sobel filter

Čime se računaju dvije komponente gradijenta (*G*): G_x i G_y. Na temelju toga može se odrediti iznos gradijenta i orijentacija.

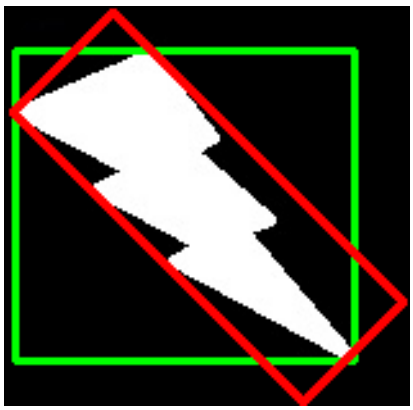
$$G = \sqrt{G_x^2 + G_y^2} \tag{2.1}$$

$$\varphi = \tan^{-1}\frac{G_y}{G_x} \tag{2.2}$$

Nakon čega se unosi uvjet pojasa iznosa gradijenta, koji je korisniku za tu aplikaciju koristan, unutar kojeg se smatra da se radi o rubu.

2.3. Pravokutna međa minimalne površine

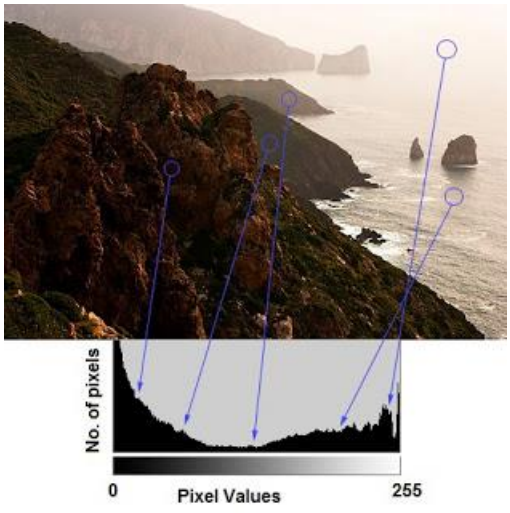
Nakon pronađenih rubova čest je zahtjev da se one na neki način ekstraktiraju za što je potrebno iste omeđiti. Međa može biti kružnica pravokutnik ili neki drugi geometrijski oblik. Kod pravokutnika najmanje površine pronalazi se središte konture i njena uzdužna os. Pravokutnik je jednostavno konstruiran na temelju četiri podatka: središte, kut između x osi pravokutnika i x osi slike, širina i duljina pravokutnika.



Slika 10. Primjer uspravnog i pravokutnika minimalne površine

2.4. Histogrami

Histogrami slika su grafovi ili podaci koji govore koliko koje boje kojeg intenziteta ima na dotičnoj slici. Kako su histogrami grafovi s iznosima na x osi od 0 do 255, y iznos histograma definirana je brojem piksela koji sadrže tu boju tog intenziteta.



Slika 11. Slika u sivim tonovima s pripadajućim histogramom

2.5. Usporedba histograma

Usporedba histograma se koristi kao zamjena direktnoj usporedbi slika. Nekoliko je algoritama za usporedbu.

Korelacijska usporedba:

$$d(H_1, H_2) = \frac{\sum(H_1(i) - \bar{H}_1)(H_2(i) - \bar{H}_2)}{\sqrt{\sum(H_2(i) - \bar{H}_2)^2 \sum(H_2(j) - \bar{H}_2)^2}} \tag{2.3}$$

Chi-kvadrat:

$$d(H_1, H_2) = \sum \frac{(H_1(i) - H_2(i))^2}{H_1(i)} \tag{2.4}$$

Intersekcija:

$$d(H_1, H_2) = \sum \min(H_1(i), H_2(i)) \tag{2.5}$$

Bhattacharyy-eva udaljenost:

$$d(H_1, H_2) = \sqrt{1 - \frac{1}{\sqrt{\bar{H}_1 \bar{H}_2 N^2}} \sum \sqrt{H_1(i) H_2 i}} \tag{2.6}$$

Prosjek histograma:

$$\bar{H}_k = \frac{1}{N} \sum H_k(n) \tag{2.7}$$

Gdje su:

$d(H_1, H_2)$ – Razlika histograma

$H_{1,2}$ – Prvi histogram

$H_{1,2,k}(i)$ – *i*-ti član histograma

N- Broj članova u histogramu

$\bar{H}_{1,2,k}$- Prosjek histograma.

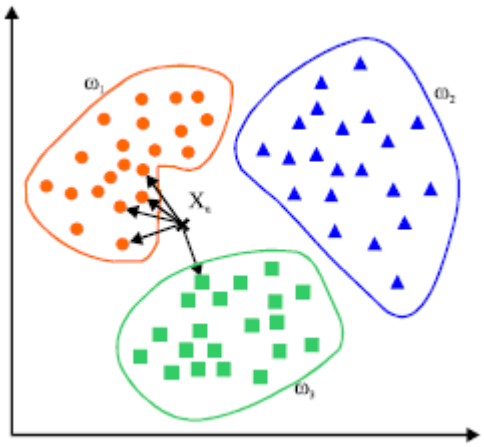
3. STROJNO UČENJE

U ovom poglavlju se postavljaju temelji za primjenu metoda strojnog učenja u zadanom problemu. Prolazi se kroz oblike kao što je k-ti najbliži susjed (eng. k-nearest neighbour). Nadalje objašnjavaju se pojmovi kao funkcija gubitaka (poprečna entropija), spuštanje gradijenta, stohastičko spuštanje gradijenta, razni momenti, Nesertove akceleracije i regularizacija. Dublje se, u ovom poglavlju, ulazi u konvolucijske neuronske mreže. Fokus poglavlja je na konvolucijama, konvolucijskim algoritmima i konvolucijskim mrežama. Uz to se pobliže razmatraju vrste i namjena svakog od slojeva u konvolucijskim neuronskim mrežama, njihova interakcija i inicijalizacija.

3.1. K-NN klasifikator

Iako je k-ti najbliži susjed (u daljnjem tekstu *k*-NN) metoda koja zapravo ne provodi nikakvo učenje tokom klasifikacije, ona je i dalje vrlo bitna za razumijevanje kompliciranijih algoritama.

Osnovna ideja ovog algoritma leži u činjenici da podaci iz iste klase leže blizu u n-dimenzionalnom prostoru. Parametri takvog prostora su nekakve osobine (eng. features) te se ovisno o tome podaci raspoređuju. Nakon dovoljnog broja podataka iz različitih klasa nastaje tzv. nakupljanje na hrpe (ω_1, ω_2, ω_3...), gdje su u istoj hrpi podaci iz iste klase. Postavljanjem neklasificiranog podatka u takav sustav i detekcijom pripadnosti njegovih *k* najbližih susjeda može se odrediti klasa tog podatka.



Slika 12. Primjer k-NN klasifikatora

Dvije su metode računanja udaljenosti. Prva je L1, još se zove i Manhattan udaljenost.

$$d(p,q) = \sum_{1=1}^{N} |\, q_i - p_i| \tag{3.1}$$

Druga, L2, je Eucledianova udaljenost.

$$d(p,q) = \sqrt{\sum_{1=1}^{N} (q_i - p_i)^2} \tag{3.2}$$

Gdje su:

$d(p,q)$ - Udaljenost

q_i – *i*-ta osobina *q*-te klase

p_i – *i*-ta osobina *p*-te klase.

Naravno, ukoliko su kao ulaz u k-NN slike one moraju biti svedene na iste dimenzije. U ovom Radu se kao ulaz unose već ekstraktirani podaci koji, opet, moraju biti istih dimenzija.

3.1.1. Prednosti i mane k-NN-a

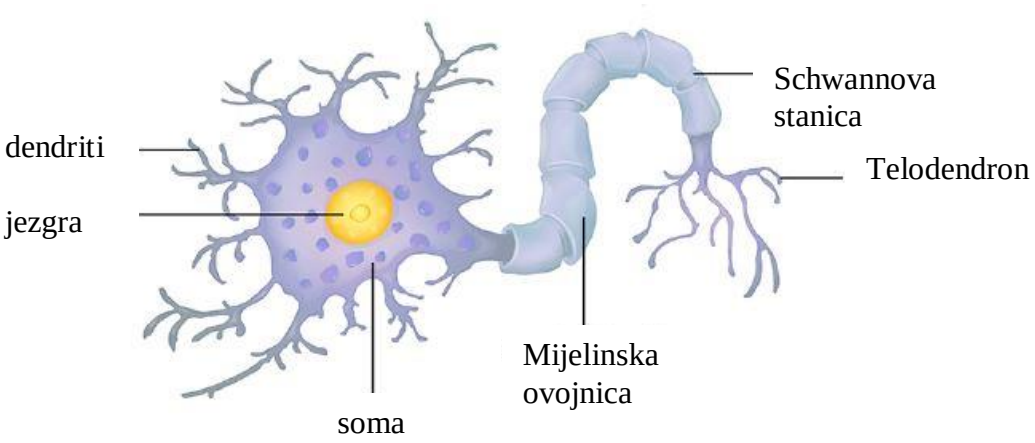
Očita prednost ovog algoritma je u njegovoj jednostavnosti za implementaciju i razumijevanje. Uz to klasifikator ne treba vremena za učenje pošto je sve što se radi spremanje podataka i računanje udaljenosti kod finalne klasifikacije. Nasuprot tome, veliko je vrijeme klasifikacije (usporedbe s ostalim podacima) jer se novi podatak uspoređuje sa svim iz baze podataka. Povećanjem baze podataka proces postaje računalno neodrživ.

Ovaj manjak može se kompenzirati algoritmom „Prosječnog najbližeg susjeda“ (eng. Approximate Nearest Neighbor) ili kraće A-NN. Primjeri A-NN-a su kd-trees [10], FLANN [11], random projections [12] [13] [14] itd.

Možda najveći manjak ovog klasifikatora je taj što su udaljenosti u n-dimenzionalnim prostorima često neintuitivne [15].

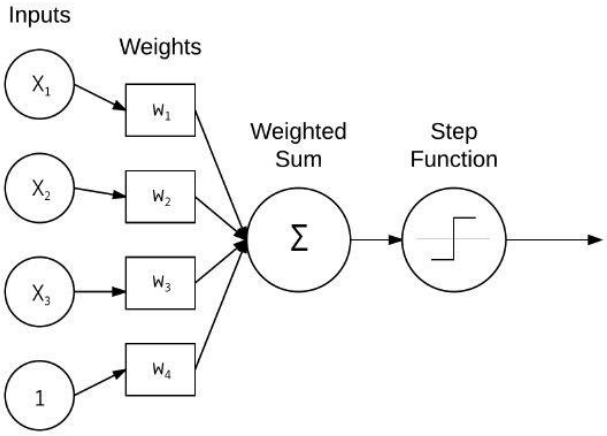
3.2. Neuronske mreže

Umjetne neuronske mreže sastavljene su od ulaza, određenog broja slojeva s konačnim brojem umjetnih neurona, težina i izlaza. Zamišljene su po modelu biološkog mozga gdje umreženi neuroni „ispaljuju“ signale kad vrijednost sume njihovih ulaza pređe određenu granicu koja je definirana tzv. aktivacijskom funkcijom.



Slika 13. Biološki neuron

Najosnovnija mreža uzima sumu svih ulaza pomnoženim s odgovarajućim težinama.



Slika 14. Jednostavna neuronska mreža

Može se pisati da je izlaz:

$$f(net) = \sum_{1=1}^{N} w_i x_i. \tag{3.3}$$

Gdje je:

$f(net)$ – izlaz mreže

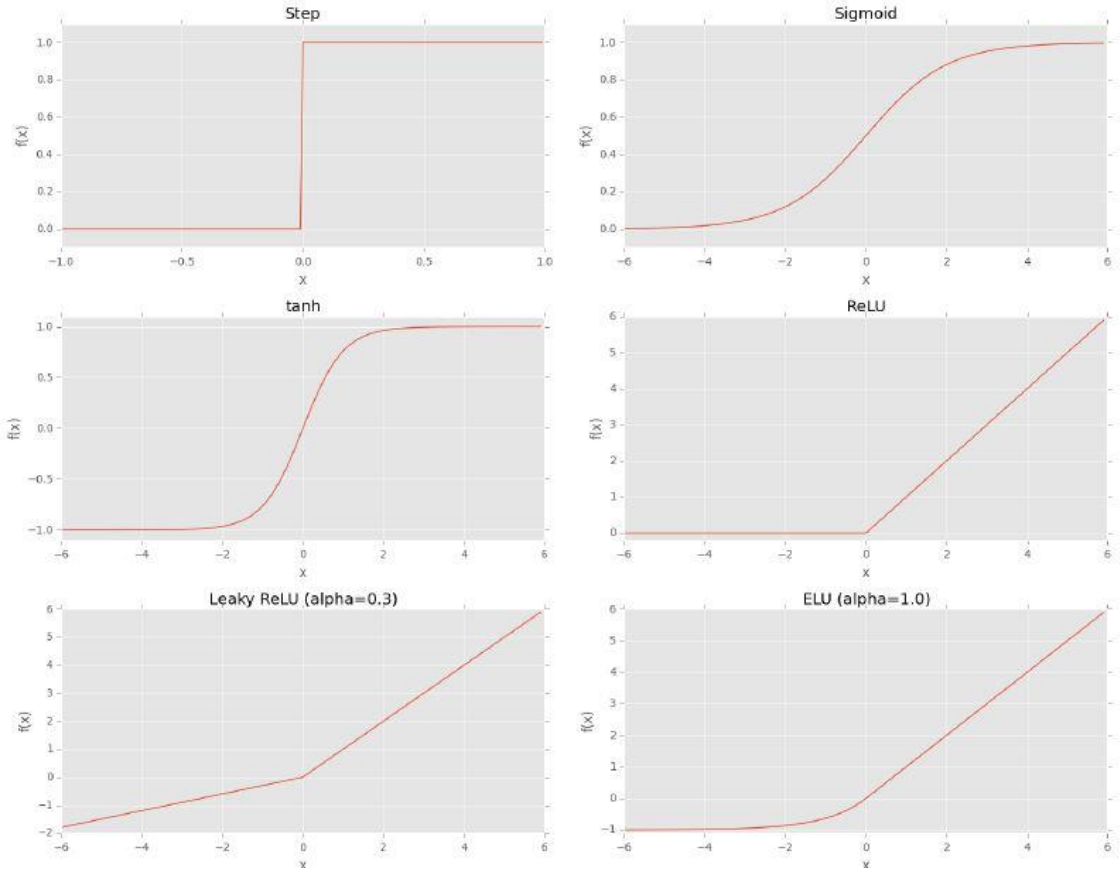
x_i – i-ti ulaz

w_i – težina i-tog ulaza.

3.2.1. Aktivacijske funkcije

Na slici 14 kao aktivacijska funkcija nalazi se najosnovnija „step“ funkcija. Uz nju postoji čitava lepeza ostalih funkcija. Iskustveno ili eksperimentiranjem se određuje koja od funkcija na raspolaganju najbolje odgovara zadanom problemu ili arhitekturi mreže.

Neke od funkcija su: Step, Sigmoid, Tanh, ReLU, Leaky ReLU, ELU itd.



Slika 15. Aktivacijske funkcije

Problem, inače dobre, *step* funkcije je taj što nije kontinuirana. Drugim riječima, njene derivacije nisu definirane u području skoka. Derivabilnost je vrlo bitna kod primjene metode spuštanja gradijenta koja je opisana u idućim poglavljima.

Zbog toga je najčešće korištena aktivacijska funkcija – sigmoidalna (3.4).

$$s(t) = 1/(1 + e^{-t}) \tag{3.4}$$

Prednosti sigmoidalne funkcije:

- Kontinuirana je i derivabilna
- Simetrična po y osi
- Asimptotsko se približava vrijednostima zasićenja

Mane sigmoidalne funkcije:

- Nije simetrična oko ishodišta
- Zasićeni neuroni ruše gradijent jer je razlika gradijenta gotovo nula.

Od ostalih navedenih vrijedi istaknuti ELU funkciju (eng. Experimental Linear Units). Prvi put iznesena u [15].

$$f(net) = \begin{cases} net & \text{if } net >= 0 \\ \alpha \times net & \text{ostalo} \end{cases} \tag{3.5}$$

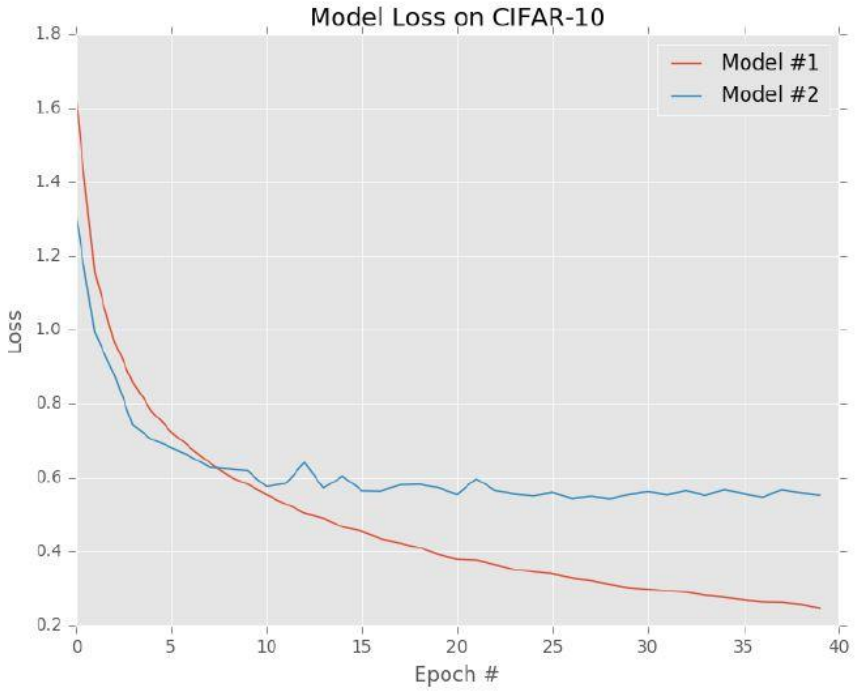
Gdje je α koeficijent nagiba.

Upravo ova funkcija je iznesena jer predstavlja predmet sve većeg istraživanja i u čestim slučajevima daje bolje odzive od konvencionalne sigmoidalne funkcije.

3.2.2. Optimizacijske metode i regularizacija

3.2.2.1. Funkcija gubitaka

Za funkciju gubitaka (eng. loss function) se može reći da je mjerilo koliko dobro dati klasifikator klasificira zadani problem. Iako njena definicija zvuči trivijalna, definiranje funkcije gubitaka, svega onog što ona predstavlja i sve ono što se na njoj temelji nije jednostavno. Matematička objašnjenja mogu se naći u [16], [17], [18], [19].



Slika 16. Grafički prikaz funkcija oblika

3.2.2.1.1. SVM funkcija gubitaka kod više-klasne klasifikacije

Više-klasni SVM (eng. Support Vector Machines) govori koliki je gubitak kod klasifikacije s inicijaliziranih više klasa. Neka su vektor **X** podaci koji se klasificiraju a vektor **y** su točne klase za svaki podatak u **X**-u.

Tad je funkcija cilja:

$$s = f(x_i, W) \tag{3.6}$$

Sto znači da može povezati *j*-tu klasu s *i*-tim podatkom.

$$s_j = f(x_i, W)_j \tag{3.7}$$

Sad se zbrajaju sve netočne klase i uspoređuju izlaz funkcije cilja *s* za *j*-tu (pogrešno pretpostavljenu) klasu i y_i-tu (točnu) klasu. To čini zavisnu funkciju gubitaka.

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1) \tag{3.8}$$

Konačno, nezavisna funkcija gubitaka glasi:

$$L = \frac{1}{N} \sum_{i=1}^{N} L_i \tag{3.9}$$

Često se nailazi na kvadratnu zavisnu funkciju oblika:

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)^2 \tag{3.10}$$

3.2.2.1.2. Gubitak poprečne entropije

Zavisna funkcija gubitaka može se zamijeniti nenormaliziranim logaritamskim vjerojatnostima za svaku klasu.

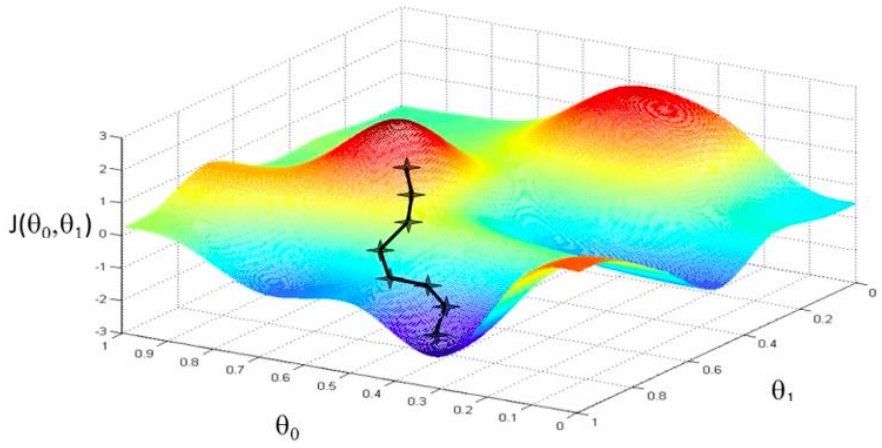
$$L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right) \tag{3.11}$$

Koja se onda proširuje na sve klase po izrazu (3.9).

3.2.2.2. Stohastičko spuštanje gradijenta

Stohastičko spuštanje gradijenta (u daljnjem tekstu SGD), iako uvedeno prije 60-ak godina [20], i danas predstavlja najvažniji algoritam kod primjene metoda dubokog učenja.

SGD je iterativna optimizacijska metoda koja operira nad poljem funkcije gubitaka (još se zove i optimizacijska površina).



Slika 17. Kretanje SGD-a po optimizacijskoj površini

Aproksimativni izraz za gradijent je:

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x-h) - f(x)}{h}. \tag{3.12}$$

Gdje je:

h - korak

$f(x-h)$ - prethodna vrijednost funkcije

$f(x)$ – sadašnja vrijednost funkcije.

Analogno izrazu (3.12) mogu se pronaći sve komponente vektora gradijenta u višedimenzionalnim prostorima. Iako je izraz (3.12) aproksimativan, potrebno je mnogo računalnih resursa da bi se koristio. Umjesto toga koriste se izrazi iz [21], [22] i [23].

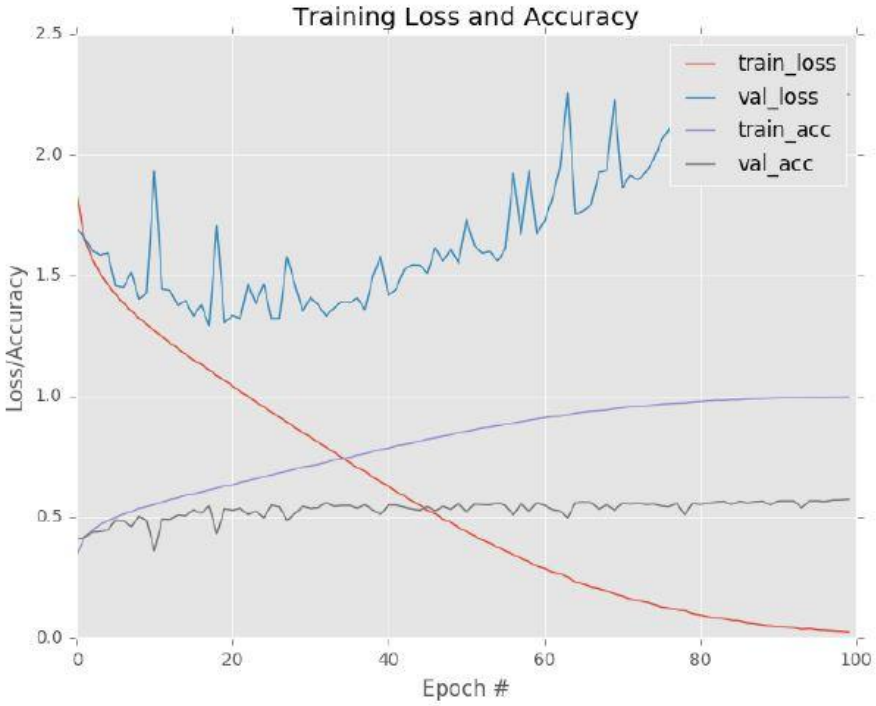
Rad SGD-a se može opisati tako da se iz trenutne točke na optimizacijskoj površini pomiče na novu koja je upravo u smjeru najvećeg pada gradijenta. Obično spuštanje gradijenta (tzv. „vanilija“) postaje nemoguće sporo s povećanjem baze podataka. Stohastički istovremeno radi

nad jednom skupinom podataka (hrpa) koja je dosta manja od cijele baze i na taj način osigurava brzinu izvođenja. Veličina hrpe se preporuča da bude potencija s bazom 2 [4].

3.2.2.3. Regularizacija

„Mnoge strategije korištene u strojnom učenju su eksplicitno zamišljene za redukciju pogreške na testnoj bazi, često na uštrp pogreške na bazi za učenje. Ove strategije se nazivaju regularizacija.“ [24]

Regularizacijom se vodi borba protiv tzv. overfitting-a, koji predstavlja slučaj kad trenirani model toliko izgubi generalizacijsko svojstvo da se ponaša gotovo idealno kod učenja, a ne uspije slične rezultate ostvariti na testu.



Slika 18. Primjer pretreniranosti

Na slici 18. vidi se da gubitak kod učenja (train_loss) vrlo lijepo pada, dok gubitak kod provjere (validation_loss) raste i nestabilan je. Ovo je čisti primjer ne generaliziranog modela.

Ako se uzme primjer gdje neki model s matricom težina **W** klasificira potpuno točno neki zadatak. Postavlja se pitanje ima li u toj matrici težina nekih vrijednosti koje suvislo odskaču od ostalih i postoji li neka bolja matrica **W** kojoj je svojstvo generaliziranja izraženije?!

Matricu **W** se može „kontrolirati“ pomoću penala koji se uključuju u funkciju gubitaka.

Izraz (3.9) sad izgleda ovako:

$$L = \frac{1}{N}\sum_{i=1}^{N} L_i + \beta R(W) \tag{3.13}$$

Gdje su:

β- hiperparametar koji određuje utjecaj regularizacije

$R(W)$- metoda regularizacije (L1, L2 ili ElasticNet [25]).

Izraz za L1 regularizaciju je:

$$R(W) = \sum_i \sum_j W_{i,j}^2. \tag{3.14}$$

Dakle ona kvadrira sve težine i zbraja ih, čime penalizira velike iznose.

Izraz za L2 regularizaciju je:

$$R(W) = \sum_i \sum_j |W_{i,j}|. \tag{3.15}$$

ElasticNet kombinira L1 i L2 regularizaciju.

$$R(W) = \sum_i \sum_j \varphi W_{i,j}^2 + |W_{i,j}| \tag{3.16}$$

3.2.2.4. Moment i Nesterova akceleracija

Kod „kretanja“ optimizacijskim poljem može se uključiti moment koji će forsirati pomake u smjeru ako je u posljednjih nekoliko epoha smjer maksimalnog pada gradijenta ostao sličan. Problem kod momenta je da se vrlo lako može preskočiti lokalni minimum ako se moment predugo nakuplja.

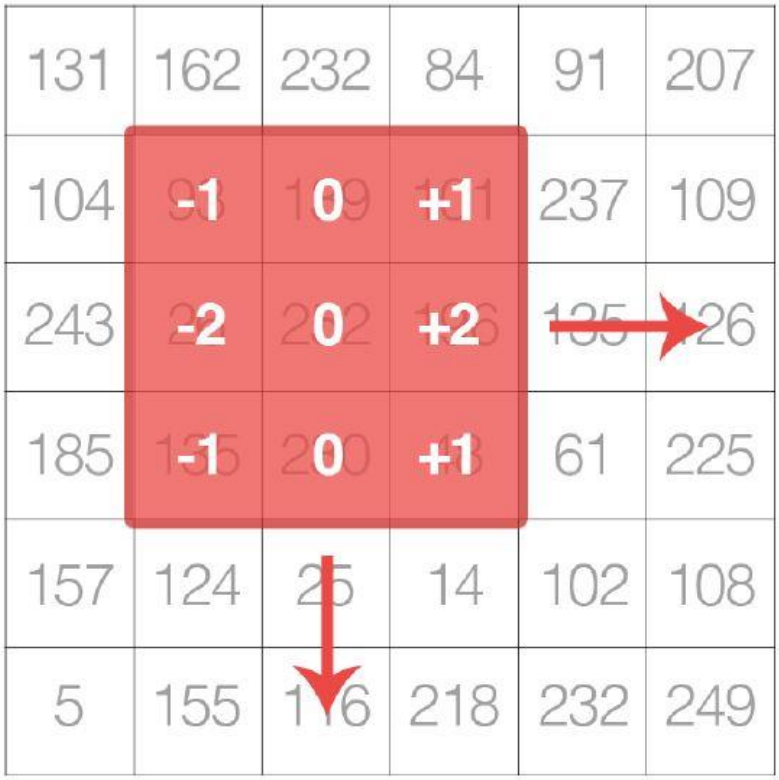
Nesterova akceleracija radi po principu da je idući pomak definiran prošlim smjerom najvećeg pada gradijenta, a novi smjer pada se računa u novoj točki i tek tada vrši manja korekcija s obzirom na rezultat [26].

3.3. Konvolucijske neuronske mreže

Cijelo dosadašnje teoretsko razmatranje dovodi do ovog poglavlja – konvolucijske neuronske mreže (eng. Convolutional Neural Network). Za razliku od konvencionalnih mreža koje se sastoje od slojeva potpuno umreženih neurona, konvolucijske mreže koriste potpuno umreženu strukturu samo kod posljednjih nekoliko slojeva, dok imaju barem jedan „konvolucijski“ sloj u arhitekturi [24]. Svaki sloj u konvolucijskim mrežama sadrži nekakav tip filtera (kernel), a često i nekoliko tisuća njih. Tokom učenja, od mreže se očekuje da sama nauči vrijednosti filtara a samim time se može zaključiti da je mreža potencijalno sposobna npr. detektirati rubove, koristeći rubove detektirati oblike, koristeći oblike detektirati strukture i osobine itd.

3.3.1. Konvolucije

U terminu dubokog učenja konvolucija je množenje matrica element po element nakon kojeg sjedi sumiranje. Iako na izgled ne izgleda tako upravo na ovaj način rade sve primjene filtara kao npr. zamućivanje, zaoštravanje itd. nad slikama. Proces se može zamisliti da se tzv. kernel (manja matrica) primjenjuje na sliku (velika matrica) na način da se nakon svake konvolucije pomakne za definirani korak po x, odnosno y osi.



Slika 19. Primjena kernela na sliku

Kernel na slici primijenjen uz korak $S = 1$ je upravo kernel koji koristi funkcija u OpenCv-u zvana cv2.filter2D( ). Preporuča se da kerneli kod konvolucijskih mreža budu neparnih dimenzija. Time postoji središnja koordinata i pojednostavnjuje se postupak uparivanja kernela s slikom. Isto tako preporuka je da je kernel istih dimenzija visine i širine kako bi do izražaja došla optimizirana linearna algebra koja najbolje radi na takvim matricama. Primjećuje se još jedan problem a to je činjenica da kad je središnja koordinata kernela na rubu slike (ujedno i rubna pozicija u koju kernel dolazi) postoji nekoliko članova koji nemaju par za izvršenje konvolucije (množenja). Zbog toga se pristupa, najčešće, dodavanju okvira oko originalne slike koji bi sadržavao samo nule. Taj proces se na eng. poznatije zove: zero padding.

Kako je već spomenuto, konvolucijske mreže koristeći konvolucijske filtre, nelinearne aktivacijske funkcije, sloj kontrakcije i povratnu propagaciju, mogu naučiti vrijednosti konvolucijskih filtara. Tako u plićim slojevima može doći do detekcije rubova a iz toga u dubljim slojevima do detekcije cijelih osobina na slikama.

3.3.2. Vrste slojeva kod konvolucijskih mreža

Postoje različite vrste slojeva koji različitim kombinacijama mogu stvoriti adekvatnu arhitekturu mreže za određenu primjenu.

Vrste slojeva su:

- Konvolucijski (CONV)
- Aktivacijski (ACT)
- Sloj kontrakcije (eng. Pooling) (POOL)
- Potpuno umreženi (eng. Fully-connected) (FC)
- Normalizacija hrpe (eng. Batch normalization) (BN)
- Odbacivanje (eng. Dropout) (DO)

3.3.2.1. Konvolucijski slojevi

Konvolucijski slojevi sastoje se od seta s *K* konvolucijskih filtara koji trebaju biti naučeni. Oni su dvodimenzionalni i relativno mali u odnosu na dimenziju ulaznog podataka (slike). Konvolucijom svakog od kernela nad slikom dobije se opet dvodimenzionalni izlaz. Nadalje primjenom svih *K* konvolucijskih filtara na originalnu sliku i slaganjem izlaza, dobije se mapa izlaza širine *h*, visine *w* i dubine *K*.

Primjenom filtara relativno malih dimenzija na sliku relativno velikih dobiju se dva efekta – lokalna nezavisnost i receptivna polja. Pod lokalnom nezavisnošću podrazumijeva se da objekt može biti bilo gdje na slici i da će izlaz mreže biti jednako pobuđen. Receptivna polja govore kako je svaki neuron u izlaznim slojevima mreže (FC slojevima) zadužen za svoje polje na ulaznoj slici.

Izlazne mape iz svakog od konvolucijskih slojeva zovu se još i aktivacijske mape. Dimenzije aktivacijskih mapa moraju se namještati. Dubina mape ovisi o broju filtara *K*, dok ostale dvije dimenzije mogu biti kontrolirane korakom filtra *S* i dimenzijom filtra. Većim korakom filtra i većim dimenzijama filtra *F* smanjuje se dimenzija aktivacijske mape. Također na dimenzije utječe i količina dodanih nula po okvirima.

Donja granica koja mora biti ispunjena za pravilno funkcioniranje konvolucijskog sloja je da idući izrazi budu cijeli brojevi.

$$A = \frac{W - F + 2P}{S} \tag{3.17}$$

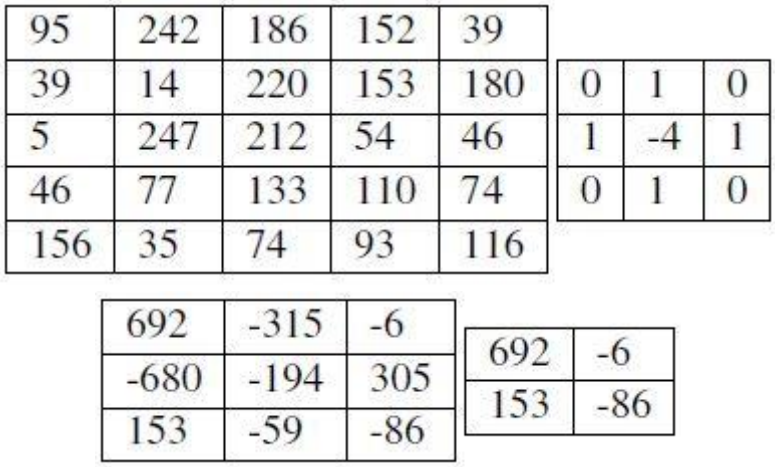
$$A = \frac{H - F + 2P}{S} \tag{3.18}$$

Gdje su:

W – širina slike

H – visina slike

P – količina dodanih nula po okviru slike.



Slika 20. Primjena konvolucijskog filtra

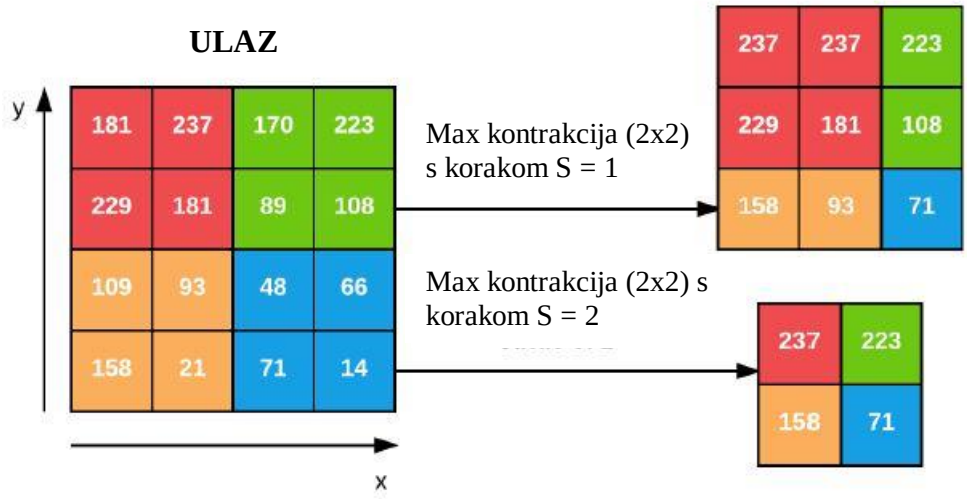
Na slici gore lijevo je zamišljeni ulaz u konvolucijski proces a gore desno je kernel (Laplaceov kernel). Na donjoj polovici slike lijevo je izlaz nakon primjene kernela uz korak S = 1 dok je na desnoj strani izlaz uz korak S = 2. Zaključuje se da dvije od tri dimenzije aktivacijskih mapa ovise o veličini kernela i koraku. Treća dimenzija ovisi o broju kernela K.

3.3.2.2. Aktivacijksi slojevi

Nakon svakog konvolucijskog sloja primjenjuje se aktivacijski sloj koji sadrži neku od aktivacijskih funkcija iz poglavlja 3.2.1. koja je derivabilna. Aktivacijski slojevi zapravo nisu slojevi jer ne sadrže nikakve parametre (težine) koje se u njima uče. Aktivacijski slojevi kao ulaz prihvaćaju aktivacijske mape (dimenzija W x H x K) i primjenjuje se aktivacijska funkcija element po element. Nakon aktivacijskog sloja podatak nije promijenio svoje dimenzije.

3.3.2.3. Kontrakcijski slojevi

Dva su osnovna načina redukcije dimenzija aktivacijskih mapa ili ulaza u konvolucijski sloj. Prvi je već opisani način povećanja koraka S. Drugi način koristi tzv. kontrakcijske slojeve koji rade na sličnom principu kao konvolucijski filtri osim u tome što ne vrše multiplikaciju i sumiranje, već unutar kernela dimenzija N x N ostavlja se samo najveći broj u aktivacijskoj mapi. Na taj način se reducira količina podataka kroz mrežu koja ima tendenciju povećanja, osobito kod dubljih konvolucijskih mreža.



Slika 21. Kontrakcijski sloj

Na slici se radi o maksimalnoj kontrakciji koja koristi funkciju *max(0,x)*. Kod konstrukcije konvolucijskih mreža ne smiju se previše koristiti kontrakcijski slojevi jer se njihovim korištenjem gubi dosta informacija.

3.3.2.4. Potpuno umreženi slojevi

Konvencionalni, potpuno umreženi slojevi se postavljaju kao posljednji slojevi u konvolucijskim mrežama prije primjene *Softmax* funkcije. Oni predstavljaju vezu između konvolucija i izlaza mreže. Potpuno umrežene slojeve nema smisla postavljati bilo gdje drugo u mreži. Standard je da se postavlja jedan do dva sloja prije pozivanja *Softmax* funkcije na izlazu mreže.

3.3.2.5. Slojevi za normalizaciju hrpe

Prvi put predstavljeni u [27], ovi slojevi služe, kako naslov ukazuje, da bi se aktivacijske vrijednosti normalizirale prije prelaženja u idući sloj. Ako se zamisli *x* kao hrpa aktivacija ona se može normalizirati na idući način:

$$\widehat{x_\imath} = \frac{x_i - \mu_\beta}{\sqrt{б_\beta^2 + \varepsilon}} \tag{3.19}$$

Gdje je:

$\widehat{x_\imath}$ – *i*-ta normalizirana hrpa

x_i – *i*-ita hrpa

ε - neki vrlo mali broj koji služi da bi se izbjeglo dijeljenje s nulom,

μ_β i $б_\beta^2$ su:

$$\mu_\beta = \frac{1}{m}\sum_{i=1}^{m} x_i \tag{3.20}$$

$$б_\beta^2 = \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_\beta)^2 \tag{3.21}$$

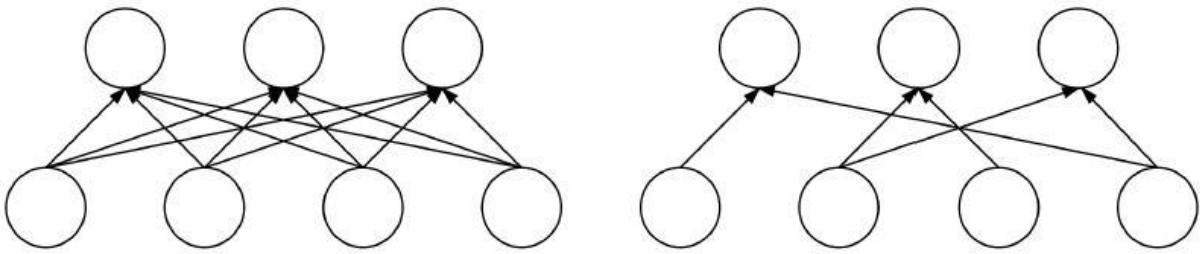
Ovim postupkom se dobije da se sve aktivacije centrirane oko nule.

Kod faze testiranja μ_β i $б_\beta^2$ se uzimaju prosječne vrijednosti što znači da na testiranje neće utjecati posljednja hrpa koja je prošla proces i na taj način se održava točnost testiranja.

Normalizacija hrpi se pokazala iznimno korisnom kod smanjivanja broja koraka učenja (eng. epoch). Korištenje normalizacije hrpi u konvolucijskim mrežama nosi i prednost daleko lakšeg namještanja ostalih hiperparametara. Mana ovog postupka je što, usprkos smanjenom broju koraka, proces učenja traje često duže (dva do tri puta) zbog dodatne algebre i statistike nad svakom od hrpa.

3.3.2.6. Slojevi odbacivanja

Sloj odbacivanja je zapravo oblik regularizacije. On se kao i ostali regularizacijski algoritmi bori protiv fenomena pretreniranosti (eng. overfitting) kad model postaje vrlo dobar za istrenirani set podataka ali ne uspijeva održati točnost nad novim podacima. Čest razlog ovog problema je što nekoliko neurona (samo nekoliko osobina) vodi cijelu priču odlučivanja zbog toga što su u procesu učenja prerasli ostale. Sloj odbacivanja se zato postavlja na krajnjim potpuno umreženim slojevima i ovaj sloj nasumično puče veze neurona u tim slojevima. Definiranim postotkom neke od veza nisu aktivne u trenutnoj epohi, dok će ostale postati neaktivne u drugoj. Česti faktor odbacivanja je od 40% do 75%.



Slika 22. Primjer odbacivanja

Na slici lijevo vidi se oblik izvornog potpuno umreženog sloja. Na desnoj strani je na istom sloju primijenjeno odbacivanje u visini od 50%.

4. BAZA PODATAKA

Kako je već u uvodnom dijelu ovog rada spomenuto, izvor svih informacija o vrsti ribe će isključivo biti slike. Svi klasifikacijski algoritmi koji se temelje na metodama strojnog učenja zahtijevaju kao temelj određenu količinu već točno razvrstanih podataka (u ovom slučaju slika). To znači da se prije bilo kakvog razvoja algoritma mora doći do dovoljne količine slika inćuna i srdela. Lako je zaključiti da se radi o vrlo specifičnoj bazi podataka koja, za razliku od konvencionalnijih kao npr. MNIST (baza podataka ručno pisanih znamenki 0 – 9), CIFAR-10 (baza od 60 000 slika aviona, automobila, ptica, mačaka, jelena, pasa, žabi, konja, brodova i kamiona), SMILES, Flowers-17, CALTECH-101, Tiny ImageNet 200, Adience, ImageNet, Indoor CVPR, Stanford Cars itd., nije dostupna kao već gotova baza podataka. Zbog toga se pristupilo izradi vlastite baze podataka koja je specifična i čija je jedina svrha pružanje kvalitetnog temelja algoritmima klasifikacije koji se u ovom Radu razvijaju. Cijela baza podataka je izrađena od strane Autora ovog Rada.

4.1. Preduvjeti kod prikupljanja baze podataka

Da bi se osigurala konzistentnost i dobra definiranost baze podataka potrebno je kod slikanja obratiti pažnju na nekoliko detalja a to su:

- Osvjetljenje
- Boja pozadine
- Refleksija pozadine
- Udaljenost slikanja
- Preklapanje riba
- Pozicija ribe

Kako je svrha ekstrakcija što više osobina koje mogu biti specifične za jednu od dvije promatrane vrste ribe, tako se mora osigurati da svjetlo osvjetljenja pruža cijeli spektar vidljivih boja kako je to opisano u poglavlju 1.1. Za osvjetljenje se odabiru izvori bijele boje koji su ravnomjerno raspoređeni da ne bi došlo do lokalnog zasićenja svjetlom. U tom slučaju rezultati bi iznimno ovisili o položaju ribe što je nedopustivo jer se protivi pretpostavci da ribe dolaze razbacane po traci. Osim osvjetljenja, prije izrade baze podataka, potrebno je

odabrati i pozadinu odnosno boju i vrst trake po kojoj će ribe biti raspoređene. S obzirom da spektar boja od kojih je sastavljen vizualni otisak riba (inćuna i srdela) ne zadrži narančastu boju, upravo ta boja je odabrana za boju pozadine. Uz svojstvo boje vrlo je bitno da pozadina nije reflektirajuća jer se u suprotnom može lako uklopiti s bijelim dijelovima ribe, što nadalje stvara problem kod detektiranja konture ribe. U ovom radu se vrši obrada na čitavim nizom drugih svojstava uključujući i one apstraktne do kojih se dolazi u procesu učenja neuronskih mreža. Imajući to na umu ne smije se dopustiti da baza podataka sadrži fotografije koje su fotografirane s različitih udaljenosti jer su sva svojstva, osim onih koji sadrže omjere, varijantna na udaljenost slikanja.

4.2. Osnova baze podataka

Iako su na prvi pogled srdele srdele, a inćuni inćuni, problematika nije tako trivijalna. Vodeći se po Charles Darwin-u [5] postoji različitost između jedinki iste vrste. Štoviše, to je osobito izraženo kod sitne plave ribe, čiji članovi su i ove vrste. Razlike ovise o doba godine, lokaciji ulova pa čak i mjesečevim mijenama. Imajući na umu da ova problematika nije zanemariva, ona ipak daleko prelazi okvire ovog rada te će se pokušati kompenzirati na najjednostavniji mogući način – fotografiranjem za bazu podataka u dva navrata. S dva različita uzorka se dobila relativno dobra invarijantnost baze podataka na primjetne različitosti između jedinki iste vrste.

Kako slike u bazi podataka moraju sadržavati isključivo po jednu jedinku ribe, a fotografiranje jedinku po jedinku je sve samo ne optimizirano, pristupa se metodi fotografiranja više razbacanih jedinki. Takva fotografija mora biti pretprocesirana da bi se iz nje iznjedrila prava baza podataka o čemu se detaljnije nalazi u idućem poglavlju. Odabrana metoda fotografiranja više jedinki i pretprocesiranja takvih fotografija je i jedini način kako se realizira primjena u industriji s ponovnim referenciranjem na pretpostavku da ribe dolaze razbacane na traci.

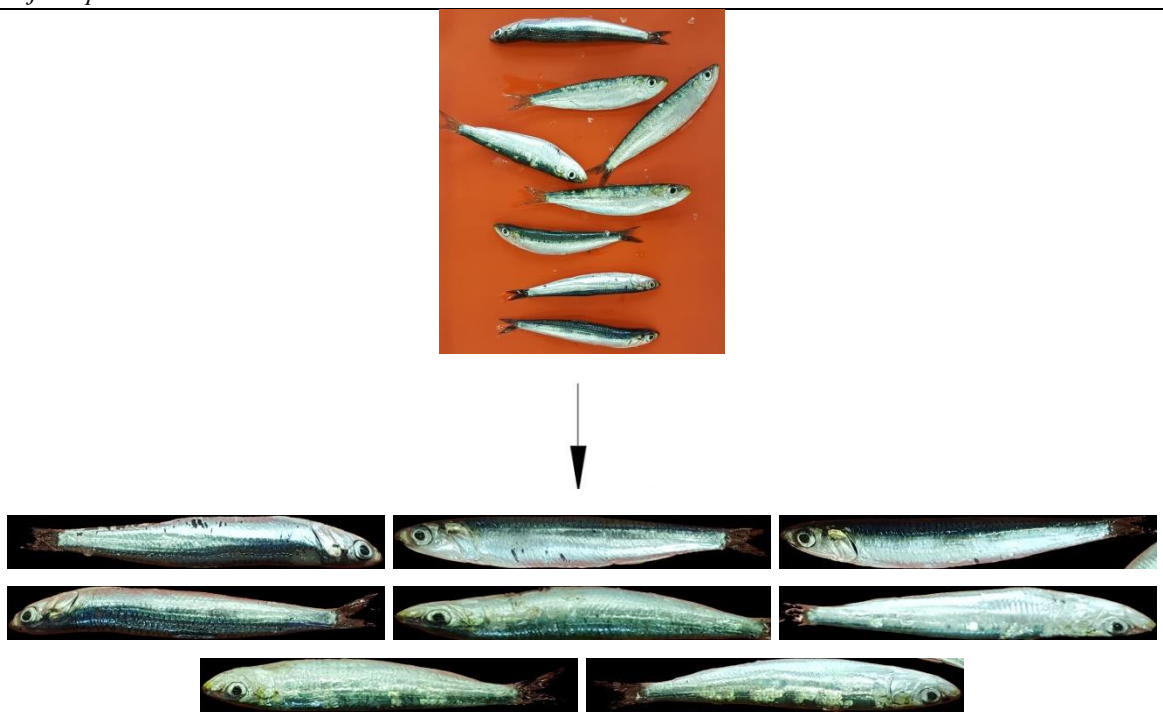
Slika 23. Neobrađena fotografija riba

Na slici se vide obje razmatrane vrste, točnije; inćuni su tri jedinke s desna i prva jedinka s lijeva, dok su ostale srdele. Usprkos tome što je fotografiranje obavljeno u prostoru velikog intenziteta svjetlom, ne dolazi do lokalnog zasićenja istim upravo zbog ispunjenja preduvjeta da izvori svjetla moraju biti ravnomjerno raspoređeni. Potvrda ispunjenja ovog uvjeta se može dokazati i slabo vidljivim sjenama koje se nalaze sa svih strana svake od jedinki. Uz sve navedeno i to što je očigledno pozadina različite boje od boja riba, na razmatranoj slici se vidi i problem slučajno razbacanih riba, a to je – preklapanje.

4.3. Pretprocesiranje slika

Nekoliko desetaka slika konceptualno istih kao slika 23 obrađuju se na način da se kao izlaz dobije niz slika jedinki, svaka posebno namještena u okvir najmanjih mogućih dimenzija i bez podloge. Kako je podloga u ovoj fazi ništa više od šuma, ista se uklanja i zamjenjuje s jednoličnom crnom bojom (vrijednost svih kanala je nula). Nadalje, sve se ribe postavljaju vodoravno i spremaju na odgovarajuću lokaciju bez svođenja na iste dimenzije.

Postavljeni zadatak se može ilustrirati i ovako:



Slika 24. Ilustracija predprocesiranja

Krećući od pretpostavke da je programska okolina (u daljnjem tekstu: environment) postavljena i da su u istoj instalirane sve primarne i sekundarne biblioteke, pristupa se izradi programskog koda koji odrađuje gore postavljen zadatak.

```
1.  from __future__ import print_function
2.  import cv2
3.  import numpy as np
4.  import random as rng
5.  from imutils import paths
6.  import crop_horizont
```

Naravno, na početku se pozivaju svi potrebni paketi. OpenCv pod imenom `cv2`, `numpy`, generator slučajnih brojeva, modul `paths` iz `imutils`-a potreban za dohvaćanje svih slika u datoteci izvornih slika i konačno funkcija `crop_horizont`.

```
7.  imagePaths = list(paths.list_images(" ## izvorne slike ## "))
8.  lower = np.array([5, 60, 50])
9.  upper = np.array([17, 255, 255])
10. threshold = 20
```

U sedmoj liniji dohvaćaju se sve putanje do svih slika u datoteci pod lokacijom `" ## izvorne slike ## "`. Dodatno, definira se nekoliko varijabli a to su dva NumPy reda i jedna cjelobrojna varijabla, koji će detaljnije biti opisani tokom korištenja u programu.

```
11. for (i, imagePath) in enumerate(imagePaths):
12.     img = cv2.imread(imagePath)
13.
14.
15.     img_hls = cv2.cvtColor(img, cv2.COLOR_BGR2HLS)
16.     mask = cv2.inRange(img_hls, lower, upper)
17.     mask = cv2.bitwise_not(mask)
18.
19.     target = cv2.bitwise_and(img, img, mask=mask)
```

Definirajući sve potrebne varijable, ulazi se u `for` petlju na način da se osigura učitavanje nove fotografije u datoteci izvornih slika svaki put kad se petlja zatvori. U petnaestoj liniji vidi se konverzija `BGR` prostora stanja u `HLS` prostor stanja. `BGR` prostor je ništa drugo doli opisani RGB prostor u poglavlju 1.1.1.2.2. s drugačijim rasporedom kanala. HLS je pak prostor koji se pobliže ilustrira u poglavlju 1.1.1.2.3. Ovom konverzijom ispunio se preduvjet eliminacije željene boje (pozadine) sa slike, što u RGB (BGR) kanalu nije moguće jer je boja rezultat spajanja tri boje u jednu. Eliminacija boje ze provodi tzv. maskom. Maska se definira gore spomenutim NumPy redovima u kojima je posebno zanimljiv prvi stupac zbog toga što se on odnosi na HUE kanal. U OpenCv numeraciji upravo raspon od 5 – 17 na HUE kanalu krije dotičnu boju pozadine. Maska nakon logičke negacije – pretvaranja crno u bijelo i bijelo u crno, za sliku 23 izgleda ovako:

Slika 25. Maska

Spajanjem maske i izvorne slike binarnim „I“, u liniji 19, dobiva se efekt da na mjestu gdje je maska crna i izlaz će biti isti zbog „I“ uvjeta koji je za izvornu sliku uvijek pozitivan (osim u slučaju potpuno crnog piksela – što onda svakako postaje irelevantno). Tamo gdje je maska bijela, na izlazu će se sačuvati piksel izvorne slike. Ovim postupkom se eliminirala podloga i `target` je u BGR formatu i nije potrebna konverzija. Vrijedi napomenuti da se zbog

zaobljenosti ribe može pojaviti efekt refleksije podloge o ribu i u tom slučaju osobitu pažnju treba obratiti na „L“ i „S“ vrijednosti u `lower` i `upper` redovima.

```
20.     src_gray = cv2.cvtColor(target, cv2.COLOR_BGR2GRAY)
21.     src_gray = cv2.GaussianBlur(src_gray, (0, 0), 1)
22.
23.     canny_output = cv2.Canny(src_gray, threshold, threshold * 2)
24.     canny_output = cv2.GaussianBlur(canny_output, (0, 0), 1)
25.
26.     _, contours, _ = cv2.findContours(canny_output, cv2.RETR_EXTERNAL, cv2.CHAIN_APP
    ROX_SIMPLE)
```

Eliminacijom podloge prelazi se na pronalaženje kontura riba. Kako funkcija `cv2.Canny()` radi nad samo jednim kanalom, slika se prebacuje u kanal sivih tonova (linija 20). Dodatno, primjenjuje se izglađivanje (zamagljivanje) slike kako bi se ublažile oscilacije u boji među susjednim pikselima. Te oscilacije su pogotovo izražene ako se radi o slici visoke rezolucije. U liniji 23 pronalaze se svi bridovi kojima intenzitet gradijenta ulazi u raspon `[threshold, threshold * 2]`. `Threshold`, korišten u funkciji `cv2.Canny()`, je definiran na početku ovog koda i do njega se došlo isključivo eksperimentalnim putem. Da se kojim slučajem nije izgladila slika raspon gradijenta bi bilo gotovo nemoguće odrediti, uz to za obradu slike potrebno bi bilo mnogo više računalnog vremena. Linija 26, zbog `cv2.RETR_EXTERNAL`, pronalazi samo vanjske zatvorene konture, za koje se pretpostavlja da su vanjske konture riba.

```
27.     contours_poly = [None] * len(contours)
28.     boundRect = [None] * len(contours)
29.     box = [None] * len(contours)
30.
31.
32.     for i, c in enumerate(contours):
33.
34.         contours_poly[i] = cv2.approxPolyDP(c, 3, True)
35.         boundRect[i] = cv2.minAreaRect(contours_poly[i])
36.         box[i] = cv2.boxPoints(boundRect[i])
37.         box[i] = np.int0(box[i])
38.
39.         area = cv.contourArea(box[i])
```

`For` petlja u liniji 32, prolazi kroz svaku detektiranu vanjsku konturu i postavlja je u pravokutnik minimalne površine (linija 35; `cv2.minAreaRect()`). U petlji se definiraju i spremaju čvorne točke pravokutnika. Linije od 27 do 29 služe isključivo za ponovnu inicijalizaciju varijabli jednom kad se `for` petlja u liniji 11 ponovi, odnosno kad na obradu dođe nova slika.

Na slici 23 se vidi da postoje konture koje su vanjske, zatvorene, a opet, nisu jedinke riba. To su najčešće riblje lustre ili više preklopljenih riba. Za razliku od [6] gdje se bilo kakve anomalije kod izdvajanja ribe od okoline mogu ručno dotjerati, u ovom slučaju ne može postojati takva opcija jer se potencijalno radi o ogromnom protoku riba kroz sustav te bi svaka ljudska intervencija usporila sustav do te mjere da on ne bi bio upotrebljiv. Stoga se taj problem rješava na alternativne načine. Prva opcija je detekcija očiju unutar pravokutnika i postavljanja uvjeta da ne smije biti više od jednog oka. Ova opcija je relativno osjetljiva na činjenicu ni jedno oko ne mora biti vidljivo od strane kamere. Druga opcija, koja je implementirana, je definiranje pojasa dopuštene površine pravokutnika. Velike površine pravokutnika ukazuju na preklopljene jedinke, dok male ukazuju na šumove na podlozi.

```
40.         if area > 50000 and area < 200000:
41.             img_crop, img_rot = crop_horizont.crop_horizont(src, boundRect[i])
42.
43.             height, width = img_crop.shape[0], img_crop.shape[1]
44.
45.             img_show = cv.resize(img_crop, (int(width * 0.5), int(height * 0.5)))
46.             cv.imshow('a', img_crop)
47.             sort = cv.waitKey(0)
48.
49.             random = rng.randint(0, 1000000)
50.             random = str(random)
51.
52.             if sort == ord('s'):
53.                 location = ' ## datoteka srdela ## ' + random + '.jpg'
54.                 cv.imwrite(location, img_crop)
55.
56.             if sort == ord('i'):
57.                 location = ' ## datoteka incuna ## ' + random + '.jpg'
58.                 cv.imwrite(location, img_crop)
59.
60.             if sort == ord('n'):
61.                 location = ' ## datoteka otpada ## ' + random + '.jpg'
62.                 cv.imwrite(location, img_crop)
63.
64.     if i == 0 or (i + 1) % 10 ==0 or (i + 1) == len(imagePaths):
65.         print('[INFO] processed {}/{}'.format(i + 1,len(imagePaths)))
```

U 40-toj liniji se provjerava uvjet površine pravokutnika i ako je on zadovoljen, poziva se funkcija koja odrezuje sliku uokvirenu pravokutnikom od originalne. Funkcija radi na način da se pravokutnik dovede u vodoravni položaj, zatim se cijela slika zakrene za isti kut i tek onda izvrši odrezivanje. Ostatak glavnog koda (linije 43 – 65) služi prikazivanju slike i, s obzirom na dani ulaz od strane korisnika, odrezana slika se separira i tako gradi početna baza podataka.

4.4. Mane osnovne baze podataka

Nakon stvaranja baze podataka kako je to opisano u poglavlju 4.3., dolazi se do čestog problema u bazama podataka za strojno učenje – manjak primjeraka jedne od klasa. Broj inćuna u osnovnoj bazi podataka je, analogno isječku na slici 1, daleko manji od broja srdela. Točnije, osnovna baza se sastoji od 49 inćuna i dosta više srdela (oko 600). Vodeći se po [7] rad s tako nebalansiranom bazom podataka nije dobar jer neuronske mreže u tom slučaju imaju preferenciju prema klasi koje više ima u bazi podataka.

4.5. Proširivanje baze podataka

Potrebno je, dakle, što više proširiti bazu inćuna. Pristupa se metodi zrcaljenja po x osi, y osi i centralnog zrcaljenja. Osim toga primjenjuju se konvolucijski filtri za izoštravanje slike i zamućivanje. Takvim načinom se iz jedne slike dobiva 12 slika.

$$N_{kon} = N_{poč}\,((1+3)*(1+2)) \tag{3}$$

Iz primarne slike (slika 26):

Slika 26. Primarna slika

Slika 27. Varijacije na primarnu sliku

dobiva se 11 dodatnih sličnih slika.

Prije samog proširivanja u posebnu datoteku se odvaja nekolicina slika (tzv. Test dataset) i, radi osiguranja pouzdanosti rješenja, taj dataset se ne proširuje. Na taj način se kod provjere točnosti klasifikacije poziva „Test“ baza podataka koja je garantirano izvorna i koja pruža najbolji temelj za provjeru. „Test“ se sastoji od 24 slike inćuna i 24 (ili više) slika srdela koje su, naravno, klasificirane. Ostatak od 25 slika inćuna i isto toliko srdela prolazi gore opisan postupak proširivanja. Svih 300 slika i inćuna i srdela se sprema u „Train“ bazu podataka u kojoj se onda nalazi 600 klasificiranih slika jedinki.

Ovim činom konačno se definirala baza podataka koja je temelj za sve radnje koje se izvršavaju s ciljem uspješne klasifikacije strojnim učenjem i računalnim vidom.

5. PRIMJENA MODELA NA ZADATAK

Nakon potpunog sređivanja baze podataka, dijeljenja iste na grupu za treniranje i grupu za testiranje i utvrđivanja točnosti cjelokupne baze može se pristupiti izradi algoritama.

Primijeniti će se tri modela klasifikacije: k-ti najbliži susjedi, konvolucijske mreže i stablo odluka.

5.1. K-ti najbliži susjedi

Model k-NN pobliže je opisan u prethodnim poglavljima, dok se u ovom koristi na tri načina. Razlikuju se, dakle, ulazi kojima se opskrbljuje ovakav klasifikator. Razmatra se slika kao NumPy red, histogrami crvene, zelene i plave kao NumPy red i HUE histogram kombiniran s omjerom duljine i širine ribe. Osim točnosti klasificiranja neki od navedenih modela imaju znatno manju tzv. matricu osobina (eng. feature matrix). Matrica osobina je izraz za cijeli ulaz u klasifikator koji, pretpostavlja se, sadrži točne podatke o osobinama klasifikacijskog subjekta.

5.1.1. K-NN sa slikom kao ulaz

Prva varijacija k-tog najbližeg susjeda je ona koja koristi sliku kao ulaznu matricu osobina. Naravno, ovakav oblik ulaza je najnepovoljniji sa stajališta memorije.

Na početku koda pozivaju se sve potrebne biblioteke od kojih su neke: SkLearn, NumPy i Keras. Osim toga pozivaju se i isprogramirane funkcije za obradu slika. Na taj način program je u stanju paralelno učitavati i obrađivati slike (svođenje na istu dimenziju) što uvelike pridonosi fluidnosti.

```
27. sp = SimplePreprocessor(500, 100)
28. sdl = SimpleDatasetLoader(preprocessors=[sp])
29. (data, labels) = sdl.load(imagePaths, verbose=100)
```

Gdje su `SimplePreprocessor` i `SimpleDatasetLoader` funkcije koje su zamišljene za procesiranje, odnosno učitavanje slika. Također, vidi se da kao izlaz osim liste slika pod varijablom `data` dolazi i pripadna lista klasa (`labels`) koja je povukla imena klasa iz imena datoteka pod kojom je klasa spremljena.

Fotografije koje su 500 x 100 x 3 potrebno je pretvoriti u NumPy red s 150000 članova.

```
48. data = data.reshape((data.shape[0], 150000))
```

Isti postupak paralelno se provodi i na dio dataseta za testiranje. Nakon završetka svih popratnih radnji uključujući i enkodiranje klasa, može se pristupiti pozivanju klasifikatora.

```
67. trainX = data
68. trainY = labels
69. testX = data_test
70. testY = labels_test
71.
72.
73. print('[INFO] evaluating k-NN clasifier...')
74. model = KNeighborsClassifier(n_neighbors=1, n_jobs=-1)
75. model.fit(trainX, trainY)
76. print(classification_report(testY, model.predict((testX)), target_names=le.classes_)
    )
```

Hiperparametri klasifikatora su namješteni da izvrši 300 koraka sa uključenim stohastičkim spuštanjem gradijenta vrijednosti 0.01.

Klasifikacijski izvještaj sada glasi:

```
[INFO] features matrix: 87.9MB
[INFO] evaluating k-NN clasifier...
              precision    recall  f1-score   support

      Incuni       0.75      0.88      0.81        24
      Srdele       0.85      0.71      0.77        24

    accuracy                           0.79        48
   macro avg       0.80      0.79      0.79        48
weighted avg       0.80      0.79      0.79        48
```

Slika 28. Klasifikacijski izvještaj (slike kao ulaz)

U izvještaju je vidljivo da, iako se radi o primitivnom klasifikatoru, se postiže relativno visoka točnost sa bazom podataka kreiranom u prethodnom poglavlju. Točnost klasifikatora je u ovom slučaju 79% dok je na inćunima klasifikacija 88, a na srdelama 71%. Sve točnosti ustvrđene su na temelju 48 slika koje nisu ni na koji način sudjelovale u treniranju mreže. Time se zaključuje da je podatak točan i da mreža, usprkos svojoj trivijalnosti, daje dobre rezultate.

Usprkos točnosti, problem leži u relativno velikoj memoriji matrice osobina koja je veličine 87.9Mb. Ako se k tome pridoda činjenica da se kod klasificiranja svaki put provjerava cijela matrica osobina, zaključuje se da proces nije optimiziran. Osobito, kod velikih baza podataka, koje bi trebale biti implementirane kod primjene ovakvog rješenja, dolazilo bi do nedopustivog kašnjenja kod klasifikacije.

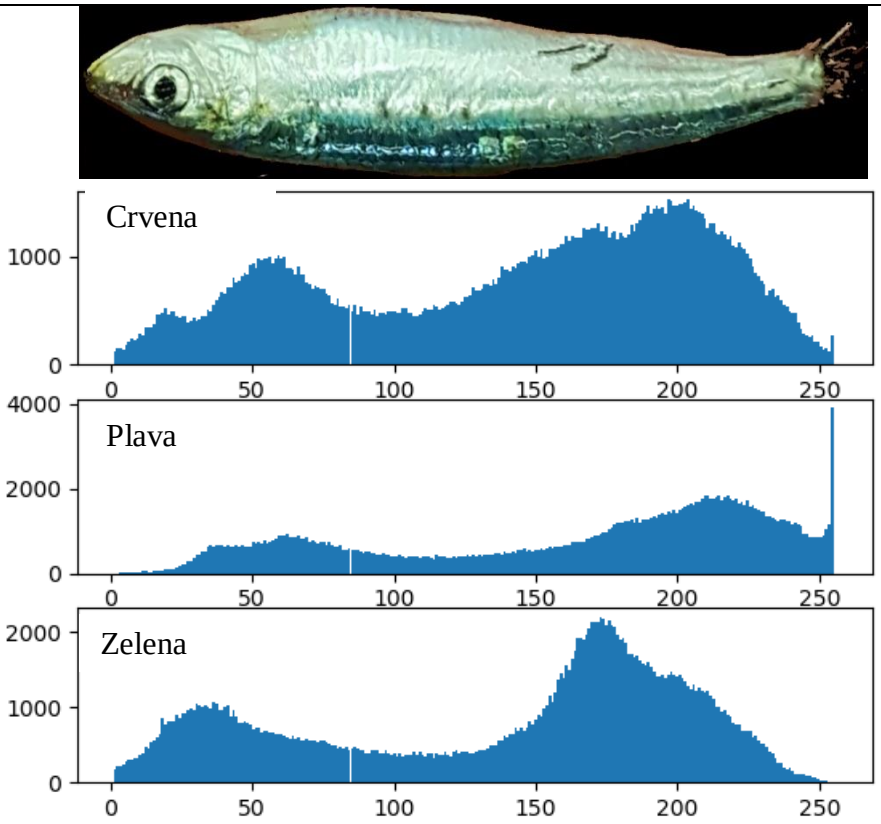
5.1.2. K-NN sa R, G i B histogramima kao ulaz

Da bi se smanjila matrica osobina koja ulazi u k-NN klasifikator, a s druge strane očuvale osobine objekata, pristupa se ekstrakciji histograma svake od slika.

```
1.  for i in range (0, len(data)):
2.      hist_b = cv2.calcHist([data[i]], [0], None, [256], [1, 256])
3.      hist_g = cv2.calcHist([data[i]], [1], None, [256], [1, 256])
4.      hist_r = cv2.calcHist([data[i]], [2], None, [256], [1, 256])
5.      hist = np.row_stack((hist_b, hist_g, hist_r))
6.      hist_data.append(hist.flatten())
```

Svaka slika koja je namijenjena za treniranje ili klasifikaciju prolazi ekstrakciju histograma. Tri histograma bivaju iznjedrena. Histogram zelene, histogram plave i histogram crvene boje. Histogrami se potom spajaju u jednu bazu gdje je njihov redoslijed nebitan jer klasifikator radi u n-dimenzionalnom prostoru koji je invarijantan na redoslijed ulaznih podataka. Iz histograma je isključena crna boja (vrijednost 0) jer se smatra da ona ne pridonosi osobinama riba, a s druge strane, zbog pozadine, ima vrlo veliku vrijednost koja može utjecati na rješenja. Takva rješenja ne bi bila osnovana na dobrim temeljima.

Za neku od srdela pripadajući histogrami su na slici 29.



Slika 29. R, B, G histogrami s pripadajućom slikom

Ovakvim podacima znatno se smanjuje matrica osobina, dok se pokušava sačuvati što više relevantnih informacija subjekta na slici.

```
[INFO] features matrix: 3.6MB
[INFO] evaluating k-NN clasifier...
              precision    recall  f1-score   support

      Incuni       0.81      0.92      0.86        24
      Srdele       0.90      0.79      0.84        24

    accuracy                           0.85        48
   macro avg       0.86      0.85      0.85        48
weighted avg       0.86      0.85      0.85        48
```

Slika 30. Klasifikacijski izvještaj (RGB histogrami)

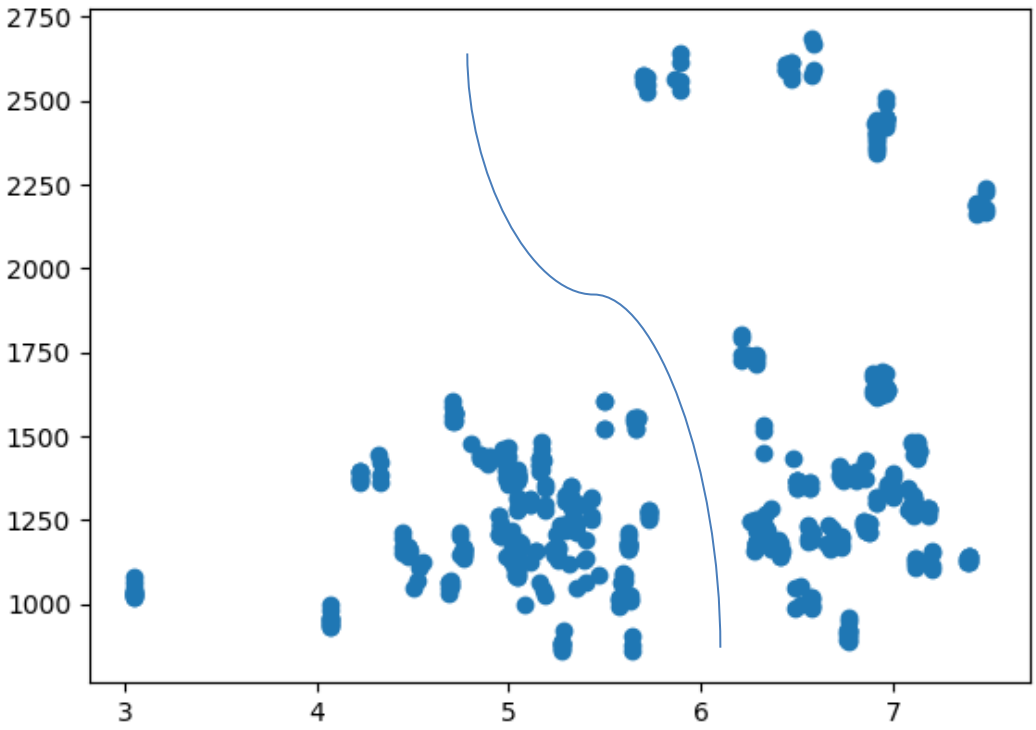
U izvještaju je vidljivo da, osim što se matrica osobina, iznimno smanjila s prethodnih 87.9Mb na 3.6Mb, točnost se povećala. Iako je očekivano da se matrica osobina smanjuje na uštrp točnosti, događa se upravo suprotno. Ovaj fenomen objašnjava se činjenicom da se slike

kao ulazi u prethodnom poglavlju trebaju svesti na istu veličinu da bi klasifikator radio. Očito se više informacija izgubilo tim postupkom nego postupkom ekstrakcije histograma gdje svođenje na iste dimenzije nije potrebno.

5.1.3. K-NN sa HUE histogramima i omjerima

Daljnji pokušaj smanjivanja matrice osobina i sačuvanja podataka je pokušaj postavljanja HUE histograma zajedno s omjerima duljine i širine na ulaz klasifikatora iz naslova. Štoviše, u ovom poglavlju ide se korak više i uzima se prosjek cijelog histograma.

Ovim postupkom matricu osobina za svaki subjekt čine samo dva broja. Iako se razumna količina podataka gubi ovakvim postupkom i dalje k-NN klasifikator uspijeva razlučiti dvije hrpe u, sada dvodimenzionalnom, prostoru koje proziva različitim klasama (slika 31).



Slika 31. Dvodimenzionalno k-NN polje

Na slici se vidi da postoji relativno vidljiva granica između dviju klasa ako se u x-y sustav unesu njihove matrice osobina.

```
[INFO] features matrix: 0.009MB
[INFO] evaluating k-NN clasifier...
              precision    recall  f1-score   support

      Incuni       0.69      0.92      0.79        24
      Srdele       0.88      0.58      0.70        24

    accuracy                           0.75        48
   macro avg       0.78      0.75      0.74        48
weighted avg       0.78      0.75      0.74        48
```

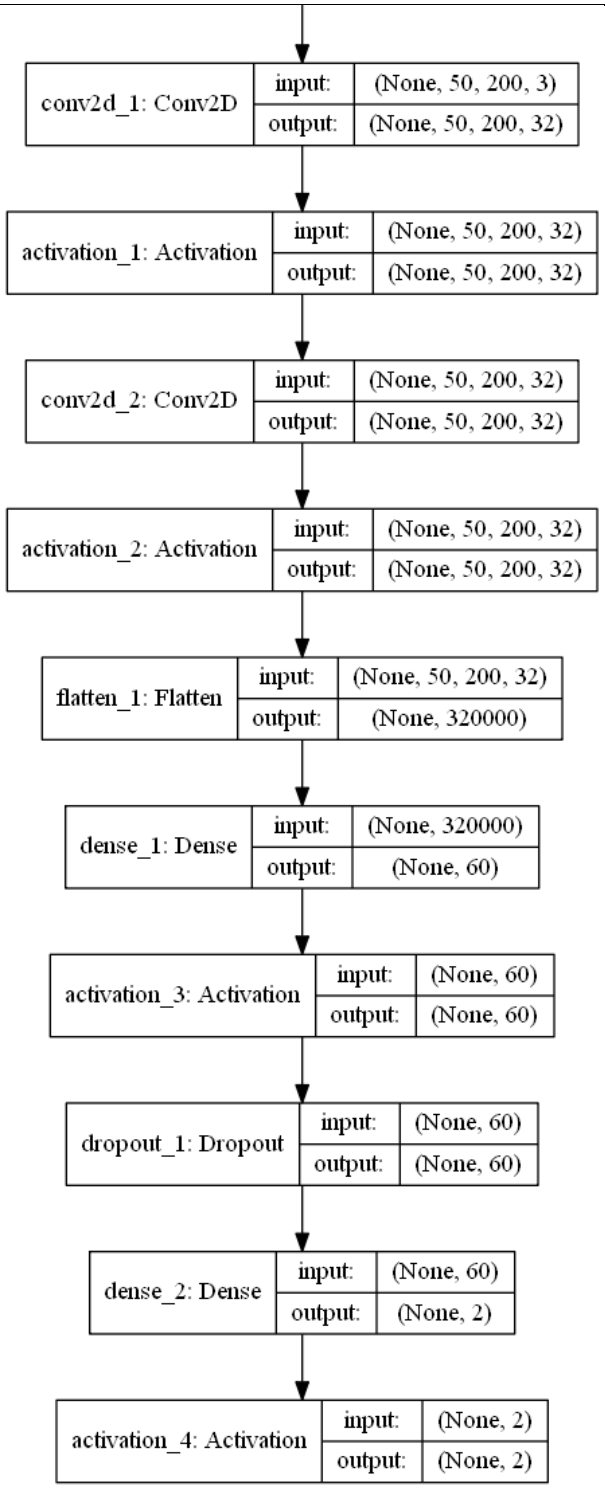
Slika 32. Klasifikacijski izvještaj (HUE i omjeri duljina)

U klasifikacijskom izvještaju primjećuje se da je točnost pala na 75%, to je i očekivano ako se u obzir uzme iznimno reducirana veličina matrice osobina koje je sad tek 0.009Mb. Ovakav klasifikator bio bi iznimno dobar ako se traži brzina klasificiranja i jednostavnost, dok točnost nije presudna.

5.2. Konvolucijska mreža

Kako se dokazuje u poglavlju 5.1., ova baza podataka je i više nego klasifitabilna. Ribe za klasifikaciju su na prvi pogled vrlo slične, pogotovo ako se ne gleda na detalje. Ako se primijeni konvolucijska mreža koja upravo filtrima (kernelima) nalazi i pronalazi oblike, bridove, karakteristične nakupine i sl., može se očekivati veća točnost od onih koje pružaju k-NN klasifikatori. Ograničavajući faktor je u ovom slučaju relativno mala baza podataka, koja po [4] mora biti razmjerno velika dubini konvolucijske mreže. Dakle, s bazom podataka od 600 slika nije se u mogućnosti ulaziti u mreže koje su preduboke poput ImageNet i sl. Ipak, u ovom poglavlju stvara se arhitektura potpuno prilagođena za problem koji se postavlja.

Arhitektura mreže izgleda ovako:



Slika 33. Arhitektura konvolucijske mreže

Ulaz u mrežu je slika 50 x 200 (x 3) nad kojom se primjenjuje, u prvom sloju, kerneli tako da se za izlaz dobiva aktivacijska mapa dimenzija 50 x 200 x 32. Potom se primjenjuje ReLu aktivacijska funkcija. Nadalje se ponovno primjenjuje samo jedan kernel tako da je iduća aktivacijska mapa jednakih dimenzija kao i prethodna. Opet, idući korak je *ReLu* aktivacijska

funkcija, nakon čega se podaci pretvaraju u red koji ulazi u potpuno umreženu mrežu. Potpuno umrežena mreža ima ugrađen sloj odbacivanja. Kako su u bazi podataka slike zrcaljene, zamagljivane i izoštravane, postoji velika mogućnost za pretreniranost mreže. Iz tog razloga dodaje se sloj odbacivanja opisan u poglavlju 3.3.2.6. Konačno, na izlazu se primjenjuje *SoftMax* aktivator. Osim ovakve arhitekture, hiperparametri mreže su: *SGD* = 0.001 i *momentum* = 0.8.

Slika 34. Proces učenja konvolucijske mreže

Točnost ovakve mreže tokom učenja dostiže 97% iz razloga što je baza podataka proširivana i uz faktor razdiobe za treniranje i verifikaciju od 50% očito je da će slične slike nad kojima je trenirano doći na test. Zbog toga model istrenirane mreže (sa svim težinama) sprema i poziva s drugim setom podataka koji je neovisan od onog nad kojim je vršeno treniranje. Na taj način dobiva se točna i neupitna uspješnost mreže.

Mreža na bazi podataka koja je izdvojena iz baze za treniranje postiže 92% točnosti.

Na slici 34 primjećuju se velike oscilacije kod gubitaka učenja i gubitaka validacije, isto tako se vide oscilacije na krivuljama točnosti ali samo u prvoj polovici. Ovakve oscilacije izazvane su slojem odbacivanja jer se u svakoj idućoj epohi odbaci čak 60% veza. Neke od tih veza u početku su dominirajuće i zbog toga se u prvoj polovici krivulje točnosti ne smiruju. Ipak, nakon dovoljnog broja epoha model postane robustan na ovoliko velik koeficijent odbacivanja, što znači da se postigla određena generalizacija modela, odnosno da model ne ovisi o samo nekoliko neurona već da svi skladno donose odluku o klasifikaciji.

Model ovakve mreže zauzima čak 150Mb memorije, iako to ne znači da je brzina klasifikacije upitna jer se samo klasificiranje provodi na potpuno drugačiji način od onog kod k-tog najbližeg susjeda. Točnost od 92%, iako primjetno veća od točnosti s k-NN-om, je ograničena veličinom baze podataka. S većom bazom podataka mogu se očekivati i daleko veće točnosti.

5.3. Stablo odluka

Posljednji u nizu klasifikacijskih modela primijenjenih na zadatak je stablo odluka. Stablo odluka je pojam za niz provjera određenih uvjeta na temelju kojih se odlučuje ili ide u niže razine odlučivanja. Za potrebe ovog zadatka nekoliko je kriterija nad kojima se donose prikladne odluke.

Podaci kao temelj za odučavanje su:

- Omjer duljine i širine ribe
- Omjer zastupljenosti plave i zelene boje
- Razlika histograma i prosječnog histograma za srdele

Da bi se mogli navedeni podaci definirati potrebno je na svakoj ribi izvršiti mjerenje njene duljine i širine, definirati HUE histograme kao i histograme plave i zelene boje. Također je potrebno nakon izvršenja svih navedenih zadataka nad bazom za treniranje pronaći i prosječan HUE histogram srdela jer se on koristi kao referentan kod usporedbe HUE histograma neklasificiranih riba. Na temelju takve usporedbe može se odrediti pripadnost klasi.

Računanje omjera duljine i širine provodi se jednostavnim mjerenjem duljina stranica pravokutnika minimalne površine kako je to opisano u poglavlju pretprocesiranja baze podataka.

Omjeri zastupljenosti zelene i plave boje se, također, jednostavno računaju iz istoimenih histograma.

Za računanje razlike histograma potrebno je prvo, nakon što se svi HUE histogrami srdela ekstraktiraju iz baze za treniranje, izračunati prosjek histograma za srdele.

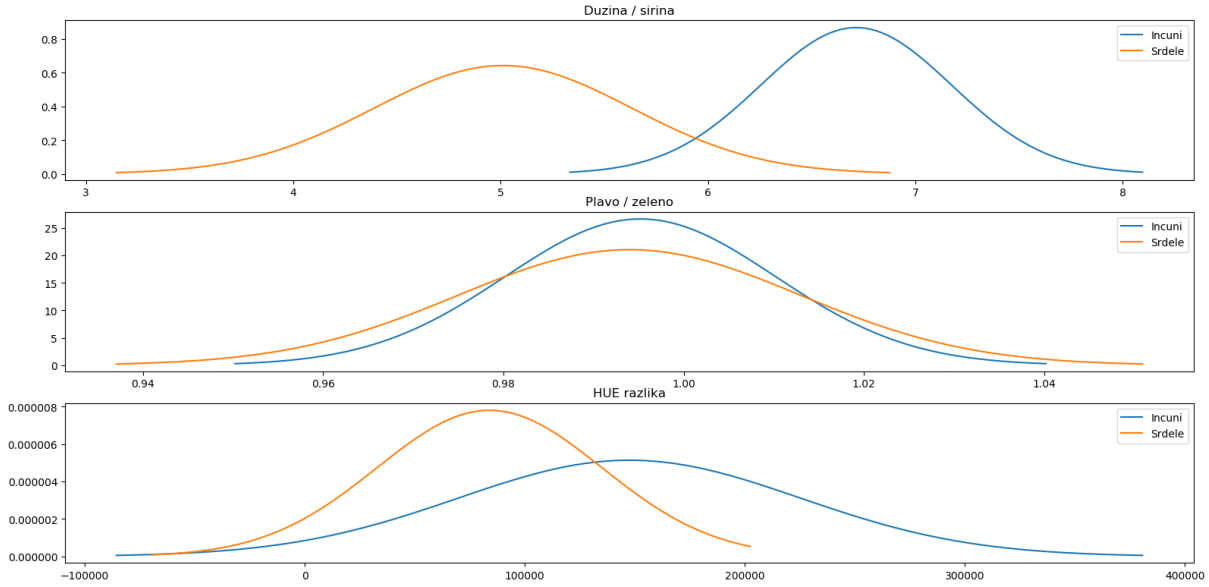
```
235.        average_hue_2 = np.average(hue_hist_data[j[0]: ], axis=0)
```

Prosjek se računa NumPy-jevom funkcijom `average()`.

```
243.        cv2.compareHist(hue_hist_data[i], average_hue_2, cv2.HISTCMP_CHISQR_ALT)
```

Razlika među histogramima je ugrađena funkcija u biblioteci OpenCv, a koja se detaljnije objašnjava u poglavlju procesiranja slika. Metoda je Chi-kvadrat.

Za definiranje stabla odluka potrebno je vizualizirati dobivene podatke i dobiti uvid u njihove razlike. Ovakvi podaci nabolje se vizualiziraju Gaussovom razdiobom, jer se osim pojasa podataka vidi i njihova zastupljenost u određenom pojasu. Gaussove razdiobe za tri relevantna podatka su:



Slika 35. Gaussova razdioba relevantnih veličina

Na temelju slike 35 može se definirati nekoliko granica, odnosno pojasa, kod odlučivanja. Tako naprimjer, za omjer duljine i širine postavlja se odluka da svi omjeri koji su ispod granice 5.35 spadaju pod klasu srdela, dok se svi oni iznad granice 6.5 proglašavaju inćunima. Pojas između dva navedena broja pada u dublje slojeve odlučivanja.

Ovakav pristup omogućuje ne samo brzinu, fluidnosti i fleksibilnost programa, već, sigurno, i određeni oblik generalizacije. Priložen model pokazuje i najveću točnost na datasetu koji ne sudjeluje u Gaussovim razdiobama a to je točnost od 96%.

Naglasak je na činjenicu da se većina odluka donese u prva dva uvjeta po hijerarhiji (omjer duljine i širine te razlika HUE histograma).

6. Zaključak

U Radu se pobliže upoznaje s metodama strojnog učenja i računalnog vida kroz primjenu istih za segmentaciju i klasifikaciju dviju vrsta riba – srdele i inćuna. Kroz prizmu zadatka uspješno je sagledano sve brže rastuće područje metoda koje, pravilno iskorištene, mogu izvršavati relativno složene zadatke, a koji su, s druge strane, čovjeku monotoni. Pod pojmom relativno složenih zadataka, zasigurno, spada i separacija vrlo sličnih ribljih vrsta kao što su vrste razmatrane u ovom radu.

Preko metoda pretprocesiranja uspijeva se doći do baze podataka koja je vrlo specifična i namijenjena samo jednoj primjeni. Otežavajući faktor je relativno mala baza podataka. Stoga se pristupa provjerenim metodama proširivanja baze podataka, uz budno oko na opasnost od pretreniranja mreže koje može nastupiti u takvim scenarijima. Klasifikacija je provedena primjenom tri različite metode: k-tim najbližim susjedom, konvolucijskom mrežom i stablom odluke. Usprkos realnoj opasnosti pretreniranja, isti fenomen se ne pojavljuje ni u kojem od navedenih modela. Točnost modela, koja međusobno varira, utvrđena je nad potpuno neovisnim skupom podataka koji ni na koji način nije sudjelovao u procesu učenja bilo kojeg modela. Na ovaj način se sa sigurnošću zaključuje da najveću točnost ima stablo odluke (96%), nakon čega slijedi točnost kod klasifikacije konvolucijskim mrežama koja iznosi 92%. Posljednja po iznosu je točnost k-tim najbližim susjedom koja dohvaća 85%.

Usprkos tome što stablo odluke ima najveću točnost, najveći potencijal zasigurno ima konvolucijska mreža koja se zbog ograničene baze podataka nije eksploatirala do svojih krajnjih mogućnosti za koje Autor smatra da mogu, uz prikladnu bazu podataka, prelaziti marginu od 99%.

LITERATURA

[1] EUMOFA. (2018). European fish market, 2018 edition.

[2] Sardina. *http://www.sardina.hr/*.

[3] Državni zavod za statistiku Republike Hrvatske. (2017). Priopćenje, ribarstvo 2017.

[4] Rosebrock, A. (2017). *Deep learning for computer vision with python.* PyImageSearch.

[5] Darwin, C. (1877). *The Different Forms of Flowers on Plants of the Same Species.*

[6] Rossi, F., Benso, A., Di Carlo, S., Politano, G., Savino, A., & Acutis, P. L. (2016). *FishAPP: a Mobile App to Detect Fish Falsification Through Image Processing and Machine Learning Techniques.* Torino: Politecnico di Torino.

[7] Weining, W., Zhao, M., & Wang, L. (2016). *A multi-scene deep learning model for image aesthetic evaluation.* Elsevier.

[8] Yazhou Yao, Jian Zhang, Fumin Shen, Xiansheng Hua, Jingsong Xu, & Zhenmin Tang. (2016). *Automatic image dataset construction with multiple textual metadata.* IEEE.

[9] *MathWorks*. Preuzeto 23. 01. 2020 iz MathWorks: htpp://www.mathworks.com

[10] Jon Louis Bentley. "Multidimensional Binary Search Trees Used for Associative Searching". ACM 18.9 (Rujan. 1975), stranice 509–517.

[11] Marius Muja, David G. Lowe. "Scalable Nearest Neighbor Algorithms for High DimensionalData". Pattern Analysis and Machine Intelligence, IEEE Transactions, stranica 36 (2014)

[12] Sanjoy Dasgupta. "Experiments with Random Projection". Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence. UAI '00. San Francisco, CA, USA:Morgan Kaufmann Publishers Inc., 2000

[13] Ella Bingham, Heikki Mannila. "Random Projection in Dimensionality Reduction: Applications to Image and Text Data". Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD '01. San Francisco, California: ACM, 2001

[14] Sanjoy Dasgupta, Anupam Gupta. "An Elementary Proof of a Theorem of Johnson and Lindenstrauss". Random Struct. Algorithms 22.1 (Sječanj. 2003)

[15] Djork-Arné Clevert, Thomas Unterthiner, Sepp Hochreiter. “Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)”. In: CoRR abs/1511.07289 (2015).

[16] Andrew Ng. Machine Learning. https : / / www . coursera . org / learn / machine - Learning.

[17] Ian H. Witten, Eibe Frank, and Mark A. Hall. Data Mining: Practical Machine Learning Tools and Techniques. 3rd. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.,2011.

[18] Peter Harrington. Machine Learning in Action. Greenwich, CT, USA: Manning Publications Co.

[19] Stephen Marsland. Machine Learning: An Algorithmic Perspective. 1st. Chapman & Hall/CRC, 2009.

[20] Stanford University. Stanford Electronics Laboratories et al. Adaptive "adaline" neuron using chemical "memistors.". 1960.

[21] Michael Zibulevsky. Homework on analytical and numerical computation of gradient and Hessian.

[22] Andrew Ng. CS229 Lecture Notes. http://cs229.stanford.edu/notes/cs229-notes1.pdf

[23] Andrej Karpathy. Optimization. http://cs231n.github.io/optimization-1/

[24] Ian Goodfellow, Yoshua Bengio, Aaron Courville. Deep Learning. http://www. deeplearningbook.org. MIT Press, 2016

[25] Hui Zou, Trevor Hastie. “Regularization and variable selection via the Elastic Net”. In: Journal of the Royal Statistical Society, Series B 67 (2005)

[26] Geoffrey Hinton. Neural Networks for Machine Learning. http://www.cs.toronto. edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf

[27] Sergey Ioffe, Christian Szegedy. “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”. CoRR abs/1502.03167 (2015).

PRILOZI

I. CD-R disc