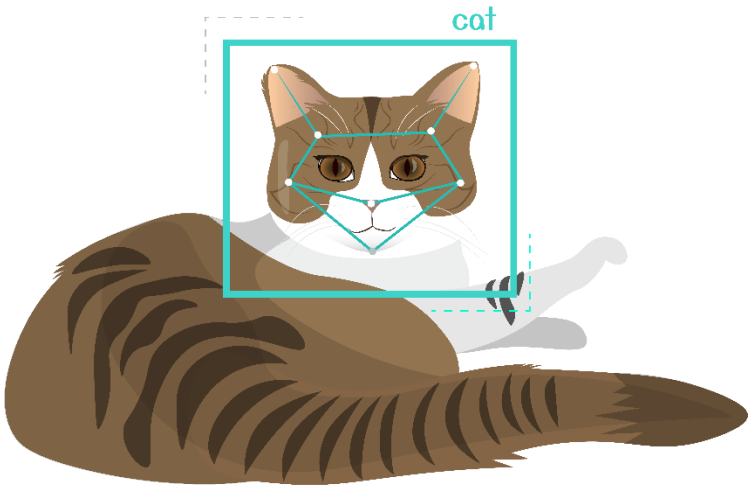
Veštačka inteligencija

Računarska vizija - prepoznavanje objekata primenom open-source biblioteka



Članovi tima ***Lispice***:

Mina Nikolić ***16786***

Mila Mirović ***16742***

Sadržaj

1. Uvod

Računarska vizija je definisana kad oblast nauke koja se bavi razvojem tehnika koje omogućavaju računarima da "vide" i razumeju sadržaj digitalnih objekata kao što su fotografije i video snimci. Problem koji računarska vizija pokušava da reši deluje jednostavno, budući da je trivijalno rešiv od strane ljudskih bića čak i na nivou vrlo mlade dece. Ipak, iako su godine rada i istraživanja dovele do bitnih pomaka u ovoj oblasti, nedostatak potpunog razumevanja načina funckionisanja biološkog vida i vizione percepcije, kao i raznolikost i dinamika fizičkog sveta dovode do toga da ostaje znatan broj nerešenih problema. Pored mnogobrojih benefita koji su nam dostupni zahvaljujući tehnološkim revolucijama tokom godina, došli smo u situaciju da stvari koje opažamo možemo da sačuvamo u digitalnoj formi uz pomoć uređaja kao što su mobilni telefoni. Upravo zbog toga smo suočeni sa činjenicom da stotine sati video i fotografskog materijala svakog minuta postaju deo brojnih internet platformi. Sadržaj internet stranica se sastoji od tekstualnog materijala i slika. Relativno je jednostavno pristupiti problemu indeksiranja i pretrage teksta, ali kako bi se pretraživale slike potrebno je da algoritam bude upoznat sa njihovim sadržajem. Taj problem je na neki način rešavan uz pomoć meta opisa koji bi morali biti ručno dodati uz svaku sliku. Alternativa takvom pristupu je upravo upoznavanje sa tehnikama računarskog vida.

Računarski vid je multidisciplinarna oblast koja bi se u širem smislu mogla definisati kao podoblast ***veštacke inteligencije*** i ***mašinskog učenja***. Upravo zbog kombinacije raznih domena nauke, neke tehnike koje se koriste su pozajmljene iz opštih inženjerskih i informatičkih oblasti. Računarski vid podrazumeva automatsku ekstrakciju informacija iz slika. Informacijom možemo da nazovemo sve što podrzumeva razne trodimenzionalne modele, pozicije kamere, detekciju objekata kao i grupisanje i pretragu sadržaja slika. *Bitno je napraviti razliku između pojmova* ***računarski vid*** *i* ***procesiranje slika****.* Procesiranje slika bi podrazumevalo proces kreiranja nove slike od postojeće, ali se posmatra kao obrada digitalnih signala gde sadržaj slike nije od značaja. Za neke primene računarskog vida je najpre potrebno procesirati sliku na određen način, a tek onda pristupiti algoritamskim aspektima analize. Takve promene se mogu ogledati u normalizaciji fotometrijskih aspekata slike kao što su osvetljenost i boja ili pak odsecanje nerelevantnih delova slike kako bi objekat od značaja bio u centru pažnje. Nekada je potrebno i baviti se redukcijom digitalnog šuma slike koji nastaje kao posledica loših uslova osvetljenja.

Problem računarske vizije se dodatno usložnjava zbog neadekvatnog razumevanja ljudskog vida. Prilikom tog proučavanja nije dovoljno uzeti u obzir samo ljudske vizuelne organe, već i shvatiti na koji se način opažanja pretvaraju u korisne informacije. Raznolikost objekata u zavisnoti od parametara, kao što su orijentacija i nivo osvetljenosti, prave dodatne poteškoće računarima čiji je domen uglavnom lokalizovan na strogo kontrolisane uslove, a ne na proizvoljnost koja je odlika stvarnog sveta.

Jedna od najpoznatijih primena računarskog vida je ***klasifikacija*** slika. Pod time se misli na određivanje pripadnosti date slike nizu predefinisanih kategorija. Posmatrajmo jednostavan problem ***binarne klasifikacije*** koji podrazumeva klasifikaciju slike prema tome da li se na njoj nalazi turistička atrakcija ili ne. Posao klasifikatora u ovom slučaju bi bio samo da odgovori na pitanje pripadnosti

kategoriji, a ne da nužno prepozna o kojoj atrakciji je reč. Još jedna primena se može posmatrati kroz lokalizaciju. ***Lokalizacija*** bi podrazumevala tačno određivanje pozicije datog objekta na slici. Standardan način izvođenja bi podrazumevao crtanje graničnog pravougaonika oko objekta koji tražimo. Korisnost lokalizacije se pogotovo ogleda u automatskom odsecanju nepotrebih delova slike kada nam je samo neki objekat od interesa, a ne celina. Pored toga, potrebno je razmotriti i domen primene prepoznavanja (detekcije) objekata. Prepoznavanje objekata bi podrazumevalo *simultanu primenu lokalizacije i klasifikacij*e o kojima je bilo reči. Međutim, treba naglasiti da detekcija može (a ne mora nužno) da funkcioniše sa varijabilnim brojem objekata na slici. Naime, moguće je u okviru iste slike prepoznati automobile, bicikle, saobraćajne znake, kao i ljude. Rešavanje ovakvog problema ponekad može biti komplikovano čak i ljudskom vizuelnom sistemu. Kompleksnost se dodatno povećava činjenicom da neki objekti mogu biti samo delimično vidljivi ili čak može doći i do preklapanja objekata koje želimo da uočimo. Veličina sličnih objekata takođe može da varira, a potrebno je voditi računa i o uslovima osvetljenja date scene. Aplikacija ovog vida računarske vizije se najčešće ogleda u brojanju. Pod time se može misliti na brojanje raznih tipova ubranog voća na plantaži, ali i na određivanje broja učesnika nekog skupa ili brojnosti ljudi u prodavnicama u datom trenutku.

Pored gore pomenutih, postoji i domen ***identifikacije objekata*** koji se malo razlikuje od detekcije objekata, iako se u implementaciji koriste slične tehnike. U ovom slučaju potrebno je naći instance datog objekta na slikama, poenta nije izvršiti klasifikaciju, već odrediti da li se objekat nalazi na slici i u slučaju potvrdnog odgovora potrebno je specificirati njegovu lokaciju. Kao primer se može uzeti monitoring slika bezbednosnih kamera u cilju identifikacije lica tražene osobe. Domen segmentacije slika se može posmatrati kao sledbenik detekcije objekata. U ovom slučaju nije neophodno samo naći objekat na slici već i kreirati masku za svaki nađeni objekat što preciznije moguće. Jedan od naprednijih domena računarske vizije je i ***praćenje objekata***. Cilj primene ove tehnike je praćenje objekata koji se u toku vremena kreću, a kao ulazni parametar se koristi kontinualni niz video frejmova. Način na koji se to najčešće ostvaruje je primena detekcije objekata na svaku sliku sadržanu u video sekvenci i upoređivanje instanci svakog objekta kako bi bilo određeno kako su se pomerili. Negativna strana ovakvog pristupa je činjenica da je izvođenje detekcije objekata za svaku pojedinačnu sliku skupo u pogledu računarskih resursa. Alternativni način bi bio da se objekat koji se prati prepozna samo jednom (prvi put kada se pojavi) i onda je potrebno vršiti tehnike praćenja bez eksplicitnog prepoznavanja istog na svakoj narednoj slici.

U ovom radu detaljnije će biti razrađen slučaj detekcije objekata, konkretnije detekcije mačijeg lica na slici ili video snimku. Model koji će se implementirati prikazaće kako se primena algoritama veštačke inteligencije, tačnije računarskog vida, može iskoristiti za realan problem kao što je detekcija frontalnog pogleda životinje u kameru, radi produkcije što boljih fotografija ili video snimaka.

2. Prepoznavanje u računarskom vidu

Kao što je u uvodnom delu pomenuto, pod prepoznavanjem možemo da podrazumevamo sledeće:

- Verifikacija objekata
- Detekcija objekata
- Identifikacija objekata
- Klasifikacija (kategorizacija) objekata

Prepoznavanje se deli u 2 kategorije:

- **prepoznavanje na nivou instanci** (*instance based recognition*) i
- **generički problem klasifikacije**, koji se smatra težim problemom.

Prepznavanje na nivu instanci je prepoznavanje pripadnosti, kao na primer prepoznavanje automobila koji je baš "Ćirin auto". U okviru prepoznavanja izdvaja se *klasifikacija objekata* koja može biti vrlo apstrakta. Ako se posmatra primer slike mačke, kategorizacija se može vršiti na nivou: vrste živih bića – ***životinja***, životinjske vrste – ***mačka***, rase životinje – ***Mejn kun mačka***, ličnog imena: ***Aurora***.

Ovaj "problem" se izučavao i rešavao u domenu psihologije u oblasti ***vizuelne klasifikacije objekata*** nad modelom ljudskog mozga i prirodne kategorizacije. *Klasifikacija je definisana kao oblast mašinskog učenja koja ima za cilj predviđanje klasnih obeležja nove instance na osnovu prethodnih opažanja*. Detaljnim proučavanjem naučnik Biderman je došao do grubog zaključka da postoji između *10.000* do *30.000* kategorija objekata, čime je dokazao da je generički problem klasifikacije izuzetno velika oblast izučavanja. Pored vizuelnih kategorija ljudski mozak je u stanju da identifikuje i ***funkcionalne klasifikacije objekata***, kao na primer *objekat na kome se može sesti* (što može biti stolica, fotelja, sofa, itd). Prepozavanje je dakle fundimentalna komponenta percepcije i rezonovanja o objektima, koja olakšava organizaciju i pristup vizuelnom sadržaju. Reč je o teškom problemu, jer na slikama objekti nisu uvek jasni i postoje različite osobine koje variraju, kao što su osvetljenje, poza objekta, zaklanjanje objekta, varijacije u okviru klase, ugao gledanja i slično. Neke stvari u oblasti prepoznavanja su naša svakodnevnica, dok su neke naprednije oblasti potpuna novina. Uzmimo kao primer očitavanje automobilskih tablica, poštanskih adresa, čekova, prepoznavanje lica, otisaka prstiju i slično. Evidentno je da su dostignuća u ovoj oblasti u velikom usponu i to zahvaljujući bazičnim fundamentima veštačke inteligencije i mašinskog (kao i dubokog) učenja.

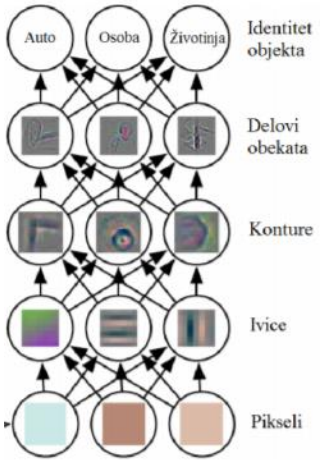
2.1. Mašinsko učenje

Mašinsko učenje je grana veštačke inteligencije koja se bavi tehnikama i metodama koje omogućavaju da računari uče na osnovu iskustva, bez eksplicitnog programiranja, tako da računar postaje inteligentan. Kao najznačajnija primena mašinskog učenja se izdvaja upravo računarski vid. ***Duboko učenje*** je relativno nova oblast mašinskog učenja i njen cilj je pomeranje mašinskog bliže veštačkoj inteligenciji. Zasnovano je na učenju reprezentacije podataka i predstavlja skup algoritama mašinskog učenja za obučavanje slojevitih struktura – slojevitih neuronskih mreža.

Primer problema koji se rešava dubokim učenjem slojevite neuronske mreže bio bi sledeći:

Pronaći sliku na kojoj se nalazi čovek, sliku na kojoj se nalazi pas i sliku na kojoj se nalazi automobil.

Ovakav problem je prilično jednostavan za čoveka, ali je sa strane računara i veštačke inteligencije je vrlo kompleksan problem. Dubokim učenjem slojevite mreže konstruišu se atributi na osnovu sirovog signala gde atributi nisu unapred poznati. Počinje se od sitnijih detalja (na najnižem nivou mreže) kao što su ivice i konture, ali se detektuju i složeniji oblici poput delova tela i lica osobe ili životinje, pa se na kraju u najvišim slojevima mreže na osnovu prethodnih informacija u prethodnim slojevima prepoznaje osoba, životinja ili predmet.



2.2. Prepoznavanje kao klasifikacija

Nadgledana klasifikacija podrzumeva postojanje kolekcije uzoraka koji su razvrstani u klase (trening podaci ili model) i cilj je predvideti klasu novih (nepoznatih) uzoraka. Potrebno je generisati dovoljno dobru funkciju koja bi vršila klasifikaciju novih uzoraka. Da li je funkcija dovoljno dobra zavisi od toga koje greške model pravi, kao i od cene pojedinih grešaka. Sa obzorom na to da klase trening podataka poznate, osnovni cilj je minimizacija očekivane greške u klasifikaciji.

Postoje 2 strategije minimizacije:

- ***Generativni pristup*** – modelovanje raspodele verovatnoća za svaku od klasa ponaosob, na osnovu čega se određuje kojoj klasi pripada konkretan uzorak.
- ***Diskriminativni pristup*** – modelovanje *granice odluke* (koje su to posmatranih klasa) između klasa, umesto nezavisnih modela klasa.

Pristup koji ćemo detaljnije obraditi je ***diskirminativni pristup***.

2.2.1. Diskriminacioni metodi za prepoznavanje

Ovi metodi su danas dominantni zbog brojnih prednosti u odnosu na generativni pristup koji stariji. Moderan svet je sa sobom doveo i činjenicu da je većina podataka vrlo visoko dimenzionalna, pa je predstavljanje kompletnih verovatnoća zahtevno. Cilj prepoznavanja nije nužno modelovanje određenih klasa, već donošenje ispravne odluke za novi uzorak. Prednost ovog pristupa je i to što sagledava složene granične slučajeve. Pri korišćenju diskiminativnog pristupa najčešće nije poznato koje fičere treba izabrati jer sam metod bira fičere i gradi odgovarajući klasfikator.

Preduslovi korišćenja diskiminativnih metoda su: *fiksni broj klasa koje su poznate unapred* i koje se neće menjati u toku rada; *ključni cilj je odrediti pravu klasu* tj. ispravno klasifikovati podatke; kroz proces mašinskog učenja težimo ka tome da dođemo do informacija o tome koji fičeri su najpogodniji za određene klase; pretpostavka je da je dostupan veliki broj trening podataka da bi došlo do napredovanja u dubokom učenju; cena pogrešne klasifikacije je ista za sve klase. Dakle, akcenat ovih metoda je fokusiranje na diskiminisanje različitih klasa, a ne primarno modelovanje, i cilja se ka tome da računar nauči koji su fičeri od značaja za konkretan zadatak.

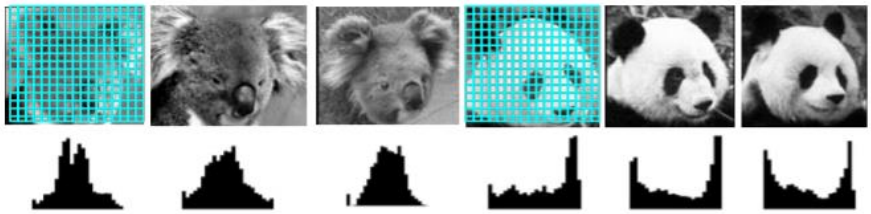
U procesu prepoznavanja kategorija postoje 2 faze: ***faza obučavanja*** i ***faza testiranja***. Faza obučavanja ima za cilj izgradnju modela tj. reprezentaciju objekata. U tom pogledu, bitno je opisati *trening uzorke* (slike) i predstaviti ih kao *fičer* [1]*vektore*. Bez trening podataka, ova faza nije moguća. Sledeći korak ove faze je *obučavanje klasifikatora*. Nakon dobijenog obučenog klasifikatora (klasifikator koji fičer vektore kalsifikuje na adekvatan način), može krenuti sledeća faza – faza testiranja. Faza testiranja je faza u kojoj se vrši generisanje kadidataka u novom uzorku (slici) i vrši se ocena klasifikatora.

[1] ***Fičer*** (*feature*) je merljivi podatak o sadržaju slike koji je jedinstven za specifičan objekat i obično govori o tome da li određeni deo slike poseduje odrećene karakteristike. Namerno se koristi naziv *fičer*, jer bukvalni prevod nije adekvatan.

2.2.1.1. Reprezentacija modela

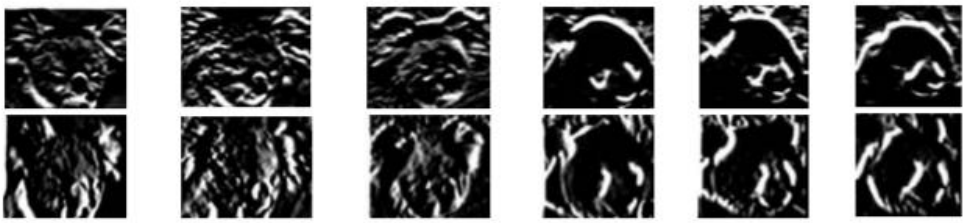
2.2.1.1.1. Modeli zasnovani na prozorima

Ovakvi modeli mogu da se posmatraju na 2 načina: ***formiranje jednostavnog holističkog opisa slike*** (ne bazira se na radu sa piskelima) i ***formiranje kadnidata metodom klizajućih prozora***. Jednostavan holistički opis sadržaja slike podrazumeva formiranje histograma intenziteta ili boja. Ukoliko



posmatramo primer slika koale i pande, uočava se da su vrednosti njihovih histograma prilično različiti, tako da za ovaj konkretan slučaj metoda histograma pokazala kao validno merilo.

Ako se umesto histograma koristi vektor intenziteta piskela i to posmatramo kao fičer vektor. Ovaj pristup je prilično problematičan, jer svaki put kada se desi pomeranje za makar jedan piksel, dobija se potpuno drugi vektor. Samim tim ovaj metod je prilično osetljiv i nije pouzdan. Sa druge strane, histogrami su osetljivi na varijacije osvetljenja, kao i varijacije u okviru same klase. Dakle, ni jedna od ovih metoda nije idealna. Zato u obzir treba uzeti ivice, konture i gradijente intenziteta:



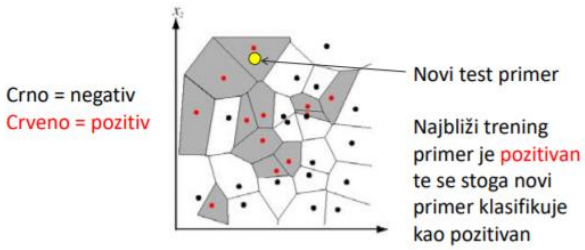
Na osnovu dobijenih rezultata možemo formirati vektore gradijenata za svaki piksel. Ideja je formirati po grupama histograme gradijenata i taj metod se naziva ***histogram orijentisani gradijenat*** (o kome će kasnije biti reči). Dobra stvar ovog pristupa je postojanje invarijantnosti na manja pomeranja i rotacije (dakle pomeraj/rotacija slike ne utiče preterano na rezultat), kao i normalizacija kontrasta koja obezbeđuje invarijantnost na promene u osvetljenju. Zaključuju se da je ova metoda dosta naprednija od prethodne dve.

Kada postoji reprezentacija modela, ulazna slika se prevodi u fičer vektor i obučava se (binarni) klasifikator. Da bi se pronašao konkretan objekat na slici, koristi se metod klizajućeg prozora. Kandidati se kreiraju tako što prozor klizi po slici sve dok ne dođe do konkretnog dela slike koji sadrži konkretan objekat.

Kod detekcije objekata, kao i kod klasifikacije, osnovna ideja je traženje granice u prostoru fičera koja na najbolji način odvaja klase. Nadalje će se razmatrati metode ***najbližeg suseda*** i ***boosting***.

2.2.1.1.2. Metod najbližeg suseda

Najjednostavniji od svih metoda i vrlo intuitivan, ali u osnovi ima mana. U fazi obučavanja se samo pamte svi mogući postojeći primeri iz trening skupa. Test tačkama se dodeljuje klasa najbližeg suseda iz trening skupa u prostoru fičera. Tačke se laberiraju bojama, na primer crno – negativno, crveno – pozitivno. Kad se pojavi novi primer za koji se određuje klasa, nalazi se najbliži primer u prostoru fičera (nakon što se odredi sam fičer vektor primera) i fičeru primera se pridodeli klasa najbližeg primera.



Voronoi particionisanje prostora fičera za 2-kategorijske 2D podatke

Koristi se Vornoi particionisanje prostora fičera u kojima postoje Vornoi ćelije koje definišu najbližeg suseda. Metod najbližeg suseda se često ne koristi u izvornom obliku već u obliku K-najbližih suseda koja ima za cilj da poveća robusnost (ne gleda samo jednog suseda, već k najbližih suseda, pa se preglasavanjem donosi odluka). Ako imamo više negativnih od pozitivnih suseda, novi uzrok će biti klasifikovan kao negativan i obrnuto.

2.2.1.1.3. Boosting

Ideja boosting metoda za obučavanje je korišćenje slabih klasifikatora (nisu dovoljno moćni da ispravno klasifikuju sve uzorke u start). Inicijalno svim trening uzorcima se dodeli ista težina i u svakoj boosting rundi se pronalazi slabi klasifikator koji postiže najmanju težinsku trening grešku (neće raditi idealno, ali nalazi najbolji za zadati skup primera). Nakon toga, uočavanjem grešaka, klasifkator loše klasifikovane primere treba težinski povećati tako da se u narednom koraku tretirati drugačije. Postupak se ponavlja dok ne dobijemo linearnu kombinaciju slabih klasifikatora, gde je težina slabog klafikatora direktno proporcionalna njegovoj tačnosti.

2.2.1.1.4. Viola Jones detektor lica

Algoritam za rapidnu detekciju objekata pomoću boosting kaskada. Prvenstvena namena je detekcija ljudskih lica, ali se može primeniti na detekciju bilo kog objekta. Zasniva se na korišćenju ***Haar*** fičera, ***AdaBoost*** algoritma i ***integralnih*** *slika*. U procesu učenja koriste se ***pozitivne*** i ***negativne*** oblasti, gde su pozitivne one na kojima se nalazi objekat od interesa, a negativne su one na kojima se nalazi sve ono što nije objekat od interesa. Ovo će biti detaljnije opisano u nastavku.

Algoritam se pojavio 2001. godine u radu *Rapid Object Detection using a Boosted Cascade of Simple Features* čiji su autori Paul ***Viola*** i Michael ***Jones***. Ovaj algoritam bio je poprilično revolucionaran kada se pojavio i vrlo brzo ušao u komercijalnu primernu. Pored naučnih doprniosa, bio je široko primenljiv.

Zahvaljujući njemu pojavili su se fotoaparati koji detektuju lice ili čak i osmeh osobe. Iako ga mnogi smatraju zastarelim, on i dan danas neverovatno brzo vrši detekciju objekata. Karakteriše ga vrlo visoka stopa otkrivanja, brzo izvršavanje (najmanje 2 okvira u sekundi) i detekcija lica (ne prepoznavanje) jer je cilj klasifikacija lica od ne-lica. Glavne ideje koje se tiču ovog algoritma su:

1) Predstavljanje lokalne teksture pomoću pravougaonih fičera koji se vrlo brzo računaju, a nalaze se u okviru prostora od interesa
2) Biranje diskriminacionih fičera tako da budu slabi klasifikatori i da se njihova kombinacija koristi kao konačni klasifikator
3) Formiranje ***kaskade*** klasifikatora koja obezbeđuje brzo odbacivanje negativnih primera, a da se malo više truda uloži da bi se utvrdilo da je nešto lice

Algoritam posmatra mnogo manje podregije i pokušava da nađe lice tražeći specifične karakteristike u svakoj podregiji. Potrebno je proveriti mnogo različitih položaja i skala jer slika može da sadrži mnogo lica i to različitih veličina. U OpenCV-u Viola-Jones se bazira na ***Haar*** fičerima. Na slici su haar fičeri (pravougaoni), gde imamo 2 oblasti: **bele** – pozitivne i **crne** – negativne. Vrši se sabiranje piksela ispod obe oblasti (i bele i crne) pa se te vrednosti oduzmu i generiše se izlaz. Inovacija koja je uvedena jesu *integralne slike* koje se vrlo jednostavno računaju linearnim prolazom (svaki piksel u slici se menja zbirom piksela od početka do te tačke). Nije potrebno čak ni skaliranje slika, jer skaliranje fičera postiže isti efekat. Šema koja se koristi je ***AdaBoost*** koja obezbeđuje određivanje optimalnih fičera kako bi se kreirao klasifikator.

Na sici je dat pseudo-kod algoritma Viola Jones sa Haar fičerima:

Algorithm: Viola-Jones Face Detection Algorithm

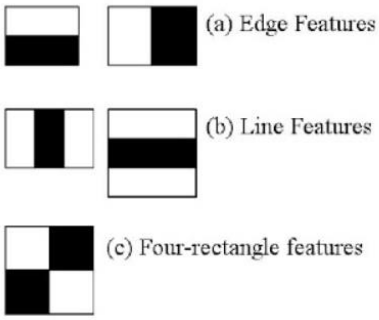
```
1: Input: original test image
2: Output: image with face indicators as rectangles
3: for i ← 1 to num of scales in pyramid of images do
4:     Downsample image to create image_i
5:     Compute integral image, image_ii
6:     for j ← 1 to num of shift steps of sub-window do
7:         for k ← 1 to num of stages in cascade classifier do
8:             for l ← 1 to num of filters of stage k do
9:                 Filter detection sub-window
10:                Accumulate filter outputs
11:            end for
12:            if accumulation fails per-stage threshold then
13:                Reject sub-window as face
14:                Break this k for loop
15:            end if
16:        end for
17:        if sub-window passed all per-stage checks then
18:            Accept this sub-window as a face
19:        end if
20:    end for
21: end for
```

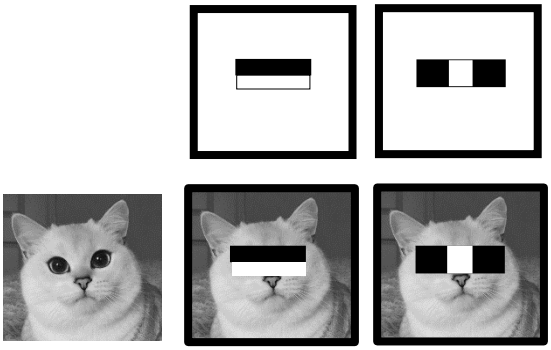
Sam algoritam ima 4 faze:

- Izbor HAAR fičera
- Kreiranje integralne slike
- Trening pomoću AdaBoost-a
- Kaskadni kalsifikatori

Treba detaljnije objasniti glavne činioce ovog algoritma. Krenimo od ***Haar fičera***. Mađarski matematičar Alfred Haar definisao je koncepte Haar talasa koji su u suštini jedna sekvenca funkcija koje su takozvanog „kvadratnog oblika“. Naučnici Viola i Jones su uvidevši ovo dostignuće, prilagodili svoju ideju i upotrebili Haar talase, te tako razvili ranije pomenute ***Haar fičere.*** To su digitalne slikovne karakteristike koje su izuzetno korisne u prepoznavanju objekata. Naime, nečije lice (kako među ljudima, tako i među mačkama), poseduje neke univerzalne karakteristike kao što je tamnija boja kod očiju, svetlija boja njuške, specifičan oblik glave i slično. Viola Jones algoritam definiše krajnje jednostavan princip razaznavanja svetlijih ili tamnijih regiona, a to je sumiranje vrednosti piksela oba takva regiona i njihovo međusobno upoređivanje. Zbir vrednosti piksela u tamnijoj regiji biće manji od zbira piksela u svetlijoj regiji. Sve ovo navedeno se postiže pomoću Haar fičera.

U svom radu, naučnici Viola i Jones nabrajaju 3 vrste Haar fičera: fičeri ivica (za otkivanje ivica), fičeri linija (za oktivanje linija) i četvorostrani fičeri (za otkivanje dijagonalnih obeležja).





Vrednost fičera računa se kao zbir vrednosti piskela u crnoj oblati umanje za zbir vrednosti piksela u beloj oblasti. Haar fičeri daju veliku vrednost kada se područja u pravougaonicima dosta razlikuju, što je ujedno pokazatelj da je taj Haar fičer korisan.

Sledeće što treba detaljnije obraditi su ***integralne slike***. Pošto je potrebno da bi za svaki fičer iskalkulisali vrednost izvršiti proračune na svim pikselima unutar tog određenog fičera, dolazimo do zaključka da neretko ovi proračuni mogu biti vrlo intenzivan. Integralna slika omogućava brzo izvođenje tih intenzivnih kalkulacija i nastala je u oblasti struktura podataka. Koristi se kao brz i efikasan način izračunavanja zbira vrednosti piksela na slici ili pravougaonog dela slike. Na integralnoj slici vrednost svake tačke je jednak sumi svih piksela iznad i levo uključujući i ciljni piksel. Ovako se štedi mnogo vremena pri računanju.

Original slika

0	1	1	1
1	2	2	3
1	2	1	1
1	3	1	0

→

0	1	2	3
1	4	7	11
2	7	11	16
3	11	16	21

Integralna slika

Sabiranjem piksela na označenom pravougaoniku dobijamo 8, što sa druge slike možemo dobiti (posmatrajući temena prethodnog pravougaonika) sabiranjem 21 sa 1 i oduzimanjem toga sa zbirom vrednosti 2 i 11, što je opet 8.

Original slika

0	1	1	1
1	2	2	3
1	2	1	1
1	3	1	0

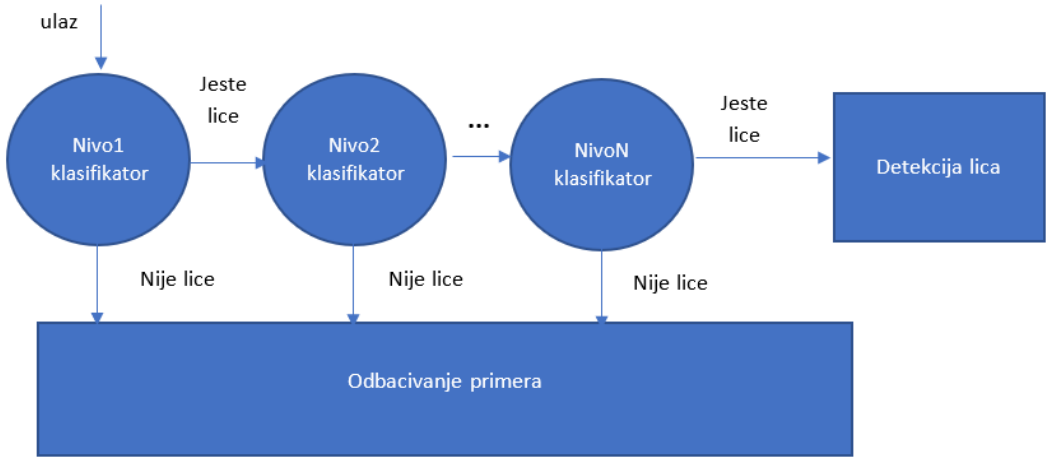
0	1	2	3
1	4	7	11
2	7	11	16
3	11	16	21

Integralna slika

Pogledajmo kako se ***AdaBoost*** koristi u algoritmu Viole Jones. Koristimo ga da bi identifikovali najbolje karakteristike u prilično velikom broju fičera (preko 160.000). AdaBoost proverava performanse svih klasifikatora koje mu se isporuče da bi odredio vrstu i veličinu fičera koji ulazi u konačni klasifikator. Kada se vrši treniranje AdaBoost-a za pronalaženje važnih karakteristika, potkrepljuje se trening podacima pa se vrši osposobljavanje za učenje, a na kraju algoritam utvrđuje da li se nešto može klasifikovati kao korisni fičer ili ne. Pri proceni klasifikatora (procena se vrši na svim podoblastima svih slika iz skupa trening podataka) neke podoblasti slika će dati jak odgovor u klasifikatoru i te oblasti će biti proglašene pozitivnim (što znači da klasifikator smatra da ta oblast sadrži lice/objekat od interesa), u suprotnom će biti proglašene negativnim. Konačan rezulatat je snažan klasifikator koji sadrži slabe klasifikatore ali koji su se pokazali kao najbolji.

Na kraju, objasnimo ***kaskadne klasifikatore***.

Posebna ideja Viola-Jones algoritma koja se izdvaja je ***formiranje kaskada*** koje obezbeđuju vrlo brzo odbacivanje negativnih primera. Smatra se izuzetno važnom i revolutivnom idejom. Kada se vrši detekcija objekata na slici (na primer lica) metodom pomerajućeg prozora, gotovo uvek postoji nešto što nije taj objekat u prozoru. Ceo postupak bi znatno efikasnije radio ako se vrši brza detekcija i odbacivanje onoga što nije traženi objekat (lice). Naravno, mora postojati određeni nivo sigurnosti da bi postojala tvrdnja da nešto nije traženi objekat (lice). Pristup koji seo vde koristi je upravo pristup kaskadnog klasifikatora, gde se klasifikator kreira kao kaskada pojedinačnih nivoa koji nam uvek daju sa jedne strane da li je nešto lice ili nije lice, sa druge strane. Pojašnjenje se može videti na slici:



Ako je nešto detektovano kao lice prelazi se na sledeći nivo, a ukoliko je okarakterisano kao da nije lice direktno se odbacuje taj primer. Tu se vidi logika brzog odbacivanja, koja može da se vrši već u prvom stadijumu. Formira se kaskada sa malom greškom u detekciji negativnih primera na početku i primenjuju se manje precizni ali brzi klasifikatori koji odmah eliminišu prozore koji očigledno nisu pozitivni.

Zaključujemo da je ovo jedan od najbitnijih algoritama u računarskom vidu za detekciju objekata i to u realnom vremenu. Treniranje jeste prilično sporo, što nije veliki problem jer se taj postupak vrši jednom, a detekcija je izuzetno brza. Sve dok je detekcija brza, nije veliki problem utrošiti više vremena na treniranje. Pojavom ovog algoritma, doplo je do pojave novih gore pomenutih ideja, kao što su ***integralne slike*** za brzo određivanje fičera, ***boosting*** za izbor fičera i ***kaskadni klasifikator*** za brzo odbacivanje negativnih prozore.

Mana ovog pristupa je što u toku prepoznavanja lika na video snimku, pri pomeranju objekta (pomeranju lica) može doći do neprepoznavanja lika zbog ugla koji ne odgovara naučenom tj. fičeri nisu tolerantni na rotaciju.

3. Veštačke neuronske mreže

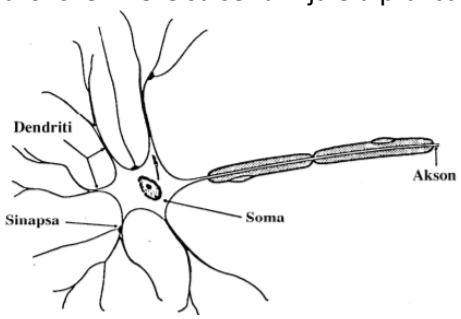
Neuronska mreža je veza većeg broja veštačkih neurona, koji su povezani na način tako da se izlazi jednih dovode na ulaze drugih neurona. One su trenutno najzastupljenije u oblasti mašinskog učenja. U tom pogledu, neuronske mreže se zasnivaju na analogiji sa mozgom gde se akcenat daje na kognitivnu sposobnost učenja i generalizaciju stečenog znanja.

Do neuronskih veštačkih mreža došlo se idejom da se oponašaju prirodni humani i animalni nervni sistemi i to u 2 smera: da se akvizicija znanja ostvaruje u procesu obučavanja (učenja) i da se to znanje skladišti u težinama veza tj. jačinama veza između neurona u mreži (koji u prirodi čine sinapse).

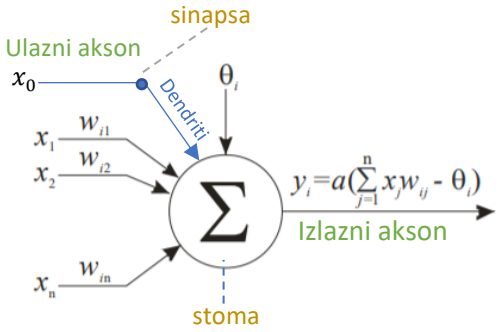
Korišćenjem i povezivanjem osnovnih neurona formira se veštačka neuronska mreža koja je u stanju da prepoznaje obrasce u zadatoj kolekciji podataka i da izgrađuje model za te podatke.

3.1. Model veštačkog neurona

Osnov veštačke neuronske mreže je model veštačkog neurona koji predstavlja jednostavan procesni element koji obavlja jednostavnu matematičku funkciju koja opisuje kako neki tip prirodnih neurona radi. Veštačke neuronske mreže su nastale kao pokušaj modelovanja neurona u mozgu, ali veštački neuron je naravno na mnogo nižem nivou od prvog neurona. Neuronske mreže su se razvijale u pravcu principa softverskog inženjerstva i optimizacionih metoda, pa su samim tim dosta drugačije od biolških. Pravi neuroni sastoje se od ***some*** ili tela neurona, ***dendrita*** koji služe da prihvate signale iz drugih neurona, ***aksona*** koji prenosi aktivnost neurona i grana se tako da izlaze vodi na ulaze (dendrite) drugih neurona i ***sinapsi*** koje predstavljaju spojeve između neurona i imaju ključnu ulogu u čitavom nervnom sistemu.



Veštački neuron se predstavlja grafički pomoću **tela** (koji ukazuje na sumiranje ulaznih vrednosti), **ulazne grane** (koje nose vrednosti označene sa x i sa težinama pojedinačnih ulaza w), **ulaznu granu Θ** koja zapravo pomera centar posmatranja levo/desno (drugačije se naziva bias, prag ili okidanje) i **izlaz** koji predstavlja matematičku funkciju. Ovakav neuron u veštačkoj neuronskoj mreži je matematička i grubo uprošćena aproksimacija biološkog neurona.



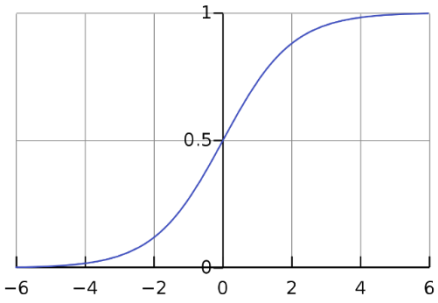
Izlaz neurona dobijamo kada kroz aktivacionu funkciju *a* propustimo razliku sume svih proizvoda svakog ulaza sa njegovom težinom i Θ.

3.2. Aktivacione funkcije neurona

Aktivaciona funkcija ili prenosna funkcija određuje vrednost izlaza neurona. To su funkcije neophodne da bi mreža imala veće mogućnosti, kao i da aproksimira čak i nelinearne funkcije. Mogu biti:

- ***Linerana funkcija*** – najjedonstavnija funkcija koja sumira sve ulaze u neuron direktno ih propuštajući na izlaz:
$$a(x) = x$$
- ***Odskočna (step ili Heaviside-ova) funkcija*** – funkcija koja zapravo simulira rad prirodnih neurona i obezbeđuje neuronu 2 stabilna stanja – aktivno i neaktivno:
$$a(x) = \begin{cases} 1, x \geq 0 \\ 0, x \leq 0 \end{cases}$$
Prva neuronska mreža se sastojala od jednog neurona i nazvana je perceptron koji kao funkciju aktivacije ima upravo step funkciju, pa uči podešavanjem težina koje odgovaraju ulaznim vrednostima na osnovu greške treninga. Glavna mana perceptrona je što ne može lako da se kombinuje u mrežu, jer step funkcija ima oštre promene što je čini nediferencijabilnom u svim tačkama i ne može se pronaći gradijent potreban za treniranje.
- ***Linearna odsečena funkcija*** – funkcija koja predstavlja varijantu odskočne funkcije koja ulazi u zasićenje posle određenog vremena
$$a(x) = \begin{cases} -1, x \leq -\frac{1}{k} \\ kx, -\frac{1}{k} < x < \frac{1}{k} \\ 1, x \geq \frac{1}{k} \end{cases}$$

- ***Sigmoidalna (logistička) funkcija*** – funkcija koja ima diferencijabilne karakteristike, te je samim tim bitna za obučavanje veštačkih neuronskih mreža. Pored toga ova funkcija je matematički kontinualna, nelinearna i neopadajuća. Naravno, ne treba ignoritsati činjenicu da kontinualnot sigmoidne funkcije može stvoriti konfuziju o tome šta neuron zapravo klasifikuje. U velikim mrežama sigmoidna funkcija stvara zasićenje i neuroni drastično usporavaju bilo kakvu promenu vrednosti.

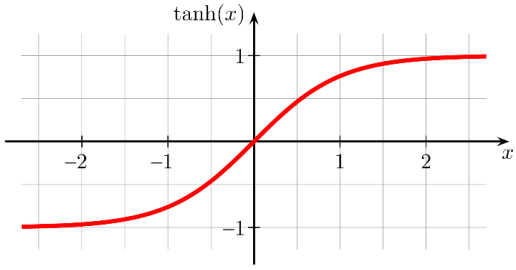


$$a(x) = \frac{1}{1 + \exp(-kx)}$$

Domen sigmoidne funkcije je R i ova funkcija taj domen preslikava u domen (0, 1) tako da su veliki pozitivni brojevi preslikani u brojeve blizu 1, a veliki negativni brojevi u brojeve blizu 0.

- **Tangens hiberbolički**

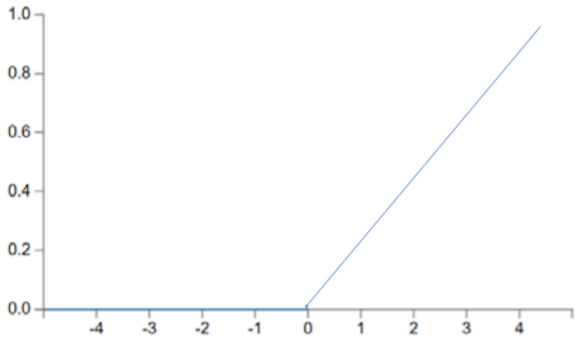
Ova funkcija uzima vrednosti u intervalu (-1, 1) što znači da je izlaz simetričan u odnosu na nulu i u praksi se ova funkcija preferira u odnosu na sigmoidnu. Pošto važi $\tanh(x) = 2\partial(2x) - 1$, hiperbolički tangens je samo skalirana sigmoidna funkcija. Problem sigmoidne funkcije postoji i ovde, pa može doći do preranog prekidanja učenja.



$$a(x) = \tanh(kx) = \frac{\exp(kx) - \exp(-kx)}{\exp(kx) + \exp(-kx)}$$

- ***Rectified Linear Unit (ReLU)*** – *ispravljačka linearna jedinica* – funkcija koja se danas koristi znatno više od svih ostalih aktivacionih funkcija. Nije diferencijabilna, ali poseduje pogodnosti u vidu optimizacije. Najveća primena se ogleda u konvolutivnim mrežama čije je obučavanje i po nekoliko puta brže u odnosu na ostale aktivacione funkcije. Dakle, osnovna prednost ove funkcije je ta što je za njeno proračunavanje potrebno znatno manje vremena. Jednostavnija je za implementaciju jer je potrebno implementirati samo linearnu funkciju, za razliku od sigmoidne i tangensa koje su složenije. Uočeno je veliko poboljšanje performansi u praksi, u odnosu na prethodne funkcije.

$$ReLU(x) = \begin{cases} x, & x > 0 \\ -\delta x, & x \leq 0 \end{cases}, \qquad \delta \approx 0$$



Kao univerzalni aproksimator neprekidnih funkcija, neuronske mreže mogu da imaju više vrsta neurona:

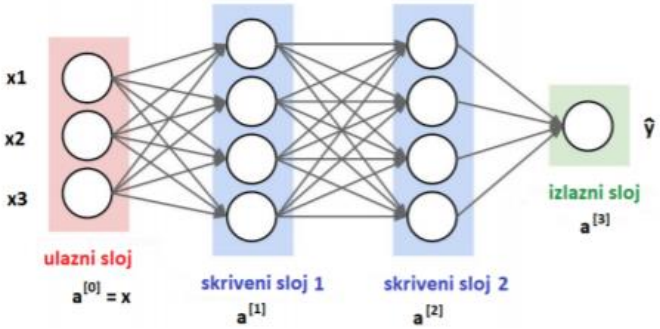
- ***ulazni neuroni*** – njihovi ulazi su ulazi u mrežu i sloj u kome se nalaze je ulazni sloj
- ***izlazni neuroni*** – njihovi izlazi su izlazi iz mreže tj. rezultati obrade i sloj u kome se nalaze je izlazni sloj
- ***skriveni neuroni*** – prosleđuju svoje izlaze drugim neuronima i sloj u kome se nalaze je skriveni sloj

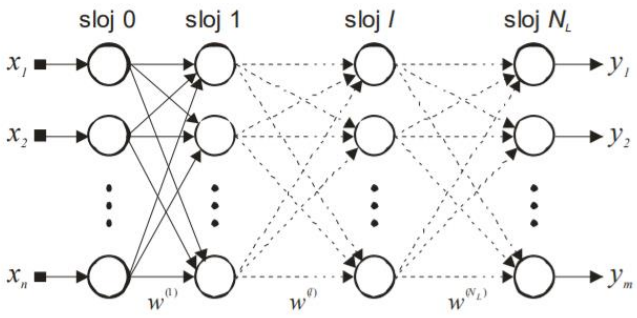
Postoje 2 glavne klase prema načinu povezivanja neuronskih mreža koje se danas paralelno koriste:

- ***sa povratnim vezama*** (*rekurentne*) – mreže sa memorijom tj. mreže koje pamte stanje. Izlaz zavisi od redosleda ulaznih podataka. Koriste se kod prevođenja nekog jezika na drugi jezik, kod sintetizacije zvuka na osnovu teksta i kod prepoznavanja glasa.
- ***bez povratnih veza*** (feed-*forward*) – mreže kod kojih izlaz zavisi isključivo od tekućeg ulaza. Najkorišćenija vrste su slojevite neuronske mreže.

Slojevite neuronske mreže su mreže kod kojih neuroni formiraju slojeve. Na ulaz jednog neurona dovode se izlazi svih neurona iz prethodnog sloja, dok se njegov izlaz vodi na ulaze svih neurona u narednom sloju.

Na slikama su dati primeri, redom, slojevite neuronske mreže sa 2 skrivena sloja i univerzalne slojevite mreže:



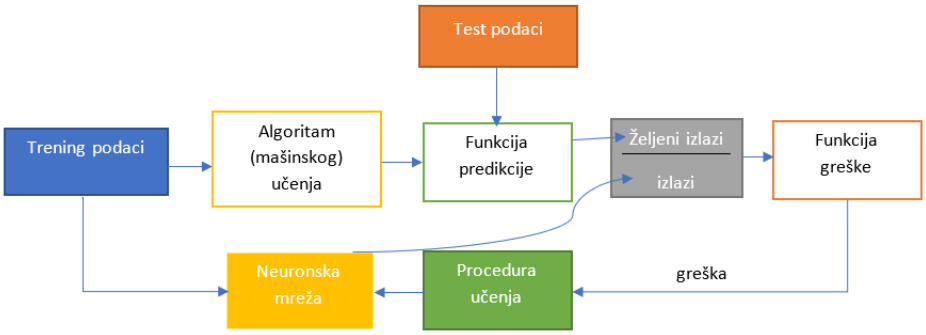


3.3. Obučavanje neuronskih mreža

Najbitnija karakteristika neuronskih mreža je da imaju sposobnost da se obučavaju tj. da (mašinski) uče. Mreža može da uči na osnovu informacija iz svog okruženja i to iterativno kroz izmenu težina i pragova u svakoj iteraciji. Mreža nakon svake iteracije praktično ima više znanja. Generalna definicija učenja (koja se ne odnosi striktno na neuronske mreže) se zapravo potpuno ukplapa u definiciju obučavanja neuronskih mreža. Ponašanje neuronske mreže zavisi od arhitekture mreže, broja neurona i težina veza između neurona, pa se obračunavanje neuronske mreže najčešće svodi na promenu težina veza između neurona. Obučavanje može bit ***nadgledano*** (supervised), ***nenadgledano*** (unsupervised), ***polunadgledano*** (semi-supervised) i ***učenje sa podrškom*** (reinforcement), u zavisnosti od toga da li su poznati očekivani izlazi mreže.

3.3.1. Nadgledano obučavanje

Glavni činioci ovakvog obučavanja su: **ulazi** tj. trenirani skup podataka; **učitelj** tj. skup podataka (data set) u kojem je definisana logika koja na osnovu odabranog ulaza tumači željene izlaze; sama **neuronska mreža** koja na osnovu ulaza generiše izlaz; **funkcija greške** koja procesira željeni izlaz i izlaz iz neuronske mreže i kao izlaz daje grešku; **procedura učenja** u koju se dovodi sračunata greška na određeni način menja težine u mreži, da bi sledeći put taj izlaz bio približniji željenom izlazu.



Cilj nadgledanog učenja je da se na osnovu poznatih parova podataka (*trening podataka*) odredi funkcija koja preslikava ulazne podatke u željene izlaze. Pre učenja, ta funkcija se naziva hipoteza ili funkcija predikcije. Funkcija predikcije se u toku obuke modela menja i optimizuje tako da što tačnije izvrši preslikavanje ulaznih podataka u željene izlazne vrednosti. Potrebno je dobiti model koji će davati tačne rezultate kako nad podacima koji su korišćeni pri obuci modela, tako i nad nepoznatim (*test*) podacima koji nisu korišćeni prilikom obuke. Ova sposobnost modela zove se ***sposobnost generalizacije***. Osnovni problemi nadgledanog učenja:

- *problem klasifikacije* – slučaj kada je vrednost izlaza diskrenta kategorijska vrednost
- *problem regresije* – slučaj kada je vrednost izlaza kontinualna vrednost

Problem klasifikacije ogleda se u razvrstavanju nepoznate instance u jednu od unapred ponuđenih kategorija tj. klasa. Dok se problem regresije predstavlja kao problem u kojem pored vrednosti svojstvava svakoj instanci odgovara i neka neprekidna numerička vrednost koju je potrebno predvideti.

Backpropagation algoritam je svojom pojavom stvorio veliki skok u popularnosti veštačkih neuronskih mreža, jer je omogućio obučavanje slojevitih neuronskih mreža. Ovaj algoritam proces nadgledanog obračunavanja svodi na minimizaciju funkcije greške (pomoću srednje kvadratne greške):

$$\varepsilon = \frac{1}{2}\sum_{i=1}^{m}(y_i - t_i)^2$$

Vrednost greške je upola manja od sume kvadrata razlike dobijenog izlaza mreže y_i i željenog izlaza mreže t_i. Način kojim se vrši modifikacija težina je:

$$w_{ij}^{(l)} = w_{ij}^{(l)} - \eta\frac{\partial\varepsilon}{\partial w_{ij}^{(l)}}$$

Tako da je nova težina jednaka razlici prethodne vrednosti težine i proizvoda koeficijenta brzine obračunavanja sa parcijalnim izvodom greške po toj težini. Na ovaj način poznato je koliko težina zapravo utiče na samu grešku. Pronalaženje parcijalnog izvoda funkcije greške radi se lančanim pravilima za razbijanje parcijalnih izvoda.

$$\frac{\partial \varepsilon}{\partial w_{ij}^{(l)}} = \frac{\partial \varepsilon}{\partial v_j^{(l)}} \cdot \frac{\partial v_j^{(l)}}{\partial w_{ij}^{(l)}} = \frac{\partial \varepsilon}{\partial y_j^{(l)}} \cdot \frac{\partial y_j^{(l)}}{\partial v_j^{(l)}} \cdot \frac{\partial v_j^{(l)}}{\partial w_{ij}^{(l)}}$$
$$= \frac{\partial \varepsilon}{\partial y_j^{(l)}} \cdot a_j^{(l)\prime}(v_j^{(l)}) \cdot y_i^{(l-1)}$$

Ono što nakon prve faze izvođenja ostaje nepoznanica je $\frac{\partial \varepsilon}{\partial y_j^{(l)}}$, pa koristeći srednju kvadratnu grešku možemo izvesti sledeću jednakost za poslednji sloj: $\frac{\partial \varepsilon}{\partial y_j^{(N_L)}} = y_j - t_j$.

Za slojeve pre poslednjeg sloja koristi se sledeća formula:

$$\frac{\partial \varepsilon}{\partial y_j^{(l)}} = \sum_{k=1}^{N_{l+1}} \frac{\partial \varepsilon}{\partial y_k^{(l+1)}} \cdot a_k^{(l+1)\prime}(v_k^{(l+1)}) \cdot w_{jk}^{(l+1)}, \quad l = 1,\ldots,N_L - 1$$

Algoritam se zove *backpropagation* jer se sastoji iz 2 faze:

- faza prostiranja unapred kada se računaju aktivacije i izlazi kroz celu mrežu do kraja
- faza prostiranja unazad kojom se računaju vrednosti za modifikovanje težina od poslednjeg sloja ka prvom.

Pri obučavanju neuronskih mreža može doći do *problema uklapanja*. Naime, može doći do overfitting-a tj. preteranog obučavanja na treniranim podacima, kojim se izgradi jako precizan model, ali nažalost ne vrši generalizaciju adekvatno za nepoznate podatke. Ovaj problem javlja se kada ima previše neurona što stvara isuviše moćan model prevelikog kapaciteta, samim tim i slojeva. Suprotan problem ovom je underfitting tj. nedovoljno obučavanje, gde model prosto ne odgovara podacima. Danas je aktuelno mišljenje da što više podataka ima, manje su šanse da se problemi ove vrste jave. Postoje razne tehnike kojima se suzbija overfitting, a najznačajnija je regularizacija.

Regularizacija je tehnika kojom se sprečava overfitting čak i kada postoje moćni modeli i prevelik broj neurona. Osnovna ideja za realizaciju regularizacije je izbegavanje velikih vrednosti težina neuronske mreže jer bi male promene na ulazu dovele do velikih promena na izlazu. Način sprečavanja ovog problema je uračunavanje velikih težina u grešku.

$$C = C_0 + \frac{\lambda}{2n} \sum_{\omega} \omega^2$$

Vrši se sumiraje kvadrata svih težina jer što je težina veća, kvadrat iste je znatno veći, pa će to proizvesti veliku vrednost koju će algoritam obučavanja pokušati da minimizuje. Još jedan način sprovođenja regularizacije je glasanje. Pošto se pri treniranju neuronske mreže kreće od slučajnih vrednosti (jer ako su sve težine iste, izlaz svih neurona bio bi isti, pa se mreža ne bi obučavala) tako da se pri svakom treniranju dobija drugačiji model. Ideja je izvršiti treniranje većeg broja mreža koristeći nasumične vrednosti, pa onda odrediti izlaz preglasavanjem.

4. Konvolucione neuronske mreže

Računarski vid kao oblast poslednjih godina obuhvata i vizuelno raspoznavanje, u šta spada klasifikacija slika i video zapisa i detekcija objekata. Razvoj konvolucionih mreža doveo je do neverovatnog napretka u rešavanju ovih izazova, pa upravo zato ove mreže čine ključni deo algoritama mašinskog dubokog učenja.

Problem klasifikacija slika ima ***fazu obučavanja*** i ***fazu eksploatacije***. Faza obučavanja sastoji se od *trening slika* i odgovarajućih *kategorija* tih slika, pri čemu se vrši ručna ekstrakcija *fičera* iz slika, kao osnovni princip rada kod konvencionalnog računarskog vida.

Fičer (*feature*) je merljivi podatak o sadržaju slike koji je jedinstven za specifičan objekat i obično govori o tome da li određeni deo slike poseduje odrećene karakteristike. Fičer može specifična boja ili određeni oblik, ivice ili neki segment na slici. Dobar (kvalitetan) fičer se koristi pri razlikovanju objekata jednix od drugih. Ako se na primer za fičer uzme točak, on predstavlja dobru razliku koja jasno razdvaja na primer pse od bicikle, ali predstavlja i lošu ako se isti fičer primeni na razlikovanju bicikle od motocikla. Osnovni cilj je transformisati sirove podatke tj. sliku u vektor fičera čime algoritmu za učenje pokazujemo kako da nauči karakteristike objekta.

No vratimo se na problem klasifikacije slika i njenu fazu obučavanja. Nad dobijenim *ekstraktovanim fičerima* vrši se *testiranje klasifikatora* koji fičere klasifikuje na određeni način. Rezultat faze obučavanja je *istrenirani klasifikator*. Pri pojavi nove ulazne slike, ponovo se vrši ekstrakcija fičera, a odgovarajući fičer vektor se propušta kroz prethodno dobijeni klasifikator. Rezultat ovog postupka je *prediktovana klasa*. Kvalitet klasifikatora zavisi od kvaliteta izabranog fičera.

Metod ručne ekstrakcije fičera je bio vrlo dominantan ranije. Ideja konvolucionih neuronskih mreža je da umesto da se fičeri ručno definišu, pokušamo da ih naučimo ekstrakovanju na osnovu podataka. Postoje sukcesivni slojevi koji vrše obradu i svaki sloj u hijerarhiji ekstraktuje fičere na osnovu izlaza prethodnog sloja. Kreće se od piksela ka samim klasama.

Konvolucione neuronske mreže (CNN) su višeslojne višestruke neuronske mreže koje se sastoje od jednog ili više konvolucionih slojeva. Ove mreže se intenzivno koriste u obradi signala poput zvuka i slike, kao i teksta. Ne zahtevaju ljudski angažman u definisanju relevantnih svojstava signala, već same ustanovljavaju koja svojstva su od koristi tako što uče adekvatne filtere. Iza konvolucionih slojeva obično se nalaze i *pooling slojevi* koji smanjuju veličinu slike, a pri tom poveća dimenziju fičera, pomoćnu odgovarajućih matematičkih funkcija. Na izlazu mreže se obično nalazi potpuno povezana slojevita neuronska mreža koja vrši klasifikaciju i uči nad atributima koje prethodni slojevi konstruišu. Naziv *konvolutivne* potiče od činjenice da ove mreže uče filtere, čijom konvolutivnom primenom detektuju određena svojstva signala.

5. Osnovni principi primene funkcija iz biblioteke OpenCV

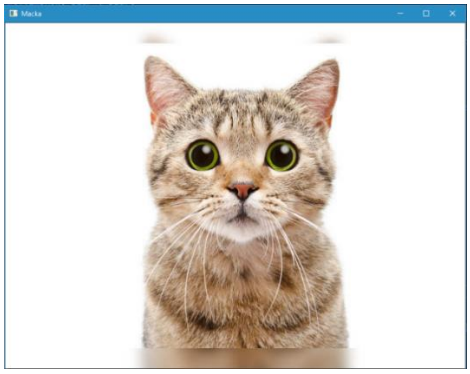
Za praktičnu primenu principa računarske vizije je korišćen ***OpenCV***. OpenCV je open-source biblioteka za računarsku viziju, mašinsko učenje i procesiranje slika koja igra veliku ulogu u operacijama koje zahtevaju izvršenje u realnom vremenu. Korišćenjem ovog alata moguće je procesirati slike i video materijal, identifikovati lica, rukopis, kao i veliki broj drugih objekata. Biblioteka je dostupna za akademsku i komercijalnu upotrebu, podržan je rad na operativnim sistemima *Linux*, *Mac OS* i *Windows* (moguć je rad i na nekim mobilnim sistemima). Omogućava rad na raznim programskim jezicima poput *C++* -a, *C*-a, *Jave* i *Python*-a. OpenCV je instaliran zadavanjem komande:

```
pip install opencv-contrib-python
```

pri čemu je korišćena verzija *Pythona 3.8.7*. Editor koji je izabran za rad je *Visual Studio Code*. Kako bi se prikazale osnovne funkcionalnosti ove biblioteke, najpre je potrebno učitati i prikazati sliku. To se može uraditi na sledeći način:

```
import cv2 as cv
img = cv.imread('photos/macka3.jpg')
cv.imshow('Macka', img)
```

Pozivanjem Python skripte u kojoj se kod nalazi dobija se sledeći prikaz:



U zavisnosti od namene slike nad kojom se vrše operacije, moguće je izvršiti brojne modifikacije. Česta modifikacija koja se vrši je prebacivanje slike u *greyscale* i to može biti postignuto na sledeći način:

```
gray= cv.cvtColor(img, cv.COLOR_BGR2GRAY)
```

Još jedan vid modifikacije koji se može izvršiti je zamućenje slike (*blur*) koji je vrlo praktičan u primenama gde je neophodan rad sa materijalima osetljive prirode. Zamućenje slike se postiže uz pomoć niskopropusnog filtera jezgra. Zapravo funkcioniše tako što uklanja sadržaj visoke frekvencije (kao što su šumovi i ivice). Na ovaj način dolazi i do zamućenja ivica (međutim postoje tehnike gde je moguće to izbeći). OpenCV koristi četiri glavne tehnike zamućivanja, a to su:

Averaging pri čemu je neophodno pozvati funkcije `cv.blur()` i `cv.boxFilter()`.

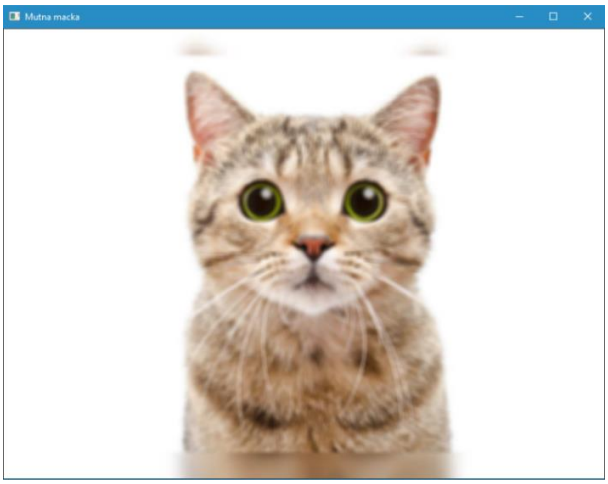
`Gaussian Blurring` gde se umesto box filtera koristi Gausovo jezgro. Vrednosti za visinu i širinu jezgra bi trebale biti pozitivne i neparne (na donjem primeru (7,7)). Potrebno je specificirati i standardnu devijaciju u pravcu X i Y ose, *sigmaX* i *sigmaY* respektivno. Ako je samo *sigmaX* specificirano, vrednost za *sigmaY* se uzima tako da je ista kao ona. Ako su obe date kao nule, onda se računaju počev od veličine jezgra.

`Median Bluring` uzima srednju vrednost svih piksela ispod oblasti jezgra i centralni element se menja ovom vrednošću. Ovakav pristup je vrlo efikasan prilikom uklanjanja šumova tipa *salt-and-peper*.

`Bilateral Filtering` se koristi kada je potrebno ukloniti šumove uz očuvanje oštrine ivica. Posledica toga je da je primena ove operacije sporija nego gore navedene.

Ukoliko bi se koristile manje vrednosti za jezgro, prikaz slike bi bio jasniji, dok veće vrednosti daju i veći efekat zamućenja.

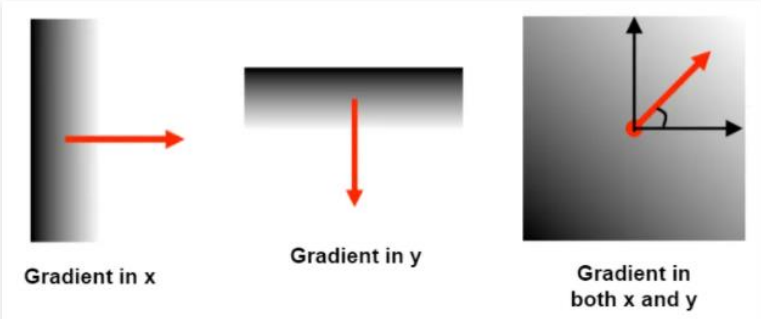
```
blur = cv.GaussianBlur(img, (7,7), cv.BORDER_DEFAULT)
```



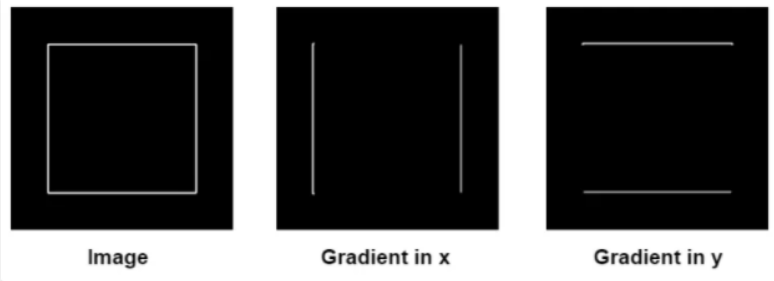
Za detekciju ivica se jako često koristi algoritam pod nazivom ***Canny***. U pitanju je algoritam koji prolazi kroz veći broj faza prilikom izvršenja:

Redukcija šuma – Budući da je detekcija ivica podložna šumu na slici, prvi korak je ukloniti šum korišćenjem Gausovog filtera dimenzija 5x5.

Nalaženje intenziteta gradijenta slike – Gradijent nam pruža dve vrste informacija. Prva se odnosi na ***magnitudu***, a druga na ***pravac***. Pravac gradijenta govori nam pravac najvećeg povećanja dok magnituda predstavlja brzinu porasta u tom pravcu. Gradijent slike po x-osi meri horizontalne promene u intenzitetu, dok gradijent po y osi meri vertikalne promene intenziteta.



Budući da su ivice nagle promene vrednosti intenziteta tako će se najveće vrednosti gradijenta pojaviti upravo preko njih. X-gradijent će naći vertikalne ivice, dok će y-gradijent biti zadužen za horizontalne ivice kao što je prikazano na donjoj slici.

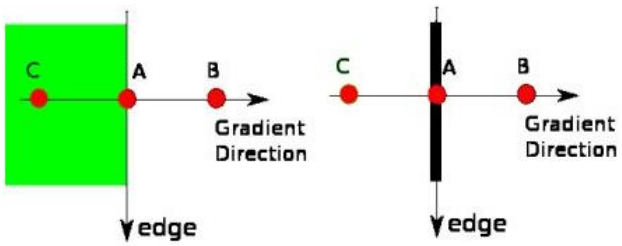


Pravac gradijenta je uvek normalan na ivice. Formula po kojoj se gradijent ivice može izračunati je sledeća:

$$Edge\_Gradient\ (G) = \sqrt{G_x^2 + G_y^2}$$

$$Angle\ (\theta) = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$

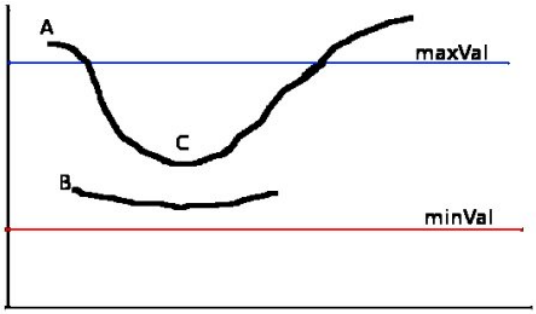
Nemaksimalno suzbijanje (*Non-maximum Suppresion*) – Nakon računanja magnitude i pravca gradijenta, radi se skeniranje slike kako bi se uklonili svi neželjeni pikseli koji ne predstavljaju ivicu. Za svaki piksel se proverava da li on predstavlja lokalni maksimum u svom okruženju u pravcu gradijenta (gledaju se njegovi levi i desni susedi).



Tačka A se nalazi na ivici (u pravcu vertikale). Kao što je ranije naglašeno, pravac gradijenta je normalan na ivicu. Tačke B i C se nalaze u pravcu gradijenta. Proverava se da li tačka A formira lokalni maksimum u odnosu na tačke B i C. U slučaju da ga formira, tačka se koristi u narednoj fazi algorirtma, u suprotnom njena vrednost postaje nula.

Prag histereze (*Hysteresis Tresholding*)

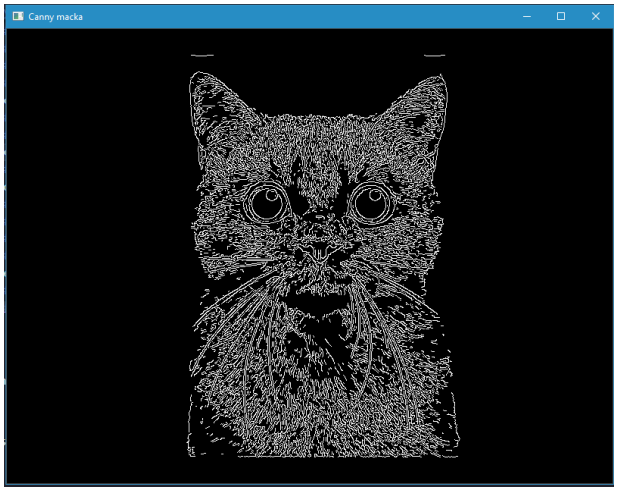
U ovoj fazi se ustanovljava koje ivice su zapravo ivice, a koje nisu. Zarad ove provere su nam potrebne dve vrednosti praga, *minVal* i *maxVal*. Svaka ivica čiji je intenzitet gradijenta veći od maksimalne vrednosti će biti detektovana kao ivica, a ona čija je vrednost ispod minimalne vrednosti će biti odbačena. Ivice koje se nalaze između vrednosti datih pragova će biti klasifikovane ili kao ivice ili kao ne-ivice u zavisnosti od povezanosti. Ako su povezane sa "*sure-edge*" pikselima onda se tretiraju kao deo ivice, u suprotnom se odbacuju.



Ivica A je iznad *maxVal* vrednosti tako da se smatra sigurnom ivicom. Iako je ivica C ispod *maxVal* vrednosti, ona je povezana sa ivicom A tako da se smatra validnom ivicom. Ivica B je iznad minimalne vrednosti praga i u istom regionu je kao ivica C, ali nije povezana ni sa jednom sigurnom ivicom tako da se odbacuje. Kako bi rezultati bili adekvatni, vrlo je bitno izabrati odgovarajuće vrednosti za minimalni i maksimalni prag. Nakon ove faze se dobijaju jake ivice na slici.

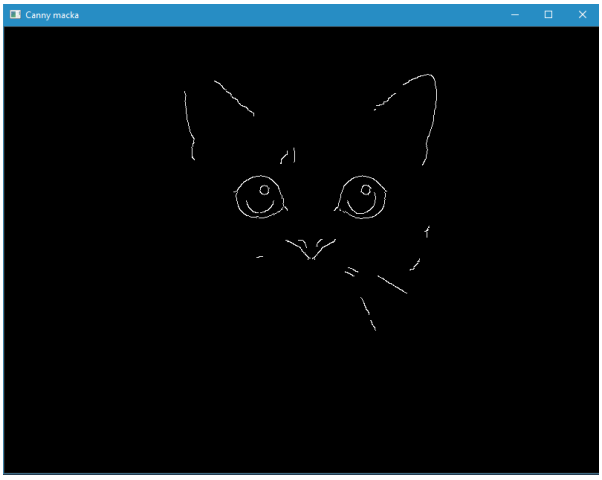
Pozivom sledeće funkcije dobija se novi prikaz mačke pri čemu su u fokusu njene ivice:

```
canny = cv.Canny(img, 125, 175)
```



U ovom slučaju je kao parametar funkcije prosleđena slika mačke bez prethodnih modifikacija. Ukoliko bismo želeli prikaz sa manjim brojem ivica, pre toga je potrebno izvršiti blur funkciju nad originalnom slikom i onda proslediti tu sliku kao parametar canny funkcije.

Rezultat je dat u nastavku:



Pored navedenih operacija za modifikaciju slika, postoje i ***morfološke operacije***. Kao što im i samo ime sugeriše, morfološke operacije predstavljaju set operacija koje procesiraju slike na osnovu njihovog oblika. Najjednostavnije morfološke operacije su ***erozija*** i ***dilatacija*** (*Erosion and Dilatation*). Njihova primena se ogleda u tome da smanje šum na datoj slici, nađu intenzitet ispupčenja ili udubljenja kao i da izoluju pojedinačne elemente slike (ali mogu i da spoje različite). Dilatacija se najpre ogleda u konvoluciji (savijanju) slike A sa nekim jezgrom (B), koje može imati proizvoljnu veličinu i oblik, mada se najčešće radi o kvadratu ili krugu. Jezgro B ima definisano sidro (*anchor point*) koje najčešće predstavlja centar jezgra. Skeniranjem jezgra B preko slike računamo maksimalnu vrednost piksela preklopljenu sa B i menjamo piksel koji se nalazi u sredini sa tom maksimalnom vrednošću. Ova operacija maksimiziranja dovodi do toga da svetli delovi slike "rastu" (odatle i naziv *dilatacija*).

Formalna reprezentacija je data sledećom formulom:

$$\texttt{dst}(x,y) = \max_{(x',y'):\, \texttt{element}(x',y') \neq 0} \texttt{src}(x+x', y+y')$$

Dat je prikaz polazne slike i njene modifikacije nakon primene dilatacije:

Na osnovu ovog prikaza se moze zaključiti da se povećala bela oblast slova oko crnih regiona koji se nalaze u pozadini.

Erozija je operacija koja je bliska dilataciji. Njena primena se ogleda u računanju lokalnog minimuma preko oblasti jezgra. Skeniranjem jezgra B preko slike računamo minimalnu vrednost piksela preklopljenu sa B i menjamo piksel koji je ispod piksela u sredini sa tom minimalnom vrednošću. Sada svetli delovi slike postaju tanji, dok se crna regija iza slova povećava. Formalna reprezentacija je data sledećom formulom:

$$\texttt{dst}(x,y) = \min_{(x',y'):\, \texttt{element}(x',y') \neq 0} \texttt{src}(x+x', y+y')$$

Dat je prikaz polazne slike i njene modifikacije nakon primene dilatacije:

Još jedan bitan koncept prilikom rada sa ovom bibliotekom su ***konture***. Konture se mogu objasniti kao krive koje povezuju kontinualne tačke (duž granice) koje imaju istu boju ili intenzitet. Konture su korisne prilikom analize oblika kao i kod detekcije i prepoznavanja objekata. Kako bi se postigli bolji rezultati pogodno je koristiti binarne slike. Dakle, pre nalaženja kontura se preporučuje primena *treshold* funkcije ili detekcija ivica uz pomoć *canny* funkcije. Način rada treshold funkcije je takav da se vrednost svakog piksela upoređuje sa vrednošću zadatog praga. Ako je vrednost piksela manja od praga onda se njegova vrednost postavlja na nulu. U suprotnom se postavlja na maksimalnu vrednost (generalno 255). Prag je vrednost koja ima dva regiona tj. vrednosti ispod zadate i vrednosti koje su iznad. Treshold tehnika se koristi uglavom za separaciju objekta za koji se smatra da je u prvom planu od njegove pozadine. Za primenu ove tehnike je potrebno konvertovati sliku u grayscale režim.

```
If f (x, y) > T
   then f (x, y) = 0
else
   f (x, y) = 255

where
f (x, y) = Coordinate Pixel Value
T = Threshold Value.
```

Postoji nekoliko verzija Simple Tresholding tehnika i to su TRESH_BINARY, TRESH_BINARY_INV, TRESH_TRUNC, TRESH_TOZERO, TRESH_TOZERO_INV. Pozivom navedene funkcije se dobija sledeći prikaz mačke:

```
ret, thresh = cv.threshold(gray,125,255, cv.THRESH_BINARY)
```

Vrednost 125 predstavlja prag, 255 je maksimalna vrednost piksela dok je treći parametar izbor odgovarajuće treshold tehnike.

Nakon primene ove funkcije, moguće je naći konture zadate slike. Funkcija koja će biti primenjena je *findContours*. Potpis ove funkcije čine tri parametra, a to su slika nad kojom će biti izvršena, način na koji će se vratiti konture i metod aproksimacije kontura. Nakon izvršenja funkcije za nalaženje kontura, broj kontura koji bi bio pronađen u primeru gore navedene mačke je 286.

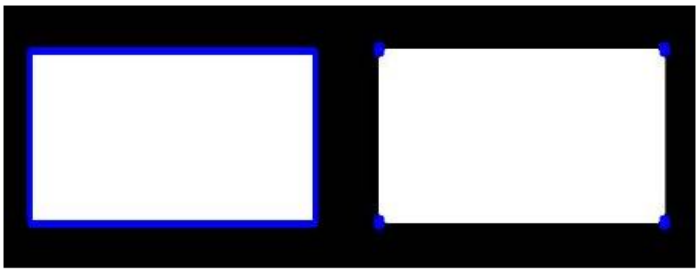
```
contours, hierarchies = cv.findContours(thresh, cv.RETR_LIST, cv.CHAIN_APPROX_NONE)
```

Ukoliko bismo kao izvornu sliku koristili onu koja je modifikovana canny funkcijom, a ne tresholdingom, za broj kontura bismo dobili vrednost 547.

```
contours, hierarchies = cv.findContours(canny, cv.RETR_LIST, cv.CHAIN_APPROX_NONE)
```

U contours promenjivoj se čuva python lista koordinata koje su nađene na slici, dok su hijerarhije način reprezentacije kontura kada imamo više ugnježdenih objekata.

RETR_LIST parametar bi značio da želimo da koristimo sve konture slike, a ne samo eksterne. U sličaju da želimo samo njih, to bi bilo moguće prosleđivanjem parametra RETR_EXTERNAL. Postoje dve metode aproksimacije kontura koje su najčešće korišćene, a to su CHAIN_APPROX_NONE i CHAIN_APPROX_SIMPLE. Razlika između ovih metoda može biti jednostavno uočena na primeru linije. Prva metoda bi vratila koordinate svih tačaka linije, dok bi druga vratila samo koordinate početka i kraja što je u slučaju linije poželjnije. Razlika između ovih metoda može biti uočena i na sledećoj slici pri čemu CHAIN_APPROX_NONE vraća 734 tacke, dok CHAIN_APPROX_SIMPLE vraća samo 4.



Nad slikama je moguće vršiti i transformacije u vidu translacije i rotacije. Translaciju slike je moguće izvršiti primenom sledeće funkcije:

```
def translate(img, x, y):
    transMat = np.float32([[1,0,x],[0,1,y]])
    dimensions= (img.shape[1], img.shape[0])
    return cv.warpAffine(img, transMat, dimensions)
```

Promenljiva *transMat* je matrica transformacije pri čemu x označava pomeraj po x-osi, dok y označava pomeraj po y-osi. Dimenzije se čuvaju u promenljivoj *dimensions* pri čemu se visini može pristupiti sa `img.shape[0]`, a širini sa `img.shape[1]`. Ukoliko želimo da pristupimo broju kanala slike, to je moguće uraditi pozivom `img.shape[2]`.

$$M = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{bmatrix}$$

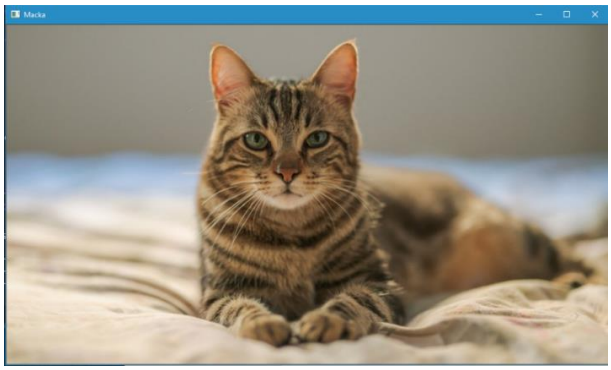
Funkcija translacije funkcioniše na sledeći način:

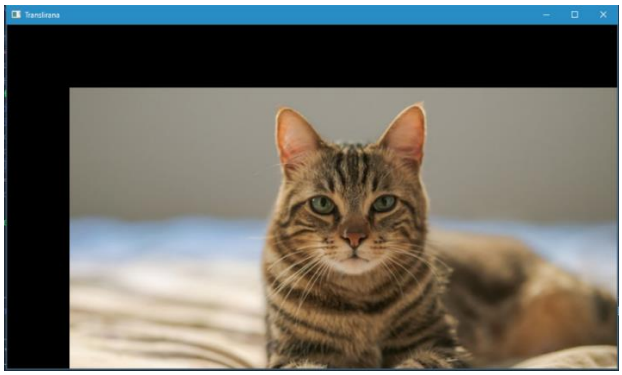
- Ukoliko je vrednost za x negativna, radi se translacija *levo*
- Ukoliko je vrednost za x pozitivna radi se translacija *desno*
- Ukoliko je vrednost za y negativma radi se translate *gore*
- Ukoliko je vrednost za y pozitivna radi se translate *dole*

Pozivom funckije dobija se sledeći prikaz:

```
translated = translate(img, 100, 100)
```

Translacija se izvršila desno po x-osi i dole po y-osi.





Rotacija slike se može izvršiti primenom sledeće funkcije:

```
def rotate(img, angle, rotPoint= None):
    (height,width)=img.shape[:2]
    if rotPoint is None:
        rotPoint= (width//2, height//2)
    rotMat= cv.getRotationMatrix2D(rotPoint, angle, 1.0)
    dimensions= (width, height)
    return cv.warpAffine(img, rotMat, dimensions)
```

Ukoliko se ne navede ugao rotacije, smatra se da će se rotacija vršiti oko centra. Matrica transformacije koja se primenjuje je data na sledećoj slici:

$$M = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix}$$

Međutim, OpenCV pruža mogućnost rotacije oko proizvoljne tačke, pa bi se modifikovana matrica mogla predstaviti na sledeći način:

$$\begin{bmatrix} \alpha & \beta & (1-\alpha)\cdot center.x - \beta \cdot center.y \\ -\beta & \alpha & \beta \cdot center.x + (1-\alpha)\cdot center.y \end{bmatrix}$$

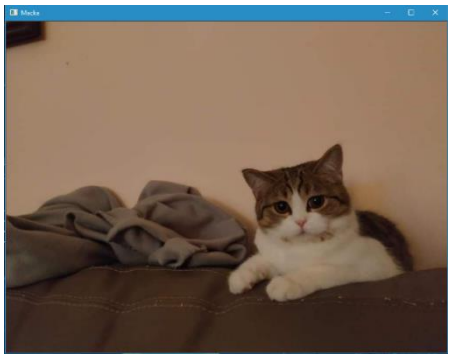
Pri tome važi :

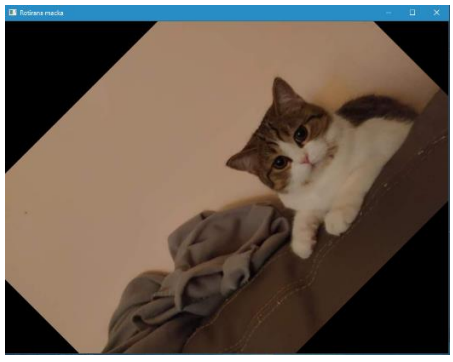
$$\alpha = scale \cdot \cos\theta,$$
$$\beta = scale \cdot \sin\theta$$

Izvršenjem sledeće funkcije dobija se dole navedeni prikaz:

```
rotated = rotate(img, 45)
```

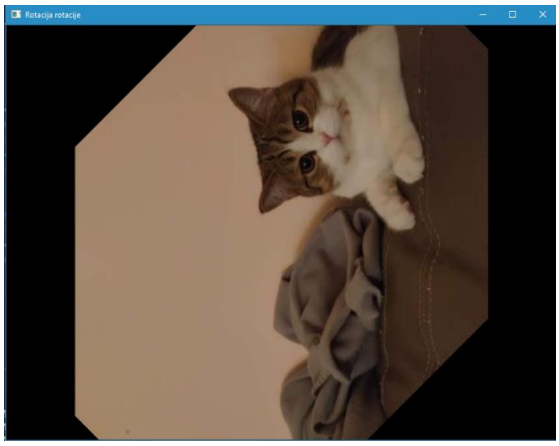
Rotacija se vrši u smeru suprotnom od smera kazaljke na satu, za rotaciju u smeru kazaljke na satu je potrebno staviti da vrednost ugla bude negativna. Budući da centar rotacije nije naveden, rotiranje je vršeno u odnosu na centar date slike.





Ukoliko bismo još jednom pozvali funkciju rotacije, rezultat koji bismo dobili najverovatnije nije onakav kakav smo očekivali. Zato je potrebno specificirati u jednoj funkciji ugao, a ne zvati više puta funkciju rotacije (ukoliko želimo rotaciju za 90 stepeni bolje je proslediti taj ugao nego dva puta zvati funkciju za 45 stepeni).

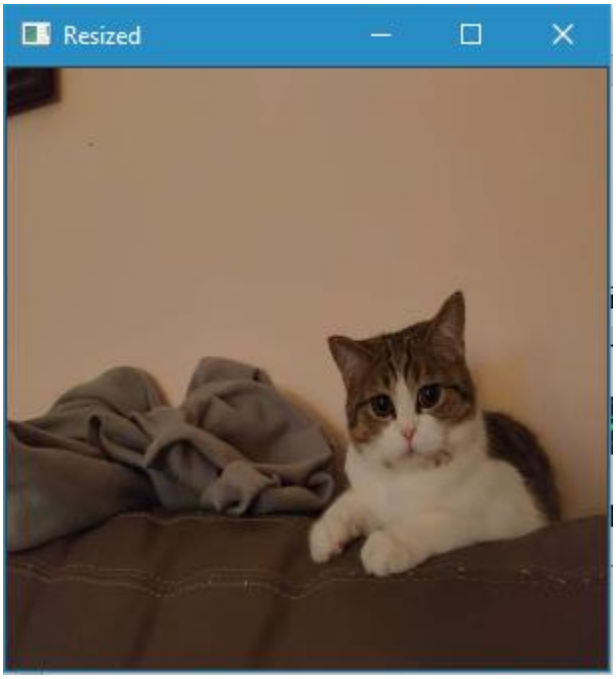
Dat je primer nakon pozivanje rotacije za 45 stepeni dva puta.



Još jedna transformacija koju je moguće izvršiti je promena veličine slike. To se može ostvariti pozivom sledeće funkcije :

```
resized = cv.resize(img, (300,300), interpolation= cv.INTER_CUBIC)
```

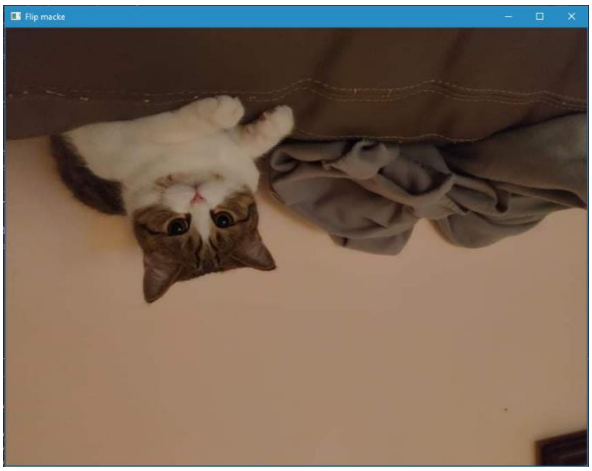
Kao vrednosti interpolacije je moguće proslediti i INTER_AREA (za smanjivanje) i INTER_LINEAR (za zumiranje).



Pored toga, postoje i funkcije za obrtanje slike oko horizontalne i vertikalne ose. To se može uraditi na sledeći način:

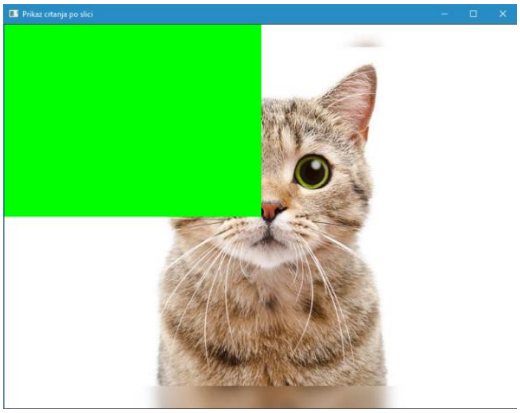
```
flip=cv.flip(img, -1)
```

Ukoliko se prosledi vrednost 0 to znači da će se obrtanje izvršiti oko x-ose, ukoliko se prosledi 1 izvršiće se oko y-ose, dok vrednost -1 signalizira i promenu oko x i y ose.



Ukoliko specifičnost primene to zahteva, po slikama je moguće i crtati odgovarajuće geometrijske primitive, ali i dodavati tekstualni materijal. Crtanje pravougaonika se može postići na sledeći način:

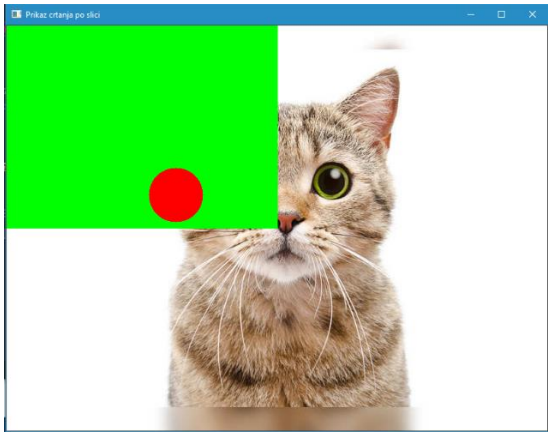
```
cv.rectangle(img, (0, 0),
             (img.shape[1]//2, img.shape[0]//2), (0, 255, 0), thickness=cv.FILLED)
```



Crtanje kruga je moguće ostvariti sledećim kodom:

```
cv.circle(img, (250,250), 40, (0,0,255), thickness=-1)
```

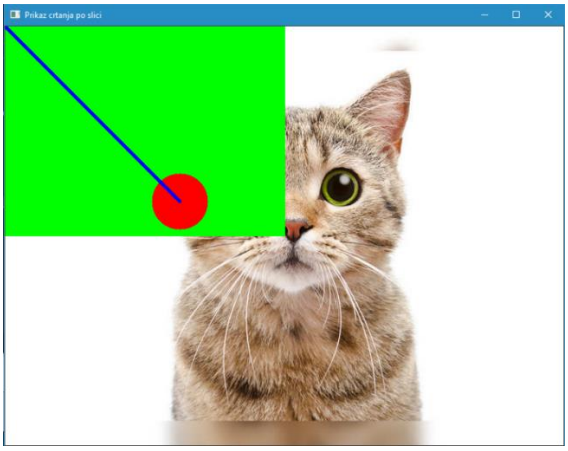
Dati parametri predstavljaju sliku nad kojom će se crtati krug, centar kruga, poluprečnik, boju i debljinu linije. Treba voditi računa o činjenici da je format prikaza boja u OpenCV-u BGR (*blue-green-red*). Ukoliko se za thickness stavi vrednost -1 znači da će se vršiti ispuna geometrijskog objekta.



Crtanje linije se može ostvariti na sledeći način:

```
cv.line(img, (0,0) ,(250,250), (255,0,0), thickness=3)
```

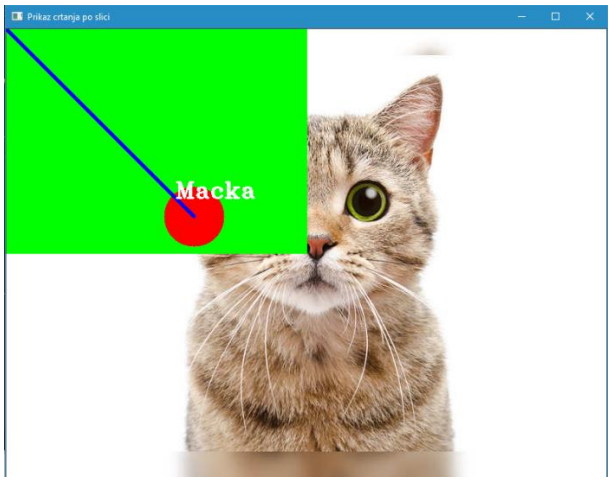
Kao parametri su korišćeni slika nad kojom se vrši iscrtavanje, početak linije, tačka završetka linije, boja, kao i debljina.



Ukoliko je na slici potrebno prikazati tekst, poziv ove funkcije će to omogućiti:

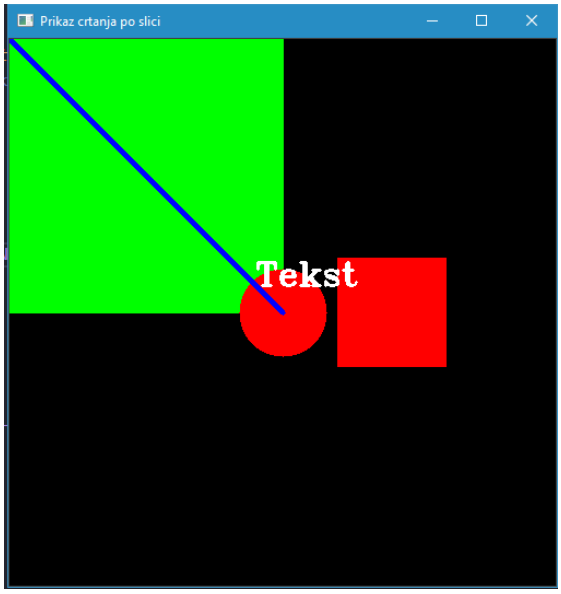
```
cv.putText(img, 'Macka', (225,225), cv.FONT_HERSHEY_TRIPLEX, 1.0, (255,255,255), 2)
```

Kao parametri se navode slika nad kojom se vrši ispis teksta, vrednost koja će biti ispisana, tačka početka ispisa teskta, font koji se koristi (*font family*), veličina fonta, boja kao i *stroke*.



Crtanje svih ovih primitiva je bilo moguće izvršiti i nad blanko slikom. To se može uraditi na sledeći način:

```
blank = np.zeros((500, 500, 3), dtype='uint8')
```



6. Opis implementacije projekta

Kod koji je implementiran se nalazi u Python skiptama, pod nazivima detekcija_macaka.py , detekcija_macaka_video.py I detekcija_macaka_web_cam.py. Unosom odgovarajuće komande u terminal editora (korišćen je VS Code) će se prikazati rezultujuća slika ili video snimak. Izlazak iz video snimka se vrsi klikom na taster '"***q***".

```
> python detekcija_macaka.py
```

Za praktičnu implementaciju ovog algoritma je najpre potrebno učitati željenu sliku i konvertovati je u grayscale režim. To se može uraditi na sledeći način:

```
img = cv.imread('photos/liza.jpg')
#cv.imshow('Macka', img)

gray= cv.cvtColor(img, cv.COLOR_BGR2GRAY)
#cv.imshow('Siva macka', gray)
```

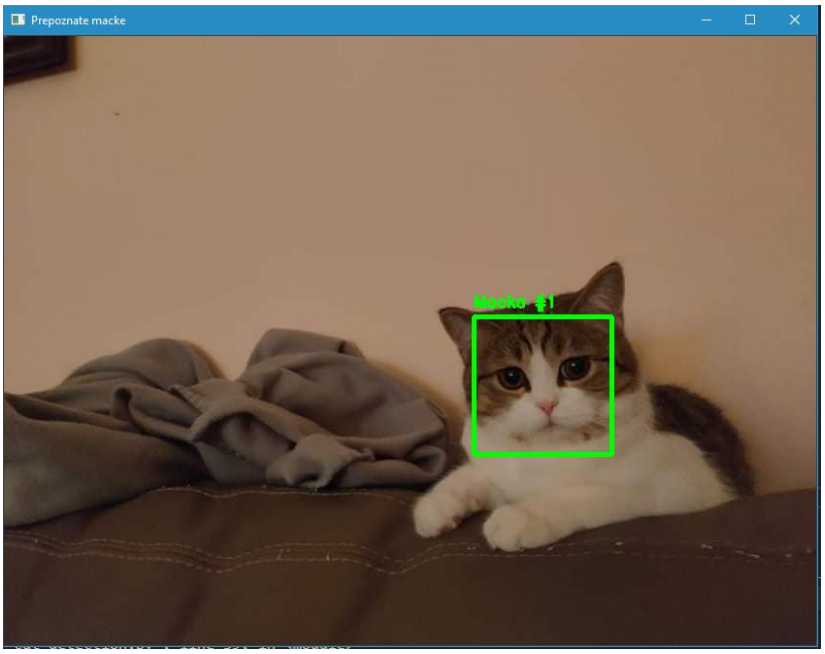
Nakon toga se izvršava funkcija kaskadnog klasifikatora :

```
haar_kaskada = cv.CascadeClassifier("cat_face_detector.xml")
```

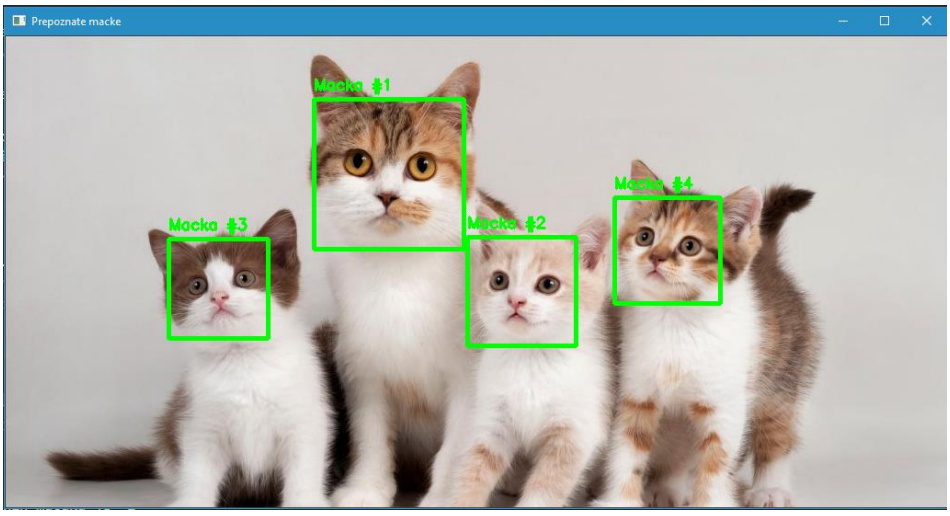
Promenom parametara funkcije ***detectMultiScale*** možemo da dobijemo različite rezultate za istu prosleđenu sliku. Parametri koji su ovde uzeti u obzir su ***scaleFactor*** i ***minNeighbors***. Prvi parametar se koristi da se kreira scale piramida. Tokom treniranja model ima fiksnu veličinu. To znači da se ta veličina lica detektuje na slici ako postoji. Ali reskaliranjem ulazne slike može se vršiti promena veličine većeg lica ka manjem čime ce postati moguće da bude detektovano od strane algoritma. Uzimanjem manje vrednosi za faktor skaliranja (npr 1.05) znači da smanjujemo velicinu slike za 5% i povećavamo šansu pronalaska lica. Drugi parametar specificira koliko suseda svaki kandidat pravougaonik treba da ima da bi bio smatran pravougaonikom lica. Povećavanjem broja suseda moguće je eliminisati lažno pozitivne vrednosti, ali se isto tako mogu izgubiti i stvarne pozitivne vrednosti.

Korišćenjem sledećih vrednosti, za prosleđenu sliku dobijamo ovakav rezultat:

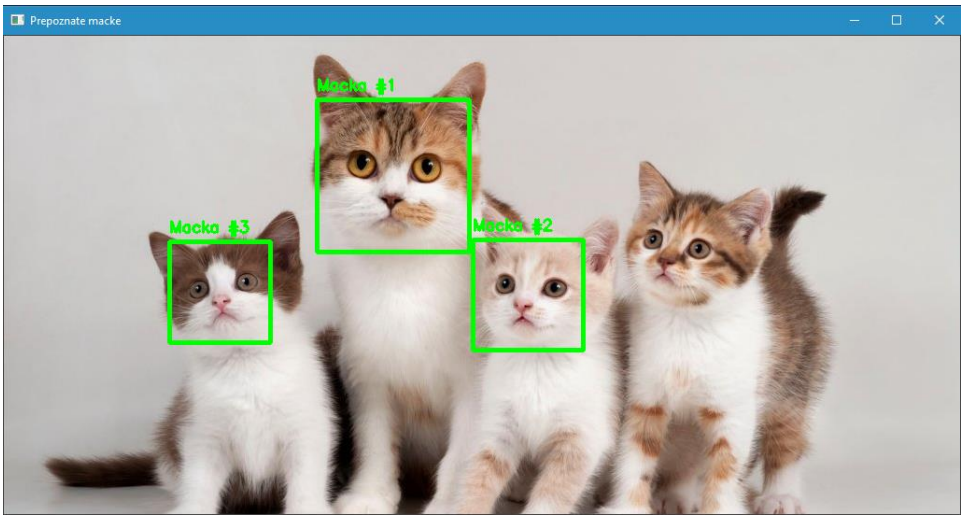
```
macka_pravougaonik = haar_kaskada.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=3)
```



Sa istim vrednostima faktora skaliranja i broja suseda je moguće detektovati i mačke na sledećoj slici:



Međutim, ako bismo broj suseda povećali na 5, jedna od mačaka ne bi bila detektovana:



Ukoliko bismo povećali i faktor skaliranja na 1.4 dobili bismo prikaz gde nijedna od mačaka nije detektovana:

Na osnovu koda za detekciju slika, moguće je uraditi i detekciju mačaka na video snimcima, budući da video snimak možemo posmatrati kao niz frejmova. Učitavanje video snimka se vrši na sledeći način:

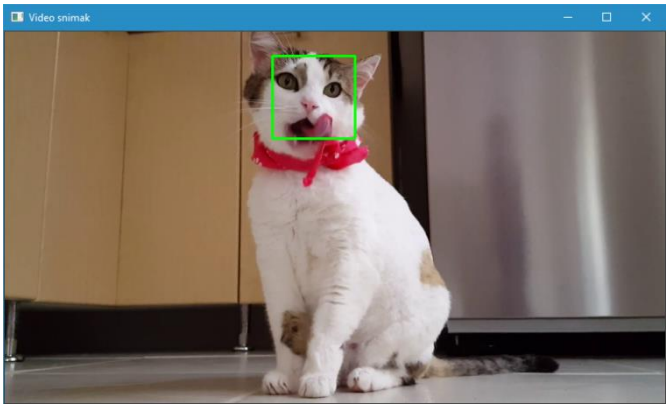
```
cap = cv.VideoCapture('videos/macka4.mp4')
```

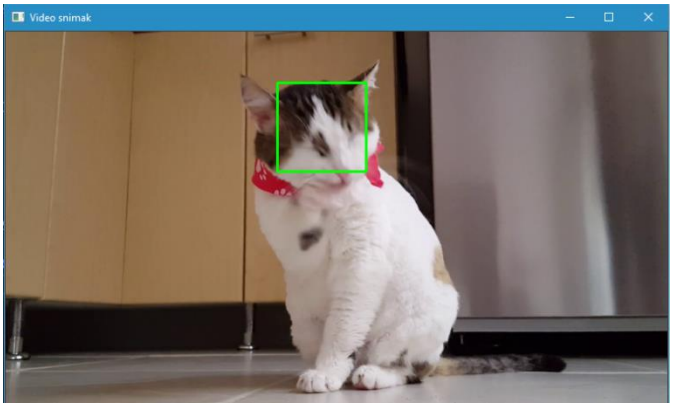
Primećeno je da bolje rezultate postižemo sa video snimcima manjih dimenzija tako da je primenjena sledeća modifikacija, odnosno reskaliranje:

```
def rescaleFrame(frame, scale =0.40):
    width = int(frame.shape[1]*scale)
    height= int(frame.shape[0]*scale)
    dimensions = (width,height)
    return cv.resize(frame,dimensions, interpolation=cv.INTER_AREA)
```

Izlazak iz video snimka se vrši klikom na taster ''***q***''.

Prikaz nakon pokretanja *python* skripte je sledeći:





Budući da je data set na kojem je treniran klasifikator namenjen detekciji frontalno orijentisanih objekata, u trenucima kada mačka pomeri glavu u stranu dešava se da dođe do prekida detekcije. Kvalitet detekcije će i u ovom slučaju zavisiti od podešavanja parametara funkcije ***detectMultiScale*** pa su vrednosti koje su izabrane za ovaj konkretan snimak date isečkom:

```
cat_faces = haar_kaskada.detectMultiScale(gray, 1.1,2)
```

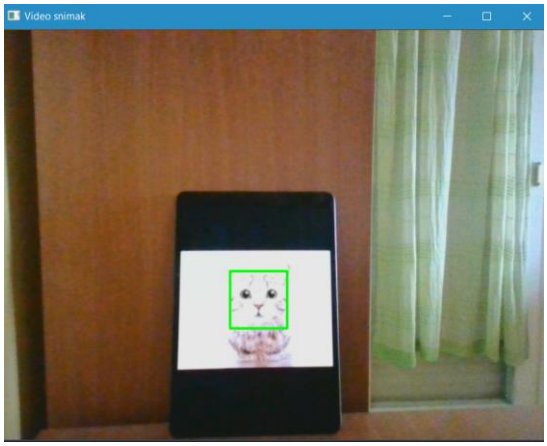
Implementirana je još jedna varijanta detekcije koja se odnosi na prikaz preko ***web kamere***. Izmene u odnosu na kod koji je naveden za video snimak su minimalne i ogledaju se u tome da nije potrebno vršiti promenu dimenzija video snimka i umesto učitavanja sa fajl sistema vrši se detekcija na osnovu prikaza web kamere računara. Izlazak iz video snimka se vrši klikom na taster "q". Sledeća linija koda će omogućiti prikaz preko web kamere:

```
cap = cv.VideoCapture(0)
```

Parametri koji su korišćeni pri detectMultiScale funkciji su sledeći:

```
cat_faces = haar_kaskada.detectMultiScale(gray, 1.06,3)
```

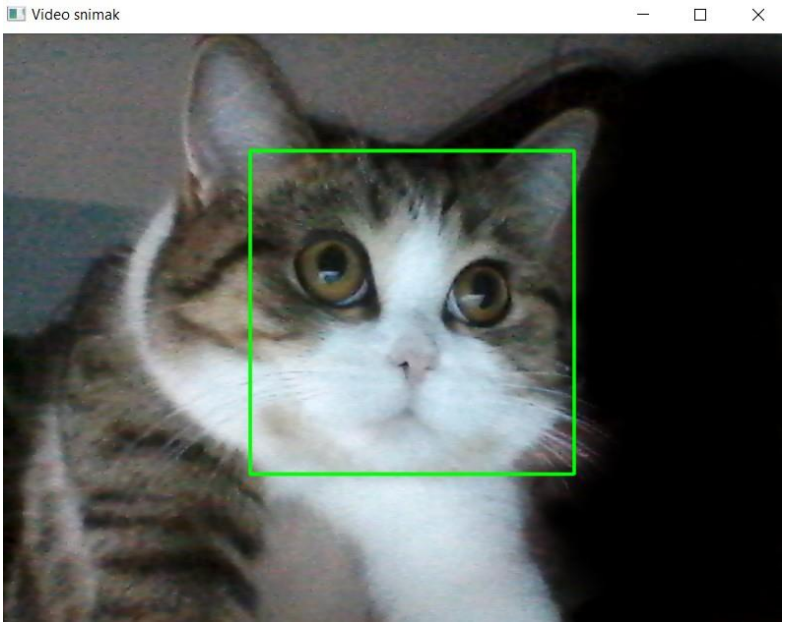
Testiranje je vršeno i na osnovu slike mačke sa tablet računara, ali i na pravoj mački. Rezultati su dati na sledećim slikama:

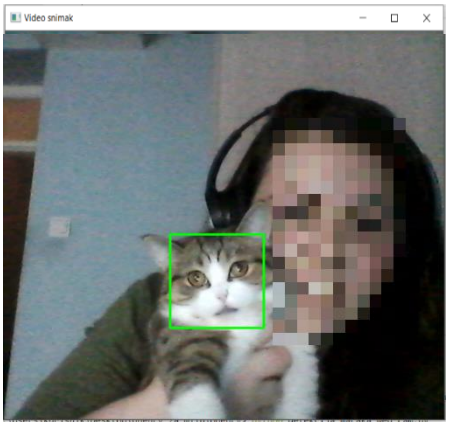


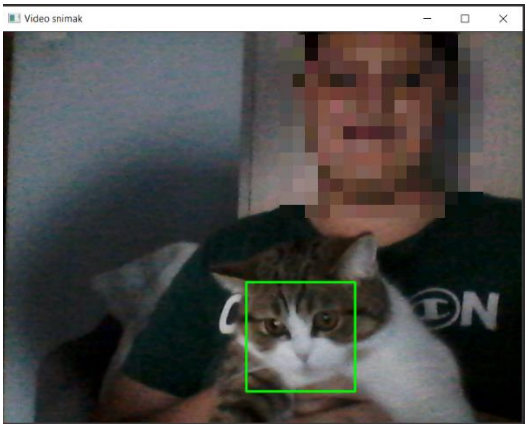
Prikaz testiranja sa pravom mačkom je dat na sledećim slikama pri čemu je izvršena sledeća promena parametara u detectMultiScale funkciji:

```
cat_faces = haar_kaskada.detectMultiScale(gray, 1.08,5)
```

Promena parametara može da dovede do situacije da i ljudi budu prepoznati.







7. Još neke primene Računarskog vida

Računarska vizija je oblast koja datira iz šezdesetih godina prošlog veka, ali je tek u skorije doba došlo do napredaka koji su doveli do mnogobrojnih primena ove oblasti. Jedna od značajnih primena se može posmatrati u domenu ***zdravstva***. Analiza medicinskih slika je najbolji primer, budući da može biti od velike pomoći prilikom ustanove dijagnoze. Pod time se može misliti na slike dobijene raznim vrstama skenera koje se analiziraju u cilju traženja anomalija poput tumora ili raznih vrsta neuroloških oštećenja. U nekim slučajevima je dovoljno primeniti tehnike analize slike koje izdvajaju određene delove koji se zatim zadaju klasifikatoru kao ulazni parametar i on ima za cilj upravo detekciju anomalija. Ipak, u nekim situacijama je neophodno vršiti detaljnije procesiranje. Ako za primer uzmemo analizu slika kolonoskopije, neophodno je izvršiti segmentaciju slika kako bi adekvatno bili traženi polipi i sprečeni kolorektalni tumori. Osim analize slika, tehnike računarskog vida u domenu medicine se mogu ogledati i u praćenju zenica i analizi očne oblasti u cilju rane detekcije kognitivnih patologija kao što su autizam i disleksija.

Računarski vid je prisutan i u ***automobilskoj industriji***, pogotovo u domenu samovozećih automobila. Detekcija objekata igra ključnu ulogu budući da je neophodno locirati varijabilne količine objekata kao što su drugi automobili, semafori, saobraćajna signalizacija i pešaci.

Domen u kojem su tehinke računarskog vida od pomoći je i ***agrikultura***. Produkcija zrnastog povrća je često otežana raznim vrstama insektne infekcije i bolesti. Rana dijagnostika omogućava farmerima da preduzmu mere na vreme i na taj način dolazi do redukcije gubitaka i povećanja kvaliteta proizvoda. Budući da je korov postao dosta rezistentiji na razne vrste herbicida, počeli su da se koriste roboti sa implementiranim algoritmima računarske vizije koji omogućavaju preciznije prskanje herbicidima. Osim toga, analiza kvaliteta zemljišta je od ključnog značaja za kvalitetan odgoj voća i povrća. U primeni su alogoritmi koji na osnovu slike mogu da prepoznaju odredjenje nutritivne defekte zemljišta i na taj način može doći do prevencije i obogaćivanja tla odgovarajućim materijama. Još jedna primena se može ogledati u sortiranju voća, povrća i cveća, što funkcioniše po principu identifikacije njihovih glavnih osobina (veličine, kvaliteta, težine, boje i teksture). Ovi algoritmi su sposobni da prepoznaju defekte određenih proizvoda i da procene koje jedinke će imati bolju šansu da duže opstanu na policama lokalnih marketa. Na ovaj način se vrši i optimizacija skladištenja proizvoda u marketima. Primena računarke vizije u marketima se može ogledati i u vrlo preciznoj analizi trenutno dostupnih proizvoda kako bi na taj način menadžeri zaduženi za nabavku bili bolje informisani. Pored toga, moguće je vršiti i analizu mesta na rafovima kako bi bile identifikovane najoptimalnije konfiguracije proizvoda i time povećana prodaja. Veliki gubitak u profitu se može desiti i prilikom nekvalitetnih proizvodnih linija tj. manufakture. Nefunkcionalnost se može ogledati u prestanku rada pojednih mašina ili u defektnoj proizvodnji. Postavljanjem robota opremljenih odgovarajućim algoritmima ovaj problem biva rešen pomoću analize vizuelnih informacija svakog pojedinačnog proizvoda. Na ovaj način, moguće je u realnom vremenu obavestiti radnike koji će potom dati proizvod ukloniti sa proizvodne linije i sprečiti njegovu dalju propagaciju.

8. Reference

1. Predavanja iz predmeta Računarski vid na Elektronskom fakultetu u Nišu
2. OpenCV Course - Full Tutorial with Python
3. Udacity kurs: Introduction to Computer Vision, Aaron Bobick
4. Computer Vision: Algorithms and Applications, 2nd ed.
5. Programming 2.0 Webinar: Autonomous Driving, prof dr Aleksandar Milosavljević
6. Uvod u veštačku inteligenciju i mašinsko učenje (2.deo), dr Zoran Stanković, dr Zlatica Marinković, dr Vera Marković, Benefit
7. https://tryolabs.com/resources/introductory-guide-computer-vision/
8. https://machinelearningmastery.com/what-is-computer-vision/#:~:text=Computer%20Vision%2C%20often%20abbreviated%20as,people%2C%20even%20very%20young%20children.
9. Modelovanje nelinearnih reaktivnih elektronskih kola primenom vestackih neuronskih mreza
10. Mladen Nikolić i Anđelka Zečević: Mašinsko učenje. Matematički fakultet, Univerzitet u Beogradu, 2018.
11. Veštačka inteligencija, Predrag Janičić i Mladen Nikolić, 2020
12. https://www.geeksforgeeks.org/opencv-overview/
13. https://docs.opencv.org/master/d4/d13/tutorial_py_filtering.html
14. https://theailearner.com/2019/05/11/understanding-image-gradients/
15. https://www.purplemath.com/modules/meanmode.htm
16. https://docs.opencv.org/3.4/d7/de1/tutorial_js_canny.html
17. https://www.geeksforgeeks.org/erosion-dilation-images-using-opencv-python/
18. https://docs.opencv.org/3.4/db/df6/tutorial_erosion_dilatation.html
19. https://docs.opencv.org/3.4/d4/d73/tutorial_py_contours_begin.html#:~:text=Contours%20can%20be%20explained%20simply,and%20object%20detection%20and%20recognition.&text=In%20OpenCV%2C%20finding%20contours%20is,white%20object%20from%20black%20background.
20. https://www.geeksforgeeks.org/python-thresholding-techniques-using-opencv-set-1-simple-thresholding/#:~:text=Thresholding%20is%20a%20technique%20in,maximum%20value%20(generally%20255)
21. https://www.pyimagesearch.com/2018/07/19/opencv-tutorial-a-guide-to-learn-opencv/
22. https://docs.opencv.org/master/da/d6e/tutorial_py_geometric_transformations.html
23. https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_geometric_transformations/py_geometric_transformations.html#:~:text=warpAffine()%20function%20is%20the,and%20height%20%3D%20number%20of%20rows.
24. https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html
25. https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_gui/py_video_display/py_video_display.html
26. An efficient and cost effective FPGA based implementation of the Viola-Jones face detection algorithm.