Veliko hvala mentoru prof. dr. sc. Domagoju Jakoboviću na ukazanom strpljenju, povjerenju, vodstvu i nesebičnoj pomoći tijekom studiranja i pisanja rada.

Hvala i obitelji na podršci kroz sve godine studija kako bi učinili ovo razdoblje života sretnim i što više bezbrižnim. Veliko hvala mojim roditeljima što su vjerovali u mene.

Sadržaj

1. Uvod ... 1
2. Opis detekcije zauzeća parkirnih mjesta ... 2
3. Strojno učenje ... 4
   3.1. Duboko učenje ... 5
      3.1.1. Umjetne neuronske mreže ... 6
      3.1.2. Učenje neuronskih mreža ... 8
4. Skupovi podataka ... 9
   4.1. COCO ... 9
   4.2. Kitti Vision ... 10
   4.3. BDD100K ... 11
   4.4. Stanford Cars ... 12
   4.5. UAVDT ... 13
   4.6. Urban Tracker ... 14
   4.7. VOC 2012 ... 16
5. Ostvarenje detekcije zauzeća parkirnih mjesta ... 17
   5.1. Automatska detekcija parkirnih mjesta ... 19
      5.1.1. Algoritam k-srednjih vrijednosti ... 22
      5.1.2. Algoritam mješavine Gaussovih gustoća ... 25
   5.2. Detekcija vozila na parkirnih mjestima ... 29
6. Usporedba modela i ostvareni rezultati ... 35
7. Zaključak ... 69
8. Literatura ... 70

1. Uvod

Detekcija objekata (engl. *object detection*) na slici je zanimljiv problem koji ima mnoge primjene u industriji, te je i prvi korak u rješavanju težih problema. Usko je povezana s računalnim vidom (engl. *computer vision*) i procesiranjem slika (engl. *image processing*). Te dvije grane bave se detekcijom objekata određene klase (ljudi, automobili, životinje) sa slika i videozapisa.

Detekcija objekata je zahtjevan problem zbog specifičnosti objekata koje želimo detektirati i raspoznati. Ako želimo detektirati čovjeka sa slike, znamo da svi ljudi ne izgledaju identično, ali imaju slične karakteristike koje ih opisuju (2 ruke, 2 noge, tijelo, glava itd.). Takve karakteristike pomažu kod učenja neuronskih mreža i njihovoj upotrebi.

Ovaj rad sastoji se od 5 dijelova. U prvom poglavlju opisan je i objašnjen problem detekcije zauzeća parkirnih mjesta kao kompletan sustav. U drugom poglavlju opisane su metode dubokog, kao i strojnog učenja pomoću kojih se ostvaruje detekcija objekata. Uz to objašnjeno je i treniranje neuronskih mreža koje su glavni alat detekcije objekata. U trećem poglavlju navedene su i opisane baze podataka (slika) koje su se koristile pri treniranju dubokih neuronskih mreža. U četvrtom poglavlju opisane su metode kako je ostvaren sustav detekcije zauzeća parkirnih mjesta, koje tehnologije i hardware je korišten. Također su navedeni i problemi oko ostvarivanja sustava i njihova rješenja. U petom poglavlju su prikazani, objašnjeni i uspoređeni rezultati gotovih modela i dodatno naučenih modela iz ovog rada.

2. Opis detekcije zauzeća parkirnih mjesta

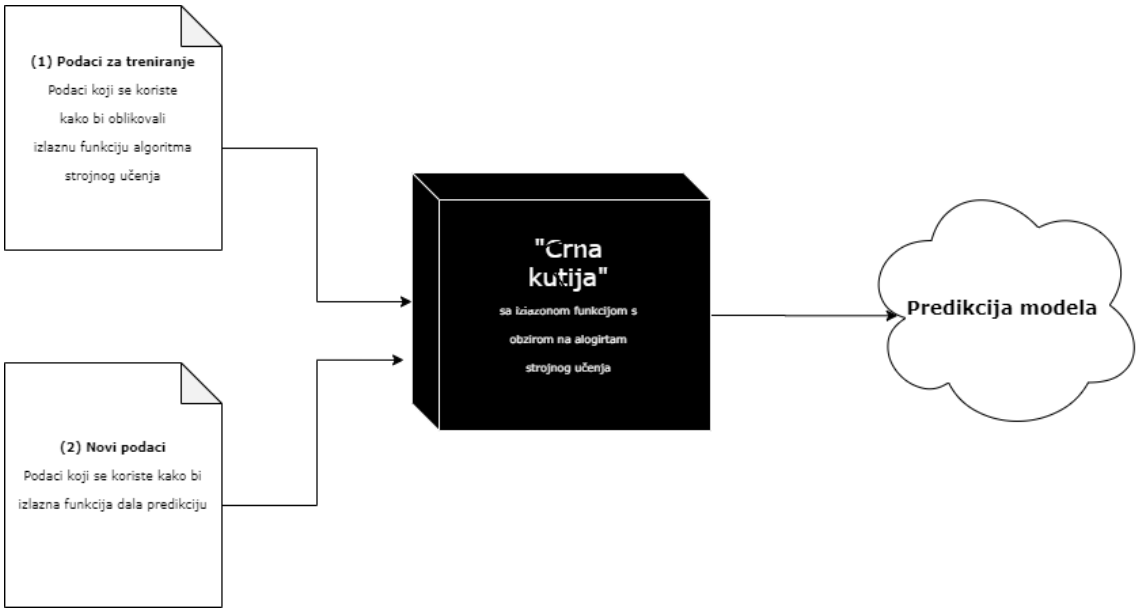
Cilj sustava detekcije zauzeća parkirnih mjesta je da se kroz ulaz u sustav, koji, je slika s video zapisa ili video stream, dobije izlaz čija je reprezentacija broj slobodnih parkirnih mjesta na tom video zapisu u datom trenutku. Kao prvi korak potrebno je postavljanje (engl. *set-up*), odnosno crtanje poligona na slici u kojima će se prepoznavati zauzeće parkirnog mjesta i upisivanje imena svakog poligona, broj mjesta koji on zauzima i *access token* tog poligona. Moguće je preskočiti prethodni korak, ako se ostavi da sustav radi određeno vrijeme. Tada preko tehnika strojnog učenja i statistike sustav sam prepoznaje gdje su parkirna mjesta i nije ih potrebno ucrtavati. Nakon određenog *set-up*-a odrađuje se detekcija, te se broj detektiranih vozila oduzima od broja slobodnih mjesta i taj broj se pomoću MQTT protokola šalje platformi koji to finalno prikazuje na LED pokazivaču. Uz LED pokazivač kao izlaz sustava može biti i slika s ulaza na kojoj se vidi određena detekcija.

Slika 2.1. Primjer izlaza sustava sa ručno nacrtanim mjestima

Slika 2.2. Primjer izlaza sustava bez ručno nacrtanih mjesta

3. Strojno učenje

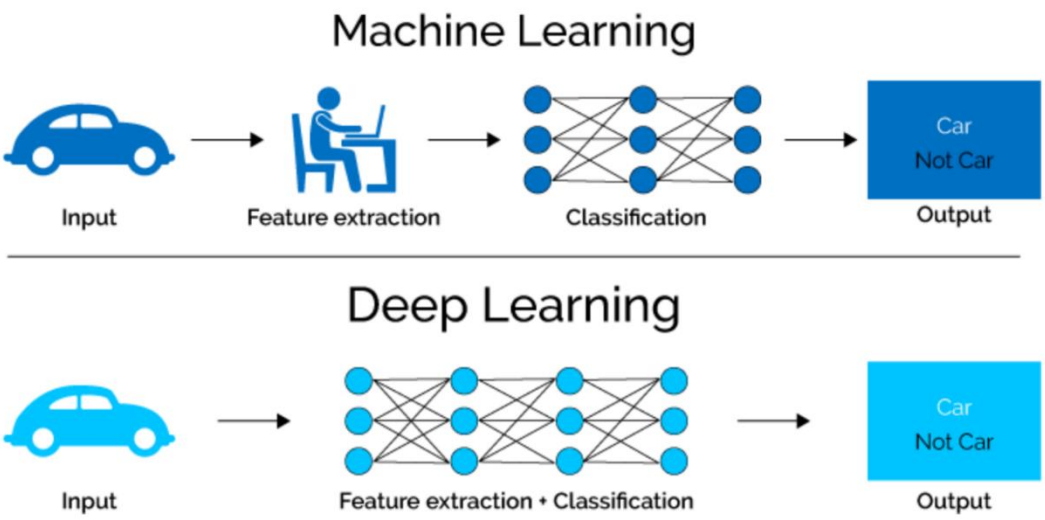
Strojno učenje je grana umjetne inteligencije koja se bavi oblikovanjem algoritama koji svoju učinkovitost poboljšavaju na temelju empirijskih podataka [1]. Algoritmi strojnog učenja grade matematičke modele, utemeljene na datim podacima, poznati kao podaci za treniranje (engl. *training data*). Modeli daju predikcije i odluke bez eksplicitnog programiranja da to naprave.



Slika 3.1. Shema modela strojnog učenja

3.1. Duboko učenje

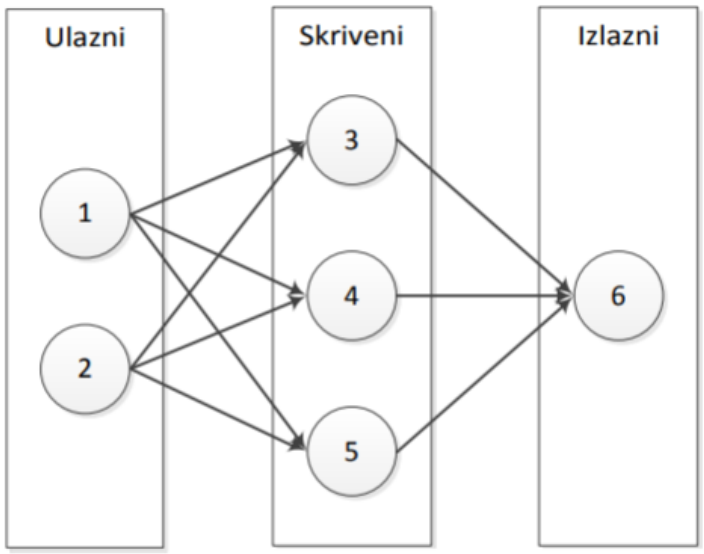
Duboko učenje (engl. *deep learning*) je grana strojnog učenja koja je posebno prikladna za rješavanja problema iz područja umjetne inteligencije. Ono se temelji na predstavljanju podataka složenim reprezentacijama koje su nastale naučenih nelinearnih transformacija [2]. Arhitekture modela dubokog učenja mogu biti duboke neuronske mreže, povratne neuronske mreže, konvolucijske neuronske mreže, kombinacija navedenih i ostale. Duboko učenje primjenjuje se na razna područja, kao što su računalni vid, prepoznavanje govora, obrada prirodnog jezika, bioinformatika itd.



Slika 3.1.1. Razlika strojnog i dubokog učenja [3]

3.1.1. Umjetne neuronske mreže

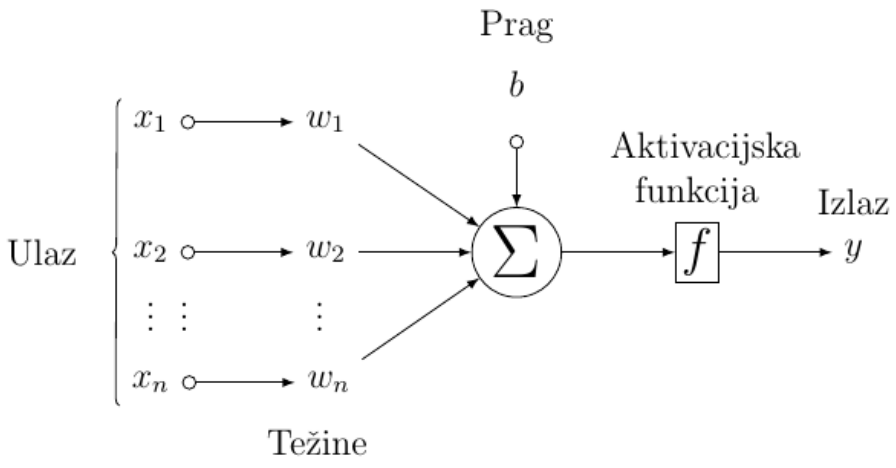
Glavni alat dubokog učenja su umjetne neuronske mreže ili skraćeno neuronske mreže (engl. *artificial neural networks*). Neuronske mreže su matematički modeli koji se koriste za rješavanje vrlo složenih problema navedenih prije. One su paradigma koja opisuje obradu informacija na sličan način na koji ljudski mozak i neuroni obrađuju informacije. Neuronska mreža, kao i mozak, sastoji se od neurona međusobno povezanih živčanim vezama odnosno sinapsama. Kada se neki neuron „pobudi“, on stvara električni impuls koji putuje sinapsom do drugih neurona i tako se stvara akcija.



Slika 3.1.1. Unaprijedna povezana jednostavna neuronska mreža

Sa slike se vidi da je arhitektura neuronske mreže slojevita. Neuroni se nalaze u slojevima. Postoje ulazni sloj, koji služi kao ulaz podataka u neuronsku mrežu, skriveni slojevi i izlazni sloj. Izlazni sloj nam daje izlaz neuronske mreže, koji može biti predikcija ili klasifikacija. Ako mreža ima veći broj skrivenih slojeva onda je ta mreža duboka neuronska mreža.

Svaka veza između neurona ima određenu težinu, koja je potrebna za izračun izlaza mreže. Mreža računa izlaz u i-tom sloju tako da se prvo izračuna težinska suma (skalarni produkt vektora ulaznih podataka u neuron i vektora težina), nakon toga se toj sumi dodaje slobodni član neurona (engl. *bias*). Kada je suma izračunata ona prolazi kroz aktivacijsku funkciju neurona koja radi nelinearno preslikavanje. Postupak se provodi od prvog sloja prema zadnjem.

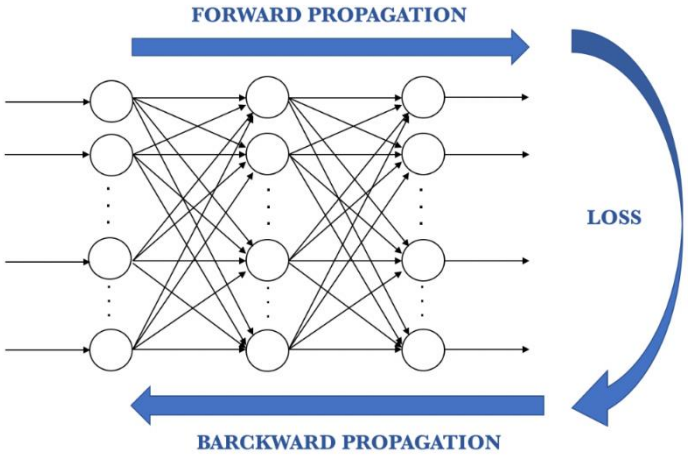


Slika 3.1.2. Model neurona neuronske mreže

3.1.2. Učenje neuronskih mreža

Temeljno obilježje neuronskih mreža je svojstvo učenja. Opet po uzoru na mozak, učenje neuronskih mreža zapravo znači jačanje sinapsa između određenih neurona. „Bitnije“ veze između neurona imaju veću težinu, dok manje „bitne“ imaju manju ili nemaju težinu. Proces treniranja neuronske mreže je iterativan postupak optimiranja težina između težina.

Najpoznatiji i najčešći algoritam za razvoj neuronskih mreža je *backpropagation.* Optimiranje težina mreže provodi se računanjem greške izlaza mreže za ulazne podatke iz skupa za učenje, te propagiranjem greške povratno kroz veze između neurona, po čemu je algoritam i dobio naziv. Uz skup za treniranje postoji i skup za validaciju kojim se postiže generalizacija mreže tj. postiže se da se mreža ne „pretrenira“ na skupu za učenje. Bitan je i skup za testiranje koji se koristi za ispitivanje točnosti mreže kada je trening gotov.



Slika 3.1.2.1. Vizualizacija *backpropagation*-a jednostavne neuronske mreže

4. Skupovi podataka

Kao što je navedeno u prijašnjem poglavlju, kako bi se neuronska mreža učila potreban je skup podataka. U ovom radu za treniranje korišteno je 7 različitih baza podataka za treniranje novih neuronskih mreža. Svaka baza podataka prvo je podijeljena na skup treniranja i testiranja tako da nasumičnih 20% slika čini skup za testiranje, dok preostalih 80% čini skup za treniranje.

4.1. COCO

COCO (*Common Object in Context*) je baza podataka koji služi za detekciju objekata, segmentaciju velikih razmjera [11]. Neki od bitnijih značajki COCO-a su:

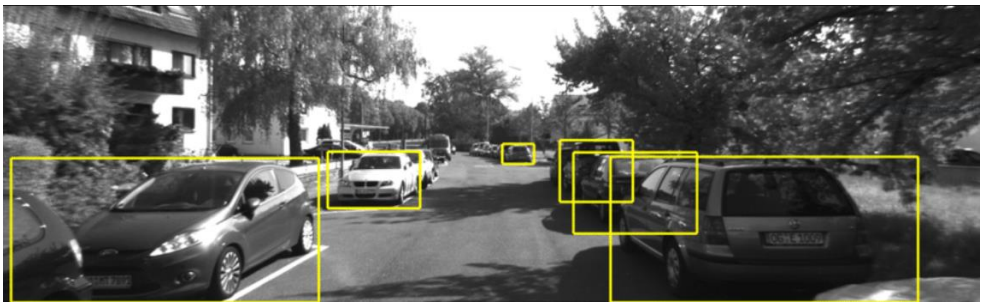
- Segmentacija objekata
- 330 000 slika (preko 200 000 označenih)
- 1 500 000 instanca objekata
- 80 klasa objekata

Anotacije su mu zapisane u JSON formatu iz kojih se mogu izvući nama najbitnije stvari, što su početne točke i dimenzije *bounding box*-a, dimenzije same slike i ID klase (u ovom radu korištena je samo klasa automobil).

Slika 4.1.1. Primjer slike iz COCO skupa podataka

4.2. Kitti Vision

Kitti Vision skup podataka razvijen je na Karlsruhe Institute of Technology and Toyota Technological Insititute u Chicagu. Slike su sakupljene u gradu Karlsruhe preko kamere na automobilu. Na slikama je vidljivo do 15 automobila i 30 pješaka. Baza podataka sastoji se od oko 15 000 slika i 80 000 označenih objekata. Anotacije su zapisane u MATLAB datoteci iz koje se može izvući podaci o *bounding box*-u traženog objekta.



Slika 4.2.1. Primjer slika iz skupa Kitti Vision

4.3. BDD100K

BDD100K (Barkley DeepDive) je baza podataka koja se koristi za treniranje mreža za autonomnu vožnju. Njegova najveća prednost nad drugim skupovima je da se sastoji od 100 000 sekvenca video zapisa, točnije 120 000 000 slika. Također slike su snimljene u više gradovima u različitim vremenskim uvjetima, u različita doba dana [17]. Problem kod ovog skupa kao i prijašnjih je da su sve slike snimane u razini očiju. Anotacije svih slika za treniranje su zapisane u 1 JSON datoteci, isto tako i za slike za testiranje.

Slika 4.3.1. Primjer neoznačene slike iz BDD100k skupa podataka

4.4. Stanford Cars

Stanford Cars skup podataka sastoji se od 16185 slika automobila razvrstanih u 196 klasa. Skup za treniranje ima 8144 slika, dok skup za testiranje ima 8041 sliku. Specifično kod ovog skupa podataka je da služi za detekciju osobina automobila (*proizvođač, model, godina*), npr. 2012 Tesla Model S [13]. Problem kod ove baze podataka je da slike u skupu za testiranje nisu označene, stoga je za potrebe ovog rada korišten samo skup za treniranje. Dodatno jedan od problema ovog skupa podataka je da su slike previše idealne za ovaj problem (na svakoj slici se cijeli automobil bez ikakvih smetnji u dobroj kvaliteti), ne podsjećaju na stvaran scenarij na parkiralištima. Anotacije su zapisane u MATLAB datoteci, te je svaka klasa pretvorena u klasu automobil, dok su dimenzije *bounding box*-a zadržane.

Slika 4.4.1. Primjeri neoznačenih slika Stanford Cars

4.5. UAVDT

Skup podataka UAVDT razvijen je u radu Dawei Du [15]. Du je htio ostvariti prepoznavanja vozila iz visoke ptičje perspektive, ali sve baze podataka koje je pronalazio bile su ostvarene iz perspektive očiju, radi autonomne vožnje. Odlučio je dohvatiti slike s UAV (*Unmanned Aerial Vehicle*), te ih sam ručno označiti i na njima natrenirati svoj model. Baza podataka sastoji se od 10 sati video zapisa, otprilike 80 000 slika s 14 različitih atributa (npr. vremenski uvjeti, visina UAV-a, kategorija vozila itd.).

Mana ovog skupa podataka, za potrebe ovog rada, je da su svi objekti jako maleni jer je snimano s velikih visina. Anotacije su zapisane u JSON datoteci. Jedna JSON datoteka odgovara jednom video zapisu, dok je u JSON-u svaki red, jedan objekt u određenom okviru video zapisa.

Slika 4.5.1. Primjer slika iz skupa podataka UAVDT

4.6. Urban Tracker

Skup podataka Urban Tracker sastoji se od 5 video zapisa [12]. Za potrebe ovog rada korištena su 3 od 5 video zapisa, te je svaki ručno označen.

1. Sherbrooke
    a. Rezolucija 800x600
    b. 1001 *frame*
    c. 15 vozila i 5 pješaka
    d. Snimano sa nadzorne kamere

Slika 4.5.1. Primjer slike Sherbrooke

2. Rouen
    a. Rezolucija 1024x576
    b. 600 *frame*-ova
    c. 5 vozila i 11 pješaka
    d. Snimano sa kamere sa 3. kata zgrade

Slika 4.5.2. Primjer slike Rouen

3. St-Marc
    a. Rezolucija 1280x720
    b. 1000 *frame*-ova
    c. 9 vozila i 19 pješaka
    d. Snimano sa nadzorne kamere na prometnom križanju

Slika 4.5.3. Primjer slike St-Marc

4.7. VOC 2012

VOC baza podatak osmišljena je u svrhu godišnjeg natjecanja. Cilj natjecanja je ostvaritit što bolji model u 3 svrhe (klasifikacija, detekcija, segmentacija). Skup podataka sastoji se od 20 klasa podijeljen u 4 skupine [16].

- *Ljudi* : ljudi
- *Životinje* : pas, mačka, krava, konj, koza
- *Vozila* : avion, bicikl, brod, automobil, motocikl, vlak
- *Unutrašnji predmeti*¨: boca, stol, stolac, kauč, biljka, tv

VOC2012 sastoji se od oko 11 530 slika koje sadrže 27 450 označenih objekata, od čega je 1161 slika na kojima su automobili. Problem je što su te slike dosta „teške“, točnije na nekima se automobili slabo do jedva vide. Anotacije su dostupne preko VOC *toolkit*-a.

Slika 4.7.1. Primjer neoznačene slike iz baze podataka VOC2012

5. Ostvarenje detekcije zauzeća parkirnih mjesta

Sustav detekcije zauzeća parkirnih mjesta ostvaren je u programskom jeziku Python, verzije 3. Od važnijih biblioteka za Python korišteni su OpenCV, NumPy, scikit-learn, Eclipse Paho MQTT i ONVIF.

OpenCV (*Open Source Computer Vision Library*) je *open source* biblioteka za *real-time* računalni vid i strojno učenje. Nudi više od 2500 optimiziranih algoritma, koji uključuju klasične i *state-of-the-art* algoritme strojnog učenja i računalnog vida. NumPy je biblioteka koja se koristi za optimizirane računske operacije s velikim poljima i matricama, koje su najčešće izlazi neuronskim mreža.Eclipse Paho MQTT omogućava klijent implementaciju MQTT protokola preko Pythona. MQQT protokol najčešći je protokol za komunikaciju IoT (*Internet of Things*) aplikacija. Scikit – learn je jednostavna i učinkovita biblioteka za predviđanje analize podataka. Ostvarena je pomoću NumPy, SciPy i matplotlib-a. ONVIF nudi standarnizirana sučelja za učinkovitu interoperabilnost IP baziranih proizvoda. U kontekstu ovog rada koristio se za komunikacijama sa PTZ (*Pan-tilt-zoom*) kamerama preko SOAP protokola.

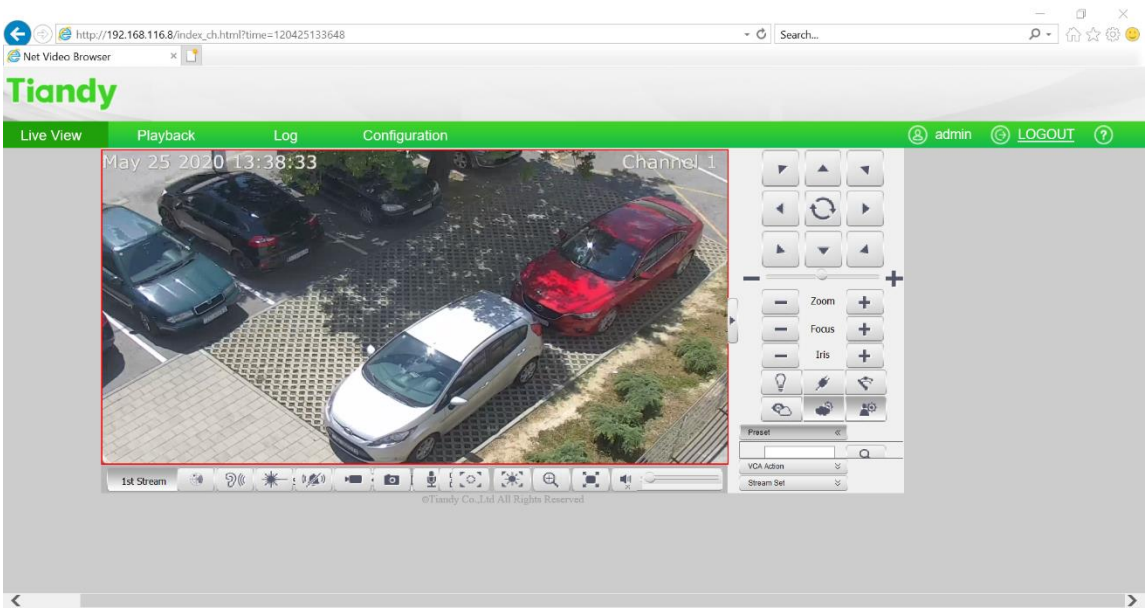
Za kamere korištene su 2 vrste Tiandy kamera:

1. PTZ kamera (pokretna kamera, sa mogućnosti uvećavanja slike)
2. „Obična“ IR (*Infrared*) kamera

Slika 5.1. IR kamera

Slika 5.2. PTZ kamera

PTZ kamera ima mogućnost spremanja *preset*-ova (određenih pozicija). *Preset* se odredi ručno preko ugrađenog web sučelja Tiandy kamere. Pritiskom na strelice sa desne strane pomiče se kamera u pritisnutom smjeru. Kada je željena pozicija odabrana sprema se pod *preset* i onda je moguće prebaciti kameru u *preset* pomoću SOAP protokola u intervalima.

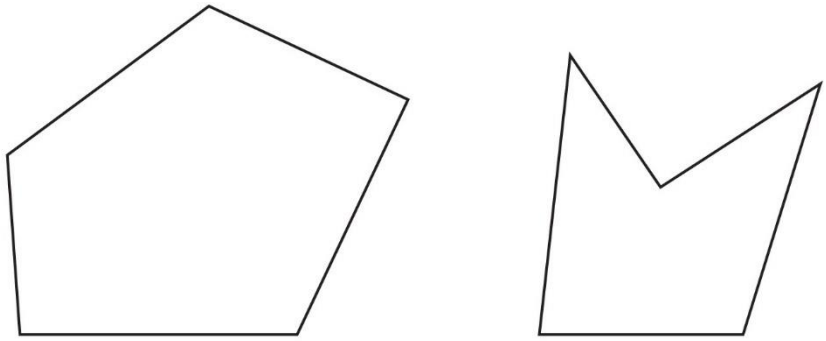


Slika 5.3. Web sučelje PTZ kamere

5.1. Automatska detekcija parkirnih mjesta

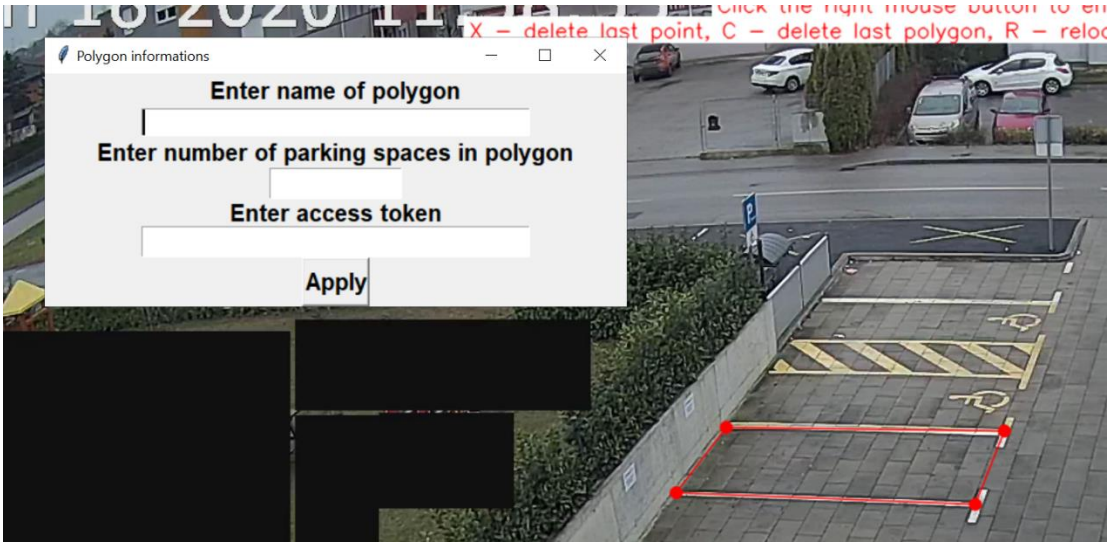
Kao što je spomenuto u drugom poglavlju, početni korak korištenja sustava detekcije parkirnih mjesta je označavanje područja interesa na slici, odnosno parkirnih mjesta. Većinom nam nije potrebno gledati zauzetost svih parkirnih mjesta stoga se mogu ručno nacrtati parkirna mjesta čija nas zauzetost zanima. Bitno je napomenuti da nacrtani poligon treba biti konveksan, ali u slučaju da nije primjenjuje se rekurzivan algoritam koji ga pretvara iz konkavnog u konveksan. Cilj je uzeti proizvoljan poligon A i segmentirati ga na konveksni i konkavni dio $A_{konv}, A_1, A_2, ..$ tj. na neki broj disjunktnih područja, koja su konveksna, a u razlici daju originalni poligon. A_{konv} je konveksan i vrijedi $A_{konv} = A \cup A_1 \cup A_2 \cup \ldots$ odnosno $A = A_{konv} \setminus (A_1 \cup A_2 \cup \ldots)$. To radimo tako da se krećemo po poligonu u smjeru obrnutom od kazaljke na sati, i gledamo vanjski kut na trenutnom vrhu (recimo vrhu i). Ako je taj kut manji od 180, onda se nalazimo u jednom od konkavnih djelova. Nastavljamo se kretati dok ne dodemo do vrha kojem je opet

vanjski kut manji od 180, recimo da je to vrh j. Tada vrhovi $i-1$ do j i sami čine poligon, sa svojstvom da je on konveksan, a ako ga zbrojimo sa originalnim poligonom, dobivamo 'konveksniji' poligon, to jest rjesavamo se jednog od njegovih konkavnih djelova. Drugim riječima, dobili smo A_k za neki k. Nakon toga vrhove i do $j-1$ izbacujemo iz originalnog poligona. Ovaj proces ponavljamo sve dok nas poligon nije do kraja konveksan, čime dobivamo A_{konv}, moguće da je potrebno napraviti više 'krugova' oko poligona, ali svakako će to biti konačan broj puta. Taj korak je nužan kako bi se mogla izračunati pripadnost točke poligonu kasnije.



Slika 5.1.1. Primjer konveksnog (lijevo) i konkavnog (desno) poligona

Pri crtanju parkirnih mjesta potrebno je označiti poligona pomoću miša i unijeti metapodatke o poligonu (ime poligona, broj mjesta koji zauzima, *access token*). *Access token* je potreban kako bi se mogla ostvariti komunikacija sa platformom pomoću MQTT protokola i javila zauzetost određenog poligona.



Slika 5.1.1. Primjer za unos metapodataka poligona

Ostvarena je i druga opcija, automatske detekcije parkirnih mjesta pomoću područja interesa tj. poligona koji obuhvaćaju više parkirnih mjesta. Kada se veći poligon označi sustav se pokrene na određeno vrijeme, te vodi evidenciju o očitanim vozilima unutar poligona i upisuje u datoteku *x* i *y* koordinatu centroida očitanog vozila za svaki poligona. Nakon toga primjenom jednog od 2 navedena algoritma grupiranja značajki:

- K-srednjih vrijednosti (engl. *K-means*)
- Gaussova mješavina (engl. *Gaussian mixture*)

pronalazi centar svih prijašnjih centroida i tehnikom N najbližih centroida (u postotku) konstruira novi poligon koji predstavlja parkirno mjesto (žute boje na slici), te konstruira kružnicu koja obgrli vanjske točke novodobivenog poligona.

Nakon rezultata algoritma na svakoj slici vidi se prikaz određenih podataka u gornjem dijelu slike. Oni su:

- *Max points* – postotak najbližih točaka od izračunatog centroida
- *N_clusters* – broj K klasa za algoritam, odnosno broj parkirnih mjesta

- *Cluster n_init* – broj iteracija koliko će se algoritam vrtiti, konačni rezultati dobivaju se kao najbolji izlaz uzastopnih vrtnji algoritma
- *Precalculate_centroids* – ako je postavljeno na *True*, početne točke algoritma nisu nasumične već su izračunate kao srednja vrijednost x i y koordinate svakog parkirnog mjesta
- *Total points* – ukupan broj točaka s kojima algoritam računa centoride
- *Distinct* – broj točaka bez duplikata
- *Filtered points* – izračunat broj točaka koje se uzimaj u obzir pri crtanju poligona
- *Distinct* – broj točaka bez duplikata koje se uzimaju u obzir pri crtanju poligona

Slika 5.1.2. Opis podataka na za algoritme

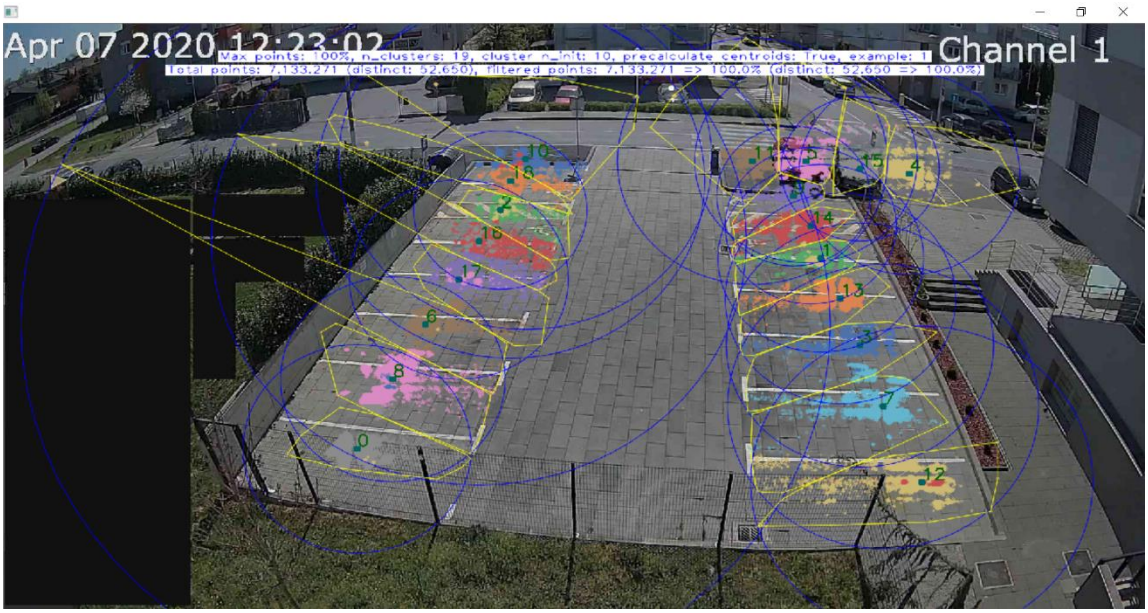
5.1.1. Algoritam k-srednjih vrijednosti

Algoritam k-srednjih vrijednosti je najjednostavniji i najpoznatiji algoritam grupiranja značajki. Algoritmom se neoznačeni primjeri grupiraju u K čvrstih grupa, gdje se parametar K zadaje unaprijed (u našem slučaju broj parkirnih mjesta unutar označenog poligona).

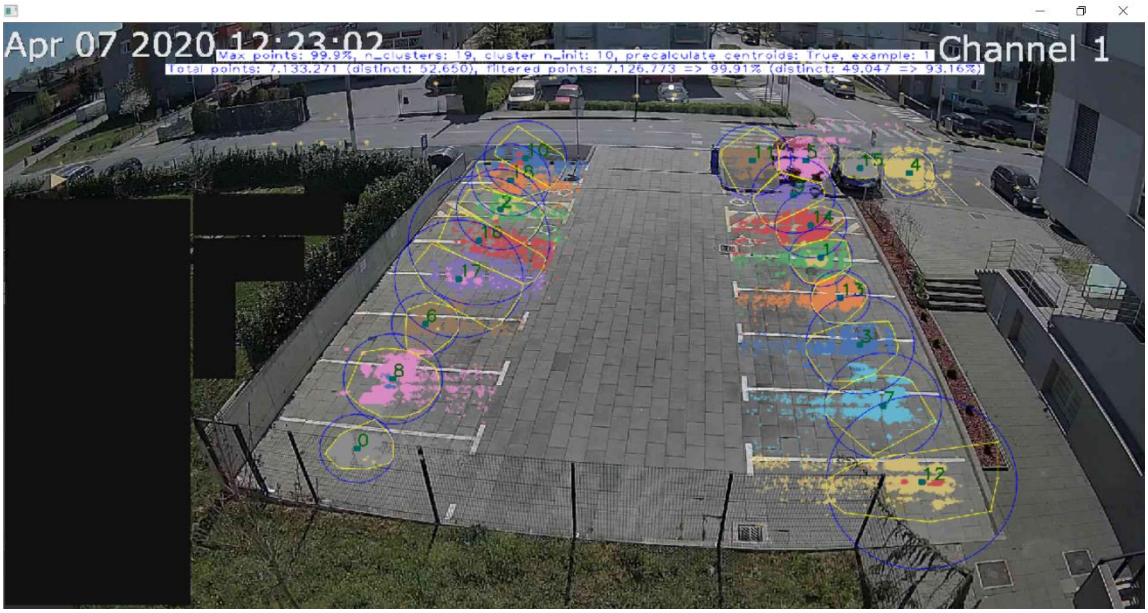
Algoritam 1. Algoritam k-srednjih vrijednosti

inicijaliziraj centroide $\boldsymbol{\mu}_k$, $k = 1, \ldots, K$ (npr. na slučajno odabrane $\mathbf{x}^{(i)}$)
ponavljaj
  za svaki $\mathbf{x}^{(i)} \in \mathcal{D}$
    $$b_k^{(i)} \leftarrow \begin{cases} 1 & \text{ako } k = \operatorname{argmin}_j \|\mathbf{x}^{(i)} - \boldsymbol{\mu}_j\| \\ 0 & \text{inače} \end{cases}$$
  za svaki $\boldsymbol{\mu}_k$, $k = 1, \ldots, K$
    $\boldsymbol{\mu}_k \leftarrow \sum_{i=1}^{N} b_k^{(i)} \mathbf{x}^{(i)} / \sum_{i=1}^{N} b_k^{(i)}$
dok $\boldsymbol{\mu}_k$ ne konvergiraju

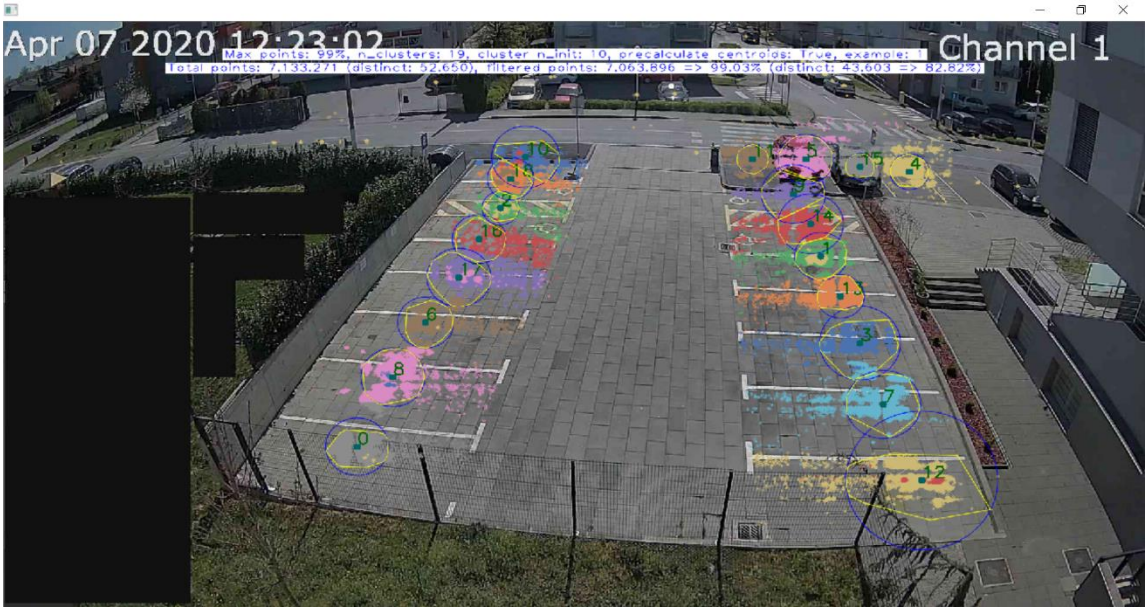
Slika 5.1.1. Prikaz pseudokoda algoritma k-srednjih vrijednosti



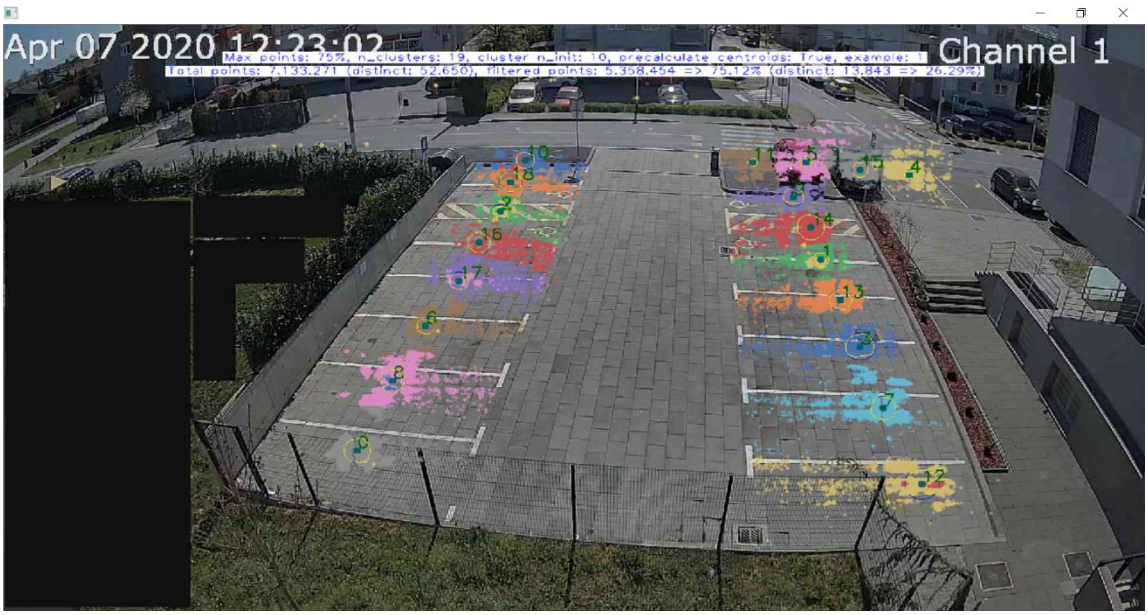
Slika 5.1.1. Izračunati poligoni preko k-sredina sa 100 % najbližih točaka



Slika 5.1.2. Izračunati poligoni preko k-sredina sa 99.9 % najbližih točaka



Slika 5.1.3. Izračunati poligoni preko k-sredina sa 99 % najbližih točaka



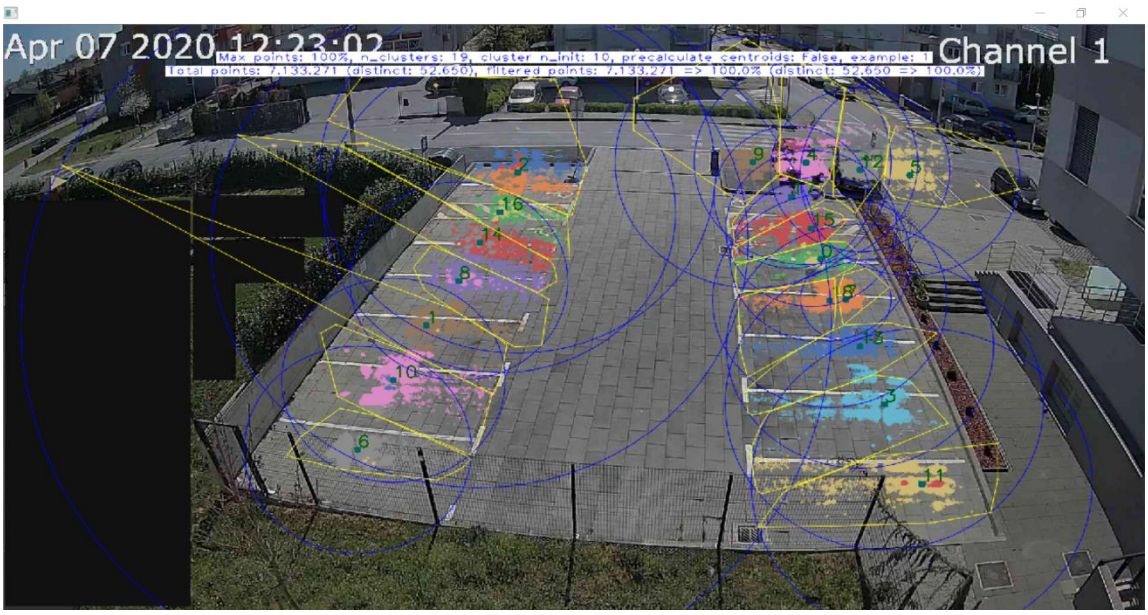
Slika 5.1.4. Izračunati poligoni preko k-sredina sa 75 % najbližih točaka

Sa slika se jasno vidi da nema smisla uzimati sve točke u obzir pri pravljenju novih poligona. Također je očito kako 75 % čini premalen poligon, te neće dovoljno često centroid detektiranog vozila upadati unutar poligona. Još se vidi koliku razliku čine 99.9 % i 100 %. Za potrebe sustava postotak koji se koristi je 99 % jer daje najbolje rezultate.

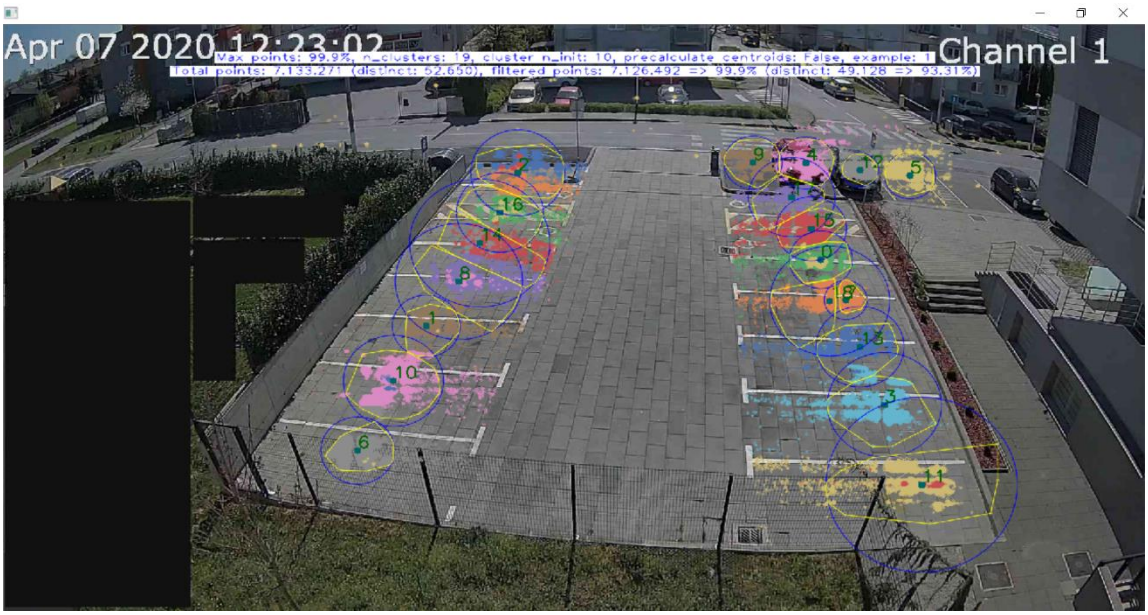
5.1.2. Algoritam mješavine Gaussovih gustoća

Sada ćemo razmotriti probabilistički pristup grupiranju. Za razliku od algoritma k-srednjih vrijednosti, kod kojega su granice između grupa čvrste, kod probabilističkog pristupa primjeri grupama pripadaju s određenom vjerojatnošću. Drugim riječima, jedan te isti primjer može pripadati u dvije ili više grupa, što dovodi do mekih granica između grupa. [1] Pretpostavljamo da se primjer iz svake grupe pokorava nekoj teorijskom razdiobi. Umjesto da modeliramo izglednosti za

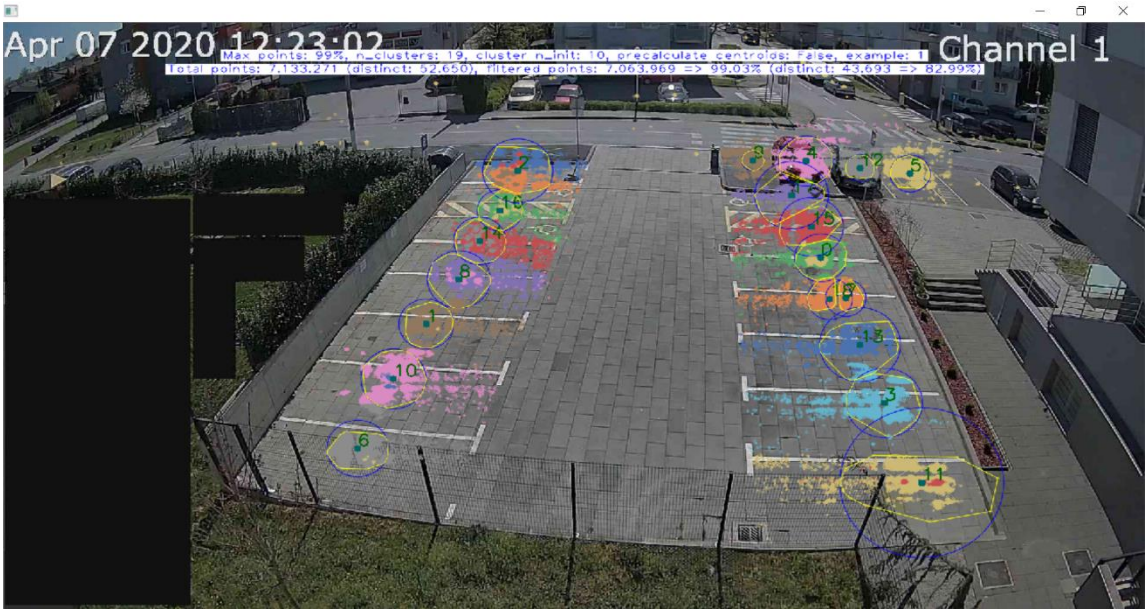
svaku pojedinačnu grupu (klasu), rješava se problem modeliranja gustoće vjerojatnosti $p(x)$. Funkcija gustoća $p(x)$ općenito je složnijeg oblika, stoga se koristi miješani model sačinjen od linearne kombinacije K osnovnih razdioba. Svaka od navedenih razdioba pripada određenoj klasi, te model utvrđuje s kojom vjerojatnošću određen primjer pripada određenoj grupi.



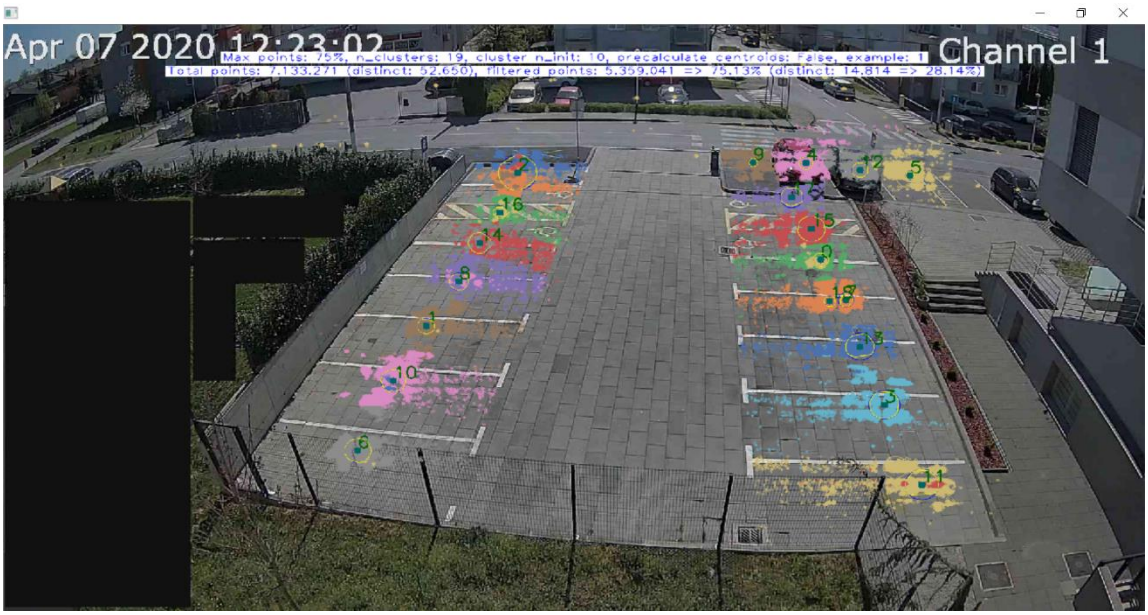
Slika 5.1.2.1. Izračunati poligoni preko mješavine Gaussovim gustoćama sa 100 % najbližih točaka



Slika 5.1.2.2. Izračunati poligoni preko mješavine Gaussovim gustoćama sa 99.9 % najbližih točaka



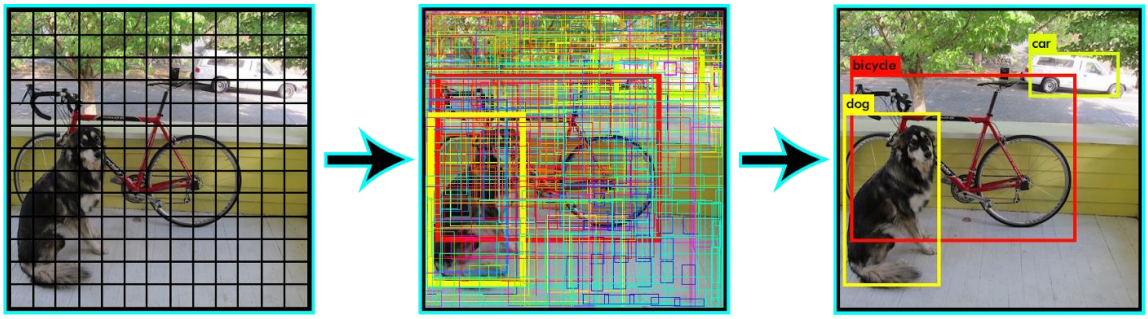
Slika 5.1.2.3. Izračunati poligoni preko mješavine Gaussovim gustoćama sa 99 % najbližih točaka



Slika 5.1.2.3. Izračunati poligoni preko mješavine Gaussovim gustoćama sa 75 % najbližih točaka

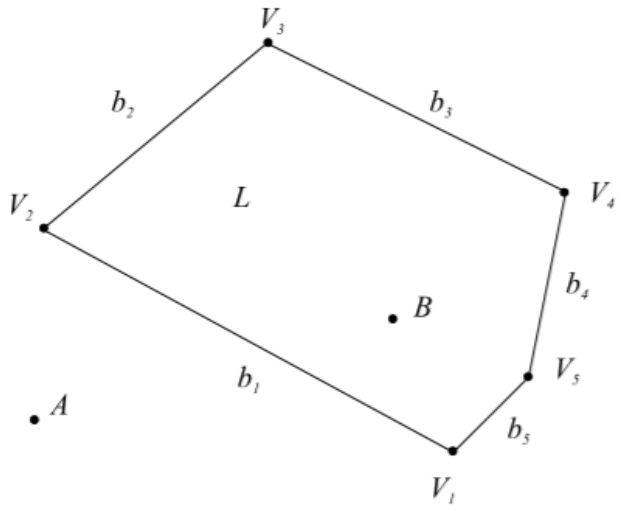
5.2. Detekcija vozila na parkirnih mjestima

Kada je *set-up* napravljen pokreće se glavni program. Pokretanjem glavnog programa detektiraju se sva vozila na slici (automobili, busevi, motocikli i kamioni). Kada se detektira vozilo model vraća *bounding box* detektiranog objekta. Točnije izlaz modela su gornje lijeve x i y koordinate (s obzirom na sliku) i širina i visina *bounding box*-a zajedno s pouzdanošću (broj u intervalu $[0,1]$) tog objekta. Što je pouzdanost veća, model je sigurniji da je to klasa koja je izlaz modela.



Slika 5.2.1. Proces dobivanja izlaza modela

Nakon što imamo dobivene koordinate, lako se dobiva centroid objekta (njegove x i y koordinate). Pomoću koordinata centroida može se izračunati pripadnost određenom poligonu, zapravo zauzeće parkirnog mjesta. Kako je prije navedeno potrebno je da poligon konveksan, jer inače nije moguće točno izračunati odnos točke i poligona.



Slika 5.2.2. Konveksan poligon L s točkama A i B

Želimo utvrditi u kakvom je odnosu proizvoljna točka i konveksan poligon L, odnosno je li točka unutar poligona ili nije. Potrebno je gledati odnos svakog brida i zadane točke. Uz to treba znati orijentaciju bridova poligona. Kriterij glasi:

1. Točka T je unutar poligona ako su vrhovi poligona zadani u smjeru kazaljke na satu i vrijedi:

   $$(\forall i) T * B_i \leq 0 \; za \; 1 \; \leq i \leq n$$

   Dakle točka je unutar poligona ako su vrhovi poligona zapisani u smjeru kazaljke na satu i ako je točka ispod svakog vrha.

2. Točka T je unutar poligona ako su vrhovi poligona zadani u suprotnom smjeru kazaljke na satu i vrijedi:

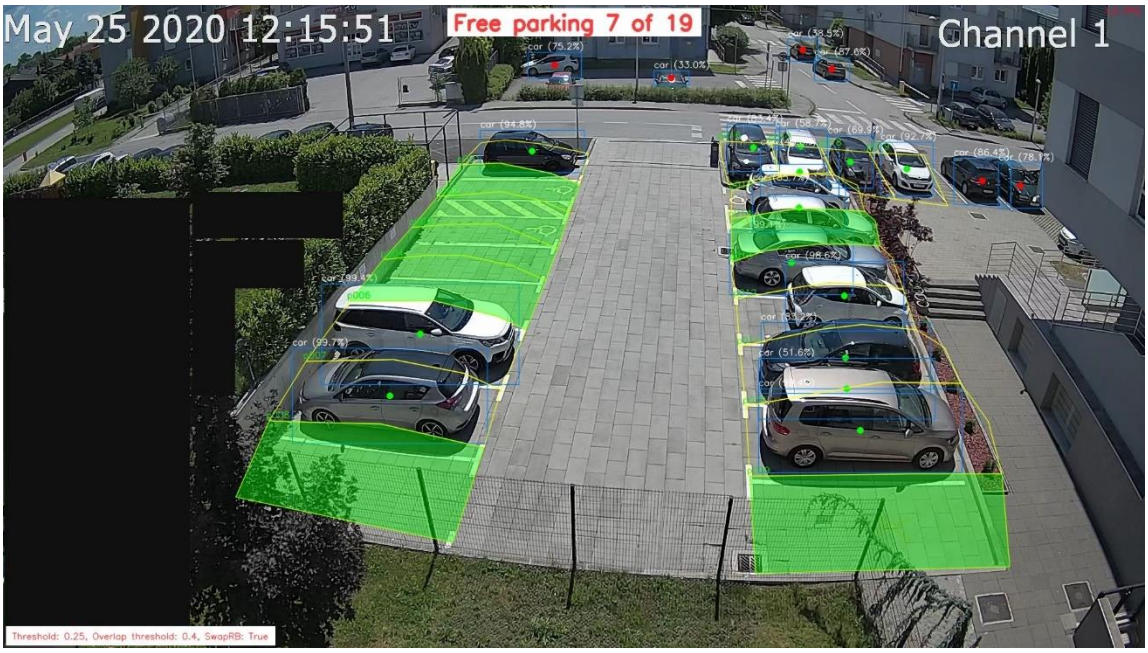
   $$(\forall i) T * B_i \geq 0 \; za \; 1 \; \leq i \leq n$$

   Dakle točka je unutar poligona ako su vrhovi poligona zapisani u suprotnom smjeru kazaljke na satu i ako je točka iznad svakog vrha.

3. Točka T je izvan poligona ako nije unutra.

Primjenom navedenih pravila i nužnim uvjetom da je zadan poligon konveksan možemo izračunati je li parkirno mjesto zauzeto. Kako je poznat podatak koliko svako parkirno mjesto ili skup parkirnih mjesta preko poligona imaju ukupno mjesta, kada vozilo zauzima parkirno mjesto onda se broj zauzetih mjesta oduzima od ukupnog broja i tako se saznaje broj slobodnih mjesta. Taj podatak tada se šalje MQTT protokolom na platformu koja onda šalje broj mjesta na LED zaslon na stvarnu fizičku lokaciju parkinga. Uz navedene funkcionalnosti dodana je i funkcionalnost izglađivanja (engl. *smoothing*). *Smoothing* je dodan radi nedeterminističkog ponašanja neuronske mreže. Izlaz neuronske mreže dobiva se svakih nekoliko sekundi, te se nekad dogodi da iako je mjesto zauzeto mreža kaže da je mjesto slobodno. *Smoothing* zapravo služi kako bi se izbjegla nepravilnosti u platformi, točnije mjesto postaje slobodno tek kada mreža određen broj puta uoči da nema vozila na njemu, odnosno da je zauzeto kada je određen broj puta vozilo detektirano. Na slici *smoothing* je prikazan kao prijelazno stanje obojeno narančastom bojom.

Slika 5.2.3. Primjer izlaza sustava bez *smoothing*-a



Slika 5.2.4. Primjer izlaza sustava sa ručno nacrtanim poligonima

Slika 5.2.5. Primjer izlaza sustava sa automatsko određenim poligonima

Slika 5.2.6. Primjer izlaza sustava na drugačijoj kameri

Slika 5.2.7. Primjer izlaza sustava na prvom *preset*-u PTZ kamere

Slika 5.2.8. Primjer izlaza sustava na drugoj poziciji PTZ kamere sa primjerom *smoothing*-a

6. Usporedba modela i ostvareni rezultati

Za potrebe sustava korištena je *darknet*-ova neuronska mreža YOLO V3 [4]. Iako je YOLO davao dobre rezultate, napravljen je trening novih neuronskih mreža. *Darknet* omogućuje trening početnih težina pod nazivom *darknet.conv.74*, ali može se raditi i *fine-tune*-ing prethodno natreniranih težina. Prilikom treniranja postoji .cfg datoteka (konfiguracijska) u kojoj je definirana arhitektura mreža i parametri bitni za treniranje. U ovom radu korišteni su idući parametri treninga:

- *Batch* – grupa slika
- *Subdivisons* – podjela *batch*-eva na manje dijelove
- *Width* – širina slike na kojoj će se trenirati
- *Heigh* – visina slike na kojoj će se trenirati
- *Momentum* – mjera koja omogućava stabilan gradijent pri treniranju
- *Decay* – mjera koja održava da vrijednosti težina ne postanu prevelike
- *Learning rate* – stopa učenja
- *Burn in* – broj iteracija do koje stopa učenja polako raste dok ne postigne zadanu vrijednost
- *Max batches* – broj iteracija učenja
- *Steps* – broj iteracija kada se stopa učenja mjenja
- *Scales* – vrijednost s kojom se stopa učenja pomnoži u *steps* iteracijama
- *Random* – zastavica koja govori jesu li početne težine postavljene na nasumične vrijednosti (ako je 1 onda jesu, ako 0 nisu)
- *Anchors* – početne veličine (širina, visina) od kojih će neke (najbliže veličini objekta) biti promijenjene veličine pomoću nekih izlaza neuronske mreže

Za svaki trening je priložena tablica s vrijednosti parametara, grafom ovisnosti gubitka i iteracija i kratkim komentarom o treningu. Crvena linija na grafu

predstavlja m*AP* (*mean average precision*) odnosno srednju vrijednost preciznosti nad svim klasama na skupu za testiranje tijekom iteracija u postotku. Na kraju treninga se novodobivene težine pokreću na slikama parkinga na kojima radi sustav i procjenjuje se njihova kvaliteta.

mAP je definiran pomoću *IoU* (*intersection over union*). IoU je udio presjeka između detektiranog *bounding* box-a objekta i stvarnog *bounding box*-a objekta i unije njih. mAP govori je li objekt klasificiran ispravno. To je srednja vrijednost prosječne preciznosti svih klasa. Nakon što se uspostavi granica za IoU, detektirani objekt se označuje kao ispravan ili neispravan. Pomoću toga se izračuna broj *true positive*-a i *false positive*-a i onda pripadno graf preciznost-odziv. Tada se izračuna prosječna preciznost što je zapravo suma N maksimalnih preciznosti vrijednosti p po svim N odzivima vrijednosti $\bar{r}$ gdje je $\bar{r}$ veći od r.

$$mAP = \frac{1}{N}\sum_{r=0}^{N}\left(\max\left(p(\bar{r})\right)\right), gdje\ je\ \bar{r}\ \geq r$$

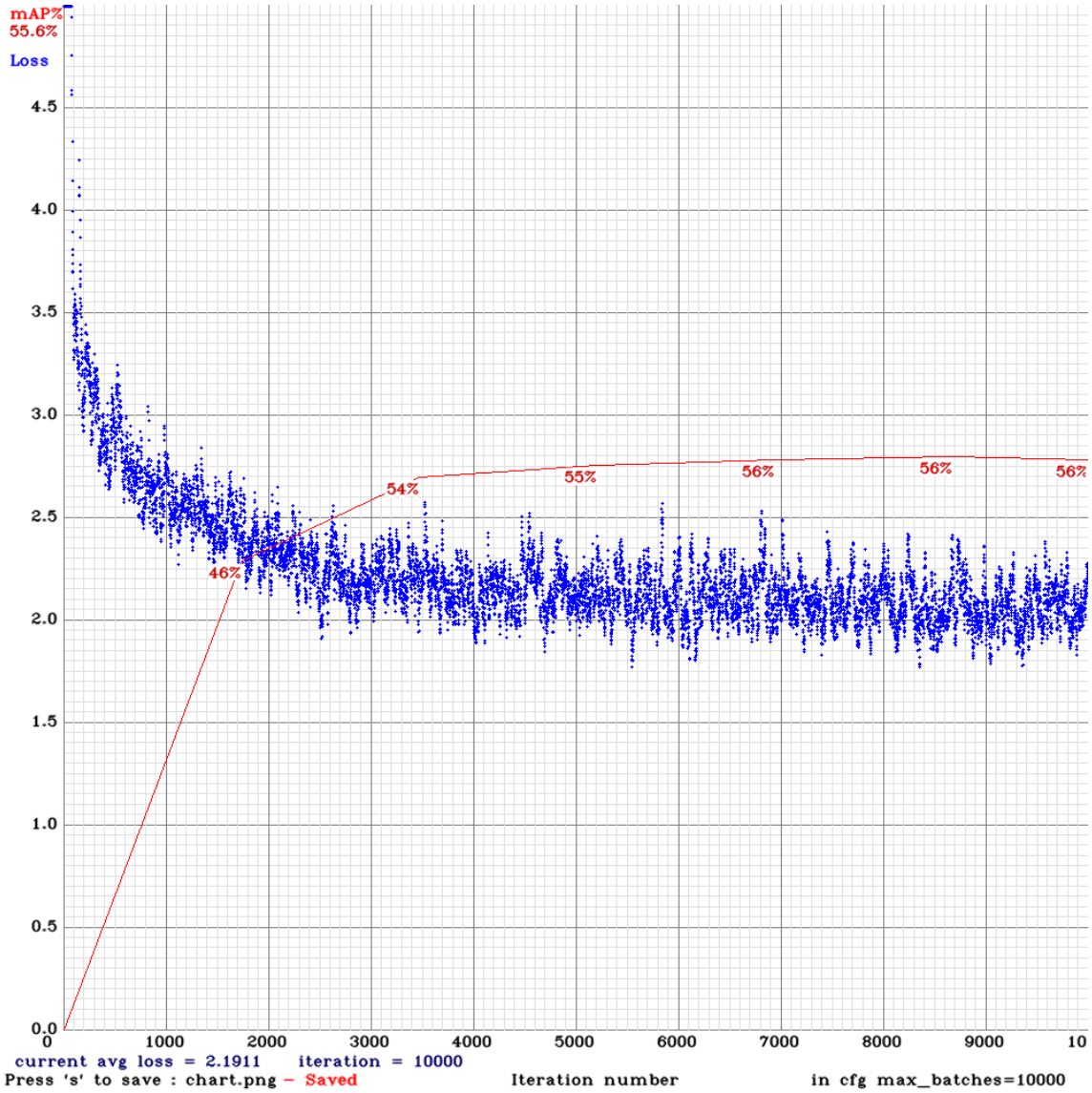
Funkcija gubitka definirana je kao suma srednjih kvadrata (engl. *mean squared error*). Ako je predikcija stvarne oznake za neku koordinatu $\hat{t}_*$ gradijent je predikcija oduzeta od vrijednosti stvarne oznake: $\hat{t}_* - t_*$ [4].

1. trening

Dataset	Kitti Vision + COCO
Početne težine	darknet.conv.74
Batch	32
Subdivisions	8
Width	416
Height	416
Momentum	0.9
Decay	0.0005

Learning rate	0.001
Burn in	100
Max batches	10000
Steps	2000, 5000
Scales	0.1, 0.1
Random	1
Anchors	10,13,16,30,33,23,30,61,62,45,59,119,116,90,15,198, 373,326

Stopa učenja postavljena je po najboljoj praksi na 0.001. Broj koraka postavljen je na 2, da bi se stopa učenja smanjila na 2000-oj i 5000-oj iteraciji. Greška je pala do globalnog optimuma koji ne daje dobre rezultate. *mAP* je finalno bio na 56 %, zbog toga što su COCO ima poprilično „teške“ slike za naučiti.



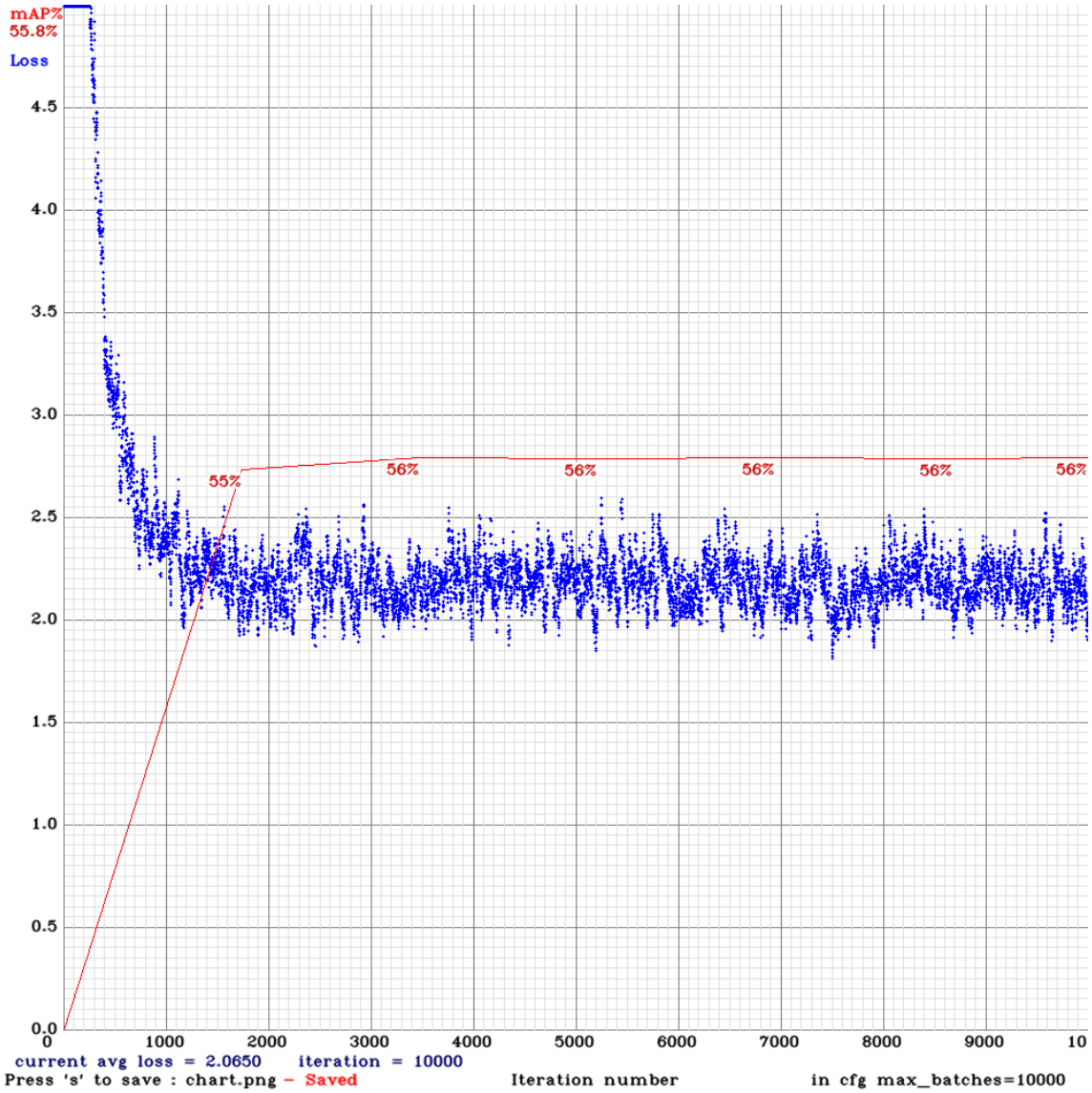
2. trening

Dodatno izračunati anchorsi u cilju boljeg treniranja Prvo pokrenut trening sa stopom učenja 0.01, kroz 200 iteracija greška divergirala u beskonačno zbog prevelike stope učenja. Nakon toga stopa učenja smanjena na 0.001 radi stabilnosti gradijenta. Povećan je broj koraka kako bi se stopa učenja dodatno smanjila s obzirom na prijašnji trening, radi postizanja manje vrijednosti funkcije gubitka.

Dataset	Kitti Vision + COCO

Početne težine	darknet.conv.74
Batch	32
Subdivisions	8
Width	416
Height	416
Momentum	0.9
Decay	0.0005
Learning rate	0.001
Burn in	500
Max batches	10000
Steps	1000, 2000, 3000, 5000, 7000
Scales	0.1, 0.1, 0.1, 0.1, 0.1
Random	1
Anchors	13,14,18,40,38,24,37,66,89,50,65,123,193,100,110,205,332,281

Iako je learning rate bio jako malen (10^{-7} nakon 7000 iteracija) trening nije uspio izaći iz optimuma.



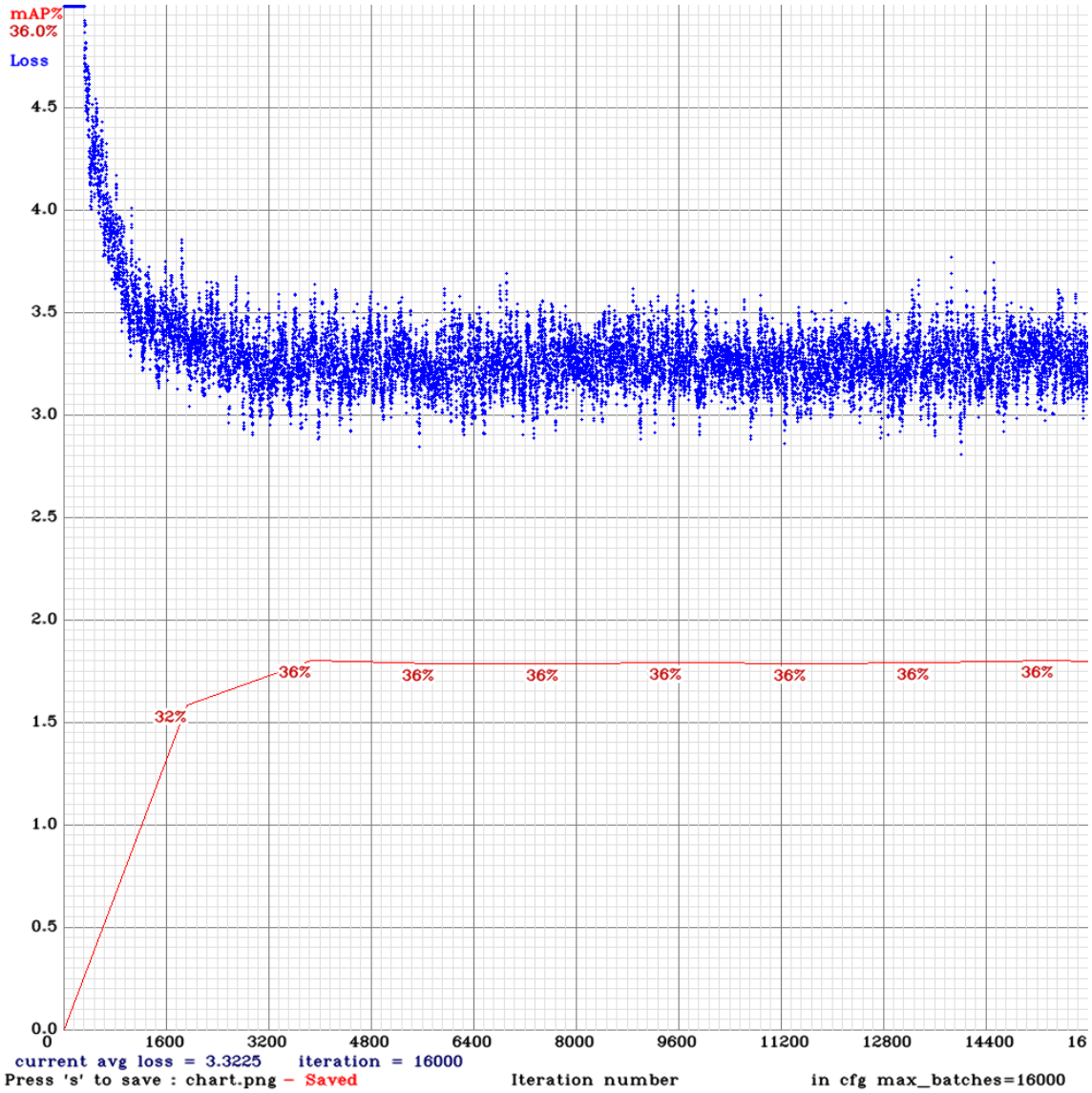
3. trening

Za dataset uzet COCO od prije i novi BDD dataset. Ukupno 15456 slika za treniranje , dok za testiranje ih ima 6631. Veći broj iteracija zbog većeg broja slika, te i stopa učenja manje pada, jer u prijašnjem treningu nije pokazan uspjeh s malom stopom učenja. Broj koraka postavljen na 4, kako bi se stopa učenja smanjila za 4 reda veličina. Time dobivamo dovoljno malu stopu učenja pri kraju

učenja da bi se omogućio izlazak iz lokalnog optimuma. Ponovno izračunati anchori za ovaj dataset.

Dataset	BDD100k + COCO
Početne težine	darknet.conv.74
Batch	32
Subdivisions	8
Width	416
Height	416
Momentum	0.9
Decay	0.0005
Learning rate	0.001
Burn in	300
Max batches	16000
Steps	1000, 3000, 5000, 10000
Scales	0.1, 0.1, 0.1, 0.1
Random	1
Anchors	7,10,16,20,31,34,45,69,75,41,83,104,204,87,129,183,334,276

Iako mu je gubitak veći i *mAP* manji, bolji su rezultati, ali i dalje nedovoljno dobri za upotrebu.

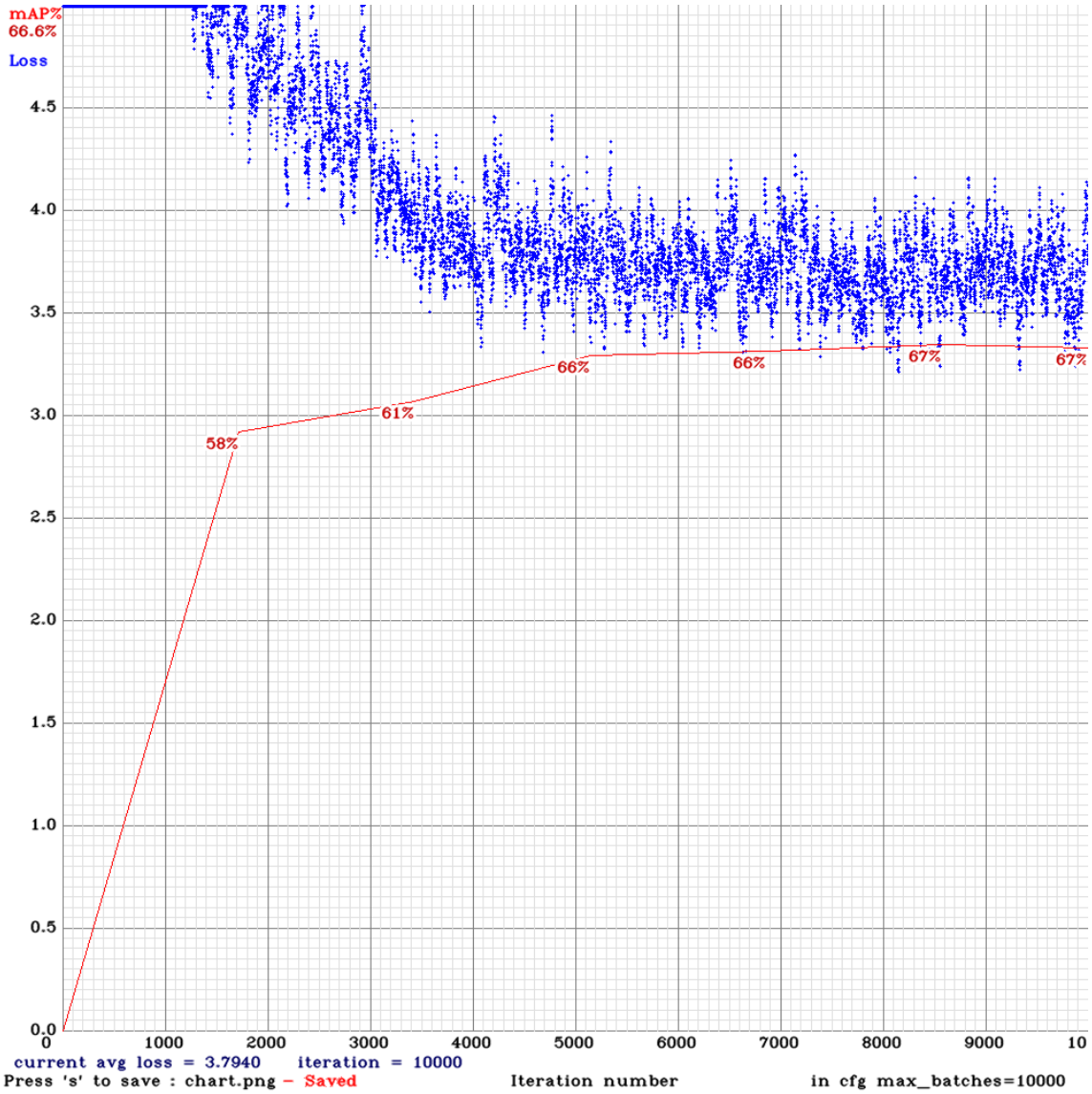


4. trening

Povećana rezolucija na 608x608, kako bi se dobili bolji rezultati. Sa batchom 32 dobiva se *Out of memory* greška zbog nedovoljne procesorske snage. Batch size tada promijenjen na 16. Učenje samo na BDD100k datasetu. Ponovo izračunati anchorsi za taj dataset u cilju dobre prakse i boljih rezultata. Burn in povećan na 1000 iteracija da mreži treba veći broj iteracija kako bi se "zagrijala", decay povećan na 0.01 radi stabilnosti gradijenta.

Dataset	BDD100k
Početne težine	darknet.conv.74
Batch	16
Subdivisions	8
Width	608
Height	608
Momentum	0.9
Decay	0.01
Learning rate	0.001
Burn in	1000
Max batches	10000
Steps	3000, 5000, 7000
Scales	0.1, 0.1, 0.1
Random	1
Anchors	8,13,16,24,28,40,47,56,65,87,105,120,69,192,149,190,199,296

mAP se povećao iako je gubitak veći, ali rezultati i dalje nedovoljno dobri za upotrebu.



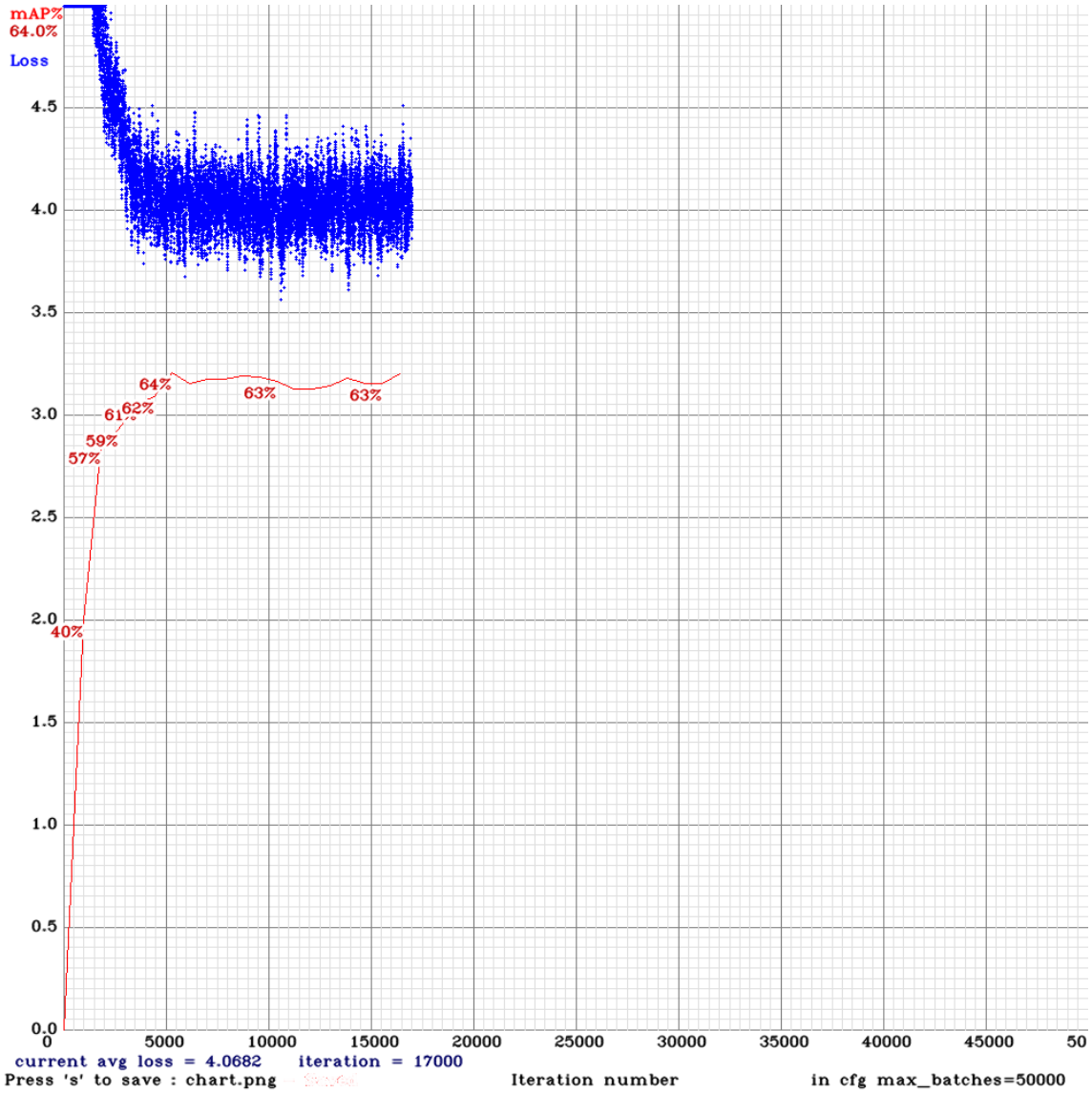
5. trening

Trening isti kao i prijašnji jedino s većim brojem iteracija.

Dataset	BDD100k
Početne težine	darknet.conv.74
Batch	32
Subdivisions	16
Width	608
Height	608

Momentum	0.9
Decay	0.01
Learning rate	0.0001
Burn in	1000
Max batches	50000
Steps	3000, 5000, 7000
Scales	0.1, 0.1, 0.1
Random	1
Anchors	8,13,16,24,28,40,47,56,65,87,105,120,69,192,149,190,199,297

Zaustavljen je nakon što je primijećeno da nema znatnih poboljšanja u odnosu na prethodni.



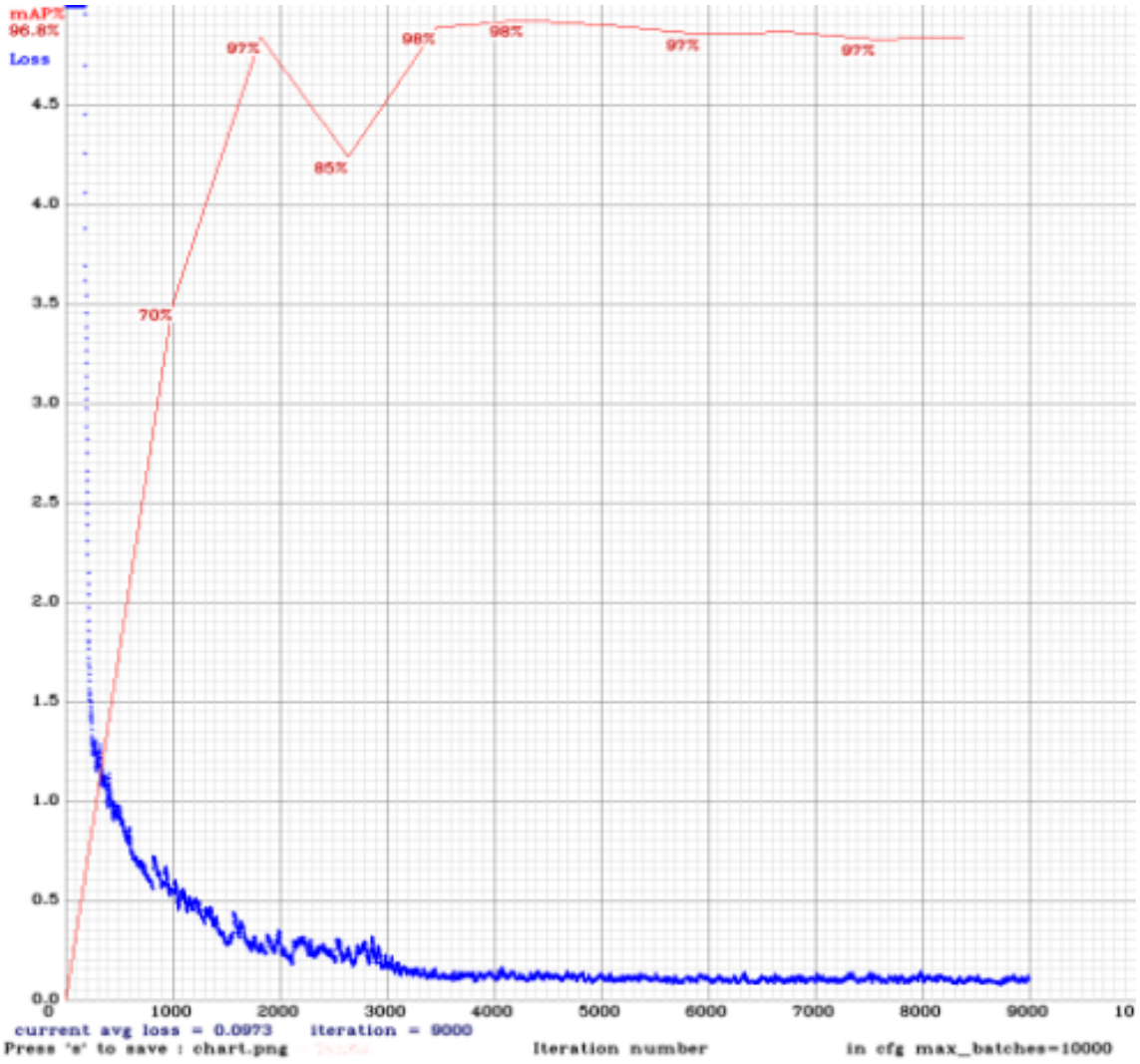
6. trening

Korišten Standford Cars skup podataka. Radi dobre prakse izračunati anchorsi za Stanford Cars skup podataka. Dimenzije slike kao i u prijašnjem treningu na 608x608 jer mreža postiže bolje rezultate na većim dimenzijama iako učenje duže traje.

Dataset	Standford Cars

Početne težine	darknet.conv.74
Batch	32
Subdivisions	32
Width	608
Height	608
Momentum	0.9
Decay	0.01
Learning rate	0.001
Burn in	1000
Max batches	10000
Steps	3000, 5000, 7000
Scales	0.1, 0.1, 0.1
Random	1
Anchors	315,230,353,368,440,296,539,261,550,342, 483,390,442,495,555,436,560,533

Postignuti znatno bolji rezultati, pretežito jer su slike u Standford Cars “lakše”. Gubitak pao do male vrijednosti i *mAP* postigao visoku vrijednost. Na slikama parkinga dobri rezultati za aute koji se jasno vide s prednje strane, dok za ostale ne.



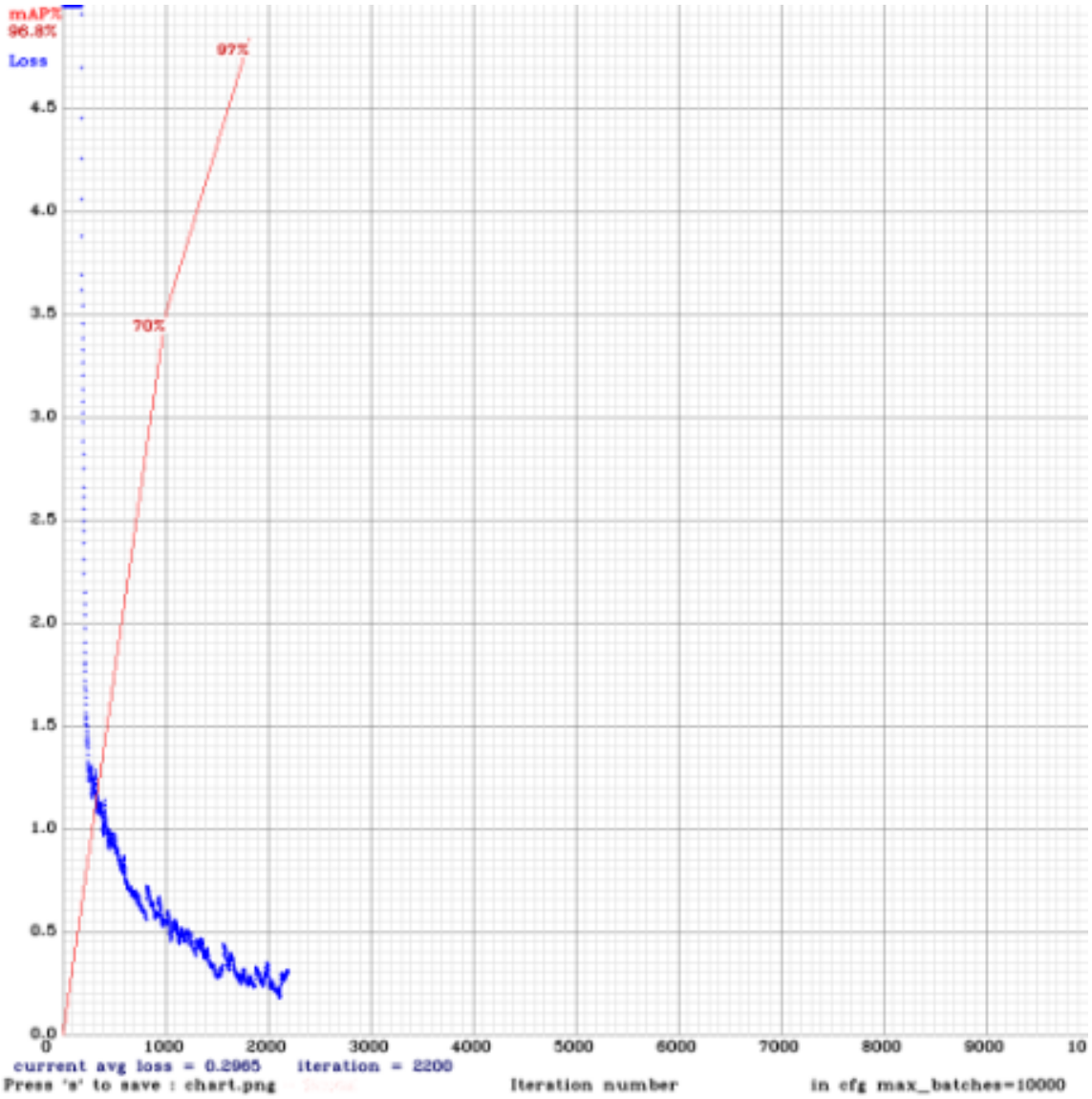
7. trening

Korišten cijeli Standford Cars, za razliku od prijašnjeg gdje je korišteno samo pola skupa podataka.

Dataset	Standford Cars
Početne težine	darknet.conv.74
Batch	32
Subdivisions	32
Width	608

Height	608
Momentum	0.9
Decay	0.01
Learning rate	0.001
Burn in	1000
Max batches	10000
Steps	3000, 5000, 7000
Scales	0.1, 0.1, 0.1
Random	1
Anchors	315,230,353,368,440,296,539,261,550,342, 483,390,442,495,555,436,560,534

Rezultati odlični za taj dataset, prepoznaje sve na testnim podacima, ali testni podaci su bitno različiti od stvarnih primjera. (97% preciznosti na testnim podacima). Gubitak stagnira nakon prikazane iteracije.



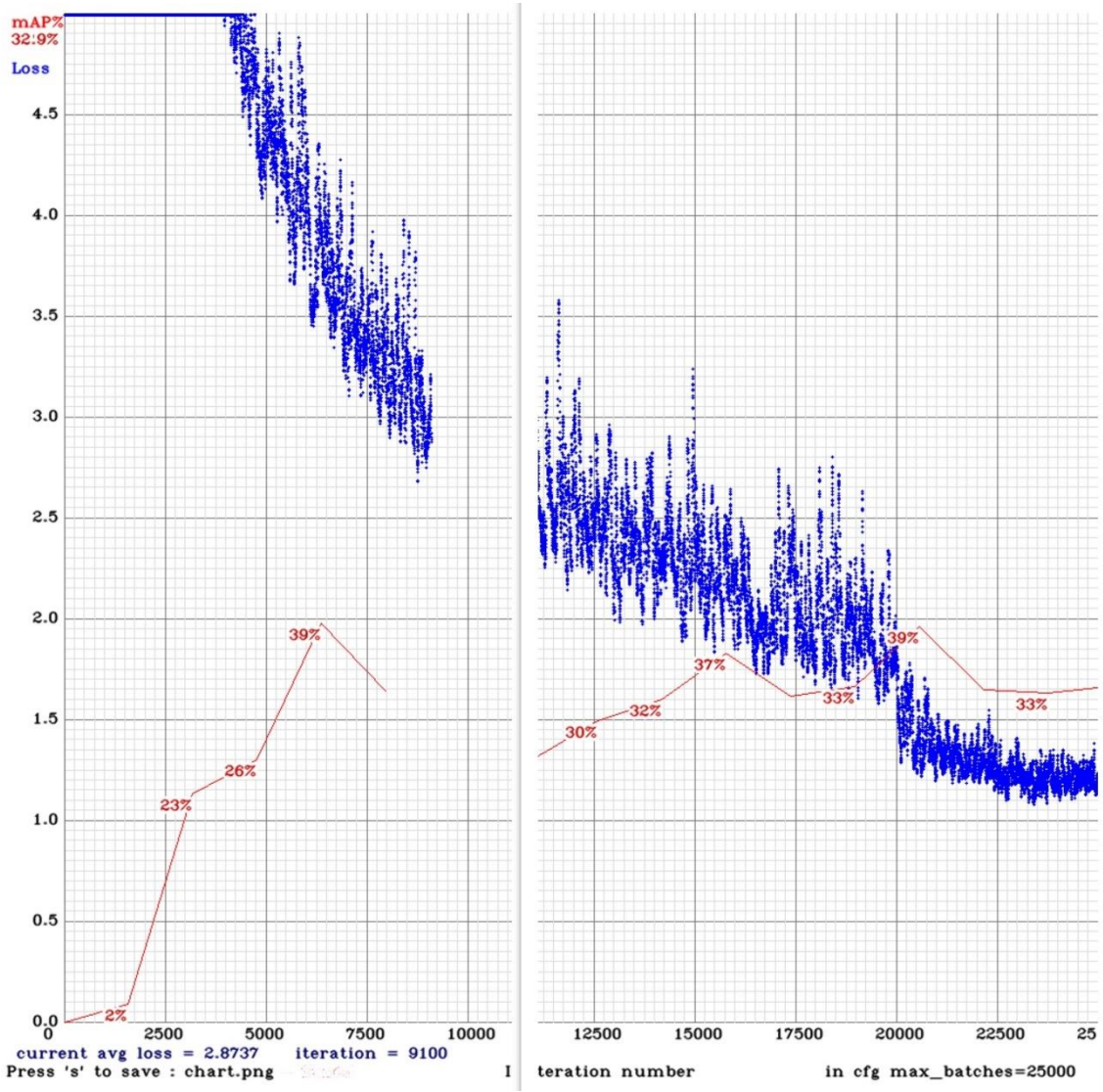
8. trening

Korišten je novi dataset UAVDT. Povećana je rezolucija za bolje rezultate.

Dataset	UAVDT
Početne težine	darknet.conv.74
Batch	64
Subdivisions	64
Width	704
Height	704

Momentum	0.9
Decay	0.01
Learning rate	0.001
Burn in	1000
Max batches	25000
Steps	20000, 22500
Scales	0.1, 0.1
Random	1
Anchors	10,13,16,30,33,23,30,61,62,45,59,119,116,90,15,198, 373,326

Vidi se naglo dobar pad funkcije gubitka kada se smanjila stopa učenja na 20000 iteraciji. Usred treninga ugasilo se računalo, stoga je slika prepolovljena. Iako se funkcija gubitka smanjuje, rezultati nisu dovoljno dobri za sustav, skup podataka ima izrazito male objekte na slikama.



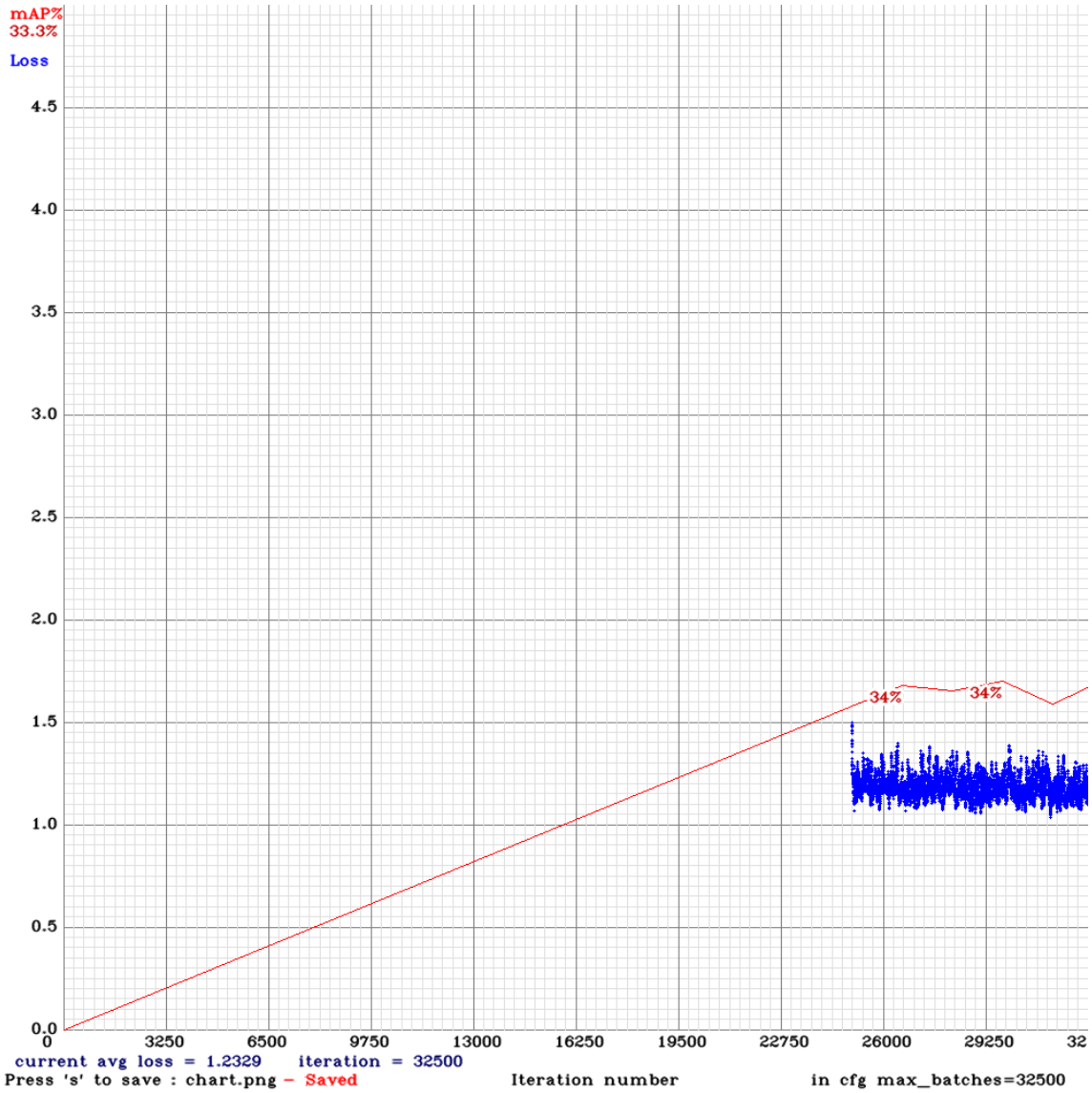
9. trening

Nastavljen trening nad skupom UAVDT, s manjom stopom učenja.

Dataset	UAVDT
Početne težine	UAVDT
Batch	darknet.conv.74
Subdivisions	64
Width	64

Height	704
Momentum	704
Decay	0.9
Learning rate	0.01
Burn in	0.0001
Max batches	1000
Steps	10000
Scales	8000, 9000
Random	0.1, 0.1
Anchors	10,13,16,30,33,23,30,61,62,45,59,119,116,90,15,198, 373,326

Rezultati se nisu poboljšali, iako sam mislio da će manja stopa učenja uzrokovati bolji pad funkcije gubitka kao što je to bilo kod prošlog treninga kod promjene stope učenja sa 0.001 na 0.0001.



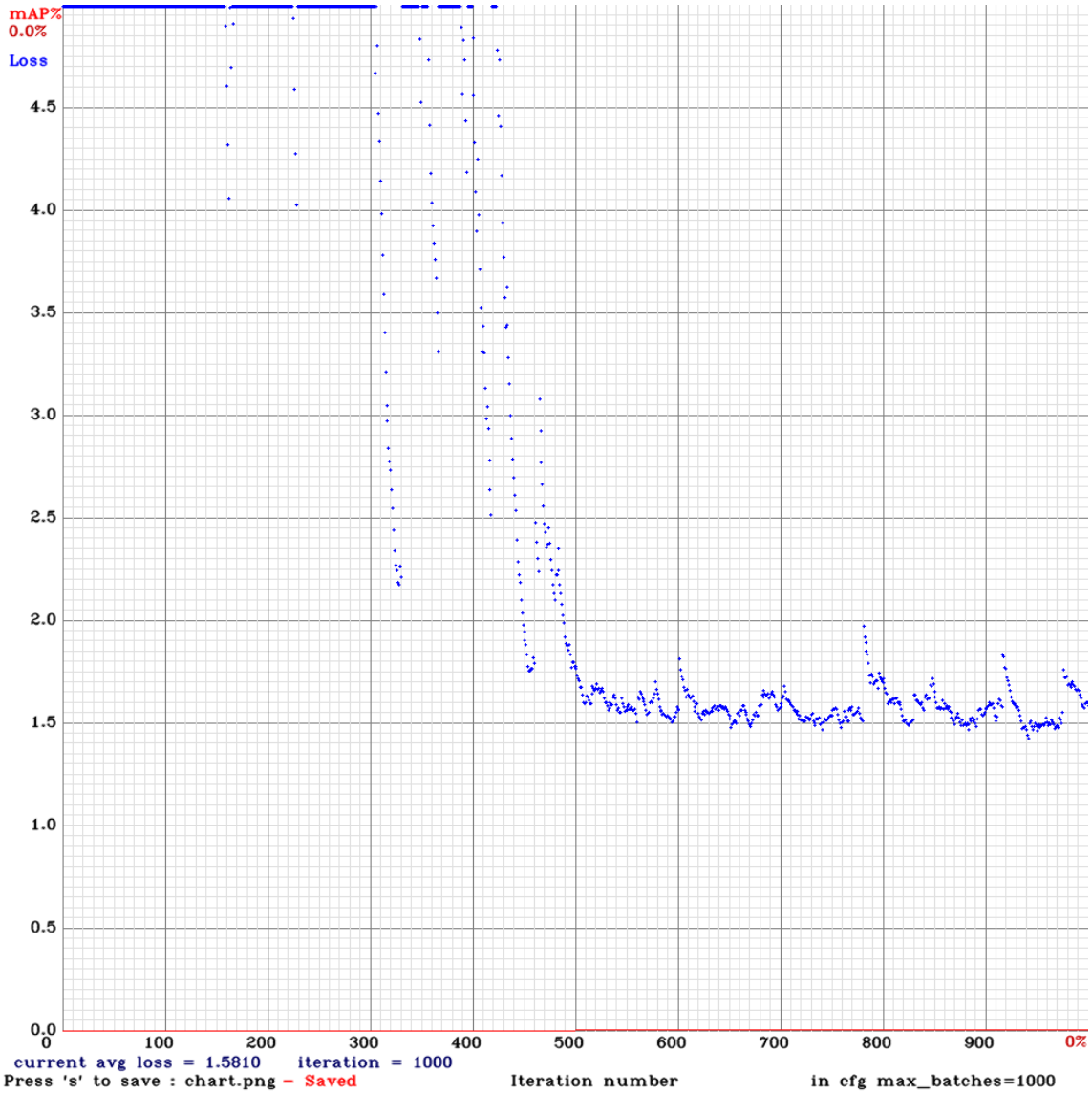
10. trening

Korišten novi skup Urban Tracker. Podaci u njemu su ručno označeni s video isječka.

Dataset	Urban Tracker
Početne težine	darknet.conv.74

Batch	64
Subdivisions	64
Width	416
Height	416
Momentum	0.9
Decay	0.01
Learning rate	0.001
Burn in	50
Max batches	1000
Steps	800, 900
Scales	0.1, 0.1
Random	1
Anchors	20, 24, 30, 31, 27, 42, 55, 47, 39, 74, 48, 87, 85, 58, 66,103, 107,114

Stopa učenja je prevelika stoga funkcija gubitka divergira i rezultira lošim rezultatima.

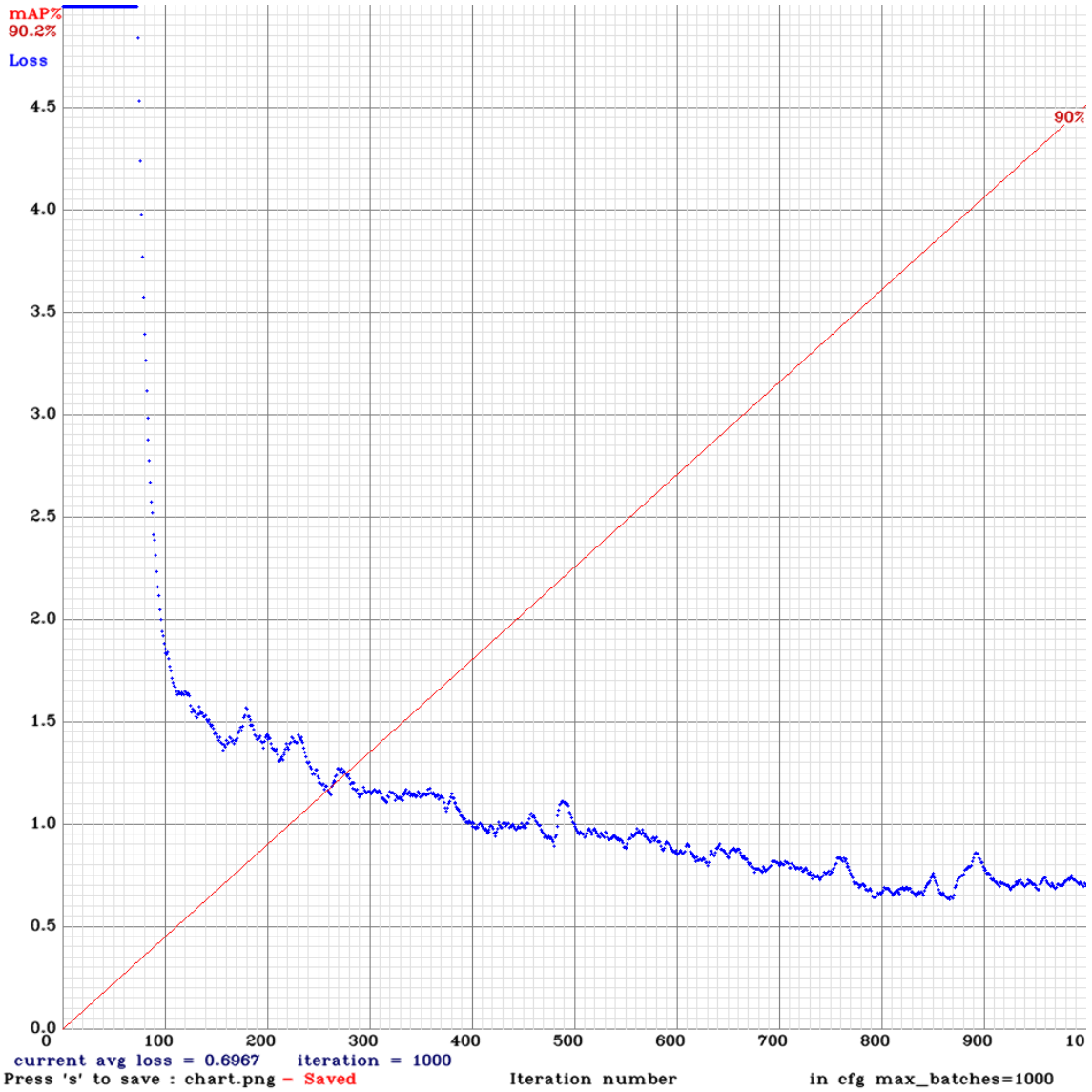


11. trening

Dataset	Urban Tracker
Početne težine	darknet.conv.74
Batch	64
Subdivisions	64
Width	416

Height	416
Momentum	0.9
Decay	0.01
Learning rate	0.0001
Burn in	50
Max batches	1000
Steps	800, 900
Scales	0.1, 0.1
Random	1
Anchors	20, 24, 30, 31, 27, 42, 55, 47, 39, 74, 48, 87, 85, 58, 66,103, 107,114

Ostvareni su dobri rezultati sa 90% *mAP*, ali je skup podataka relativno jednostavan. Ostvaruje zadovoljavajuće rezultate na stvarnim podacima, ali i dalje znatno lošije nego predtrenirani YOLO model.

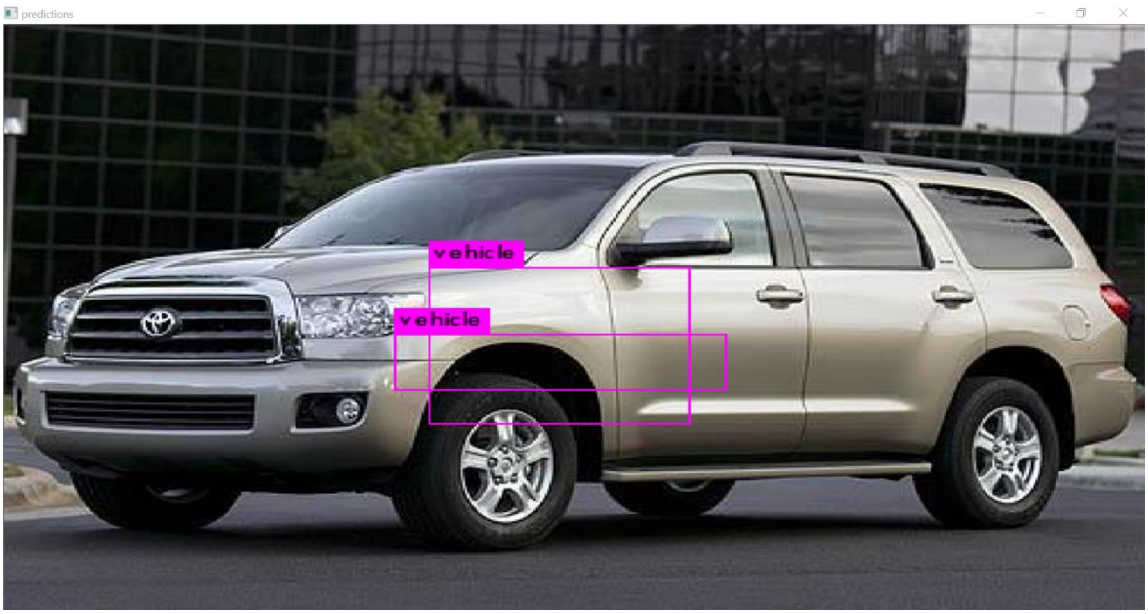


12. trening

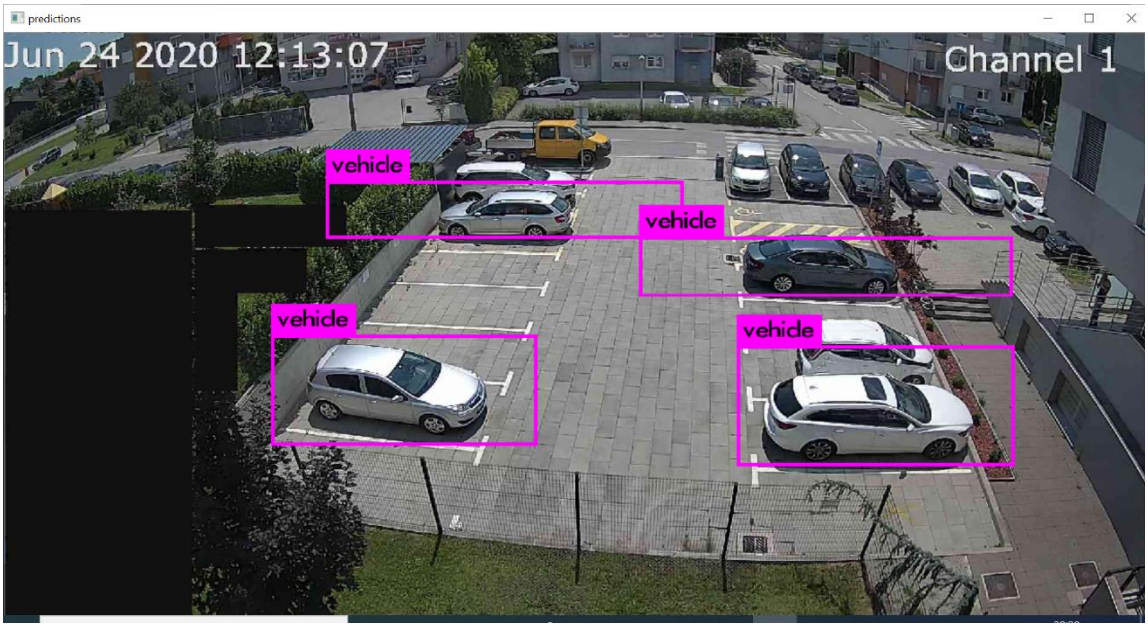
Korištena *transfer learning* metoda učenja. To je uzimanje prijašnje natreniranih težina jednim skupom i onda dodatno treniranje tih težina drugim skupom.

Dataset	Urban Tracker
Početne težine	težine nakon 7. treninga
Batch	64
Subdivisions	64
Width	416
Height	416
Momentum	0.9
Decay	0.01
Learning rate	0.0001
Burn in	50
Max batches	1000
Steps	800, 900
Scales	0.1, 0.1
Random	1
Anchors	20, 24, 30, 31, 27, 42, 55, 47, 39, 74, 48, 87, 85, 58, 66,103, 107,115

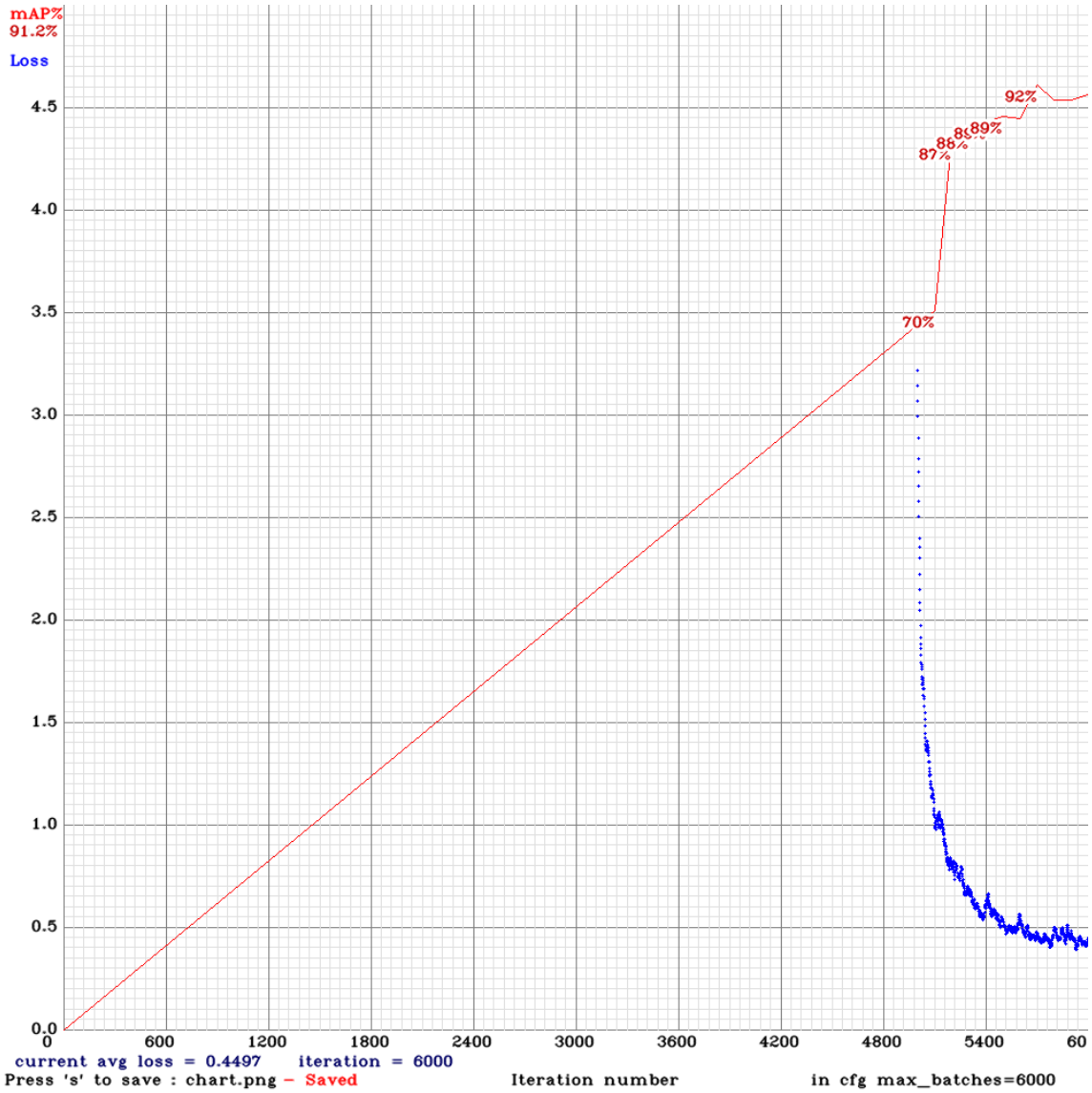
Dobri rezultati na testnom setu, loši rezultati na nekoliko slika testnog seta Standford cars, ali poboljšani rezultati na stvarnim slikama s pravog parkirališta s obzirom na prijašnje modele.



Slika 6.2. Primjer izlaza naučenih težina 12. Učenja na slici iz skupa Standford Cars



Slika 6.2. Primjer izlaza naučenih težina 12. učenja na stvarnom parkingu



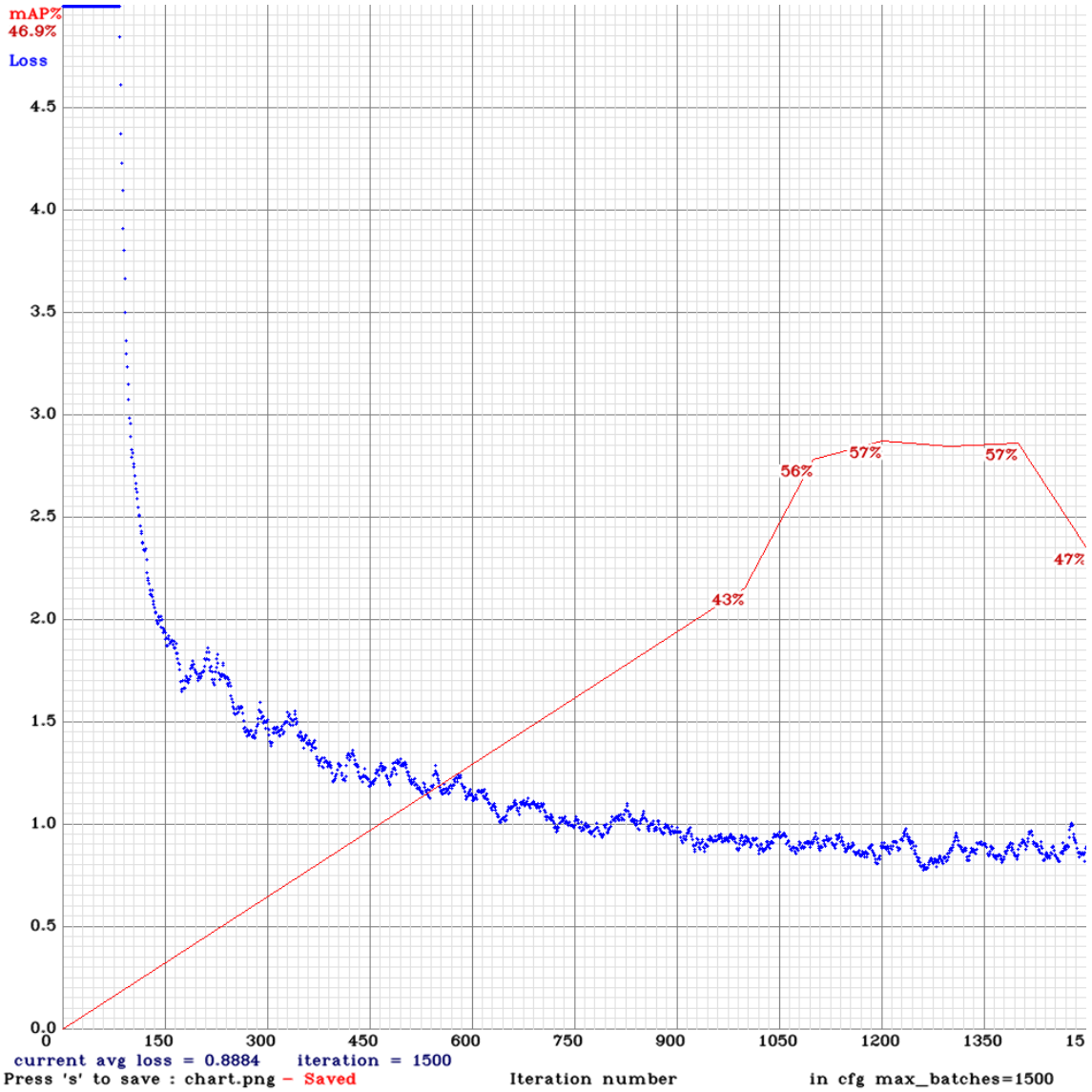
13. trening

Uzet poznat VOC dataset iz 2012. Od tamo su preuzete samo slike 61utomobile, te je trening pokrenut na tome.

Dataset	VOC 2012
Početne težine	darknet.conv.74

Batch	64
Subdivisions	64
Width	416
Height	416
Momentum	0.9
Decay	0.01
Learning rate	0.0001
Burn in	50
Max batches	1000
Steps	800, 900
Scales	0.1, 0.1
Random	1
Anchors	20, 24, 30, 31, 27, 42, 55, 47, 39, 74, 48, 87, 85, 58, 66,103, 107,116

Problem je što su auti na slikama jako mali, nekad i neprepoznatljivi za čovjeka. Rezultati nisu dovoljno dobri za upotrebu.

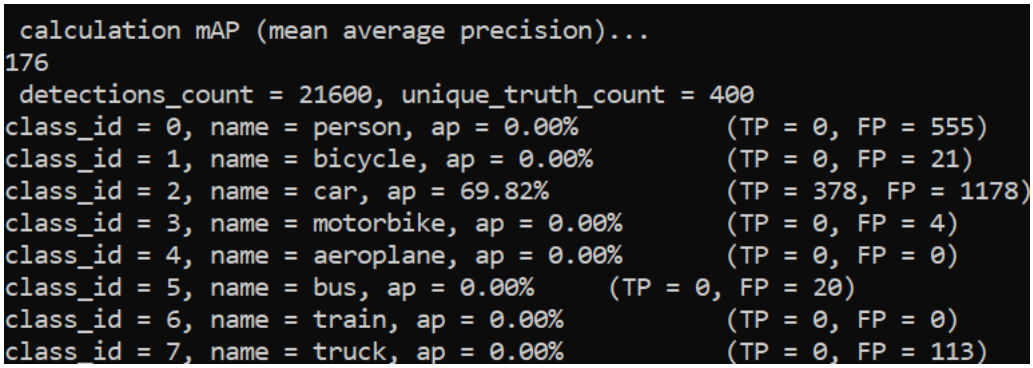


14. trening

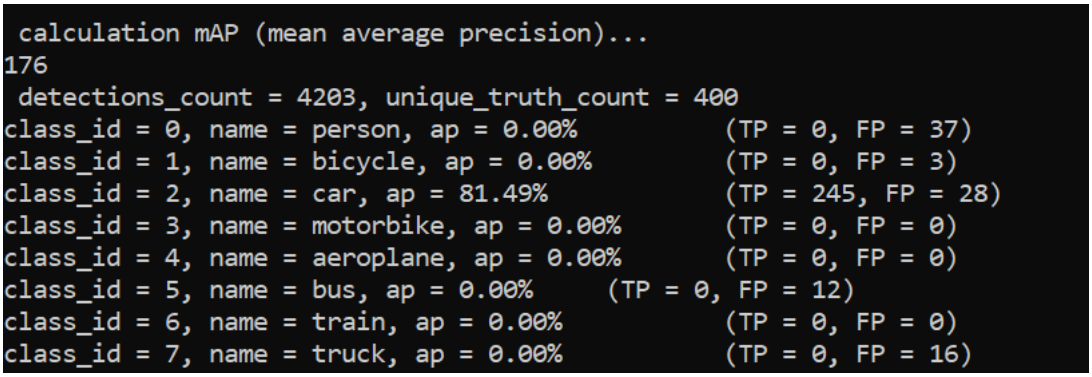
U ovom treningu napravljen je *fine-tune* predtreniranih težina YOLO V3. U .cfg datoteci prije zadnjeg sloja upisana je dodatna linija *stopbackward=1*, koja kaže da je svaki sloj prije toga "zamrznut", točnije ne mijenjaju se vrijednosti težina.

Dataset	Urban Tracker
Početne težine	YOLO V3 početne težine
Batch	64
Subdivisions	64
Width	416
Height	416
Momentum	0.9
Decay	0.01
Learning rate	0.0001
Burn in	50
Max batches	1000
Steps	800, 900
Scales	0.1, 0.1
Random	0
Anchors	10,13,16,30,33,23,30,61 62,45,59,119,116,90,156,198,373,326

Preciznost je povećana sa 69.82 % na 81.49 % na skupu za testiranje skupa Urban Tracker. Težine daju zadovoljavajuće rezultate za rad sustava, iako nisu jednako dobre kao rezultati početnih težina.

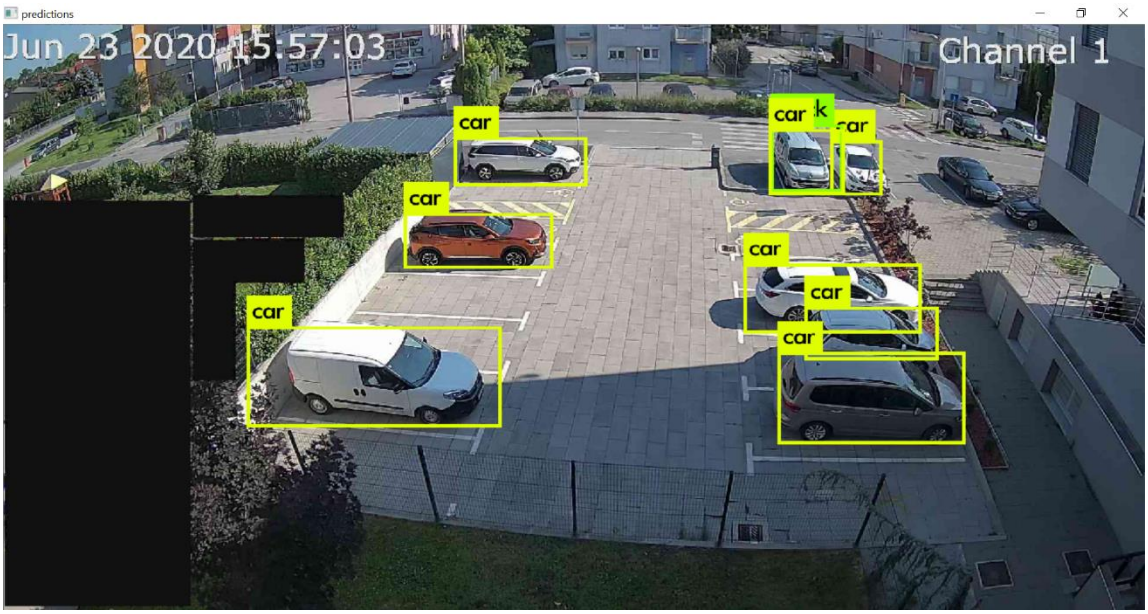


Slika 7.1. Prikaz vrijednosti preciznosti prije *fine-tuneing*-a

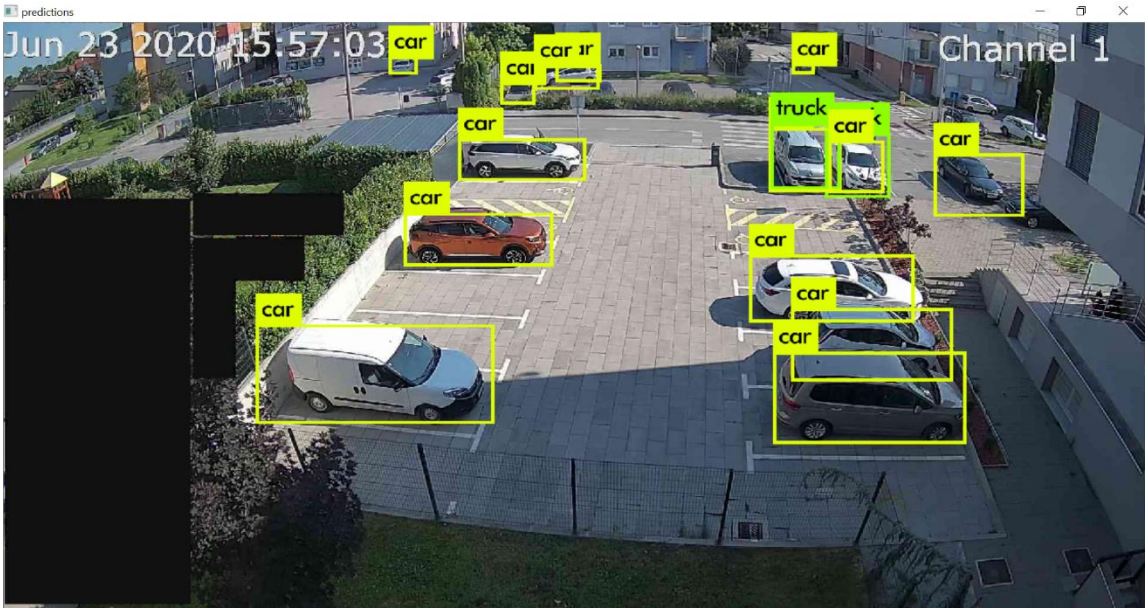


Slika 7.2. Prikaz vrijednosti preciznosti nakon *fine-tuneing*-a

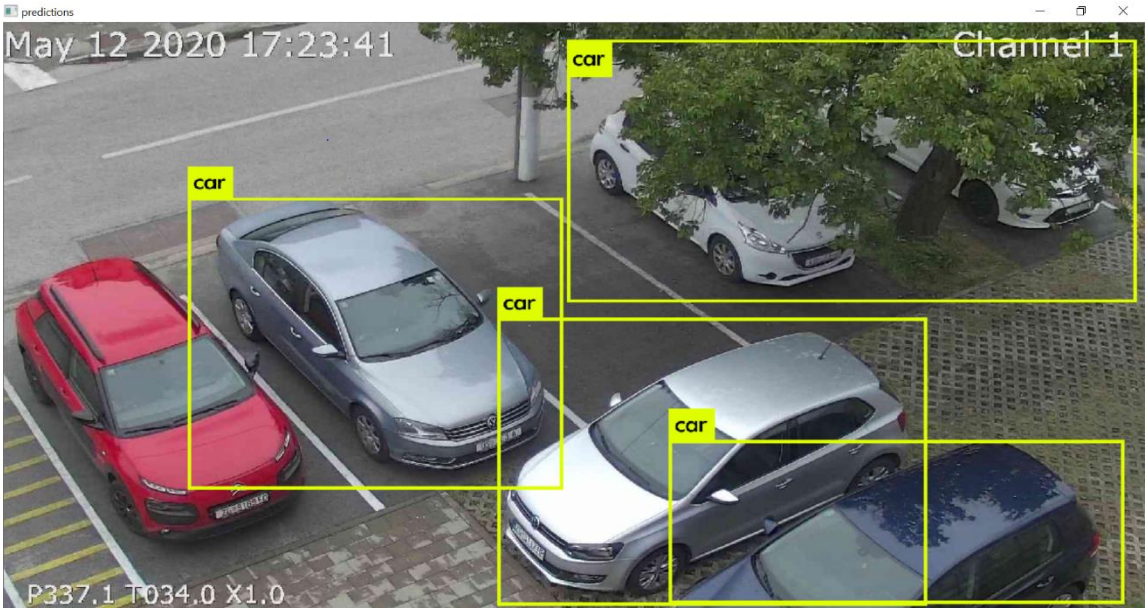
Najbolje rezultate od navedenih treninga postižu težine nakon *fine-tuning*-a kada ih uspoređujemo sa predtreniranim težinama YOLO V3. Tu vidimo da je poprilično teško postići bolje rezultate od njihovog što se tiče preciznosti. Što se tiče brzine težine 14. učenja skoro pa su dvostruko brže od početnih težina YOLO V3. Dok je vrijeme potrebno za obradu slike početnih težina oko 25 milisekunda (40 FPS), težinama 14. treninga treba 14 milisekundi (71.43 FPS) na istim hardverskim uvjetima. Ispod je prikazana usporedba preciznosti dva modela.



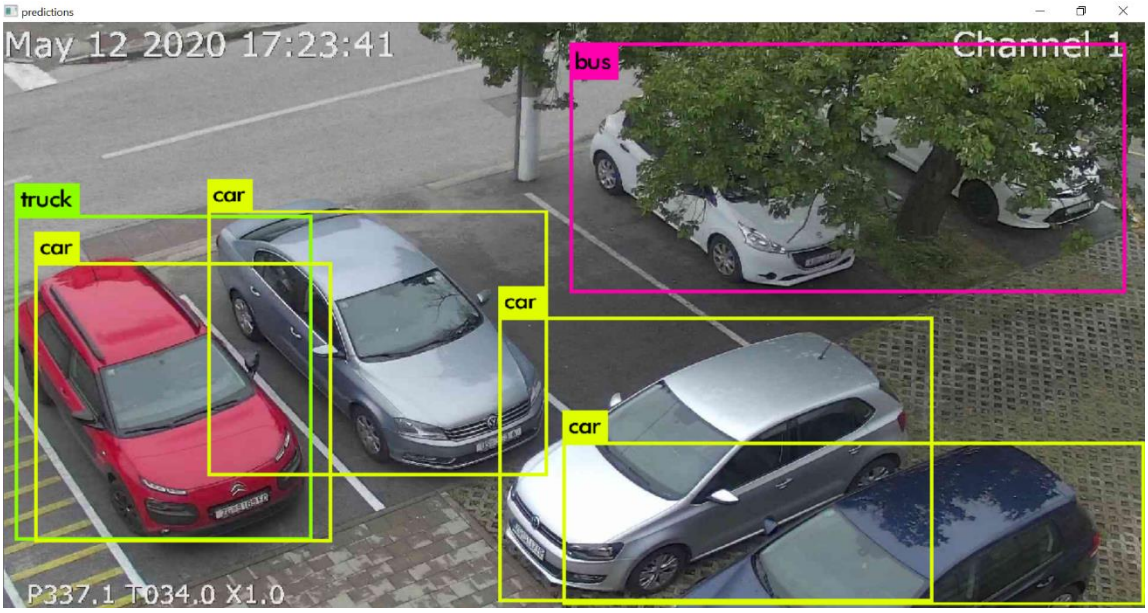
Slika 7.3. Izlaz modela 14. treninga na stvarnom parkingu 1



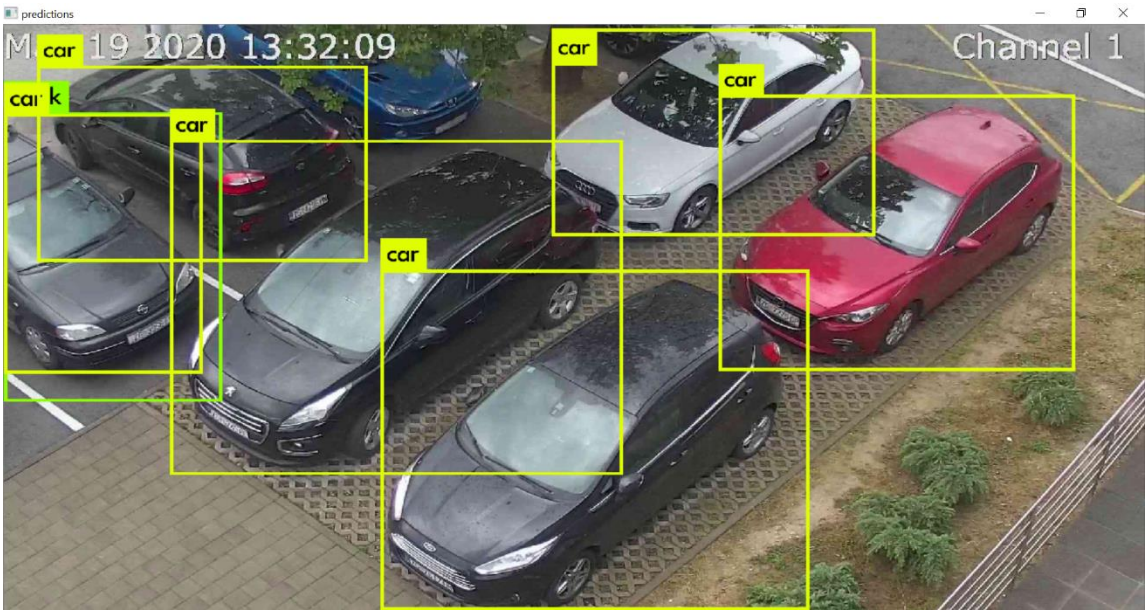
Slika 7.4. Izlaz YOLO V3 na stvarnom parkingu 1



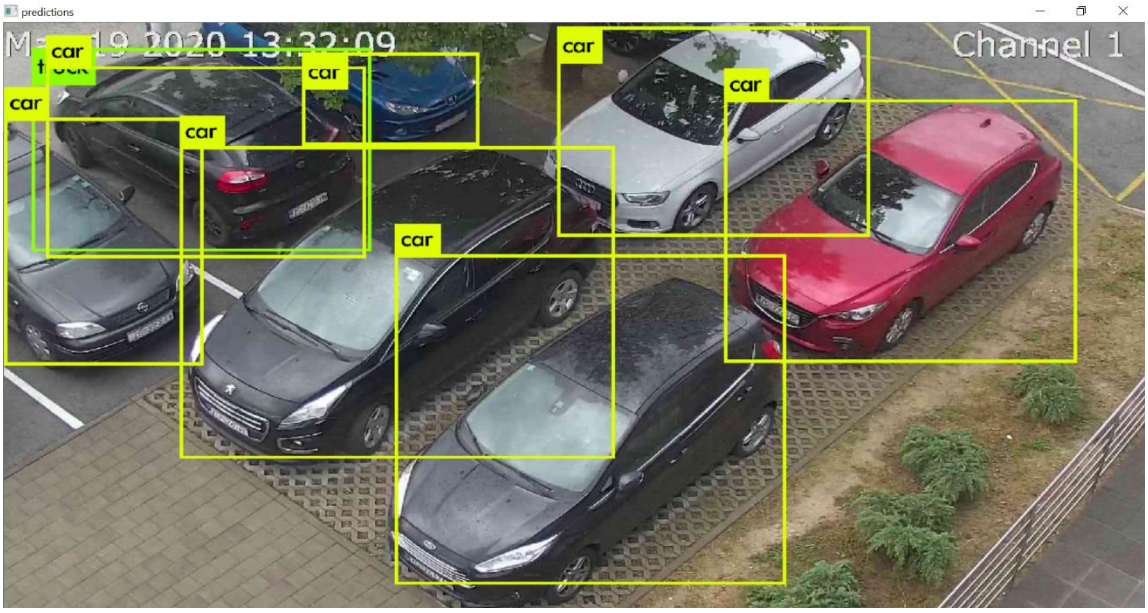
Slika 7.5. Izlaz modela 14. treninga na stvarnom parkingu 2



Slika 7.6. Izlaz YOLO V3 na stvarnom parkingu 2



Slika 7.7. Izlaz modela 14. treninga na stvarnom parkingu 3



Slika 7.8. Izlaz YOLO V3 na stvarnom parkingu 3

7. Zaključak

Cilj ovog rada bio je ostvariti sustav za detekciju zauzeća parkirnih mjesta i naučiti nove neuronske mreže i usporediti ih s postojećim. Pokazano je da iako je moguće ostvariti nove modele neuronskih mreža, teško ih je nadmašiti po pitanju preciznosti detekcije zbog manjka kvalitetnih skupova podataka i skupe procesorske snage.

Moguća rješenja za problem manjka kvalitetnih skupova podataka je ručno označavanje podataka, što vremenski zahtjevno pogotovo s velikim brojem podataka, a velik broj podataka je nužan za precizniji model. Rješenje za ovaj konkretni problem bio bi ručno označavanje svih automobila na stvarnim parkiralištima gdje se provodi detekcija. Postoji opasnost od prenaučenosti, ali kako nije potrebna velika generalizacija modela, jer se koristi uvijek na relativno istim slikama, ne predstavlja prevelik problem.

Još jedno rješenje za ubrzavanje izlaza modela je jača procesorska moć. Njome se postiže češća frekvencija detekcije, samim time i sustav će raditi bolje i brže. Problem je što je dovoljno jaka procesorska snaga za kvalitetnu detekciju financijski skupa.

8. Literatura

1. Jan Šnajder, Bojana Dalbelo Bašić; *Strojno učenje (skripta)*¸
2. Petra Bevandić, Marin Oršić, Ivan Grubišić, Josip Šarić i Siniša Šegvić; Duboko učenje
3. CloudIQ Tech; Introduction To Machine Learning and How It Works
4. Redmon, Joseph and Farhadi, Ali (2018): YOLOv3: An Incremental Improvment
5. Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, i Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org
6. *Schmidhuber, J. (2015). "Deep Learning in Neural Networks: An Overview". Neural Networks*
7. Bradski, G. (2000). The OpenCV Library. Dr. Dobb's Journal of Software Tools.
8. Milan Sonka; Vaclac Hlavac; Roger Boyle (2008). *Image Processing, Analysis and Machine Vision*
9. Jiang; Hadid; Pang; Granger; Feng (2019). – „Deep Learning in Object – Detection and Recognition"
10. Čupić; Mihajlović (2018). – Interaktivna računalna grafika kroz primjere u OpenGL-u

11. Tsung-Yi, Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, Piotr Dollár (2015); Microsoft COCO: Common Objects in Context
12. Jean-Philippe Jodoin ; Guillaume-Alexandre Bilodeau ; Nicolas Saunier (2014); Urban Tracker: Multiple object tracking in urban mixed traffic
13. Jonathan Krause, Michael Stark, Jia Deng, Li Fei-Fe (2013); 3D Object Representations for Fine-Grained Categorization
14. Dawei Du, Yuankai Qi, Hongyang Yu, Yifan Yang, Kaiwen Duan, Guorong Li, Weigang Zhang, Qingming Huang, Qi Tian, " The Unmanned Aerial Vehicle Benchmark: Object Detection and Tracking", European Conference on Computer Vision (ECCV), 2018.
15. Oliphant, T. E. (2006). *A guide to NumPy* (Vol. 1). Trelgol Publishing USA.
16. Everingham, M. and Van~Gool, L. and Williams, C. K. I. and Winn, J. and Zisserman, A.(2012); The {PASCAL} {V}isual {O}bject {C}lasses {C}hallenge 2012 {(VOC2012)} {R}esults
17. Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, Trevor Darrell (2018); BDD100K: A Diverse Driving Dataset for Heterogeneous Multitask Learning
18. Pedregosa, F., Varoquaux, Ga"el, Gramfort, A., Michel, V., Thirion, B., Grisel, O., … others. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12(Oct), 2825–2830.
19. Lawrence, S., Burns, I., Back, A., Tsoi, A. C., Giles, C. L. Neural Network Classification and Prior Class Probabilities. Neural Networks: Tricks of the Trade. Prvo izdanje. Berlin: Springer-Verlag, 1998