**SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU**

**FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK**

Sveučilišni diplomski studij

**DETEKCIJA VOĆA**

**Raspoznavanje uzoraka i strojno učenje**

**Toni Kunštek**

**Osijek, 2022.**

**SADRŽAJ**

[1. UVOD 3](#__RefHeading___Toc108_4270011443)

[2. DUBOKO UČENJE 4](#__RefHeading___Toc110_4270011443)

[2.1. Konvolucijske neuronske mreže](#__RefHeading___Toc116_4270011443) 5

[3. YOLO ALGORITAM](#__RefHeading___Toc112_4270011443) 7

[3.1. Treniranje YOLOv4 mreže](#__RefHeading___Toc116_4270011443) 10

[4. REZULTATI 1](#__RefHeading___Toc118_4270011443)2

[5. ZAKLJUČAK 1](#__RefHeading___Toc120_4270011443)6

[6. LITERATURA 1](#__RefHeading___Toc237_4270011443)7

# UVOD

Projektni zadatak obuhvaća detekciju voća na slikama. Korišten je programski jezik Python u razvojnom okruženju Anaconda. Cilj projektnog zadatka je izgraditi detektor objekata koji će izbaciti bounding box za određene vrste voća te ih klasificirati.

Detekcija objekata je vrlo koristan postupak u mnogim područjima. Omogućava autonomnu vožnju, analizu broja vozila na raskrižjima, sigurnosni sustavi detektiraju ljude, pronalaženje bolesti unutar medicinskih nalaza. Agrikultura nije iznimka. Detekcijom voća moguće je davati informacije o lokaciji plodova, razvrstavati vrste plodova te nezrele plodove od zrelih, pratiti godišnja stanja voća i slično.

A picture containing text, indoor, fruit, colorful

Description automatically generated

*Slika 1.1. Primjer detekcije voća*

# DUBOKO UČENJE

Duboko učenje je grana strojnog učenja posebno prikladna za rješavanje problema s područja umjetne inteligencije, poput razumijevanja govora, obrade fotografije i računalnog vida, obrade prirodnih i govornih jezika i slično. Temelji se na predstavljanju podataka složenim reprezentacijama do kojih se dolazi slijedom naučenih nelinearnih transformacija. Na slici 2.1. prikazana je razlika strojnog i dubokog učenja.

Chart, scatter chart

Description automatically generated

*Slika 2.1. Prikaz razlike strojnog i dubokog učenja*

Pridjev „duboko“ u izrazu „duboko učenje“ označava slojevitost mreža koje rade na taj način, pri čemu ulazni podatci prolaze kroz svaki od slojeva i transformiraju se, nakon svakog sloja u složenije i apstraktnije reprezentacije. Duboko učenje oponaša način na koji ljudi stječu određene vrste znanja. Kod strojnog učenja je izvdajanje značajki bilo potrebno napraviti samostalno te se tek onda događa klasifikacija, no duboko učenje je to sve automatiziralo.

Duboko učenje rješava sve kompliciranije primjene sa sve većom preciznošću tijekom vremena. Ta preciznost rezultata u nekim slučajevima je ili usporediva ili čak nadmašuje rezultate koje mogu postići ljudi.

## Konvolucijske neuronske mreže

Konvolucijske neuronske mreže ili CNN se sastoje od neurona koji se samo-optimiziraju kroz učenje. Svaki neuron prima ulaz i izvodi operaciju. Za razliku od umjetne neuronske mreže, konvolucijske neuronske mreže lakše provode složene operacije potrebne za izračunavanje slikovnih podataka većih dimenzija. Osnova konvolucijskih neuronskih mreža je konvolucijski sloj čijom se primjenom zadržava prostorna struktura ulaznih podataka. Svaki neuron povezan je samo s malim područjem u ulaznoj slici ili u prethodnom sloju. Na slici 2.2. prikazana je skica konvolucijske neuronske mreže.

Diagram

Description automatically generated

Slika 2.2. Konvolucijska neuronska mreža

Primjena filtara na određenom dijelu slike rezultira u skalarnoj vrijednosti. Filtar pokriva malo prostorno područje, ali se prostire preko sva tri kanala ulazne slike. Primjenom istog filtra na različite pozicije dobiva se aktivacijska mapa što je dvodimenzionalna matrica koja sadrži odziv filtra na pojedinom dijelu ulazne slike. Svi neuroni iste aktivacijske mape imaju zajedničke parametre te je moguće primjeniti više filtara. Time se dobivaju zasebne aktivacijske mape koje se slažu kedna pored druge.

Diagram

Description automatically generated

Slika 2.3. Prikaz aktivacijskih mapa

CNN temelji se na sekvenci slojeva između kojih se nalaze aktivacijske funkcije od kojih je najčešća ReLU funkcija. Sekvencijalnom se prijmenom konvolucijskih filtara gubi prostorna veličina kao što vidimo na *slici 2.4.* (32 🡪 28 🡪 24). ReLU (Rectified Linear Unit) je aktivacijska funkcija ispravljača koja se koristi u aktivacijskom sloju CNN-a. ReLU je računalno efikasna i dobro oponaša biološku neuronsku aktivnost.

A picture containing diagram

Description automatically generated

*Slika 2.4. Sekvencijalna primjena konvolucijskog sloja*

Slojevi sažimanja primjenjuju se na svaku aktivacijsku mapu te se tako provodi smanjivanje rezolucije pojedine aktivacijske mape. Na kraju postoji izlazni sloj u kojemu se koristi neuron s linearnom aktivacijskom funkcijom.

# YOLO ALGORITAM

YOLO je konvolucijska neuronska mreža (CNN) za otkrivanje objekata u stvarnom vremenu, akronim za “You Only Look Once”. Kao što ime sugerira, dovoljan je jedan "pogled", jedno širenje naprijed kroz neuronsku mrežu da se pronađu i identificiraju svi objekti na slici. U terminima strojnog učenja, možemo reći da se svi objekti detektiraju putem jednog pokretanja algoritma. Ova tehnika pruža poboljšane rezultate detekcije u usporedbi s drugim tehnikama detekcije objekata kao što su Fast R-CNN i Retina-Net. Prethodni sustavi detekcije prenamjenjuju klasifikatore ili lokalizatore za obavljanje detekcije. Oni primjenjuju model na sliku na više mjesta i mjerila. Područja slike s visokim bodovanjem smatraju se detekcijama. YOLO koristi potpuno drugačiji pristup. Na cijelu sliku primjenjuje se jedna neuronska mreža. Ova mreža dijeli sliku na regije i predviđa granične okvire i vjerojatnosti za svaku regiju. Ovi granični okviri ponderirani su predviđenim vjerojatnostima.

Chart, line chart

Description automatically generated

*Slika 3.1. Usporedba brzina algoritama za detekciju objekata*

Na slici 3.1. prikazana je usporedba brzine obrade slika različitih algoritama za detekciju objekata. Testirani su na COCO datasetu i grafičkoj kartici *NVIDIA Tesla V100*. Ovaj model ima nekoliko prednosti u odnosu na sustave temeljene na klasifikatorima. Gleda cijelu sliku u vrijeme testiranja tako da su njegova predviđanja utemeljena na globalnom kontekstu na slici. Također daje predviđanja s procjenom jedne mreže, za razliku od sustava kao što je R-CNN koji zahtijevaju tisuće za jednu sliku. To ga čini iznimno brzim, više od 1000 puta bržim od *R-CNN*-a i 100 puta bržim od *Fast R-CNN*-a.

Algoritam zahtijeva samo jedno širenje naprijed kroz neuronsku mrežu da bi otkrio objekte.

Prednosti YOLO algoritma:

1. Brzina: Ovaj algoritam poboljšava brzinu detekcije jer može predvidjeti objekte u stvarnom vremenu.
2. Visoka točnost: YOLO je tehnika predviđanja koja daje točne rezultate s minimalnim pozadinskim pogreškama.
3. Mogućnosti učenja: Algoritam ima izvrsne sposobnosti učenja koje mu omogućuju da nauči prikaze objekata i primijeni ih u otkrivanju objekata.

Graphical user interface

Description automatically generated

*Slika 3.2. Skica YOLO algoritma*

YOLO algoritam radi koristeći sljedeće tri tehnike:

1. Preostali blokovi (*eng. residual blocks*)
2. Regresija graničnog okvira (*eng. bounding box regression*)
3. Presjek preko unije (*eng. Intersection Over Union*) (IOU)

Slika je podijeljena na različite mreže. Svaka mreža ima dimenziju S x S. Svaka ćelija mreže će otkriti objekte koji se pojavljuju unutar nje. Na primjer, ako se centar objekta pojavi unutar određene ćelije mreže, ta će ćelija biti odgovorna za njegovo otkrivanje.

Granični okvir je obrub koji ističe objekt na slici. Svaki okvir sadrži sljedeće atribute: klasu, koordinate centra, širinu i visinu.

Svaka ćelija mreže odgovorna je za predviđanje graničnih okvira i njihovih rezultata pouzdanosti. IOU je jednak 1 ako je predviđeni granični okvir isti kao i stvarni okvir. Ovaj mehanizam eliminira granične okvire koji nisu jednaki stvarnom okviru. Na slici 3.3. prikazani su primjeri loših i dobrih IOU. Zeleni okvir predstavlja željeni okvir, a crveni onaj detektirani. Ukoliko je više okvira detektirano, odbacuju se oni s manjom vrijednosti IOU.

Shape, rectangle

Description automatically generated

*Slika 3.3. Dobrota IOU*

**3.1. Treniranje YOLOv4 mreže**

Kako bi istrenirali svoj YOLOv4 neuronsku mrežu, potrebno je preuzeti programsko rješenje s github linka [1]*.* Ovo je službeni repozitorij YOLOv4 koji održava AlexeyAB. Postoji više načina kako pripremiti i kompajlirati darknet program za treniranje. U detalje su objašnjeni unutar README datoteke sa istoga repozitorija. Za windows računalni sustav preporuča se korištenjem *windows powershell* na slijedeći način:

Instalirati *visual studio* i alat *CUDA*

1. Otvoriti *powershell* na željenoj lokaciji
2. Upisati sljedeće naredbe:
   1. *Set-ExecutionPolicy unrestricted -Scope CurrentUser –Force*
   2. *git clone https://github.com/AlexeyAB/darknet*
   3. *cd darknet*
   4. *.\build.ps1 -UseVCPKG -EnableOPENCV –EnableCUDA*

Nakon što je program za treniranje kompajliran, potrebno je pripremiti postavke za treniranje modela unutar nekoliko koraka (uzeto iz iste README datoteke):

1. Preuzeti unaprijed istreniranu *weights* datoteku *yolov4.conv.137* u svrhu transfernog učenja. Transferno učenje je tehnika koja uzima model koji je već osposobljen za povezani zadatak i koristi ga kao početnu točku za stvaranje novog modela.
2. Stvoriti *yolo-obj.cfg* datoteku i prekopirati sadržajdatoteke *yolov4-custom.cfg*
3. Izmijeniti sljedeće unutar datoteke *yolo-obj.cfg*:
   1. *batch=64*
   2. *subdivisions=16*
   3. *max\_batches=2000\**broj\_klasa
   4. *steps=0.8\* max\_batches,0.9\* max\_batches*
   5. *width* i *height* postaviti na višekratnik broja 32, npr. *416*
   6. *classes*, ali samo uz *[yolo]*, na 3 mjesta izjednačiti s željenim brojem klasa
   7. *filters*, ali samo iznad *[yolo]*, na 3 mjesta formulom: *(classes + 5)\*3*
4. Stvoriti *obj.names* s nazivima objekata (svaki naziv u novi red)
5. Stvoriti *obj.data sa sljedećim linijama*
   1. *classes =* broj\_klasa
   2. *train = data/train.txt*
   3. *valid = data/test.txt*
   4. *names = data/obj.names*
   5. *backup = backup/*
6. Postaviti slike koje sadrže objekte u direktorij *build/darknet/x64/data/obj/*
7. Postaviti tekstualnu datoteku za svaku sliku. U ovoj datoteci je za svaki objekt na slici potrebno zapisati id klase, normalizirane koordinate centra i dimenzije kvadrata koji označava objekt na slici, to jest graničnog okvira. Postupak označavanja objekata na slikama znatno olakšava aplikacija *YoloMark* [2]. Primjer izlistanja 10 objekata iste klase sa slike:

|  |  |
| --- | --- |
| 0 0.1116 0.3026 0.0845 0.2271  0 0.5168 0.5084 0.0415 0.0645  0 0.6697 0.4912 0.0386 0.0945  0 0.0779 0.4835 0.1040 0.1289  0 0.0588 0.5813 0.0493 0.0696  0 0.0193 0.6216 0.0386 0.1341  0 0.1721 0.6066 0.0405 0.0938  0 0.1572 0.4187 0.0381 0.0901  0 0.1714 0.3502 0.0498 0.0454  0 0.2073 0.2916 0.0825 0.0718 | *Slika 3.1.1. Primjer slike s objektima* |

1. Stvoriti datoteku *train.txt* koja će sadržavati relativnu putanju slika s *darknet.exe*

Treniranje se započinje sljedećom naredbom pokrenutom unutar *CMD* prozora:

*darknet.exe detector train data/obj.data yolo-obj.cfg yolov4.conv.137*

Ukoliko ne odgovaraju, zadnja 3 argumenta naredbe potrebno je prilagoditi stvarnoj lokaciji datoteka relativno s *darknet.exe* datotekom.

1. **REZULTATI**

Odabrano voće za učenje detekcije pomoću YOLO neuronske mreže je grožđe. Korištena zbirka podataka koja sadrži 300 slika i 300 tekstualnih datoteka preuzetih s repozitorija [3]. Ove tekstualne datoteke sadrže podatke o graničnim okvirima svakoga grozda prikazanog na pojedinoj slici. Za treniranje je navedeno 242 različite slike unutar *train.txt* datoteke.

Odrađene su sljedeće izmjene unutar konfiguracijske datoteke *yolo-obj.cfg* prije početka učenja:

1. *batch*=32
2. *subdivisions*=16
3. *width*=320
4. *height*=320
5. *max\_batches* = 2000
6. *steps*=1600,1800
7. *classes*=1
8. *filters*=18

Zbog određenih nedostataka sklopovlja napravljene su neke izmjene koje nisu preporučene za učenje mreže, kao što je smanjivanje *batch* na 32, *width* i *height* na 320.

A bunch of grapes from a vine

Description automatically generated with low confidence

Slika 4.1. Rezultati detekcije – slika s web stranice [4]

Na slikama 4.1. i 4.2. prikazani su rezultati detekcije istrenirane YOLO mreže. Prva slika je uzeta s web stranice [4], a druga je uslikana u vrtu autora ovoga rada. Prva slika je puno manjih dimenzija. Zbog nedostatka kvalitete se grozdovi u pozadini ne detektiraju. Osim toga, granični okviri nisu postavljeni onako kako bi ih čovjek ucrtao, nisu dovoljno dobro odvojeni grozdovi zbog zbijenosti samih grozdova. Ovo je problem koji nastaje zbog samog izgleda grožđa. Još jedan nedostatak nalazi se u brojevima. Većina grozdova je označena s vjerojatnošću od ispod 0.5 što mrežu čini vrlo nesigurnom.

Druga slika je puno većih dimenzija. Okviri su postavljeni dobro. Vjerojatnosti su nešto veće, ali i dalje puno manje od željenih. Označeno je samo 4 grozda, iako se može primjetiti još nekoliko. Slabija osvjetljenost i prekrivenost lišćem smanjuju vjerojatnosti prepoznavanja grozdova.

Grapes growing on a vine

Description automatically generated with low confidence

Slika 4.2. Rezultati detekcije – slika iz vrta autora

Na slici 4.3. prikazan je graf treniranja. X-os predstavlja iteraciju, a Y-os predstavlja prosječni gubitak *Loss*. Što je manji *Loss*, mreža je bolje naučena.

Graphical user interface, chart, scatter chart

Description automatically generated

Slika 4.3. Graf treniranja

Na grafu se može primjetiti kako nije baš najbolje naučena mreža. Preporučene vrijednosti pri završetku treniranja su unutar raspona od 0.05 za mali model do 3 za veliki model. Za poboljšanje rezultata bi najbolje bilo promijeniti parametre *.cfg* datoteke. Kod zadnjih iteracija na grafu se vidi kako *Loss* prestaje padati, stoga dodatne iteracije učenja ne bi puno pomogle. Naredbom: *darknet.exe detector map data/obj.data yolo-obj.cfg backup\yolo-obj\_2000.weights* mogu se validirati *weights* datoteke i vidjeti koja je najbolja. Prilikom treniranja se svakih 1000 iteracija sačuva po jedna *weights* datoteka. Potrebno je odabrati onu datoteku s najvećim parametrom *mAP* (*mean average precision*)i/ili *IOU*. Nekada kasnije iteracije imaju niže parametre nego prethodne. Ovo je slučaj *overfitting*-a.

U slučaju trenirane mreže za detekciju grožđa, *yolo-obj\_2000.weights* ima parametar *mAP* jednak 81.32%, a prosječni *IoU* iznosi 55.8%. Ponovnim pogledom na slike rezultata 4.1 i 4.2 može se reći da ove vrijednosti odgovaraju naučenoj mreži.

A picture containing text, bunch, different, several

Description automatically generated

*Slika 4.4. Rezultati detekcije – veličina grozda utječe na* uspješnost detekcije

Na slici 4.4. prikazan je primjer detekcije grozdova, izoliranih od okoline, različitih dimenzija. Može se primjetiti kako je veličina grozda bitna za uspješnost detekcije. Najveći grozd je označen s vjerojatnošću od samo 0.35, dok je najmanji označen s visokih 0.92. Na primjeru koji se nalazi u gornjem desnom kutu slike se vidi problem detekcije grožđa koje neke druge vrste voća ne bi imale. Ukupno 3 puta je označen isti grozd. Ovo isto uvelike ovisi o veličini grozda.

1. **ZAKLJUČAK**

Koristeći izvorni kod [1] naučena je YOLO konvolucijska neuronska mreža koja prepoznaje i označava grozdove. Programskim jezikom python je mreža testirana na stvarnim primjerima pomoću biblioteke Keras. Korišten je dataset od 300 slika i 300 tekstualnih datoteka koje sadrže podatke o lokacijama grozdova na slikama. Od tih 300 slika, 242 slike je iskorišteno za treniranje, to jest približno 80%. Ostale su bile predviđene za testiranje.

Na različitim primjerima pokazano je da YOLO mreža uspijeva prepoznavati i označavati grozdove, ali uz znatne nedostatke. Različito osvjetljenje i prekrivenost grozdova smeta, veliki grozdovi nisu dobro označeni te su im vjerojatnosti niske. Kada su grozdovi na slici zbijeni, tada se može dogoditi da su dva grozda označena kao jedan ili je jedan grozd razlomljen u više njih.

Na grafu testiranja bilo je vidljivo da treniranje nije prošlo kako bi bilo poželjno. Parametri mAP i IOU su puno niži nego što bi trebali biti za pravilno ponašanje mreže. Dodatan trening s drugim parametrima bi bio poželjan za svrhu bolje detekcije. Na primjer, širinu *width* i visinu *height* povećati, a *batch* ne bi trebalo smanjiti na 32. Parametar *subdivision* povećati ukoliko je memorija grafičke kartice problem.

1. **LITERATURA**

[1]<https://github.com/AlexeyAB/darknet>

[2] <https://github.com/AlexeyAB/Yolo_mark>

[3] <https://github.com/thsant/wgisd>

[4] <https://grapaes.com/213-egypt-grapes-success/>

[5] <https://pjreddie.com/darknet/yolo/>

[6] <https://neptune.ai/blog/object-detection-with-yolo-hands-on-tutorial>

[7]<https://www.section.io/engineering-education/introduction-to-yolo-algorithm-for-object->detection/