Semantička segmentacija kretanja uporabom konvolucijskih neuronskih mreža

Latinčić, Ante

SVEUČILIŠTE U SPLITU

POMORSKI FAKULTET U SPLITU

ANTE LATINČIĆ

SEMANTIČKA SEGMENTACIJA KRETANJA UPORABOM KONVOLUCIJSKIH NEURONSKIH MREŽA

DIPLOMSKI RAD

SPLIT, 2020

	POMORSKI FAKULTET U SPLITU	STRANICA:	2/1
		ŠIFRA:	F05.1.-DZ
	DIPLOMSKI ZADATAK	DATUM:	22.10.2013..

SPLIT, 25.02.2020.

ZAVOD/STUDIJ: POMORSKE ELEKTROTEHNIČKE I INFORMATIČKE TEHNOLOGIJE

PREDMET: NOVE TEHNOLOGIJE U DIJAGNOSTICI I UPRAVLJANJU

D I P L O M S K I ZADATAK

STUDENT/CA: ANTE LATINČIĆ

MATIČNI BROJ: 0171268178

ZAVOD/STUDIJ: POMORSKE ELEKTROTEHNIČKE I INFORMATIČKE TEHNOLOGIJE

ZADATAK: SEMANTIČKA SEGMENTACIJA SLIKE KONVOLUCIJSKIM NEURONSKIM MREŽAMA

OPIS ZADATKA: OBJASNITI KAKO SE MOGU KORISTITI NEURONSKE MREŽE U OBRADI SLIKA I VIDEO SEKVENCI. OBRADITI POJAM DUBOKOG UČENJA. OBRADITI VRSTE SEMANTIČKE SEGMENTACIJE. U PROGRAMSKOM PAKETU MATLAB IZRADITI PRIMJERE UPOTREBE KONVOLUCIJSKIH NEURONSKIH MREŽA U OBRADI SLIKE.

CILJ: CILJ RADA JE DEMONSTRIRATI KAKO SE KONVOLUCIJSKA NEURONSKA MREŽA MOŽE KORISTITI ZA SEMANTIČKO OZNAČAVANJE PODRUČJA PIKSELA U SLICI.

ZADATAK URUČEN STUDENTU/CI: **25.02.2020.**

POTPIS STUDENTA/CE:

MENTOR: **IZV. PROF. DR. SC. IGOR VUJOVIĆ**

SVEUČILIŠTE U SPLITU

POMORSKI FAKULTET U SPLITU

STUDIJ: POMORSKE ELEKTROTEHNIČKE I INFORMATIČKE TEHNOLOGIJE

SEMANTIČKA SEGMENTACIJA KRETANJA UPORABOM KONVOLUCIJSKIH NEURONSKIH MREŽA

DIPLOMSKI RAD

MENTOR:

izv. prof. dr. sc. Igor Vujović

STUDENT:

Ante Latinčić

(MB:0171268178)

SPLIT, 2020.

SAŽETAK

Cilj ovog rada bio je detaljnije objasniti što je semantička segmentacija i detekcija objekata, te objasniti to na primjeru. Također je čitatelju približen koncept dubokog učenja sa nekim njegovim podvrstama i principima na kojima se zasniva njihov rad. U teorijskom dijelu može se vidjeti kako duboko učenje, semantička segmentacija i detekcija objekata pronalaze jako široko područje primjene u današnjem svijetu. Jako puno sustava koji koriste umjetnu inteligenciju zasnivaju svoj rad na nekoj vrsti dubokog učenja. U praktičnom dijelu pokazana su dva primjera semantičke segmentacije, te jedan primjer detekcije objekta. U prvom primjeru semantičke segmentacije korišten je CamVidDataset koji sadržava slike iz prometa. Drugi primjer koristi ADE20K dataset, odnosno jednu mapu iz tog dataseta koja je korištena za semantičku segmentaciju prirodne okoline. Cilj je bio kreirati neuronsku mrežu koja će razlikovati kopno, more i nebo. Praktični dio za detekciju objekata pokazuje primjer detekcije prometnog znaka STOP.

Ključne riječi: semantička segmentacija, detekcija objekata, duboko učenje

ABSTRACT

The main goal of this thesis was to thoroughly explain concept of semantic segmentation and object detection and show that on example. The concept of deep learning and some of his subgroups are explained as well. Theoretical part of thesis shows that semantic segmentation, object detection and deep learning are widely used nowadays. Big number of AI (Artificial Intelligence) systems are based on some sort of deep learning. Practical part of thesis shows two examples of semantic segmentation and one example of object detection. CamVidDataset was used in the first example of semantic segmentation. CamVidDataset contains images of traffic scenes. Other example uses the ADE20K dataset, that is, one folder from that dataset that was used for semantic segmentation of the natural scenes. The goal was to create a neural network that would distinguish land, sea, and sky. The practical part for object detection shows an example of detection of the sign STOP.

Key words: semantic segmentation, object detection, deep learning

SADRŽAJ

1 UVOD 1
2 DUBOKO UČENJE 2
2.1 Povijest dubokog učenja 3
2.2 Vrste dubokog učenja 4
2.2.1 Višeslojna neuronska mreža 4
2.2.2 Neuronska mreža s povratnom vezom 4
2.2.3 Konvolucijska neuronska mreža 5
2.2.4 LSTM 6
2.2.5 Ponavljajuće neuronske mreže 6
2.2.6 Vremenski konvolucijski strojevi 6
2.2.7 Zbijeni autoenkoderi 6
2.2.8 Ekstremni stroj za učenje 7
2.2.9 Generativno duboko učenje 7
2.3 Primjena dubokog učenja 7
2.3.1 Prepoznavanje slika 7
2.3.2 Prepoznavanje govora 8
2.3.3 Analiza rukopisa 9
2.3.4 Strojno prevođenje 10
2.3.5 Ciljno usmjeravanje 11
3 NEURONSKE MREŽE 12
4 SEMANTIČKA SEGMENTACIJA 14
4.1 Uvod u semantičku segmentaciju 14
4.2 Semantička segmentacija temeljena na prepoznavanju regije 15
4.3 Semantička segmentacija temeljena na potpuno konvolucijskoj neuronskoj mreži 16
4.4 Slabo nadzirana semantička segmentacija 17
5 PRAKTIČNI RAD – SEMANTIČKA SEGMENTACIJA (PRVI PRIMJER) 18
5.1 Postavljanje osnovnih parametara 18
5.2 Preuzimanje CamVid dataseta 18
5.3 Učitavanje CamVid slika 19
5.4 Učitavanje CamVid slika sa označenim pikselima 20
5.5 Analiziranje statistika Dataseta 22
5.6 Priprema obučavanja, validacije i setova slika za testiranje 24
5.7 Kreiranje mreže 24

5.8 Odabir opcija za obuku mreže 26
5.9 Augmentacija podataka 27
5.10 Početak obuke 27
5.11 Testiranje mreže na jednoj slici 27
5.12 Procjena obučene mreže 29
6 PRAKTIČNI RAD – SEMANTIČKA SEGMENTACIJA (DRUGI PRIMJER) 32
6.1 Postavljanje osnovnih parametara 32
6.2 ADE20K Dataset 32
6.3 Image labeler 32
6.4 Učitavanje slika iz ADE20K dataseta 33
6.5 Učitavanje ADE20K slika za označenim pikselima 34
6.6 Analiziranje statistika ADE20K dataseta 35
6.7 Podjela slika na set za obučavanje, validaciju i testiranje 36
6.8 Kreiranje neuronske mreže 36
6.9 Odabir opcija za obuku neuronske mreže 37
6.10 Augmentacija podataka 37
6.11 Početak obuke neuronske mreže 38
6.12 Testiranje neuronske mreže na jednoj slici 39
6.13 Evaluacija mreže 40
7 DETEKCIJA OBJEKTA 42
7.1 Početak rada detekcijom objekata pomoću dubokog učenja 42
7.2 Upoznavanje s osnovnim parametrima 42
8 PRAKTIČNI RAD – DETEKCIJA OBJEKATA 44
8.1 Obučvanje R-CNN detektora stop znaka 44
8.2 Nastaviti obuku R-CNN detektora 46
8.3 Kreiranje R-CNN detektora objekata za dvije klase 46
8.4 Upotrijebiti spremljenu mrežu za R-CNN detektor objekata 47
9 ZAKLjUČAK 50
10 LITERATURA 51
11 POPIS SLIKA 53
12 POPIS KRATICA 54
13 PRILOG 55

1 UVOD

Duboko učenje (engl. deep learning) sve se češće koristi u problemima analize video signala. Analiza video signala jako je bitan segment u današnjem svijetu i ima jako široku primjenu. Gotovo je sigurno da će se u budućnosti cijela automobilska industrija temeljiti na autonomnim vozilima, čiji rad ne bi bio moguć bez detaljne analize video signala. Duboko učenje, njegov koncept, podvrste i primjena opisani su u drugom poglavlju.

U trećem poglavlju objašnjen je koncept i princip rada neurona i neuronskih mreža.

Četvrto poglavlje se bavi problematikom semantičke segmentacije, opisuje njen koncept, podvrste i područja primjene.

Peto poglavlje prikazuje primjer semantičke segmentacije. Korišten je CamVidDataset koji sadrži slike sa prethodno označenim pikselima i klasama, te je korištena unaprijed obučena Resnet-18 neuronska mreža.

U šestom poglavlju prikazan je još jedan primjer semantičke segmentacije. Za razliku od prethodnog primjera, korišten je ADE20K dataset. Slike iz ADE20K dataseta nisu imale unaprijed označene piksele i klase, pa je to napravljeno ručno u Image Labeler-u. Kao i u prethodnom primjeru korištena je Resnet-18 mreža, no nije bila unaprijd obučena. Mreža je obučena na slikama sa ručno označenim pikselima iz ADE20K dataseta.

Sedmo poglavlje se bavi problematikom, konveptom i primjenom detekcije objekata, i služi kao uvod u sedmo poglavlje.

Osmo poglavlje prikazuje primjer detekcije objekata. Kao objekt koji se detektira korišten je prometni STOP znak.

U devetom poglavlju prezentirani su zaključci.

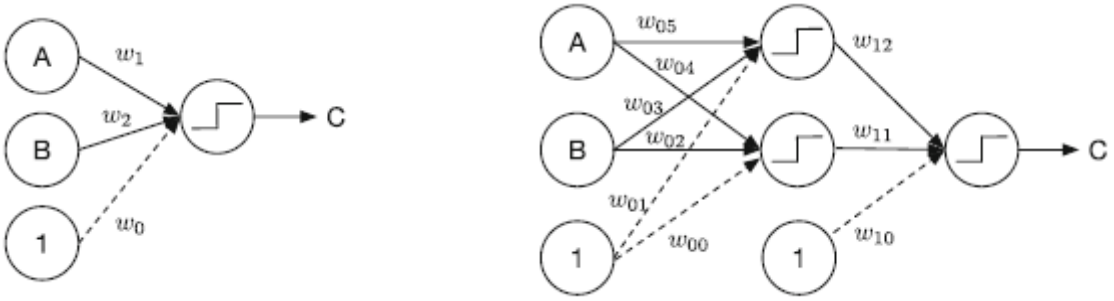
2 DUBOKO UČENJE

Duboko učenje (engl. Deep learning) je grana strojnog učenja, a strojno učenje je grana umjetne inteligencije i statistike. Proučavanje umjetne inteligencije je započelo nedugo nakon drugog svjetskog rata. Rani počeci su bili bazirani na znanju o strukturi mozga, prijedloškoj logici, i na Turingovoj teoriji računanja. Warren McCulloch i Walter Pitts su kreirali matematičku formulu za neuronske mreže zasnovanu na logici praga (engl. treshold logic). To je omogućilo da se istraživanju neuronskih mreža može pristupiti na dva načina: prvi baziran na biološkim procesima mozga i drugi na primjeni neuronskih mreža kod umjetne inteligencije. Demonstrirano je da se svaka funkcija uz pomoć seta neurona i da je neuronska mreža sposobna učiti.

Prvi kompjuterski programi, a i većina programa danas, imaju znanje ugrađeno u kod od strane programera. Programer može koristiti mnogo baza podataka. Na primjer, model letjelice može koristiti multidimenzionalne tablice aerodinamičkih koeficijenata. Stoga rezultirajući softver zna mnogo o letjelicama i simulacije modela mogu dati iznenađujuće rezultate, kako za programere tako i za korisnike. Ipak, programske veze između podataka i algoritama su predeterminirane programskim kodom.

Kod strojnog učenja odnose između podataka formira sustav učenja. Podaci se unose zajedno s rezultatima koji se odnose na podatke. To se naziva obučavanje sustava. Sustav za strojno učenje povezuje podatke s rezultatima i kreira pravila koja postaju dio sustava. Kada su uneseni novi podaci, sustav može kreirati nove rezultate koji nisu bili dio seta za obuku.

Duboko učenje se odnosi na neuronske mreže s više slojeva neurona. Samo ime „duboko učenje“ implicira nešto produbljenije, i u popularnoj literaturi taj izraz se koristi kako bi se impliciralo da je sustav učenja sposoban „misliti“. Na slici 1.1 se mogu vidjeti jednoslojna i višeslojna neuronska mreža.



Slika 1 Jednoslojna (lijevo) i višeslojna (desno) neuronska mreža [6]

Ispostavlja se da su višeslojne neuronske mreže sposobne za učenje stvari koje jednoslojne neuronske mreže ne mogu procesuirati. Elementi mreže su čvorovi u kojima se kombiniraju signali, težine i pristranosti (engl. bias). Kod jednoslojne neuronske mreže, ulazi se množe s težinom, a zatim se zbrajaju na kraju, nakon što prođu kroz graničnu funkciju praga. Kod višeslojne neuronske mreže, odnosno mreže za duboko učenje, ulazi se kombiniraju u drugom sloju prije nego što se dobiju rezultati. Postoji više težina, a dodane veze omogućavaju mreži sposobnost učenja i rješavanja kompleksnijih problema.

Postoje mnoge vrste strojnog učenja. Bilo koji računalni algoritam koji se može prilagoditi na osnovu ulaznih parametara iz okruženja je sustav za učenje. U nastavku su navedeni neki od tipova strojnog učenja:

- neuronske mreže,
- metoda potpornih vektora (engl.Support vector machines),
- adaptivno upravljanje (engl. Adaptive control),
- identifikacija sustava (engl. System identification),
- identifikacija parametara (engl. Parameter identification),
- adaptivni stručni sustavi (engl. Adaptive expert systems),
- upravljački algoritmi (engl. Control algorithms).

2.1 Povijest dubokog učenja

Knjiga „Perceptrons“ koju su napisali Minsky i Seymour Papert 1969. godine je rana analiza umjetnih neuronskih mreža. U knjizi je rečeno kako neuroni ne mogu implementirati neke od logičkih funkcija kao što je ekskluzivno-ili (XOR) i implicirano je da bi i višeslojne neuronske mreže imale isti problem. Kasnije je kroz praksu dokazano da mreže koje imaju tri ili više slojeva mogu implementirati takve funkcije. Višeslojne neuronske mreže otkrivene su u šezdesetim godinama prošlog stoljeća, ali nisu bile temeljno proučavane sve do osamdesetih godina. Kao što je rekao Peter Jackson, ekspertni sustav je računalni program koji raspolaže znanjem o nekom specijaliziranom predmetu s ciljem rješavanja problema ili davanja savjeta. Veliki napredak je ostvaren 80-ih godina kada su AI (engl. artificial intelligence) istraživači počeli s primjenom matematičke i statističke analize u svrhu razvoja algoritama. „Skriveni Markov modeli“ (engl. Hidden Markov models, HMM) su primijenjeni u području prepoznavanja govora. Skriveni Markovljev model (HMM)je model koji sadrži nepromatrana (skrivena) stanja.

U kombinaciji s ogromnim bazama podataka rezultirao je u ogromnom napretku na području prepoznavanja govora.

Ranih 90-ih godina Vladimir Vapnik i njegovi suradnici su kreirali skup nadziranih mreža za učenje s iznimnim mogućnostima koja je danas u svijetu poznata kao metoda potpornih vektora. Ove mreže su sposobne za rješavanje problema kod prepoznavanja uzoraka, regresije i ostalih problema koji se postavljaju pred sustave strojnog učenja.

U posljednjih nekoliko godina došlo je do znatnih napredaka na području dubokog učenja. Novi alati su razvijeni da olakšaju implementaciju dubokog učenja. Neki od najpoznatijih su TensorFlow od Amazon AWS-a. On sadrži izrazito moćne alate za vizualizaciju. TensorFlow omogućava korištenje dubokog učenja na uređajima koji nisu stalno povezani na internet. Također postoji IBM Watson. On omogućava da se koriste TensorFlow, Keras, PyTorch, Caffe i ostalih programskih alata. Keras je popularni alat za duboko učenje koji se može koristiti u Python-u.

2.2 Vrste dubokog učenja

Postoji mnogo vrsta dubokog učenja. Mnogo novih metoda se razvija i bit će u primjeni uskoro. Duboko učenje se primjenjuje znatno više nego ranije i postaje neizostavan faktor kada govorimo o razvoju robotike, umjetne inteligencije i sličnih područja znanosti. U narednom dijelu će biti navedeni i ukratko opisani neki od tipova dubokog učenja.

2.2.1 Višeslojna neuronska mreža

Višeslojne neuronske mreže sadrže:

- ulazne neurone,
- više slojeva skrivenih neurona,
- izlazne neurone.

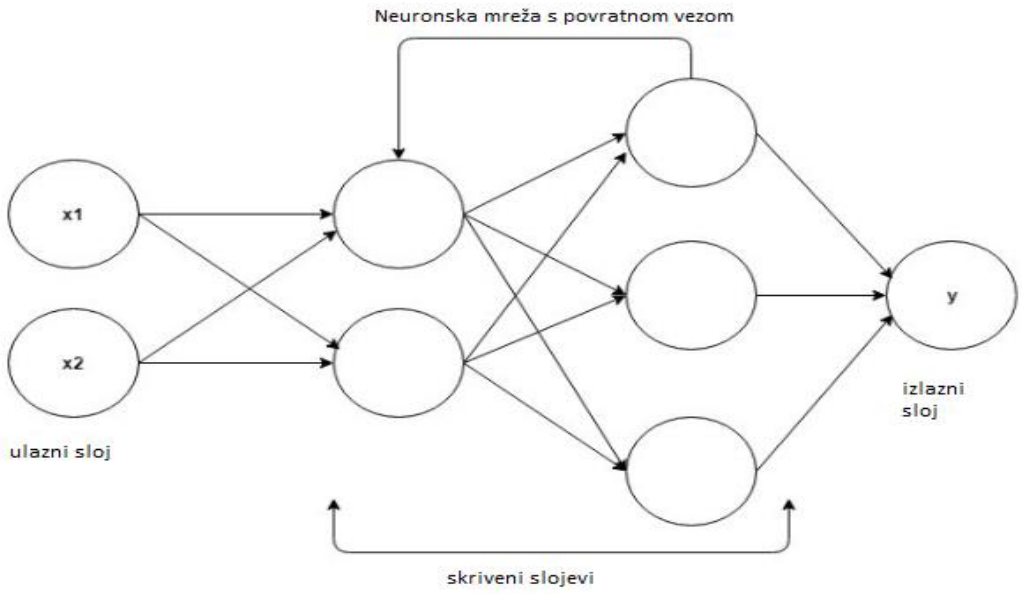
Različiti slojevi imaju različite aktivacijske funkcije. Također mogu biti i funkcijski različiti tako da jedan sloj bude konvolucijski dok je drugi sloj za spajanje.

2.2.2 Neuronska mreža s povratnom vezom

Neuronska mreža s povratnom vezom (Recurrent Neural Network-RNN) je tip ponavljajuće neuronske mreže. Neuronska mreža s povratnom vezom se koristi kod problema koji ovise o vremenu. Kombiniraju podatke koraka od prethodnog vremena s podacima iz skrivenog ili srednjeg sloja, kako bi se dobio prikaz trenutnog vremenskog koraka [6].

Neuronska mreža s povratnom vezom sadrži petlju. Ulazni vektor u nekom trenutku se koristi za kreiranje izlaza koji se zatim prosljeđuje sljedećem elementu mreže.

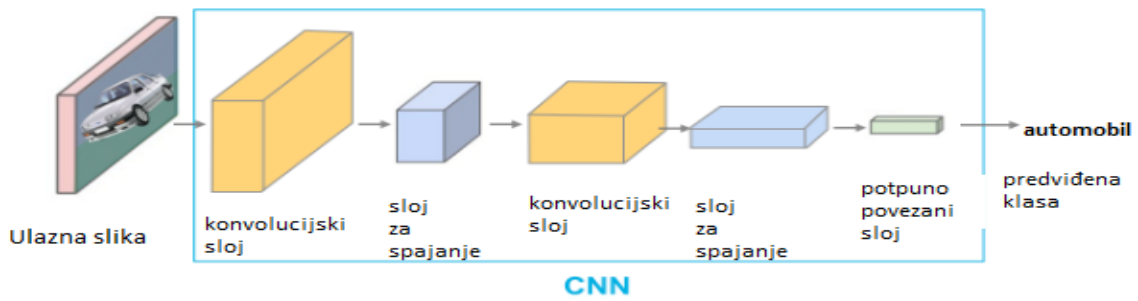
To se provodi rekurzivno s tim da je svako stanje identično s vanjskim ulazima i izlazima iz prethodnog stanja. Ove neuronske mreže se koriste kod prepoznavanja govora, prevođenja jezika i u sličnim područjima.



Slika 2 Neuronska mreža s povratnom vezom [10]

2.2.3 Konvolucijska neuronska mreža

CNN (konvolucijska neuronska mreža) sadrži konvolucijske slojeve (od tud i dolazi ime). Sadrži značajku s ulaznom matricom takvom da izlaz naglašava tu značajku. Jako je efektivna u pronalaženju uzoraka. Na primjer, može se koristiti kod L uzorka kako bi se pronašli svi kutovi. Ljudsko oko sadrži detektore ruba, stoga je i sustav ljudskog vida jedan oblik konvolucijske neuronske mreže.



Slika 3 Konvolucijska neuronska mreža [3]

2.2.4 LSTM

Ovaj tip neuronskih mreža je dizajniran kako bi se izbjegla zavisnost od „starih“ informacija. Standardna neuronska mreža s povratnom vezom (RNN) ima ponavljajuću strukturu. LTSM (Long Short-Term Memory Networks) također ima ponavljajuću strukturu, no za razliku od neuronske mreže s povratnom vezom, svaki element ima četiri sloja. LTSM slojevi odlučuju koju od starih informacija prosljeđuju sljedećem sloju. Ovisno o potrebi, mogu poslati sve stare informacije, ili neće poslati nijednu. Postoji više vrsta ovih neuronskih mreža, ali sve se temelje na mogućnosti zaboravljanja stvari.

2.2.5 Ponavljajuće neuronske mreže

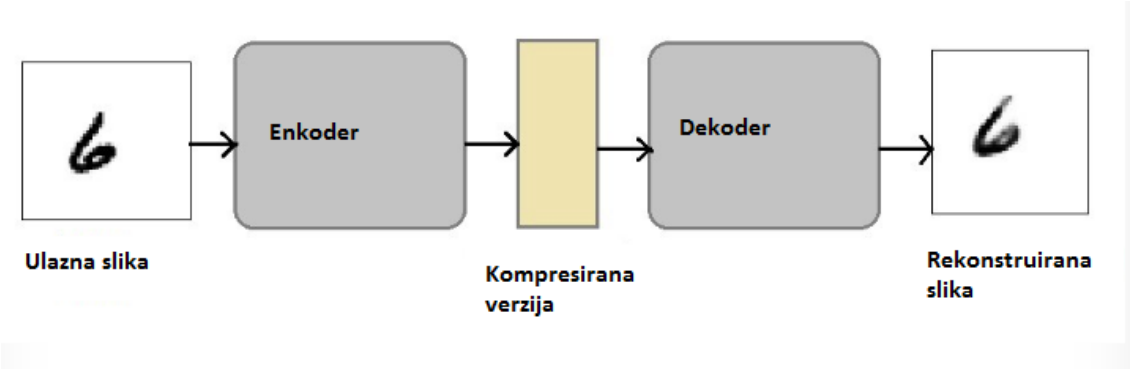
Pojam ponavljajuće neuronske mreže se često miješa s neuronskom mrežom s povratnom vezom, no neuronska mreža s povratnom vezom je podvrsta ponavljajuće neuronske mreže. Ponavljajuće neuronske mreže rade sa strukturiranim podacima. Uspješno se koriste kod obrade jezika (engl. language processing) zbog činjenice da je jezik strukturiran.

2.2.6 Vremenski konvolucijski strojevi

Vremenski konvolucijski stroj (Temporal Convolutional Machines - TCMs) je konvolucijska arhitektura dizajnirana za učenje vremenskih sekvenci. Izrazito se uspješno primjenjuju za statističko modeliranje vremenskih sekvenci. Statističko modeliranje se najčešće koristi kada podaci koji se obrađuju u sebi sadrže šum.

2.2.7 Zbijeni autoenkoderi

Zbijeni autoenkoder (Stacked Autoencoder) je neuronska mreža sastavljena od serija raštrkanih autoenkodera. Autoenkoder je tip neuronske mreže koja je nenadzirani algoritam za učenje koji koristi backpropagation (algoritam koji se koristi na nadzirano učenje).



Slika 4 Zbijeni autoenkoder [11]

Raštrkanost (engl. sparsity) je mjera za količinu aktiviranih neurona, odnosno onih neurona koji imaju takve ulazne parametre koji uzrokuju aktiviranje izlaza za odgovarajuću aktivacijsku funkciju. Izlazi jednog sloja prosljeđuju podatke u idući sloj.

2.2.8 Ekstremni stroj za učenje

Koncept ekstremnog stroja za učenje (Extreme Learning Machine - ELM) je izumio Guang-Bin Huang. ELM je tip feedforward neuronske mreže koja sadrži jedan skriveni sloj. Pod feedforward neuronskom mrežom podrazumijevamo mrežu kod koje su čvorovi povezani tako da ne ostvaruju cikluse. Nasumično odabire težinu skrivenih čvorova i analitički izračunava težinu izlaznih čvorova. ELM osiguravaju dobre performanse i uče izrazito brzo.

2.2.9 Generativno duboko učenje

Generativno duboko učenje omogućava neuronskoj mreži učenje uzoraka i na temelju toga stvaranje potpuno novih podataka. Neuronska mreža koja koristi generativno duboko učenje je u stanju kreirati članke, slike, fotografije i razne druge vrste materijala.

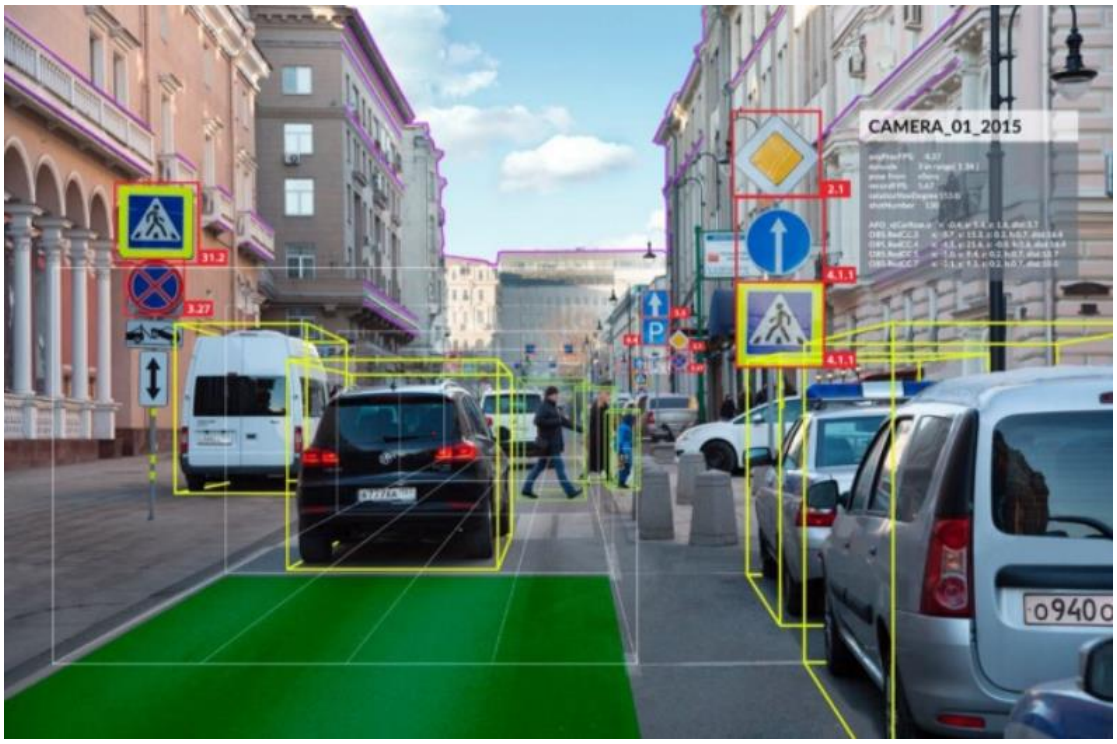
2.3 Primjena dubokog učenja

Danas je duboko učenje temelj mnogih suvremenih grana znanosti. Duboko učenje ima jako široku primjenu i nastavlja se razvijati iz dana u dan. U sljedećem dijelu navedena su neka od značajnijih područja primjene dubokog učenja.

2.3.1 Prepoznavanje slika

S velikom sigurnošću se može reći da je primjena dubokog učenja najraširenija kod prepoznavanja slika (engl. Image recognition). Pojam prepoznavanja slika se često miješa s pojmom kompjuterskog vida (engl. Computer Vision - CV). Glavno pitanje je koja je razlika između CV-a i prepoznavanja slika. CV su razvijale najveće svjetske kompanije koje se bave umjetnom inteligencijom (Google, Amazon i mnogi drugi). CV nastoji imitirati ljudski vid i na temelju tih podataka izvršiti potrebne radnje. Na primjer, jedan način primjene CV-a je u auto industriji. Mnogi suvremeni automobili imaju u sebi sustave sigurnosti koji na temelju CV-a poduzimaju mjere. Ako sustav primijeti dijete na cesti on će poduzeti potrebne mjere da ne dođe do nesreće, odnosno zaustavit će automobil neovisno o reakciji vozača. Također se koristi i kod autonomnih vozila. Sustav analizira okolna područja i na temelju analize podataka procjenjuje najbolju i najsigurniju putanju.

Za razliku od CV-a prepoznavanje slika se temelji na analizi piksela i uzoraka slike kako bi prepoznao u slici dređeni objekt.



Slika 5 Primjer analize slike algoritmom računalnog vida [12]

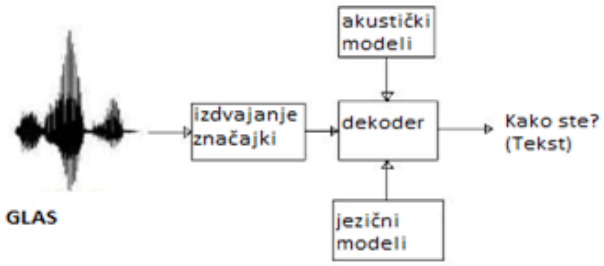
Ljudski vid se uzima zdravo za gotovo. Kada ljudski mozak obradi sliku, bez problema može razlučiti objekte na slici. No kod računala je to mnogo teže, zbog same činjenice da je ljudski mozak dizajniran tako da bude izrazito dobar u prepoznavanju uzoraka i razlučivanja objekata na slici. Jedan od najpoznatijih tipova prepoznavanja slike je OCR (engl. Optical Character Recognition – optičko prepoznavanje znakova). Skener može prepoznati tekst na slici i konvertirati ga u tekstualnu datoteku. Također vrijedi i obratno. Jedan od poznatijih primjera je da skener može prepoznati tekst s tablica automobila i konvertirati taj tekst u sliku.

Također prepoznavanje slike koristi većina pametnih telefona i njima sličnih uređaja za prepoznavanje lica. Sustav prepoznaje ljudsko lice i uspoređuje ga s licima u svojoj bazi podataka. Čak i s različitim uvjetima osvjetljenja, vremenskih uvjeta i ostalih parametara sustav pruža vrlo zadovoljavajuće rezultate.

2.3.2 Prepoznavanje govora

Pod prepoznavanjem govora (engl. Speech recognition) podrazumijeva se sposobnost stroja ili programa za identificiranje riječi i fraza iz govornog jezika u format razumljiv

računalu. Osnovni softveri za prepoznavanja govora sadrže limitiran vokabular riječi i fraza, i vjerojatno će prepoznati samo one riječi koje su izrečene čisto, pod uvjetom da se te riječi nalaze u bazi podataka programa. Napredniji softveri mogu prepoznati duže fraze i imaju mogućnost očitavanja prirodnog govora. Prije nego što softver interpretira govor, mikrofon mora prevesti vibracije ljudskog glasa u električni impuls. Ta pretvorba se vrši preko hardvera sustava, na primjer zvučna kartica računala prevodi govor u digitalni signal. Softver analizira digitalni signal kako bi analizirao i izdvojio foneme, koji su temeljni blokovi govora. Zatim se fonemi rekombiniraju u riječi. No mnoge riječi zvuče slično, tako da se u mnogo slučajeva softver mora osloniti na kontekst kako bi donio odluku. Tada se koriste trigram analize, odnosno metodu grupacije riječi. Obično su to skupine od tri riječi koje se većinom koriste skupa. Ako softver prepozna prve tri riječi te skupine on će sam dodati treću riječ. Te grupacije od tri riječi se nalaze u bazama podataka softvera. Softveri za prepoznavanje izoliranih riječi rade gotovo besprijekorno za sve korisnike, dok softveri koji analiziraju cijele fraze i rečenice nailaze na neke poteškoće. Postotak grešaka je sveden na svega 5% u softverima koji sadržavaju desetke tisuća riječi i fraza. Kada dva ljudska bića razgovaraju, slušač ne samo da snima govor sugovornika nego i predviđa što bi sugovornik mogao reći, te popunjava praznine i neispravnu gramatiku. Na sličan način rade i softveri za prepoznavanje govora. Na primjer, danas su softveri toliko uznapredovali da je nekada teško razlučiti da li je s druge strane telefonske linije stvarna osoba ili automatska sekretarica.



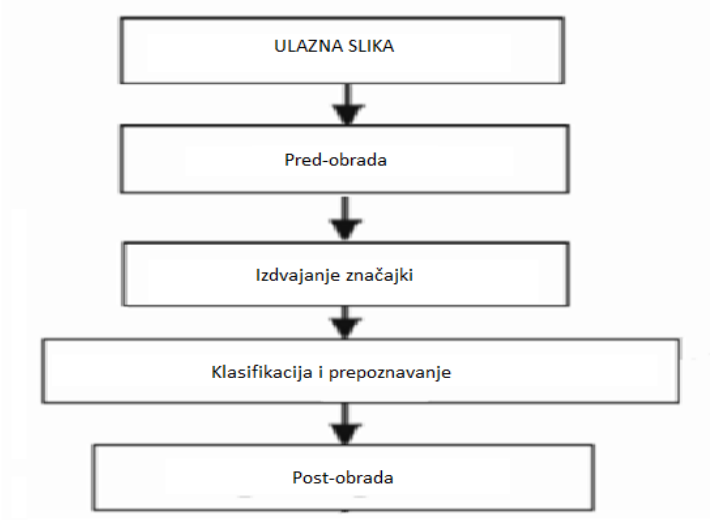
Slika 6 Sustav prepoznavanja govora [20]

2.3.3 Analiza rukopisa

Na temelju tehnike akvizicije, metode za prepoznavanje rukopisa se mogu svrstati u dvije kategorije: Online akvizicija i offline akvizicija.

Online metoda zahtjeva posebne uređaje za unos, kao što su elektrostatički ili elektrodinamički tableti i olovke. Offline metoda se znatno razlikuje od online metode.

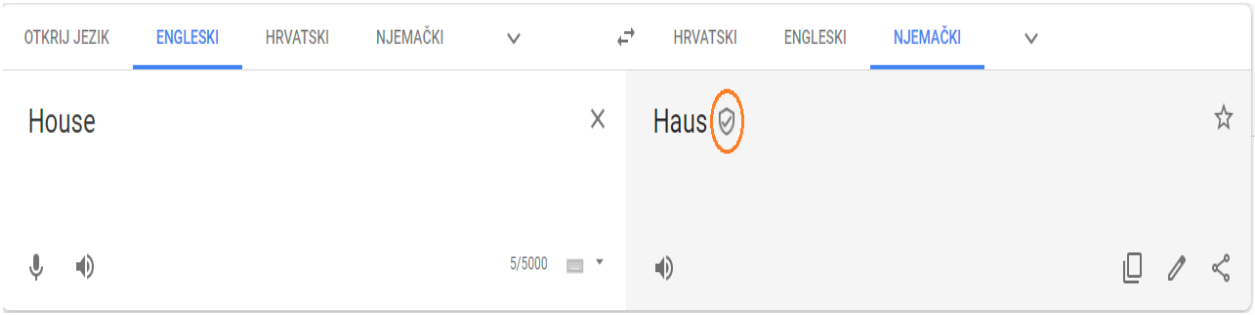
U ovom slučaju skener ili kamera visoke rezolucije skeniraju tekst koji je zapisan na papiru. Danas je gotovo svaki pametni telefon sposoban za neku razinu analize, odnosno prepoznavanja rukopisa. Kod starijih verzija bilo je potrebno unositi jedno po jedno slovo, no kod većine novijih modela je moguće očitavanje riječi i rečenica. Jedan od pionira na ovom području je američka pošta. Prije mnogo godina razvili su sustav za očitavanje poštanskog koda. Prvotno je kod morao biti strogo smješten na za to predviđenom mjestu, no s razvojem tehnologije omogućeno je očitavanje koda bez obzira na kojem dijelu omotnice se kod nalazi.



Slika 7 Analiza rukopisa [1]

2.3.4 Strojno prevođenje

Najpoznatiji softver za strojno prevođenje je Google prevoditelj. On odlično obavlja prevođenje gotovo svih svjetskih jezika. Google prevoditelj je primjer sustava s online obukom. Jedan od indikatora toga je kada se uz prijevod pojavi kvačica. Ta kvačica označava da je ljudsko biće označilo taj prijevod kao točan.



Slika 8 Google prevoditelj

2.3.5 Ciljno usmjeravanje

Pod pojmom ciljno usmjeravanje (engl. targeting) se misli na prepoznavanje uzoraka preferencija korisnika. U današnje vrijeme gotovo je nemoguće ne biti podložan targetingu. Na primjer, kada na svom računalu ili pametnom telefonu pretražujete neke artikle u online trgovinama, gotovo sigurno će te vidjeti reklame sa sličnim ili istim artiklima.

Slika 9 Ciljno usmjeravanje [6]

3 NEURONSKE MREŽE

Neuronske mreže su najpopularniji način implementiranja „strojne inteligencije“. Osnovna ideja je da se neuronske mreže ponašaju kao neuroni u ljudskom mozgu. Za početak pogledajmo jedan neuron s dva ulaza, koji je prikazan na slici 10. Ovaj neuron ima ulaze x_1 i x_2, bias b (dodatni parametar u neuronskoj mreži koji se koristi za podešavanje izlaza zajedno s težinskim zbrojem ulaza u neuron. Prema tome, bias je konstanta koja pomaže modelu tako da se najbolje uklopi s obzirom na date podatke.), težine w_1 i w_2, te izlaz z. Aktivacijska funkcija σ uzima ulaze i njihove težine i na temelju toga producira izlaz.

$$z = \sigma(y) = \sigma(x_1 w_1 + x_2 w_2 + b) \tag{1}$$

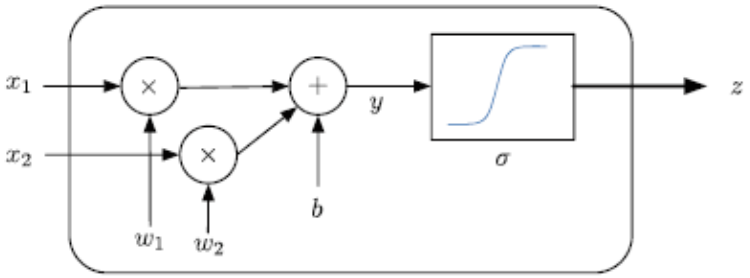
Usporedimo ovo sa stvarnim neuronom koji je prikazan na slici 11. Stvarni neuron ima više ulaza preko dendrita, koji su protoplazmatski produžeci neurona. Neki od dendrita se granaju, što znači da se više ulaza može povezati na tijelo stanice preko jednog dendrita. Izlaz se ostvaruje preko aksona. Akson je dio neuronske stanice preko kojega se provode impulsi od tijela stanice prema drugim stanicama. Svaki neuron ima jedan izlaz. Signali se od aksona do dendrita prenose preko sinapsi. Zapanjujući podatak je da stvarni neuron može imati i do 10000 ulaza. Postoje brojne standardne aktivacijske funkcije. Ovdje su prikazane tri:

$$\sigma(y) = \tanh(y) \tag{2}$$

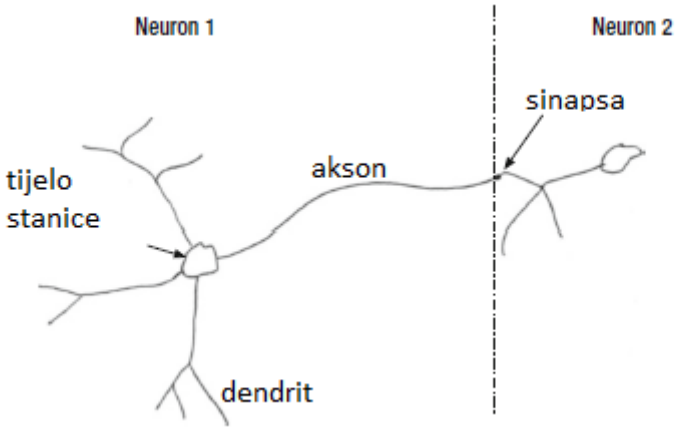
$$\sigma(y) = \frac{2}{1-e^{-y}} - 1 \tag{3}$$

$$\sigma(y) = y \tag{4}$$

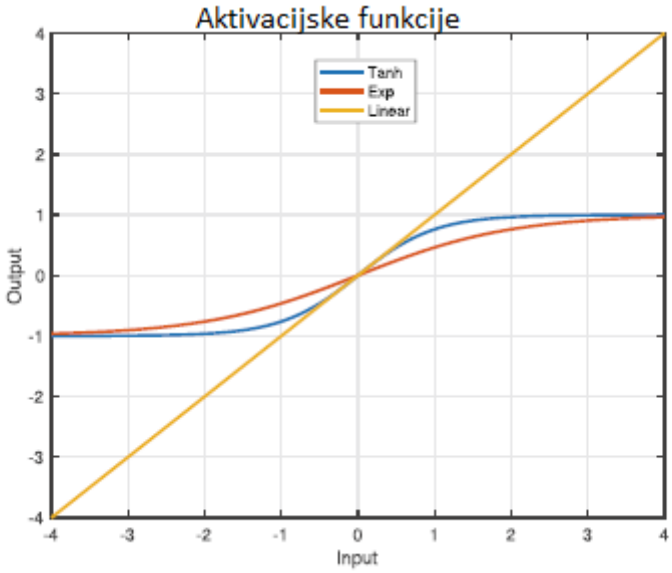
Eksponencijalna funkcija je normalizirana i pomaknuta s nule, pa se kreće od -1 do 1. Aktivacijska funkcija koja jednostavno prolazi kroz vrijednost y, se naziva linearna aktivacijska funkcija.



Slika 10 Neuron sa dva ulaza [6]



Slika 11 Veza između dva neurona [6]



Slika 12 Tri aktivacijske funkcije [6]

Aktivacijske funkcije koje zasićuju ili dosežu ulaznu vrijednost nakon koje je izlaz konstantan ili se mijenja vrlo sporo, modeliraju biološki neuron koji ima maksimalnu brzinu okidanja (engl. firing rate). Te određene funkcije imaju i dobra numerička svojstva koja su od pomoći pri učenju.

4 SEMANTIČKA SEGMENTACIJA

4.1 Uvod u semantičku segmentaciju

Danas je semantička segmentacija jedan od ključnih problema na području računalnog vida. Gledajući široku sliku, semantička je segmentacija jedan od zadataka na visokoj razini koji utire put prema cjelovitom razumijevanju scene. Važnost razumijevanja scene kao temeljnog problema s računalnim vidom ističe činjenica da sve veći broj aplikacija potiče iz prikupljanja znanja iz slika. Neke od tih aplikacija uključuju autonomna vozila, interakciju čovjeka i računala, virtualnu stvarnost itd. S popularnošću dubokog učenja posljednjih godina, mnogi problemi semantičke segmentacije se rješavaju pomoću dubokih arhitektura, najčešće uporabom konvolucijskih neuronskih mreža, koje uvelike nadmašuju druge pristupe u pogledu točnosti i učinkovitosti. Semantička segmentacija prirodan je korak u prelasku iz grubog u fino zaključivanje: izvorište se može nalaziti u klasifikaciji, koja se sastoji od predviđanja za cijeli unos. Sljedeći korak je lokalizacija / detekcija, koje pružaju ne samo klase već i dodatne informacije u vezi s prostornim smještajem tih klasa. Na kraju, semantička segmentacija postiže fino zrnato zaključivanje (engl. fine-grained inference) stvaranjem gustih predviđanja za pridruživanje oznaka za svaki piksel, tako da je svaki piksel označen s klasom svog priloženog objekta ili regije.

Slika 13 Semantička segmentacija [7]

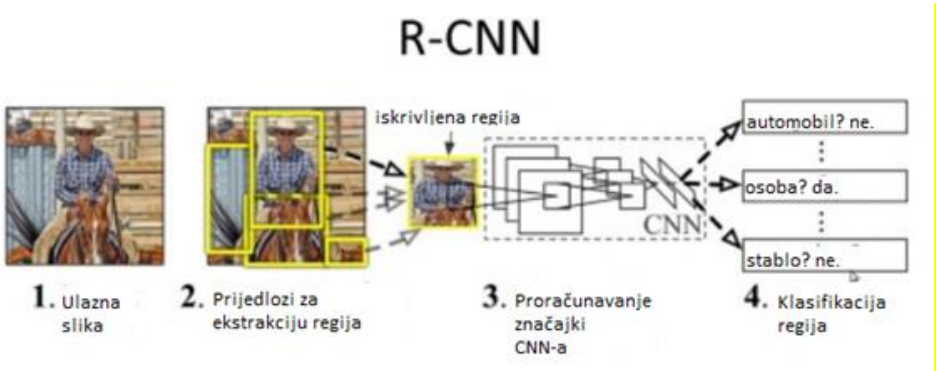
Općenita arhitektura semantičke segmentacije može se pojednostavljeno prikazati kao mreža kodera praćena mrežom dekodera:

- Koder je obično prethodno obučena klasifikacijska mreža kao što je VGG / ResNet, iza koje slijedi dekoderska mreža.
- Zadatak dekodera je semantički projicirati diskriminacijske značajke (niža razlučivost) koje je koder naučio na prostor piksela (viša razlučivost) kako bi se dobila gusta klasifikacija.

Za razliku od klasifikacije gdje je jedino važan krajnji rezultat duboke mreže, semantička segmentacija ne zahtijeva samo diskriminaciju na razini piksela, već i mehanizam za projiciranje diskriminirajućih karakteristika naučenih u različitim fazama kodera na prostor piksela. Različiti pristupi koriste različite mehanizme kao dio mehanizma dekodiranja.

4.2 Semantička segmentacija temeljena na prepoznavanju regije

Metode utemeljene na prepoznavanju regija općenito slijede princip "segmentacije pomoću prepoznavanja", koji prvo izvlači regije slobodnog oblika sa slike i opisuje ih, nakon čega slijedi klasifikacija na temelju regije. Za vrijeme ispitivanja, predviđanja temeljena na regiji transformiraju se u predviđanja piksela, obično označavanjem piksela prema regiji s najvećom sličnošću koja ga sadrži.



Slika 14 Semantička segmentacija temeljena na prepoznavanju regije [2]

R-CNN (regije sa značajkama CNN-a) je reprezentativni primjer metoda temeljenih na prepoznavanju regije. Izvodi semantičku segmentaciju na temelju rezultata detekcije objekata. Odnosno, R-CNN prvo koristi selektivno pretraživanje za izdvajanje velike količine prijedloga objekata, a zatim izračunava CNN značajke za svakog od njih. Konačno, klasificira svaka regija pomoću linearnih SVM-ova specifičnih za klasu.

U usporedbi s tradicionalnim CNN strukturama koje su uglavnom namijenjene klasifikaciji slika, R-CNN može rješavati složenije zadatke, poput detekcije objekata i segmentacije slika. Štoviše, R-CNN se može graditi na bilo kojoj referentnoj strukturi CNN-a, kao što su AlexNet, VGG, GoogLeNet i ResNet.

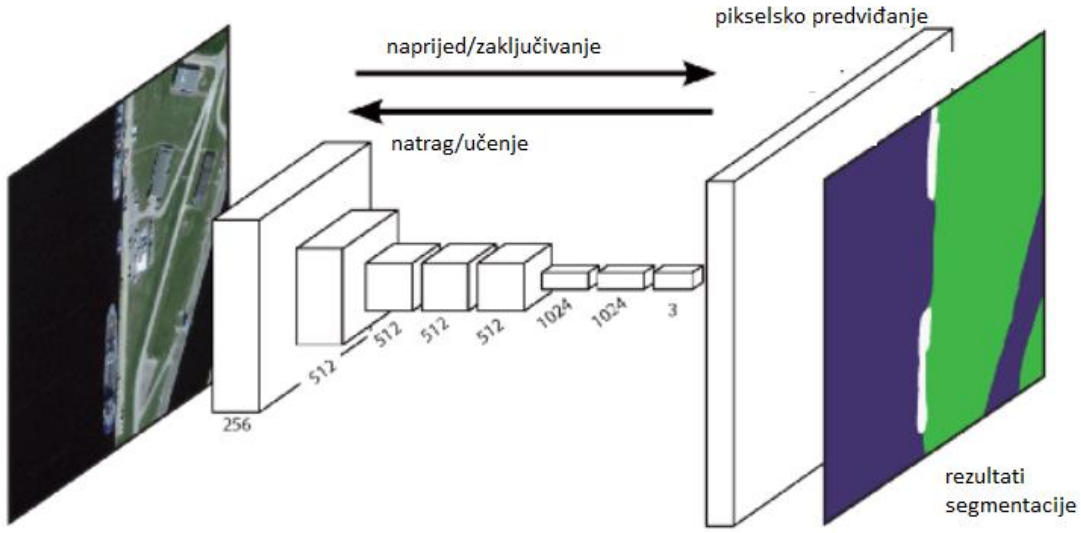
Za zadatak segmentacije slike, R-CNN izdvaja dvije vrste značajki za svaku regiju: značajku pune regije (engl. full region feature) i značajku prvog plana (engl. foreground feature) s ciljem da bi to moglo dovesti do boljih performansi prilikom njihovog spajanja kao značajke regija (engl. region feature). R-CNN postiže značajna poboljšanja performansi zahvaljujući korištenju visoko diskriminativnih značajki CNN-a.

4.3 Semantička segmentacija temeljena na potpuno konvolucijskoj neuronskoj mreži

Izvorna potpuno konvolucijska mreža (engl. Fully Convolutional Network - FCN) uči mapiranje s piksela na piksel, bez izvlačenja prijedloga regije. FCN mrežni sustav produžetak je klasičnog CNN-a. Glavna ideja je učiniti da klasični CNN uzima kao ulaz slike proizvoljne veličine.

Ograničenje CNN-a da prihvaćaju i generiraju oznake (engl. labels) samo za ulaze određene veličine dolazi iz potpuno povezanih slojeva koji su fiksni. Suprotno njima, FCN-ovi imaju samo konvolucijske slojeve i slojeve za spajanje (engl. pooling layers) koji im daju mogućnost predviđanja na ulazima proizvoljne veličine.

Jedan od problema kod FCN-a je da se prolaženjem kroz više naizmjeničnih konvolucijskih slojeva i slojeva za spajanje rezolucija izlaznih značajki smanjuje. Stoga su izravna predviđanja FCN-a generalno niske rezolucije što rezultira relativno nejasnim granicama objekta. Za rješavanje ovog problema predložen je niz naprednijih pristupa temeljenih na FCN-u, uključujući SegNet, DeepLab-CRF i proširene konvolucije.



Slika 15 Struktura FCN-a [5]

4.4 Slabo nadzirana semantička segmentacija

Većina relevantnih metoda u semantičkoj segmentaciji oslanja se na velik broj slika s pikselnim maskama za segmentaciju. Međutim, ručno bilježenje ovih maski prilično je dugotrajno, frustrirajuće i komercijalno skupo. Stoga su nedavno predložene neke slabo nadzirane metode (engl. Weakly supervised methods), koje su posvećene ispunjavanju semantičke segmentacije korištenjem označenih ograničavajućih okvira. Na primjer, Boxsup je upotrijebio oznake ograničavajućih okvira kao nadzor kako bi obučio mrežu i iterativno poboljšao procijenjene maske za semantičku segmentaciju (slika 16).

Slika 16 Slabo nadzirana semantička segmentacija [5]

5 PRAKTIČNI RAD – SEMANTIČKA SEGMENTACIJA (PRVI PRIMJER)

5.1 Postavljanje osnovnih parametara

Ovaj primjer kreira Deeplab v3 + mrežu s utezima inicijaliziranim iz unaprijed obučene mreže Resnet-18. ResNet-18 učinkovita je mreža koja je vrlo pogodna za aplikacije s ograničenim resursima za obradu. Da bi dobili prethodno obučenu Resnet-18 mrežu, potrebno je instalirati Deep Learning Toolbox ™ Model for Resnet-18 Network. Nakon završetka instalacije pokrenite sljedeći kod da biste provjerili je li instalacija ispravna.

```
resnet18();
```

Kao dodatak, potrebno je preuzeti unaprijed obučenu verziju DeepLab v3+. Unaprijed obučeni model omogućuje vam pokretanje cijelog primjera bez čekanja da se obuka završi.

```
pretrainedURL = 'https://www.mathworks.com/supportfiles/vision/data/deeplabv3plusResnet18CamVid.mat';
pretrainedFolder = fullfile(tempdir,'pretrainedNetwork');
pretrainedNetwork = fullfile(pretrainedFolder,'deeplabv3plusResnet18CamVid.mat');
if ~exist(pretrainedNetwork,'file')
    mkdir(pretrainedFolder);
    disp('Downloading pretrained network (58 MB)...');
    websave(pretrainedNetwork,pretrainedURL);
end
```

5.2 Preuzimanje CamVid dataseta

Potrebno je preuzeti CamVid dataset sa sljedećih linkova:

```
imageURL = 'http://web4.cs.ucl.ac.uk/staff/g.brostow/MotionSegRecData/files/701_StillsRaw_full.zip';
labelURL = 'http://web4.cs.ucl.ac.uk/staff/g.brostow/MotionSegRecData/data/LabeledApproved_full.zip';

outputFolder = fullfile(tempdir,'CamVid');
labelsZip = fullfile(outputFolder,'labels.zip');
imagesZip = fullfile(outputFolder,'images.zip');

if ~exist(labelsZip, 'file') || ~exist(imagesZip,'file')
    mkdir(outputFolder)

    disp('Downloading 16 MB CamVid dataset labels...');
    websave(labelsZip, labelURL);
    unzip(labelsZip, fullfile(outputFolder,'labels'));

    disp('Downloading 557 MB CamVid dataset images...');
    websave(imagesZip, imageURL);
    unzip(imagesZip, fullfile(outputFolder,'images'));
end
```

Komande u prethodnom dijelu koda zaustavljaju rad MATLAB-a dok se preuzimanje ne završi. Također je moguće prethodno preuzeti dataset i pohraniti ga na disk. Kako bi se onda dataset upotrijebio potrebno je usmjeriti funkciju *outputfolder* na lokaciju na disku gdje je dataset pohranjen.

5.3 Učitavanje CamVid slika

Za učitavanje se koristi funkcija *imageDatastore*. Ta funkcija omogućava učinkovito učitavanje velikog broja slika na disk.

```
imgDir = fullfile(outputFolder,'images','701_StillsRaw_full');
imds = imageDatastore(imgDir);
```

Prikaz jedne od slika iz dataseta.

```
I = readimage(imds,1);
I = histeq(I);
imshow(I)
```

Slika 17 Prikaz slike iz dataseta

5.4 Učitavanje CamVid slika sa označenim pikselima

Potrebno je koristiti *pixelLabelDatastore* za učitavanje podataka o označenim pikselima sa slika. *PixelLabelDatastore* povezuje podatke o oznakama piksela i ID oznake sa mapiranjem imena klase. Kako bi se obuka olakšala, 32 originalne klase su grupirane u svega 11 klasa.

```
classes = [
    "Sky"
    "Building"
    "Pole"
    "Road"
    "Pavement"
    "Tree"
    "SignSymbol"
    "Fence"
    "Car"
    "Pedestrian"
    "Bicyclist"
    ];
```

U svrhu reduciranja klasa, više klasa iz originalnog dataseta su grupirane zajedno. Na primjer, klasa „Car“ je kombinacija klasa „Car“, „SUVPickupTruck“, „Truck_Bus“, „Train“, i „OtherMoving“. Grupirani ID oznaka piksela se vraćaju uz pomoć funkcije *camvidPixelLabelIDs.*

```
function labelIDs = camvidPixelLabelIDs()
% Vratite ID-ove naljepnica koji odgovaraju svakoj klasi
% CamVid dataset ima 32 klase. Grupirati ih u 11 klasa slijedeci
originalne
% SegNet metodologije
%
% 11 klasa su:
%   Sky" "Building", "Pole", "Road", "Pavement", "Tree", "SignSymbol",
%   "Fence", "Car", "Pedestrian",  i "Bicyclist".
%
 % "Sky"
 labelIDs = { ...
    [
    128 128 128; ... % "Sky"
    ]

    % "Building"
    [
    000 128 064; ... % "Bridge"
    128 000 000; ... % "Building"
    064 192 000; ... % "Wall"
    064 000 064; ... % "Tunnel"
    192 000 128; ... % "Archway"
    ]
```

```
    % "Pole"
    [
    192 192 128; ... % "Column_Pole"
    000 000 064; ... % "TrafficCone"
    ]

    % Road
    [
    128 064 128; ... % "Road"
    128 000 192; ... % "LaneMkgsDriv"
    192 000 064; ... % "LaneMkgsNonDriv"
    ]

    % "Pavement"
    [
    000 000 192; ... % "Sidewalk"
    064 192 128; ... % "ParkingBlock"
    128 128 192; ... % "RoadShoulder"
    ]

    % "Tree"
    [
    128 128 000; ... % "Tree"
    192 192 000; ... % "VegetationMisc"
    ]

    % "SignSymbol"
    [
    192 128 128; ... % "SignSymbol"
    128 128 064; ... % "Misc_Text"
    000 064 064; ... % "TrafficLight"
    ]

    % "Fence"
    [
    064 064 128; ... % "Fence"
    ]
    % "Car"
    [
    064 000 128; ... % "Car"
    064 128 192; ... % "SUVPickupTruck"
    192 128 192; ... % "Truck_Bus"
    192 064 128; ... % "Train"
    128 064 064; ... % "OtherMoving"
    ]
    % "Pedestrian"
    [
    064 064 000; ... % "Pedestrian"
    192 128 064; ... % "Child"
    064 000 192; ... % "CartLuggagePram"
    064 128 064; ... % "Animal"
    ]
    % "Bicyclist"
    000 128 192; ... % "Bicyclist"
    192 000 192; ... % "MotorcycleScooter"
    ]
    };
end
```

Klase i ID oznaka se koriste kako bi se kreirao *pixelLabelDatastore.*

```
labelDir = fullfile(outputFolder,'labels');
pxds = pixelLabelDatastore(labelDir,classes,labelIDs);
```

Učitavanje i prikaz jedne od slika sa označenim pikselima i preklapanje te slike preko obične slike.

```
C = readimage(pxds,1);
cmap = camvidColorMap;
B = labeloverlay(I,C,'ColorMap',cmap);
imshow(B)
pixelLabelColorbar(cmap,classes);
```

Slika 18 Preklapanje slike sa označenim pikselima i obične slike

Regije koje nemaju preklapanje boja nemaju oznake piksela i ne koriste se za vrijeme obučavanja.

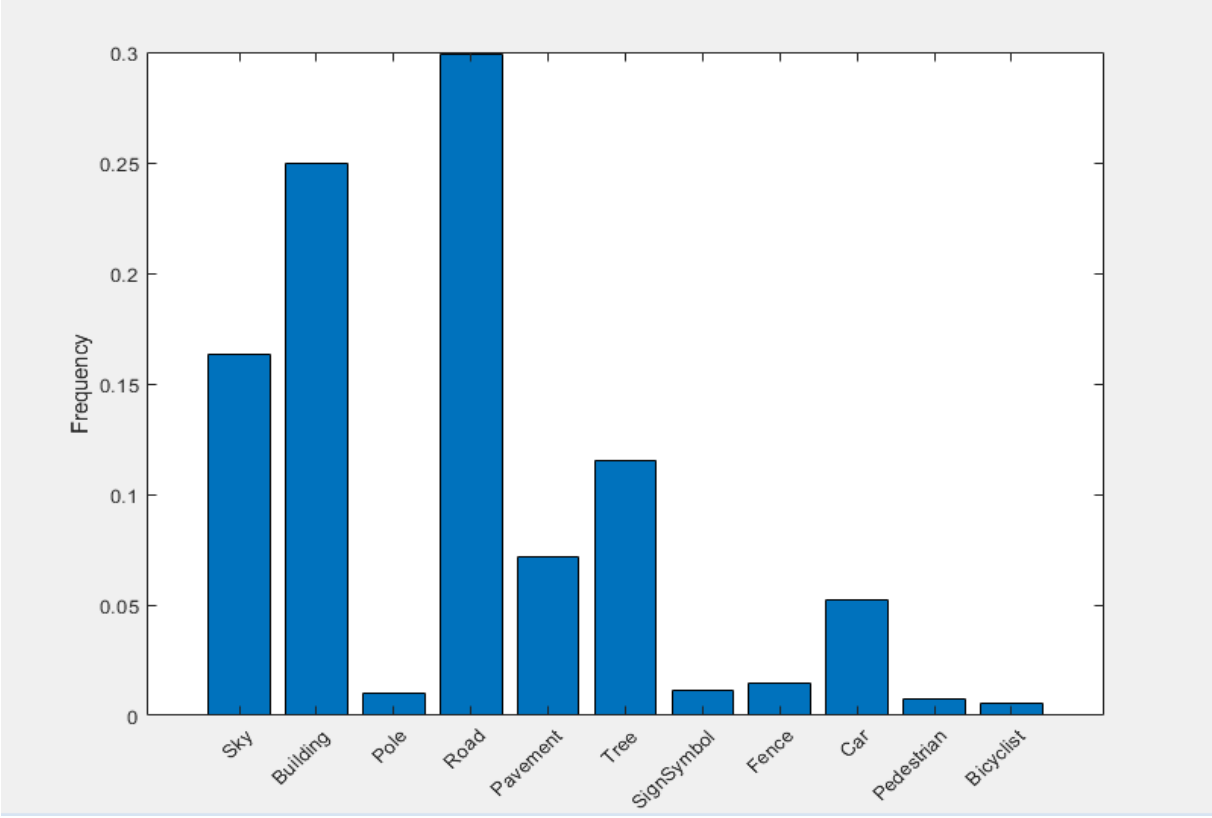
5.5 Analiziranje statistika Dataseta

Da bi se vidjela distribucija oznaka klasa u CamVid datasetu, upotrebljava se funkcija *countEachLabel*. Ova funkcija broji količinu piksela prema oznaci klase.

```
tbl = countEachLabel(pxds)
```

```
tbl =

  11×3 table

        Name          PixelCount     ImagePixelCount
    ________________    __________    _______________

    {'Sky'       }     7.6801e+07      4.8315e+08
    {'Building'  }     1.1737e+08      4.8315e+08
    {'Pole'      }     4.7987e+06      4.8315e+08
    {'Road'      }     1.4054e+08      4.8453e+08
    {'Pavement'  }     3.3614e+07      4.7209e+08
    {'Tree'      }     5.4259e+07       4.479e+08
    {'SignSymbol'}     5.2242e+06      4.6863e+08
    {'Fence'     }     6.9211e+06       2.516e+08
    {'Car'       }     2.4437e+07      4.8315e+08
    {'Pedestrian'}     3.4029e+06      4.4444e+08
    {'Bicyclist' }     2.5912e+06      2.6196e+08
```

Na sljedećem grafu vizualiziran je broj piksela po klasi.



Slika 19 Broj piksela po klasi

Idealno bi bilo da sve klase imaju jednak broj opažanja. Međutim, klase u CamVid datasetu su neuravnotežene, što je čest problem u automobilskim datasetovima za promatranje uličnih scena. Takve scene imaju više piksela za nebo, zgradu i cestu od piksela za pješake i bicikliste jer nebo, zgrade i ceste pokrivaju više regija na slici. Ako se s njom ne postupa pravilno, ova neravnoteža može štetiti procesu učenja jer je učenje pristrano u korist dominantnih klasa. Kasnije u ovom primjeru koristit će se težinsko izjednačavanje klasa kako bi se riješio ovaj problem.

5.6 Priprema obučavanja, validacije i setova slika za testiranje

Deeplab v3 + obučava se se koristeći 60% slika iz dataseta. Ostatak slika podijeljen je ravnomjerno na 20% i 20% za provjeru validacije i za testiranje. Sljedeći kod nasumično dijeli podatke o slici i oznakama piksela u set za obuku, provjeru valjanosti i test.

```
[imdsTrain, imdsVal, imdsTest, pxdsTrain, pxdsVal, pxdsTest] = partitionCamVidData(imds,pxds);
```

Podjela 60/20/20 rezultira sljedećim brojem slika obučavanja, provjere valjanosti i testiranja:

```
numTrainingImages = numel(imdsTrain.Files)
```

```
numTrainingImages = 421
```

```
numValImages = numel(imdsVal.Files)
```

```
numValImages = 140
```

```
numTestingImages = numel(imdsTest.Files)
```

```
numTestingImages = 140
```

5.7 Kreiranje mreže

Upotrijebite funkciju *deeplabv3plusLayers* za stvaranje mreže DeepLab v3 + koja se temelji na ResNet-18. Odabir najbolje mreže za vašu aplikaciju zahtijeva empirijsku analizu i detaljno proučavanje parametara.

Odredite veličinu mrežne slike. To je obično ista veličina kao i veličina slika za obuku:

```
imageSize = [720 960 3];
```

Specifikacija broja klasa:

```
numClasses = numel(classes);
```

Kreirati Deeplab v3 + :

```
lgraph = deeplabv3plusLayers(imageSize, numClasses, "resnet18");
```

Kao što je pokazano ranije, klase u CamVid datasetu nisu uravnotežene. Kako bi se poboljšala obuka, koristi se izjednačavanje težine klasa (engl. class weighting) za bolju uravnoteženost klasa. Koristi se broj oznaka piksela proračunat ranije s *countEachLabel* funkcijom i računa se srednja frekvencija težina klasa.

```
imageFreq = tbl.PixelCount ./ tbl.ImagePixelCount;
classWeights = median(imageFreq) ./ imageFreq
```

```
classWeights =

    0.3182
    0.2082
    5.0924
    0.1744
    0.7103
    0.4175
    4.5371
    1.8386
    1.0000
    6.6059
    5.1133
```

Navođenje težine klasa pomoću *pixelClassificationLayer*.

```
pxLayer = pixelClassificationLayer('Name','labels','Classes',tbl.Name,'ClassWeights',classWeights);
lgraph = replaceLayer(lgraph,"classification",pxLayer);
```

5.8 Odabir opcija za obuku mreže

```
% Definiranje podataka za validaciju
pximdsVal = pixelLabelImageDatastore(imdsVal,pxdsVal);

% Definiranje opcija za obuku
options = trainingOptions('sgdm', ...
    'LearnRateSchedule','piecewise',...
    'LearnRateDropPeriod',10,...
    'LearnRateDropFactor',0.3,...
    'Momentum',0.9, ...
    'InitialLearnRate',1e-3, ...
    'L2Regularization',0.005, ...
    'ValidationData',pximdsVal,...
    'MaxEpochs',30, ...
    'MiniBatchSize',1, ...
    'Shuffle','every-epoch', ...
    'CheckpointPath', tempdir, ...
    'VerboseFrequency',2,...
    'Plots','training-progress',...
    'ValidationPatience', 4);
```

Stopa učenja je raspoređena po dijelovima. Stopa učenja smanjuje se za faktor 0,3 na svakih 10 epoha. To omogućuje mreži da brzo uči s većom početnom stopom učenja, dok istovremeno može pronaći gotovo optimalno rješenje čim stopa učenja padne.

Mreža se testira prema podacima validacije svaku epohu postavljanjem parametra *'ValidationData'*. *'ValidationPatience'* postavljen je na 4 da zaustavi obuku rano kad se preciznost validacije približi. To sprječava mrežu da se prekomjerno prilagodi datasetu za obučavanje.

Mini-batch veličine 1 koristi se za smanjenje upotrebe memorije tijekom obuke. Ovu vrijednost možete povećati ili smanjiti na temelju količine GPU memorije koju imate na vašem sustavu.

Uz to, *'CheckpointPath'* postavljen je na privremenu lokaciju. Time se omogućava spremanje mrežnih kontrolnih točaka (engl. checkpoint) na kraju svake epohe obuke. Ako je obuka prekinuta zbog kvara sustava ili nestanka struje, može se nastaviti sa spremljene kontrolne točke. Potrebno je provjeriti ima li lokacija određena funkcijom *"CheckpointPath*" dovoljno prostora za pohranu mrežnih kontrolnih točaka. Na primjer, za pohranu 100 kontrolnih točaka Deeplab v3 + potrebno je ~ 6 GB prostora na disku jer je svaka kontrolna točka 61 MB.

5.9 Augmentacija podataka

Augmentacija (uvećanje) podataka koristi se tijekom obuke kako bi se pružilo više primjera mreži, jer to pomaže u poboljšanju točnosti mreže. Ovdje se za augmentaciju podataka koristi slučajni lijevi / desni odraz i slučajna X / Y translacija od +/- 10 piksela. Upotrijebite *imageDataAugmenter* da odredite ove parametre za augmentaciju podataka.

```
augmenter = imageDataAugmenter('RandXReflection',true,...
    'RandXTranslation',[-10 10],'RandYTranslation',[-10 10]);
```

5.10 Početak obuke

Kombinirajte podatke o obuci i odabiru augmentacije podataka pomoću *pixelLabelImageDatastore*. *PixelLabelImageDatastore* čita serije podataka o obuci, primjenjuje augmentaciju podataka i šalje augmentirane podatke algoritmu obuke.

```
pximds = pixelLabelImageDatastore(imdsTrain,pxdsTrain, ...
    'DataAugmentation',augmenter);
```

Započnite obučavati pomoću *trainNetwork* ako je *doTraining* flag istinita. U suprotnom, učitajte prethodno obučenu mrežu.

```
doTraining = false;
if doTraining
    [net, info] = trainNetwork(pximds,lgraph,options);
else
    data = load(pretrainedNetwork);
    net = data.net;
end
```

5.11 Testiranje mreže na jednoj slici

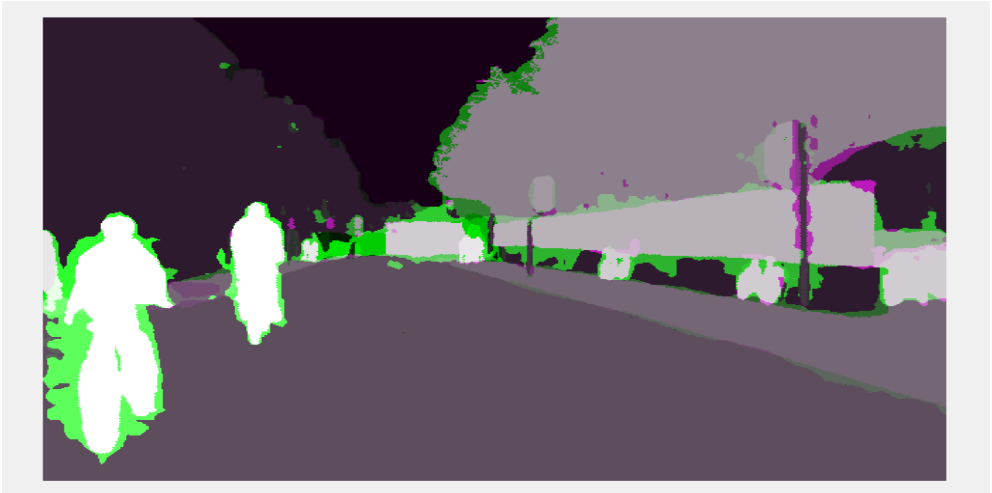
Kako bi se uvjerili da mreža radi ispravno, moguće ju je testirati na jednoj slici.

```
%Pokreni obučenu mrežu na jednoj slici
I = readimage(imdsTest,35);
C = semanticseg(I, net);
%Prikaz rezultata
B = labeloverlay(I,C,'Colormap',cmap,'Transparency',0.4);
imshow(B)
pixelLabelColorbar(cmap, classes);
```

Slika 20 Testiranje mreže na jednoj slici

Usporedba rezultata u C s očekivanim rezultatima pohranjenim u *pxdsTest* (slika 21). Zelena i magenta regija ističu područja u kojima se rezultati segmentacije razlikuju od očekivanih rezultata

```
expectedResult = readimage(pxdsTest,35);
actual = uint8(C);
expected = uint8(expectedResult);
imshowpair(actual, expected)
```



Slika 21 Usporedba dobivenih i očekivanih rezultata

Vizualno se rezultati semantičke segmentacije dobro preklapaju za klase kao što su cesta, nebo i zgrada. Međutim, manji objekti poput pješaka i automobila nisu toliko precizni. Količina preklapanja po klasi može se izmjeriti pomoću mjerenja intersection-over-union (IoU), također poznatog kao Jaccardov indeks. Upotrijebiti *jaccard* funkciju za mjerenje IoU-a.

```
iou = jaccard(C,expectedResult);
table(classes,iou)
```

```
ans =

  11×2 table

      classes          iou
    _____________    _______

    "Sky"            0.91837
    "Building"       0.84479
    "Pole"           0.31203
    "Road"           0.93698
    "Pavement"       0.82838
    "Tree"           0.89636
    "SignSymbol"     0.57644
    "Fence"          0.71046
    "Car"            0.66688
    "Pedestrian"     0.48417
    "Bicyclist"      0.68431
```

5.12 Procjena obučene mreže

Da bi se izmjerila točnost više testnih slika, potrebno je pokrenuti *semanticseg* na cijelom datasetu. *Mini-batch* veličine 4 koristi se za smanjenje upotrebe memorije tijekom segmentiranja slika. Ova se vrijednost može povećati ili smanjiti na temelju količine GPU memorije koja je dostupna sustavu.

```
pxdsResults = semanticseg(imdsTest,net, ...
    'MiniBatchSize',4, ...
    'WriteLocation',tempdir, ...
    'Verbose',false);
```

Semanticseg vraća rezultate za dataset kao objekt *pixelLabelDatastore*-a. Stvarni podaci oznake piksela za svaku testnu sliku u *imdsTest* zapisuju se na disk na mjestu navedenom parametrom '*WriteLocation*'. Upotrijebite *evaluateSemanticSegmentation* za mjerenje podataka semantičke segmentacije na rezultatima dataseta

```
metrics = evaluateSemanticSegmentation(pxdsResults,pxdsTest,'Verbose',false);
```

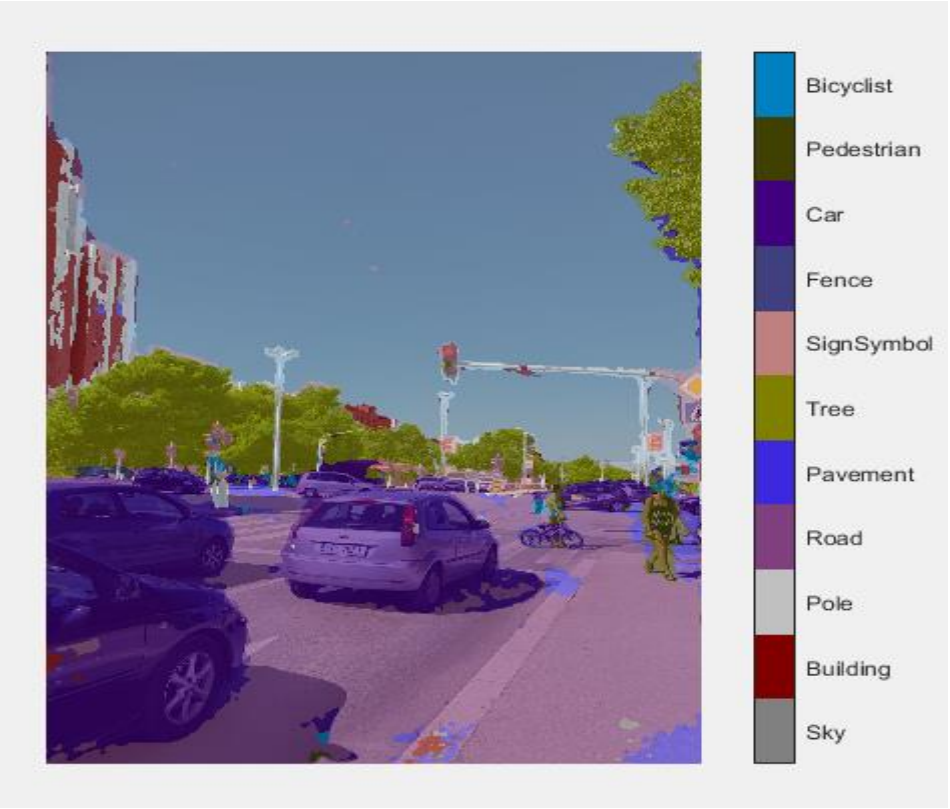
```
ans =

  1×5 table

    GlobalAccuracy    MeanAccuracy    MeanIoU    WeightedIoU    MeanBFScore
    ______________    ____________    _______    ___________    ___________

        0.87695           0.85392       0.6302       0.80851        0.65051
```

Metrics.ClassMetrics daje uvid u preciznost mreže za svaku klasu. Još jednom je vidljivo kako su najbolji rezultati za one klase koje zauzimaju najveći dio na slici, odnosno oni dijelovi koji sadrže najviše piksela, dok je preciznost manja za ostale klase, kao što su pješaci, prometni znakovi, motociklisti itd.

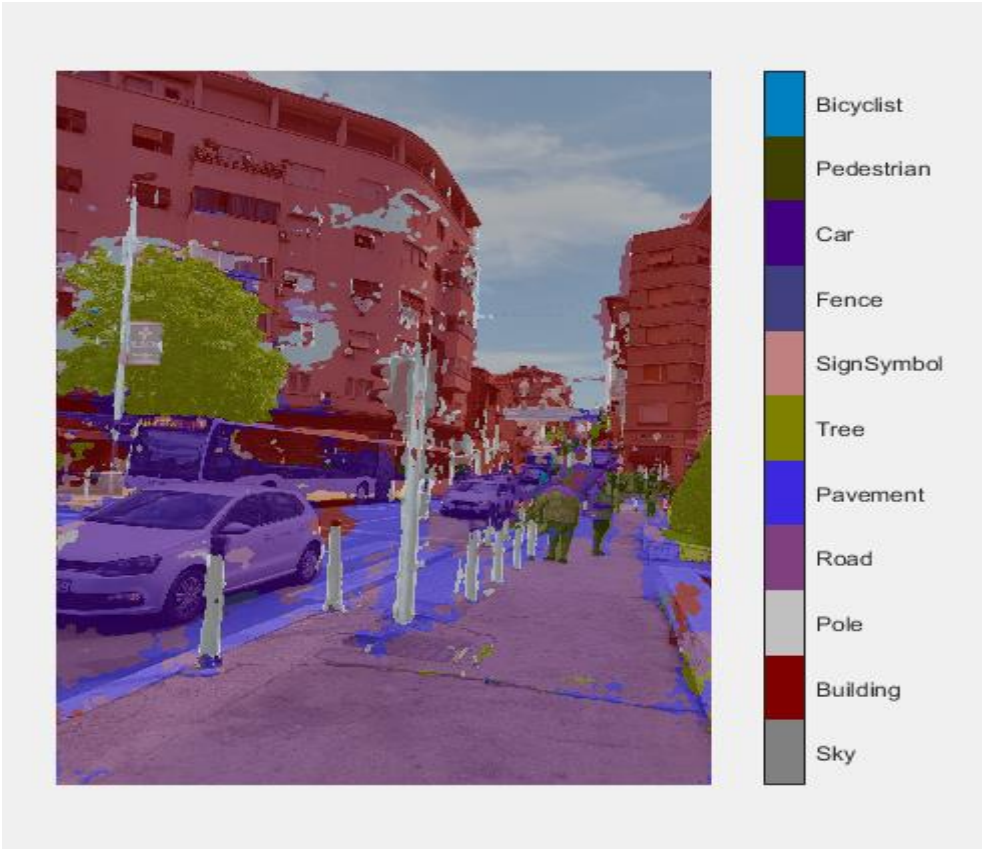
```
metrics.ClassMetrics
```

.

```
ans =

  11×3 table

                Accuracy      IoU      MeanBFScore
                ________    _______    ___________

    Sky         0.93111     0.90209      0.8952
    Building    0.78453     0.76098     0.58511
    Pole        0.71586     0.21477      0.5144
    Road        0.93024     0.91465     0.76696
    Pavement    0.88466     0.70571     0.70919
    Tree        0.87377     0.76323     0.70875
    SignSymbol  0.79358     0.39309     0.48303
    Fence       0.81506     0.46484     0.48564
    Car         0.90956     0.76799     0.69233
    Pedestrian  0.87629     0.43659     0.60792
    Bicyclist   0.87844     0.60829     0.55089
```



Slika 22 Testiranje mreže na slici koja nije u datasetu (1)



Slika 23 Testiranje mreže na slici koja nije u datasetu (2)

6 PRAKTIČNI RAD – SEMANTIČKA SEGMENTACIJA (DRUGI PRIMJER)

6.1 Postavljanje osnovnih parametara

Kao i u prethodnom primjeru kreira se Deeplab v3 + mreža s utezima inicijaliziranim iz unaprijed obučene mreže Resnet-18. Za razliku od prethodnog primjera nije potrebno preuzeti unaprijed obučenu verziju DeepLab v3+, zato što će se odraditi obuka mreže.

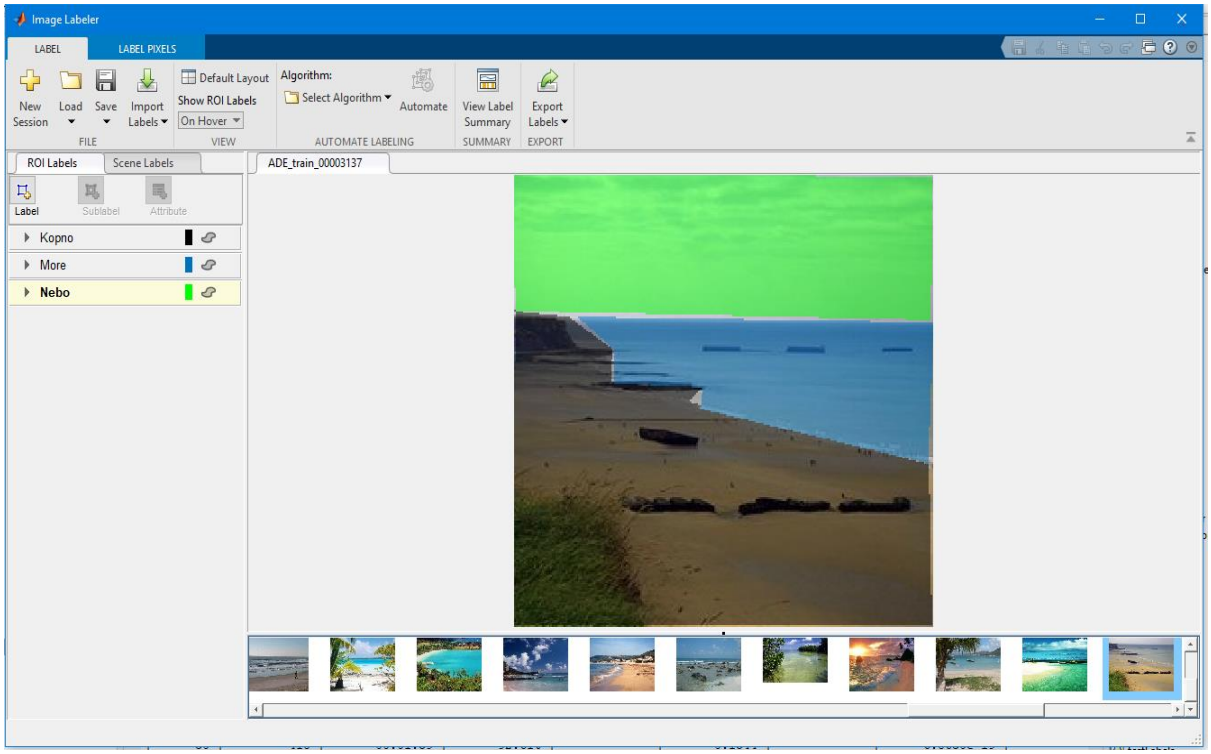
```
resnet18();
```

6.2 ADE20K Dataset

Za ovaj primjer korišten je ADE20K dataset koji je napravljen na MIT-u (Massachusetts Institute of Technology). U ovom datasetu se nalazi preko 22000 slika. Komprimirana verzija dataseta zauzima 3.8Gb kako bi se omogućilo brže preuzimanje. U ovom primjeru korištene u slike iz mape 'beach'.

6.3 Image labeler

Image Labeler je izuzetno koristan alat za dodavanje oznaka na sliku. Na slikama su dodavane oznake piksela koje su potrebne za kasniju obuku neuronske mreže



Slika 24 Image Labeler

.Mapa 'beach' u datasetu ADE20K sadrži 72 slike koje su učitane u Image Labeler. Na svakoj od 72 slike za potrebe ovog primjera ručno su označeni pikseli i raspodijeljeni u tri klase. Klasa 'Kopno' označena je crnom bojom, klasa 'More' plavom bojom, te klasa 'Nebo' zelenom bojom. Nakon što se ručno označe sve slike potebno je pohraniti podatke dobivene označavanjem piksela. Podaci se spremaju u varijablu *gTruth* koju je moguće odmah učitati u *Workspace* ili je pohraniti na disk. U varijabli *gTruth* su upisani svi podaci o označenim slikama. Ti podaci će biti potrebni za obuku mreže.

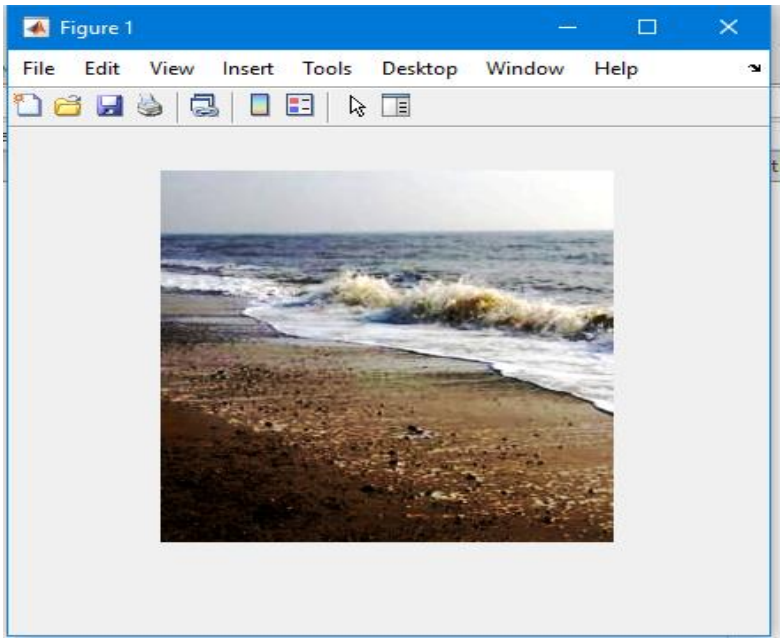
6.4 Učitavanje slika iz ADE20K dataseta

Za učitavanje većeg broja slika koristimo funkciju *imageDatastore*.

```
imgDir = fullfile('C:\Users\anter\Desktop\ante_matlab','beachim');
imds = imageDatastore(imgDir);
```

Prikaz jedne od slika:

```
I = readimage(imds,1);
I = histeq(I);
imshow(I)
```



Slika 25 Jedna slika iz dataseta

6.5 Učitavanje ADE20K slika za označenim pikselima

pxds = pixelLabelDatastore(gTruth) ekstraktira sve podatke iz varijable *gTruth*.

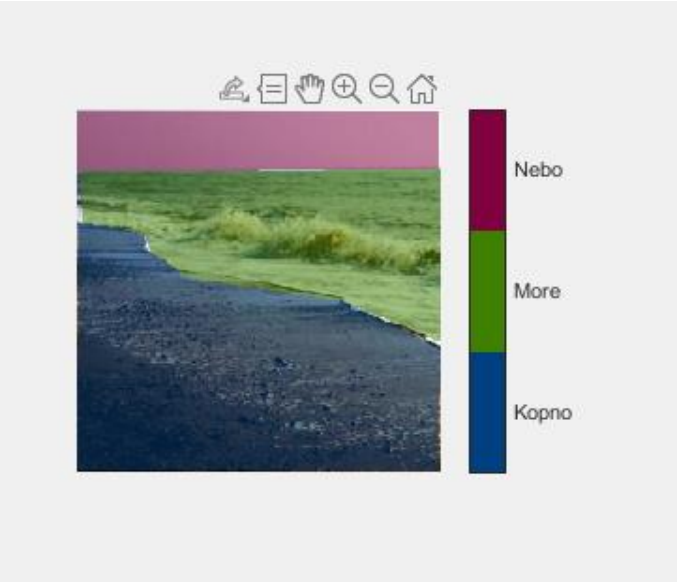
```
labelDir = fullfile('C:\Users\anter\Desktop\ante_matlab\matlab\PixelLabelData','PixelLabelData');
labelIDs = [1 2 3];
pxds = pixelLabelDatastore(gTruth);
```

Kreiranje klasa:

```
classes = [
    "Kopno"
    "More"
    "Nebo"
    ];
```

Učitavanje i prikaz jedne od slika sa označenim pikselima i preklapanje te slike preko obične slike.

```
C = readimage(pxds,1);
cmap = camvidColorMap;
B = labeloverlay(I,C,'ColorMap',cmap);
imshow(B)
pixelLabelColorbar(cmap,classes);
```



Slika 26 Preklapanje obične slike i slika s označenim pikselima

6.6 Analiziranje statistika ADE20K dataseta

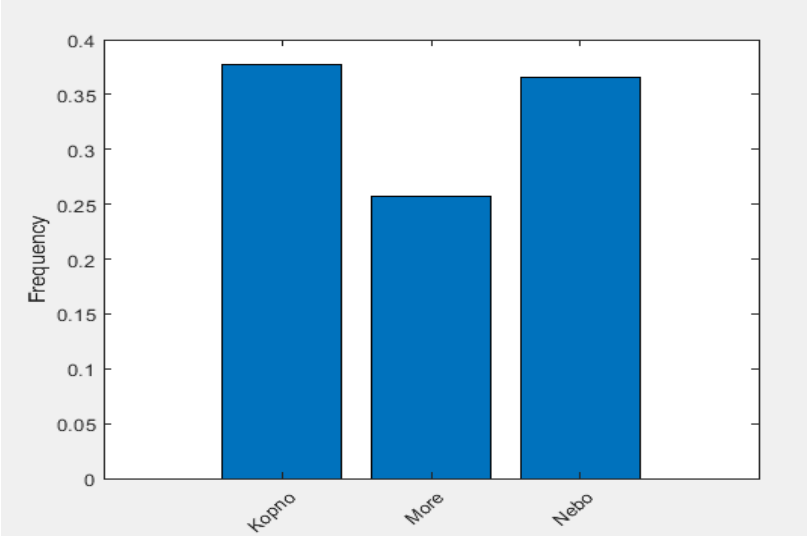
countEachLabel funkcija broji količinu piksela prema oznaci klase.

```
tbl = countEachLabel(pxds);
```

```
tbl =

  3×3 table

      Name        PixelCount    ImagePixelCount
    _________    ____________    _______________

    {'Kopno'}     1.8684e+06       5.0825e+06
    {'More' }     1.2699e+06       4.9514e+06
    {'Nebo' }     1.8101e+06       5.0825e+06
```

Na sljedećem grafu predočena je vizualizacija broja piksela po pojedinoj klasi:

```
frequency = tbl.PixelCount/sum(tbl.PixelCount);

bar(1:numel(classes),frequency)
xticks(1:numel(classes))
xticklabels(tbl.Name)
xtickangle(45)
ylabel('Frequency')
```



Slika 27 Broj piksela po pojedinoj klasi

6.7 Podjela slika na set za obučavanje, validaciju i testiranje

Dioba se vrši tako da se 60% slika koristi za obučavanje mreže, dok se 20% koristi za validaciju, te preostalih 20% za testiranje mreže:

```
rng(0);
numFiles = numel(imds.Files);
shuffledIndices = randperm(numFiles);

numTrain = round(0.60 * numFiles);
trainingIdx = shuffledIndices(1:numTrain);

numVal = round(0.20 * numFiles);
valIdx = shuffledIndices(numTrain+1:numTrain+numVal);

testIdx = shuffledIndices(numTrain+numVal+1:end);

trainingImages = imds.Files(trainingIdx);
valImages = imds.Files(valIdx);
testImages = imds.Files(testIdx);

imdsTrain = imageDatastore(trainingImages);
imdsVal = imageDatastore(valImages);
imdsTest = imageDatastore(testImages);

classes = pxds.ClassNames;


trainingLabels = pxds.Files(trainingIdx);
valLabels = pxds.Files(valIdx);
testLabels = pxds.Files(testIdx);

pxdsTrain = pixelLabelDatastore(trainingLabels, classes, labelIDs);
pxdsVal = pixelLabelDatastore(valLabels, classes, labelIDs);
pxdsTest = pixelLabelDatastore(testLabels, classes, labelIDs);
```

6.8 Kreiranje neuronske mreže

ADE20K dataset sadrži slike koje imaju rezoluciju 256x256 piksela. Postavljamo zadani broj klasa te kreiramo Deeplab v3+ :

```
imageSize = [256 256 3];


numClasses = numel(classes);


lgraph = deeplabv3plusLayers(imageSize, numClasses, "resnet18");
```

Koristi se broj oznaka piksela proračunat ranije s *countEachLabel* funkcijom i računa se srednja frekvencija težina klasa. Izjednačavanje težine klasa olakšava obuku mreže.

```
imageFreq = tbl.PixelCount ./ tbl.ImagePixelCount;
classWeights = median(imageFreq) ./ imageFreq
```

Navođenje težine klasa pomoću *pixelClassificationLayer* funkcije:

```
pxLayer = pixelClassificationLayer('Name','labels','Classes',tbl.Name,'ClassWeights',classWeights);
lgraph = replaceLayer(lgraph,"classification",pxLayer);
```

6.9 Odabir opcija za obuku neuronske mreže

Detaljniji opis parametara neuronske mreže je prikazan u prethodnom primjeru, u podnaslovu 5.8.

```
pximdsVal = pixelLabelImageDatastore(imdsVal,pxdsVal);


options = trainingOptions('sgdm', ...
    'LearnRateSchedule','piecewise',...
    'LearnRateDropPeriod',1,...
    'LearnRateDropFactor',0.3,...
    'Momentum',0.9, ...
    'InitialLearnRate',1e-3, ...
    'L2Regularization',0.005, ...
    'ValidationData',pximdsVal,...
    'MaxEpochs',30, ...
    'MiniBatchSize',1, ...
    'Shuffle','every-epoch', ...
    'CheckpointPath', tempdir, ...
    'VerboseFrequency',2,...
    'Plots','training-progress',...
    'ValidationPatience', 4);
```

6.10 Augmentacija podataka

Ovdje se za augmentaciju podataka koristi slučajni lijevi / desni odraz i slučajna X / Y translacija od +/- 10 piksela. Upotrijebljena je *imageDataAugmenter* funkcija za određivanje ovih parametara za augmentaciju podataka.

```
augmenter = imageDataAugmenter('RandXReflection',true,...
    'RandXTranslation',[-10 10],'RandYTranslation',[-10 10]);
```

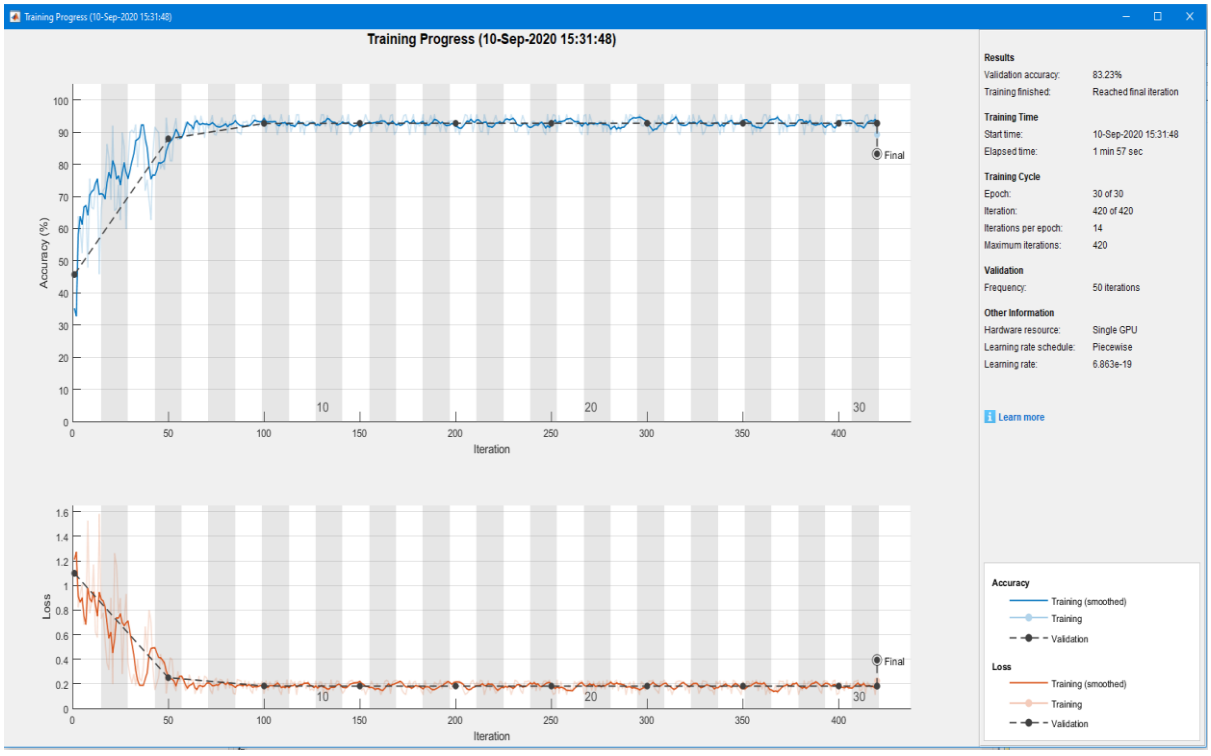
6.11 Početak obuke neuronske mreže

Kombiniraju se podaci o obuci i odabiru augmentacije podataka pomoću *pixelLabelImageDatastore*. *PixelLabelImageDatastore* čita serije podataka o obuci.

```
pximds = pixelLabelImageDatastore(imdsTrain,pxdsTrain, ...
    'DataAugmentation',augmenter);
```

Započinje se obuka mreže pomoću funkcije *trainNetwork.:*

```
data = trainNetwork(pximdsVal,lgraph,options);
```



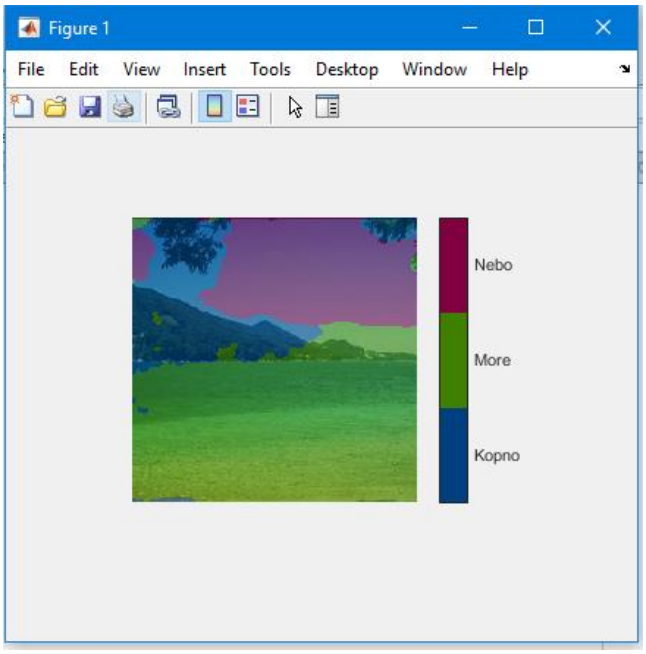
Slika 28 Obuka neuronske mreže

Može se vidjeti kako je u prvih 50 iteracija preciznost bila jako niska, na svega 50%. Što se više obuka bližila kraju rezultati su sve bolji. Obavljeno je 420 iteracija, sa ukupnom preciznošću validacije od 83.23%.

6.12 Testiranje neuronske mreže na jednoj slici

Učitava se jedna slika iz dataseta. Zatim se preko funkcije *semanticseg* vrši semantička segmentacija te slike i upisuje se u varijablu *C*. Nakon toga prikazuje se originalna slika i preko nje se preklapa segmentirana slika.

```
I = imread('plaza5.jpg');
C = semanticseg(I, data);


B = labeloverlay(I,C,'Colormap',cmap,'Transparency',0.4);
imshow(B)
pixelLabelColorbar(cmap, classes);
```



Slika 29 Testiranje mreže na jednoj slici

Količina preklapanja po klasi može se izmjeriti pomoću mjerenja intersection-over-union (IoU), također poznatog kao Jaccardov indeks. Upotrijebiti *jaccard* funkciju za mjerenje IoU-a.

```
iou = jaccard(C,expectedResult);
table(classes,iou)
```

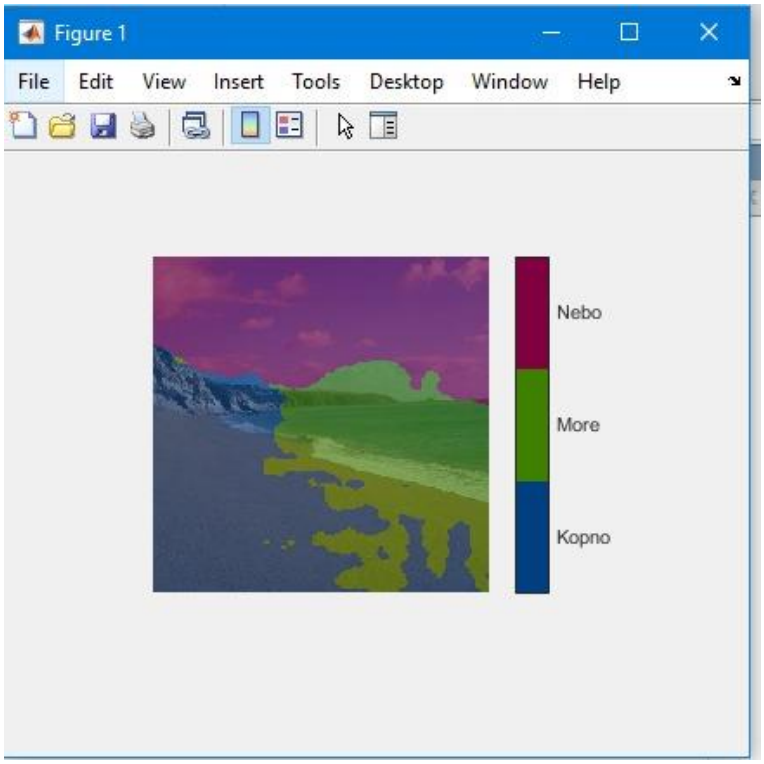
```
ans =

  3×2 table

     classes       iou
    _________    _______

    {'Kopno'}     0.31684
    {'More' }    0.029029
    {'Nebo' }     0.47493
```

Vidimo da broj preklapanja nije idealan, no zadovoljavajući je kada uzmemo u obzir da ovaj cijeli dataset sadrži 72 slike, dok je u prethodnom primjeru CamVidDataset sadržavao oko 700 slika.

6.13 Evaluacija mreže

Da bi se izmjerila točnost više testnih slika, potrebno je pokrenuti *semanticseg* na cijelom datasetu:

```
pxdsResults = semanticseg(imdsTest,data, ...
    'MiniBatchSize',4, ...
    'WriteLocation',tempdir, ...
    'Verbose',false);
```



Slika 30 Rezultat segmentacije na slici koja nije u datasetu

Semanticseg vraća rezultate za dataset kao objekt *pixelLabelDatastore*-a. Stvarni podaci oznake piksela za svaku testnu sliku u *imdsTest* zapisuju se na disk na mjestu navedenom parametrom '*WriteLocation*'. Upotrijebiti *evaluateSemanticSegmentation* za mjerenje podataka semantičke segmentacije na rezultatima dataseta:

```
metrics = evaluateSemanticSegmentation(pxdsResults,pxdsTest,'Verbose',false);

metrics.DataSetMetrics

ans =

  1×5 table

    GlobalAccuracy    MeanAccuracy    MeanIoU    WeightedIoU    MeanBFScore
    ______________    ____________    _______    ___________    ___________

        0.7345            0.72428      0.57664      0.58789        0.46833

>>
```

```
metrics.ClassMetrics


ans =

  3×3 table

              Accuracy      IoU       MeanBFScore
              ________    _______    ___________

    Kopno     0.62725     0.49101      0.45489
    More      0.62474     0.46068      0.30818
    Nebo      0.92087     0.77822      0.64192
```

Kao što se može vidjeti u rezultatima evaluacije, mreža je zadovoljavajuća s obzirom na količinu podataka za obuku. Što više podataka, u ovom slučaju slika damo na raspolaganje neuronskoj mreži, to će rezultati biti bolji.

7 DETEKCIJA OBJEKTA

7.1 Početak rada detekcijom objekata pomoću dubokog učenja

Detekcija objekata (engl. Object detection) pomoću dubokog učenja pruža brzo i precizno sredstvo za predviđanje mjesta objekta na slici. Duboko učenje moćna je tehnika strojnog učenja u kojoj detektor objekta automatski uči značajke slike potrebne za zadatke otkrivanja. Dostupno je nekoliko tehnika za detekciju objekata pomoću dubokog učenja, poput bržeg R-CNN-a (Faster R-CNN), You only look once (YOLO) v2 i Single shot detection (SSD).

Slika 31 Detekcija objekta [2]

Neke od primjena detekcije objekata su:

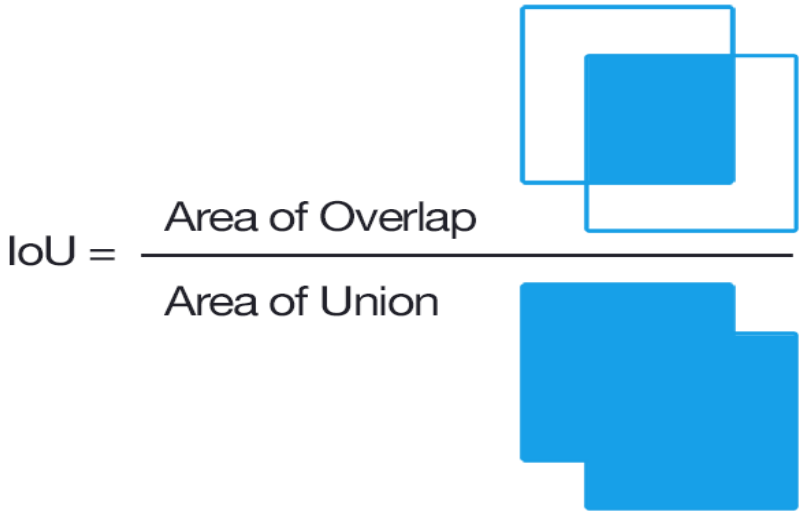
- klasifikacija slika,
- razumijevanje scene,
- autonomna vozila,
- nadzor.

Za početak potrebno je kreirati podatke za obuku detektora. Upotrebljava se neka od aplikacija za interaktivno označavanje potrebnih podataka i regija na videozapisu, sekvenci slika, zbirci slika ili prilagođenom izvoru podataka. Podatke o objekta možete označiti pomoću pravokutnih oznaka (labels) koje definiraju položaj i veličinu objekta na slici.

7.2 Upoznavanje s osnovnim parametrima

Presijecanje preko unije (engl. Intersection over Union - IoU): Ne može se očekivati da predviđanje graničnog okvira bude precizno na razini piksela, pa stoga treba definirati parametre za opseg preklapanja između 2 granična okvira. Presijecanje preko unije čini točno ono što mu naziv govori.

Uzima područje presijecanja 2 uključena granična okvira i dijeli ga s područjem njihovog preklapanja. To daje ocjenu između 0 i 1, koja predstavlja kvalitetu preklapanja između 2 okvira.



Slika 32 Presijecanje preko unije (IoU) [4]

Prosječna preciznost i prosječni opoziv: Preciznost varira ovisno o tome koliko su točna naša predviđanja, dok opoziv objašnjava jesmo li u stanju otkriti sve objekte prisutne na slici ili ne. Prosječna preciznost (engl. Average Precision - AP) i prosječni opoziv (engl. Average Recall - AR) dva su uobičajena parametra koji se koriste za otkrivanje objekata.

8 PRAKTIČNI RAD – DETEKCIJA OBJEKATA

8.1 Obučvanje R-CNN detektora stop znaka

Učitavanje podataka o obuci i mrežnih slojeva

```
load('rcnnStopSigns.mat', 'stopSigns', 'layers')
```

Dodati direktorij slika na MATLAB putanju

```
imDir = fullfile(matlabroot, 'toolbox', 'vision', 'visiondata',...
  'stopSignImages');
addpath(imDir);
```

Postaviti mogućnosti mrežne obuke za upotrebu *Mini-batch* veličine 32 za smanjenje upotrebe GPU memorije. Smanjiti *InitialLearningRate* da bi se smanjila brzina promjene mrežnih parametara. To je korisno kod finog podešavanja unaprijed obučene mreže i sprječava prebrzu promjenu mreže.

```
options = trainingOptions('sgdm', ...
  'MiniBatchSize', 32, ...
  'InitialLearnRate', 1e-12, ...
  'MaxEpochs', 10);
```

Zatim je potrebno obučiti R-CNN detektor. Obuka može potrajati neko vrijeme, ovisno o raspoloživom hardveru.

```
rcnn = trainRCNNObjectDetector(stopSigns, layers, options, 'NegativeOverlapRange', [0 0.3]);
```

detector = trainRCNNObjectDetector(trainingData,network,options) obučava R-CNN (regions with convolutional neural networks) detektor objekata.

Funkcija koristi duboko učenje za osposobljavanje detektora za otkrivanje više klasa objekata.

Ova implementacija R-CNN ne obučava SVM klasifikator za svaku klasu objekta.

```
********************************************************************
Training an R-CNN Object Detector for the following object classes:

* stopSign

--> Extracting region proposals from 27 training images...done.

--> Training a neural network to classify objects in training data...

Training on single CPU.
Initializing input data normalization.
|=========================================================================================|
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Mini-batch  |  Base Learning  |
|         |             |   (hh:mm:ss)   |   Accuracy   |     Loss     |      Rate       |
|=========================================================================================|
|       1 |           1 |       00:00:03 |       96.88% |       0.0381 |    1.0000e-12 |
|       2 |          50 |       00:00:46 |       96.88% |       0.4982 |    1.0000e-12 |
|       3 |         100 |       00:01:28 |      100.00% |       0.0002 |    1.0000e-12 |
|       5 |         150 |       00:02:09 |       96.88% |       0.0641 |    1.0000e-12 |
|       6 |         200 |       00:02:49 |      100.00% |       0.0021 |    1.0000e-12 |
|       8 |         250 |       00:03:30 |       96.88% |       0.2457 |    1.0000e-12 |
|       9 |         300 |       00:04:11 |       96.88% |       0.0230 |    1.0000e-12 |
|      10 |         350 |       00:04:52 |       96.88% |       0.3410 |    1.0000e-12 |
|=========================================================================================|

Network training complete.

--> Training bounding box regression models for each object class...100.00%...done.

Detector training complete.
********************************************************************
```

Testiranje R-CNN detektora na jednoj slici:

```
img = imread('stop1.jpg');

[bbox, score, label] = detect(rcnn, img, 'MiniBatchSize', 32);
```

Prikaz dobivenih rezultata:

Maknuti direktorij slike sa MATLAB putanje:

```
rmpath(imDir);
```

8.2 Nastaviti obuku R-CNN detektora

Nastaviti obučavati R-CNN detektor objekata koristeći dodatne podatke. Da bi se ilustrirao ovaj postupak, pola provjereno točnih podataka koristit će se za početnu obuku detektora. Zatim se nastava nastavlja s korištenjem svih podataka.
Učitati podatke o obuci i inicijalizirati opcije obuke.

Obučiti R-CNN detektor s dijelom provjerenih podataka (ground-truth):

```
load('rcnnStopSigns.mat', 'stopSigns', 'layers')

stopSigns.imageFilename = fullfile(toolboxdir('vision'),'visiondata', ...
    stopSigns.imageFilename);

options = trainingOptions('sgdm', ...
    'MiniBatchSize', 32, ...
    'InitialLearnRate', 1e-12, ...
    'MaxEpochs', 10, ...
    'Verbose', false);
```

Uzeti obučene mrežne slojeve iz detektora. Kad proslijedite niz mrežnih slojeva u *trainRCNNObjectDetector*, oni se koriste takvi kakvi jesu za nastavak obuke.

```
network = rcnn.Network;
layers = network.Layers;
```

Nastaviti obučavanje uz korištenje svih podataka:

```
rcnnFinal = trainRCNNObjectDetector(stopSigns, layers, options);
```

8.3 Kreiranje R-CNN detektora objekata za dvije klase

Kreirati R-CNN detektor objekata za dvije klase objekata: pse i mačke.

```
objectClasses = {'dogs','cats'};
```

Mreža mora biti u mogućnosti klasificirati i pse, mačke i "pozadinsku" klasu kako bi se mogla obučavati pomoću *trainRCNNObjectDetector*.

U ovom se primjeru dodaje jedna klasa koja uključuje pozadinu.

```
numClassesPlusBackground = numel(objectClasses) + 1;
```

Potpuno povezani sloj mreže definira broj klasa koje mreža može klasificirati. Potrebno je postaviti veličinu izlaza potpuno povezanog sloja da odgovara zbroju broja klasa i pozadinske klase.

```
layers = [ ...
    imageInputLayer([28 28 1])
    convolution2dLayer(5,20)
    fullyConnectedLayer(numClassesPlusBackground);
    softmaxLayer()
    classificationLayer()];
```

Ovi mrežni slojevi se sad mogu koristiti za obuku R-CNN detektora objekata za dvije klase.

8.4 Upotrijebiti spremljenu mrežu za R-CNN detektor objekata

Kreirati R-CNN detektor objekata i postaviti ga tako da koristi spremljenu mrežnu kontrolnu točku (checkpoint). Mrežna kontrolna točka sprema se za vrijeme svake epohe tijekom mrežne obuke kada je postavljen parametar *'CheckpointPath' (Trainingpoint)*. Mrežne kontrolne točke korisne su u slučaju da se obuka mreže neočekivano završi.

Učitajte podatke o obuci stop znaka.

```
load('rcnnStopSigns.mat','stopSigns','layers')
```

Dodati slike na putanju.

```
stopSigns.imageFilename = fullfile(toolboxdir('vision'),'visiondata', ...
      stopSigns.imageFilename);
```

Postaviti *'CheckpointPath'* koristeći *trainigOptions* funkciju

```
checkpointLocation = tempdir;
options = trainingOptions('sgdm','Verbose',false, ...
    'CheckpointPath',checkpointLocation);
```

Obučiti R-CNN detektor objekata sa jednom slikom.

```
rcnn = trainRCNNObjectDetector(stopSigns(1:1,:),layers,options);
```

Učitati spremljenu mrežnu kontrolnu točku.

```
wildcardFilePath = fullfile(checkpointLocation,'net_checkpoint__1__2020_08_25__19_35_28.mat');
contents = dir(wildcardFilePath);
```

Učitati jednu od mreža kontrolnih točaka.

```
filepath = fullfile(contents(1).folder,contents(1).name);
checkpoint = load(filepath);

checkpoint.net
```

```
ans =

  SeriesNetwork with properties:

         Layers: [15×1 nnet.cnn.layer.Layer]
     InputNames: {'imageinput'}
    OutputNames: {'classoutput'}
```

Kreirati novi R-CNN detektor objekata i postaviti ga tako da koristi spremljnu mrežu.

```
rcnnCheckPoint = rcnnObjectDetector();
rcnnCheckPoint.RegionProposalFcn = @rcnnObjectDetector.proposeRegions;
```

Postaviti mrežu na spremljenu mrežnu kontrolnu točku.

```
rcnnCheckPoint.Network = checkpoint.net
```

```
rcnnCheckPoint =

  rcnnObjectDetector with properties:

              Network: [1×1 SeriesNetwork]
    RegionProposalFcn: @rcnnObjectDetector.proposeRegions
           ClassNames: {'stopSign'  'Background'}
    BoxRegressionLayer: ''
```

9 ZAKLjUČAK

U teoretskom dijelu rada su razrađeni koncepti dubokog učenja, semantičke segmentacije i detekcije objekata. Može se zaključiti da ta područja jako brzo napreduju i pronalaze sve širu primjenu u svakidašnjem svijetu. Automobilska industrija će se u budućnosti zasnivati na ovim konceptima, odnosno autonomna vozila ne bi bila moguća bez ovih koncepata. Na rezultatima prvog primjera semantičke segmentacije očitavaju se dobri rezultati. Za bolje rezultate nužno je imati vrlo moćan hardver kako bi se sustavu omogućila potrebna memorija. Drugi primjer semantičke segmentacije pokazuje nešto lošije rezultate. Uzrok tomu je manja količina podataka za obuku mreže. Kada bi se za tu istu mrežu opskrbilo nekoliko stotina slika za obuku, validaciju i testiranje rezultati bi bili mnogo bolji. Također je nužno za bolje rezultate neuronsku mrežu opskrbiti velikom količinom podataka koji su joj potrebni za učenje. Konkretno za semantičku segmentaciju i detekciju objekata ti podaci su slike koje imaju označene piksele. Što više takvih slika mreža obradi, bolje će učiti i dati će bolje rezultate. Danas postoji mnogo datasetova koji sadrže tisuće označenih slika kako bi uštedili vrijeme, a također postoji i aplikacije pomoću kojih to možemo činiti ručno.

10 LITERATURA

[1] Ansari, A.A., Borse, R.Y.*: Automatic Recognition of Offline Handwritten Urdu Digits In Unconstrained Environment Using Daubechies Wavelet Transforms*, 2013., https://www.semanticscholar.org/paper/Automatic-Recognition-of-Offline-Handwritten-Urdu-Ansari-Borse/d8633f4fc4a94b45c4c18bbbc0a640e3db368a05, (pristupljeno 13.8.2020.)

[2] Brownlee, J.: *A Gentle Introduction to Object Recognition With Deep Learning*, 2019., https://machinelearningmastery.com/object-recognition-with-deep-learning/, (pristupljeno 19.8.2020.)

[3] Camacho, C.: *Convolutional Neural Networks*, 2018., https://cezannec.github.io/Convolutional_Neural_Networks/, (pristupljeno 17.8.2020.)

[4] Ganesh, P.: *Object Detection : Simplified*, 2019., https://towardsdatascience.com/object-detection-simplified-e07aa3830954, (pristupljeno 21.8.2020.)

[5] Le, J.: *How to do Semantic Segmentation using Deep learning*, 2020., https://nanonets.com/blog/how-to-do-semantic-segmentation-using-deep-learning/, (pristupljeno 10.8.2020.)

[6] Paluszek, M., Thomas, S.: *Practical MATLAB Deep Learning: A Project-Based Approach*, Apress, New York, 2020

[7] Sharma, P.: *Computer Vision Tutorial: A Step-by-Step Introduction to Image Segmentation Techniques (Part 1)*, 2019., https://www.analyticsvidhya.com/blog/2019/04/introduction-image-segmentation-techniques-python/, (pristupljeno 11.8.2020)

[8] Valada, A., Burgard, W., Vertens, J.: *SMSnet: Semantic motion segmentation using deep convolutional neural networks*, Conference Paper, 2017.

[9] *Wang, M.: Computer Vision, a Branch of Artificial Intelligence on Personal Devices.* 2018., https://medium.com/@miccowang/computer-vision-the-closet-thing-to-ai-on-our-personal-device-d2ff63994856, (pristupljeno 11.8.2020.)

[10] https://hackernoon.com/rnn-or-recurrent-neural-network-for-noobs-a9afbb00e860, (pristupljeno 13.8.2020.)

[11] https://mc.ai/a-beginners-guide-to-build-stacked-autoencoder-and-tying-weights-with-it/, (pristupljeno 7.8.2020.)

[12] https://medium.com/@aicompare/optical-character-recognition-ocr-which-solution-to-choose-cd4f829c4e5, (pristupljeno 5.8.2020.)

[13] https://www.mathworks.com/help/vision/deep-learning-semantic-segmentation-and-detection.html, (pristupljeno 15.7.2020.)

[14] https://www.mathworks.com/help/vision/examples/semantic-segmentation-using-deep-learning.html, (pristupljeno 15.7.2020.)

[15] https://www.mathworks.com/help/vision/ref/trainrcnnobjectdetector.html, (pristupljeno 20.7.2020.)

[16] https://www.mathworks.com/help/vision/ug/getting-started-with-object-detection-using-deep-learning.html, (pristupljeno 19.7.2020.)

[17] https://www.mathworks.com/help/vision/ug/getting-started-with-r-cnn-fast-r-cnn-and-faster-r-cnn.html#mw_b13dd767-cabe-43d9-a9ac-1b42716c4294.mw_55d9c48a-757e-41d2-9e65-fb9d4f0fff9b, (pristupljeno 22.7.2020.)

[18] https://www.mathworks.com/help/vision/ug/getting-started-with-semantic-segmentation-using-deep-learning.html, (pristupljeno 15.7.2020.)

[19] https://www.researchgate.net/figure/Typical-structure-of-a-feed-forward-multilayer-neural-network_fig1_291339457, (pristupljeno 15.8.2020.)

[20] https://www.rfwireless-world.com/Terminology/automatic-speech-recognition-system.html, (pristupljeno 15.8.2020.)

11 POPIS SLIKA

Slika 1 Jednoslojna (lijevo) i višeslojna (desno) neuronska mreža [6] 2
Slika 2 Neuronska mreža s povratnom vezom [10] 5
Slika 3 Konvolucijska neuronska mreža[3] 5
Slika 4 Zbijeni autoenkoder[11] 6
Slika 5 Primjer analize slike algoritmom računalnog vida [12] 8
Slika 6 Sustav prepoznavanja govora [20] 9
Slika 7 Analiza rukopisa [1] 10
Slika 8 Google prevoditelj 10
Slika 9 Ciljno usmjeravanje [6] 11
Slika 10 Neuron sa dva ulaza [6] 12
Slika 11 Veza između dva neurona [6] 13
Slika 12 Tri aktivacijske funkcije [6] 13
Slika 13 Semantička segmentacija [7] 14
Slika 14 Semantička segmentacija temeljena na prepoznavanju regije [2] 15
Slika 15 Struktura FCN-a [5] 17
Slika 16 Slabo nadzirana semantička segmentacija [5] 17
Slika 17 Prikaz slike iz dataseta 19
Slika 18 Preklapanje slike sa označenim pikselima i obične slike 22
Slika 19 Broj piksela po klasi 23
Slika 20 Testiranje mreže na jednoj slici 28
Slika 21 Usporedba dobivenih i očekivanih rezultata 28
Slika 22 Testiranje mreže na slici koja nije u datasetu (1) 31
Slika 23 Testiranje mreže na slici koja nije u datasetu (2) 31
Slika 24 Image Labeler 32
Slika 25 Jedna slika iz dataseta 33
Slika 26 Preklapanje obične slike i slika s označenim pikselima 34
Slika 27 Broj piksela po pojedinoj klasi 35
Slika 28 Obuka neuronske mreže 38
Slika 29 Testiranje mreže na jednoj slici 39
Slika 30 Rezultat segmentacije na slici koja nije u datasetu 40
Slika 31 Detekcija objekta [2] 42
Slika 32 Presijecanje preko unije (IoU) [4] 43

12 POPIS KRATICA

AI - Artificial Inteligence

AP - Average Precision

AR - Average Recall

CNN - konvolucijska neuronska mreža

CV - Computer Vision

ELM - Extreme Learning Machine

FCN - Fully Convolutional Network

FR - CNN-Faster R-CNN-

IoU - Intersection over UnioN

LSTM - Long Short-Term Memory Networks

OCR - Optical Character Recognition

R-CNN - regije sa značajkama CNN-a

RNN - Recurrent Neural Network

SSD - Single shot detection

SVM - Support Vector Machines

TCMs - Temporal Convolutional Machines

YOLO - You only look once

MIT - Massachusetts Institute of Technology

13 PRILOG

Na CD-u se nalazi kompletan kod za oba primjera semantičke segmentacije, te kod za primjer detekcije objekata. Također su dodane sve slike iz CamVidDataseta, uključujući i one sa označenim pikselima, te korištene slike iz ADE20K dataseta i slike koje su ručno označene.