SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 2293

VREDNOVANJE MODELA SWIFTNET ZA SEMANTIČKU SEGMENTACIJU

Matija Krivošić

Zagreb, lipanj 2020.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 2293

VREDNOVANJE MODELA SWIFTNET ZA SEMANTIČKU SEGMENTACIJU

Matija Krivošić

Zagreb, lipanj 2020.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 13. ožujka 2020.

DIPLOMSKI ZADATAK br. 2293

Pristupnik:	**Matija Krivošić (0036483165)**
Studij:	Računarstvo
Profil:	Računarska znanost
Mentor:	prof. dr. sc. Siniša Šegvić

Zadatak: **Vrednovanje modela SwiftNet za semantičku segmentaciju**

Opis zadatka:

Semantička segmentacija prirodnih scena važan je zadatak računalnog vida s mnogim zanimljivim primjenama. U posljednje vrijeme vrlo zanimljive rezultate na tom zadatku postižu konvolucijski modeli s lakom klasifikacijskom osnovom i ljestvičastim naduzorkovanjem. U okviru rada, potrebno je istražiti postojeće pristupe za semantičku segmentaciju. Posebnu pažnju posvetiti memorijskim zahtjevima naduzorkovanja latentnih reprezentacija. Naučiti i vrednovati model SwiftNet na javno dostupnim skupovima za semantičku segmentaciju. Validirati hiperparametre, prikazati i ocijeniti ostvarene rezultate te provesti usporedbu s rezultatima iz literature. Predložiti pravce budućeg razvoja. Radu priložiti izvorni kod razvijenih postupaka uz potrebna objašnjenja i dokumentaciju. Citirati korištenu literaturu i navesti dobivenu pomoć.

Rok za predaju rada: 30. lipnja 2020.

SADRŽAJ

SAŽETAK

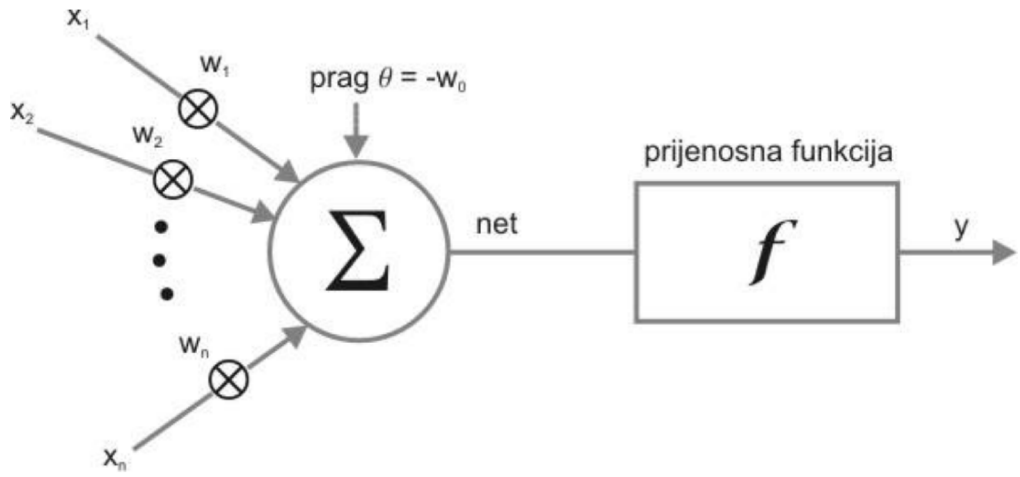
Semantička segmentacije je težak problem računalnog vida. Zbog svoje primjene u brojnim područjima kao što su automobilska industrija i robotika važno je da se segmentacija izvršava s visokom točnošću i u stvarnom vremenu. Zadatak ovog rada je treniranje i evaluiranje SwiftNet model. Eksperimenti su provedeni na CamVid i Cityscapes skupovima podataka. SwiftNet model je pokazao vrlo dobre rezultate. Postignuti rezultati su prikazani vizualno i numerički.

1. Uvod

U ovom diplomskom radu proučavat će se, trenirati i evaluirati model SwiftNet. Ovaj model obrađuje težak problem računalnog vida, a to je semantička segmentacija. Za rješavanje problema semantičke segmentacije najčešće se koriste konvolucijske neuronske mreže. Samim nizanjem novih slojeva i produbljavanjem modela neće se postići mnogo jer osim visoke točnosti modela potrebno je da se taj model izvršava u stvarnom vremenu. Rješavanje problema semantičke segmentacije u stvarnom vremenu je vrlo bitno zbog sve šire primjene u autonomnoj vožnji, robotici i drugim područjima gdje su točnost i brzina izvođenja ključni faktori. Model se bazira na tri gradivna bloka to su: koder za raspoznavanje, dekoder za povećanje dimenzija mape značajki i modul za povećanje receptivnog polja. Uloga svakog od ta tri bloka je objašnjena u nastavku. Model SwiftNet je učen nadziranim učenjem na dva skupa podataka koji se razlikuju po veličini i dimenzijama slika.
U uvodnom djelu ovog rada upoznajemo se s osnovama dubokih neuronskih mreža i dubokih konvolucijskih mreža te treniranjem takvih modela nadziranim učenjem. U idućem poglavlju objašnjeno je što je to semantička segmentacija. Upoznajemo se i s arhitekturom ResNet modela, funkcijom gubitka, optimizacijom, metodom smanjenja stope učenja i interpolacijom koje sačinjavaju SwiftNet model. U poglavlju su objašnjene mjere kvalitete kojima procjenjuje moć generalizacije modela u poglavlju s rezultatima. Kao što je ranije spomenuto detaljno su objašnjena tri gradivna bloka ovog modela u trećem poglavlju. Kako bi bilo moguće reproducirati ovaj rad u poglavlju pod nazivom "Programska izvedba" i vanjske biblioteke opisan je način na koji su postignuti rezultati i koje biblioteke su korištene. Zatim opisujemo skupove izvornih slika. Opisan je ImageNet jer su preuzete težine za ResNet-18 arhitekturu predtrenirane na njemu. Opisani su još Cityscapes i CamVid na kojima je treniran i evaluiran SwiftNet. U zadnjem dijelu rada prikazani su eksperimentalni rezultati i prokomentirani su u zaključku.

1.1. Uvod u duboke neuronske mreže

U ovom poglavlju mogu se saznati osnove dubokog učenja, neuronskih mreža i konvolucijske neuronske mreže. Duboko učenje je podskupina strojnog učenja. Neuronske mreže su inspirirane ljudskim mozgom. Modeli dubokog učenja mogu se promatrati kao napredne neurosnke mreže. Umjetne neuronske mreže se sastoje od međusobno povezanih neurona.



Slika 1.1: Slike prikazuje umjetni neuron [1].

Umjetni neuron prima podatke na ulaz u neuron označene s $x_1, x_2, \ldots, x_n$, zatim se vezama od ulaza do tijela neurona prosljeđuju podaci te veze imaju težine koje se treniraju, a u tijelu neurona izvršava se aktivacijska funkcija $f(x)$ na podacima i prosljeđuje se na idući sloj izlazom iz neurona označenim s y. Neuronske mreže najčešće se koriste za raspoznavanje uzoraka, obradu slike i govora, obradu nepreciznih i nepotpunih podataka, razne simulacije i još mnogo toga.

1.2. Unaprijedne neuronske mreže

Duboka unaprijedna mreža je najjednostavniji oblik duboke neuronske mreže. Cilj unaprijednih neuronskih mreža je aproksimirati funkciju f^*. Funkcija $y = f^*(x)$ opisuje stvaran odnos između ulaza x i izlaza y, a $\hat{y} = f(x, \Theta)$ je aproksimacija. Θ predstavlja parametre koje želimo naučiti da najbolje aproksimiraju funkcije $f^* \approx f(x, \Theta)$. Funkcija f^* nije poznata za svaki x, ali znamo kako izgleda za nekih N podataka iz skupa za učenje $(x_i, y_i)_{i=1}^N$. Model je prilagođen podacima ako dobro aproksimira podatke x koje nikada nije vidio ta sposobnost naziva se sposobnost generalizacije. U ovakvom tipu neuronskih mreža informacija proizašla iz ulaznog podatka struji od ulaza prema izlazu. Svaki model se sastoji od ulaznog sloja kao što mu i samo ime govori u tom sloju podaci ulaze u model nakon toga slijedi jedan ili više skrivenih slojeva za koje algoritam učenja sam mora zaključiti kako ih iskoristiti da bi postigao željene izlaze i posljednji je izlazni sloj. Model se može izraziti preko pomoćnih varijabli $h_L, h_{L-1}, h_{L-2}, \cdots, h_1$. L označava dubina modela [12].

$$h_1 = f_1(x, \Theta_1)$$

$$\vdots$$

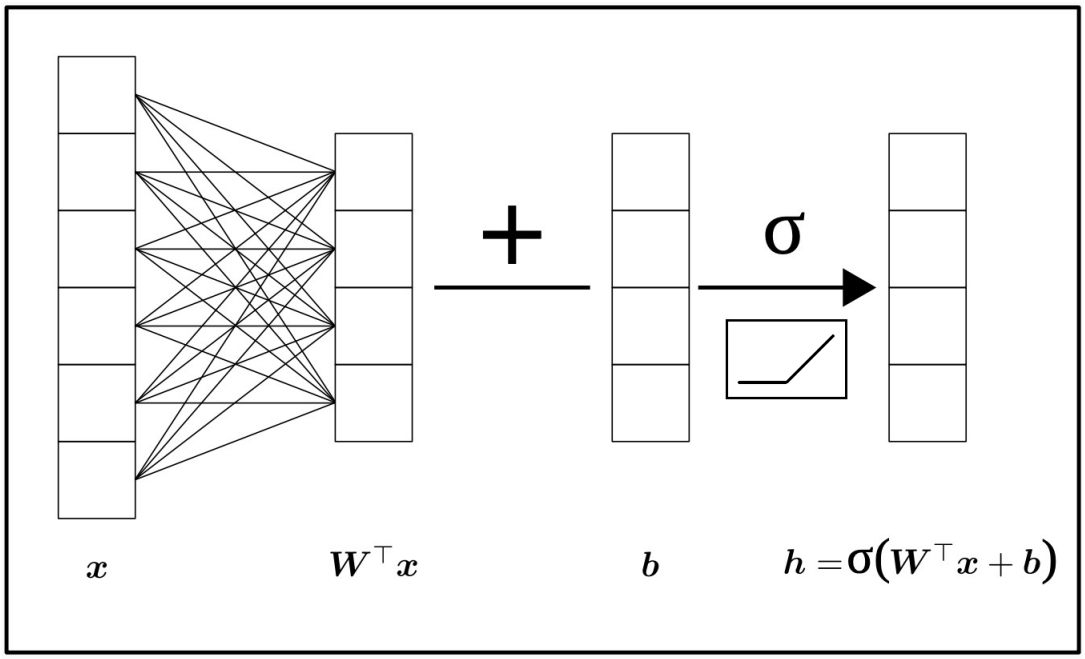
$$h_{L-1} = f_{L-1}(h_{L-2}, \Theta_{L-1})$$

$$h_L = f_L(h_{L-1}, \Theta_L)$$

$$f(x, \Theta) = o(h_L)$$

Potpuno povezana neuronska mreža osnovni je tip neuronske mreže iako se u praksi uglavnom ne koristi zbog velike računske složenosti i velikog broja parametara važno je poznavanje ovog oblika mreže. Svakom funkcijom f_i iz prethodnog izraza uvodi se nelinearnost.

Na slici 1.2 se može vidjeti kako izgleda potpuno povezani sloj u neuronskoj mreži. Svaki sloj sastoji se od više neurona. Potpuna povezanost slojeva može se matematički prikazati kao matrično množenje $W^T x$, zatim se dodaje pomak zbrajanjem po elementima te na izlazu iz sloja se provodi nelinearnost sigmoidnom funkcijom ili sve češće noviji modeli koriste zglobnicom. Nelinearne funkcije se moraju koristiti kako bi se mogli rješavati problemi koji nisu linearni. Uglavnom se biraju takve funkcije koje se mogu izvršiti na svakom elementu. To su na primjer sigmoida, ReLu, funkcija skoka i druge. Modeli strojnog učenja pa tako i dubokog se razlikuju po modelu, funkciji gubitka i optimizacijskom postupku. Nadolazeća poglavlja detaljnije opisuju optimizacijski postupka, funkcije modela i model.



Slika 1.2: Slika prikazuje potpuno povezani sloj neuronske mreže.

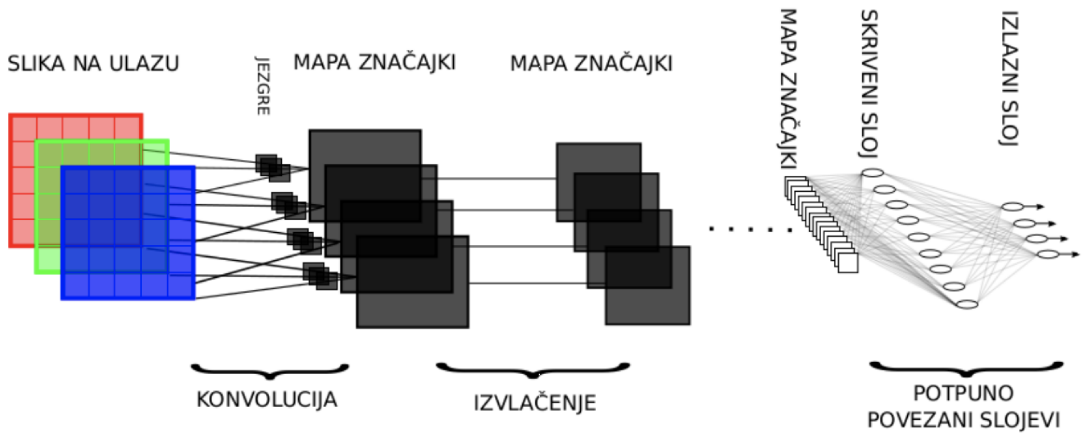
1.3. Konvolucijske neuronske mreže

Kao što je već ranije navedeno neuronske mreže se sastoje od ulaznog sloja, skrivenog sloja ili više njih i od izlaznog sloja. Ono što razlikuje klasičnu neuronsku mrežu i konvolucijsku neuronski mrežu su dva sloja koji se ponavljaju u mreži, a to su konvolucijski sloj i sloj sažimanja. Uobičajeno se ova dva prethodno navedena sloja koriste tako da najprije slijedi jedan ili više konvolucijskih slojeva zatim sloj sažimanja i tako sve dok ne dobijemo željene vrijednosti koje se prosljeđuju potpuno povezanom sloju. ti potpuno povezani slojevi prosljeđuju podatke na izlazni sloj. Način gradnje konvolucijske mreže ovisi o njenoj primjeni.

Kada ne bismo imali konvolucijski sloj, neuronska mreža bi na svom ulazu na svakom neuronu primala svaki piksel slike. Takvim načinom učenje neuronske mreže došlo bi do gubitka sposobnosti generalizacije, prenaučenosti. Još se mora uzeti u obzir da ako svaki neuron na ulazu dobije sliku vrlo brzo bi se popunila memorija računala jer za svaki neuron bi se morale pohraniti vrijednosti težina za svaku od RGB vrijednosti svakog piksela. Zato svaki neuron djeluje na samo jedan dio slike u ovakvoj vrsti neuronskih mreža.

Konvolucija je specijalna vrsta linearne operacije. Konvolucijske mreže su neuronske mreže koje koriste konvoluciju na mjestima gdje bi klasične neuronske mreže

koristile matrično množenje. Svaka neuronska mreža da bi bila konvolucijska mora imati barem jedan konvolucijski sloj umjesto potpuno povezanog. Uz konvolucijski moraju biti i sloj sažimanja i aktivacijska funkcija.



Slika 1.3: Slika prikazuje primjer konvolucijske neuronske mreže [12].

1.3.1. Konvolucijski sloj

Konvolucijski sloj transformira tenzore trećeg i četvrtog reda. Kada u strojnom učenju govorimo o konvoluciji u stvari govorimo o unakrsnoj korelaciji, a definira se formulom:

$$h(t) = (w * x)(t) = \int_{D(w)} w(\tau) \times (t + \tau) d\tau$$

Funkcija x predstavlja ulaz, w je jezgra i h je funkcija čiji se izlaz naziva mapa značajki. Konvolucija je diferencijalna operacija sa slobodnim parametrima. Iz formule je vidljivo da jezgru w se koristi za ekstrakciju lokalnih značajki. Funkcija $w(\tau)$ govori koliko $x(t + \tau)$ doprinosi izlazu $h(t)$. Konvolucija se može definirati u diskretnom obliku tada formula glasi:

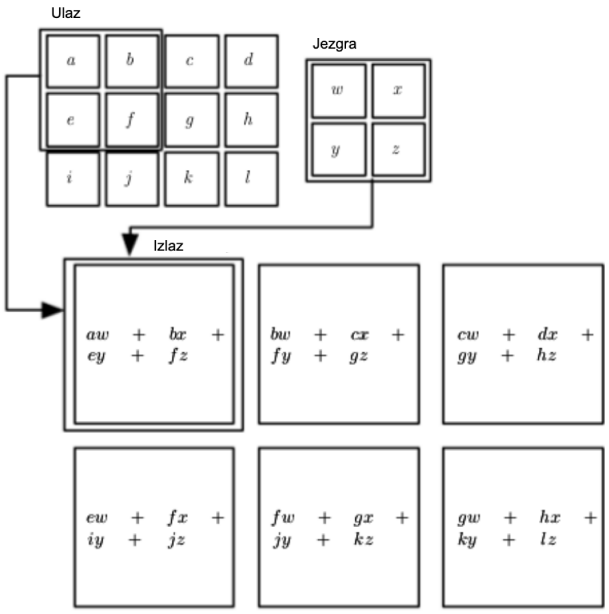
$$h(t) = \sum_{\tau=-\infty}^{\infty} w(\tau) x(t + \tau)$$

Korelacija se može primijeniti kroz više dimenzija što je potrebno kod obrade slike.

$$S(i, j) = (K * I)(i, j) = \sum_{m=m_{min}}^{m_{max}} \sum_{n=n_{min}}^{n_{max}} K(m, n) \cdot I(i + m, j + n)$$

rasponi min i max su određeni vrijednostima na kojima su I i K definirani. Konvolucijski sloj se sastoj od više filtara. U filtru se nalaze vrijednosti koje se moraju naučiti.

Filtri moraju biti manjih dimenzija od ulaza kojeg prima konvolucijski sloj, ali jednakih dubina što znači da ako ulazni uzorak ima tri kanala za svaku od te tri komponente mora postojati posebna kriška filtra. U unaprijednoj fazi potrebno je filterom proći po cijeloj širini i visini okna ulazne slike i pomnožiti ulazne aktivacije s odgovarajućim težinama filtera. Nakon toga se moraju sumirati sve dobivene vrijednosti. Rezultat na-

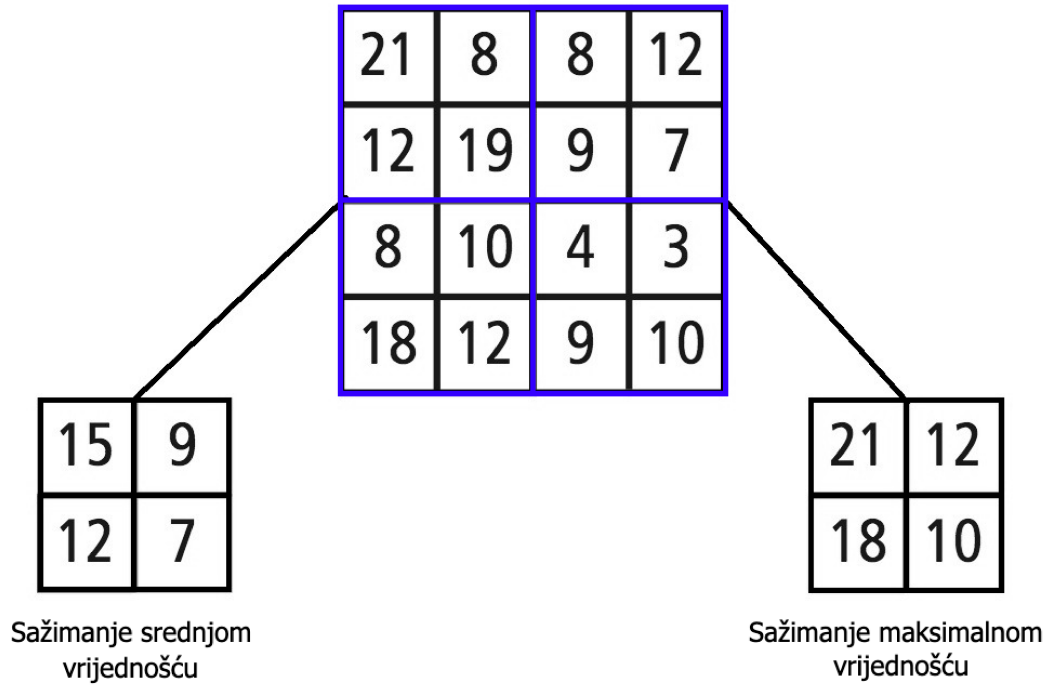


Slika 1.4: Slika prikazuje prolazak jezgre po slici [12].

šeg prolaska po slici je dvodimenzionalna aktivacijska mapa. Filteri neuronske mreže će se aktivirati u nižim slojevima zbog ruba ili boje dok u višim slojevima bi to što aktivira filter mogao biti kotač. Nakon što je svaki filter na izlaz poslao dvodimenzionalnu aktivacijsku mapu, mape se prikupljaju po svim dubinama i na izlazu šaljemo više dvodimenzionalnih matrica. Bitno je napomenuti da se mora odrediti i pomak filtera da znamo koliko će se okno pomicati po širini i po visini. Usporede li se veze u potpuno povezanom sloju i u konvolucijskom vidljivo vidljivo je da kod potpuno povezanog svaki izlaz povezan je sa svim ulazima dok kod konvolucijskog sloja izlazi su povezani samo manjim dijelom ulaza što omogućuje model s manje težina koje model mora trenirati.

1.3.2. Sloj sažimanja

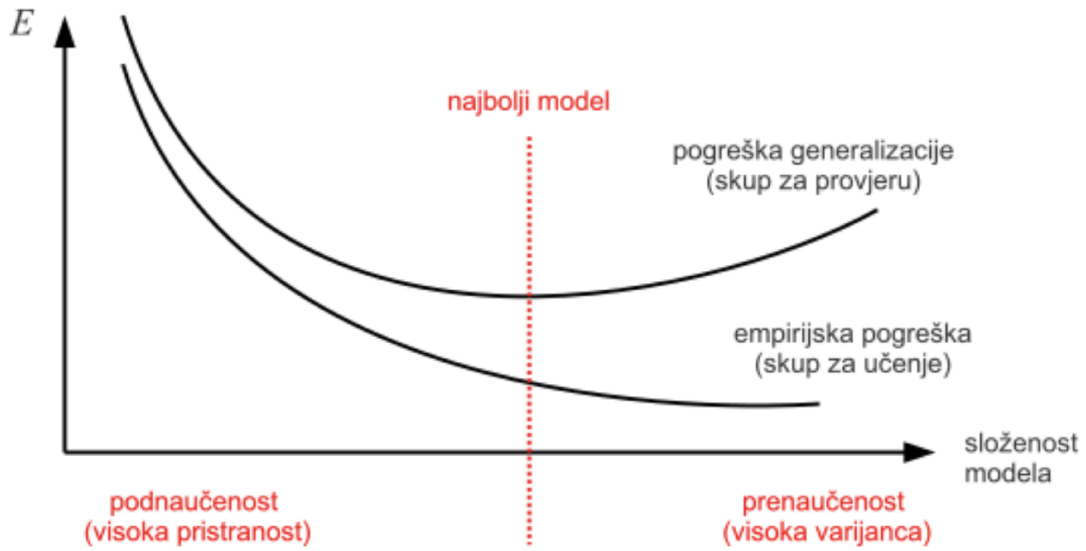
Nakon nekoliko konvolucijskih slojeva slijedi sloj sažimanja. Sloj sažimanja je bitan jer se njime smanjuje broj parametara i općenito računanja u mreži. Funkcija sažimanja mapira skup prostorno bliskih značajki na ulazu u jednu značajku na izlazu. Uglavnom se kao funkcije sažimanja koriste sažimanje srednjom vrijednosti, maksimalnom vrijednosti te Gaussovo usrednjavanje i druge. Sažimanje se može obaviti i preko različitih mapa, a ne samo preko susjednih značajki u istoj mapi. Sažimanje se najčešće provodi tako da se mapa značajki podjeli na regije uglavnom bez preklapanja te se tada svaka regija sažima u jednu značajku. Uzme li se regiju dimenzija 2×2 na njoj se provodi funkcija sažimanja i izlaz će biti 1 vrijednost ovisno o funkciji koju smo odabrali.



Slika 1.5: Slika prikazuje primjer mape značajki koja je podjeljena na regije 2×2 na svakoj od tih regija se izvršava sažimanje srednjom vrijednošću i sažimanje maksimalnom vrijednošću.

1.4. Treniranje dubokih modela

Postoje dvije faze rada neuronskih mreža prvo se provodi faza treniranja, a druga faza je faza obrade podataka. Postoji više načina učenja nadzirano učenje, nenadzirano učenje, polunadzirano učenje i podržano učenje. U ovom radu model se uči nadziranim učenjem. Učenje se provodi tako da se predočavaju podaci za koje su poznate ispravne vrijednosti izlaza, dolazi do promjene jačine veza između neurona tako se neuronske mreže uče tj. prilagođavaju viđenim podacima. Predočavanje jednog uzorka naziva se iteracijom, a jedno predočavanje svih uzoraka naziva se epoha. Složenost mreže ovisi o broju slojeva i broju neurona u tim slojevima, njihovim funkcijama i načinu povezivanja neurona u mreži. Neuronska mreža se trenira dok se moć generalizacije modela povećava.



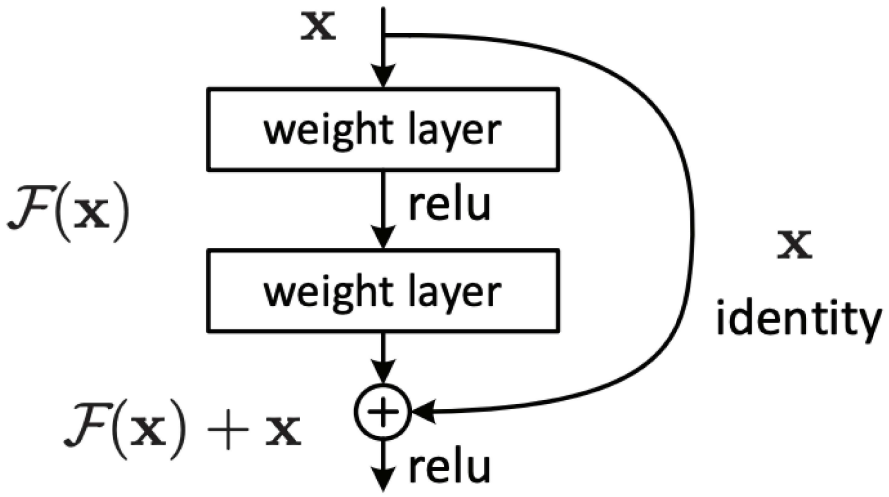
Slika 1.6: Grafički prikaz empirijske pogreške i pogreške generalizacije

U slučaju da treniranje nije zaustavljeno na vrijeme dolazi do prenaučenosti modela čime se gubi moć generalizacije. Skup podataka za učenje se dijeli na skup za treniranje, skup za validaciju i skup za testiranje. Većina skupova podataka je već podijeljena na ove skupove no ima i onih koji nisu. Takvi skupovi se mogu podijeliti na više načina na ranije navedena 3 skupa kao na primjer 60% podataka ulazi u skup za učenje, 20% skup za validaciju i 20% skup za ispitivanje. Kao što i samo ime govori na skupu za trening treniramo model, povremeno provjerimo kvalitetu modela na skupu za validaciju i na kraju testiramo na procjenjujemo generazlizacijsku točnost skupu.

2. Korišteni algoritmi i matematički aparat

2.1. ResNet

U početcima razvoja arhitektura neuronskih mreža se smatralo da dodavanjem novih slojeva u skrivenom sloju modela se postižu sve bolji rezultati. U početcima razvoja arhitektura neuronskih mreža se smatralo da dodavanjem novih slojeva u skrivenom sloju modela se postižu sve bolji rezultati. No takve duboke neuronske mreže je teško trenirati. Dolazi do zasičenje točnosti modela, a zatim dolazi i do degradacije točnosti što nije uzrokovano pretreniranošću što znači da problem nije moguće rješiti dodavanjem novih slojeva. Degradacijski problem rješava se dubokim rezidualnim modelima. Glavna ideja ResNet-a su preskočne veze koje preskaču jedan ili više slojeva. Zbog takvih veza gradijent se prilikom učenja može propagirati u ranije slojeve mreže što gladi funkciju gubitka [9]. Dodavanjem preskočnih veza ne dolazi do rasta računalne složenosti niti do rasta broja parametara. Takvoj arhitekturi mreže se može povećavati dubina kojom se može dobiti veći kapacitet modela.

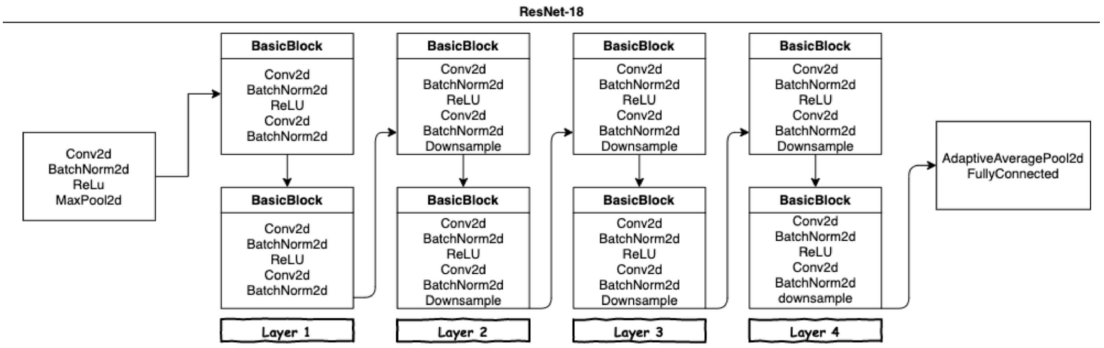


Slika 2.1: Primjer ResNet-18 arhitekture [6].

Kao što je prikazano na slici 2.1 gradivna jedinica se definira:

$$y = F(x, W_i) + x.$$

U formuli x predstavlja ulaz, a y izlaz $F(x, W_i)$ predstavlja rezidualno mapiranje koje se uči. Drugi sloj prikazane arhitekture predstavljen je formulom $F = W_2\sigma(W_1x)$. σ predstavlja zglobnica funkciju, a pomaci su izostavljene radi pojednostavljivanja notaciji. $F+x$ se izvodi preskočnim vezama i zbrajanjem po elementima [6]. Funkcija zglobnice se koristi kako bi se postigla nelinearnost.



Slika 2.2: Primjer ResNet-18 arhitekture.

2.2. Semantička segmentacija

Važna tema računalnog vida je analiza scene.Semantička segmentacija je važan alat za razumjevanje scene na razini piksela. Razumijevanje scene ima široku primjenu u području autonomne vožnje, medicinskoj dijagnostici, robotici i drugim područjima. Semantička segmentacija je postupak kojim se svakom pikselu na slici pridjeljuje semantički razred. Na primjer te klase mogu biti: nebo, auto, cesta, trava, voda. Specifično kod semantičke segmentacije je da se na izlazu iz mreže dobiva matrica u kojoj su zapisane cjelobrojne vrijednosti, a dimenzije matrice su jednake dimenzijama ulazne slike. Dok bi kod klasifikacije na izlazu dobili jednojedinični vektor. Konstruiranju arhitekture mreže namijenjene za semantičku segmentaciju može se pristupiti na više načina. Od naivnog pristupa gdje bi se naredao niz konvolucijskih slojeva je odbačen zbog prevelike računalne snage koja je potrebna kako bi se slika punih dimenzija provela kroz mrežu. Znamo da početni slojevi uče koncepte niže razine, a slojevi dublje u mreži uče specijaliziranije koncepte. Kako bismo održali učinkovitost mora se povećati broj mapi značajki. Popularan pristup koji se koristi u ovome radu je da smanje dimenzije mape značajki na početku mreže, a na kraju se poveća na dimenzije ulazne slike. Povećanje na dimenzije ulazne slike se postiže ljestvičastim naduzorkovanjem i interpolacijom.

Slika 2.3: Primjeri semantičke segmentacije.

2.3. Mjere kvalitete modela

U radu se koristi više mjera kvalitete modela kako bi se dobio objektivna informacija o generaliziranoj točnosti. Mjere u nastavku uglavnom se računaju preko matice zabune. Matrica zabune u ovom slučaju ima dimenzije:

$$broj\_klasa \times broj\_klasa$$

U matici zabune koja je korištena u ovom redu možemo vidjeti za svaki piksel na slici koliko ih je pripadalo klasi 1 i ispravno su dodijeljeni klasi 1, a koliko ostalim klasama. I tako za svaku od klasa. Iz matrice zabune takvog tipa lako je vidjeti koje klase model najčešće miješa.

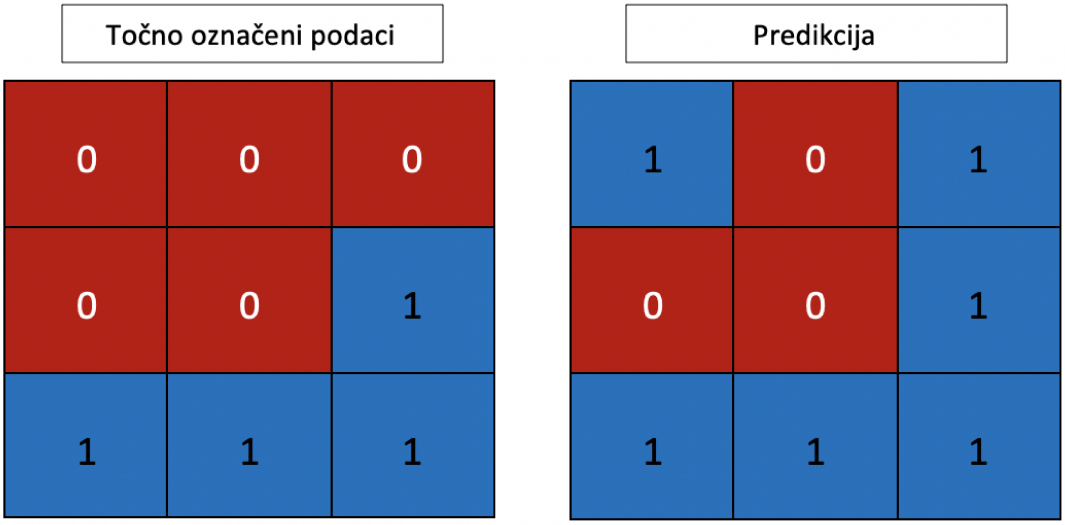
	Stvarno pozitivna	Stvarno negativna
Predviđena pozitivna	TP	FP
Predviđena negativna	FN	TN

Tablica 2.1: Prikaz jednostavne matrice zabune u slučaju binarne klasifikacije.

2.3.1. Točnost piksela

$$Acc = \frac{TP + TN}{TP + TN + FN + FP}$$

Jedna od korištenih mjera preciznosti je točnost piksela. Ova mjera nam govori kolikom postotku piksela na slici je dodijeljena točna oznaka klase. Ova mjera nije uvijek najbolji pokazatelj preciznosti. Neke klase mogu prevladavati slikom dok druge zauzimaju manju površinu slike. Svrstaju li se svi pikseli slike u klasu koja prevladava slikom dobiva se visoka preciznost što nije dobro jer su zanemarene sve klase koje su površinom manje od prevladavajuće. Iz tog razloga koristi se više mjera kvalitete modela.



Slika 2.4: Lijeva slika prikazuje točno označene podatke, a desna označava predikciju

Crvenom bojom su označeni pikseli koji pripadaju klasi 0, a plavom bojom pikseli koji pripadaju klasi 0. Na slici se može vidjeti točno označeni pikseli i primjer predikcije nekog modela i iz toga možemo izračunati točnosti piksela po ranije navedenoj formuli.

$$Acc = \frac{3+4}{3+4+0+2} = 77.8\%$$

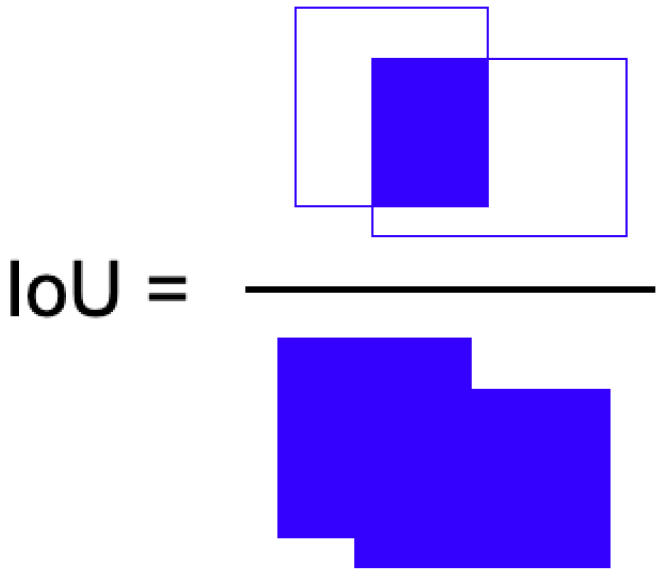
Iz slika se vidi da imamo 3 točno klasificirana piksela u klasu 0 i 4 točno klasificirana piksela u klasu 1 te 2 piksela koji u stvari pripadaju klasi 0, a dodijeljena im je klasa 1.

2.3.2. Srednji omjer presjeka i unije

Novi algoritmi semantičke segmentacije obično se ocjenjuju mjerom mean Intersection over Union (mIoU). IoU se izračunava za svaku klasu na razini piksela kao:

$$IoU = \frac{TP}{TP + FN + FP}$$

. Ova mjera računa se kao presjek točnih podataka i onih dobivenih na izlazu iz modela podijeljeno s unijom točnih podataka i onih dobivenih. To se računa za svaku klasu. Mjera mIoU se dobiva tako da izračunamo IoU vrijednost za svaku pojedinu klasu, a zatim se računa prosječna vrijednost svih klasa.

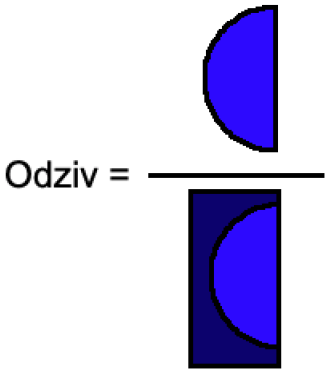


Slika 2.5: Na slici je prikazana ilustracija IoU mjere. U brojniku je presjek, a u nazivniku unije točno označenog primjera i predikcije.

2.3.3. Odziv

$$Odziv = \frac{TP}{TP + FN}$$

Recall ili odziv predstavlja omjer ispravno pozitivnih predikcija i svih ispravnih predikcija. Ova mjera nam govori koliko od ukupnog broja označenih primjera ima točno predviđenih. Odziv kao mjeru točnosti dobro je koristiti u slučajevima kada su klase u disbalansu.

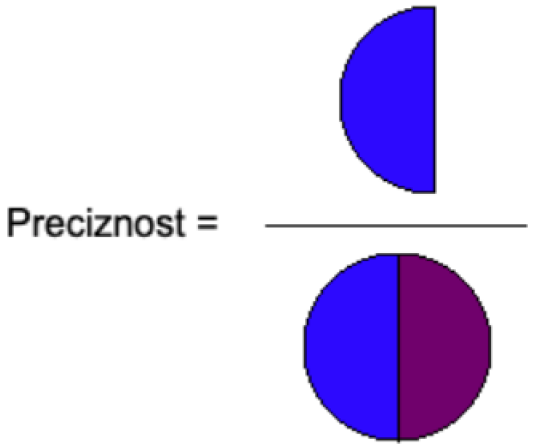


Slika 2.6: Na slici je ilustrirana mjera odziva.

2.3.4. Preciznost

$$Preciznost = \frac{TP}{TP + FP}$$

Preciznost opisuje koliko je precizan model u predikciji podataka čija je stvarna vrijednost pozitivna. Ova mjera govori koliko puta je model imao pozitivnu predikciju, a koliko puta je ta vrijednost bila točna. Kao što je vidljivo iz formule brojnik odgovara broju točnih predikcija pozitivnih primjera. U nazivniku je ukupan broj predikcija pozitivnih vrijednosti tj. i ispravne i neispravne predikcije pozitivne vrijednosti. Preciznost je pogodna za korištenje kada je visok trošak lažno pozitivnih predikcija, a trošak lažne negacije nizak. Kada govorimo o preciznosti u marici gubitaka, gubitak bi bio puno veći za lažno pozitivne nego za lažno negativne.



Slika 2.7: Na slici je ilustrirana mjera preciznosti.

2.4. Funkcija gubitka

Uobičajeno je kada se govori o semantičkoj segmentaciji kodersko-dekoderske strukture da se kao koder koristi predtrenirana mreža za klasifikaciju u našem slučaju je to ResNet-18 predtretiran na skupu podataka ImageNet kako bi se dobile mape značajki s bogatim semantičkim informacijama. Dekoder se koristi kako bi se postigla prostorna rezolucija jednaka onoj ulazne slike iz izlaza kodera. Unakrsna entropija se koristila kao funkcije gubitka kako bi se propagirala vrijednost gubitka, no uočena je slabost ove metode jer metoda na sve piksele gleda jednako. Iz toga proizlazi da pikseli na rubovima nisu precizni što ih čini teškim primjerima i samim time ih treba tretirati drugačije. Zbog ranije navedenog razloga koristi se boundry-aware funkcija gubitka. Takva funkcija više obraća pažnju na piksele uz rub. Preteča ove funkcije je (focal loss) funkcija gubitka koja obraća više pažnje na piksele s većim gubitkom. Funkcija gubitka se definira:

$$loss(L, L^{gt}) = -\frac{1}{N}\sum_{j=1}^{K}\sum_{I_i \epsilon S_j}\sum_{c=1}^{C}\alpha_j L_{i,c}^{gt}\omega(L_{i,c})logL_{i,c}$$

L predstavlja rezultat aktivacijske funkcije mape značajki na izlazu dok L^{gt} su točno označeni podaci. I_i je i-ti piksel slike I, a C je broj klasa. Svih N piksela slike I raspodijeljeno je u nekoliko grupa S_j. Grupe se baziraju na udaljenosti piksela od granice objekta. Primjenjuje se operaciju dilatacije slike na granici s promjenjivom veličinom jezgre, koja se odnosi na širinu pojasa, kako bi se dobio drugačiji skup piksela koji okružuju granicu. α_j je težina uravnoteženja, a $\omega(L_{i,c})$ je funkcija težine pažnje. Kod ovakve funkcije gubitka koriste se dva načina definiranja funkcije težine pažnje, a to su polinomijalni i eksponencijalni [14].

Polinomijalna funkcija	Eksponencijalna funkcija
$\omega(L_{i,c}) = (1 - L_{i,c})^{\lambda}$	$\omega(L_{i,c}) = e^{-\lambda(1-L_{1-L_{i,c}})}$

Tablica 2.2: Prikazane su funkcije težine pažnje koje se često koriste u ovom pristupu.

λ se koristi za kontroliranje težine pažnje. U ovome radu [10] se koristi eksponencijalna funkcija težine pažnje jer se u radu [14] se pokazala boljom.

2.5. Optimizacija

Adam je optimizacijski algoritam koji prati uprosječeno kretanje gradijenta i kvadrat gradijenta uz eksponencijalno zaboravljanje. Zalet je implicitno ugrađen u praćenje kretanja gradijenta. Ovim algoritmom korekcija se radi prema uprosječenom gradijentu, a ne prema izračunatom. Implementacija optimizacijskog algoritma Adam je robusna i radi sa širokim skupom parametara. Postoje brojne inačice ovog algoritma. Ovaj optimizacijski algoritam se koristi u radu [8].

Algorithm 1 Adam

Require: Step size ϵ

Require: Exponential decay rates for moment estimates, ρ_1 and ρ_2 in [0,1).

Require: Small constant δ used for numerical stabilization.

Require: Initial parameters θ

Initialize 1st and 2nd moment variables s=0, r=0

Initialize time step t = 0

1: **while** stopping criterion not met **do**

Sample minibatch of m examples from the training set $x^i, ..., x^m$ with corresponding targets y^i.

Compute gradient: $g \leftarrow \frac{1}{m}\nabla_\theta \sum_i L(f(x^1;\theta), y^i)$

$t \leftarrow t + 1$

Update biased first moment estimate: $s \leftarrow \rho_1 s + (1-\rho_1)g$

Update biased first moment estimate: $r \leftarrow \rho_2 r + (1-\rho_2)g \odot g$

Correct bias in first moment:$\hat{s} \leftarrow \frac{s}{1-\rho_1^t}$

Correct bias in second moment: $\hat{r} \leftarrow \frac{r}{1-\rho_2^t}$

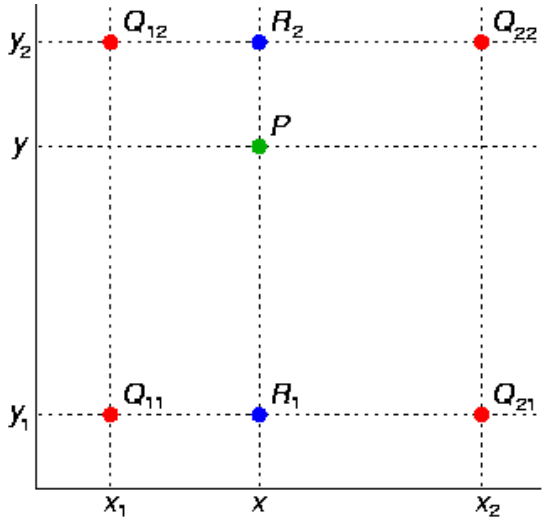
Compute update: $\nabla\theta = -\epsilon\frac{\hat{s}}{\sqrt{\hat{r}}+\delta}$

Apply update: $\theta \leftarrow \theta + \nabla\theta$

2: **end while**

2.6. Interpolacija

Interpolacija se koristi za povećanje dimenzije mape značajki kako bi se na izlazu mreže dobio izlaz jednake veličini ulazne slike. Uzmemo li na primjer sliku dimenzija 32 x 32 i želimo povećati njene dimenzije na 128 x 128 poznat je samo jedan piksel od njih 16 koliko bi svaki piksel iz početne slike trebao predstavljati u uvećanoj slici. Dva važna tipa interpolacija su bilinearna i interpolacija najbližih susjeda. Interpolacija najbližih susjeda je najjednostavnija i najbrža metoda interpolacije. Ovom metodom se promatra samo najbliži piksel onome kojeg se interpolira. Ovom metodom se postiže to da je svaki proširi na nekoliko susjednih piksela što znači da prijelazi nisu toliko glatki. Bilinearna interpolacija se računa tako da se uzme težinski prosjek četiri najbliža susjeda to jest piksela. Što je neki od ta četiri piksela bliže onome koji se interpolira to je njegov utjecaj na vrijednost veći. Bilinearnom interpolacijom se postižu glađi prijelazi nego u interpolaciji najbližih susjeda, ali je metoda nešto sporija.



Slika 2.8: Prikazan je grafički prikaz računanja bilinearne interpolacije [13].

$$R1 = f(x1, y1) + (x - x1)/(x2 - x1) * (f(x2, y1) - f(x1, y1))$$
$$R2 = f(x1, y2) + (x - x1)/(x2 - x1) * (f(x2, y2) - f(x1, y2))$$
$$P = R2 + (y - y2)/(y2 - y1) * (R1 - R2)$$

Ove četiri crvene točke na 2.8 prikazuju piksele koji su poznati. Ova metoda interpolacije radi se u 2 koraka. U prvom koraku računamo interpolaciju po X osi i to 2 puta prvo za gornje dvije točke i za doljnje dvije. Tako dobijemo točke R1 i R2 koje su na slici označene plavom bojom. Zatim te dvije točke interpoliramo po Y osi i dobivamo točku P za koju smo računali vrijednost.

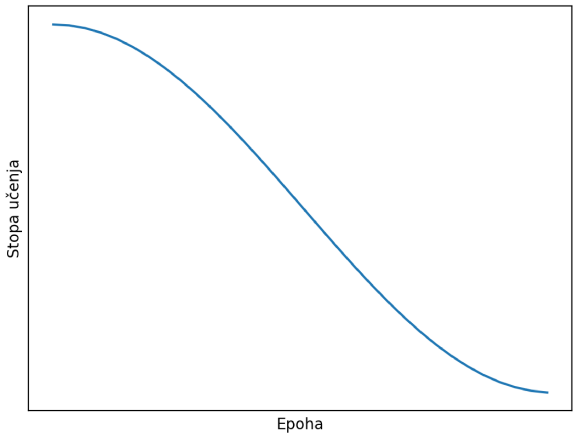
2.7. Kosinusno kaljenje

Postoje različite metode i načini na koje se smanjuje stopa učenja. Jedna jednostavna metoda je smanjivati stopu učenja svakih nekoliko iteracija. U implementaciji SwiftNeta koristi se raspored stope učenja kosinusno kaljenje. Ovdje η_{max} predstavlja početnu stopu učenja, a η_{min} posljednju stopu učenja. T_{cur} je broj epohe tekućeg pokretanja stohastičkog gradijentnog spusta.

$$\eta_t = \eta_{max} + \frac{1}{2}(\eta_{max} - \eta_{min})(1 + \cos(\frac{T_{cur}}{T_{max}}\pi)), T_{cur} \neq (2k+1)T_{max}$$

$$\eta_{t+1} = \eta_t + \frac{1}{2}(\eta_{max} - \eta_{min})(1 - \cos(\frac{1}{T_{max}}\pi)), T_{cur} = (2k+1)T_{max}$$

Kosinusno kaljenje je dobra metoda jer se učenje može započeti s visokom stopom učenja kako bi se brzo približilo lokalnom minimumu funkcije gubitka koju se želi minimizirati. Stopa učenja se postepeno smanjuje kako se približava lokalnom minimumu funkcije i kraju treninga. Trening završava s vrlo niskom stopom učenja. Na slici 2.9 je prikazan graf na kojem možemo vidjeti kako pad stope učenja izgleda kao kosinusna funkcija od pola perioda.



Slika 2.9: Na grafu se može vidjeti da pada stopa učenja kako se povećava broj epoha kroz koje je model prošao.

3. Opis metode koju rad predlaže

U ovom poglavlju opisana je arhitektura modela kao i pretpostavke na kojima se on zasniva. Model smo zasnovali na modulu za ekstrakciju značajki koji je predtreniran na ImageNet skupu podataka kako bi se postigla korist od regularizacije potaknute prijenosnim učenjem. Dekoder treba vratiti rezoluciju značajki dobivenih iz kodera. To se radi kako bi se zadržali detalji. Važno je da postupak povećanja dimenzija mape značajki bude što jednostavniji kako bi se obrada mogla vršiti u stvarnom vremenu. Gradijent se treba propagirati kroz cijelu mrežu kako bi se podržalo uspješno učenje modela. Gradijenti se propagiraju zahvaljujući ranije spomenutoj arhitekturi ResNet mreže.
Metoda segmentacije korištena u ovome radu zasniva se na tri osnovna gradivna bloka. To su koder za raspoznavanje, dekoder za povećanje dimenzija mape značajki, modul za povećanje receptivnog polja. Ovi blokovi detaljnije se opisuju u nastavku kao i način na koji se koriste.

3.1. Koder za raspoznavanje

Rad [11] predlaže korištenje modela kao što su ResNet-18 i MobileNet V2 za koje se koriste dostupne težine predtrenirane na ImageNetu. Takvi modeli sačinjavaju segmentacijski koder. U ovom radu koristi se ResNet-18. Brojni su razlozi zašto se koristi ResNet-18. Jedan od razloga je široka dostupnost predtreniranih težina za ovaj model koji je pogodan za daljnje učenje modela. Zbog svoje rezidualne strukture kao i umjerene dubine mreže ResNet-18 je pogodan za učenje od nule to jest bez predtreniranih težina. Zbog svoje umjerene dubine ResNet-18 može raditi u stvarnom vremenu.

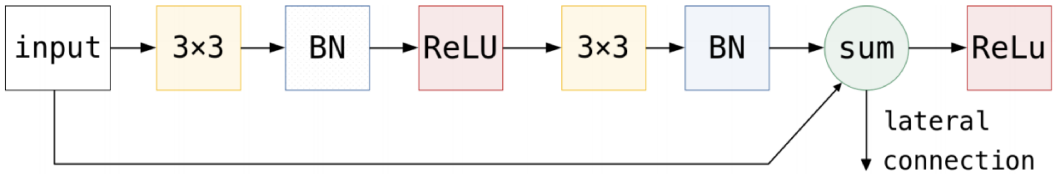
3.2. Dekoder za povećanje dimenzija mape značajki

Koder za prepoznavanje pretvara ulaznu sliku u semantički bogate mape značajki. Ove značajke moraju imati grubu prostornu razlučivost da bi se uštedjela memorija i vri-

jeme obrade. Svrha ovakvog dekodera je prikazati značajke u rezoluciji ulazne slike.

Koristimo jednostavan dekoder organiziran u niz modula za povećanje dimenzija mape značajki te su ti moduli povezani lateralnim vezama. Ovakvi moduli imaju 2 ulaza. Prvi ulaz je mapa značajki niske rezolucije kojoj trebaju biti povećana rezolucija, a drugi ulaz su lateralne značajke dobivene iz ranijih slojeva kodera. Značajke niske rezolucije prvo su povećane bilinearnom interpolacijom na dimenzije jednake dimenzijama lateralnih značajki. Ulazne značajke koje su poprimile dimenzije lateralnih izmešane su sumacijom sa lateralnim koderskim značajkama, a zatim promiješane s 3×3 konvolucijom.

Lateralne značajke s izlaza zbrajanja po elementima unutar zadnjeg rezidualnog bloka se prosljeđuju na odgovarajuću razinu poduzorkovanja. Usmjeravanje lateralnih značajki iz izlaza iz sljedećeg ReLu dovodi do znatnog pada validacijske točnosti [11]. Zamjena 3×3 konvolucije ili s 1×1 konvolucijom ili dubinski razdvojenom konvolucijom isto smanjuje točnost validacije.



Slika 3.1: Slika prikazuje strukturni dijagram zadnje rezidualne jedinice unutar konvolucijskog bloka koji koristi istu rezoluciju. Ne koristi se predaktivacija pošto nije nađena predtrenirana parametrizacija za ResNet-18 [7]. Lateralna veza je uzeta sa sumacije po elementima nakon posljednjeg rezidualnog bloka. Izlaz ReLu čvora se prosljeđuje na idući rezidualni blok [11].

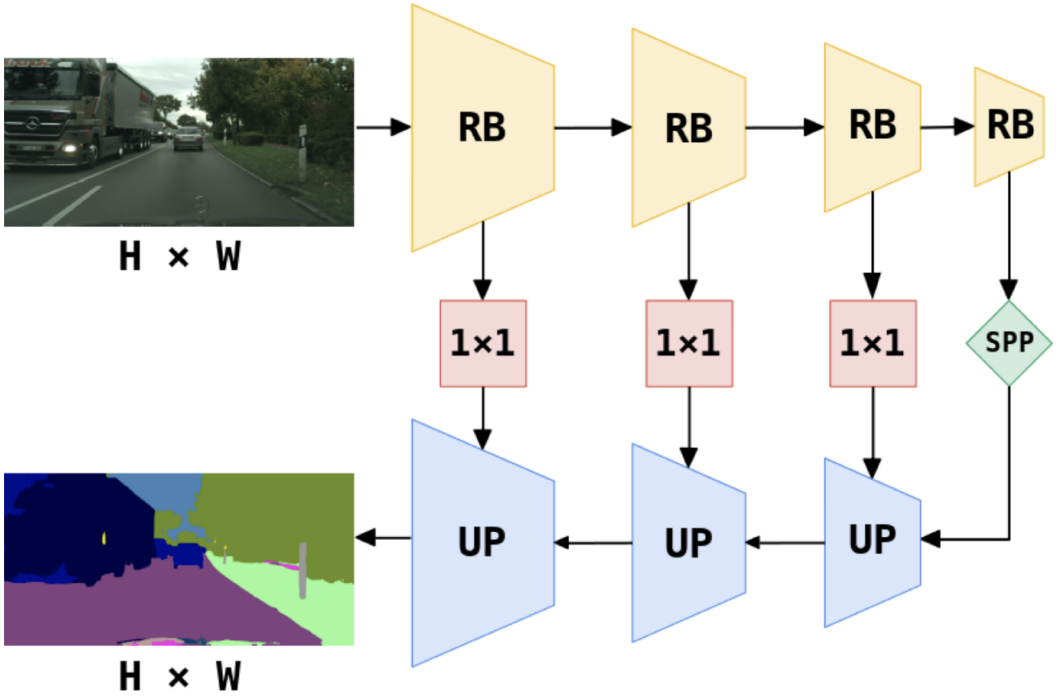
3.3. Modul za povećanje receptivnog polja

Kao što samo ime govori, modul za povećanje receptivnog polja povećava receptivno polje. Rad [11] predlaže dva pristupa kojima se to povećanje postiže, a zadržava mogućnost izvršavanja predikcije u stvarnom vremenu. To su: prostorno piramidalno sažimanje (engl. *spatial pyramid pooling*) i piramidalna fuzija (engl. *pyramid fusion*). SPP blok skuplja značajke koje je proizveo koder na različitim razinama pooling sloja i tako nastaje prikaz s različitom razinom detalja. Uporaba SPP-a je prikazana u single scale modelu. PF stvara reprezentacije na više razina koje moraju biti pažljivo spojene s dekoderom kako bi se izbjegla prenaučenost na neželjenu razinu detalja. Rad predlaže pristup grupiranja piramida koji spaja reprezentacije na različitim razinama

apstrakcije i tako proširuje receptivno polje bez žrtvovanja prostorne rezolucije.

3.3.1. Single scale model

Single scale model pretvara ulaznu sliku u semantičke predikcije tako da napjprije sliku prosljeđuje na koder koji smanjuje rezoluciju mape značajki u odnosu na ulaznu sliku, a zatim na dekoder koji povećava rezoluciju mape značajki. Na slici su prikazani trapezi žute boje koji predstavljaju konvolucijske grupe. Te konvolucijske su dio kodera za prepoznavanje, a rade na jednakim prostornim rezolucijama.



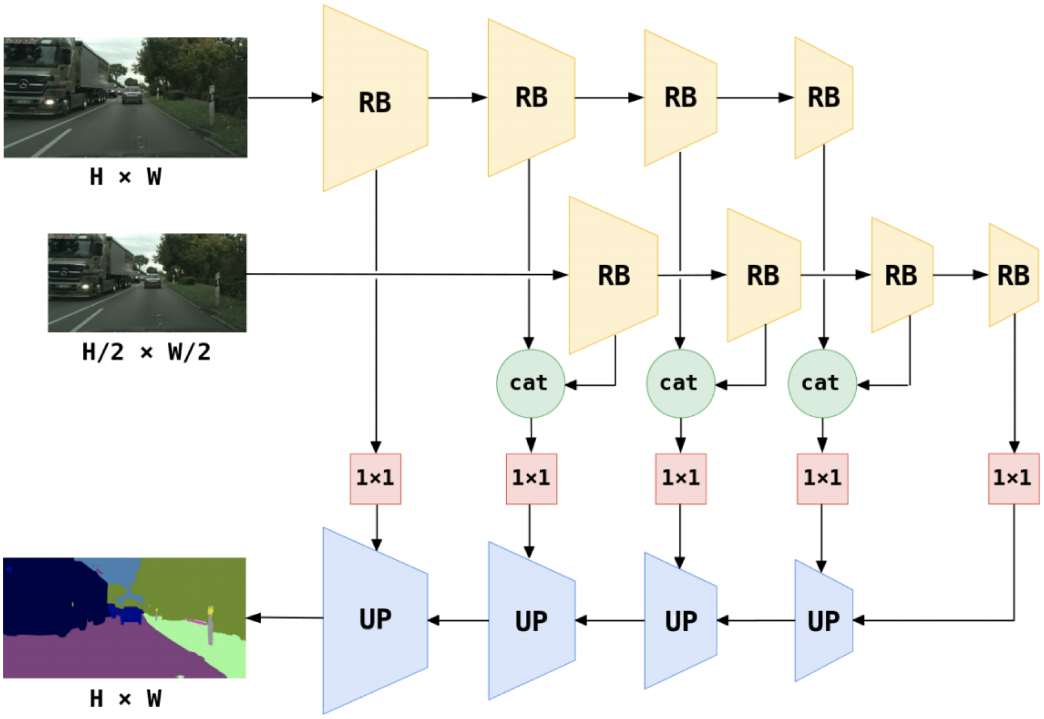
Slika 3.2: Strukturni diagram single scale modela [11].

Kodera se bazira na četiri takve konvolucijske grupe. Na izlazu iz prve konvolucijske grupe dobivaju se značajke rezolucije $H/4 \times W/4$ gdje H predstavlja visinu,a W širinu ulazne slike. Svaka iduća grupa smanjuje rezoluciju značajki za faktor 2 tako na samome kraju kodera dimenzije značajki su $H/32 \times W/32$. Tako dobivene značajke prosljeđuju se na modul za piramidalno sažimanje kako bi se povećalo receptivno polje modela. Taj sloj na slici je prikazan zelenim rombom. Dobiveni tenzori preusmjeravaju se na dekoder koji se sastoji od modula za povećanje rezolucije na slici prikazani plavim trapezima. Važno je primijetiti kako su koder i dekoder asimetrični. Koder se sastoji od više konvolucija u svakoj konvolucijskoj grupi dok dekoder ima samo jednu konvoluciju po modulu poduzorkovanja. Dimenzionalnost značajki kodera se

povećava kako se značajki približavaju kraju kodera dok je dimenzionalnost dekodera konstantna. Zbog toga lateralne veze moraju se prilagoditi dimenzionalnosti 1×1 konvolucijom. Lateralne veze su na slici prikazane crvenim kvadratima. Moduli za povećanje dimenzija mapi značajki izvršavaju se u tri koraka: 1) značajke niske rezolucije se uvećan bilinearno, 2) dobiveni reprezentacije zbrajaju se s bočnom vezom, 3) zbrajanje se miješa pomoću konvolucije 3×3.

3.3.2. Interleaved pyramid fusion model

Korištenje kompaktnog kodera doprinosi brzini (inference), ali isto tako rezultira manjim receptivnim poljem i manjim kapacitetom u usporedbi s konvolucijskim modelima osnovne svrhe za vizualno prepoznavanje. Kako bi se umanjili ti nedostaci koriste se piramide za povećanje receptivnog polja modela i smanjuje se kapacitet modela. Model je prikazan na slici. Žutom bojom prikazana su 2 instance kodera koje se primjenjuju na ulazne slike na različitim razinama piramide rezolucije. To doprinosi povećanju receptivnog polja aktivacije koje (zahvaćaju) najnižu rezoluciju slikovne piramide. Dijeljenje parametara omogućava prepoznavanje objekata različitih veličina sa zajedničkim skupom parametara, što ublažava potrebu za kompleksnosti modela. Dodaju se lateralne veze u svrhu poboljšanog protoka gradijenata kroz koder, povezuju se značajke tensora susjednih razina različitih kodera. Takva povezanost je prikazana na slici zelenim krugovima. Nakon povezivanja slijedi prosljeđivanje isprepletenih značajki kodera na prostor značajki dekoder pomoću 1×1 konvolucije, na slici prikazano crvenim kvadratima. U ovom pristupu koder funkcionira jednako kao i onaj iz prethodnog potpoglavlja jedina je razlika , ali mora postojati modul za povećanje dimenzija mapi značajki za svaku dodatnu razinu piramide.



Slika 3.3: Strukturni dijagram interleaved pyramid fusion modela [11].

4. Programska izvedba i vanjske biblioteke

4.1. NumPy

U ovom radu korišten je programski jezik Python, a jedna od važnijih biblioteka koja se koristi je NumPy. NumPy je paket za znanstvene izračune u programskom jeziku Python. Biblioteka sadrži višedimenzionalne nizove i matrice. NumPy se koristi za brojne matematičke operacije, a na brzini izvođenja može zahvaliti biblioteci napisanoj u programskom jeziku C oko koje je NumPy omot.

4.2. PyTorch

PyTorch je biblioteka koja se koristi za razvoj i učenje modela dubokog učenja. Prvenstveno ga je razvijao Facebook. PyTorch se može koristiti s Pythonom. PyTorch se često koristi u istraživanjima. Za razliku od većine drugih popularnih okvira dubokog učenja poput TensorFlowa, koji koriste statičko računanje, PyTorch koristi dinamičko računanje, što omogućava veću fleksibilnost u izgradnji složenih arhitektura.

4.3. Scikit-learn

Scikit-learn je Python biblioteka namijenjena za strojno učenje. Ova biblioteka podržava brojne klasifikacijske, regresijske i druge modele strojnog učenja. Podržava biblioteke kao što su NumPy i SciPy. Scikit-learn sadrži podršku za pretprocesiranja podataka. Biblioteka sadrži implementirane optimizacijske algoritme, regularizacije te brojne metrike za procjenu kvalitete modela.

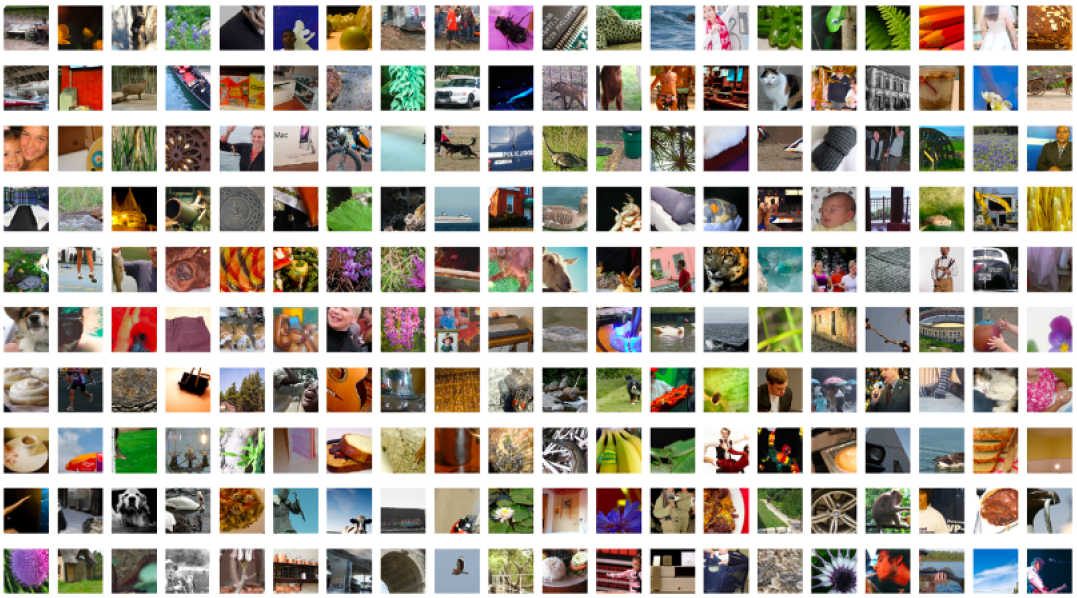
4.4. Programska izvedba

Eksperimenti su izvršavani, a implementacija eksperimenata je pisana na Google Colabu koji je omogućio korištenje grafičke kartice NVIDIA Tesla P100. Prvi od dva provođena eksperiment je bilo učenje i evaluacija SwiftNet modela. Kako bi se samo učenje uopće mogao pokrenuti najprije je trebalo prilagoditi originalni programski kod. Pošto implementacija matrice zabune preko Cythona nije bila podržana, zamijenjena je funkcijom $confusion\_matrix()$ podržanoj u biblioteci Scikit-learn. Kako bi se moglo započeti učenje potrebno je bilo dohvatiti skup podataka. Model je učen na slikama formata ppm, a slike iz skupa CityScapes su originalno png formata. Slike sam pretvorio iz png formata u ppm. Tu je došlo do problema s memorijskim ograničenjima Google Colaba. CityScapes je vrlo velik skup podataka i Colab ne može pohraniti istovremeno cijeli skup png i ppm formata. Svaka pojedinačna slika formata png zauzima oko 2.3MB dok slika ppm formata zauzima oko 6MB što jako povećava memorijske zahtjeve. Taj problem je riješen tako da se svaki od tri podskupa: train, val, test posebno pretvori u željeni format, a zatim obrišu sve png slike iz tog podskupa kako bi se oslobodio dio memorije. Nakon pretvorbe slika učenje može započeti. Zbog ranije spomenute veličine skupa podataka za učenje je dugo trajao što je dovelo do problema s vremenskim ograničenjima Colaba te se nije cijelo učenje moglo izvršiti u jednom pokretanju. Problem je riješen tako da je model učen do prekida tada bi se spremile težine te bi se učenje ponovno pokretalo od epohe na kojoj je stao s učenjem. Nakon brojnih pokušaja postignuti su rezultati koji se mogu vidjeti u 7. poglavlju. Drugi eksperiment koji se provodio je učenje i evaluacija na skupu podataka CamVid. Ovaj skup podataka je znatno manji od skupa CityScapes i po broju slika i po njihovoj veličini. Uz to model je učen na slikama formata png, a ne ppm kao što je u prvom eksperimentu bio slučaj. Radi manje veličine skupa podataka nije došlo do problema sa memorijskim ograničenjima niti s vremenskim. Nakon učenja spremljene su težine, a potom je provedena evaluacija na skupu za učenje i validacijskom skupu. Tijekom evaluacije pohranjene su ulazne slike, točne oznake i predikcije radi vizualizacije dobivenih rezultata.

5. Skupovi izvornih slika

5.1. ImageNet

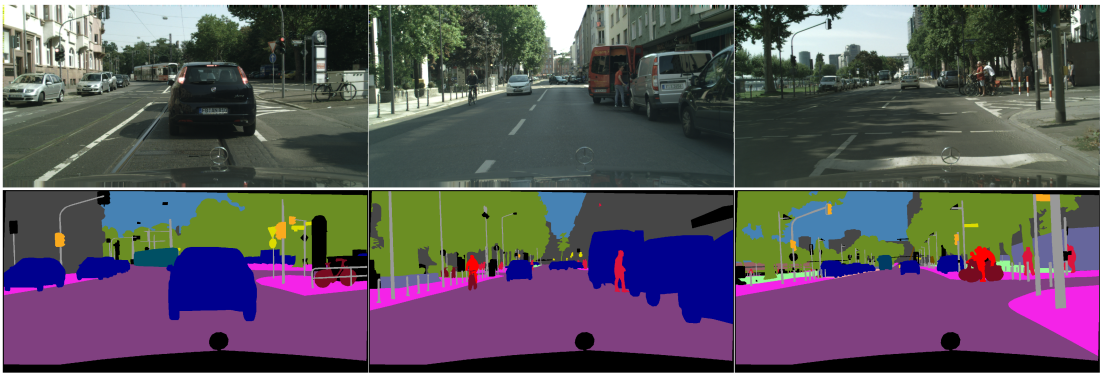
ImageNet je velika baza podataka koja sadrži mnogo slikovnih podataka koji se koriste za znanstvena istraživanja i edukaciju u području računalnog vida. Sastoji se od 14 milijuna slika koje su ručno označene. Te slike su raspoređene u više od 20 000 klasa. Neke od klasa su: auto, kuća, riba, drvo. U ovom radu korištene su dostupne težine za ResNet koje su predtrenirane na ImageNetu [4].



Slika 5.1: Prikazana je raznovrsnost slikovnih podataka skupa ImageNet [5].

5.2. Cityscapes

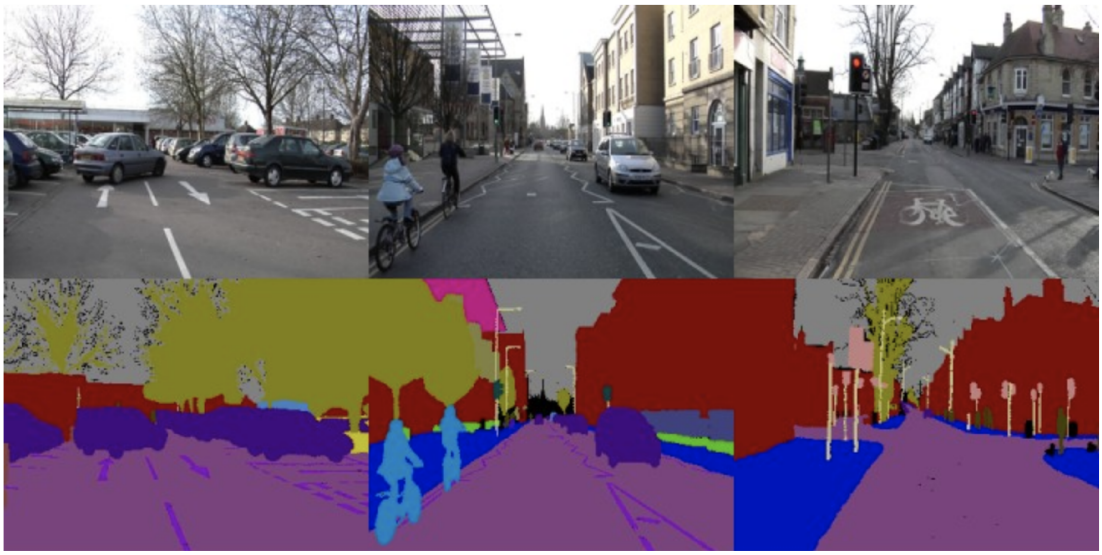
Cityscapes je skup podataka u kojem se nalaze slike snimljene iz auta u pokretu. Uglavnom su slike iz Njemačkih gradova i iz gradova u susjednim državama. Slike su slikane u različitim godišnjim dobima, ali ne i za vrijeme jakih padalina kao što su kiša i snijeg jer autori ovog skupa podataka smatraju da su za takve vremenske uvjete potrebne posebne tehnike i drugačiji skupovi podataka [3]. Slike su slikane tako da su svaka od 3 boje crvena, zelena i plava zapisane u 16 bitnom zapisu. Kasnije su autori objavili skup podataka sa slikama u 8 bitnim zapis kako bi ovaj skup bio usporedi sa već postojećim skupovima. U ovom skupu podataka nalazi se 5000 slika koje su ručno odabrane iz videa koje su bile slikane u 27 različitih gradova. Tih 5000 slika je precizno označeno. Slike iz ostala 23 grada su odabrane tako da svakih 20 sekundi ili svakih 20 metara se slika jedna slika za skup. Te slike nisu precizno označene kao one iz prvih 27 gradova. Dolazimo do ukupnog broja od 20000 slika koje čine grubi skup. U ovom radu koristio sam samo precizno označenih 5000 slika koje su podijeljene na skup za učenje u kojem je 2975 slika, skup za validaciju koji sadrži 500 slika i ostalih 1525 slika čini skup za testiranje. Na slikama iz ovog skupa označeno je 19 klasa [3].



Slika 5.2: Primjeri iz CityScapes skupa podataka [3].

5.3. CamVid

CamVid ili punim nazivom The Cambridge-driving Labeled Video Database je prvi skup videozapisa označenih za semantičku segmentaciju. Svakom pikselu dodjeljuje se jedna od 32 semantičke klase. CamVid se koristi za vrednovanje novih algoritama u režimu male količine podataka za učenje. Videozapisi su snimljeni iz perspektive vozača u vožnji. Skup zadrži preko 10 minuta videozapisa visoke kvalitete od 30Hz. Slike su uzete iz videa na 1Hz i 15Hz te su označene. Skup podataka doprinosi 4 relevantna djela za istraživanja analize objekata. Prvi od njih je semantička segmentacija po svakom pikselu za preko 700 ručno označenih slika, a te slike je pregledala druga osoba radi bolje preciznosti. Idući doprinos su videi u boji visoke rezolucije i kvalitete te dugog trajanja posebno zanimljivi onima koje zanimaju scenariji vožnje. Treće, snimljene su intrinzične kalibracijske sekvence za reakciju kamere na boje uz to izračunata je i 3D položaj camere za svaku sliku u nizu. Posljednji doprinos je software za označavanje videa u svrhu proširenja ili izradu vlastitog skupa podataka [2]. U radu se koriste slike iz CamVid skupa podataka koji se sastoji od 12 klasa uključujući i klasu nedefinirano. Slike su dimenzija 480×380.



Slika 5.3: Primjeri iz CamVid skupa podataka [2].

6. Eksperimentalni rezultati

Eksperimenti u ovome radu su trenirani i evaluirani na dva skupa podataka. To su Cityscapes i CamVid skupovi. U nastavku rada prikazani su rezultati postignuti na trening i evaluacijskim skupovima. Prikazane su IoU mjere točnosti po klasama, mIoU mjera, odziv, preciznost i točnost piksela. SwiftNet model je treniran i testiran na grafičkoj kartici NVIDIA Tesla P100.

6.1. Detalji treninga

Modeli su trenirani na skupu podataka Cityscapes i CamVid. Model treniran na Cityscapes skupu treniran je kao i u originalnom radu [11]. Treniran je optimizacijskim algoritmom Adam s početnom stopom učenja $4 \cdot 10^{-4}$ koja se smanjuje kosinusnim kaljenjem do stope učenja $1 \cdot 10^{-6}$ u posljednjoj epohi. Propadanje težina je postavljeno na $1 \cdot 10^{-4}$. Model je treniran 250 epoha. U drugom slučaju kada je model treniran na skupu podataka CamVid povećan je broj epoha na 400 i povećana je početna stopa učenja u odnosu na trening na skupu Cityscapes i iznosila je na $4 \cdot 10^{-3}$, a krajnja je postavljena na $1 \cdot 10^{-5}$. Ostali parametri kao i optimizacijski algoritam te kosinusno kaljenje su korišteni kao i u originalnom radu.

6.2. Rezultati na skupu Cityscapes

Iz tablici 6.1 je vidljiva IoU točnost po klasama. Može se vidjeti kako su modelu najteže klase bile wall (52.33%) i fence (58.38%). Najbolji rezultati su postignuti na klasama road, sky, vegetation i car. Na svakoj od tih klasa je IoU točnost je bila preko 90%. Zaključujemo da objekti velike površine koji se često pojavljuju u skupu za treniranje su najbolje segmentirani. Klase wall i fence su veliki objekti, ali se puno rjeđe pojavljuju od ranije navedenih klasa koje su postigle najbolje rezultate. Ove dvije klase vrlo su često prikrivene drugim objektima što dodatno otežava njihovo segmentiranje.

Intersection over Union	%
road	97.87
sidewalk	83.05
building	92.32
wall	52.33
fence	58.38
pole	65.07
traffic light	70.41
traffic sign	78.72
vegetation	92.28
terrain	60.99
sky	94.88
person	82.00
rider	62.38
car	94.78
truck	74.34
bus	83.82
train	65.45
motorcycle	61.52
bicycle	76.72

Tablica 6.1: Prikazani su IoU rezultati po klasama na vaidacijskom skupu podataka.

Ponovnim treniranjem modela postignuti su slični rezultati kao i u radu [11]. U tablici 6.2 su prikazane mjere mIoU, odziv, preciznost i točnost piksela. To su vrlo dobri rezultati dobiveni su na slikama dimenzija 2048×1024. Tako velike dimenzije

slike omogućavaju veći broj piksela na kojima se model trenira što pridonosi kvaliteti modela. Na slici 6.1 mogu se vidjeti primjeri kako model radi na primjerima iz skupa za evaluaciju. S lijeve strane su prikazani točno označeni podaci na ulaznoj slici a na desnoj strani možemo vidjeti kako je treniran model izvršio predikciju.

mean IoU class accuracy	76.17 %
mean class recall	83.59 %
mean class precision	88.53 %
pixel accuracy	95.86 %

Tablica 6.2: Prikaz rezultata po mjerama točnosti na validacijskom podskupu Cityscapesa.

U tablicama 6.3 i 6.4 prikazani su podaci koji vrijede za skup za treniranje CityScapes skupa podataka. U prvoj tablici možemo vidjeti IoU mjeru točnosti po klasama, a u drugoj mIoU, odziv, preciznost i točnosti piksela na trening skupu.

Intersection over Union	%
road	99.08
sidewalk	93.51
building	95.62
wall	88.27
fence	86.09
pole	70.43
traffic light	74.24
traffic sign	83.54
vegetation	94.61
terrain	87.59
sky	96.21
person	87.89
rider	77.27
car	96.57
truck	94.18
bus	95.13
train	95.26
motorcycle	82.13
bicycle	81.11

Tablica 6.3: Prikazani su IoU rezultati po klasama na podskupu za učenje Cityscapes skupa podataka.

mean IoU class accuracy	87.85 %
mean class recall	93.35 %
mean class precision	93.29 %
pixel accuracy	97.75 %

Tablica 6.4: Prikaz rezultata po mjerama točnosti na podskupu za učenje iz skupa podataka Cityscapes.

Slika 6.1: Na slikama su s lijeve strane prikazani točno označeni podaci preko originalne slike, a s desne strane su vizualizirane predikcije na isti način. Sve slike su iz podskupa za validaciju skupa podataka Cityscapes.

6.3. Rezultati na skupu CamVid

Kao i u prethodnom odjeljku najprije je prikazana tablica IoU točnosti po klasama na validacijskom skupu. U tablici se može vidjeti kako najlošijiji rezultati su postignuti na klasama tree (22.50%) i bicycle (29.83%). Najbolji rezultati su postignuti za sky (96.85%) i sign (91.41%). Postignuti rezultati su nešto lošiji nego oni potignuti na skupu CityScapes što je i očekivano zbog toga što je skup znatno manji. Isto tako dimenzije slike su manje što znači da je puno manje piksela na kojima se model uči.

Intersection over Union	%
building	85.56
tree	22.50
sky	96.85
car	86.94
sign	91.41
road	63.17
pedestrian light	68.51
fence	80.87
column pole	63.10
sidewalk	83.48
bicyclist	29.83

Tablica 6.5: Prikazani su IoU rezultati po klasama na skupu za evaluaciju skupa podataka CamVid.

mean IoU class accuracy	70.20 %
mean class recall	80.76 %
mean class precision	80.35 %
pixel accuracy	92.78 %

Tablica 6.6: Prikaz rezultata po mjerama točnosti na evaluacijskim skupovima podataka iz skupa CamVid

U tablicama 6.7 i 6.8 prikazani su podaci dobiveni na podskupu za učenje CamVid skupa podataka. U tablici 6.7 su prikazane IoU vrijednosti po klasama, a u tablici 6.8 dobiveni rezultati po već ranije korištenim mjerama točnosti. Ispod tablica može se vidjeti slika 6.2 gdje su s lijeve strane prikazani točno označeni podaci, a na desnoj strani predikcija modela na evaluacijskom skupu.

Intersection over Union	%
building	92.77
tree	53.64
sky	97.86
car	86.40
sign	90.04
road	80.75
pedestrian light	85.47
fence	93.50
column pole	70.31
sidewalk	82.19
bicyclist	70.95

Tablica 6.7: Prikazani su IoU rezultati po klasama na skupu za validaciju skupa podataka CamVid.

IoU mean class accuracy	82.19 %
mean class recall	89.46 %
mean class precision	90.01 %
pixel accuracy	95.75 %

Tablica 6.8: Prikaz rezultata po mjeram točnosti na podskupu za učenje skupa podataka CamVid.

Slika 6.2: Na slikama su s lijeve strane prikazani točno označeni podaci preko originalne slike, a s desne strane su vizualizirane predikcije na isti način. Sve slike su iz validacijskog podskupa skupa podataka CamVid.

7. Zaključak

U ovom radu evaluirali smo model za semantičku segmentaciju u stvarnom vremenu SwiftNet. Radi boljeg razumijevanja rada na njegovom početku objašnjene su osnove dubokih neuronskih mreža kao i konvolucijskih mreža. Konvolucijske mreže su pridonijele smanjenju računalne složenosti pa tako i brzini izvođenja. Objašnjene su i mjere kojima je izmjerena kvaliteta modela. U četvrtom poglavlju je predstavljena arhitektura SwiftNet modela. Arhitektura se zasniva na koderu koji je predtreniran na ImageNet skupu podataka. Zbog korištenja predtreniranih težina na ImageNetu postiže se veća točnost modela. Važnu ulogu ima propagiranje gradijenata koje se postiže ResNet arhitekturom. Ta arhitektura je dobar odabir zbog svoje male složenosti koja omogućava izvršava rad u stvarnom vremenu. Koder SwiftNet modela sadrži ResNet arhitekturu. Zadaća kodera je pretvoriti ulaznu sliku u bogate semantičke mape značajki. Nakon kodera slijedi dekoder čija je uloga povećanje dimenzija mape značajki. Brzini modela pridonosi jednostavnost metode kojom se povećavaju dimenzije mape značajki na dimenzije ulazne slike. Ta metoda se naziva bilinearna interpolacija. U radu je korišten modul za povećanje receptivnog polja piramidalna fuzija (engl. *pyramid fusion*). Ovaj modul uz povećanje receptivnog polja ne ugrožava mogućnost izvršavanja u stvarnom vremenu. Nakon prilagodbe programskog koda modela i skupova podataka mogli smo započeti treniranje model. Model je treniran na 2 skupa podataka to su: Cityscapes i CamVid. Na skupu Cityscapes postignuta je mIoU točnost od 76.17% i točnosti piksela od 95.86% na validacijskom skupu. To su očekivano bolji rezultati nego oni dobiveni na CamVid skupu podataka zbog veličine skupa podataka i kvalitete slika. Na CamVid validacijskom skupu je postignuta mIoU točnost od 70.20% i točnosti piksela od 92.78%. To su vrlo dobri rezultati u odnosu na ostale modele semantičke segmentacije kojima je cilj izvršavanje u stvarnom vremenu.

LITERATURA

[1] Bojan Dalbelo Bašić, Marko Čupić, i Jan Šnajder. Umjetne neuronske mreže, 2008.

[2] Gabriel J. Brostow, Julien Fauqueur, i Roberto Cipolla. Semantic object classes in video: A high-definition ground truth database. *Pattern Recognition Letters*, 2008.

[3] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, i Bernt Schiele. The cityscapes dataset for semantic urban scene understanding, 2016.

[4] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, i L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database, 2009.

[5] Martins Frolovs. How to create and use custom pytorch dataset from the imagenet. https://mf1024.github.io/2019/06/22/Create-Pytorch-Datasets-and-Dataloaders/, Jun 2019.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, i Jian Sun. Deep residual learning for image recognition, 2015.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, i Jian Sun. Identity mappings in deep residual networks, 2016.

[8] Diederik P. Kingma i Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[9] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, i Tom Goldstein. Visualizing the loss landscape of neural nets. U S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, i R. Garnett, urednici, *Advances in Neural Information Processing Systems 31*, stranice 6389–6399. Curran Associates, Inc., 2018. URL `http://papers.nips.cc/paper/`

7875-visualizing-the-loss-landscape-of-neural-nets.pdf.

[10] Siniša Šegvić Marin Oršić. Efficient semantic segmentation with pyramidal fusion, 2020.

[11] Marin Oršić, Ivan Krešo, Petra Bevandić, i Siniša Šegvić. In defense of pre-trained imagenet architectures for real-time semantic segmentation of road-driving images, 2019.

[12] Josp Krapac Siniša Šegvić, Marko Čupić. Duboko učenje. 2017.

[13] Wikipedia. Plagiarism — Wikipedia, the free encyclopedia, 2020. URL https://en.wikipedia.org/wiki/Bilinear_interpolation.

[14] Mingmin Zhen, Jinglu Wang, Lei Zhou, Tian Fang, i Long Quan. Learning fully dense neural networks for image semantic segmentation. 2019.

Vrednovanje modela SwiftNet za semantičku segmentaciju

Sažetak

Sažetak na hrvatskom jeziku.

Ključne riječi: Ključne riječi, odvojene zarezima.

Title

Abstract

Abstract.

Keywords: Keywords.