SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 7203

SEMANTIČKA SEGMENTACIJA S DVOSMJERNOM PIRAMIDOM ZNAČAJKI

Jakov Rukavina

Zagreb, lipanj 2021.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 7203

SEMANTIČKA SEGMENTACIJA S DVOSMJERNOM PIRAMIDOM ZNAČAJKI

Jakov Rukavina

Zagreb, lipanj 2021.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 12. ožujka 2021.

ZAVRŠNI ZADATAK br. 7203

Pristupnik:	**Jakov Rukavina (0036509435)**
Studij:	Računarstvo
Modul:	Računarska znanost
Mentor:	prof. dr. sc. Siniša Šegvić

Zadatak: **Semantička segmentacija s dvosmjernom piramidom značajki**

Opis zadatka:

Semantička segmentacija važan je zadatak računalnog vida s mnogim zanimljivim primjenama. U posljednje vrijeme vrlo zanimljive rezultate postižu konvolucijski modeli koji dio zaključivanja provode na poduzorkovanoj reprezentaciji. Nedavno se pojavio postupak koji taj pristup obogaćuje informacijskim putom na visokoj rezoluciji pri čemu su visokorezolucijske i poduzorkovane reprezentacije povezane dvosmjernim vezama. U okviru rada, potrebno je odabrati okvir za automatsku diferencijaciju te upoznati biblioteke za rukovanje matricama i slikama. Proučiti i ukratko opisati postojeće pristupe za klasifikaciju slike. Odabrati slobodno dostupni skup slika te oblikovati podskupove za učenje, validaciju i testiranje. Implementirati najmanje složenu inačicu razmatrane arhitekture. Uhodati postupke učenja modela i validiranja hiperparametara. Primijeniti naučene modele te prikazati i ocijeniti postignutu točnost. Radu priložiti izvorni i izvršni kod razvijenih postupaka, ispitne slijedove i rezultate, uz potrebna objašnjenja i dokumentaciju. Citirati korištenu literaturu i navesti dobivenu pomoć.

Rok za predaju rada: 11. lipnja 2021.

Zahvaljujem se mentoru prof. dr. sc. Siniši Šegviću na svoj pomoći, smjernicama i objašnjenjima te na uloženom trudu i vremenu. Također veliko hvala i Filipu Oreču na savjetima i pomoći.

Sadržaj

Uvod

Računalni vid grana je umjetne inteligencije koja se bavi analizom i razumijevanjem digitalnih slika. Ljudi od pojave računala teže simuliranju ljudske inteligencije, a računalni vid nastoji simulirati glavno ljudsko osjetilo – oko.

Zahvaljujući nedavnim napredcima vezanim uz duboke neuronske mreže računalni vid postaje sve popularniji te nam omogućuje rješavanje mnogih zadataka koji prije nisu bili rješivi računalima. Nadalje, nedavno sve popularnija postaje i semantička segmentacija slika kao jedan od važnih zadataka računalnog vida. Segmentacijom određujemo točne granice objekata na slici te klasificiramo iste. Ovaj se postupak može koristiti za razne primjene poput zamućivanja ili mijenjanja pozadine na slikama osoba, virtualnog isprobavanja odjeće ili analize satelitskih snimaka. Jedna primjena će nas posebno zanimati zbog nedavnog razvoja autonomnih vozila – analiza cestovnih scena iz perspektive vozila.

Ovaj rad razmatra modele s dvosmjernom piramidom značajki. Arhitektura BPNet koristi se piramidom značajki za efikasnije otkrivanje objekata različitih veličina. BPNet se ističe jer ne koristi predtrenirani model za izvlačenje značajki te sadrži dvosmjerne tokove informacija između poduzorkovanih značajki s većom semantičkom vrijednošću te visokorezolucijskih značajki manje semantičke vrijednosti.

Rad će se fokusirati na ispitivanje modela BPNet-S3 koji je fokusiran na segmentaciju u realnom vremenu, a pri tome ne gubi puno na točnosti. Ispitivanje je izvedeno na skupu podataka Cityscapes koji se sastoji od 5,000 slika cestovnih scena.

1. Duboko učenje

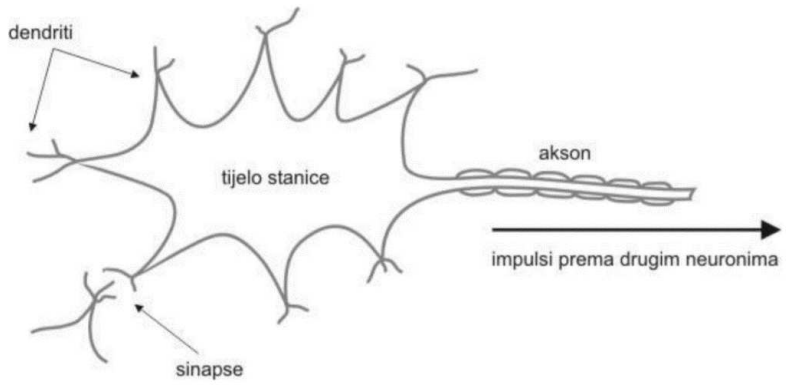
Duboko učenje (eng. *deep learning*) grana je strojnog učenja (eng. *machine learning*) temeljena na neuronskim mrežama (eng. *artificial neural netwoks*) s velikim brojem skrivenih slojeva. Ovakve mreže često se koriste za obradu velikih i kompleksnih skupova podataka.

Duboko učenje obično koristimo pri rješavanju UI-potpunih problema, tj. problema koje nije moguće riješiti na klasičan algoritamski način. Neke od takvih primjena ove metode uključuju: računalni vid, raspoznavanje govora, obradu prirodnog jezika, otkrivanje lijekova, analizu medicinskih slika i mnoge druge.

1.1. Umjetni neuron

Inspiracija za umjetne neuronske mreže dolazi od biološke građe ljudskog mozga, a osnovna funkcionalnost takvih mreža temelji se na biološkom neuronu kojih u mozgu ima oko $8.6 \cdot 10^{10}$. Svaki takav neuron povezan je sa 1,000 do 10,000 drugih neurona. Informacije se prenose jednosmjerno između neurona te obrađuju serijski i paralelno

Biološki neuron sastoji se od tijela, dendrita, aksona i završnih članaka. Iz računarske perspektive mogli bismo pojednostavljeno reći da dendriti služe kao ulaz u neuron, zatim da tijelo neurona vrši obradu ulaznih signala, dok se akson koristi za prijenos obrađenih signala, a završni članci su izlazni dio stanice koji se spajaju na ulaze drugih neurona.



Slika 1.1: Građa biološkog neurona. Neuron se sastoji od tijela, dendrita, aksona i završnih članaka. Završni članci nalazili bi se na samom desnom kraju slike odnosno na završetku aksona. Slika je preuzeta iz [1].

Funkcionalnost biološkog neurona imitira McCulloch-Pitts model umjetnog neurona osmišljen 1943. godine, tzv. *Treshold Logic Unit* (TLU). Ovakav model zamjenjuje biološke signale numeričkim vrijednostima. Te vrijednosti se na ulazu u neuron množe težinskim faktorom (eng. *weight*) koji opisuje jakost veze između dva neurona. Tako dobivene vrijednosti se tada sumiraju što je analogno sumiranju potencijala u tijelu stanice te se sumi pribraja tzv. prag ili pomak (eng. *bias*) [1]. Takva težinska suma zadana je izrazom 1.1 te ćemo ju označiti s net.

$$net = \sum_{i=1}^{n} \omega_i x_i + b \tag{1.1}$$

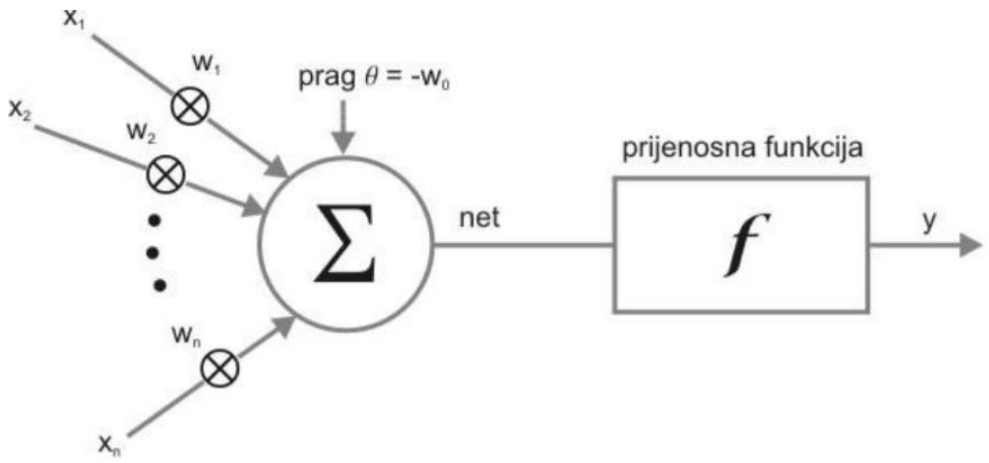
Ulazne signale neurona s n ulaza označavamo sa $x_1, x_2, \ldots x_n$, a težinske faktore sa $\omega_1, \omega_2, \ldots \omega_n$ te ćemo prag označiti s b. Treba napomenuti da obično ove vrijednosti zapisujemo vektorski odnosno matrično radi kompaktnosti te efikasnosti računalnih operacija s ovakvom vrstom zapisa. Također, nekad ćemo iz istih ovih razloga prag zapisati kao nultu težinu ω_0 u sumi te dogovorno dodati nulti fiksirani ulaz $x_0 = 1$ kao u izrazu 1.2.

$$net = \omega_0 x_0 + \omega_1 x_1 + \cdots + \omega_n x_n = \sum_{i=0}^{n} \omega_i x_i \tag{1.2}$$

Konačno, rezultat se propušta kroz prijenosnu funkciju (eng. *activation function*) koja nam omogućuje da na različite načine mijenjamo izlaz neurona y kao što je vidljivo u izrazu 1.3. Povijesno su se koristile razne prijenosne funkcije, a jedan od najpoznatijih primjera je sigmoidalna funkcija.

$$y = f\left(\sum_{i=0}^{n} \omega_i x_i\right) = f(net) \tag{1.3}$$

Umjetni neuron vizualiziran je slikom 1.2.



Slika 1.2: Vizualizacija umjetnog neurona. Ulaz x_i množi se s težinom ω_i, svi takvi umnošci se sumiraju te im se dodaje prag ω_0. Ovakva suma predstavlja ulaz prijenosne funkcije dok njen izlaz predstavlja izlaz neurona u cjelini. Slika je preuzeta iz [1].

1.2. Umjetne neuronske mreže

Umjetne neuronske mreže prvotno su težile repliciranju ljudskog mozga te su nastojale simulirati postupak učenja [1]. Ovakvi su modeli zapravo pojednostavljenje ljudskog mozga te mnoge karakteristike umjetnih neuronskih mreže odstupaju od stvarnih mehanizama našeg mozga. No ovakva odstupanja ne znače da ovaj model nije efektivan, upravo naprotiv.

Neuronske mreže sastoje se od međusobno povezanih elemenata koji se temelje na biološkom neuronu. Takvi su elementi funkcionalno vrlo jednostavni, no zajedno tvore kompleksnu cjelinu. Njihova je sposobnost obrade sadržana u snazi veza između umjetnih neurona tj. u već ranije objašnjenim težinama koje se prilagođavaju problemu kojeg pokušavamo riješiti prilikom postupka učenja. Učenje neuronskih mreža temelji se na izračunu svih parcijalnih derivacija i njihovoj primjeni na određivanje iznosa kojim korigiramo svaku od težina. Ovaj postupa zove se propagacija pogreške unatrag (eng. Error Backpropagation) [7].

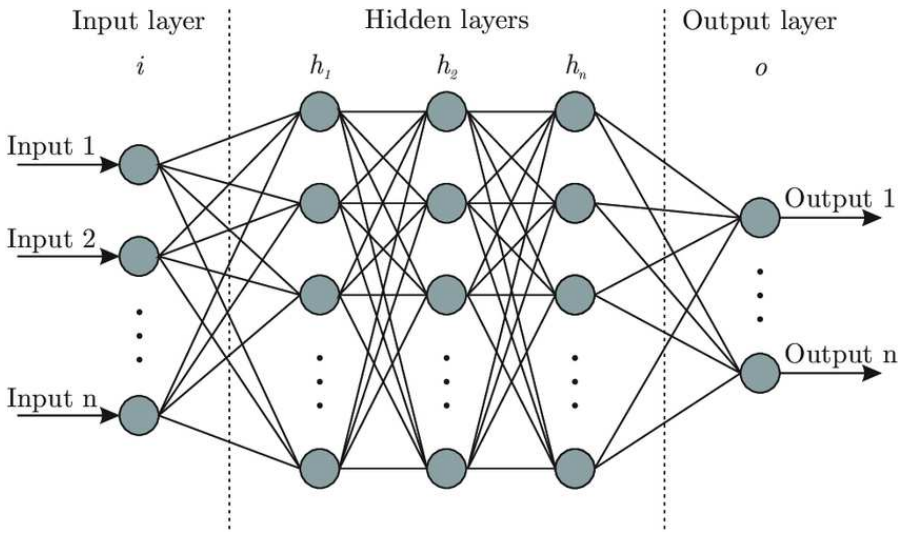
Arhitekturu mreže određuje način na koji se neuroni međusobno povezuju [1].

- Unaprijedna neuronska mreža (eng. *feedforward network*) - nema povratnih veza između neurona što znači da se signali propagiraju jednosmjerno od ulaza prema izlazu. Iz tog razloga takve arhitekture još zovemo i acikličkim mrežama. Kod

ovakve vrste mreža razlikujemo ulazni, izlazni i skriveni sloj neurona. Neuroni ulaznog sloja nemaju funkcionalnost drugih neurona, već oni samo prosljeđuju ulazne vrijednosti. Stoga njih možemo smatrati osjetilnim odnosno senzornim neuronima.

- Neuronske mreže s povratnom vezom (eng. *recurrent network*) - mreže koje sadrže barem jednu petlju odnosno povratnu vezu. Kod takvih mreža umjesto ulaznih i izlaznih slojeva imamo vidljive slojeve te već ranije spomenute skrivene slojeve.
- Rešetkaste odnosno lateralno povezane mreže – mreže koje sadrže veze između neurona u istom sloju
- Hibridne mreže

U dubokom učenju koristimo se dubokim neuronskim mrežama. Da bi se neuronska mreža smatrala dubokom ona mora imati barem dva skrivena sloja, no u praksi se obično susrećemo s mrežama koje u sebi sadrže mnogo veći broj skrivenih slojeva. Dubinu takvih mreža računa se kao zbroj broja skrivenih slojeva plus izlazni sloj [3]. Za mrežu na slici 1.3 reći ćemo da ima n + 1 slojeva, gdje n označava broj skrivenih slojeva.



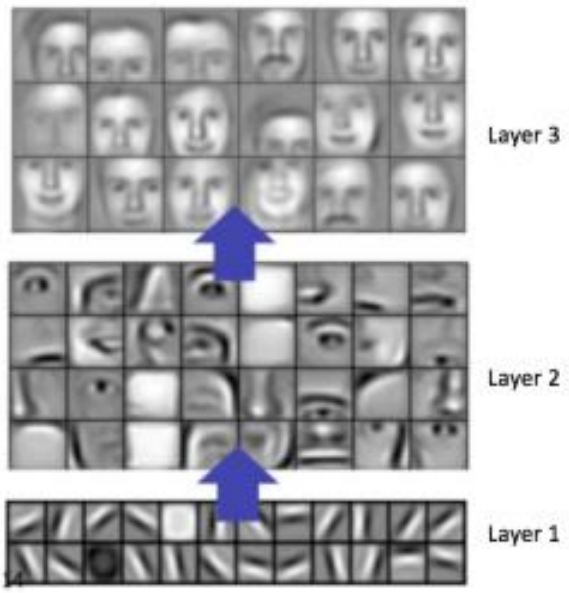
Slika 1.3: Unaprijedna potpuno povezana neuronska mreža s više skrivenih slojeva. Slika je preuzeta iz [11].

Neuronske mreže za klasifikaciju slika algoritmi su strojnog učenja kod kojih je model proizvoljna jednom diferencijabilna funkcija ulaza. Gubitak se dobiva kao negativna log-izglednost, a optimizacijski postupak je neki od oblika gradijentnog spusta [13].

1.3. Konvolucijske mreže

Konvolucijske neuronske mreže (eng. Convolutional Neural Networks) dizajnirane su za rad sa podatcima koji imaju pravilnu topološku strukturu. Jedna od takvih primjena je prepoznavanje slika te su prvotno konvolucijske neuronske mreže bile primijenjene na prepoznavanje rukom napisanih znamenaka. Osnovni cilj CNN-a bio je kreirati mrežu u kojoj će neuroni u početnim slojevima mreže „izvlačiti" lokalne vizualne značajke kako bi ih neuroni u kasnijim slojevima mogli kombinirati u kompleksnije značajke [3]. Kako bi se osigurala takva lokalnost aktivacije kovolucijskog modela ovise o malom broju aktivacija iz prethodnog sloja koje su po udaljenosti bliske trenutnoj. Nadalje, sve se aktivacije sloja računaju na isti način i dijele slobodne parametre. Ovakva struktura modela osigurava kovarijantnost na translaciju; translatirani ulaz rezultirat će translatiranim aktivacijama.

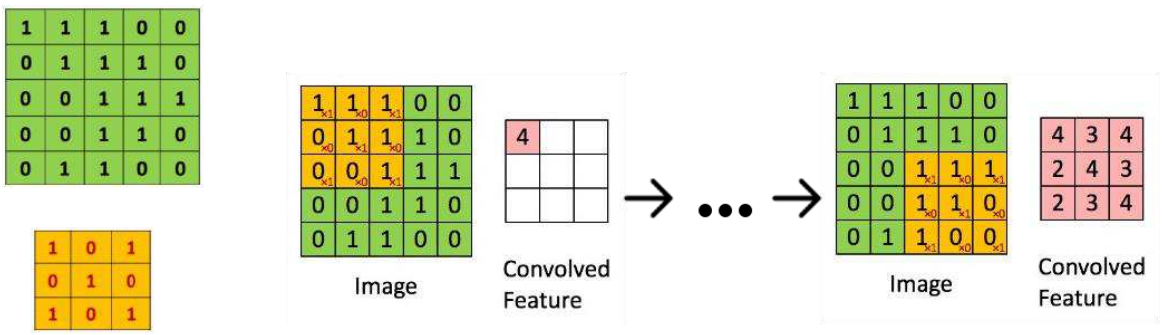
Na primjer, primijenimo li CNN na problem prepoznavanja lica, neuroni početnih slojeva aktivirat će se na jednostavne značajke poput linija pod određenim kutom ili dijela krivulje, dok će neuroni dublje u mreži kombinirati te aktivacije u kompleksnije značajke poput usta, nosa ili očiju. Konačno, neuroni u posljednjim slojevima mreže uspjet će prepoznati cijelo lice na slici kombinirajući aktivacije koje predstavljaju određene dijelove lica [3]. Ovakav hijerarhijski ustroj značajki prikazan je na slici 1.4.



Slika 1.4: Idealizirani prikaz hijerarhijskog ustroja konvolucijskih filtara od najjednostavnijih u prvom sloju do najkompleksnijih u posljednjem sloju. U stvarnosti bi razlike u kompleksnosti filtara susjednih slojeva bile puno manje. Drugim riječima, bilo bi potrebno mnogo više slojeva kako bi došli do vrlo kompleksnih filtara poput onih u sloju 3 na slici. Slika je preuzeta iz [5].

Slika 1.5 vizualno opisuje operaciju konvolucije. Pretpostavimo da na ulaz CNN-a dovodimo slike zapisane u matričnom obliku. U ovom će slučaju radi jednostavnosti to biti crno-bijele slike čiji elementi matrice mogu biti 0 (crna) ili 1 (bijela). Matrica dimenzija 5×5 obojena zelenom bojom predstavlja upravo jednu takvu sliku na kojoj ćemo provesti konvoluciju. Žutu matricu dimenzija 3×3 zovemo jezgrom (eng. kernel, convolution mask). To je obično kvadratna matrica težina neparnih dimenzija 3×3, 5×5 ili 7×7 kojom ćemo obraditi našu ulaznu sliku pomičući jezgru po cijeloj slici za vrijednost pomaka (eng. stride). Npr. ako koristimo pomak od jednog piksela počet ćemo postavljajući jezgru u gornji lijevi kut slike. Skalarno ćemo pomnožiti svaki element jezgre sa svakim preklapajućim elementom slike te sve umnoške sumirati kako bismo dobili jednu vrijednost izlaza. Tada ćemo jezgru pomaknuti za jedan piksel udesno i ponoviti istu operaciju. To ponavljamo dok ne dođemo do desnog ruba slike što će se u našem slučaju dogoditi u sljedećoj iteraciji. Nakon što smo završili s obradom jednog retka pomičemo jezgru za jedan piksel prema dolje i ponovo započinjemo obradu novog retka s lijeva nadesno. Naposljetku, prelazeći jezgrom po svim mogućim pozicijama na slici dobivamo izlaz odnosno matricu značajki koja je na slici 1.5 obojena rozom bojom. Izlazna matrica smanjene je dimenzionalnosti što će ovisiti o pomaku te veličini same jezgre.

Treba primijetiti kako jezgra u danom trenutku „vidi“ samo dio ulazne matrice što uvelike smanjuje kompleksnost procesuiranja ovakvih slojeva u odnosu na potpuno povezane mreže zbog mnogo manjeg broja veza između slojeva. Ovakav će pristup također očuvati lokaciju značajki u odnosu na potpuno povezane neuronske mreže.



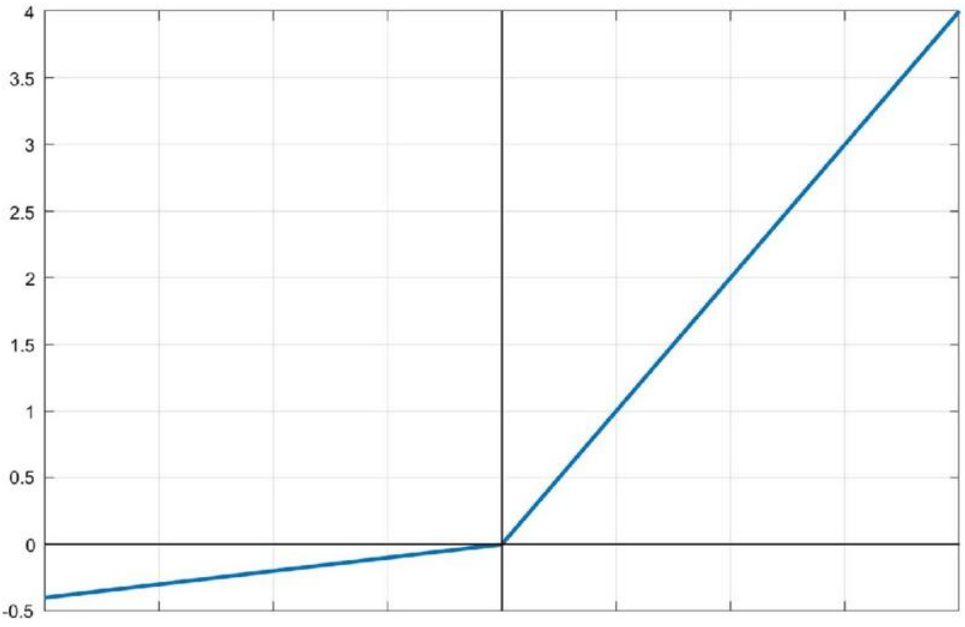
Slika 1.5: Zelena matrica predstavlja ulaz odnosno sliku, žuta matrica je jezgra, a roza matrica je matrica značajki. Slika je preuzeta iz [4].

Kao što je već spomenuto u poglavlju 1.1, na izlazu iz neurona nalazi se prijenosna funkcija pa ćemo tako i nakon konvolucije htjeti imati jednu takvu funkciju. U konvolucijskim mrežama najčešće u tu svrhu koristimo zglobnicu (eng. Rectified Linear Unit, ReLU). To je funkcija koja propušta sve pozitivne vrijednosti kakve jesu dok negativne vrijednosti guši. Opisana je izrazom 1.4.

$$f(x) = \max(0, x) \tag{1.4}$$

Razlog zašto koristimo zglobnicu je kako bismo uveli nelinearnost u našu mrežu, budući da je većina podataka iz stvarnog svijeta na kojima bismo htjeli učiti konvolucijske mreže nelinearna, a konvolucija uključuje samo linearne operacije – množenje i zbrajanje matrica [4]. Kada ne bi bilo nelinearnosti naš model ne bi mogao naučiti raspoznavati objekte iz stvarnog svijeta.

Kod obične se zglobnice može pojaviti problem stvaranja tzv. mrtvih neurona (eng. *dead neuron*) – neurona koji se nikada ne aktiviraju. Kako bismo riješili taj problem uvest ćemo malu preinaku u našoj prijenosnoj funkciji: umjesto da sve negativne vrijednosti množimo s 0, množit ćemo ih s odabranom vrlo malom konstantom npr. $\alpha = 0.01$. Takva funkcija prikazana je slikom 1.6, a zvat ćemo ju propusna zglobnica (eng. *Leaky Rectified Linear Unit*).

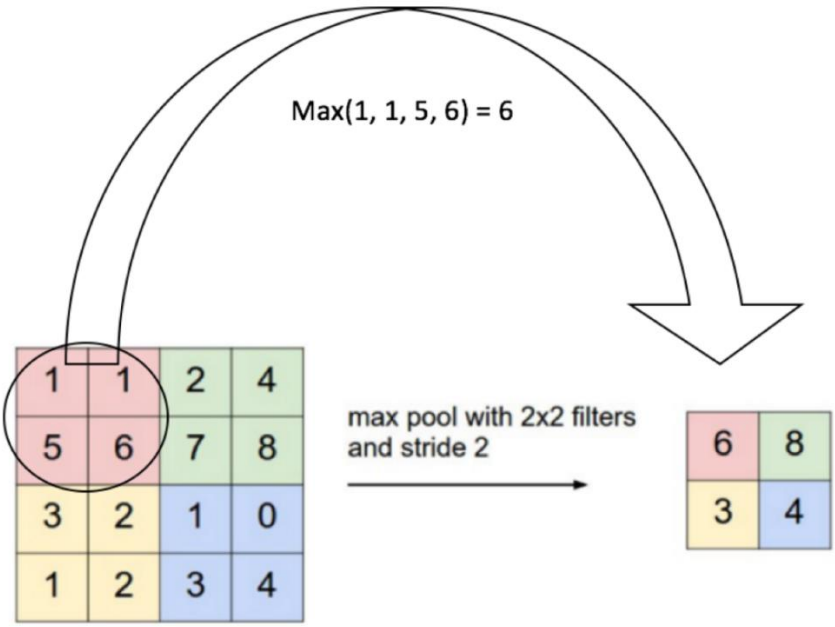


Slika 1.6: Propusna zglobnica (*Leaky ReLU*) uz koeficijent $\alpha = 0.01$. Slika je preuzeta iz [6].

Konačno, nakon konvolucije i prolaska kroz zglobnicu koristit ćemo još i sažimanje (eng. *Pooling*). Sažimanje smanjuje dimenzije mape značajki, ali pritom zadržava najvažnije informacije [4].

Postoje razne vrste sažimanja, a slika 1.7 prikazuje sažimanje uz odabir najveće vrijednosti (eng. *Max Pooling*). Ono funkcionira tako da pomičemo filter veličine npr. 2×2 po mapi značajki i uzimamo najveću vrijednost značajke unutar tog filtera. Pomicanje filtera se, za razliku od konvolucije, odvija bez preklapanja.

Umjesto uzimanja najveće vrijednosti mogli bismo uzeti prosječnu vrijednost (eng. *Average Pooling*) ili sumu svih značajki [4]. Ovakvo je sažimanje prvotno korišteno jer je smanjivalo broj parametara u mreži što je uvelike utjecalo na vrijeme procesuiranja te negiralo utjecaj malih nepravilnosti ili translacija u početnoj slici uz mnoge druge prednosti.

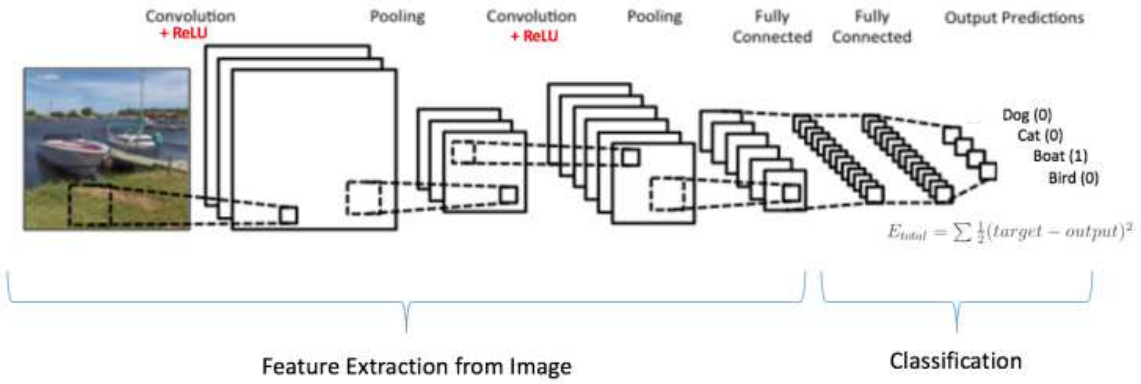


Slika 1.7: Sažimanje uz odabir najveće vrijednosti (eng. *Max Pooling*). Prikazano je sažimanje uz filtar veličine 2×2 na mapi značajki veličine 4×4. Time dobivamo novu mapu značajki upola manjih dimenzija u odnosu na početak. Slika je preuzeta iz [4].

Navedene tri uzastopne operacije: konvolucija, uvođenje nelinearnosti i sažimanje u prošlosti je činilo standardni konvolucijski sloj (eng. *convolution layer*), no danas se većina konvolucijskih slojeva ne sastoji od sažimanja, a neki niti od nelinearne aktivacije.

Konvolucijski će sloj otkriti značajke neovisno o njihovoj lokaciji na ulaznoj slici te će proizvesti mapu značajki manjih dimenzija u odnosu na početnu sliku [3][4].

Rani konvolucijski modeli su na kraju CNN-a koristili potpuno povezane slojeve. Svaki neuron potpuno povezanog sloja povezan je sa svim elementima na izlazu iz svakoga filtra odnosno konvolucijskog sloja [3]. Ovakav će pristup dobro funkcionirati primijeni li se na problem klasifikacije. Moderni konvolucijski modeli sadrže samo jedan potpuno povezani sloj.



Slika 1.8: Prikaz jednostavne konvolucijske mreže namijenjene klasifikaciji slika. Sastoji se od dva konvolucijska sloja koji se pak sastoje od operacija konvolucije, uvođenja nelinearnosti te sažimanja. Na kraju mreže nalaze se dva potpuno povezana sloja koji obavljaju zadatak klasifikacije. Slika je preuzeta iz [4].

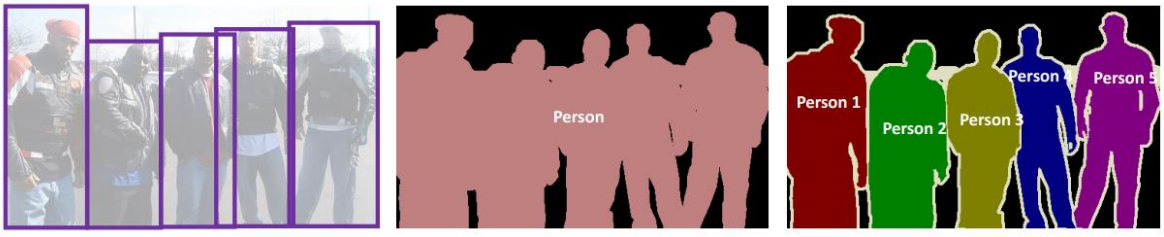
2. Semantička segmentacija

Duboko učenje je vrlo uspješno pri radu sa slikama, a neke zadaće čak rješava bolje od čovjeka. Najvažniji problemi koji se pokušavaju riješiti računalnim vidom uključuju klasifikaciju slika, detekciju objekata i segmentaciju [8].

Klasifikacijom slika (eng. *Image Classification*) želimo samo saznati što se nalazi na slici. Nadalje, kod detekcije objekata (eng. *Object Detection*) osim toga što se nalazi, želimo znati i gdje se ti objekti nalaze te ih onda označujemo pravokutnicima na slici. Segmentacija se bavi istim ovim problemom no uz razliku što ćemo ovdje htjeti odrediti točne granice između semantičkih razreda. To znači da ćemo određivati klasu svakome pikselu na slici pa segmentaciju možemo predstaviti i kao klasifikacijski problem po svakome pikselu [8].

Postoje tri vrste segmentacije:

- Semantička segmentacija (eng. *Semantic Segmentation*) – problem klasificiranja svakog piksela u određenu klasu, pritom ne razlikujući različite objekte iste klase. Npr. ako se na slici nalaze dvije osobe obje ćemo označiti klasom osoba umjesto da svaku od osoba podijelimo u zasebnu klasu.
- Segmentacija instanci, tj. objekata (eng. *Instance Segmentation*) – svaki objekt na slici dobiva različitu oznaku.
- Panoptička segmentacija (eng. *Panoptic Segmentation*) – kombinacija prethodna dva pristupa; svakom pikselu se dodijeljuje semantička oznaka i oznaka instance



Slika 2.1: Tri različita temeljna postupka u računalnom vidu. Lijevo – detekcija objekata, sredina – semantička segmentacija, desno – segmentacija instanci. Slika je preuzeta iz [12].

Najviše će nas zanimati segmentacija cestovnih scena koja je primjenjiva u razvoju autonomnih vozila. Vrlo je bitno točno segmentirati slike koje se koriste u ovu svrhu jer male

pogreške u prometu mogu ugroziti ljudske živote. Autonomno vozilo trebat će potpuno razumijevanje okoline u kojem se nalazi te će morati moći točno i brzo prepoznati opasnosti koje prijete. Primjer jedne takve segmentirane scene je prikazan slikom 2.2.

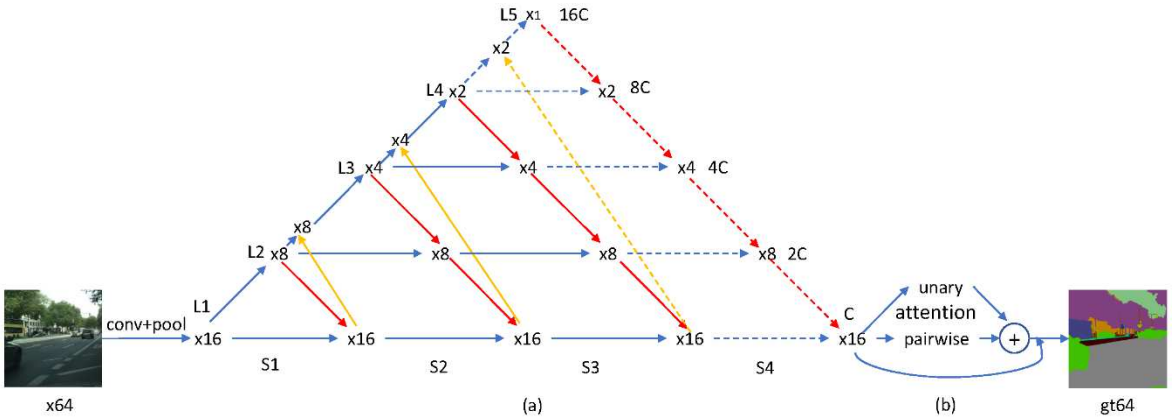
Slika 2.2: Segmentacija cestovnih scena. Boje su naknadno nadodane slici kako bi se lakše razlikovale klase objekata ljudski okom. Primjene ovog postupka uključuju sve popularniji razvoj autonomnih sustava vožnje. Slika je preuzeta iz [8].

Treba napomenuti kako se svi spomenuti postupci bave raspoznavanjem toga što se nalazi na slici. Računalni vid bavi se i 3D rekonstrukcijom (eng. *3D Reconstruction*) objekata i oblika no ovaj rad će se fokusirati samo na raspoznavanje, odnosno semantičku segmentaciju.

3. Model s dvosmjernom piramidom značajki

Piramidalne arhitekture često su korištene za prepoznavanje objekata različitih veličina. Dobile su naziv prema strukturi koja podsjeća na piramidu jer penjući se prema vrhu, rezolucija mapa značajki se smanjuje odnosno piramida se sužava. Pri tome se naravno povećava i semantička vrijednost značajki na višim razinama. U nastavku će se promatrati BPNet, arhitektura mreže s dvosmjernom piramidom značajki (eng. *Bidirectional Pyramid Network*) koja je testirana u ovome radu.

BPNet se ističe jer ne koristi predtrenirane (eng. *pretrained*) modele za izvlačenje značajki već se svi parametri uče iznova počevši od standardne inicijalizacije. Model se sastoji od 4 ili 5 slojeva (3 ili 4 koraka poduzorkovanja) i 3 ili 4 stadija konvolucije na najnižem sloju (najvećoj rezoluciji). BPNet s 4 sloja i 3 stadija zvat ćemo BPNet-S3, a model s 5 slojeva i 4 stadija zvat ćemo BPNet-S4. Rezoluciju ulaza označit ćemo s 64x (x je odabrani cijeli broj koji pomnožen sa 64 daje rezoluciju ulazne slike). Ulaz prolazi kroz preliminarni konvolucijski korak koji smanjuje rezoluciju značajki na 16x (donji lijevi kut slike 3.1). Tu dolazi do grananja tokova konvolucije koji su na slici 3.1 označeni plavim strelicama. Prvi se tok kreće po najnižem sloju piramide slijeva nadesno zadržavajući rezoluciju značajki na 16x i prolazeći 3 stadija konvolucije ako se radi o BPNet-S3 modelu odnosno 4 stadija ako je riječ o modelu BPNet-S4 [2].



Slika 3.1: Vizualni prikaz BPNet arhitekture. Piramidalna struktura ove mreže sadrži 4 ili 5 slojeva konvolucije koji smanjuju rezoluciju značajki za pola u svakom novom sloju. Najniži sloj mreže sadrži 3 ili 4 stadija konvolucije koji ne mijenjaju rezoluciju značajki. Tokovi informacija osnovnog modela BPNet-S3 označeni su punim strelicama, dok su isprekidanim strelicama označeni tokovi

koji nadopunjuju osnovni model i grade BPNet-S4. Plave strelice označuju smjerove konvolucije, crvene strelice označuju tok informacija odozgo prema dolje, a žute strelice tok odozdo prema gore. Slika je preuzeta iz [2].

Drugi tok konvolucije kreće se prema vrhu piramide pritom smanjujući rezoluciju značajki za faktor 2 na svakom novom sloju. Na drugom sloju rezolucija značajki je 8x te se tokovi konvolucije opet granaju. Kao što je bio slučaj i na prethodnom sloju jedan tok vodi horizontalno održavajući rezoluciju konstantom na 8x dok drugi vodi prema višim slojevima gdje se rezolucija smanjuje. Ovakva obrada se ponavlja do vrha piramide koji je za BPNet-S3 na 4. sloju (uz rezoluciju 2x), a za BPNet-S4 na 5. sloju (uz rezoluciju 1x) [2].

Za razliku od mnogih drugih mreža korištenih za segmentaciju BPNet nema jasno definirane strukture enkodera i dekodera (eng. *Encoder-Decoder architecture*). Druga najznačajnija razlika je da se informacije ne prenose direktno s lijevog na desni kraj piramide, već značajke, ovisno o sloju u kojem se nalaze, prolaze kroz nekoliko stadija konvolucije (viši slojevi – manje stadija) [2].

Informacijski tokovi različitih rezolucija kreću se u tri osnovna smjera [2].

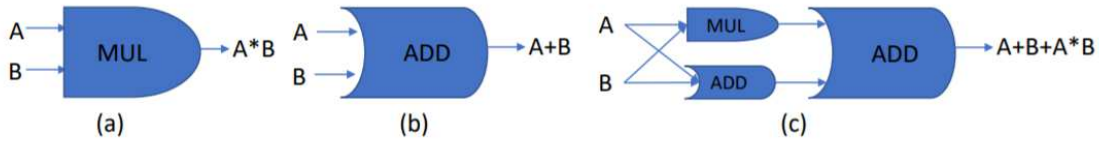
- Tok odozgo prema dolje (eng. *bottom-up information flow*). Informacije se kreću prema dolje u piramidi nakon svakog koraka obrade. Time se obrada u slojevima s višom rezolucijom obogaćuje semantički bogatijom i apstraktnijom informacijom. Na primjer, značajke iz drugog sloja (nakon konvolucije i poduzorkovanja pri rezoluciji 8x) se integriraju sa značajkama iz prvog sloja koje su prošle prvi stadij konvolucije (rezolucija 16x). Slično, značajke iz trećeg sloja (nakon dva stadija konvolucije i poduzorkovanja pri rezoluciji 4x) integriraju se značajkama iz drugog sloja koje su prvo prošle konvoluciju i poduzorkovanje penjući se po piramidi, a nakon toga su prošle stadij konvolucije uz konstantnu rezoluciju krećući se horizontalno prema desno. Primijetite kako se uvijek kombiniraju značajke koje su prošle isti broj konvolucijskih blokova (no ne i isti broj poduzorkovanja).
- Tok odozdo prema gore (eng. *top-down information flow*). Značajke veće rezolucije se, nakon integracije sa semantički bogatijim značajkama manje rezolucije (iz viših slojeva), integriraju s tim istim značajkama manje rezolucije. Kombiniranje značajki se odvija na višim razinama.

- Horizontalni (eng. *lateral information flow*) tok, odnosno tok slijeva-nadesno već je ranije opisan. Značajke prolaze kroz konvolucijske blokove bez smanjivanja rezolucije.

Ovakav dizajn obilježuje model s piramidom značajki; informacije teku horizontalno, odozdo prema gore i odozgo prema dolje te se integriraju u svakom koraku obrade [2].

Autori rada [2] predlažu AMA (eng. *add-multiply-add*) blok za integraciju (stapanje, kombiniranje) značajki. AMA blok je, prema njihovim saznanjima, korisniji od elementarnog zbrajanja ili množenja značajki, a čak daje i bolju točnost od drugih procesorski zahtjevnijih metoda.

Ako A i B predstavljaju matrice značajki, s time da matrica A sadrži značajke s niže razine (veće rezolucije), a B značajke s više razine (manje rezolucije), onda matrica B mora proći naduzorkovanje kako bi se mogle obaviti elementarne operacije nad te dvije matrice. Izlaz iz AMA bloka bit će izračunat prema formuli $A + B + A \cdot B$ kao na slici 3.2 [2].



Slika 3.2: Operacije (a) i (b) su standardne jednostavne operacije korištenje za kombinaciju značajki – elementarno množenje (a) i elementarno zbrajanje (b). Pod (c) je prikazan predloženi AMA blok koji dalje bolje rezultate od pristupa (a) i (b). Slika je preuzeta iz [2].

4. Implementacija, vanjske biblioteke i sklopovska podrška

4.1. Vanjske biblioteke

Za treniranje i testiranje modela u ovome radu korišten je Python. Python je programski jezik visoke razine koji, između ostalog, podržava objektno orijentiranu paradigmu te je sintaktički vrlo jasan i čitljiv. Uz Python koristile su se i mnoge druge vanjske biblioteke od kojih će najvažnije biti navedene.

PyTorch je biblioteka otvorenog koda namijenjena strojnom učenju. Početno je objavljena 2016. godine te se od tada učestalo nadograđuje, a razvija ju Facebook. Praktična je jer omogućava vrlo jednostavno sučelje za automatsko računanje parcijalnih derivacija preko računskog grafa te transparentno računanje na grafičkim karticama. U radu će se koristiti uz grafičku karticu tvrtke Nvidia koja podržava CUDA-u (eng. *Compute Unified Device Architecture*) – sučelje koje omogućuje paralelnu obradu opće namjene bez korištenja grafičkih biblioteka koje nisu stvorene za takvo što. Biblioteka se temelji na radu sa *tensorima* odnosno n-dimenzionalnim matricama.

Torchvision je biblioteka koja služi kao dodatak PyTorchu. Specijalizirana je za područje računalnog vida.

NumPy je jedna od najkorištenijih pythonskih biblioteka, a specijalizira se za rad s multidimenzionalnim matricama. Također je biblioteka otvorenog koda i temelji se na efikasnoj implementaciji operacija u „nižim“ programskim jezicima. Takve su operacije često vektorizirane što omogućava dodatna ubrzanja.

Matplotlib je biblioteka za crtanje koja se temelji na NumPy-u.

OpenCV je biblioteka koja se koristi za mnoge zadatke računalnog vida te podržava ubrzano računanje na grafičkim karticama. Biblioteka je u radu korištena za učitavanje i prikaz slika.

4.2. Skup podataka

Modeli su u ovome radu trenirani i testirani na skupu slika Cityscapes. Skup se sastoji od 5,000 slika cestovnih scena (i njihovih oznaka) iz perspektive automobila snimljenih u 50 različitih gradova. Ovaj je skup mnogo veći od prijašnjih skupova slične namjere. Sastoji se od ukupno 30 klasa no mi ćemo za naše potrebe koristiti samo 19.

Sastoji se od podskupova:

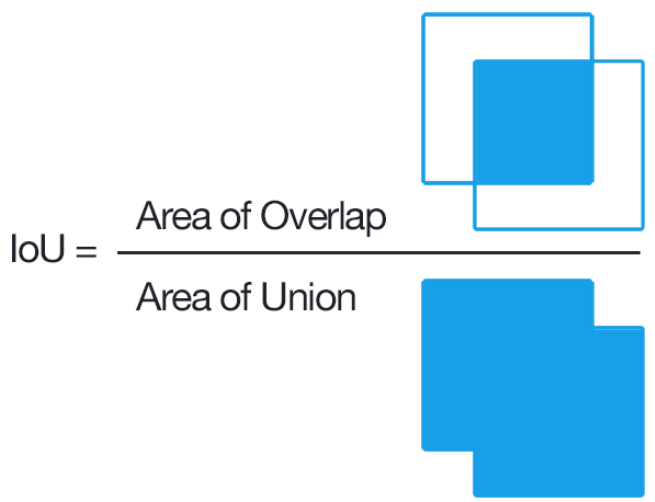
- *train* – skup za učenje (eng. *training dataset*), 2975 slika
- *test* – skup za testiranje (eng. *test dataset*), 1525 slika
- *val* – skup za provjeru (eng. *validation dataset*), 500 slika

4.3. Metrike

4.3.1. Točnost

Za zadatak semantičke segmentacije najčešće koristimo dvije različite metrike. Prva je točnost piksela (eng. *pixel accuracy*) i ona označava postotak piksela na slici koji su točno klasificirani. Ovakva će nam metrika biti nešto manje korisna kada u skupu podataka postoje klase koje dominiraju u učestalosti nad drugim klasama.

Iz tog razloga kao glavnu metriku koristit ćemo mjeru IoU (eng. *Intersection over Union*) također znanu i kao Jaccardov indeks koja je prikladnija za ovu zadaću [9]. IoU se računa kao omjer površine presjeka predikcije i točne oznake te unije predikcije i oznake. Računanje ove mjere vizualizirano je slikom 4.1.



Slika 4.1: Formula za izračun IoU. IoU se računa kao omjer površine presjeka predikcije i točne oznake te unije predikcije i oznake. Slika je preuzeta iz [9].

Primarno ćemo koristiti metriku mIoU (eng. *mean IoU*) koju ćemo dobiti kao srednju vrijednost IoU-a svih klasa.

4.3.2. Brzina

Za mjerenje brzine primarno će se koristiti broj slika u sekundi (eng. *frame rate, FPS*). FPS ćemo dobiti podijelimo li broj iteracija (koliko je slika prošlo kroz model) s ukupnim vremenom za odradu ovog zadatka. Druga mjera koju ćemo koristiti je prosječno vrijeme obrade za jednu sliku, a računat ćemo je kao recipročnu vrijednost prošle mjere, odnosno kao omjer ukupnog vremena i broja iteracija. Ovu mjeru ćemo izraziti u milisekundama (ms).

4.4. Sklopovska podrška

Za učenje i provjeru modela korištena je grafička kartica GTX 1060 6GB. Grafička kartica omogućava puno brži prolazak slike kroz model nego što bi to bilo moguće na običnim procesorima stoga štedi mnogo vremena budući da učenje ovako kompleksnih modela može trajati vrlo dugo. Jedna epoha učenja modela BPNet-S3 trajala je oko 20 minuta.

Kartica sadrži 6 GB video memorije što će biti glavno ograničenje za rješavanje zadaća dubokog učenja. Iz tog razloga modeli će se učiti na isječcima veličine 448x448 koji sadrže oko 4 puta manje piksela u odnosu na isječke na kojima su trenirani originalni promatrani modeli. Budući da pri provjeri slike prolaze kroz model jedna po jedna (veličina *batcha* je 1) to neće biti problem te će se moći provjeravati slike u punoj rezoluciji.

Također treba napomenuti da je grafička kartica postavljena na zadane postavke no brzina obrade slika bi se mogla značajno poboljšati povećanjem frekvencije video memorije uz minimalan dodatni utrošak energije.

5. Eksperimentalni rezultati

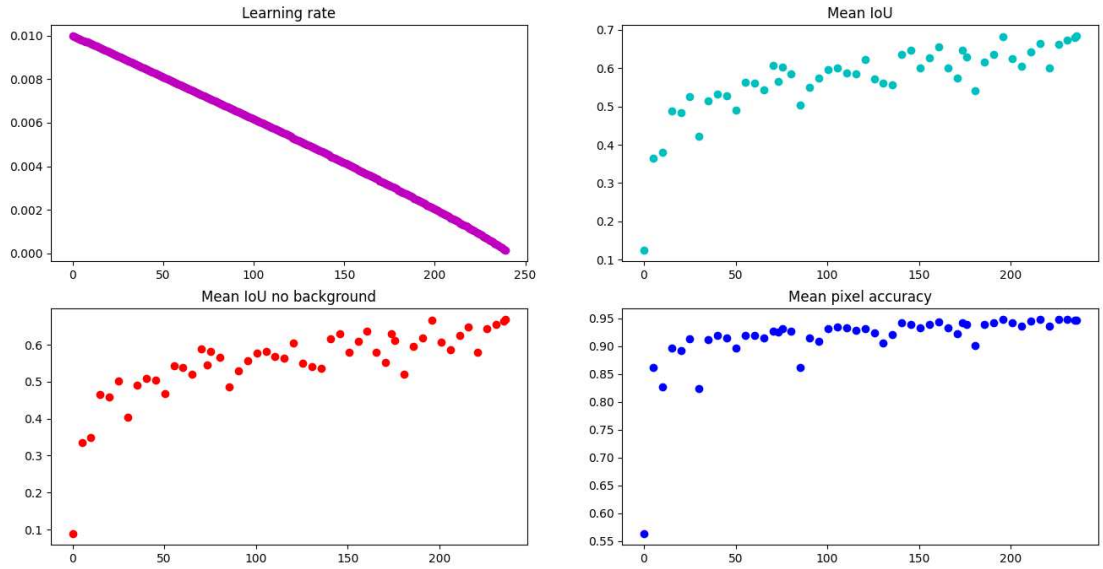
U ovom će se poglavlju usporediti točnost i performanse modela koji se koriste za semantičku segmentaciju. Modeli koji se uspoređuju trenirani su na slikama iz skupa Cityscapes. Za treniranje je korišten podskup *train* koji se sastoji od 2975 slika, a za testiranje podskup *val* koji sadrži 500 slika. Radi ograničenja količinom VRAM-a u korištenoj grafičkoj kartici modeli su bili trenirani na isječcima rezolucije 448x448 piksela što je zasigurno utjecalo na točnost predviđanja oba modela. Također, za treniranje je korišteno slučajno uvećanje odnosno smanjivanje te slučajno zrcaljenje slika po uputama iz radova [2] i [10].

5.1. Točnost modela

Treba napomenuti kako BPNet-S3 zauzima veliku količinu VRAM-a te prilikom treniranja u korištenu grafičku karticu stane 7 slika po *batchu*. Originalan *batch size* korišten za treniranje ovoga modela u radu [2] naveden je kao 12. Zbog navedenih ograničenja nije moguće ocijeniti utjecaj ovog parametra na točnost modela.

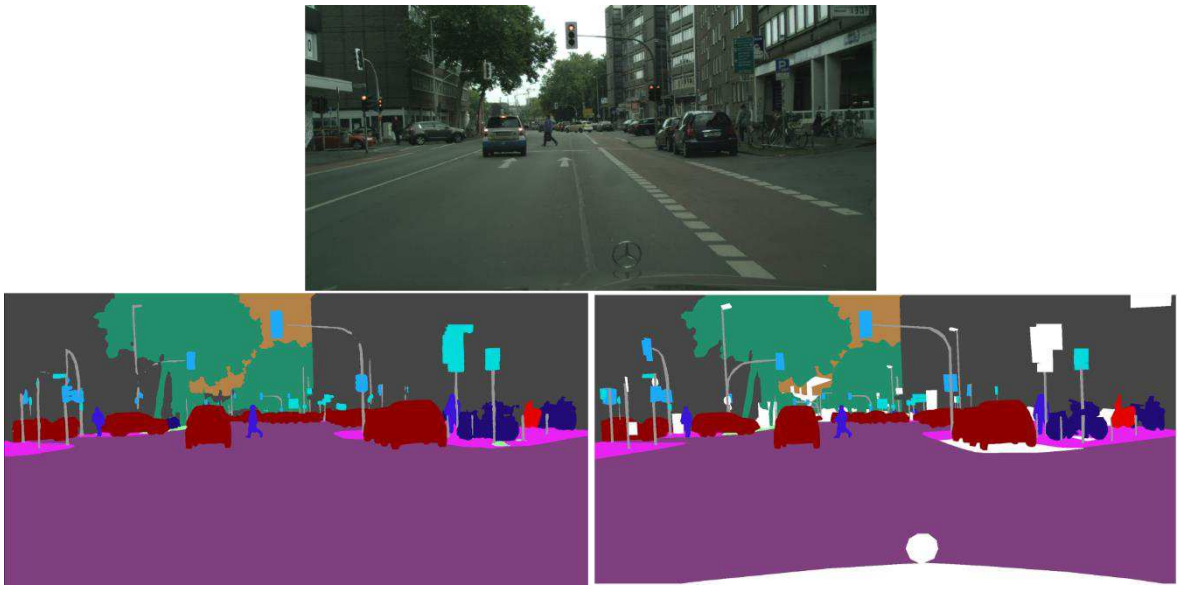
Ostali parametri modela namješteni su kao što je opisano u radu [2]. Oni glase: optimizator modela je stohastički gradijentni spust (eng. *stochastic gradient descent*, SGD), početna stopa učenja (eng. *learning rate*) je 10^{-4}, faktor zaleta (eng. *momentum*) je 0.9, broj epoha treniranja je 250.

BPNet-S3 treniran na pola rezolucije (poduzorkovanjem isječaka rezolucije 896x896 na 448x448 piksela) na validacijskom skupu *val* postiže mIoU od 69.05% uz točnost piksela od 94.77%. Isti model treniran na punoj rezoluciji te uz istu veličinu isječaka na *valu* postiže mIoU od 69.19% uz točnost piksela od 94.63%. Prikaz porasta točnosti i mIoU po epohama za prvi od ova dva pristupa treniranja dan je slikom 5.1.



Slika 5.1: Grafički prikaz porasta točnosti piksela i mIoU u odnosu na epohe treniranja. Slika se odnosi na treniranje modela BPNet-S3 pri pola rezolucije (poduzorkovanjem isječaka rezolucije 896x896 na 448x448 piksela).

U nastavku se na slikama 2.2 i 2.3 mogu vidjeti primjeri označenih slika. Izlazi iz modela obojeni su kako bi se jasnije uočile razlike u predviđenim klasama pojedinih piksela. Slike 2.1 i 2.2 sastoje se od originalnih slika koje predstavljaju ulaz u model te označenih i obojenih izlaza iz modela i točno označenih slika koje koristimo za usporedbu točnosti izlaza iz modela.



Slika 5.2: Primjer dobrog zaključivanja modela BPNet-S3 na slici iz validacijskom skupa Cityscapesa s mnogo sitnih detalja. Ulazna slika – gore; označen izlaz – dolje lijevo; točne oznake – dolje desno.

Slika 5.3: Primjer lošijeg zaključivanja modela BPNet-S3 na slici iz validacijskom skupa Cityscapesa. Najizraženije greške označene su bijelim elipsama. Jedna od grešaka je da model kamion sa slike djelomično označava kao automobil, djelomično kao autobus, a djelomično ga točno označava kao kamion. Druga očita greška je da je dio neba označen kao cesta. Ulazna slika – gore; označen izlaz – dolje lijevo; točne oznake – dolje desno.

BPNet-S3 u radu [2] treniran na punoj rezoluciji te isječcima veličine 896x896 postiže mIoU od 78.3% što je dosta više od eksperimentalno postignutih 69.2%. Razlika u mIoU pri treniranju BPNeta na punoj te na pola rezolucije (pri korištenju isječaka veličine 448x448 piksela) nije velika, ona iznosi samo 0.14% postotnih bodova. Može se zaključiti kako rezolucija treniranja ne utječe previše na točnost. Međutim, razlika između dobivenih rezultata i onih postignutih u radu [2] je velika, stoga se može pretpostaviti kako je točnost ovog modela vrlo osjetljiva na veličinu isječaka za treniranje te *batch size*.

5.2. Usporedba sa SwiftNetom

BPNet ćemo, radi sličnosti arhitektura, usporediti s piramidalnim SwiftNetom iz rada [10]. SwiftNet, za razliku od BPNeta, koristi predtreniranje na skupu slika ImageNet.

SwiftNet je također treniran na isječcima manjih dimenzija kako bi što više slika stalo u *batch*. U originalnom se radu za treniranje koriste isječci veličine 768x768 piksela. Kao što je bio slučaj i kod BPNeta, isprobana su dva pristupa; prvi je uključivao jednostavno izrezivanje isječaka dimenzija 448x448 piksela, dok su u drugom slučaju isječci dimenzija 768x768 poduzorkovani na dimenzije 448x448 piksela (model je efektivno učio na pola rezolucije). Korišten je *batch size* 14 kao što je i sugerirano u radu [10] iz čega se odmah da zaključiti da SwiftNet ima puno manje memorijske zahtjeve u odnosu na BPNet. Svi ostali parametri učenja također su bili namješteni na zadane vrijednosti, a one glase: korišteni optimizator je Adam, početna stopa učenja je $4 \cdot 10^{-4}$, *weight decay* je 10^{-4}, broj epoha treniranja je 250.

Konačna usporedba točnosti ova dva modela dana je tablicom 5.1. Iz tablice se vidi da na piramidalni SwiftNet manji isječci za učenje puno manje utječe. Konačan mIoU od 74.3% nije daleko od 76.4% mIoU izmjerenih u radu [10] nakon učenja na originalnim isječcima (uz punu rezoluciju). Učenje na poduzorkovanim isječcima je rezultiralo nešto manjim mIoU od 72.1%.

Tablica 5.1: Izmjerene točnosti ova dva modela na validacijskom skupu nakon 250 epoha treniranja na skupu za učenje. Modeli označeni sa „poduzorkovani isječci“ trenirani su poduzorkovanjem isječaka originalnih veličina iz radova [2] i [10] na isječke veličine 448x448. Kod ostalih su modela isječci veličine 448x448 izravno izrezivani iz slika.

model	točnost piksela	mIoU
BPNet-S3	94.63%	69.19%
BPNet-S3 (poduzorkovani isječci)	94.77%	69.05%
piramidalni SwiftNet	94.96%	74.30%
piramidalni SwiftNet (poduzorkovani isječci)	94.76%	72.06%

5.3. Brzina obrade

Brzina obrade ova dva modela također je testirana na sklopovlju opisanom u poglavlju 4.4. Modeli su testirani u 1,000 iteracija tako što im je uvijek dana ista ulazna slika kako bi mogla biti u memoriji grafičke kartice za vrijeme cijelog testa. Slika je iz skupa Cityscapes, a rezolucije je 1024x1024. Modeli su u testu trebali proizvesti segmentiranu izlaznu sliku iste rezolucije te su prosječno vrijeme obrade i broj slika u sekundi izračunati uprosječivanjem vremena obrade za ukupan broj iteracija.

Rezultati su prikazani tablicom 5.2, a pokazuju nam kako SwiftNet obradi jednu sliku u 39.2% vremena koje treba BPNetu za isti zadatak. Odnosno, drugim riječima, SwiftNet je oko 2.5 puta brži u obradi slika u odnosu na BPNet-S3.

Tablica 5.2: Izmjerene brzine obrade ova dva modela na slikama rezolucije 1024x1024 piksela.

model	FPS	vrijeme obrade slike
BPNet-S3	10.96	91.24 ms
piramidalni SwiftNet	27.98	35.74 ms

Zaključak

Semantička segmentacija slika vrlo je težak zadatak s raznim industrijskim i znanstvenim primjenama. Zadnjih se godina uspješno rješava uz duboke neuronske mreže. BPNet je vrlo uspješan u semantičkoj segmentaciji na što ukazuje da najkompleksniji model opisan radom [2] – BPNet-S4 postiže jedne od najboljih rezultata na skupu slika Cityscapes. BPNet-S3 je model srednje kompleksnosti te je glavni fokus ovoga rada. Radi se o pojednostavljenju modela BPNet-S4 što u praksi znači da ovaj model ostvaruje veću brzinu zaključivanja bez većeg gubitka točnosti.

Radi memorijskih ograničenja BPNet-S3 je u ovom radu učen na manjim isječcima te je uspoređen s piramidalnim modelom SwiftNeta također učenim na isječcima iste veličine. Rezultati pokazuju kako BPNet-S3 postiže mIoU od oko 69% što je podosta manje u odnosu na mIoU od 78.3% kojeg postiže isti model učen na isječcima duplo većih dimenzija. SwiftNet puno manje gubi na točnosti kada se trenira na manjim isječcima – 74.3% je mIoU dobiven na isječcima dimenzija 448x448 dok pri punoj veličini isječaka model postiže mIoU od 76.4%.

SwiftNet je također i nešto brži. Za oba je modela testirana brzina obrade slika rezolucije 1024x1024 na grafičkoj kartici GTX 1060 6GB. BPNet-S3 uspijeva obraditi 10.96 slika u sekundi dok SwiftNet obrađuje 27.98 slika u sekundi. Promatrani model BPNeta bio bi pogodan za obradu u stvarnom vremenu uz korištenje brže grafičke kartice.

Budući rad bi mogao uključiti treniranje oba modela na usluzi Google Colab koja omogućava izvršavanje koda na grafičkim karticama s mnogo većim dostupnim količinama video memorije. Tada bi se mogla provjeriti točnost oba modela istrenirana na punoj rezoluciji.

Također bi trebalo provjeriti i točnost ovih modela na drugim skupovima podataka kao što su CamVid ili COCO kako bi se provjerilo ponašanje modela na različitim podatcima.

Konačno, bilo bi zanimljivo i usporediti ponašanje modela nakon što se prevedu u TensorRT. TensorRT često omogućuje značajno poboljšanje brzine obrade što bi dalo uvid u mogućnost optimizacije ovakvih modela.

Literatura

[1] Bašić, B. D., Čupić, M., Šnajder, J. *Umjetne neuronske mreže*. Zagreb, 2008. URL: https://www.fer.unizg.hr/_download/repository/UmjetneNeuronskeMreze.pdf

[2] Nie, D., Xue, J., Ren, X. *Bidirectional Pyramid Networks for Semantic Segmentation*, ACCV, 2020.

[3] Kelleher, J. D., *Deep Learning*, Mit Press, 2019.

[4] *An Intuitive Explanation of Convolutional Neural Networks*, (2016., kolovoz) URL: https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

[5] Lee, H., Grosse, R., Ranganath, R., Ng, A. Y. *Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations*, ICML '09, 2009.

[6] *Journal of Big Data,* URL: https://www.researchgate.net/figure/Leaky-rectified-linear-unit-a-01_fig2_334947119

[7] Bašić, B. D., Čupić, M., Šnajder, J. *11. Umjetne neuronske mreže*. URL: https://www.fer.unizg.hr/_download/repository/UI_12_UmjetneNeuronskeMreze[1].pdf

[8] Matcha, A. C. N. *A 2021 guide to Semantic Segmentation*, (svibanj, 2021.) URL: https://nanonets.com/blog/semantic-image-segmentation-2020/

[9] Tiu, E. *Metrics to Evaluate your Semantic Segmenatation Model*, (2019., kolovoz) URL: https://towardsdatascience.com/metrics-to-evaluate-your-semantic-segmentation-model-6bcb99639aa2

[10] Oršić, M., Šegvić, S. *Efficient semantic segmentation with pyramidal fusion*, Elsevier (2020.)

[11] Shukla, L., Fund*amentals of neural networks*, (2019., kolovoz) URL: https://wandb.ai/site/articles/fundamentals-of-neural-networks

[12] Tsang, S. *Review: SharpMask — 1st Runner Up in COCO Segmentation 2015 (Instance Segmentation)*, (2018., prosinac) URL: https://towardsdatascience.com/review-sharpmask-instance-segmentation-6509f7401a61

[13] Bašić, B. D., Šnajder, J. *Strojno učenje*. Zagreb, 2014. URL: https://www.fer.unizg.hr/_download/repository/StrojnoUcenje.pdf

Sažetak

Ovaj rad razmatra razumijevanje prirodnih scena računalnim vidom. Stanje tehnike u tom području trenutno postižu konvolucijski modeli. U središtu interesa su modeli za semantičku segmentaciju koji svaki piksel ulaza svrstavaju u jedan od poznatih razreda. Opisuje se model s dvosmjernom piramidom značajki koji uvodi mnoge preinake u već poznati piramidalni model koji uspješno rješava zadaću segmentacije. Opisani model - BPNet - ne koristi predtreniranje na drugim skupovima podataka. Također, model se koristi dvosmjernim tokovima informacija između poduzorkovanih značajki i značajki visoke rezolucije te za kombiniranje značajki koristi vrlo uspješan, ali i efikasan pristup. Ispituje se uspješnost i brzina modela na skupu slika Cityscapes koji se sastoji od slika cestovnih scena što nam je zanimljivo zbog mogućnosti korištenja ovakvih modela u sustavima za autonomnu vožnju. Konačno, BPNet se uspoređuje s još jednom piramidalnom arhitekturom - SwiftNetom.

Ključne riječi: BPNet, SwiftNet, semantička segmentacija, računalni vid, duboko učenje, umjetna inteligencija, Cityscapes, dvosmjerna piramida značajki

Summary

This paper considers understanding natural scenes with computer vision, which is currently achieved by convolutional models. The center of interest are semantic segmentation models, which classify each pixel of input as one of the known classes. A bidirectional feature pyramid model is described, which introduces many modifications to the already known pyramid model that successfully solves the task of segmentation. The described model - BPNet - does not use a feature extractor pretrained on other datasets. Also, this model uses bidirectional flow of information between low and high resolution features and a very successful, but also efficient, approach to feature fusion. Speed and accuracy are tested on Cityscapes image set consisting of images recoreded in street scenes. This use case interests us because of the possibility of using such models in autonomous driving. Finally, BPNet is compared with another pyramidal architecture - SwiftNet.

Keywords: BPNet, SwiftNet, semantic segmentation, computer vision, deep learning, artificial intelligence, Cityscapes, bidirectional feature pyramid

Skraćenice

AMA	*add-multiply-add*	zbroji-pomnoži-zbroji
CNN	*Convolutional Neural Network*	konvolucijska neuronska mreža
CUDA	*Compute Unified Device Architecture*	
FPS	*frames per second*	broj slika u sekundi
GB	*Gigabyte*	gigabajt
IoU	Intersection over Union	omjer presjeka i unije
ReLU	*Rectified Linear Unit*	zglobnica
SGD	*Stohastic Gradient Descent*	stohastički gradijentni spust
UI (AI)	*Artificial Intelligence*	umjetna inteligencija
3D	*three-dimensional*	trodimenzionalno