SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 6989

RAZUMIJEVANJE SCENA IZ PERSPEKTIVE VOZAČA SEMANTIČKOM SEGMENTACIJOM

Klara Marijan

Zagreb, lipanj 2020.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 6989

RAZUMIJEVANJE SCENA IZ PERSPEKTIVE VOZAČA SEMANTIČKOM SEGMENTACIJOM

Klara Marijan

Zagreb, lipanj 2020.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 13. ožujka 2020.

ZAVRŠNI ZADATAK br. 6989

Pristupnica:	**Klara Marijan (0036505959)**
Studij:	Računarstvo
Modul:	Računarska znanost
Mentor:	prof. dr. sc. Siniša Šegvić

Zadatak: **Razumijevanje scena iz perspektive vozača semantičkom segmentacijom**

Opis zadatka:

Semantička segmentacija prirodnih scena važan je zadatak računalnog vida s mnogim zanimljivim primjenama. U posljednje vrijeme vrlo zanimljive rezultate u okviru tog zadatka postižu konvolucijski modeli s ljestvičastim naduzorkovanjem. Ovaj rad razmatra specifičnosti raspoznavanja prometnih sudionika te elemenata okoliša u urbanim cestovnim scenama iz perspektive vozača. 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 semantičku segmentaciju. Predložiti arhitekturu dubokog modela koja bi bila prikladna za semantičku segmentaciju cestovnih scena. Uhodati postupke učenja modela i validiranja hiperparametara. Primijeniti naučene modele te prikazati i ocijeniti ostvarene rezultate.. 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: 12. lipnja 2020.

Zahvaljujem mentoru prof. dr. sc. Siniši Šegviću na izdvojenom vremenu i stručnim savjetima te mag. ing. Marinu Oršiću na pruženoj pomoći.

Sadržaj

Uvod

U posljednje vrijeme vidljiv je sve veći i brži napredak u području strojnog i dubokog učenja koje je iz dana u dan sve popularnije. Računalni vid izdvaja se kao jedna od grana vrlo široke primjene. Ono što je omogućilo značajan napredak jest korištenje dubokih konvolucijskih modela u rješavanju problema toga područja. Istovremeno učenje svih slojeva dubokog modela omogućilo je rješavanje problema koji su presloženi da bi se mogli riješiti algoritamski. Jedan od takvih problema, a ujedno i jedan od glavnih zadataka računalnog vida je problem semantičke segmentacije, odnosno pridjeljivanja odgovarajuće klase svakom pikselu na slici.

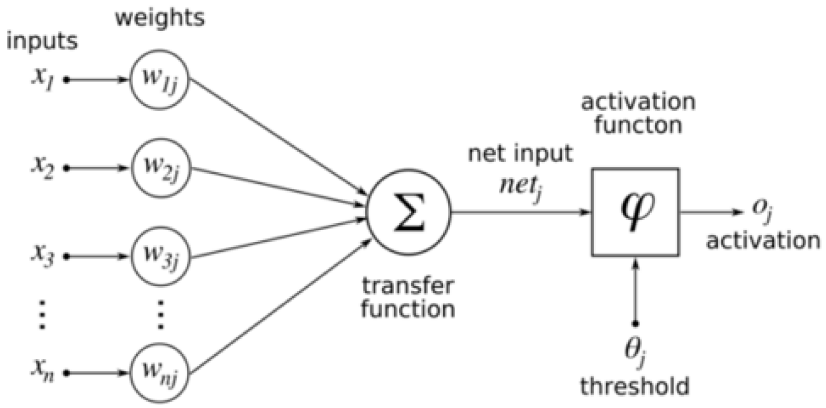
Za semantičku segmentaciju najčešće se koriste konvolucijske neuronske mreže u kojima je naglasak na konvolucijskim slojevima koji izvlače pojedine značajke, poput primjerice rubova, iz slika i latentnih reprezentacija. Konvolucijski slojevi su ekvivarijantni na translaciju, što omogućuje opisivanje složenih lokalnih zavisnosti malim brojem parametara. To svojstvo posebno je prikladno za složene podatke s rešetkastom topološkom strukturom poput slika, teksta, govora itd. Iako postoje razne vrste neuronskih mreža, posebno su se istaknule rezidualne neuronske mreže koje su svojom pojavom omogućile brzo treniranje vrlo dubokih modela. U ovom radu detaljnije je opisana jedna od njih, ResNet-18, koja je također nadograđena u sklopu rada.

U prvom poglavlju rada opisana je osnovna arhitektura općenite umjetne neuronske mreže, a zatim su u drugom poglavlju pojašnjeni koncepti prema kojima neuronske mreže funkcioniraju. Treće poglavlje govori detaljnije o samoj semantičkoj segmentaciji koja je tema ovog rada. U konačnici je opisana programska implementacija mreže za raspoznavanje prometnih sudionika te elemenata okoliša u urbanim cestovnim sredinama te su navedeni dobiveni rezultati. Osnovni model mreže nadograđen je dodavanjem ljestvičastog naduzorkovanja te je provedena usporedba među rezultatima dobivenih uz korištenje osnovnog modela i nadograđenog modela.

1. Duboko učenje - neuronske mreže

1.1. Umjetni neuron i arhitektura mreže

Umjetne neuronske mreže su vrsta parametarskih modela strojnog učenja koji se koriste kako bi se aproksimirale složene funkcije s velikim brojem parametara. Inspiracija za nastankom umjetnih neuronskih mreža proizašla je iz bioloških neurona i središnjeg živčanog sustava ljudi i životinja. Temeljna jedinica mreža je umjetni neuron koji je prikazan na slici 1.1.



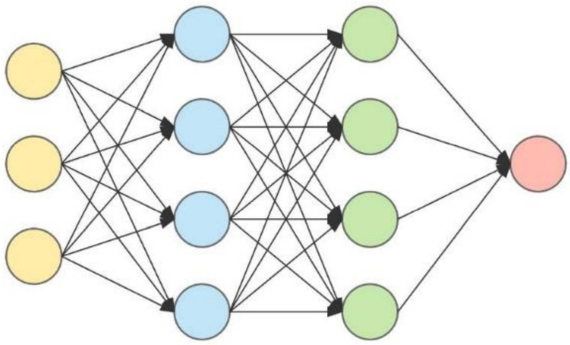
Slika 1.1: Umjetni neuron. (preuzeto dana 05.05.2020. s: https://upload.wikimedia.org/wikipedia/commons/6/60/ArtificialNeuronModel_english.png)

Umjetni neuron na ulazu prima određeni broj ulaza (x_1, x_2, …, x_n). Navedeni ulazi množe se sa pripadnim težinama u obliku realnih brojeva (w_1, x_2, …, w_n) koje određuju u kojoj mjeri jedan neuron ima utjecaj na drugi neuron. Umnožak ulaza s pripadnim težinama zbraja se s pragom θ te se na zbroj u konačnici primjenjuje prijenosna funkcija φ. Tako se dobiva izlaz iz neurona y.

Ovaj izraz matematički se zapisuje u obliku zadanom izrazom 1.1:

$$y = \varphi\left(\sum_i w_i x_i - \theta\right) \tag{1.1}$$

Neuronske mreže sastoje se od skupa međusobno povezanih neurona koji su podijeljeni u slojeve. Prvi sloj je ulazni sloj te se na njega nadovezuje određeni broj skrivenih slojeva nakon kojih slijedi izlazni sloj. Ukoliko je riječ o unaprijednim neuronskim mrežama kojima se ovaj rad bavi, tada veze između neurona ne stvaraju cikluse te je svaki sloj mreže povezan samo sa susjednim slojevima na način da izlaz prethodnog sloja predstavlja ulaz idućeg sloja te se tako redom računaju izlazi slojeva počevši od ulaznog sloja prema izlaznom sloju. Ako je svaki neuron jednog sloja povezan sa svakim neuronom sljedećeg sloja, tada govorimo o potpuno povezanom sloju. Slojevi mogu, ali ne moraju biti potpuno povezani. Na slici 1.2 prikazan je primjer unaprijedne neuronske mreže koja se sastoji od tri potpuno povezana sloja (dva skrivena sloja i jedan izlazni sloj).



Slika 1.2: Unaprijedna neuronska mreža. (preuzeto dana 05.05.2020. s: https://cdn-images-1.medium.com/max/1000/1*livHOtvW8PSptrSb7OXpyA.jpeg)

Arhitektura neuronske mreže može se opisati kompozicijom funkcija s obzirom na to da izlaz jednog sloja predstavlja ulaz drugog sloja. Neka je funkcija svakog skrivenog sloja i izlaznog sloja definirana sljedećim izrazom:

$$f_i(x) = \varphi(W_i \cdot x - \theta_i) \tag{1.2}$$

Pri tome i označava indeks sloja, φ predstavlja prijenosnu funkciju, W_i matricu težina pripadnog sloja, a θ_i vektor pragova. Tada ukupni izlaz iz neuronske mreže sa slike 1.2 možemo zapisati kao:

$$f(x) = f_3(f_2(f_1(x))) \tag{1.3}$$

1.2. Prijenosne funkcije

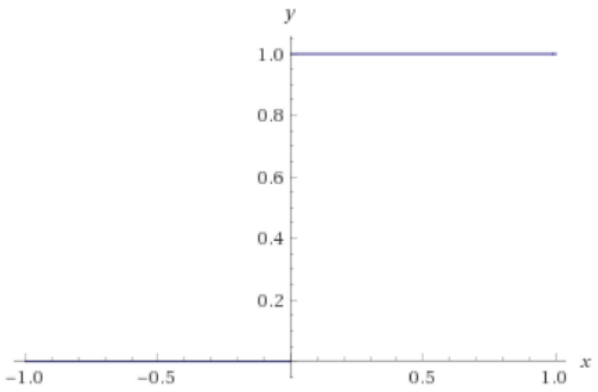
Prijenosne funkcije imaju ulogu pretvaranja ulaznih vrijednosti u izlazne vrijednosti. Postoje različite vrste prijenosnih funkcija i odabir funkcije koja je prikladna ovisi o samoj arhitekturi mreže. Linearne prijenosne funkcije se koriste samo u iznimno jednostavnim neuronskim mrežama jer svojstvo linearnosti ograničava mogućnosti mreže. Budući da je kompozicija dvije linearne funkcije također linearna funkcija, zaključujemo da bez obzira na to od koliko skrivenih slojeva se mreža sastoji, svi će se slojevi ponašati na isti način. Iz tog razloga, za kompleksnije modele koristimo nelinearne funkcije koje neuronima omogućuju učenje propagacijom pogreške unatrag (engl. *back-propagation*) uz pomoć gradijenata.

Najpoznatije prijenosne funkcije su funkcija skoka, sigmoidalna funkcija, funkcija tangens hiperbolni i zglobnica koje su opisane u nastavku [1].

Funkcija skoka (engl. *step function*) se koristila u ranijim mrežama, no danas se ne koristi često. Funkcija skoka na izlazu daje nulu za sve ulazne vrijednosti koje su manje od nule, a jedinicu za ostale vrijednosti:

$$\mathrm{u}(x) = \begin{cases} 1, x \geq 0 \\ 0, x < 0 \end{cases} \tag{1.4}$$

Ova funkcija nije diferencijabilna zbog skoka u vrijednosti $x = 0$, te iz tog razloga nije prikladna za učenje na temelju algoritma propagacije pogreške unatrag koji koristi gradijente pri učenju. Na slici 1.3 je prikaz funkcije skoka.

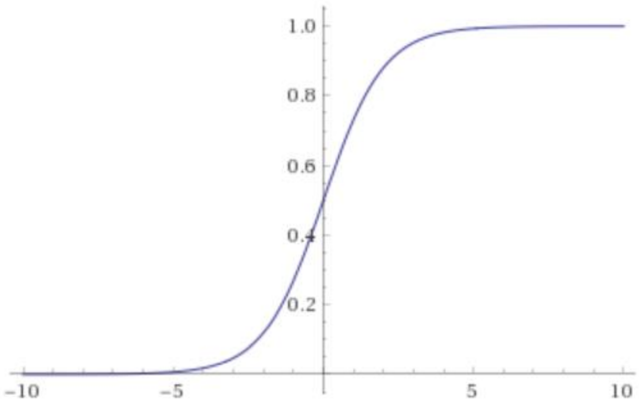


Slika 1.3: Funkcija skoka.

Sigmoidalna funkcija se matematički zapisuje u obliku:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{1.5}$$

Često je primjenjiva zbog svojih svojstava. Kodomenu funkcije čini interval [0, 1] iz čega slijedi da je sigmoidalna funkcija ponajviše primjenjiva u modelima pomoću kojih je potrebno izraziti vjerojatnost kao izlaz iz mreže, s obzirom na to da vjerojatnosti poprimaju isključivo realne vrijednosti u intervalu [0, 1]. Nadalje, sigmoidalna funkcija za izrazito velike pozitivne brojeve na izlazu daje vrijednost približno jednaku jedinici, dok za velike negativne brojeve daje približno nulu. Ovo svojstvo nije poželjno usprkos tome što je funkcija diferencijabilna. Naime, kako funkcija na krajevima teži u nulu, odnosno jedinicu, na tim mjestima je vrlo slabo osjetljiva na promjene te tada staje učenje uz pomoć lokalnih gradijenata s obzirom na to da gradijenti pri stohastičkom gradijentnom spustu (o kojem će biti riječ u poglavlju 1.5) ne mogu napraviti dovoljno značajne promjene. Izgled sigmoidalne funkcije prikazan je na slici 1.4.



Slika 1.4: Sigmoidalna funkcija.

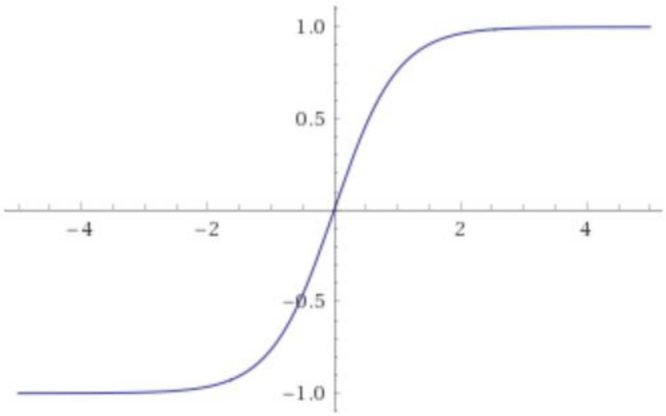
Tangens hiperbolni je prijenosna funkcija koja se pokazala uspješnijom od sigmoidalne funkcije pri učenju umjetnih neuronskih mreža, a njezina jednadžba glasi:

$$tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{1.6}$$

Na slici 1.5 je grafički prikaz funkcije. Kodomena funkcije tangens hiperbolni je interval

[-1,1] te funkcija ima važno svojstvo da se izlazne y vrijednosti grupiraju oko nule. Iako je i ova funkcija diferencijabilna, i kod nje je prisutan problem zasićenja neurona. Sigmoidalna funkcija i funkcija tangens hiperbolni povezane su sljedećim izrazom:

$$\tanh(x) = 2\sigma(2x) - 1 \tag{1.7}$$



Slika 1.5: Tangens hiperbolni.

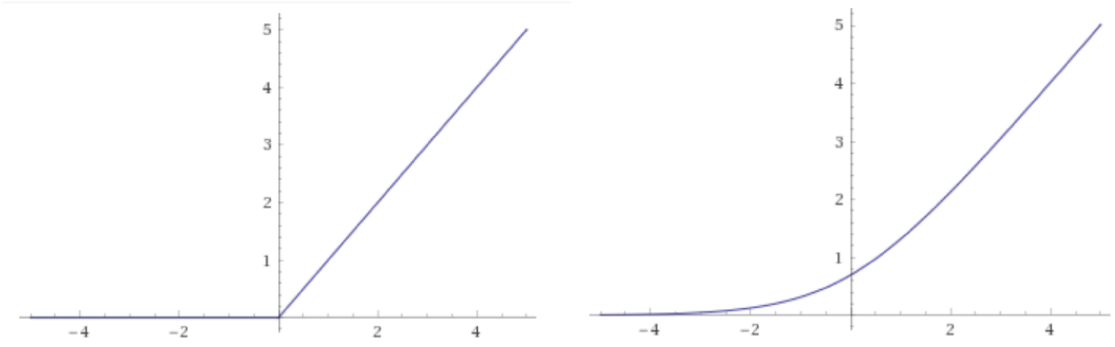
ReLU (engl. *Rectified Linear Unit*), odnosno **zglobnica**, prikazana na slici 1.6, jedna je od najpopularnijih prijenosnih funkcija. Matematički se zapisuje kao:

$$f(x) = max(0, x) \tag{1.8}$$

Dakle, funkcija na izlazu daje nulu za sve negativne ulazne vrijednosti, dok pozitivne ulazne vrijednosti propusti na izlaz u jednakom obliku kao što su bile na ulazu. Prednost zglobnice u odnosu na ostale prijenosne funkcije je njezina brzina računanja i efikasnost pri rješavanju problema koji se javljaju pri učenju neuronskih mreža. Ipak, i zglobnica ima nedostatak koji se očituje iz toga što se negativne ulazne vrijednosti automatski pretvaraju u nulu te se tako smanjuje mogućnost modela da pravilno uči iz danih podataka. No, ovaj nedostatak se ublažava normalizacijom grupe, jednom od metoda regularizacije koja je detaljnije pojašnjena u poglavlju 1.7.

Zglobnica ima i svoju glatku aproksimaciju koja je prikazana na slici 1.6, tzv. funkciju **softplus** čija je derivacija sigmoidalna funkcija i koja glasi:

$$softplus(x) = \ln(1 + e^x) \tag{1.9}$$



Slika 1.6: Zglobnica i softplus.

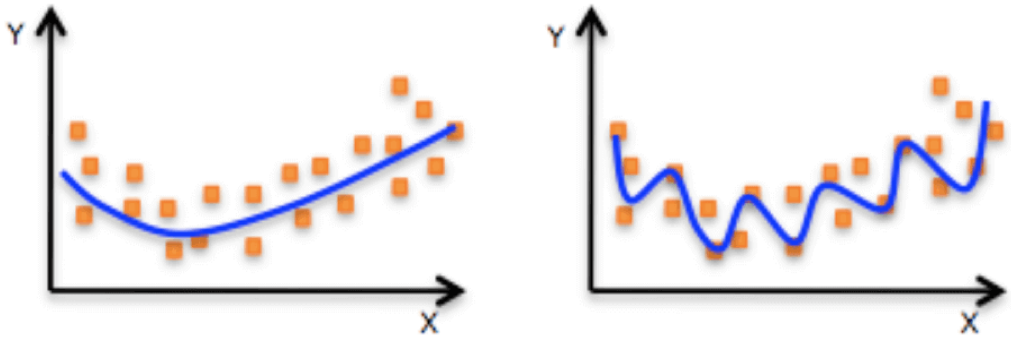
Također, funkcija slična zglobnici je i **propusna zglobnica** (engl. *Leaky Rectified Linear Unit*) koja se vrlo često koristi u reverzibilnim modelima. Njezina jednadžba definirana je izrazom (1.10) iz kojega se vidi da se negativne ulazne vrijednosti ne pretvaraju u nulu, već se smanjuju množenjem s koeficijentom 0.01.

$$\mathrm{f}(x) = \begin{cases} x\,, x \geq 0 \\ 0.01x\,, x < 0 \end{cases} \tag{1.10}$$

1.3. Učenje neuronske mreže

Ovaj rad se bavi nadziranim učenjem (engl. *supervised learning*) koje je tip strojnog učenja kojega karakterizira svojstvo da su podaci u obliku $(ulaz, izlaz) = (x, y)$ te je potrebno naći preslikavanje $y = f(x)$. Ukoliko je y kontinuirana, brojčana vrijednost, tada je riječ o regresijskom problemu. Ako je pak y kategorička ili nebrojčana vrijednost, onda govorimo o klasifikaciji. Pri nadziranom učenju neuronskoj mreži se predočavaju ulazni podaci te pripadajući izlazni, odnosno ciljni podaci. Temeljem toga se pri učenju izmjenjuju težine i pragovi u neuronskoj mreži (parametri) kako bi mreža adekvatno aproksimirala model koji najbolje opisuje dane podatke. Bitno svojstvo koje mreža mora imati jest mogućnost generalizacije, odnosno potrebno je paziti da pri učenju ne dođe do prenaučenosti mreže (engl. *overfitting*). Do prenaučenosti dolazi kada model daje odlične rezultate na skupu za učenje, no loše predviđa izlazne vrijednosti na skupu za treniranje. Na slici 1.7 je vizualno prikazan problem prenaučenosti. Iz slike je vidljivo da se model prilikom učenja previše prilagodio podatcima iz skupa za učenje te iako za taj skup daje gotovo savršene predikcije, model nije u stanju generalizirati na neviđenim podacima,

odnosno donositi općenite zaključke te se dobivaju nezadovoljavajući rezultati. Jedan od načina kojim se sprječava prenaučenost modela je stvaranje podjele skupa podataka na skup za učenje, provjeru i testiranje te metoda regularizacije koja će biti detaljnije opisana u nastavku poglavlja, odnosno u potpoglavlju 1.7.



Slika 1.7: Graf koji ilustrira prikladan (lijevo) i prenaučen model (desno). (preuzeto dana 07.05.2020. s: https://3gp10c1vpy442j63me73gy3s-wpengine.netdna-ssl.com/wp-content/uploads/2018/03/Screen-Shot-2018-03-22-at-11.22.15-AM-e1527613915658.png)

Da bismo mogli odrediti u kojoj mjeri neuronska mreža odgovara ciljnoj funkciji, koristimo funkciju gubitka koja predstavlja mjeru pogreške te ju nastojimo optimizirati jer na taj način mreža uči.

1.4. Funkcija gubitka

Funkcija gubitka pokazuje koliko znatno naš model, dakle neuronska mreža, odstupa od ispravnih rezultata. Pri odabiru prikladne funkcije gubitka postoji više opcija. Ako naš model rješava regresijski problem, često se koristi srednja kvadratna pogreška (engl. *mean square error*):

$$MSE = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n} \tag{1.11}$$

Kao što i samo ime kaže, srednja kvadratna pogreška mjeri se kao prosječna vrijednost kvadrirane razlike predviđenih vrijednosti i stvarnih vrijednosti izlaza. Zbog kvadriranja, predviđanja koja su vrlo različita od stvarnih vrijednosti u velikoj mjeri povećavaju iznos pogreške, dok manje razlike ne utječu znatno na pogrešku. Još jedna od mogućih opcija je

korištenje srednje apsolutne pogreške (engl. *mean absolute error*) koja se računa vrlo slično kao i srednja kvadratna pogreška, s time da je razlika u tome da se umjesto kvadriranja primjenjuje operator modula:

$$MAE = \frac{\sum_{i=1}^{n} |y_i - \hat{y}_i|}{n} \tag{1.12}$$

Srednja apsolutna pogreška više je robusna na stršeće vrijednosti od srednje kvadratne pogreške.

Kada se model bavi traženjem rješenja za problem klasifikacijske prirode, poput problema kojim se ovaj rad bavi, tada se pogreška može računati unakrsnom entropijom (engl. *cross-entropy loss*) koja je prikladna uz prijenosnu funkciju softmax:

$$E = -\frac{1}{n}\sum_{i=1}^{n}\sum_{c=1}^{C} y_{ic} \ln \hat{y}_{ic} = -\frac{1}{n}\sum_{i=1}^{n} \ln \hat{y}_{ic} \tag{1.13}$$

Pri tome $\hat{y}_{ic}$ predstavlja procijenjenu, a y_{ic} stvarnu vjerojatnost da određeni podatak pripada klasi s oznakom c, dok n označava ukupni broj podataka. Stvarna izlazna vrijednost y_{ic} za određeni uzorak će imati vjerojatnost jednaku jedinici samo za jednu od c klasa, onu kojoj zaista pripada, a stvarne vjerojatnosti ostalih klasa bit će nula s obzirom da uzorak pripada samo jednoj klasi. Zato je pri sumiranju po uzorcima dovoljno za svaki uzorak izračunati samo $\ln \hat{y}_{ic}$ za onu klasu kojoj taj uzorak stvarno pripada, a ostale zanemariti jer će biti pomnoženi s nulom.

1.5. Gradijentni spust

Pri učenju neuronske mreže, cilj je minimizirati gubitak dobiven funkcijom gubitka na način da se iterativno podešavaju parametri mreže, dakle težine i pragovi. U ovu svrhu koristi se gradijentni spust koji izračunava gradijente, odnosno parcijalne derivacije funkcije gubitka po parametrima mreže. Neka je zadana funkcija pogreške J koja sadrži parametre $\theta_1, \theta_2, \ldots, \theta_n$. Tada za ΔJ vrijedi formula:

$$\Delta J \approx \nabla J^T \cdot \Delta \boldsymbol{\theta} \tag{1.14}$$

Pri čemu je pomak definiran kao

$$\Delta\theta = (\Delta\theta_1, \Delta\theta_2, \ldots, \Delta\theta_n) = -\eta \nabla J \tag{1.15}$$

Oznaka η označava stopu učenja koja je uobičajeno vrlo mali realan i pozitivan broj. Kombiniranjem prethodne dvije formule zaključujemo da će razlika pogreške ΔJ biti pozitivan broj (ukoliko za funkciju vrijedi Taylorova aproksimacija prvog reda), odnosno da će se iznos pogreške postupno smanjivati. Minimizaciju funkcije gubitka postižemo tako da parametrima mreže umjesto starih pridružujemo nove vrijednosti koje računamo na sljedeći način:

$$w_i = w_i - \eta \frac{\partial \mathrm{J}}{\partial w_i} \tag{1.16}$$

$$b_i = b_i - \eta \frac{\partial \mathrm{J}}{\partial b_i} \tag{1.17}$$

Pri tome oznake w označavaju težine, a oznake b pristranosti ($b = -prag$). U svakom koraku se polako približavamo minimumu funkcije gubitka.

Postupak pronalaska minimuma funkcije gubitka može se ubrzati na način da se u svakom koraku ne koristi cijeli skup podataka za računanje pogreške, već samo njegov manji dio (engl. *batch*) koji je različit u svakom koraku. Takvim načinom računanja pogreške ujedno izbjegavamo i računanje gradijenata za sve podatke. Ovaj postupak nazivamo stohastičkim gradijentnim spustom i on se pokazao efikasnijim pri izbjegavanju lokalnih minimuma od klasičnog gradijentnog spusta.

1.6. Optimizacijski algoritmi

Kako bi se postupak traženja minimuma funkcije gubitka gradijentnim spustom ubrzao, postoje različite varijante metode optimizacije koje to omogućuju. S obzirom na to da je proces učenja umjetne neuronske mreže uobičajeno vremenski zahtjevan, optimizacijski algoritmi imaju veliku ulogu jer smanjuju ukupno vrijeme treniranja mreže. Uz to, pronalazak minimuma funkcije može biti zahtjevniji od očekivanog zbog mogućnosti upadanja u regije sa slabim gradijentom ili u područja lokalnih minimuma. Budući da nam je cilj pronaći globalni minimum, pokušavamo pronaći onakvu metodu traženja minimuma koja će izbjeći situaciju u kojima se zaglavi na lokalnim minimumima. Uz lokalne minimume, prethodno spomenute regije sa slabim gradijentom su još veći problem na koji potencijalno nailazimo i koji pokušavamo izbjeći. Njega uzrokuju sedlaste točke (analogne točkama infleksije kod funkcija jedne varijable). U sedlastim točkama je iznos gradijenata

vrlo blizu nuli u svim smjerovima što otežava „izlazak“ iz takvih područja. Postoje brojni optimizacijski algoritmi, a u nastavku su nabrojani i ukratko opisani neki od njih (momentum, RMSprop i Adam) [2].

1.6.1. Učenje sa zaletom

Metoda zaleta ili momentum pospješuje ubrzanje gradijentnog spusta u slučaju kad imamo površine koje su više zakrivljene u jednom smjeru nego u ostalim smjerovima. Umjesto ovisnosti isključivo o gradijentima trenutnog prolaza, uzimaju se u obzir gradijenti iz trenutnog i prošlih koraka učenja neuronske mreže te se tako brže dolazi do konvergencije. Uzimanjem zamaha omogućuje se to da iako lokalni gradijent ima iznos nula, algoritam ne zaglavi iz razloga što se oslanja na vrijednosti gradijenata iz prethodnog koraka. Definicija ovog algoritma zadana je sljedećim izrazima:

$$\omega_{t+1} = \omega_t - \alpha m_t \tag{1.18}$$

$$m_t = \beta m_{t-1} + (1 - \beta)\frac{\partial L}{\partial \omega_t} \tag{1.19}$$

U navedenim jednadžbama m predstavlja kombinaciju trenutnog gradijenta i rekurzivnog poziva svih prošlih gradijenata pomnoženih s koeficijentom β koji predstavlja parametar zaglađenja i obično se postavlja na vrijednost 0.9. Koeficijent α označava stopu učenja [3].

1.6.2. RMSprop

RMSprop (engl. *Root Mean Squared Propagation*) također je jedan od algoritama optimizacije. RMSprop je adaptivan algoritam, što znači da ostvaruje individualno podešavanje stope učenja za svaki pojedini parametar modela umjesto korištenja jedne konstante za stopu učenja kroz cijeli proces. Podešavanje stope učenja odvija se automatski te se time nije potrebno zamarati. Također, prije ažuriranja vrijednosti parametara neuronske mreže, odgovarajuće komponente gradijenata se dijele s drugim korijenom sume kvadrata. Time se stopa učenja brže smanjuje kod parametara s velikim gradijentom, a sporije kod parametara s malim gradijentom. Na taj način RMSprop algoritam smanjuje oscilacije. Sljedeće formule opisuju definiciju algoritma:

$$\omega_{t+1} = \omega_t - \frac{\alpha}{\sqrt{v_t + \epsilon}} \cdot \frac{\partial L}{\partial \omega_t} \tag{1.20}$$

$$v_t = \beta v_{t-1} + (1 - \beta) \left[\frac{\partial L}{\partial \omega_t}\right]^2 \tag{1.21}$$

1.6.3. Adam

Adam algoritam (engl. *Adaptive Moment Estimation*) jedan je od novijih i u zadnje vrijeme sve popularnijih optimizatora. Važne njegove karakteristike su da nije računski zahtjevan te ne zauzima previše memorije u odnosu na druge algoritme optimizacije. Adam algoritam kombinira algoritme zaleta i RMSprop algoritam koji su prethodno opisani. Adam koristi kvadrirane gradijente za podešavanje stope učenja poput RMSprop algoritma. Konfiguracijski parametri algoritma Adam se uobičajeno postavljaju na β_1=0.9, β_2=0.999 i $\varepsilon=10^{-8}$. Obično se parametri β_1, β_2 i ε ne moraju podešavati jer navedeni iznosi generalno dobro funkcioniraju, a stopa učenja α je ta koja se podešava u svrhu ubrzavanja postupka učenja neuronske mreže [4]. U nastavku su prikazane matematičke formule koje opisuju Adam optimizacijski algoritam.

$$\omega_{t+1} = \omega_t - \frac{\alpha}{\sqrt{\hat{v}_t + \epsilon}} \cdot \hat{m}_t \tag{1.22}$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{1.23}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{1.24}$$

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) \frac{\partial L}{\partial \omega_t} \tag{1.25}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) \left[\frac{\partial L}{\partial \omega_t}\right]^2 \tag{1.26}$$

1.7. Regularizacija

Jedan od najvećih problema strojnog učenja koji je već objašnjen u ovom poglavlju je problem prenaučenosti neuronske mreže (engl. *overfitting*). Jedan od načina kojima se taj problem rješava su tehnike regularizacije. Regularizacijom se ograničava kapacitet modela. U nastavku su opisane najčešće korištene tehnike regularizacije.

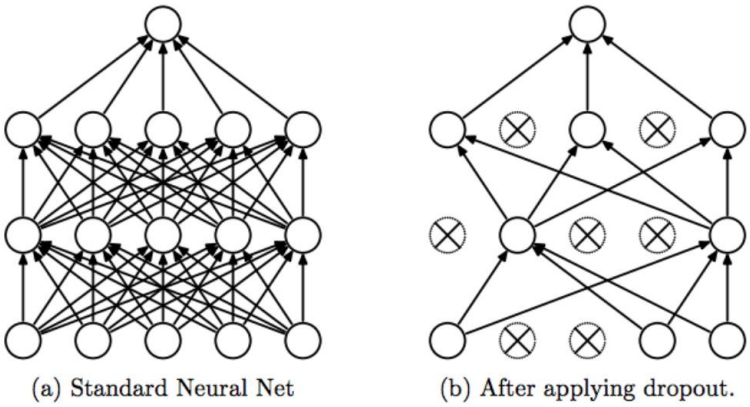
L1 i L2 regularizacija ažuriraju funkciju gubitka tako da joj dodaju regularizacijski član. Najčešći regularizacijski članovi gubitka su $L1$ i $L2$ norme koje se matematički zapisuju na sljedeći način, pri čemu je λ hiperparametar regularizacije, a L funkcija gubitka:

$$L1 = L + \lambda \sum_{j=1} |\omega| \tag{1.27}$$

$$L2 = L + \lambda \sum_{j=1} \omega_j^2 \tag{1.28}$$

Ovime se iznosi težina smanjuju jer se neuronske mreže s manjim iznosima težina smatraju jednostavnijima, a time i manje ranjivima na problem prenaučenosti. $L1$ ima svojstvo postavljanja težina na nulu, dok $L2$ tjera težine da idu prema nuli, ali ne točno u nulu.

Isključivanje neurona (engl. *dropout*) je tehnika regularizacije koja nasumično bira neki broj neurona u mreži te ih izbaci iz mreže, odnosno isključuje ih zajedno s njihovim poveznicama prema drugim neuronima mreže. Time svaka iteracija ima drugačiji skup neurona. Isključivanjem pojedinih neurona sprječava se da njihov utjecaj postane veći od poželjnog. Isključivanje neurona se može primijeniti na neurone skrivenih slojeva, ali i na neurone ulaznog sloja. Važan je utjecaj nasumičnosti pri izboru neurona za isključenje. Na slici 1.8 vizualno je prikazano pojednostavljenje mreže nakon isključivanja neurona.



Slika 1.8: Isključivanje neurona. (preuzeto dana 07.05.2020. s: https://miro.medium.com/max/1200/1*iWQzxhVlvadk6VAJjsgXgg.png)

Normalizacija grupe (engl. *batch normalization*) je tehnika kojom se normalizira izlaz prethodnog aktivacijskog sloja oduzimanjem srednje vrijednosti grupe i dijeljenjem sa standardnom devijacijom grupe. Normalizacija ima efekt stabilizacije procesa učenja i značajnog reduciranja potrebnog broja epoha pri učenju neuronske mreže [5].

Rano zaustavljanje je tehnika regularizacije kojom se sprječava prenaučenost tako da se zaustavi učenje neuronske mreže. Uobičajeno je kao provjeru kvalitete generalizacije koristiti validacijski skup te kada se primijeti da se generalizacija na validacijskom skupu pogoršava, odnosno kada se pogreška na validacijskom skupu počne povećavati, tada se prekida učenje.

Rastresanje podataka sprječava prenaučenost na način da se prilikom učenja dodaje šum slikama iz skupa za učenje. To je također vezano uz nasumično zrcaljenje, rotaciju i izrezivanje slike tako da samo njezin dio čini ulaz u mrežu. Izrezivanjem slike model može maknuti rubne dijelove slike, povećati objekte te izvaditi više korisnih značajki [6].

1.8. Propagacija pogreške unatrag

Propagacija pogreške unatrag je algoritam za brzo računanje gradijenta gubitka s obzirom na parametre modela. Ono što se prvo radi je unaprijedni prolazak kroz mrežu (engl. *feedforward*). Dakle, za svaki ulazni podatak iz skupa za učenje se izračuna izlaz mreže krećući se redom po slojevima mreže. Nakon što se dobije odziv mreže za određeni ulaz, izračunavaju se parcijalne derivacije gubitka s obzirom na aktivacije izlaznog sloja. Zatim

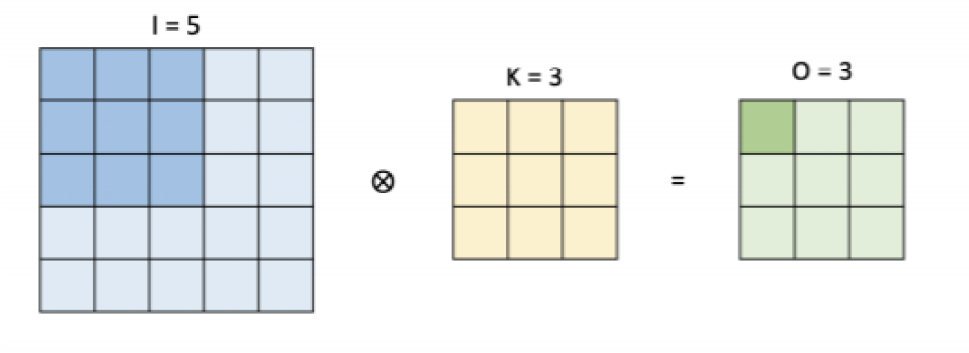
se iterativno računaju gradijenti gubitka prethodnih slojeva primjenom pravila ulančavanja. Ovakvom obradom izbjegava se višestruko računanje istih parcijalnih derivacija te se stoga algoritam propagacije pogreške unatrag može kvalificirati kao oblik dinamičkog programiranja. Opisani postupak ponavlja se za svaki sloj mreže gledajući unatrag, dakle od izlaznog prema ulaznom sloju. Jedan prolazak kroz mrežu unaprijed i unatrag za svaki podatak iz skupa za učenje čini jednu epohu pri treniranju neuronske mreže. No, uvjet zaustavljanja algoritma ne mora nužno biti određen brojem epoha, već je za uvjet zaustavljanja moguće postaviti dovoljno malu vrijednost pogreške.

2. Konvolucijske neuronske mreže

Tip umjetne neuronske mreže koji se najčešće koristi pri rješavanju problema iz područja računalnog vida i koja se pokazala boljom od ostalih jest konvolucijska neuronska mreža. Takva vrsta mreže se ne sastoji od potpuno povezanih slojeva, odnosno veze među neuronima su prorijeđene i neuroni susjednih slojeva nisu povezani svaki sa svakim. Ovo svojstvo neuronske mreže znatno smanjuje broj parametara koji može biti poprilično velik. Budući da su u području računalnog vida ulazne vrijednosti u mrežu svi pikseli jedne slike, zaključujemo da ulaza u mrežu ima $C \cdot H \cdot W$, pri čemu H i W predstavljaju redom visinu (engl. *height*) i širinu (engl. *width*), a C predstavlja broj kanala (engl. *channel*) koji je kod slika u RGB formatu jednak tri, a kod crno-bijelih slika jednak jedan (jednokanalne slike). Uočljivo je da na samom početku, na ulaznom sloju mreže, već imamo veliki broj parametara te iz toga možemo zaključiti da potpuno povezani slojevi nisu prikladno rješenje ovog problema. Iz tog razloga, konvolucijske neuronske mreže uvode konvolucijski sloj i sloj sažimanja, a potpuno povezani sloj (koji je već prethodno opisan) je uobičajeno posljednji sloj u mreži.

2.1. Slojevi u konvolucijskim mrežama

Konvolucijski slojevi (engl. *convolution layers*) su specifični za konvolucijske neuronske mreže. Parametri konvolucijskog sloja sastoje se od skupa filtera. Operacija konvolucije zapravo je prolazak filtera preko cijele ulazne slike. Filter podrazumijevano ima male veličine visine i širine. Prolaskom filtera po slici, vrijednosti piksela koji se u tom trenutku poklapaju s filterom skalarno se množe sa odgovarajućim vrijednostima samog filtera i proizvedu vrijednost jedne ćelije koja će biti dio izlaza iz sloja. Treba imati na umu da se nakon množenja na dobivenu vrijednost ćelije primjenjuje prijenosna funkcija koja daje konačan izlaz. Nakon toga, filter se pomiče udesno po ulaznoj slici za određeni broj piksela te se postupak množenja ponavlja. Kada filter prođe preko cijele slike, dobiva se mapa značajki određenih dimenzija, ovisno o veličini ulazne slike i filtera.

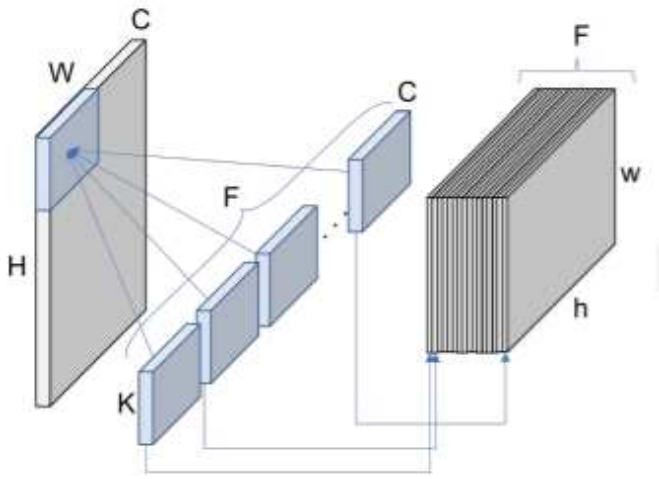


Slika 2.1: Konvolucija s filterom veličine 1x3x3. (preuzeto dana 10.05.2020. s: http://pabloruizruiz10.com/resources/CNNs/Convolution_Pooling.pdf)

Uzmimo za primjer sliku 2.1. Sa slike je vidljivo da je ulazna slika dimenzija 1x5x5 te da je veličina filtera 1x3x3. Ukoliko pomak udesno, odnosno prema dolje (engl. *stride*) u svakom koraku bude za samo jedan piksel, tada će izlaz iz sloja imati dimenzije 1x3x3. Ukoliko bi se u svakom koraku pomicali za $stride = 2$, tada bi dimenzije izlaza bile 1x2x2.

Često se događa da želimo da izlaz iz sloja ima jednaku visinu i širinu kao i ulaz. Kada bismo u svakom sloju značajno smanjivali dimenzije, u konačnici bi visina i širina izlaza postale vrlo male, što je neprikladno jer iz njih ne bi bilo moguće izvući potrebne značajke koje nas zanimaju. Kako bi se to spriječilo, koristi se tehnika popunjavanja nulama (engl. *zero-padding*). Time se na sve rubove ulazne slike nadodaju nule kako bi se visina i širina slike povećale za dva. Tada bi ulaz za primjer iz slike 2.1 bio dimenzija 1x7x7 te uz primjenu filtera istih dimenzija (1x3x3) uz korak = 1, dobije se izlaz dimenzija 1x5x5, što odgovara dimenzijama ulaza prije primjene tehnike popunjavanja nulama.

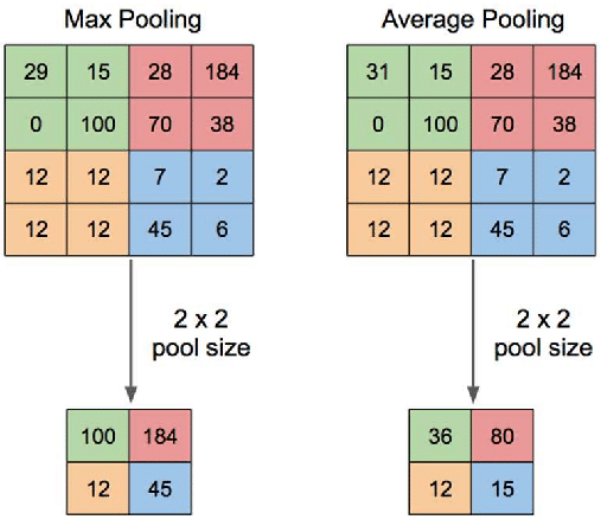
Važno je napomenuti da broj kanala, tj. dubina filtera, odgovara dubini ulaza, pa je dubina izlaza jednaka jedan. U konvolucijskom sloju se ne koristi samo jedan filter, već više njih. Budući da prolaskom jednog filtera kroz cijeli ulaz u sloj dobivamo jednu mapu značajki, zaključujemo da će broj mapa značajki na izlazu iz sloja odgovarati broju primijenjenih filtera. Vizualni prikaz toga vidljiv je na slici 2.2.



Slika 2.2: Vizualni prikaz konvolucije. (preuzeto dana 10.05.2020. s: http://pabloruizruiz10.com/resources/CNNs/Convolution_Pooling.pdf)

Svrha operacije konvolucije je ekstrahiranje i uočavanje značajki i uzoraka na slikama promatranjem odnosa bliskih neurona. Konvolucijske neuronske mreže sastoje se od većeg broja konvolucijskih slojeva.

Sloj sažimanja (engl. *pooling layer*) u konvolucijskim neuronskim mrežama ima svrhu reduciranja broja parametara neuronske mreže smanjenjem prostornih dimenzija ulaza (visine i širine). Time se umanjuju i prostorna i vremenska složenost algoritma učenja neuronske mreže. Postoje dva načina sažimanja, sažimanje maksimalnom vrijednošću (engl. *max pooling*) i sažimanje srednjom vrijednošću (engl. *average pooling*). Najčešće se sažimanje primjenjuje s filterom širine = 2 i visine = 2 te s korakom = 2 kako ne bi dolazilo do preklapanja. Filter prilikom sažimanja maksimalnom vrijednošću također prolazi preko cijelog ulaza, no umjesto operacije konvolucije u svakom koraku, izvlači se najveća vrijednost među onima koje se prostorno podudaraju s filterom. Sažimanje srednjom vrijednošću je vrlo slično, no umjesto odabira najveće vrijednosti, računa se srednja vrijednost među vrijednostima koje se u trenutnom koraku podudaraju s filterom. Slika 2.3 ilustrira opisani postupak. Sažimanje se događa na svim dubinama ulaza neovisno o ostalim dubinama.



Slika 2.3: Sažimanje maksimalnom (lijevo) i srednjom vrijednošću (desno). (preuzeto dana 10.05.2020. s: https://qph.fs.quoracdn.net/main-qimg-939c3123c48e27301f1a89c0a299dca8)

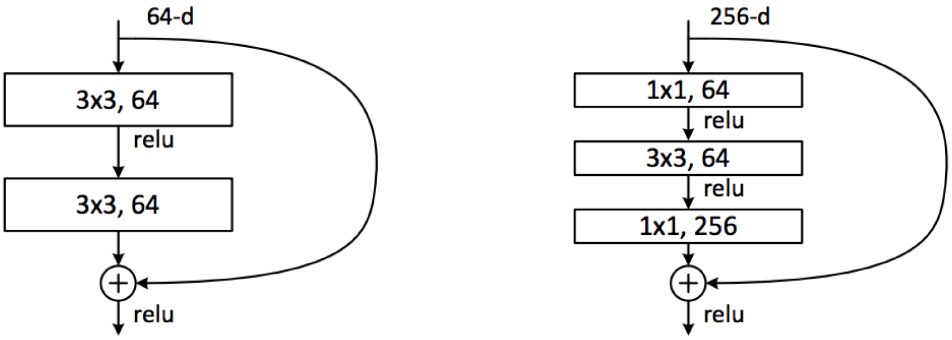
U konvolucijskim neuronskim mrežama najčešće se naizmjenično pojavljuju grupe konvolucijskih slojeva, a zatim sloj sažimanja. Na samom kraju neuronske mreže pojavljuje se potpuno povezani sloj.

2.2. Rezidualne neuronske mreže

Duboke neuronske mreže vrlo su uspješne pri rješavanju problema iz područja računalnog vida. Ipak, klasične neuronske mreže imaju nedostatke koji su prebrođeni pojavom rezidualnih neuronskih mreža. Jedan od nedostataka neuronskih mreža je problem sporog i dugotrajnog procesa učenja mreže. Rezidualne neuronske mreže su vrlo pogodne za probleme iz područja računalnog vida jer uče brže zbog glađeg gubitka [7].

Rezidualne neuronske mreže karakterizira uvođenje rezidualnih jedinica [8]. Rezidualne jedinice specifične su po tome što koriste prečice (engl. *skips*). Ulaz rezidualne jedinice se račva na dvije grane. Prva grana prolazi kroz određeni broj konvolucijskih slojeva, dok druga grana primjerice ostaje identična kao na samom ulazu. Zatim se grane ponovno spajaju na način da se izlazi iz obje grane zbroje. Grana koja je u opisanom primjeru ostala jednaka kao na ulazu naziva se prečicom i u ovom slučaju se nad njom primjenjuje samo funkcija identiteta, iako to inače ne mora nužno biti tako. Prilikom zbrajanja grana potrebno je pripaziti na kompatibilnost dimenzija dvaju izlaza koji se zbrajaju. Na slici 2.4

prikazana su dva primjera rezidualne jedinice. Riječ je o osnovnoj jedinici i jedinici s uskim grlom mreže ResNet-18.



Slika 2.4: Osnovna jedinica (lijevo) i jedinica s uskim grlom (desno). (preuzeto dana 10.05.2020. s: https://cdn-images-1.medium.com/max/1000/1*j_lC2gsO1Kbia8PIQGHUZg.png)

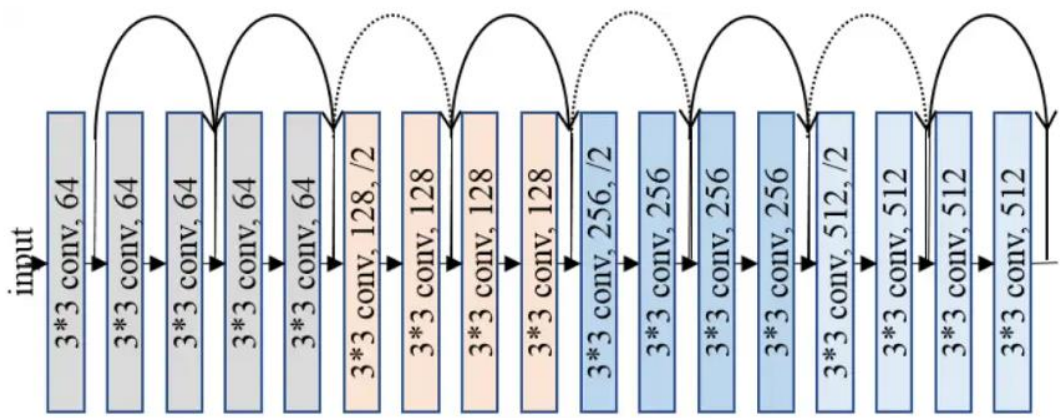
2.2.1. Arhitektura mreže ResNet-18

Eksperimentalni dio ovoga rada bavi se neuronskom mrežom ResNet-18. U nastavku je opisana arhitektura neuronske mreže radi lakšeg praćenja implementiranih promjena koje su opisane u četvrtom poglavlju. Prvi korak mreže ResNet-18 je primjena konvolucije s visinom i širinom filtera $7x7$ s korakom dva te uzastopnim popunjavanjem nulama tri puta. Ulaz u konvolucijski sloj čini RGB slika, dakle ulaz je dubine tri, dok je izlaz dubine 64. Nakon toga, primjenjuju se redom normalizacija, prijenosna funkcija ReLU (zglobnica) i zatim sažimanje maksimalnom vrijednošću filterom $3x3$ s korakom dva.

Idući koraci se odnose na prolazak kroz rezidualne blokove. Mreža ima četiri sloja koja dolaze redom jedan za drugim, a svaki sloj se sastoji od dva slijedno povezana rezidualna bloka. Kao što je već spomenuto, postoje dvije vrste blokova kod rezidualnih mreža, no u fokusu ovog rada bit će osnovni blok. U osnovnom bloku ulaz se grana na dvije grane od kojih ona koja predstavlja prečicu ostaje ista, dok se na drugoj primjenjuju dvije uzastopne konvolucije $3x3$ (između kojih se javlja prijenosna funkcija zglobnica) te se u konačnici izlazi iz dviju grana zbrajaju.

Nakon prolaska kroz drugi rezidualni blok, slijedi idući sloj koji se također sastoji od dva rezidualna bloka kao što je već spomenuto. Ipak, među četiri sloja se svaki put prilikom ulaska u novi sloj dvostruko povećava broj mapa značajki. Tako na početku pri ulasku u

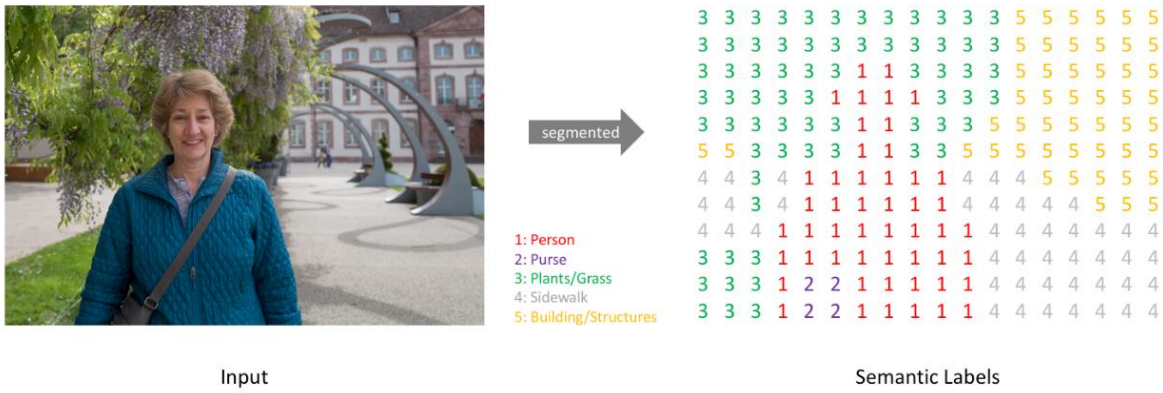
prvi sloj krećemo od 64 mapa značajki, a pri izlasku iz četvrtog sloja ih je 512. Naposljetku mreža ima još jedan sloj sažimanja (sažimanje srednjom vrijednošću) i potpuno povezani sloj na čijem je izlazu podrazumijevano 1000 mapa značajki. Opisana arhitektura mreže prikazana je na slici 2.5.



Slika 2.5: Arhitektura mreže ResNet-18. (preuzeto dana 10.05.2020. s: https://hackernoon.com/hn-images/1*uJ0IrP9JXYE2hHAzMjorQA.png)

3. Semantička segmentacija

Semantička segmentacija je jedan od problema kojima se bavi područje računalnog vida. Cilj semantičke segmentacije je odrediti što se sve nalazi na slici i gdje, odnosno za svaki piksel na slici odrediti pripadnost odgovarajućoj klasi. Pri tome se jedinke na slici ne razlikuju kao zasebni objekti, već su sve jedinke iste klase označene na jednak način. Dakle, ako na ulazu u mrežu primjerice predočimo sliku na kojoj se nalaze trava, nebo i osoba, na izlazu očekujemo matricu veličine ekvivalentne ulaznoj slici na kojoj su za svaki piksel slike odgovarajućim elementima matrice pridruženi određeni cijeli brojevi koji označavaju semantički razred kojemu piksel pripada. Primjer toga je na slici 3.1 na kojoj se radi lakšeg razumijevanja ne podudaraju rezolucije ulazne slike i izlaza, dok u stvarnosti to nije tako.



Slika 3.1: Primjer semantičke segmentacije. (preuzeto dana 10.05.2020. s: https://www.jeremyjordan.me/content/images/2018/05/Screen-Shot-2018-05-17-at-9.02.15-PM.png)

Izlaz iz svakog sloja konvolucijske neuronske mreže je neki broj mapa značajki koje zapravo služe kao izlučivači značajki (engl. *feature extractors*), a oni pri početnim slojevima prepoznaju jednostavnije značajke poput rubova i osnovnih uzoraka, a svaki sljedeći sloj prepoznaje kompleksnije značajke više razine koristeći izlaz iz prethodnog sloja konvolucijske neuronske mreže. Kako bi se zadržala ekspresivnost, obično se povećava broj mapa značajki što se ulazi dublje u mrežu. Pri tome se događa poduzorkovanje koje smanjuje složenost izračuna smanjenjem rezolucije prilikom

primjene operacija konvolucije i sažimanja. Spomenuto smanjenje rezolucije kod rješavanja problema klasifikacije ne predstavlja nepremostivi problem jer nas tada ne zanima gdje je objekt na slici nego samo kojoj klasi on pripada. Ipak, kod semantičke segmentacije to je drugačije jer na izlazu želimo dobiti predikciju iste visine i širine kao ulazna slika, odnosno želimo da izlaz bude pune rezolucije. Iz tog razloga se primjenjuju razni postupci naduzorkovanja (engl. *upsampling*) koji nastoje riješiti navedeni problem. Još jedan od pristupa u svrhu rješenja navedenog problema je izbjegavanje poduzorkovanja dilatiranjem, no to je lošiji pristup zbog većeg zauzeća memorije i sporijeg učenja [9].

Također, valja napomenuti da tik prije izlaska iz mreže, predikcija ima veličinu $C \cdot H \cdot W$,

pri čemu je dubina veličine ukupnog broja klasa u podatkovnom skupu. Naime, za svaku moguću klasu se stvara jedan izlazni kanal te se zatim funkcijom „*argmax*“, koja za svaki piksel bira onu klasu za koju je vjerojatnost pripadnosti najveća, dobiva izlaz dimenzija $1 \cdot H \cdot W$ [10].

Pri rješavanju problema semantičke segmentacije, različite vrste neuronskih mreža pokazale su zadovoljavajuće rezultate. Ovaj rad bavi se semantičkom segmentacijom uz primjenu rezidualne neuronske mreže ResNet-18 te usporedbom rezultata implementacijom ljestvičastog naduzorkovanja [9,11] nad istom mrežom, što će biti detaljnije objašnjeno u četvrtom poglavlju.

4. Programska implementacija

4.1. Korištene tehnologije

Za izradu eksperimentalnog dijela rada korišten je programski jezik Python te radno okruženje PyTorch koje je u posljednje vrijeme sve popularnije. Jedna od vrlo bitnih značajki PyTorcha je u tome što nudi mogućnost korištenja strukture podataka zvane tenzor (`torch.Tensor`) koja služi za jednostavnije manipuliranje multidimenzionalnim pravokutnim poljima brojeva. Tenzori nalikuju na NumPy polja (engl. *Arrays*), no njima se može upravljati i na grafičkoj procesnoj jedinici (GPU) koja podržava CUDA u svrhu veće brzine izvođenja. Uz to, PyTorch koristi automatsku diferencijaciju koja ima posebnu važnost pri računanju gradijenata parametara. Uz to, korištene su biblioteke NumPy za baratanje matricama, matplotlib i pandas za grafički prikaz, odnosno vizualizaciju rezultata te scikit-learn za izračun matrice zabune.

4.2. Sastav skupa podataka

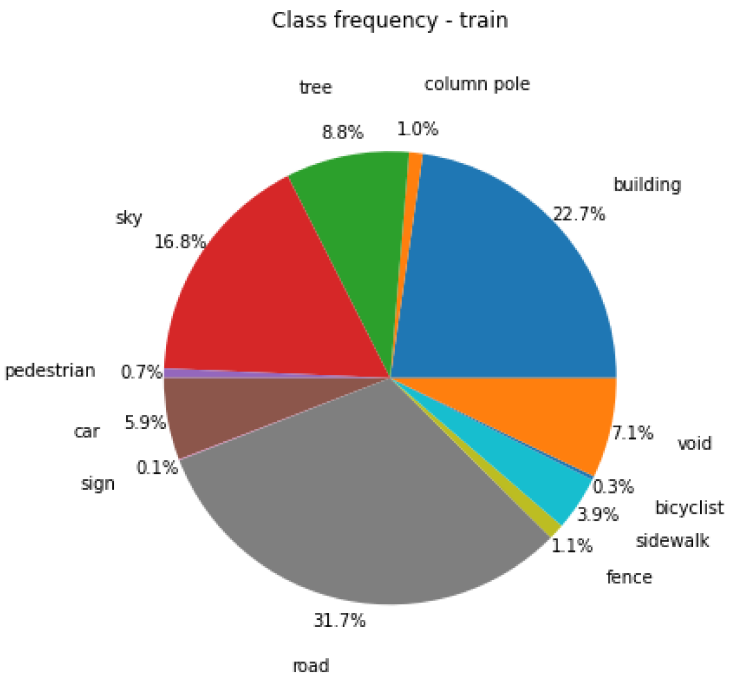
Za potrebe rada korišten je skup podataka (engl. *dataset*) CamVid koji se sastoji od 701 slike koje su dobivene iz video isječaka snimljenih iz vozačke perspektive za vrijeme gradske vožnje automobilom. Skup se sastoji od 32 klase od kojih jedanaest pripada u one osnovne koje se uobičajeno koriste. Dodatno, u radu je korištena još jedna klasa, „*void*“ u koju se preslikavaju preostale klase koje ne pripadaju u osnovnih jedanaest klasa. U jedanaest klasa pripadaju redom oznake na engleskom: *building, tree, sky, car, sign, road, pedestrian, fence, column pole, sidewalk, bicyclist.*

Pri učenju, validaciji i testiranju mreže, umjesto originalnog skupa označenih slika (engl. *target*) koji je trokanalan (RGB), korišten je već remapirani jednokanalni skup koji za svaki piksel slike ima vrijednost identifikacijskog broja (engl. *id*) odgovarajuće klase umjesto RGB vrijednosti kojima su klase „kodirane“. Ovakav remapirani skup podataka znatno ubrzava postupak treniranja neuronske mreže.

CamVid skup podataka podijeljen je na skup za učenje koji sadrži 367 slika, skup za validaciju sa 101 slikom te skup za testiranje s 233 slike. Slike imaju rezoluciju 960x720 i formata su .png.

Na slici 4.1 prikazan je primjer ulazne slike i pripadajućih semantičkih oznaka iz skupa podataka CamVid. Na slici 4.2 grafički je prikazan sastav skupa za učenje po udjelima klasa iz kojega se vidi koje su klase najzastupljenije, a koje najmanje zastupljene.

Slika 4.1: Primjer slike iz skupa podataka CamVid.



Slika 4.2: Udio pojedinih klasa u skupu podataka za učenje.

4.3. Arhitektura mreže za semantičku segmentaciju

Kao osnovni model bez ljestvičastog naduzorkovanja korištena je rezidualna neuronska mreža ResNet-18 uz manje modifikacije na samom kraju mreže, odnosno na zadnjim slojevima. Korištena je pretrenirana mreža, dakle inicijalizirana je parametrima naučenima na ImageNet-u. Izbačeni su posljednji globalni sloj sažimanja srednjom vrijednošću i potpuno povezani sloj. Umjesto njih dodan je konvolucijski sloj veličine filtera $1x1$ i s korakom veličine 1 koji na izlazu daje 12 kanala, što odgovara broju klasa u skupu CamVid. Naposljetku se provodi bilinearno naduzorkovanje kako bi dimenzije izlaza bile jednake kao i dimenzije ulaza.

Eksperiment je proveden na pola rezolucije, dakle na dimenzijama 480x360. Na ulaz modela stavljene su grupe (engl. *batches*) od četiri slike. Kao optimizator je korišten Adam s podrazumijevanim hiperparametrima. Učenje je provedeno na 40 epoha. Dobiveni su rezultati navedeni u petom poglavlju.

4.4. Arhitektura mreže za semantičku segmentaciju uz ljestvičasto naduzorkovanje

Arhitektura mreže je konstruirana po uzoru na neuronsku mrežu iz rada [9] na način da je napravljena nadogradnja rezidualne neuronske mreže ResNet-18 modificirane prema opisu iz prethodnog potpoglavlja na sljedeći način. Mreža je podijeljena u dvije podatkovne putanje, od kojih je prva putanja identična putanji koju sadrži model opisan u prethodnom potpoglavlju, dakle modificirani ResNet-18. Druga je pak putanja sastavljena od TransitionUp blokova implementiranih po uzoru na opis iz radova [12] i [9] čija je uloga da obavljaju postupno naduzorkovanje ekstrahiranih mapa značajki. Primjenjuje se ljestvičasta arhitektura preskočnih veza, izgledom nalik na ljestve, te se povezuju mape značajki iz prve putanje s odgovarajućim mapama značajki druge putanje. Povezivanje se obavlja tako da se prvenstveno nad mapama značajki iz prve putanje (izlazi iz svakog od četiri sloja) primijeni konvolucija s filterom veličine 1 koja smanjuje broj mapi značajki iz prve putanje. Broj mapi značajki iz prve putanje se mora podesiti tako da odgovara broju mapi značajki iz druge putanje, te da na taj način oba ulaza u TransitionUp blok imaju jednaki broj značajki. Nakon toga se izlaz iz prethodnog TransitionUp bloka poveća

bilinearnom interpolacijom tako da oba ulaza zadovoljavaju uvjet da su jednakih prostornih dimenzija. Zatim se oba ulaza zbrajaju, što je moguće jer su im nakon primjene transformacija dimenzije postale usklađene. Naposljetku se nad dobivenim zbrojem primjenjuje $3x3$ konvolucija čija je uloga izvlačenje značajki iz svih dosad skupljenih informacija na trenutnoj rezoluciji. U nastavku je prikazan kôd TransitionUp bloka.

```
1.  class TransUp(nn.Module):
2.    def __init__(self, in_channels, out_channels):
3.      super().__init__()
4.      self.norm1 = nn.BatchNorm2d(in_channels)
5.      self.relu = nn.ReLU(inplace=True)
6.      self.conv1x1 = nn.Conv2d(in_channels, out_channels,kernel_size=1)
7.      self.norm2 = nn.BatchNorm2d(out_channels)
8.      self.conv3x3 = nn.Conv2d(out_channels, out_channels,kernel_size=3, padding=1)

9.      return
10.
11.   def forward(self, x, skip):
12.     skip_b, skip_c, skip_h, skip_w = skip.size()
13.     skip = self.conv1x1(self.relu(self.norm1(skip)))
14.     out = nn.functional.interpolate(x, size=(skip_h, skip_w),mode="bilinear",
15.                                     align_corners=True)
16.     out = torch.add(out,skip)
17.     out = self.conv3x3(self.relu(self.norm2(out)))
18.     return out
```

Ovakvom implementacijom putanje naduzorkovanja, mreža ima mogućnost uvida u mape značajki iz nižih slojeva mreže (koje zapažaju sitne detalje na slici poput rubova ili manjih dijelova većeg objekta) te u mape značajki širokog konteksta (koje uočavaju informacije o slici kao cjelini) [12]. Nakon izlaza iz posljednjeg TransitionUp bloka potrebna je primjena bilinearne interpolacije da bi se konačan izlaz iz mreže visinom i širinom podudarao s rezolucijom ulaza.

4.5. Priprema i pristup podacima

Učitavanje, pretvorba, obrada i transformacija podataka često mogu biti vremenski zahtjevni te iz tog razloga posežemo za jednostavnijim rješenjem kojega PyTorch nudi. Naime, postoji standardna konvencija interakcije s podacima koja obuhvaća klase „*Dataset*“ i „*DataLoader*“.

Klasa „*Dataset*“ omogućuje dohvaćanje podataka koji će se koristiti pri učenju, validaciji i testiranju umjetne neuronske mreže. Prilikom programiranja klase pogodne za naš skup

podataka, potrebno je zadovoljiti uvjete koji su propisani apstraktnom klasom „*Dataset*“. Neophodno je implementirati metodu `__len__(self)` koja vraća veličinu skupa podataka te metodu `__getitem__(self, index)` koja dohvaća podatke (u ovom slučaju slike) iz skupa podataka. Uobičajeno se nad podacima primjenjuje niz transformacija prije nego što se predaju mreži. U ovom radu se nad slikama redom provode transformacija smanjivanja slike na pola rezolucije, pretvorba slike u tenzor te normalizacija tenzora oko određene srednje vrijednosti i standardne devijacije. Normalizacija je vrlo bitna iz razloga što će prolaskom ulaznog tenzora kroz mrežu mnogo puta biti primijenjena operacija množenja. Zbog toga za vrijeme učenja mreže vrijednosti mogu postati ekstremno velike, što je poznato kao problem eksplodirajućih gradijenata. Primjenom normalizacije, dolazeće vrijednosti se mogu držati pod kontrolom. Za iznose srednje vrijednosti i standardne devijacije uzimaju se konstante definirane prema ImageNetu. Nad labelama se također provodi smanjivanje na pola rezolucije i pretvorba u tenzor, te nakon toga interpolacija metodom najbližeg susjeda (engl. *nearest neighbour*).

„*DataLoader*“ klasa pak ima ulogu prosljeđivanja podataka mreži na određeni način, primjerice moguće je odrediti koliko procesa (engl. *worker processes*) predočava podatke mreži, želimo li izmiješati podatke prilikom njihovog prosljeđivanja u mrežu te hoćemo li podatke dohvaćati u grupama, i ako da, koja će biti veličina grupe [13].

5. Rezultati

Pri računanju uspješnosti razvijenog modela neuronske mreže postoje različite mjere koje pokazuju rezultat dobiven testiranjem neuronske mreže. Model ResNet-18 bez i sa ljestvičastim naduzorkovanjem evaluiran je na skupu podataka CamVid opisanom u četvrtom poglavlju. Kao mjera točnosti korišteni su točnost piksela te srednji omjer presjeka i unije.

5.1. Mjere

Točnost piksela (engl. *pixel accuracy*) jedna je od najjednostavnijih mjera te predstavlja omjer točno klasificiranih piksela naspram ukupnog broja piksela u skupu podataka. Matematički, točnost piksela se računa na sljedeći način:

$$točnost = \frac{broj\ točno\ predviđenih\ piksela}{ukupni\ broj\ piksela} \tag{5.1}$$

Omjer presjeka i unije (engl. *intersection over union*) često je realnija mjera uspješnosti modela zbog toga što je udio pojedinih razreda u skupu podataka često neravnomjeran te dolazi do problema ako se to ne uzima u obzir. Primjerice, ukoliko neke klase dominiraju na slici, a neke se javljaju u malom postotku, moguće je da će točnost piksela biti vrlo visoka iako model uopće ne prepoznaje manje zastupljene klase. Iz toga razloga posežemo za drugim mjerama poput srednjeg omjera presjeka i unije. Jednostavno rečeno, omjer presjeka i unije predstavlja površinu poklapanja između slike koju je mreža predvidjela i točnog (očekivanog) izlaza podijeljenu s površinom unije predviđene slike i točne slike. Mjera je u intervalu od 0 do 1, odnosno 0% - 100% pri čemu nula signalizira da nema podudaranja dok 1 označava savršeno poklapanje.

Srednji omjer presjeka i unije (engl. *mean intersection over union*) dobiva se zbrajanjem omjera presjeka i unije svake klase i dijeljenjem s ukupnim brojem klasa te se matematički izražava kao:

$$mIoU = \frac{1}{C}\sum_{c=1}^{C}\frac{|Y_{c-pred} \cap Y_{c-true}|}{|Y_{c-pred} \cup Y_{c-true}|} \tag{5.2}$$

Pri tome C predstavlja broj klasa unutar skupa podataka, Y_{c-pred} onaj skup piksela za koji je neuronska mreža predvidjela da pripadaju klasi c, a Y_{c-true} skup piksela koji zaista pripadaju klasi c prema oznakama iz skupa podataka.

5.2. Matrica zabune

Matrica zabune (engl. *confusion matrix*) je matrica dimenzija CxC pri čemu je C broj mogućih klasa u skupu podataka. Retci matrice predstavljaju stvarne klase kojima pojedini pikseli pripadaju, dok stupci označavaju predviđene klase. Dakle, u ćelije matrice se u odgovarajući redni broj retka i stupca upisuje broj piksela ovisno o tome koja je njegova stvarna klasa te u koju klasu ga je neuronska mreža smjestila. Za hipotetski model koji savršeno klasificira sve piksele vrijedilo bi da su sve ćelije osim onih na dijagonali matrice ispunjene nulama. Što je naš rezultat bliži onom savršenom, dakle što je više vrijednosti na dijagonali matrice, to je model uspješniji. Iz matrice zabune se vrlo lako može zaključiti koje je klase model dobro naučio prepoznavati, koje klase lošije prepoznaje te miješa li pojedine klase jednu s drugom.

5.3. Rezultati na skupu podataka CamVid

Prvo je trenirana neuronska mreža bez ljestvičastog naduzorkovanja te je učenje provedeno u 40 epoha s grupama za učenje veličine 4. Sav kod pokretan je preko Google Colab platforme koja omogućuje izvođenje na grafičkoj kartici. Dobiveni rezultati koje mreža postiže izraženi su preko mjera točnosti piksela i omjera presjeka i unije i prikazani su u tablici 1. Nakon toga, trenirana je mreža s ljestvičastim naduzorkovanjem te su i njeni rezultati navedeni u tablici 1.

Tablica 1: Prikaz dobivenih rezultata za oba modela.

Prikaz dobivenih rezultata	ResNet-18 bez ljestvičastog naduzorkovanja		ResNet-18 s ljestvičastim naduzorkovanjem	
	točnost piksela	srednji omjer presjeka i unije	točnost piksela	srednji omjer presjeka i unije
Skup za učenje	93.10%	66.82%	96.14%	79.19%
Skup za validaciju	89.31%	53.51%	91.50%	60.01%
Skup za testiranje	82.44%	44.40%	85.97%	52.37%

Dobivena točnost piksela uspoređena je s rezultatima prijašnjih radova koji su koristili isti skup podataka. Također, napravljen je eksperiment radi usporedbe s rezultatima iz rada [11] u kojemu je korištena mreža SwiftNet koja je trenirana na skupu podataka Cityscapes. S obzirom na veliku sličnost između klasa unutar skupa Cityscapes i CamVid, te uzevši u obzir da Cityscapes ima veći broj klasa, možemo zaključiti da bi mreža SwiftNet potencijalno mogla davati dobre rezultate i na skupu CamVid. Iz tog razloga napravljen je eksperiment na način da je implementacija mreže SwiftNet s već podešenim parametrima dobivenima treniranjem na skupu Cityscapes iskorištena za testiranje na skupu CamVid. Promijenjeni su dijelovi koda kako bi mreža učitavala podatke iz skupa CamVid umjesto Cityscapes te je dodan programski kod za mapiranje klasa iz skupa Cityscapes u klase skupa CamVid.

Mreža SwiftNet je također, kao i na skupu Cityscapes, trenirana na skupu podataka CamVid te je dobiven srednji omjer presjeka i unije od 63.33% na testnom skupu što je navedeno u radu [11]. No, mreža SwiftNet koja je trenirana na skupu Cityscapes, a eksperimentalno testirana na skupu CamVid kao što je opisano u prethodnom odlomku, daje srednji omjer presjeka i unije od 59.09% na testnom skupu. Razlika u rezultatima se javlja iz razloga što učenje na skupu Cityscapes rezultira prenaučenošću na model kamere kojom je sniman taj skup te su zato rezultati za skup CamVid lošiji.

U nastavku su tablično prikazani svi prethodno spomenuti rezultati na testnom skupu CamVid iskazani u postotcima (tablica 2).

Tablica 2: Usporedba dobivenih rezultata s prijašnjim radovima.

Neuronska mreža	Srednji omjer presjeka i unije
LDN121 16→2 DenseNet [9]	78.1
Tiramisu DenseNet [14]	66.9
SwiftNet (ResNet-18, single scale, učenje na skupu CamVid) [11]	63.33
SwiftNet (ResNet-18, single scale, učenje na skupu Cityscapes)	59.09
ResNet-18 bez ljestvičastog naduzorkovanja	44.4
ResNet-18 s ljestvičastim naduzorkovanjem	52.37

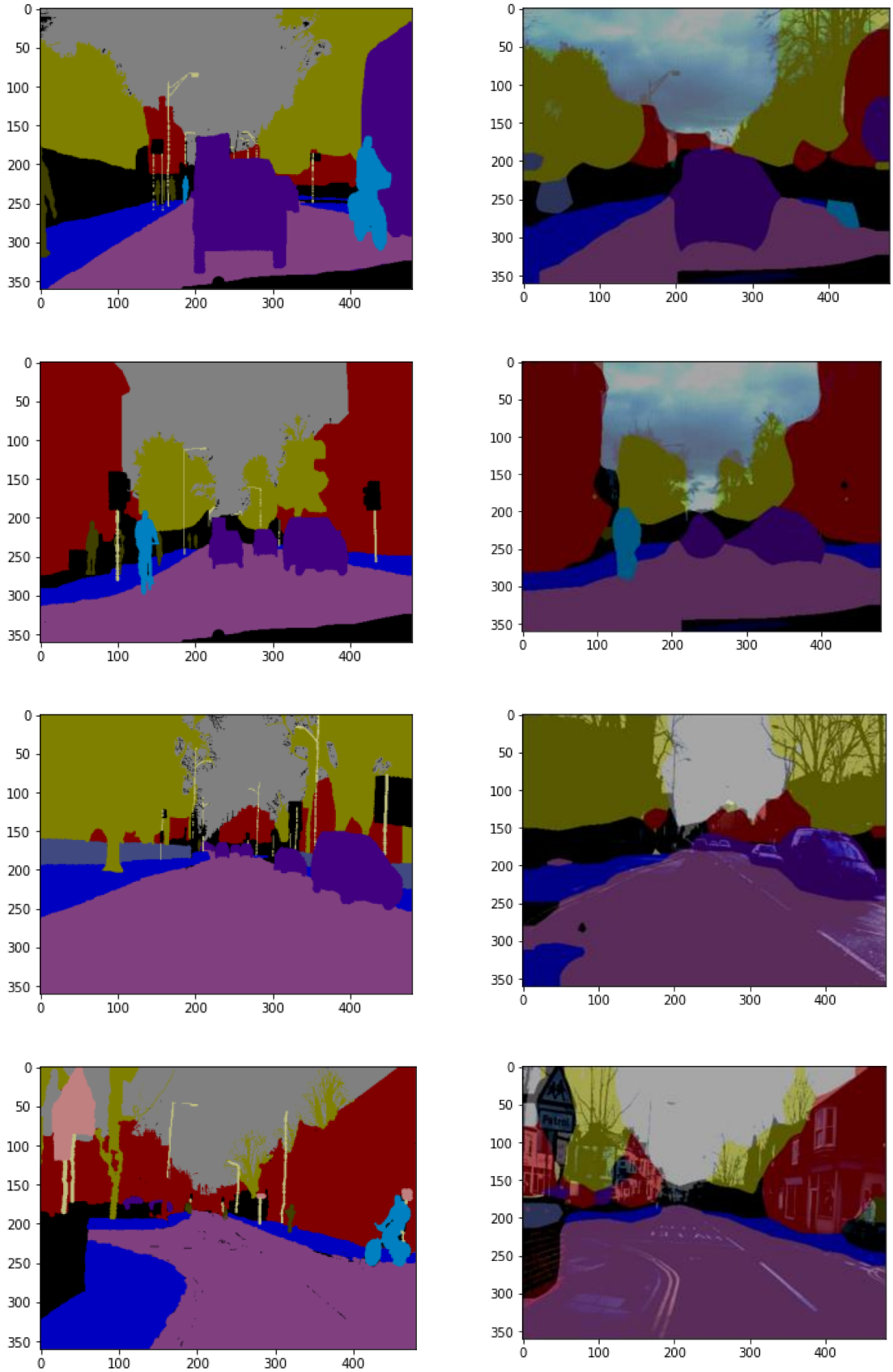
Na slici 5.1 je prikazana matrica zabune dobivena na skupu za testiranje mrežom ResNet-18 uz ljestvičasto naduzorkovanje. Iz matrice zabune vidi se da neuronska mreža odlično prepoznaje zgrade, drveće, nebo, automobile, pločnik i pozadinu, dok najbolje prepoznaje cestu. Bicikliste, pješake i ogradu mreža prepoznaje nešto lošije, dok najvećih problema pri zaključivanju mreža ima za klase prometnog znaka i stupa.

Ukoliko usporedimo to sa slikom 4.2 iz prethodnog poglavlja na kojoj je prikazan udio pojedinih razreda u skupu za treniranje, možemo zaključiti da model generalno bolje prepoznaje klase koje su više zastupljene u skupu podataka. Potencijalni uzrok tomu je činjenica da neuronska mreža ne koristi funkciju gubitka koja uzima u obzir učestalost klasa u skupu podataka te iz tog razloga nema tendenciju zaključiti da pojedini skup piksela pripada manje zastupljenoj klasi.

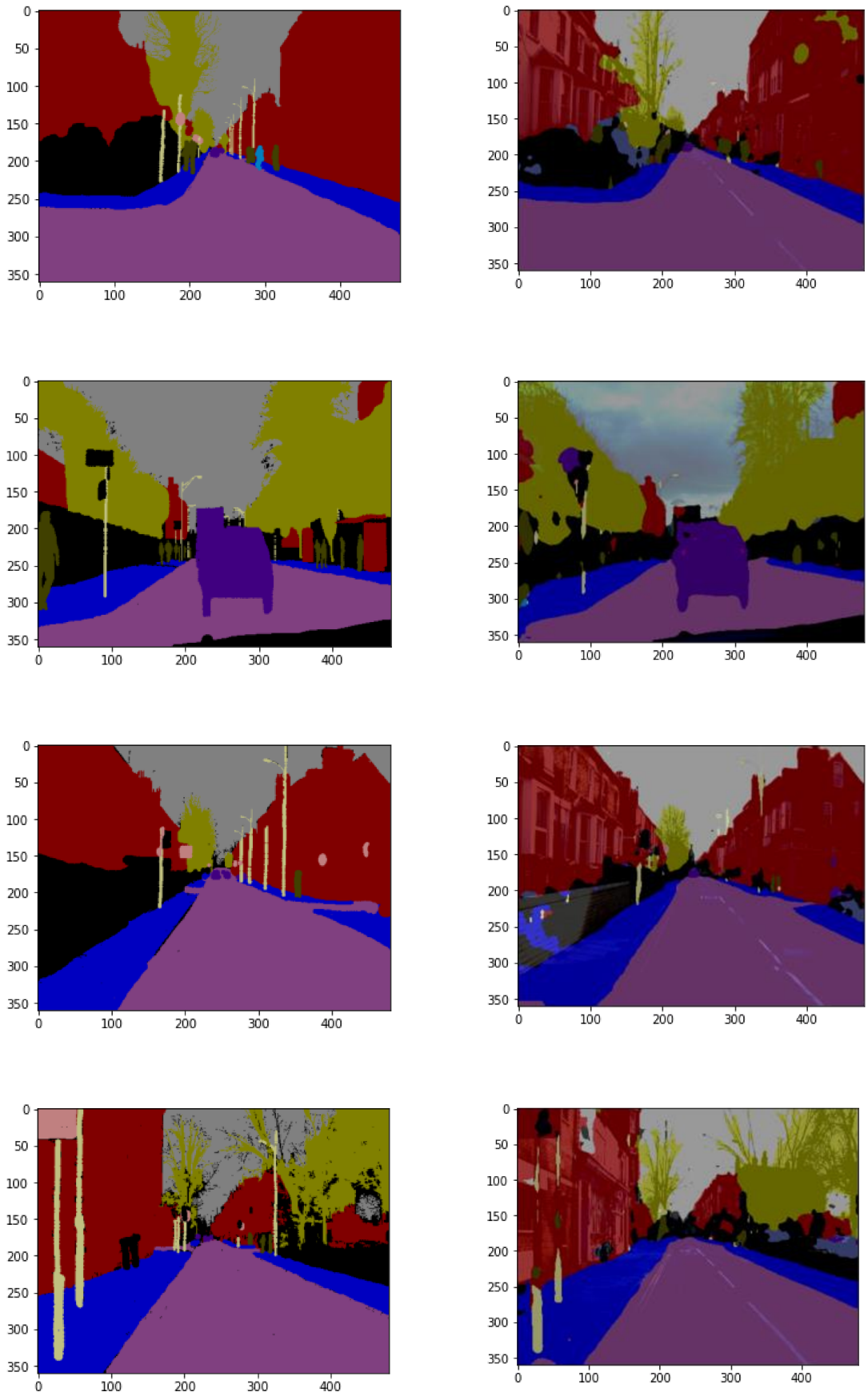
Slika 5.1: Matrica zabune modela ResNet-18 s ljestvičastim naduzorkovanjem za testni skup.

Usporedbom rezultata dobivenih korištenjem mreže bez ljestvičastog naduzorkovanja te s njim dolazimo do zaključka da model s ljestvičastim naduzorkovanjem daje bolje rezultate. Na slikama 5.2 i 5.3 prikazani su izlazi iz oba modela uspoređeni sa stvarnim oznakama. Zbog kombiniranja značajki viših i nižih slojeva pri ljestvičastom naduzorkovanju postiže se očuvanje sitnih detalja i objekata što u konačnici povećava točnost. Na slici 5.4 vidljivo je da uz korištenje ljestvičastog naduzorkovanja dobivene slike imaju oštrije rubove i jasnije definirane prijelaze između dviju klasa, dok mreža koja nema ljestvičasto naduzorkovanje daje općeniti izgled scene uz rubove koji izgledaju oblo, te se oblici ne razaznaju jasno kao oni na izlazu mreže s ljestvičastim naduzorkovanjem.

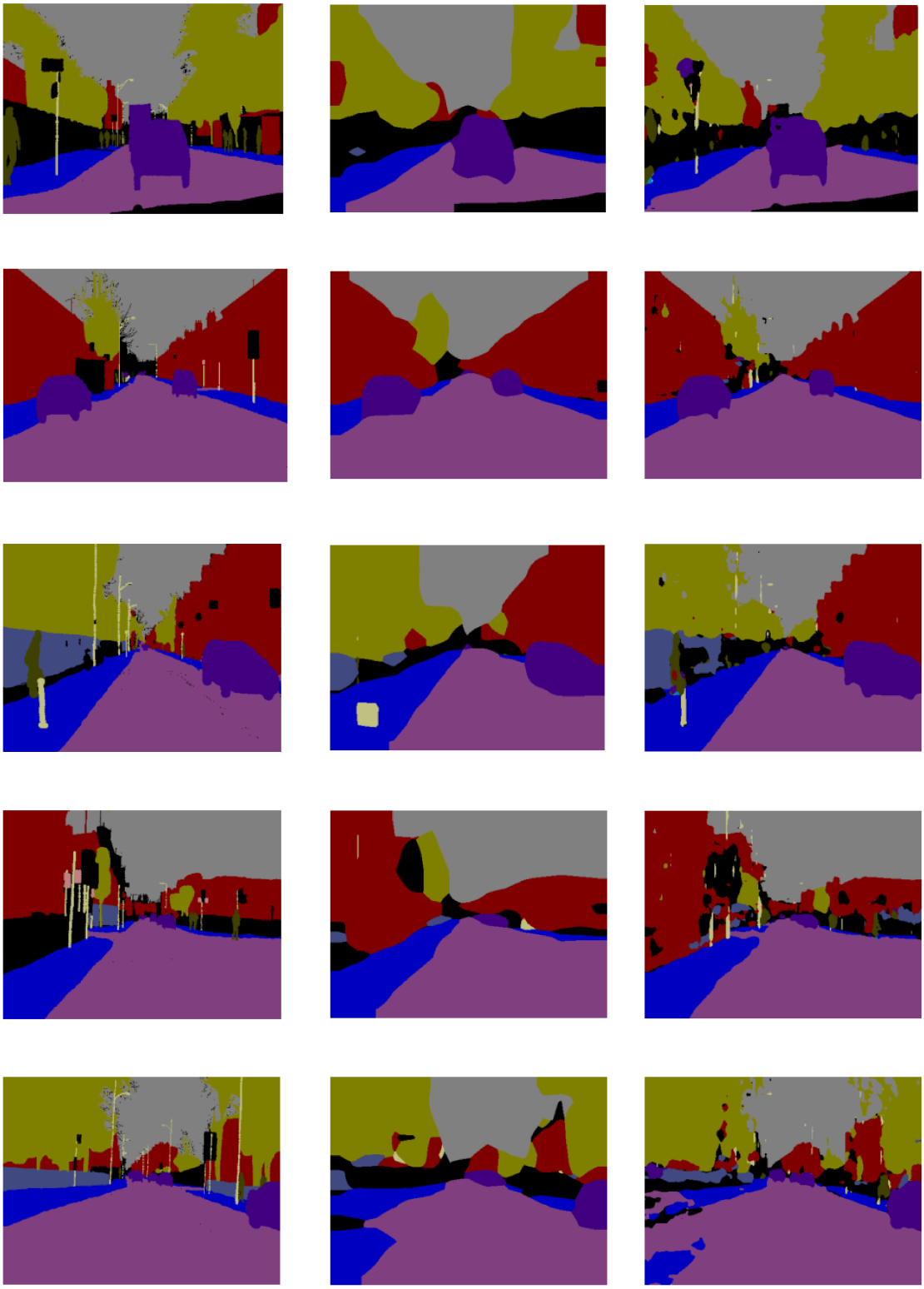
Buduća nadogradnja modela potencijalno bi mogla koristiti drugačiju funkciju gubitka, odnosno onakvu koja pri izračunu uzima u obzir nejednake udjele pojedinih razreda u podatkovnom skupu. Također, treba imati na umu da je skup CamVid relativno malen u odnosu na uobičajene skupove podataka te bi umjetno proširivanje skupa podataka, odnosno dodavanje novih slika moglo rezultirati većom točnošću piksela i većim omjerom presjeka i unije. Dodavanje novih slika moglo bi se postići primjenom različitih transformacija nad postojećim slikama te njihovim umetanjem u originalni skup podataka. Transformacije koje bi se mogle provesti su primjerice zrcaljenje (horizontalno i vertikalno preokretanje slika) i translacija (izrezivanje slika i njihovih pripadnih oznaka na nasumično odabranim mjestima).



Slika 5.2: Prikaz rezultata modela ResNet-18 bez ljestvičastog naduzorkovanja.



Slika 5.3: Prikaz rezultata modela ResNet-18 s ljestvičastim naduzorkovanjem.



Slika 5.4: Usporedba stvarnih oznaka te izlaza iz modela bez i sa ljestvičastim naduzorkovanjem redom.

Zaključak

Ovaj rad se bavi jednim od glavnih problema računalnog vida, semantičkom segmentacijom, čija je svrha pridjeljivanje pripadajuće klase svakom pikselu na slici. U radu su objašnjeni osnovni pojmovi vezani uz umjetne neuronske mreže, koncepti prijenosnih funkcija, funkcija gubitka, gradijentnog spusta, optimizacije, regularizacije i propagacije pogreške unatrag.

U posljednje vrijeme vrlo dobre rezultate u rješavanju problema semantičke segmentacije postižu rezidualne neuronske mreže koje zahvaljujući rezidualnim blokovima s prečicama omogućavaju brzo učenje vrlo dubokih modela. Konvolucijski slojevi mreže služe kao ekstraktori značajki sa slika, a slojevi sažimanja osiguravaju smanjenje broja parametara mreže. U radu je objašnjena arhitektura rezidualne neuronske mreže ResNet-18.

Cilj rada bio je ostvarenje modela neuronske mreže koja će služiti za razumijevanje scene iz perspektive vozača u prometu. Za to je korištena mreža ResNet-18 uz dodavanje modifikacija potrebnih za rješenje problema semantičke segmentacije. Programska implementacija izvedena je korištenjem biblioteke PyTorch koja omogućuje jednostavnu izvedbu korištenjem ugrađenih modula. Implementirana mreže trenirana je i zatim testirana na skupu podataka CamVid. Takav osnovni model mreže nadograđen je dodavanjem ljestvičastog naduzorkovanja koje kombiniranjem mapi značajki iz viših i nižih slojeva uspijeva očuvati sitne detalje i male objekte na slikama.

U radu je pokazano kako se bolji rezultati, odnosno veća točnost i veća vrijednost srednjeg omjera presjeka i unije dobivaju korištenjem ljestvičastog naduzorkovanja. Izlazi iz mreže s ljestvičastim naduzorkovanjem znatno su precizniji zbog oštrine rubova među prijelazima između pojedinih klasa na slici, dok izlazi iz osnovnog modela mreže imaju vrlo blage prijelaze i oblici se lošije razaznaju. Važne prednosti primjene ljestvičastog naduzorkovanja su veća brzina, jednostavnost i manji zahtjevi za memorijom u odnosu na druge tehnike naduzorkovanja.

Literatura

[1] S. Sharma, “Activation Functions in Neural Networks” https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6, datum pristupa: 28.05.2020.

[2] S. Ruder, “An overview of gradient descent optimization algorithms,” CoRR vol.abs/1609.04747, 2017.

[3] V. Bushaev, “Stochastic Gradient Descent with momentum” https://towardsdatascience.com/stochastic-gradient-descent-with-momentum-a84097641a5d, datum pristupa: 12.05.2020.

[4] D. P. Kingma and J. L. Ba, “ADAM: A method for stochastic optimization,” ArXiv e-prints, 2017.

[5] S. Santurkar, D. Tsipras, A. Ilyas, and A. Madry, “How Does Batch Normalization Help Optimization?,” CoRR, vol.abs/1805.11604, 2019.

[6] D. Miko, “Duboki konvolucijski modeli za lokalizaciju objekata,” Diplomski rad, Sveučilište u Zagrebu Fakultet elektrotehnike i računarstva, 2018.

[7] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein, “Visualizing the Loss Landscape of Neural Nets,” The Thirty-second Annual Conference on Neural Information Processing Systems (NIPS), 2018.

[8] K. He, X. Zhang, S. Ren, and J. Sun, “Identity Mappings in Deep Residual Networks,” The 14th European Conference on Computer Vision, 2016.

[9] I. Kreso, J. Krapac, and S. Segvic, “Efficient ladder-style densenets for semantic segmentation of large images,” CoRR, vol. abs/1905.05661, 2019.

[10] J. Jordan, “An overview of semantic image segmentation.” https://www.jeremyjordan.me/semantic-segmentation/, datum pristupa: 20.05.2020.

[11] M. Orsic, I. Kreso, P. Bevandic, and S. Segvic, “In Defense of Pre-trained ImageNet Architectures for Real-time Semantic Segmentation of Road-driving Images,” CoRR, vol. abs/1903.08469v2, 2019.

[12] L. Ivković, “Semantička segmentacija osoba konvolucijskim modelima,” Diplomski rad, Sveučilište u Zagrebu Fakultet elektrotehnike i računarstva, 2019.

[13] M. Avendi, “PyTorch Computer Vision Cookbook,” 1. Izadnje, Birmingham: Packt Publishing Ltd., 2013.

[14] S. Jégou, M. Drozdzal, D. Vázquez, A. Romero, and Y. Bengio, “The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation,” CoRR, vol. abs/1611.09326, 2016.

[15] S. Saha, “A Comprehensive Guide to Convolutional Neural Networks – the ELI5 way“ https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53, datum pristupa: 10.05.2020.

[16] E. Shelhamer, J. Long, and T.Darrell, “Fully Convolutional Networks for Semantic Segmentation,“ CoRR vol.abs/1605.06211, 2016.

[17] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,“ IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015.

[18] M.D. Zeiler and R. Fergus, “Visualizing and Understanding Convolutional Networks,“ CoRR vol.abs/1311.2901, 2013.

[19] J. Schmidhuber, “Deep Learning in Neural Networks: An Overview,“ CoRR vol.abs/1404.7828, 2014.

Sažetak

Razumijevanje scene iz perspektive vozača semantičkom segmentacijom

U ovom radu objašnjen je princip funkcioniranja umjetnih neuronskih mreža s naglaskom na konvolucijske neuronske mreže. U radu su opisane rezidualne neuronske mreže kao i njihova implementacija na primjeru mreže ResNet-18. Programski je ostvareno rješenje problema raspoznavanja prometnih sudionika te elemenata okoliša iz perspektive vozača. Uz osnovni model mreže ResNet-18, programski je ostvarena i nadogradnja modela ljestvičastim naduzorkovanjem. Napravljena je usporedba rezultata koje daju oba modela te su oni navedeni na kraju rada.

Ključne riječi: neuronske mreže, konvolucijske neuronske mreže, ResNet, semantička segmentacija, ljestvičasto naduzorkovanje

Abstract

Scene understanding from the driver's perspective with semantic segmentation

This paper explains the principle of functioning of artificial neural networks with emphasis on convolutional neural networks. The paper describes residual neural networks as well as their implementation on the example of ResNet-18 network. The problem of recognizing traffic participants and elements of the environment from the perspective of the driver was solved programatically. In addition to the basic model of the ResNet-18 network, the model was also upgraded with ladder upsampling. A comparison of the results given by both models is made and they are listed at the end of the paper.

Keywords: neural networks, convolutional neural networks, ResNet, semantic segmentation, ladder upsampling