SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 483

Semantička segmentacija harmoničkim gusto povezanim modelima

Anto Matanović

Zagreb, srpanj 2022.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 11. ožujka 2022.

ZAVRŠNI ZADATAK br. 483

Pristupnik:	**Anto Matanović (0036525570)**
Studij:	Elektrotehnika i informacijska tehnologija i Računarstvo
Modul:	Računarstvo
Mentor:	prof. dr. sc. Siniša Šegvić

Zadatak: **Semantička segmentacija harmoničkim gusto povezanim modelima**

Opis zadatka:

Semantička segmentacija važan je zadatak računalnog vida s mnogim zanimljivim primjenama. U posljednje vrijeme vrlo zanimljive rezultate postižu modeli s gusto povezanim konvolucijskim slojevima. Nedavno se pojavila poboljšana organizacija koja postiže veću učinkovitost obrade izbjegavanjem memorijski intenzivnih konvolucija. 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 semantičlu segmentaciju 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. Vrednovati naučene modele, prikazati postignutu točnost te provesti usporedbu s rezultatima iz literature. Komentirati učinkovitost učenja i zaključivanja. Predložiti pravce budućeg razvoja. 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: 10. lipnja 2022.

Zahvaljujem se svojoj obitelji na neizmjernoj podršci. Zahvaljujem se svome mentoru Siniši Šegviću na strpljenju, savjetima i pomoći prilikom izrade rada.

SADRŽAJ

1. Uvod

Umjetna inteligencija je jedno od glavnih istraživačkih područja u današnje vrijeme. Neki zadatci koji su naime vrlo jednostavni čovjeku, računalu predstavljaju problem. S porastom računalne snage dolazi i do povećanja popularnosti dubokog učenja jer se kompleksni duboki modeli mogu trenirati u razumnom vremenu. [4] Računalni vid je područje dubokog učenja koje istražuje kako funkcionira ljudski vid. To je grana dubokog učenja koja koristi duboke konvolucijske mreže, a bavi zadatcima poput klasifikacije objekata i semantičke segmentacije. Posao klasifikacije jest odrediti koju oznaku klase pridijeliti ulaznoj slici dok se semantička segmentacija bavi raspoznavanjem objekata na slici tako da se svakom pikselu dodijeli oznaka klase. Klasifikacijom se rješavaju zadatci poput prepoznavanja slova, brojki ili prometnih znakova. Semantička segmentacija ima mnoge primjene u raznim područjima, a vrlo je popularna za područje prometa.

Za semantičku segmentaciju koristimo kompleksne duboke modele koji zahtijevaju velike računalne resurse. Modeli se međusobno razlikuju po uspješnosti. Performanse su vrlo bitne kod dubokih modela. Neki od modela poput ResNeta ili DenseNeta postižu vrlo visoke točnosti, ali nisu baš vremenski učinkoviti. Poseban slučaj su sustavi koji funkcioniraju u stvarnom vremenu(real-time). Oni su posebno zanimljivi jer omogućavaju prenosivost i ugrađivanje u svakodnevno korištene uređaje. Krajnji uređaji(engl. edge devices) su ograničeni po pitanju računalne snage i memorije. Cilj je smanjiti broj skupih operacija s decimalnim brojevima i operacija koje pristupaju memoriji.

Postoji više pristupa ovom problemu. Jedan od popularnih pristupa za poboljšanje performansi modela jest kvantizacija [11] [21], ona se fokusira na smanjivanje operacija s decimalnim brojevima i na taj način poboljšava efikasnost modela. Bolja učinkovitost modela može se dobiti i pažljiivm odabirom arhitekture. Arhitektura koju istražujemo u ovom radu pomno je osmišljena i iznimno je efikasna kada je riječ i operacijama množenja i zbrajanja te memorijskom prometu.

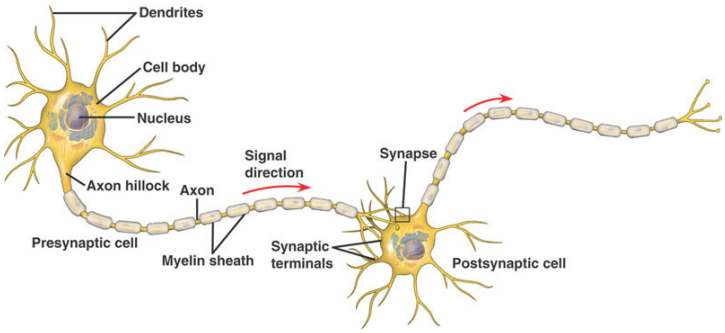
2. Duboko učenje

2.1. Umjetne neuronske mreže

Ljudski mozak sastoji se od desetke milijardi neurona. Neuroni međusobno komuniciraju i prenose signale putem sinapsi tvoreći kompleksne mreže. [12] Umjetne neuronske mreže su nastale po uzoru na ljudski centralni živčani sustav. Nastoji se imitirati rad neurona i sinapsi pomoću raznih matematičkih i aktivacijskih funkcija te na taj način prenijeti informacije odnosno, učiti.

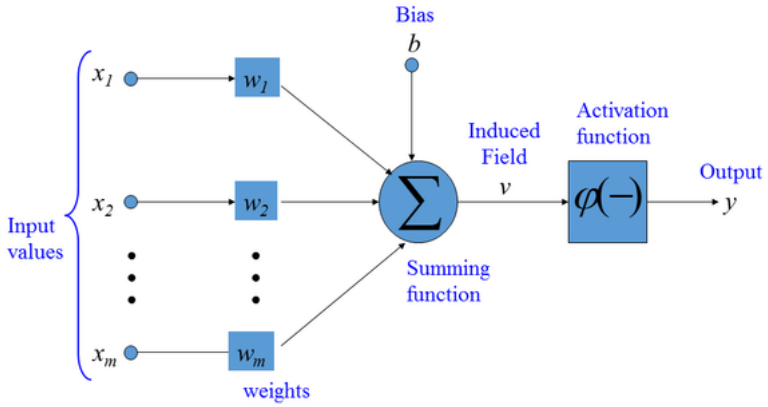
2.1.1. Umjetni neuron

Biološki neuron se sastoji od receptora(dendrita) koji ulaze u njega i efektora(aksona). Receptori skupljaju informacije od mnoštvo drugih neurona i prosljeđuje te informacije drugim neuronima pomoću efektora koji završavaju sinapsama.



Slika 2.1: Prikaz izgleda biološkog neurona.
preuzeto sa: https://www.kurzweilai.net/images/neuron_structure.jpg

Umjetni neuron je osnovna jedinica u neuronskoj mreži i on je pojednostavljeni biološki neuron. Sastoji se od ulaza u neuron, težina koje predstavljaju sinapse i izlaza. (slika 2.2)



Slika 2.2: Prikaz izgleda umjetnog neurona.
preuzeto sa: https://www.gabormelli.com/RKB/File:artificial-neuron-model.png

Ulazni podatci $x_1..x_m$ su izlazi prethodnih neurona koji ulaze u trenutni. Svaki od tih podataka se množi sa svojom težinom w_i gdje je $i = 1..m$ i zbrajaju se zajedno s pragom neurona(bias) i na njih se primjenjuje prijenosna funkcija. Konačan izlaz iz neurona možemo izraziti formulom

$$y = \varphi(\sum_{i=1}^{m} x_i w_i + b) \tag{2.1}$$

Često je vrlo praktično pokupiti sve vrijednosti u matrice pa nakon toga formula izgleda ovako

$$y = \varphi(\mathbf{w}^T \mathbf{x} + \mathbf{b}) \tag{2.2}$$

2.1.2. prijenosne funkcije

Prijenosne funkcije su funkcije kojima se kontrolira izlaz iz neurona. Dobre prijenosne funkcije trebaju biti monotono rastuće i nelinearne. Kada se ne bi koristila ili ako bi se koristila linearna prijenosna funkcija, imali bi linearnu jednadžbu koja je jednostavna za riješiti, ali je ograničenih mogućnosti i ne može prepoznati složeno grupiranje podataka. [19] Također bi se svi neuroni posljednjeg sloja mogli prikazati kao linearna kombinacija svih prethodnih neurona. Upravo zbog toga dobra funkcija mora biti nelinearna da može modelirati mreže s puno skrivenih slojeva koje mogu bolje učiti i prepoznati kompleksne uzorke u podatcima. Neke od najpoznatijih su binarna step

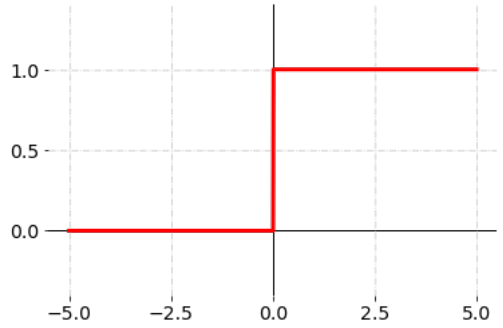
funkcija, sigmoida, zglobnica ili ReLU(Rectified Linear Unit), softmax i druge.

binarna step funkcija

Formula step funkcije je

$$f(x) = \begin{cases} 0 & \text{if x<0} \\ 1 & \text{if x>=0} \end{cases}$$

Step funkcija zapravo aktivira trenutni neuron, ako se je on veći ili jednak graničnoj vrijednosti. U slučaju da nije aktiviran, izlaz iz trenutnog neurona se ne šalje dalje. Nije derivabilna na cijeloj svojoj domeni i stoga nije najbolji odabir, ako koristimo metode koje koriste gradijente tijekom učenja.



Slika 2.3: Prikaz izgleda binarne step funkcije.
preuzeto sa: https://towardsdatascience.com/getting-to-know-activation-functions-in-neural-networks-125405b67428

sigmoida

Formula sigmoide glasi

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

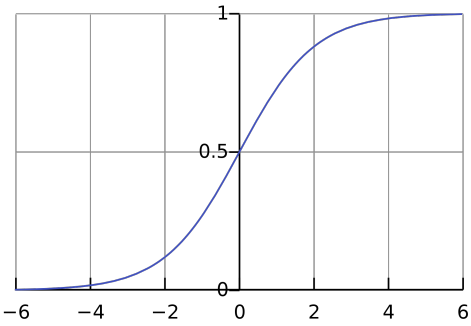
Sigmoidalna funkcija je jedna od najpopularnijih aktivacijskih funkcija i često se upotrebljava. Diferencijabilna je pa je pogodna za modele koji uče na temelju gradijenata. Njezina derivacija glasi

$$\sigma'(x) = \frac{e^{-x}}{(1 + e^{-x})^2}$$

odnosno

$$\sigma'(x) = \sigma(x)(1 - \sigma(x))$$

Vrlo pogodna za rješavanje problema binarne klasifikacije. Sigmoida spljošćuje vrijednosti na interval [0,1] te je ona zapravo poopćenje binarne step funkcije. Postoji mogućnost od zasićenja u slučaju kad sigmoida poprima rubne vrijednosti 0 ili 1 čime se poništava djelovanje gradijenta jer je njegova vrijednost približno 0.



Slika 2.4: Prikaz izgleda Sigmoide.
preuzeto sa: https://upload.wikimedia.org/wikipedia/commons/thumb/8/88/Logistic-curve.svg/1920px-Logistic-curve.svg.png

softmax

Formula za softmax glasi

$$\sigma(s_j) = \frac{e^{s_j}}{\sum_k e^{s_k}}, k = 1..C, \textit{gdje je C broj klasa}$$

Softmax je zapravo poopćenje sigmoide i koristi se kod klasifikacije kada ima više od dvije klase. Primjenom softmax-a dobivamo vjerojatnost pripadnosti nekoj klasi, a klasificira se u onu klasu s najvećom vjerojatnosti. Kao i sigmoida, softmax također ima problem sa zasićenjem.

Zglobnica

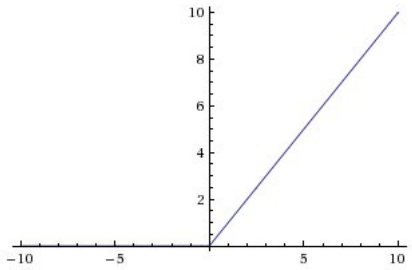
Vrlo popularna prijenosna funkcija i sve češće korištena. Funkcionira tako što pozitivne vrijednosti propušta, a negativne ne propušta odnosno postavlja ih na 0. Njena funkcija glasi

$$f(x) = max(0, x)$$

a derivacija

$$f(x) = \begin{cases} 0 & \text{if x<=0} \\ 1 & \text{if x>0} \end{cases}$$

Prednost je u tome što je vrlo laka za računanje i implementiranje. Može doći do nestanka pomaka gradijentom(engl. vanishing gradient), ali to je rijetkost.



Slika 2.5: Prikaz izgleda Zglobnice (ReLU).
preuzeto sa:https://www.kaggle.com/code/dansbecker/rectified-linear-units-relu-in-deep-learning/notebook

2.2. Učenje neuronske mreže

Parametri modela su određeni arhitekturom neuronske mreže. Početne vrijednosti parametra modela se odrede slučajno, a onda se treniranjem prilagođavaju podatcima. Kada pričamo o parametrima modela, mislimo na težine i pragove svakog neurona koji se nalazi u mreži. Učenje se odvija tako da model dobije podatak **x** i oznaku **y** pa pomoću njih mijenja svoje parametre i prilagođava se podatcima. To spada pod vrstu nadziranog učenja(engl. supervised learning), suprotno tome bilo bi nenadzirano učenje(engl. unsupervised learning) gdje model dobije podatak, ali ne dobije izlaz nego traži pravilnosti i po njima grupira podatke. Podatci koji se daju kao ulaz mogu dolaziti samostalno, u malim grupama(engl. mini batches) ili u grupama. Prolazom podatka unaprijed kroz mrežu primjenjuju se sve težine i pragovi u tom trenutku i posljednji sloj nam daje izlaz. Izlaz se uspoređuje sa stvarnim podatcima i izračunava se pogreška. Zatim se po mreži prolazi unazad i uz pomoć funkcije gubitka prilagođavaju parametri modela [6] da greška bude što manja odnosno da model bude što točniji.

2.2.1. Funkcije gubitka

Kako bismo mogli vrednovati i učiti naš model moramo uvesti funkciju koja nam govori koliko je model dobar. Umjesto da tražimo maksimum koliko je model dobar, funkcije gubitka su se pokazale vrlo praktične jer se moraju minimizirati pa nam to uvelike olakšava rad i računanje s njima. Postoji nekolicina takvih funkcija, svaka sa svojim prednostima i nedostacima, no spomenuti ćemo 2 najčešće korištene.

Srednja kvadratna pogreška
Funkcija gubitka koja gleda kvadriranu razliku između stvarnih vrijednosti i predikcija

modela te uzima srednju vrijednost. Formula za srednju kvadratnu pogrešku glasi

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

gdje su n - broj podataka, $\hat{y}_i$ - predikcije modela, y_i - stvarne vrijednosti

Unakrsna entropija

Pokazala se vrlo praktičnom i puno češće se koristi u praksi za rješavanje problema poput klasifikacije i segmentacije. Formula glasi

$$L = -\sum_{i=1}^{C} t_i log(P(Y = Y_i|x))$$

gdje je t_i-vrijednost 0 ili 1 i označava pripadnost klasi, a $P(Y = Y_i|x)$-vjerojatnost pripadanja klasi

2.2.2. Optimizacijski algoritmi

Optimizacijski algoritmi služe za pronalazak globalnog minimuma na funkciji gubitka. Nakon svake iteracije dolazi do računanja gubitka, nakon čega slijedi propagiranje greške unatrag(engl. backpropagation) po mreži. Propagiranje greške unatrag se radi pomoću izračuna parcijalnih derivacija funkcije gubitka po parametrima odnosno gradijenata. Nakon što su gradijenti izračunati, slijedi pomicanje u smjeru negativnog gradijenta jer se traži minimum. Optimizacijski algoritmi koji koriste gradijent često koriste i zalet. Time se poboljšava traženje globalnog minimuma kod nekonveksnih funkcija jer se može prijeći preko lokalnih minimuma.

$$v_t = \gamma v_{t-1} + \epsilon g_t \tag{2.3}$$

$$\theta = \theta - v_t \tag{2.4}$$

gdje je γ - hiperparametar koji određuje koliki je zalet, v_{t-1} - pomak prošlog koraka, g_t - gradijent funkcije gubitka po parametrima θ

Gradijentni spust

Gradijentni spust nastoji optimirati funkciju gubitka tako da ju pomiče u smjeru negativnog gradijenta. Neka je naša funkcija $\mathcal{L}$ gubitka skalarna, odnosno neka vrijedi preslikavanje

$$\mathcal{L} : R^n \to R \tag{2.5}$$

Tada je gradijent funkcije $\mathcal{L}$ stupčani vektor koji je zapravo transponirana Jacobijeva matrica

$$\nabla\mathcal{L}(\mathbf{x}) = \frac{d\mathcal{L}(\mathbf{x})}{d\mathbf{x}}^{\top} \tag{2.6}$$

Koristeći Taylorov razvoj prvog reda dobivamo aproksimaciju

$$\mathcal{L}(\mathbf{x} + \Delta\mathbf{x}) \approx \mathcal{L}(\mathbf{x}) + \frac{d\mathcal{L}(\mathbf{x})}{d\mathbf{x}}\Delta\mathbf{x} = \mathcal{L}(\mathbf{x}) + \nabla\mathcal{L}(\mathbf{x})^{\top}\Delta\mathbf{x} \tag{2.7}$$

Uvrstimo da je

$$\Delta\mathbf{x} = -\eta\mathbf{g},\ \mathbf{g} = \nabla\mathcal{L}(\mathbf{x}) \tag{2.8}$$

Uvrštavanjem izraza natrag u 2.7 dobivamo sljedeće

$$\mathcal{L}(\mathbf{x} - \eta\mathbf{g}) \approx \mathcal{L}(\mathbf{x}) - \eta\|\mathbf{g}\|^2 < \mathcal{L}(\mathbf{x})\ . \tag{2.9}$$

Vrijedi $\|\mathbf{g}\|^2 > 0$ pa će se sigurno pronaći lokalni minimum funkcije nekonveksne ili globalni minimum konveksne funkcije. η je hiperparametar koji se naziva stopa učenja. U pravilu je točnije određivanje minimuma što je η manji, ali ne smije biti ni premalen jer bi onda učenje predugo trajalo. Osnovna verzija ovog algoritma koristi cijeli skup podataka za optimiranje gradijenata i stoga nije baš praktična za velike skupove podataka.

Stohastički gradijentni spust

U odnosu na osnovnu verziju algoritma, stohastički gradijentni spust ne radi ažuriranje pomoću cijelog skupa podataka nego se gradijenti ažuriraju za svaki ulazni podatak i oznaku. [18] Ovim se uvelike ubrzava rad jer za izračun gradijenta nije potrebno koristiti cijeli skup podataka. Za ovaj pristup najčešće se koriste hiperparamteri $\eta = 10^{-2}$ i $\gamma = 0.9$.

Prilagodljiva procjena zaleta (Adam)

Algoritam optimizacije koji koristi različite stope učenja, kao i stohastički gradijenti spust. Adam sprema podatke poput eksponencijalno padajućeg prosjeka kvadrata prošlih gradijenata(v_t) i eksponencijalno padajućeg prosjeka prošlih gradijenata(m_t) [18]

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1)g_t \tag{2.10}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2)g_t^2 \tag{2.11}$$

S obzirom na to da se početni m_t i v_t postavljaju na 0, postoji pristranost, stoga se najprije mora ukloniti pristranost. [8] [18]

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{2.12}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{2.13}$$

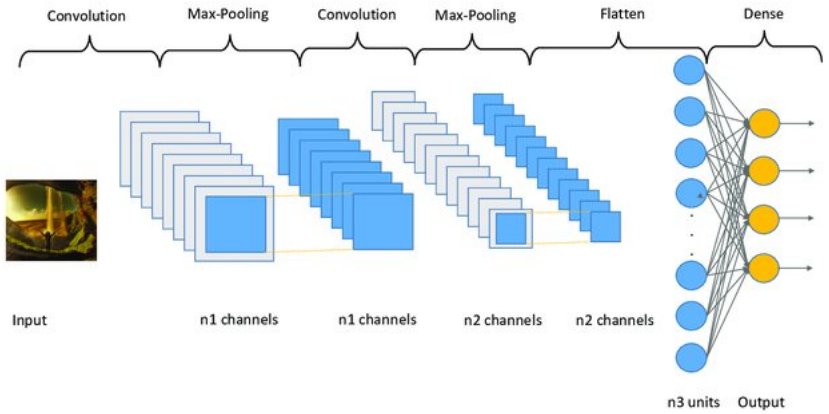
Konačno, novi parametri se računaju

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon}\hat{m}_t \tag{2.14}$$

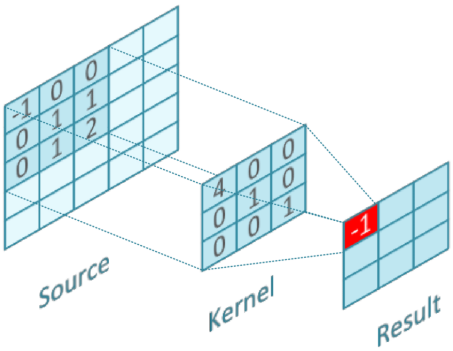
Adam se najčešće koristi s parametrima $\eta = 10^{-3}$, $\beta_1 = 0.9$ i $\beta_2 = 0.999$.

2.3. Konvolucijske neuronske mreže

Ideja konvolucijski mreža bila je smanjiti broj parametara modela koji obrađuju podatke poput slika. [1] [14] Pridruživanje neurona svakom pikselu bi vrlo brzo učinilo modele memorijski zahtjevnima za treniranje. [14] Zbog toga se jedan neuron dodjeljuje jednom lokalnom području i ima svoje receptivno polje. Uzmimo sliku dimenzija 64x64x3(RGB slika u boji) i receptivno polje 6x6. Obična umjetna neuronska mreža bi za svaki neuron imala 12288 težina, a konvolucijska samo 108(6x6x3 veličina filtra). Konvolucijske mreže su otporne na lokalizaciju objekata, što znači da će jednako otkriti objekt bilo gdje na slici, dok to nije slučaj kod običnih neuronskih mreža. Za konvolucijske neuronske mreže su posebno važni konvolucijski sloj i sloj sažimanja.



Slika 2.6: Prikaz jednostavne konvolucijske mreže.
preuzeto sa: https://www.researchgate.net/profile/Jose-Benitez-Andrades/publication/339447623/figure/fig2/AS:862056077082627@1582541593714/A-vanilla-Convolutional-Neural-Network-CNN-representation_W640.jpg



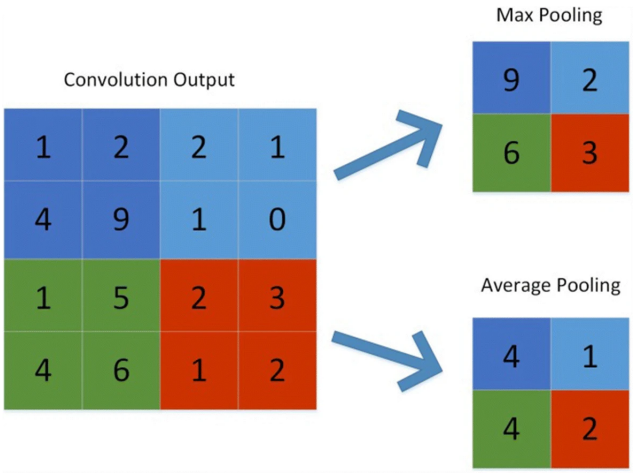
Slika 2.7: Primjena filtra na podatke i stvaranje mape značajki.
preuzeto sa:https://medium.com/s-a-a-s/dl-basic-concept-of-cnn-2ef4fc9b039b

2.3.1. Konvolucijski sloj

Sastoji se od filtera(engl. filter) ili jezgre(engl. kernel), koraka(engl. stride) i nadopune(engl. padding). Konvolucijskim slojem se zapravo pomoću filtra izvlače informacije i stvaraju se 2D mape značajki. (slika 2.7) Složenost sloja može se odrediti namještanjem parametara poput veličine jezgre, koraka i nadopune. Korakom određujemo za koliko se pomiče filter, a nadopunom pokušavamo ukloniti diskriminaciju rubnih elemenata. Broj izlaznih kanala iz konvolucije određen je brojem filtera koji se primjenjuju nad ulazom. (slika 2.6)

2.3.2. Sloj sažimanja

Slojevi sažimanja se nalaze između konvolucijskih i njihova uloga je još dodatno smanjiti kompleksnost i dimenzionalnost modela, a time i broj parametara i složenost računanja u idućim slojevima. [1] Sažimanje se koristi za izvlačenje najbitnijih značajki na nekom području. U sloju sažimanja također postoji filtar no razlika u odnosu na konvolucijski sloj je što nema učenja, nego je filter uvijek isti. Najčešći načini sažimanja su srednjom vrijednošću ili maksimalnom vrijednošću. Sažimanje maksimalnom vrijednošću(engl. max pooling) odabire najveću vrijednost značajke na području gdje se nalazi filter, a sažimanje srednjom vrijednošću(engl. average pooling) uzima prosjek na području gdje se nalazi filter. Obično se uzima filter dimenzija 2x2 i pomak 2 tako da nema presjeka. U ovom radu korišteno je sažimanje srednjom vrijednošću.



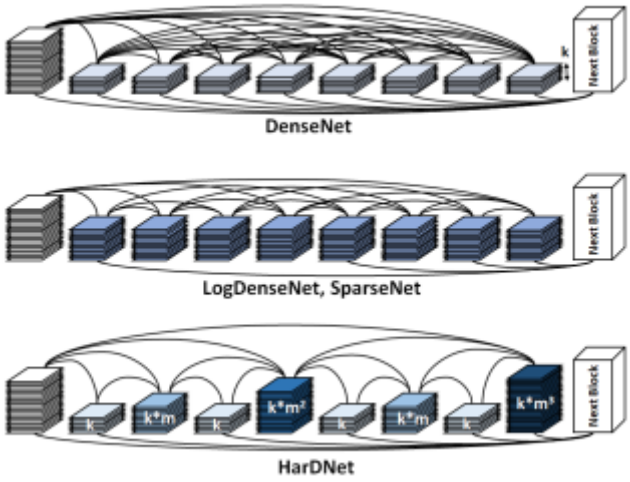
Slika 2.8: Primjera sažimanja maksimalnom i srednjom vrijednošću.
preuzeto sa: https://link.springer.com/article/10.1007/s11042-019-08345-y/figures/4

2.3.3. Arhitektura mreže HarDNet-70

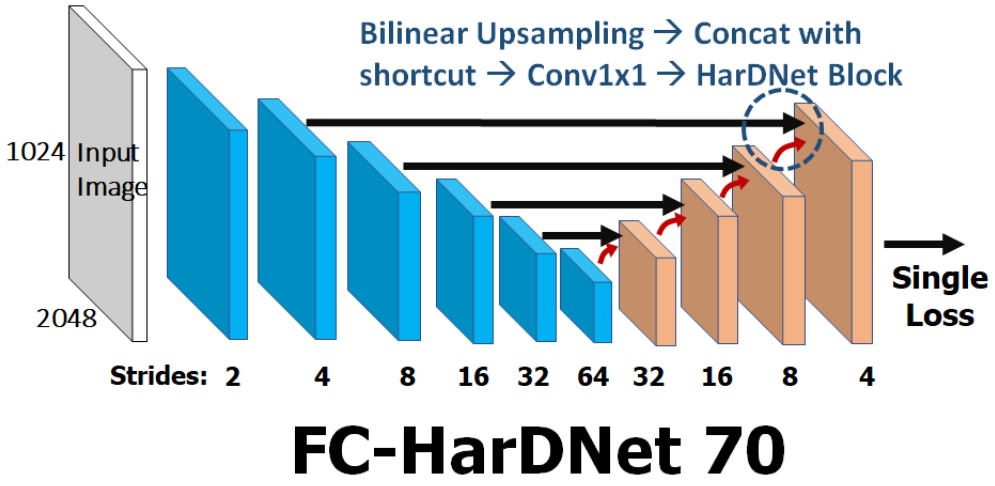
Harmonička gusto povezana mreža

Harmonička gusto povezana mreža oblikovana je po uzoru na gusto povezane mreže [7]. U gusto povezanim mrežama izlazi svih prethodnih slojeva se nadodaju na ulaz u trenutni sloj mreže. Kada se neki slijed slojeva često ponavlja, praktično je skupiti te slojeve u blok (slika 2.9). Model u radu se sastoji od harmonički gusto povezanih blokova međusobno povezanih konvolucijom 1x1. Harmonička gusto povezana mreža pažljivim odabirom arhitekture postiže poboljšanje računske gustoće [5]. Svaki sloj k se povezuje sa slojem $k - 2^n$, ako je k djeljiv sa 2^n i vrijedi $k - 2^n \geq 0$ gdje je n nenegativni cijeli broj, a sloj 0 je ulazni sloj [4]. Ovakvim povezivanjem prilikom unaprijednog prolaza nakon obrade sloja 2^n slojevi od 1 do $2^n - 1$ se mogu ukloniti iz memorije. Ovo je vrlo praktično za krajnje uređaje koji imaju malu memoriju. Učenje modela i dalje ostaje zahtjevno. FC-HarDNet 70 je potpuno konvolucijska neuronska mreža koja se koristi za semantičku segmentaciju, a sastoji se samo od konvolucijskih i slojeva sažimanja. Naime, ovo nije prva takva arhitektura, nekoliko ih je navedeno u radu [20]. Model koristi Enkoder-Dekoder strukturu oblika U. Prethodni radovi sličnih struktura su U-net [17] koji se koristio za obradu medicinskih slika i Ladder-Style DenseNet [9]. Takav oblik znači da se prvo intenzivno radi poduzorkovanje pa se onda povećava dimenzionalnost naduzorkovanjem. Naduzorkovanje u ovom modelu se vrši bilinearnom interpolacijom. Unutar blokova nalazi se kombinacija konkatenacije, konvolucije 1x1 i konvolucije 3x3, a između blokova konvolucija 1x1. Arhitektura konvolucijske mreže u ovom radu izgleda kao na slici 2.10 uz naglasak da su ulazne

dimenzije slike drukčije.



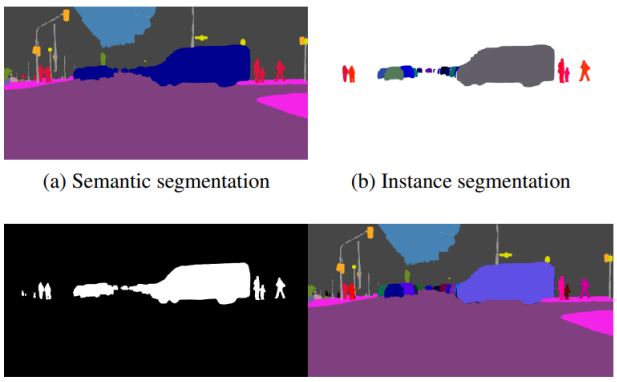
Slika 2.9: Usporedba gusto i harmonički gusto povezane mreže [4].



Slika 2.10: Prikaz FC-HarDNet 70 arhitekture [3].

3. Semantička segmentacija

Jedan od osnovih izazova kojim se bavi područje računalnog vida jest upravo segmentacija. Glavna ideja je pridijeliti oznaku svakom pikselu te ju svrstati u određenu klasu. S obzirom na to da se predikcija radi za svaki piksel na slici, često se naziva gustom predikcijom. Pod probleme segmentacije spadaju semantička segmentacija, segmentacija primjeraka i panoptička segmentacija. Semantičkom segmentacijom zapravo dodjeljujemo klasu svakom pikselu na slici. Segmentacijom instanci određujemo različite objekte na slici. Semantička segmentacija svim objektima iste klase dodjeljuje istu oznaku, a segmentacija primjeraka unutar neke klase raspoznaje zasebne objekte i pridjeljuje im različitu oznaku. Segmentacija primjeraka određuje 2 vrste objekata. To su prebrojivi(engl. things) i neprebrojivi(engl. stuff). Prebrojivi su ljudi, auti i slično. Neprebrojivi su nebo, cesta i ostali. Panoptička segmentacija je zapravo kombinacija prethodne dvije vrste.



(a) Semantic segmentation (b) Instance segmentation

(c) Things and Stuff Consistency (d) Panoptic segmentation

Slika 3.1: Razdvajanje prebrojivih i neprebrojivih objekata te prikaz različitih vrsta segmentacije [10].

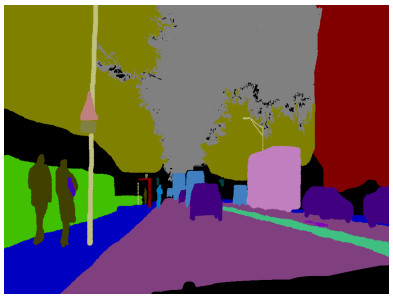
U ovome radu razmatramo semantičku segmentaciju, koja ima važne primjene u mnogim područjima poput medicine, prometa i općenito obradi i analizi slika. U području medicine se koristi za obradu medicinskih slika iz raznih uređaja, a jedna od primjena je za segmentaciju tumora mozga na slikama magnetske rezonance [15]. Od primjena

u prometu, vrlo bitna je analiza scena iz prometa, koja se obrađuje u ovom radu i bitna je za područje autonomne vožnje. Za rješavanje ovakvih problema tipično se koriste potpuno konvolucijske mreže (eng. fully convolutional networks) već ranije spomenute, koje nemaju niti jedan potpuno povezani sloj nego samo konvolucijske slojeve i slojeve sažimanja. Intenzivnim poduzorkovanjem slike se degradiraju. Kako ne bismo izgubili bitne značajke uvode se preskočne veze i time se omogućuje bolja predikcija. Zatim slijedi naduzorkovanje koje kombinira izlaz iz prethodnog sloja i slojeve preskočnih veza (slika 2.10). Naduzorkovanjem se povećavaju dimenzije slike na originalne i radi se predikcija. Kada bismo radili samo naduzorkovanje bez preskočnih veza, dobivali bi lošije predikcije jer bi se neke značajke izgubile u degradiranju. Za naduzorkovanje se koristi bilinearna interpolacija.

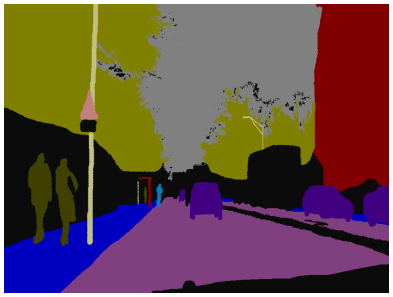
4. CamVid skup podataka

U sklopu ovog eksperimenta korišten je skup podataka CamVid. CamVid skup podatak je nastao iz preko 10 minuta dugog videomaterijala. [2] Sastoji se od 701 slike za koje su ručno označene klase te provjerene od strane još jedne osobe. Slike su podijeljene na tri skupa. Skup za učenje sadrži 367 parova slika i oznake za tu sliku, skup za validaciju 101, a skup za testiranje 233 para. Dimenzije slika su 960x720 (širina x visina). Originalno skup podataka koristi 32 razreda, ali puno češće se koristi reducirani model od 11 razreda [13]. Klase koje se koriste u reduciranom modelu su: nebo, zgrada, stup, cesta, pločnik ili nogostup, drvo, prometni znak/simbol, ograda, auto, pješak, biciklist, a sve ostalo se vodi pod neoznačeno. Treniranje se provodi na kombinaciji skupa za treniranje i skupa za validaciju, a evaluacija na skupu za testiranje [13], to je česta praksa kad je skup podataka malen.

Slika 4.1: Originalna slika koju treba segmentirati.



Slika 4.2: Originalne oznake s 32 razreda.



Slika 4.3: Reducirane oznake s 11 razreda.

5. Implementacija

Za implementaciju ovog eksperimenta korišten je programski jezik Python verzije 3. Eksperiment se sastojao od implementacije arhitekture FC-Hardnet 70 [3] i svih dodatnih stavki bitnih za funkcioniranje mreže poput pripreme i učitavanja podataka te treniranje modela na skupu podataka CamVid. Za implementaciju i izvođenje programa korištena je bilježnica na Kaggle platformi koja omogućuje korištenje GPU-a. Platforma Kaggle nudi mogućnost besplatnog korištenja NVIDIA TESLA P100 GPU-a što je vrlo korisno za učenje dubokih modela.

5.1. biblioteka NumPy

Vrlo korisna biblioteka koja olakšava i ubrzava rad s matricama. To je vrlo praktično jer je često brže skupiti derivacije u matricu pa pomnožiti dvije matrice umjesto korištenja iteracija. U ovom radu biblioteka NumPy je korištena u svrhu pripreme i promjena nad ulaznim podatcima odnosno slikama koje su zapravo matrice dimenzija W x H x 3 gdje su W - širina slike i H - visina slike, a 3 su udjeli RGB odnosno udio crvene, zelene i plave boje. Osim za pripremu, također je korištena za izračun rezultata i stvaranje konfuzijske matrice.

5.2. biblioteka PyTorch

Biblioteka PyTorch je optimizirana biblioteka za rad s tenzorima te namijenjena za područje dubokog učenja. U njoj se vrlo lako slažu neuronske mreže. Osim slaganja neuronskih mreža, nudi mnoštvo opcija za obradu slika, implementirane najčešće funkcije gubitka, optimizatori te raspoređivači(engl. schedulers) koji služe za podešavanje hiperparametra stope učenja(η). U radu su korišteni gubitak `CrossEntropyLoss` s parametrom `ignore_index=11` jer se neoznačenim dijelovima na slici pridjeljuje oznaka 11, optimizator `Adam` s pretpostavljenim parametrima [18] i raspoređivač `CosineAnnealingLR` s parametrima `T_max=400` i `eta_min=1e-6`. Funkcija

gubitka `CrossEntropyLoss` nalazi se u modulu `torch.nn`, a preostala dva u modulu `torch.optim`.

5.2.1. Učitavanje podataka

Implenetiran je već postojeći razred za učitavanje podataka `DataLoader` koji se može koristiti za svoje privatne ili već gotove skupove podataka, a nalazi u modulu `torch.utils.data`. On kao argumente zahtjeva `DataSet` koji se također nalazi u `torch.utils.data` te opcionalne parametre. Od opcionalnih parametara korišteni su veličina grupe(engl. batch size) i slučajno miješanje(engl. shuffle). `DataSet` je razred koji mora imati implementirane metode `__len__()` koja vraća broj elemenata skupa podataka i metodu `__getitem__()` koja kao argument prima indeks `i`, a vraća `i`-ti uzorak podatka odnosno ulaznu sliku i označenu sliku. Prilikom dohvaćanja uzoraka podataka, često se rade i transformacije slika radi boljeg učenja. Programski je ostvaren vlastiti razred Compose po uzoru na razred `Compose` iz modula `torchvision.transforms` te razredi `RandomHorizontallyFlip`, `RandomScaleCrop` i `RandomCrop` po uzoru na razrede iz istog modula. Jedina razlika između tih razreda je što novoimplementirani algoritmi se pozivaju na obje slike odjednom odnosno na ulaznoj slici i označenoj.

5.2.2. Izgradnja arhitekture

U modulu `torch.nn` postoje već implementirani bitni dijeli za izgradnju neuronske mreže. Model mreže je zapravo spremnik koji sadrži podspremnike odnosno blokove i slojeve. Za implementaciju slojeva napravljen je novi razred `ConvLayer` koji se sastoji od elemenata iz `torch.nn` modula redom `Conv2d`, `BatchNorm2d` i `ReLU`. `Conv2d` je razred pomoću kojega se provodi konvolucija, neki od važnijih parametara koji se mogu namjestiti su veličina ulaznih i izlaznih kanala, jezgre(engl. kernel_size), pomaka(engl. stride) i nadopune(engl. padding). `ReLU` je aktivacijska ili prijenosna funkcija koja je već prethodno spomenuta. (poglavlje 2.1.2.) `BatchNorm2d` se koristi za normalizaciju vrijednosti grupe tako da se oduzima srednja vrijednost i dijeli sa standardnom devijacijom. `AvgPool2d` je razred koji imitra sloj sažimanja, konkretno u implementaciji koristi se sažimanje srednjom vrijednošću.

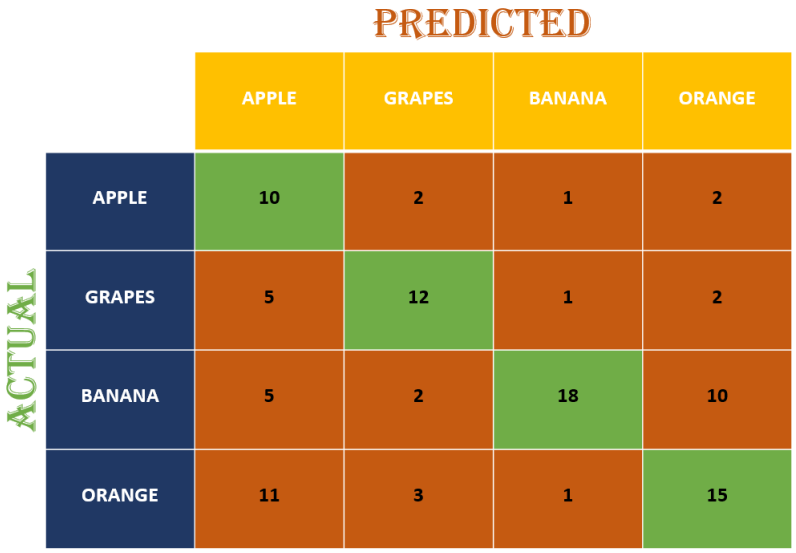
5.2.3. Operacije nad tenzorima

PyTorch-ev Tenzor je struktura podataka koja je vrlo slična NumPy-jevom polju. Tenzori su nepromjenjiva (engl. immutable) struktura podataka u odnosu na NumPy-jeve nizove. Dok se nizovi mogu obrađivati samo uz pomoć CPU-a, tenzore PyTorcha možemo obrađivati na CPU, GPU i TPU. Jedna od najvažnijih razlika je što nam PyTorch omogućuje automatsku diferencijaciju i pohranjivanje vrijednosti gradijenata. Navedena svojstva PyTorch-evih tenzora omogućavaju praktično i učinkovito računanje gradijenata dubokih modela. Zbog toga PyTorch predstavlja jedan od najpopularnijih izbora za izražavanje algoritama za strojno učenje velikih modela.

6. Eksperimentalni rezultati

6.1. Korišteni alati mjerenja

Nakon učenja nastoji se provjeriti točnost modela. Za ocjenu se gotovo nikad ne koristi samo jedan način za mjerenje jer se uvijek može zavarati i dati rezultate kao da model radi uspješno, iako to nije slučaj. U ovome radu koristimo sljedeće metrike za kvantitativno izražavanje generalizacijske moći: točnost (engl. accuracy), srednja točnost (engl. mean accuracy) i srednja vrijednost omjera presjeka i unije (engl. mean intersect over union). Pri izračunu ovih mjera vrlo je praktično koristiti matricu konfuzije (slika 6.1). Matrica konfuzije (**M**) je kvadratna matrica dimenzija (NxN) gdje je N broj klasa koje postoje. Svaki redak u matrici sadrži podatke o tome koliko je primjera točno klasificirano, te koliko je primjera klasificirano u svaku od ostalih klasa. Na slici 6.1 može se vidjeti u prvom retku da je 10 jabuka točno klasificirano, 2 jabuke su klasificirane kao grožđe, 1 kao banana te 2 kao naranča. Osim mjerenja točnosti, mjeri se i brzina modela u okvirima po sekundi (engl. FPS, frames per second).



Slika 6.1: Prikaz izgleda matrice konfuzije na jednostavnom primjeru.
preuzeto sa:https://miro.medium.com/max/1400/1*Q7FRqO3uk-nuw8L2E1BLiQ.png

Točnost je mjera kojom se određuje uspješnost tako da se uzme postotak točno klasificiranih piksela i podijeli s ukupnim brojem piksela na slici. Primijetimo kako nam se točno klasificirani pikseli nalaze na dijagonali matrice **M**. Matematički bismo to mogli izraziti ovako

$$Accuracy = \frac{\sum_{i}^{N}(\mathbf{M})_{ii}}{\sum_{i,j}(\mathbf{M})_{ij}} \tag{6.1}$$

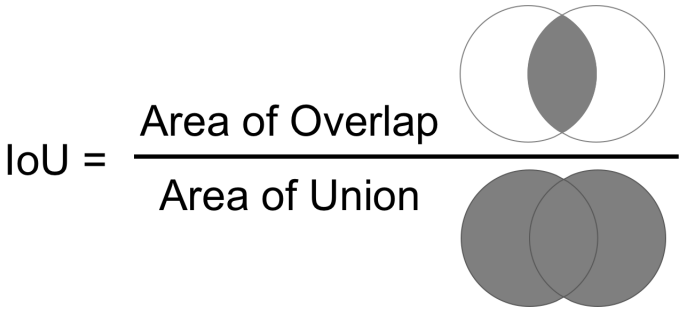
Kao što smo i ranije naveli, N predstavlja broj klasa.
Uz točnost vrlo često se koristi i srednja točnost jer jednako zastupa sve klase. Recimo da imamo klasu A, klasu B i klasu C. Neka se u podatcima nalazi jako puno primjeraka klase A, a malo primjeraka za klasu B i C. Vidimo da, ako uvijek stavimo da se predviđa A točnost bi i dalje bila velika. Međutim, srednja točnost bi pokazivala realnije rezultate. Srednju točnost računamo prema sljedećoj formuli

$$Mean\ Accuracy = \frac{1}{N}\sum_{i=1}^{N}\frac{(\mathbf{M})_{ii}}{\sum_{j=1}^{N}(\mathbf{M})_{ij}} \tag{6.2}$$

Omjer presjeka i unije (engl. IoU intersection over union) jedna je od najpopularnijih metrika za evaluaciju u semantičkoj segmentaciji i detekciji objekata, poznata i kao Jaccardov indeks [16]. Matematički izraz za mjerenje omjera presjeka i unije možemo izraziti pomoću formule

$$IoU = \frac{A \cap B}{A \cup B} \tag{6.3}$$

Vizualna reprezentacija omjera presjeka i unije prikazana je na slici 6.2.



Slika 6.2: U brojniku lijevi krug je stvarni prikaz objekta. Desni krug je prikaz objekta nakon izlaza iz modela. Omjer presjeka i unije je omjer sivog dijela u brojniku i sivog dijela u nazivniku.

Omjer presjeka i unije također možemo računati iz matrice konfuzije na sljedeći način

$$IoU(c) = \frac{(\mathbf{M})_{cc}}{\sum_{i=1}^{N}[(\mathbf{M})_{ic} + (\mathbf{M})_{ci}] - (\mathbf{M})_{cc}} \tag{6.4}$$

Broj c predstavlja klasu od 1 do N.
Kao i kod točnosti, prevelika zastupljenost jedne klase može uzrokovati probleme pri interpretaciji rezultata i evaluaciji modela. Stoga se puno češće koristi srednja vrijednost omjera presjeka i unije (engl. mIoU mean intersection over union). Tako su sve klase jednako zastupljene i dobiva se bolja ocjena o uspješnosti modela. Vrijednost mIoU računa se kao srednja vrijednost svih izmjerenih omjera presjeka i unije (jednadžba 6.4).

$$mean\ IoU = \frac{1}{N}\sum_{i=1}^{N} IoU(i) \tag{6.5}$$

6.2. Rezultati korištenog modela

Eksperimenti su provedeni na platformi Kaggle[1] koja nudi mogućnost korištenja GPU jedinice vrste NVIDIA TESLA P100. Eksperimentom je obuhvaćeno treniranje modela FC-HardNet 70 na CamVid skupu podataka. Učenje se odvijalo na skupu od 468 slika, odnosno na kombinaciji skupa za testiranje i validaciju, a evaluacija se odvijala na testnom skupu koji obuhvaća 233 slike. Prilikom treniranja korišteni su sljedeći elementi

```
1    loss_fn = torch.nn.CrossEntropyLoss(ignore_index=11)
2    optimizer = torch.optim.Adam(model.parameters(), lr=1e-3, betas
    =(0.9, 0.999), weight_decay=2.5e-5)
3    scheduler = torch.optim.lr_scheduler.CosineAnnealingLR(optimizer
    , T_max=400, eta_min=1e-6)
```

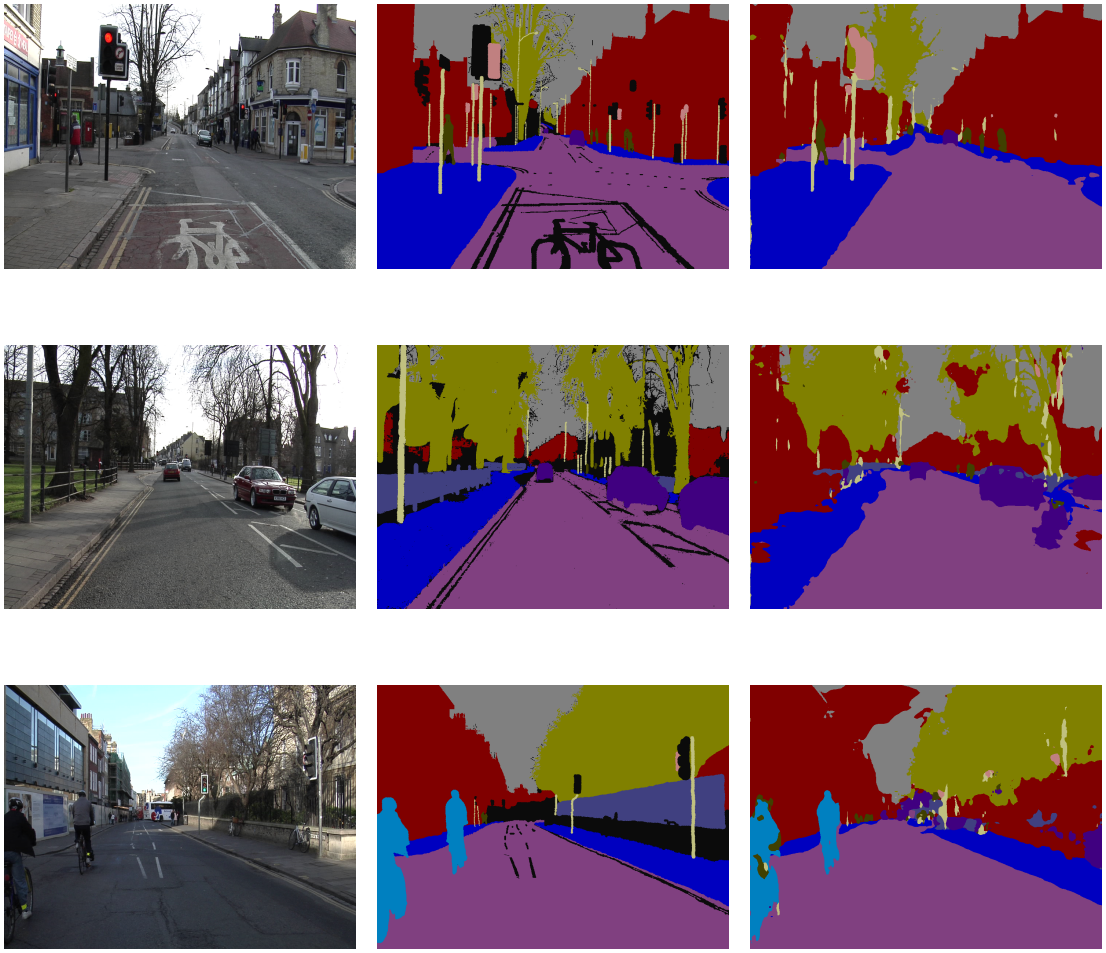
To su redom funkcija gubitka, optimizator i raspoređivač koje smo već spominjali u poglavlju 5.2, a koriste se u eksperimentu. S obzirom na to da se koristi mali skup podataka, morali smo ga dodatno uvećati (engl. augment). Uvećanje podataka provodimo perturbiranjem slika i odgovarajućih oznaka. U eksperimentu su korištene promjene horizontalnog zrcaljena (engl. horizontal flip) i slučajno skaliranje pa izrezivanje (engl. random scale and crop). Postoji još mnoštvo različitih transformacija, ali ove dvije su jedne od najkorisnijih pri učenju modela za gustu predikciju. Treniranje

[1] https://www.kaggle.com/

se odvijalo u 400 epoha i trajalo je ukupno 8 sati. Iz tablice 6.1 vidimo da model postiže točnost od 94% i mIoU oko 73%. Model teže prepoznaje objekte poput stupova i prometnih znakova, dok veće objekte poput ceste, neba i auta prepoznaje s visokom točnosti (tablica 6.2). Brzinu modela smo mjerili po formuli

$$FPS = \frac{N}{\sum t_i}$$

gdje je N broj slika u skupu za testiranje, a t_i vrijeme provedeno u modelu za i-tu sliku. Izmjerena brzina modela je 50.85 okvira po sekundi. Pažljivim odabirom arhitekture modela uspjeli smo smanjiti broj pristupa memoriji i time smo ubrzali model u odnosu na dotadašnje gusto povezane modele, mjerenja su dostupna u [4].



Slika 6.3: U lijevom stupcu su originalne slike, u sredini su označene slike, a u desnom stupcu su predikcije modela.

Tablica 6.1: Rezultati mjerenja modela FC-HarDNet 70 za semantičku segmentaciju na CamVid skupu podataka.

Metrika	Vrijednost[%]
Preciznost	94.28
Srednja preciznost	80.18
mIoU	72.81

Tablica 6.2: Izračunate vrijednosti omjera presjeka i unije po razredima za model FC-HarDNet 70 na CamVid skupu podataka.

Klasa	Vrijednost[%]
nebo	93.62
zgrada	89.89
stup	34.69
cesta	96.36
pločnik	86.07
drvo	80.30
prometni znak/simbol	48.46
ograda	64.39
auto	89.28
pješak	55.66
biciklist	62.18

7. Zaključak

Ovaj rad obradio je harmoničke gusto povezane modele za semantičku segmentaciju. Računalni vid jedan je od važnih zadataka umjetne inteligencije. Za taj problem najčešće se koriste duboki modeli. Semantička segmentacija se još nekad naziva i gustom predikcijom jer se određuje oznaka svakom pikselu.

Opisani su osnovni gradivni elementi dubokih modela poput potpuno povezanih slojeva i prijenosnih funkcija. Predstavljen je način na koji se ostvaruje učenje modela iz danih podataka optimirajući gubitak gradijentnim spustom. Opisani korišteni optimizator(Adam) i raspoređivač(Cosine Annealing) koji su pripomogli u traženju minimuma.

Dane su osnovne informacije o funkcioniranju konvolucijskih modela te njenim osnovnim elementima kao što su konvolucijski sloj i sloj sažimanja. Opisana je arhitektura s gusto povezanim konvolucijskim slojevima [7] kao i njena prorijeđena inačica HarDNet [4] koju smo koristili u eksperimentima. Testirana je arhitektura FC-HarDNet 70 koja je rađena po uzoru na gusto povezane modele.

Ukratko su dane informacije čime se točno bavi segmentacija u računalnom vidu. Detaljnije je opisana semantička segmentacija kojom se bavimo u ovom radu. Opisane su korištene tehnologije poput Kaggle online platforme koja nam omogućava korištenje GPU jedinica za učenje naših modela. Također su opisane i biblioteke koje su nam pomogle pri implementaciji. Biblioteka NumPy nam je pomogla za pripremu podataka i matrični račun, a PyTorch za izgradnju arhitekture i treniranje modela (efikasnost i automatska diferencijacija). Opisan je korišteni skup podataka (CamVid) na kojem smo obavljali treniranje i evaluaciju.

Prikazani su rezultati mjerenja uspješnosti i brzine modela. Daljnji napredak za povećanje točnosti bi mogao ići u smjeru da se model prethodno istrenira na Cityscapes skupu podataka. Također bi bilo zanimljivo ugraditi harmonički povezanu okosnicu u piramidalni SwiftNet (umjesto ResNet-18) [13].

LITERATURA

[1] Saad Albawi, Tareq Abed Mohammed, i Saad Al-Zawi. Understanding of a convolutional neural network. U *2017 International Conference on Engineering and Technology (ICET)*, stranice 1–6, 2017. doi: 10.1109/ICEngTechnol.2017.8308186.

[2] Gabriel J. Brostow, Julien Fauqueur, i Roberto Cipolla. Semantic object classes in video: A high-definition ground truth database. *Pattern Recognition Letters*, 30 (2):88–97, 2009. ISSN 0167-8655. doi: https://doi.org/10.1016/j.patrec.2008.04.005. URL https://www.sciencedirect.com/science/article/pii/S0167865508001220. Video-based Object and Event Analysis.

[3] Ping Chao. Fully convolutional hardnet for segmentation in pytorch. https://github.com/PingoLH/FCHarDNet, posjećeno 24.5.2022.

[4] Ping Chao, Chao-Yang Kao, Yu-Shan Ruan, Chien-Hsiang Huang, i Youn-Long Lin. Hardnet: A low memory traffic network. *ICCV*, 2019. URL http://arxiv.org/abs/1909.00948.

[5] Anja Delić. Semantička segmentacija harmoničkim gusto povezanim modelima, 2021. URL http://www.zemris.fer.hr/~ssegvic/project/pubs/delic21bs.pdf.

[6] Geoffrey E. Hinton. How neural networks learn from experience. *Scientific American*, 267(3):144–151, 1992. ISSN 00368733, 19467087. URL http://www.jstor.org/stable/24939221.

[7] Gao Huang, Zhuang Liu, Laurens van der Maaten, i Kilian Q. Weinberger. Densely connected convolutional networks. *CVPR*, 2017. doi: 10.48550/ARXIV.1608.06993. URL https://arxiv.org/abs/1608.06993.

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

ICLR, 2014. doi: 10.48550/ARXIV.1412.6980. URL https://arxiv.org/abs/1412.6980.

[9] Ivan Kreso, Josip Krapac, i Sinisa Segvic. Efficient ladder-style densenets for semantic segmentation of large images. *IEEE Trans. Intell. Transp. Syst.*, 22(8): 4951–4961, 2021. doi: 10.1109/TITS.2020.2984894. URL https://doi.org/10.1109/TITS.2020.2984894.

[10] Jie Li, Allan Raventos, Arjun Bhargava, Takaaki Tagawa, i Adrien Gaidon. Learning to fuse things and stuff. *CoRR*, abs/1812.01192, 2018. URL http://arxiv.org/abs/1812.01192.

[11] Darryl Lin, Sachin Talathi, i Sreekanth Annapureddy. Fixed point quantization of deep convolutional networks. U Maria Florina Balcan i Kilian Q. Weinberger, urednici, *Proceedings of The 33rd International Conference on Machine Learning*, svezak 48 od *Proceedings of Machine Learning Research*, stranice 2849–2858, New York, New York, USA, 20–22 Jun 2016. PMLR. URL https://proceedings.mlr.press/v48/linb16.html.

[12] Jyh-Woei Lin. Artificial neural network related to biological neuron network: a review. *Advanced Studies in Medical Sciences*, 5(1):55–62, 2017.

[13] Marin Oršić i Siniša Šegvić. Efficient semantic segmentation with pyramidal fusion. *Pattern Recognition*, 110:107611, 2021.

[14] Keiron O'Shea i Ryan Nash. An introduction to convolutional neural networks, 2015. URL https://arxiv.org/abs/1511.08458.

[15] Sérgio Pereira, Adriano Pinto, Victor Alves, i Carlos A. Silva. Brain tumor segmentation using convolutional neural networks in mri images. *IEEE Transactions on Medical Imaging*, 35(5):1240–1251, 2016. doi: 10.1109/TMI.2016.2538465.

[16] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, i Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. U *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[17] Olaf Ronneberger, Philipp Fischer, i Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. U Nassir Navab, Joachim Hornegger, William M. Wells, i Alejandro F. Frangi, urednici, *Medical Image Computing*

and Computer-Assisted Intervention – MICCAI 2015, stranice 234–241, Cham, 2015. Springer International Publishing. ISBN 978-3-319-24574-4.

[18] Sebastian Ruder. An overview of gradient descent optimization algorithms. *CoRR*, abs/1609.04747, 2016. URL `http://arxiv.org/abs/1609.04747`.

[19] Sagar Sharma, Simone Sharma, i Anidhya Athaiya. Activation functions in neural networks. *towards data science*, 6(12):310–316, 2017.

[20] Evan Shelhamer, Jonathan Long, i Trevor Darrell. Fully convolutional networks for semantic segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4):640–651, 2017. doi: 10.1109/TPAMI.2016.2572683.

[21] Hao Wu, Patrick Judd, Xiaojie Zhang, Mikhail Isaev, i Paulius Micikevicius. Integer quantization for deep learning inference: Principles and empirical evaluation. *CoRR*, abs/2004.09602, 2020. URL `https://arxiv.org/abs/2004.09602`.

Semantička segmentacija harmoničkim gusto povezanim modelima

Sažetak

Tehnološki napredak omogućuje treniranje dubokih modela u razumnom vremenu. Modeli imaju raznolike upotrebe u raznim područjima poput medicine i prometa. Ovaj rad bavi se analizom prometnih scena te određivanjem objekata na slici, a to spada u područje semantičke segmentacije. Za rješavanje tog problema koristi se FC-HarDNet 70 arhitektura koja je pažljivo osmišljena tako da se smanji memorijsko opterećenje modela. FC-Hardnet 70 se pokazao puno efikasnijim od drugih sličnih modela poput DenseNet-a. Za treniranje i evaluaciju korišten je skup podataka CamVid. Opisani su osnovni pojmovi vezani za duboko učenje, korištene tehnologije i komentirani dobiveni rezultati.

Ključne riječi: Duboko učenje, konvolucijske neuronske mreže, semantička segmentacija, HardNet, CamVid

Title

Abstract

Technological advance allows us to train deep models in real-time. These models have a variety of uses in many different fields like medicine or traffic. This paper is focused on interpreting and understanding scenes from traffic. That problem is known as semantic segmentation and the main goal is to classify every pixel in the picture to its label. To solve this problem, we used architecture FC-HarDNet 70 which was designed carefully to reduce dynamic random memory access(DRAM). FC-HarDNet 70 has proven to be more efficient than previous models like DenseNets. For training and evaluation purposes we used the CamVid dataset. In this paper, we explained fundamental concepts in deep learning as well as used technologies. In the end, we published and interpreted the given results of the experiment.

Keywords: Deep learning, convolutional neural networks, semantic segmentation, HarDNet, CamVid