SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1654

POLUNADZIRANO UČENJE SEMANTIČKE SEGMENTACIJE

Vedran Moškov

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1654

POLUNADZIRANO UČENJE SEMANTIČKE SEGMENTACIJE

Vedran Moškov

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

ZAVRŠNI ZADATAK br. 1654

Pristupnik:	**Vedran Moškov (0036543413)**
Studij:	Elektrotehnika i informacijska tehnologija i Računarstvo
Modul:	Računarstvo
Mentor:	prof. dr. sc. Siniša Šegvić

Zadatak: **Polunadzirano učenje semantičke segmentacije**

Opis zadatka:

Semantička segmentacija važan je zadatak računalnog vida s mnogim važnim primjenama. Jedna od velikih prepreka prema boljoj generalizaciji je nedostatak označenih slika za učenje. Zbog toga su vrlo zanimljive metode za učenje nad nepotpuno označenim skupom slika. Ovaj rad razmatra polunadzirano učenje gdje je većina slika za učenje potpuno neoznačena. U okviru rada, potrebno je odabrati okvir za automatsku diferencijaciju te upoznati biblioteke za rukovanje matricama i slikama. Proučiti i ukratko opisati postojeće segmentacijske arhitekture utemeljene na konvolucijama i pažnji. Uhodati standardno učenje modela na potpuno označenim podatcima te isprobati polunadzirani pristup utemeljen na jednostavnoj konzistenciji. Validirati hiperparametre, vrednovati generalizacijsku izvedbu te prikazati i ocijeniti postignutu točnost. Radu priložiti izvorni i izvršni kod razvijenih postupaka, ispitne slijedove i rezultate te potrebna objašnjenja i dokumentaciju. Citirati korištenu literaturu i navesti dobivenu pomoć.

Rok za predaju rada: 14. lipnja 2024.

Zahvaljujem prof. dr. sc Siniši Šegviću te asistentu univ. mag. ing. Ivanu Martinoviću na pomoći i korisnim savjetima tijekom izrade ovog rada

Sadržaj

Uvod

Računalni vid je područje u računalnoj znanosti koje se fokusira na omogućavanju računalima da identificiraju i prepoznaju različite objekte na slikama i videozapisima. Kao i drugi oblici umjetne inteligencije, računalni vid nastoji izvoditi i automatizirati zadatke koji repliciraju ljudske mogućnosti. U ovom slučaju, cilj je reproducirati način na koji ljudi vide, i način na koji interpretiraju ono što vide. Postoje brojne grane računalnog vida, no u ovom radu ćemo fokus staviti na semantičku segmentaciju slike.

Semantička segmentacija algoritam je dubokog učenja koji umjesto klasificiranja cijele slike, klasificira svaki piksel slike zasebno. Na taj način, modelu je omogućeno prepoznavanje pojedinih skupina piksela koji formiraju različite kategorije odnosno klase. Uzevši to u obzir, izlaz modela temeljenog na obavljanju semantičke segmentacije neće biti klasa u koju slika pripada, već takozvana maska slike u kojoj svaki piksel poprima jednu od boja, gdje svaka boja predstavlja jednu kategoriju odnosno klasu.

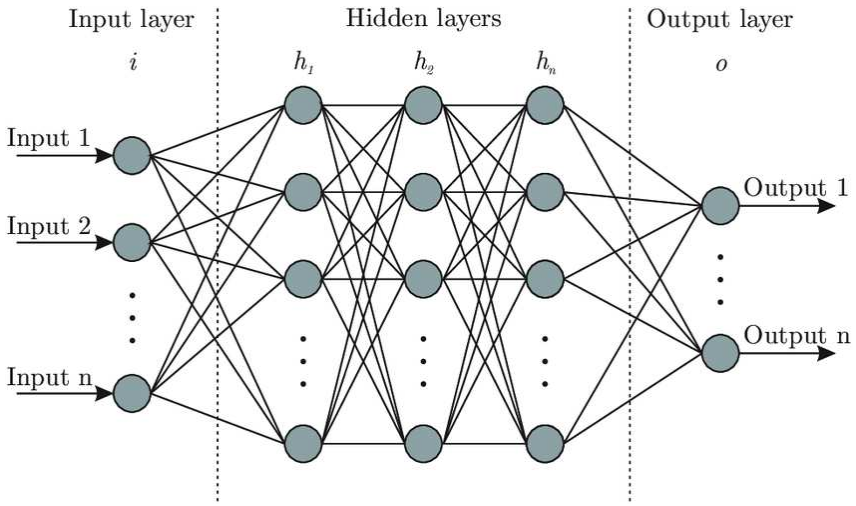
Naime, jedna od najvećih prepreka u dubokom učenju i računalnom vidu, pa tako i u semantičkoj segmentaciji, je nedostatak označenih slika. Danas najpopularniji pristup tome problemu jest polunadzirano učenje. Polunadzirano učenje je grana dubokog učenja koja kombinira nadzirano i nenadzirano učenje koristeći označene i neoznačene podatke kako bi se uspješno obavili zadaci klasifikacije i regresije. Primjer jednog takvog skupa podataka jest Cityscapes, koji sadrži ogromnu količinu podataka od kojih je veći dio precizno označen, a nešto je manje grubo označenih primjera. Za neoznačene podatke uzimamo dio označenih podataka te ignoriramo njihove oznake. Cilj ovog rada je uzeti neku arhitekturu baziranu na semantičkoj segmentaciji, npr. SwiftNet baziran na ResNet-u, uzeti određeni udio označenog skupa za treniranje te istrenirati model. Zatim ćemo se ponašati da je ostatak označenog skupa za treniranje neoznačen te pomoću tog modela generirati oznake taj ostatak skupa. Konačno, istrenirat ćemo model ponovno na cijelom skupu podataka koji je sačinjen djelomično od označenih podataka i djelomično od podataka s generiranim oznakama te usporediti rezultate s modelom treniranom na potpuno označenom skupu podataka.

1. Duboko učenje

Duboko učenje je podskup strojnog učenja koji koristi višeslojne neuronske mreže, odnosno duboke neuronske mreže, kako bi se simulirala kompleksna moć donošenja odluka koju ima ljudski mozak. Te mreže se nazivaju dubokima, zato što se sastoje od više slojeva koji idu dubinu i međusobno su povezani nelinearnim transformacijama. U klasifikacijskim problemima računalnog vida kao vrlo dobra arhitektura pokazale su se konvolucijske neuronske mreže koje uče na značajkama.

1.1. Neuronske mreže

Neuronska mreža matematički je model inspiriran strukturom i funkcijom bioloških neuronskih mreža u mozgovima ljudi i životinja. Umjetne neuronske mreže sastoje se od takozvanih umjetnih neurona koji oponašaju neurone u mozgu. Oni su povezani rubovima koje nazivamo težinama, a one oponašaju sinapse u biološkim neuronima. Tipično, neuroni su posloženi u slojeve, a neuronska se mreža može sastojati od jednog ili više takvih slojeva gdje svaki sloj primjenjuje neku transformaciju nad svojim ulazima. Signali koje neuronska mreža primi na prvom (ulaznom) sloju putuje kroz mrežu prema zadnjem (izlaznom) sloju prolazeći kroz opcionalne skrivene slojeve. Mrežu koja sadrži barem dva skrivena sloja nazivamo dubokom neuronskom mrežom.



Slika 1.1.1 - Arhitektura umjetne neuronske mreže [2]

Svaki neuron *k*-tom sloju dobiva signale od povezanih neurona iz *k-1*-og sloja, procesira ih te ih šalje neuronima s kojima je povezan u *k+1*-om sloju *(Slika 1.1.1)*. Transformacije koje

pojedini slojevi neurona rade nad signalima prije nego ih šalju sljedećem sloju nazivamo aktivacijskim funkcijama.

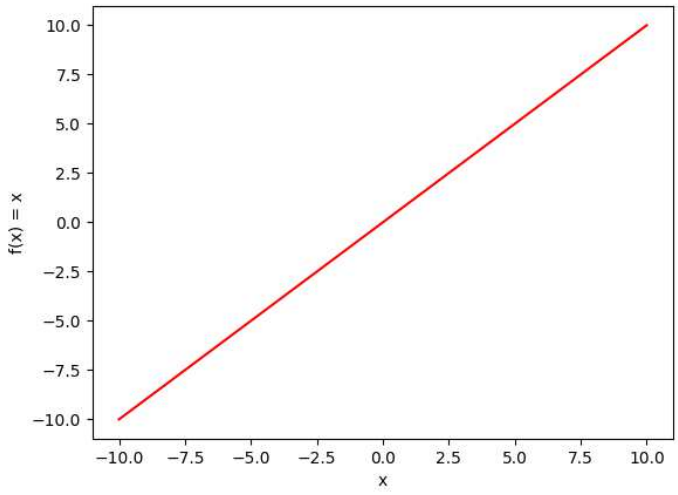
1.2. Aktivacijske funkcije

Aktivacijska funkcija neurona u umjetnoj neuronskoj mreži jest funkcija koja računa izlaz neurona temeljeno na njegovom ulazu i težinama. Težine u neuronskim mrežama predstavljaju hiperparametre koje koristimo za množenje izlaznih vrijednosti pojedinih neurona koji su usmjereni prema drugim neuronima. One označavaju važnost veze između dva neurona. Najčešće korištene aktivacijske funkcije su logistička (sigmoidalna) funkcija, tangens hiperbolni, funkcija skoka te sve popularnije funkcije ReLU (zglobnica) i propusni ReLU (propusna zglobnica) koje nam omogućavaju puno bolje treniranje dubokih neuronskih mreža.

1.2.1. Funkcija identiteta

Funkcija identiteta je funkcija oblika:

$$f(x) = x \tag{1.2.1}$$



Slika 1.2.1 - Funkcija identiteta

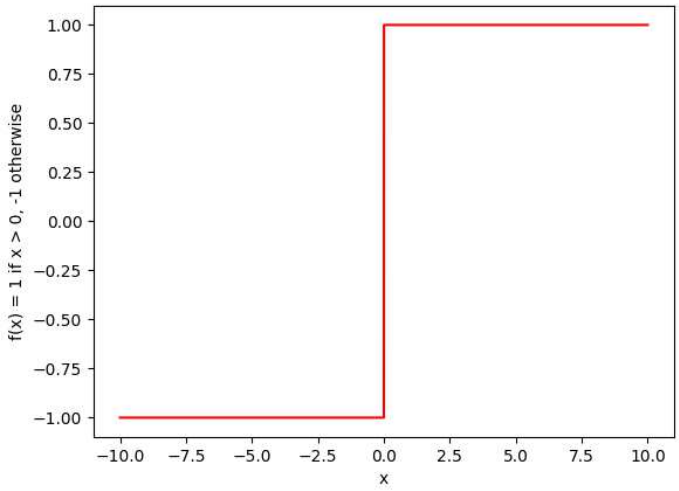
Funkcija identiteta *(Slika 1.2.1)* je funkcija koja kao svoj izlaz ima vrijednost identičnu ulazu, stoga se jako rijetko koristi za rješavanje složenijih problema zbog toga što je jako neinformativna. Razlog tomu je što ako na neku linearnu kombinaciju primijenimo linearnu

kombinaciju, rezultat je nova linearna kombinacija. Stoga mreža koja sadrži više slojeva, a sve aktivacijske funkcije među njima su linearne, je jednako informativna kao i neuronska mreža koja ima samo ulazni i izlazni sloj.

1.2.2. Funkcija skoka

Funkcija skoka je funkcija oblika:

$$f(x) = \begin{cases} -1, & x < 0 \\ 1, & x \geq 0 \end{cases} \tag{1.2.2}$$



Slika 1.2.2 - Funkcija skoka

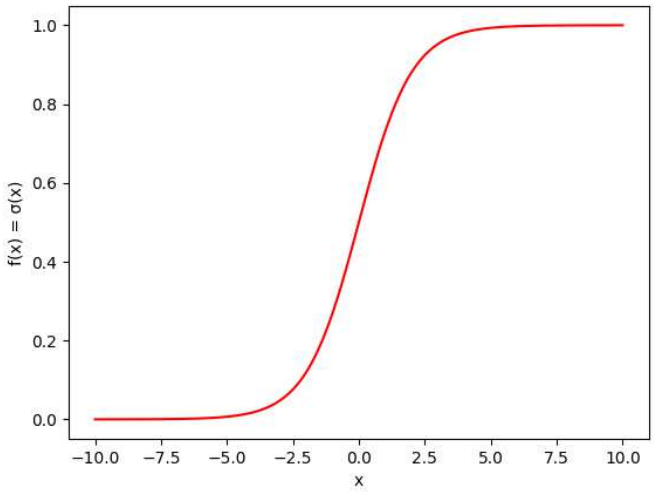
Funkcija skoka *(Slika 1.2.2)* je jedna od najjednostavnijih i najstarijih funkcija aktivacije neuronskih mreža. Ova funkcija se istaknula kod algoritma perceptrona, gdje se koristila za aktivaciju odnosno deaktivaciju neurona ovisno o tome prelazi li njihova vrijednost određenu granicu. Međutim, unatoč svojoj jednostavnosti i inicijalnoj popularnosti, funkcija skoka ima podosta ograničenja zbog čega se sve manje koristi u modernim modelima dubokog učenja. Prvi problem je to što je funkcija skoka nediferencijabilna u $x = 0$, dok je na svim drugim mjestima gradijent jednak nuli. Ta dva svojstva čine funkciju vrlo nepovoljnom za ikakvu optimizaciju temeljenu na gradijentu. Drugi problem je to što su izlazi te funkcije binarni, što uvelike ograničava ekspresivnost. Jedini mogući izlazi su -1 i 1 što sprječava mogućnost da model razazna ikakve složenije uzorke. Također, čak i ako bi

zagladili funkciju skoka tako da ona postane diferencijabilna, gradijenti bi i dalje bili ili 0 ili nedefinirani, što bi dovodilo do problema pri učenju dubokih neuronskih mreža.

1.2.3. Logistička (sigmoidalna) funkcija

Funkcija sigmoide je funkcija oblika:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{1.2.3}$$



Slika 1.2.3 - Sigmoidalna funkcija

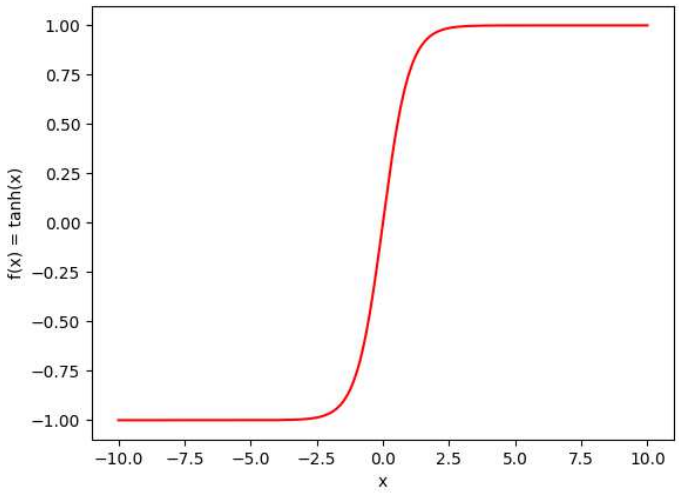
Sigmoidalna funkcija *(Slika 1.2.3)* je funkcija koja svaki realni broj preslikava na interval između 0 i 1. To je svojstvo jako korisno za transformiranje ulaza u vrijednost izlaza u obliku vjerojatnosti. Još jedna od korisnih karakteristika ove funkcije je činjenica da je funkcija sigmoide glatka i diferencijabilna, što ju čini vrlo pogodnom za razne optimizacijske algoritme temeljene na gradijentu kao što je prolaz unatrag *(engl. Backpropagation)*. Također, nelinearna priroda funkcije dozvoljava neuronskim mrežama da uče puno kompleksnije veze između podataka. No, unatoč svim tim kvalitetama, sigmoidalna funkcija ima i svoje nedostatke. Jedan od većih problema ove funkcije jest problem curenja gradijenta, koji se manifestira tako da za vrlo velike ili vrlo male vrijednosti ulaza, gradijent postaje ekstremno malen. U tim slučajevima gradijenti bivaju premaleni da bi se težine mogle efektivno mijenjati što značajno usporava konvergenciju te u ekstremnim slučajevima može dovesti i do prestanka učenja mreže. Još jedan problem je što vrijednosti funkcije nisu

centrirani oko 0. Svi izlazi iz funkcije sigmoide su uvijek pozitivni, što može dovesti do neefektivnosti u optimizaciji temeljenoj na gradijentu. Također, logistička funkcija je relativno skup proces, zbog eksponencijalne funkcije koja se nalazi u njoj, stoga ona sve više i više biva zamijenjena drugim funkcijama s manjom složenosti.

1.2.4. Tangens hiperbolni

Tangens hiperbolni jest funkcija koja ima sljedeći oblik:

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{1.2.4}$$



Slika 1.2.4 - Tangens hiperbolni

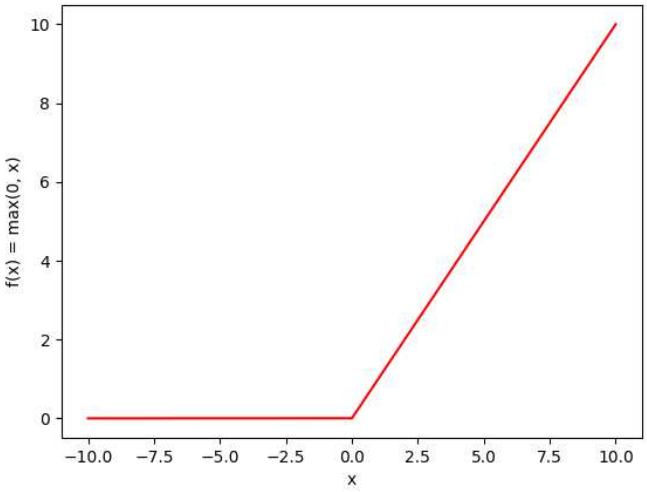
Funkcija tangensa hiperbolnog *(Slika 1.2.4)* je funkcija slična sigmoidalnoj, no ona svoje ulaze mapira na izlaze u intervalu između -1 i 1. To nam svojstvo pomaže u centriranju vrijednosti podataka što čini optimizaciju gradijentom puno balansiranijom zbog uravnoteženih pozitivnih i negativnih aktivacija. Isto kao i sigmoidalna funkcija, tangens hiperbolni je gladak i diferencijabilan na cijeloj svojoj domeni, zbog čega lakše uči kompleksne uzorke. Tangens hiperbolni se, za razliku od sigmoidalne funkcije, češće koristi u skrivenim slojevima neuronskih mreža upravo zbog toga što je centriran oko nule, no još jedna prednost naspram sigmoide jest da su njegovi gradijenti puno strmiji što često dovodi do brže konvergencije. No, u tangensu hiperbolnom su također prisutni problemi nestajućih

gradijenata te kompleksnost računanja, zbog čega biva zamijenjen drugim, konkurentnijim aktivacijskim funkcijama.

1.2.5. Zglobnica (ReLU)

Funkcija zglobnice odnosno ReLU funkcija ima oblik:

$$ReLU(x) = \max(0, x) \tag{1.2.5}$$



Slika 1.2.5 - Funkcija zglobnice (ReLU)

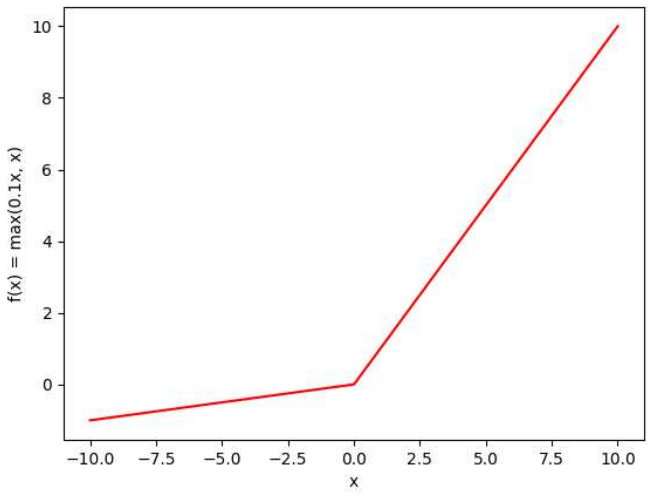
ReLU *(eng. Rectified Linear Unit) (Slika 1.2.5)* je funkcija koja je sve popularnija u procesu učenja dubokih neuronskih mreža. Njen oblik *(1.2.5)* nam govori da je vrijednost izlaza jednaka ulazu u slučaju kada je $x > 0$, a inače je izlaz jednak nuli. Funkcija zglobnice je također nelinearna, no ova funkcija nam nudi svojstvo rijetkosti. Rijetkost je pojava u kojoj imamo rijetke aktivacije neurona, što znači da će dosta neurona ostati neaktivno (ugašeno), odnosno imat će izlaz 0. To nam je korisno zato što su rijetke reprezentacije obično učinkovitije i puno lakše za optimizirati. Također, funkcija zglobnice je računski vrlo jednostavno izvediva, što značajno ubrzava i prolaz unaprijed i prolaz unatrag. Derivacija ove funkcije je 1 ili 0, što rješava probleme nestajućih gradijenata koje smo imali u ranije spomenutim aktivacijskim funkcijama, rješava problem brzine računanja i znatno poboljšava i ubrzava konvergenciju. Međutim, i ova funkcija unatoč svim svojim kvalitetama povlači i neke nedostatke. Najprominentniji problem ove funkcije je takozvani „Umirući ReLU

problem“, gdje neuroni znaju „umrijeti“ za vrijeme treniranja modela. Ako neki neuron predugo zaglavi na negativnoj strani zglobnice, njegov izlaz pa time i gradijent će biti jednaki nuli te se on neće nikada ažurirati za vrijeme treniranja. To nas dovodi do situacije gdje su dijelovi mreže potpuno neaktivni. Također, zglobnica je neograničena funkcija, stoga aktivacijske vrijednosti znaju dosezati velike vrijednosti, što nekada zna destabilizirati proces učenja.

1.2.6. Propusna zglobnica (Leaky ReLU)

Propusna zglobnica je funkcija oblika:

$$Leaky\ ReLU(x) = \begin{cases} x, & x > 0 \\ \alpha x, & x \leq 0 \end{cases} \tag{1.2.6}$$



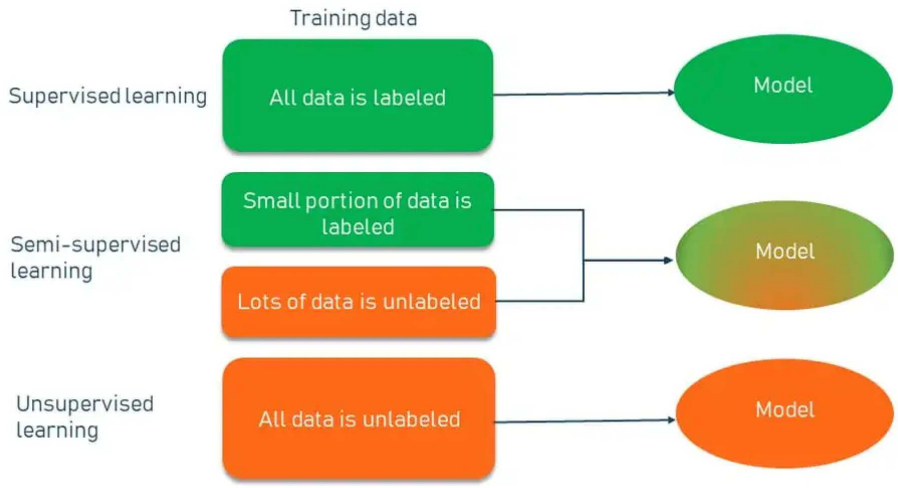
Slika 1.2.6 - Propusna zglobnica (Leaky ReLU)

Propusna zglobnica *(Slika 1.2.6)* je funkcija nastala kao odgovor na probleme sa već spomenutom funkcijom zglobnice. Definirana je jednadžbom *(1.2.1)* gdje je α neka mala pozitivna konstanta (često postavljena na vrijednost 0.01). Ova modifikacija na funkciju zglobnice omogućava mali gradijent različit od nule kada je ulaz negativan. Dobra stvar ove funkcije je što zadržava nelinearnost, rijetkost i jednostavnu izračunljivost funkcije zglobnice. Njena prednost nad zglobnicom se očituje ublažavanjem pojave umirućih neurona dozvoljavajući mali gradijent negativnih ulaza i time osiguravajući da oni i dalje nastave učiti tijekom treniranja. Taj mali gradijent također popravlja tok gradijenta u prolazu

unatrag, potencijalno vodeći do boljih performansa i brže konvergencije. Jedini nedostatak ove funkcije, unatoč neomeđenosti, jest taj da izbor koeficijenta α zna biti dosta težak izazov, zato što premala vrijednost može dovesti do vrlo malih prednosti nad običnom zglobnicom, dok prevelike vrijednosti mogu rezultirati eksplodirajućim gradijentima.

1.3. Učenje neuronskih mreža

Učenje neuronskih mreža je iznimno kompleksan proces koji označava brojne pristupe različitim implementacijama matematičkih i računalnih algoritama i formula. To su procesi kojima treniramo neuronsku mrežu da obavljaju neki unaprijed definirani posao. Glavna podjela učenja je na sljedeće tri skupine, nadzirano učenje, polunadzirano učenje te nenadzirano učenje.



Slika 1.3.1 – Različiti pristupi dubokom učenju [3]

1.3.1. Nadzirano učenje

Nadzirano učenje je paradigma u dubokom učenju gdje u procesu učenja, model „hranimo“ podacima koji su strukturirani kao parovi ulaza i željenih izlaza. Ti željeni izlazi su također poznati kao ljudski označeni nadzorni signali. Na temelju dobivenih i željenih izlaza, raznim algoritmima gradi se funkcija modela čiji je cilj mapiranje ulaza na željene izlaze. U optimalnom slučaju, ta funkcija bi trebala jednako dobro raditi i za još neviđene podatke. Za to je potrebno da model na temelju skupa za treniranje dobro generalizira populaciju kako bi radilo dovoljno dobro i za neviđene primjerke. Mjera kvalitete obavljanja posla modela

se računa statističkim algoritmom generalizacijske pogreške, koji se još naziva i funkcijom gubitka.

1.3.2. Nenadzirano učenje

Nenadzirano učenje jest proces učenja modela temeljenih na dubokom učenju na kompletno neoznačenim podacima. Ideja ovog pristupa je da model samostalno pokuša izvući i prepoznati skrivene uzorke, razlike i sličnosti među pojedinim dijelovima podatkovnog skupa, bez ikakve ljudske intervencije. U sklopu te metode, provodi se proces grupiranja *(eng. Clustering)* kako bi se podaci grupirali u zasebne skupine temeljene na međusobnim sličnostima.

1.3.3. Polunadzirano učenje

Polunadzirano učenje [4] je tehnika kojom kombiniramo nadzirano učenje i nenadzirano učenje *(Slika 1.3.1)*, na način da koristimo mali udio označenih podataka i veliku količinu neoznačenih podataka kako bi istrenirali prediktivne modele. Problem nadziranog učenja jest da zahtjeva ogromne količine označenih podataka, što najčešće uključuje veliku količinu ljudskih napora kako bi podaci bili precizno označeni. S druge strane, problem nenadziranog učenja jest da je spektar primjene vrlo uzak, i da su rezultati vrlo često jako niske točnosti, stoga se ta metoda najčešće koristi u pretprocesiranju podataka kako bi ljudi sami sebi lakše objasnili pojedine sličnosti i razlike među podacima prije nego ih krenu označavati. Stoga polunadzirano učenje uparuje te dvije tehnike i tako premošćuje njihove nedostatke. Tom metodom najprije istreniramo model na malom udjelu označenih podataka, a zatim za ostatak neoznačenih podataka iterativno generiramo oznake iz naučenog modela. Ova metoda je vrlo obećavajuća, zato što postoji obilje neoznačenih podataka, i ono je vrlo dostupno, dok točnost rezultata pritom ne pati. Cilj ovog rada je utvrditi točnost polunadziranog učenja te usporediti rezultate s dobivenim rezultatima nadziranog učenja.

1.3.4. Funkcija gubitka

U matematičkoj optimizaciji, pa tako i u algoritmima dubokog učenja, funkcija gubitka odnosno funkcija cijene je funkcija koja mapira vrijednosti jedne ili više varijabli na neki realni broj koji efektivno predstavlja cijenu odnosno pogrešku nekog algoritma ili događaja.

Unakrsna entropija

Funkcija unakrsne entropije *(engl. Cross entropy loss function)* je funkcija gubitka koja je uglavnom korištena u višeklasnim klasifikacijskim problemima, a kako je problem semantičke segmentacije upravo jedan takav problem, i u ovom radu smo odlučili koristiti unakrsnu entropiju. Unakrsna entropija jest funkcija sljedećeg oblika:

$$loss(y, \hat{y}) = -\sum_{c=1}^{C} y_c \cdot \log(\hat{y}_c)$$

(1.3.1)

Gdje je C broj različitih klasa, y_c je binarni indikator koji nam govori je li klasa c točna klasa za danu obzervaciju, dok je $\hat{y}_c$ predviđena vjerojatnost pripadnosti dane obzervacije klasi c koja je određena softmax funkcijom u izlaznom sloju. U konačnici, cilj postupka treniranja klasifikacijskog modela dubokog učenja jest minimizacija upravo ovakve jedne funkcije, u našem slučaju, funkcije unakrsne entropije.

1.3.5. Optimizacijski algoritmi

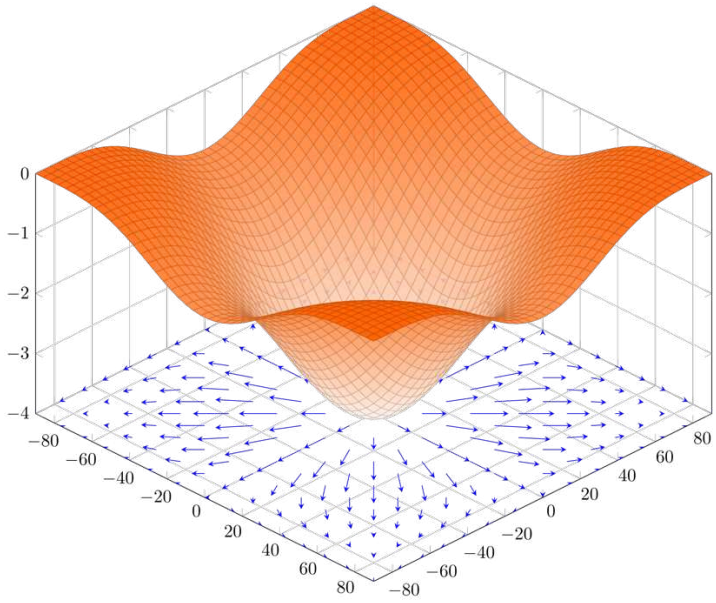
Optimizacijski algoritmi su ključna stvar u treniranju dubokih neuronskih mreža. Upravo optimizacijski algoritmi su oni koji nam govore u kojem smjeru i u kojim razmjerima moramo mijenjati hiperparametre naše neuronske mreže kako bi onda davala bolje rezultate, odnosno kako bi minimizirali funkciju gubitka. Uz obilje dostupnih optimizacijskih algoritama, najveći izazov današnjice jest odabrati onaj algoritam koji je najprikladniji za problem koji se nalazi pred nama. Algoritmi se biraju u ovisnosti o segmentu procesa treniranja kojeg želimo unaprijediti, bilo to cilj da se unaprijedi točnost, skrati vrijeme potrebno za obavljanje treninga, ili pak nešto drugo. U našem slučaju, korišten je algoritam gradijentnog spusta *(eng. Gradient Descent)*.

Gradijentni spust

Gradijentni spust [5] je algoritam dizajniran kako bi uspješno minimizirao funkciju inkrementalnim pomicanjem prema njezinoj minimalnoj vrijednosti. U matematičkoj analizi, gradijent je definiran kao vektor koji nam govori u kojem smjeru funkcija najbrže raste iz točke u kojoj se nalazimo. Gradijent označavamo znakom ∇ i računamo ga prema sljedećem izrazu:

$$\vec{\nabla} f(p) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(p) \\ \vdots \\ \frac{\partial f}{\partial x_n}(p) \end{bmatrix}, \quad p = (x_1, x_2, \dots, x_n)$$

(1.3.2)



Slika 1.3.2 - Vizualni prikaz gradijenta [6]

Stoga, imajući to na umu, postupak gradijentnog spusta se efektivno svodi na računanje gradijenta funkcije gubitka u zadanoj točki, a zatim kretanje u smjeru suprotnom od gradijenta, i tako inkrementalno provoditi postupak za svaku točku u kojoj se nađemo na putu prema minimumu funkcije.

Ako je naša funkcija $f(\vec{x})$ skalarna, tada gradijent te funkcije sadrži parcijalne derivacije po svim varijablama. Ako gradijent promatramo kao stupčasti vektor, gradijent će odgovarati transponiranoj Jakobijevoj matrici:

$$\vec{\nabla} f(\vec{x}) = \frac{df(\overrightarrow{x})}{\vec{x}}^{\top}$$

(1.3.3)

Ako funkciju f sada aproksimiramo Taylorovim razvojem prvog reda, dobivamo sljedeću aproksimaciju:

$$f(\vec{x} + \Delta\vec{x}) \approx f(\vec{x}) + \frac{df(\overrightarrow{x})}{\vec{x}}\,\Delta\vec{x}$$

(1.3.4)

Sada kada bismo uvrstili izraz (1.3.3) u jednadžbu (1.3.4), dobili bismo sljedeći izraz:

$$f(\vec{x} + \Delta\vec{x}) \approx f(\vec{x}) + \vec{\nabla} f(\vec{x})^{\top}\,\Delta\vec{x}$$

(1.3.5)

Ako u pomak u vektoru $\vec{x}$ uvrstimo pomak u smjeru negativnog gradijenta, dobivamo formulu:

$$\Delta\vec{x} = -\epsilon \cdot \vec{g}\,, \quad \vec{g} = \vec{\nabla} f(\vec{x})$$

(1.3.6)

Nakon što izraz (1.3.6) uvrstimo u izraz (1.3.5) te ako Taylorova aproksimacija vrijedi, vrijednost funkcije f bi trebala opadati:

$$f(\vec{x} - \epsilon \cdot \vec{g}) \approx f(\vec{x}) - \epsilon \cdot \vec{g}^{\top}$$

(1.3.7)

Sada vidimo da iterativno pomicanje u smjeru negativnog gradijenta može dovesti do lokalnog minimuma funkcije f. Zbog toga nam je ova metoda poznata kao gradijentni spust. U praksi, aproksimacija će biti tim točnija što je hiper-parametar ϵ manji. Međutim, nije dobro da ϵ bude premali kako učenje ne bi previše trajalo. Zbog toga su smišljene brojne poboljšane metode ovog pristupa, no sve se temelje na istoj spomenutoj ideji.

1.3.6. Prolaz unatrag

U dubokom učenju, prolaz unatrag *(engl. Backpropagation)* [7] je metoda procjene gradijenta koju koristimo kao optimizacijsku metodu za treniranje dubokih neuronskih mreža. Ta metoda je korak na temelju kojeg se mijenjaju hiperparametri neuronske mreže, a provodi se nakon što se izvede cijela jedna epoha treniranja i izračuna se funkcija gubitka. Algoritam prolaska unatrag je vrlo efektivna primjena Leibnizovog pravila ulančavanja:

$$\frac{df(g(x))}{dx} = \frac{df(g(x))}{dg(x)} \cdot \frac{dg(x)}{dx}$$

(1.3.8)

Formula (1.3.8) nam treba upravo iz razloga što se ideja ovog algoritma temelji na računanju gradijenta funkcije gubitka s obzirom na težine modela i s obzirom na vektore pristranosti. To možemo izraziti sljedećim formulama koje koristimo:

$$\frac{\partial L_i}{\partial \vec{w}_{j_k}} = \frac{\partial L_i}{\partial s_{j_{ik}}} \cdot \frac{\partial s_{j_{ik}}}{\partial \vec{w}_{j_k}}$$

(1.3.9)

$$\frac{\partial L_i}{\partial b_{j_k}} = \frac{\partial L_i}{\partial s_{j_{ik}}} \cdot \frac{\partial s_{j_{ik}}}{\partial b_{j_k}}$$

(1.3.10)

gdje je L naša funkcija gubitka, a $s_{j_{ik}}$ označava izlaz iz k-tog neurona u j-tom sloju za i-ti uzorak. U izrazu (1.3.9) $\vec{w}_{j_k}$ označava k-ti redak matrice W_j gdje j označava težine prema $j+1$-om sloju, nam je u izrazu (1.3.10) b_{j_k} oznaka za pristranost *(eng. bias)* k-tog neurona za $j+1$-i sloj [8]. Naime, ovakav postupak računanja gradijentnog spusta na razini cijelog podatkovnog skupa bila bi preskupa operacija, dok bi postupak za svaki uzorak zasebno bio previše nepredvidiv i imao šumova i time znatno usporio konvergenciju. Zato koristimo postupak stohastičkog gradijentnog spusta *(eng. Stohastic Gradient Descent)* koji koristi pristup računanja po malim podskupovima podataka *(eng. mini batches)* veličine n.

Kada bi taj pristup pretvorili u formule, one bi izgledale ovako:

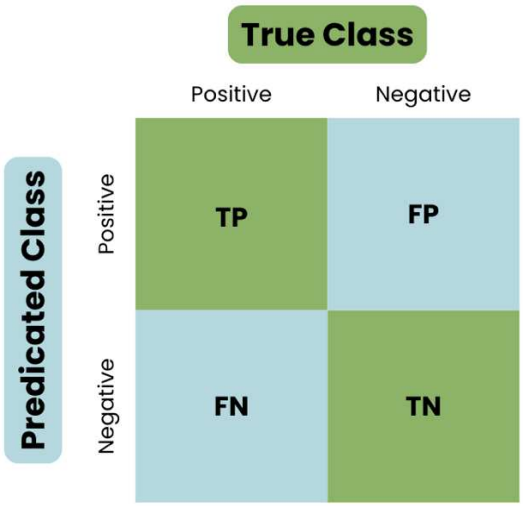
$$\vec{w_i}' = \vec{w_i} - \frac{\epsilon}{n}\sum_{k=1}^{n}\frac{\partial L_k}{\partial \vec{w_i}} \tag{1.3.11}$$

$$\vec{b_i}' = \vec{b_i} - \frac{\epsilon}{n}\sum_{k=1}^{n}\frac{\partial L_k}{\partial \vec{b_i}} \tag{1.3.12}$$

gdje su $\vec{w_i}'$ i $\vec{b_i}'$ i-ta težina odnosno vektor pristranosti nakon što je primijenjena korekcija.

1.3.7. Matrica zabune

Matrica zabune *(eng. confusion matrix)* je tablica koja je vrlo popularna i vrlo često korištena kao mjera u rješavanju klasifikacijskih problema. Matrica zabune nam predstavlja broj zapažanja predviđenih i stvarnih klasa.



Slika 1.3.3 - Osnovna struktura matrice zabune [9]

Matricu zabune dijelimo na sljedeće dijelove *(Slika 1.3.3)*:

- **True Positive (TP)** – model je ispravno predvidio pozitivnu klasu
- **True Negative (TN)** – model je ispravno predvidio negativnu klasu
- **False Positive (FP)** – model je previdio pozitivnu klasu, dok je točna klasa negativna
- **False Negative (FN)** – model je predvidio negativnu klasu, dok je točna klasa pozitivna

Na temelju tih kategorija definiramo sljedeće metrike:

- **Točnost (Accuracy)** – označava ukupan broj točnih klasifikacija podijeljen ukupnim brojem klasifikacija

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

(1.3.13)

- **Opoziv (Recall)** – označava ukupan broj točno označenih pozitivnih klasa podijeljen ukupnim brojem stvarnih pozitivnih klasa

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

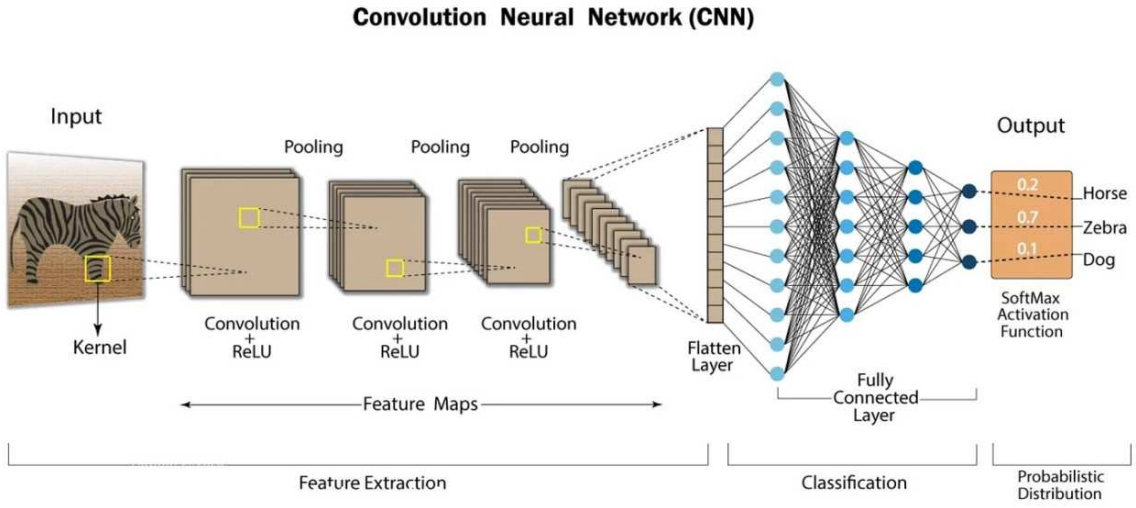
(1.3.14)

- **Preciznost (Precision)** – označava ukupan broj točno označenih pozitivnih klasa podijeljen ukupnim brojem previđenih pozitivnih klasa

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

1.4. Konvolucijski modeli

Konvolucijski modeli [10] odnosno konvolucijske neuronske mreže su regularizirani oblik neuronske mreže s povratnim prijenosom koji sami uče inženjering značajki putem optimizacije filtera ili jezgri. Ovi modeli rješavaju problem nestajućih ili eksplodirajućih gradijenata s kojima smo se susreli kod prolaza unatrag u ranije spomenutim neuronskim mrežama. Ovakve neuronske mreže su danas postale glavni temelj računalnog vida zato što se koriste regularizirane težine s puno manje poveznica. Na primjer, za svaki neuron u potpuno povezanom sloju, 10,000 težina bi bilo potrebno da se procesira slika veličine 100x100 piksela, dok uz primjenu konvolucija možemo postići da je samo 25 neurona potrebno za procesiranje pločica veličine 5x5 piksela.

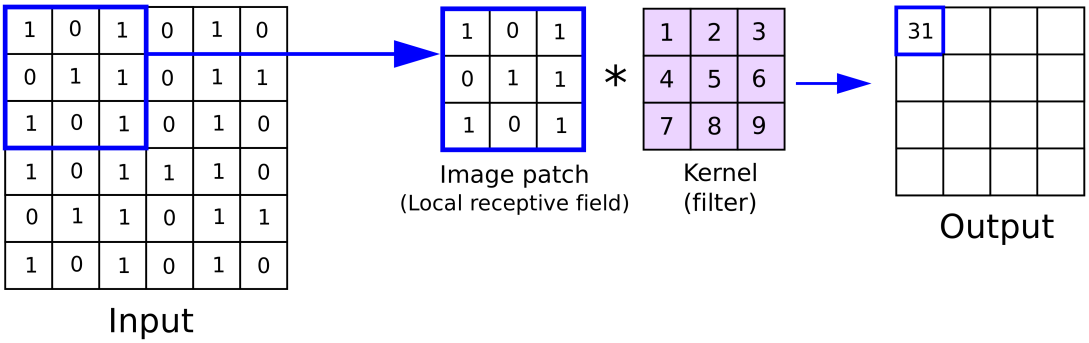


Slika 1.4.1 - Arhitektura konvolucijske neuronske mreže [11]

Konvolucijska mreža sastoji se od konvolucijskih slojeva, aktivacijskih funkcija, slojeva sažimanja *(eng. pooling layers)* te potpuno povezanih slojeva klasične neuronske mreže *(Slika 1.4.1).*

Konvolucijski sloj

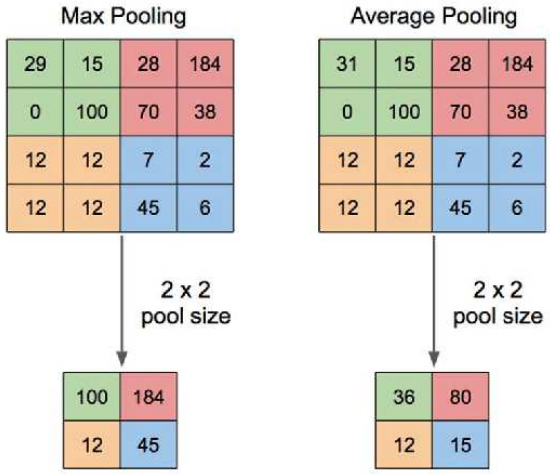
Konvolucijski sloj je sloj konvolucijskih neuronskih mreža sastoji se od filtara odnosno jezgri i mapa značajki *(eng. feature maps)*. Filtri su male matrice (npr. 5x5 ili 3x3) koje imaju sposobnost učenja, koje „kližu“ po ulaznoj slici kako bi detektirali lokalne uzorke kao što su rubovi, teksture ili boje. Filteri se primjenjuju tako da se na svakoj lokaciji obavi operacija množenja po elementima, a rezultati se zatim sumiraju u jednu vrijednost. Mape značajki su rezultati primjenjivanja tih filtera, isticajući specifične značajke ulaza na različitim lokacijama *(Slika 1.4.2)*.



Slika 1.4.2 - Klasifikacijski sloj konvolucijskih modela [12]

Sloj sažimanja

Osnovna zadaća sloja sažimanja je smanjivanje dimenzionalnosti mapi značajki. Time efektivno smanjujemo i računalnu snagu potrebnu za daljnje izračunavanje značajki kroz mrežu. Dva su najčešća pristupa sažimanju, maksimalno sažimanje *(eng. max pooling)* i prosječno sažimanje *(eng. average pooling)*.

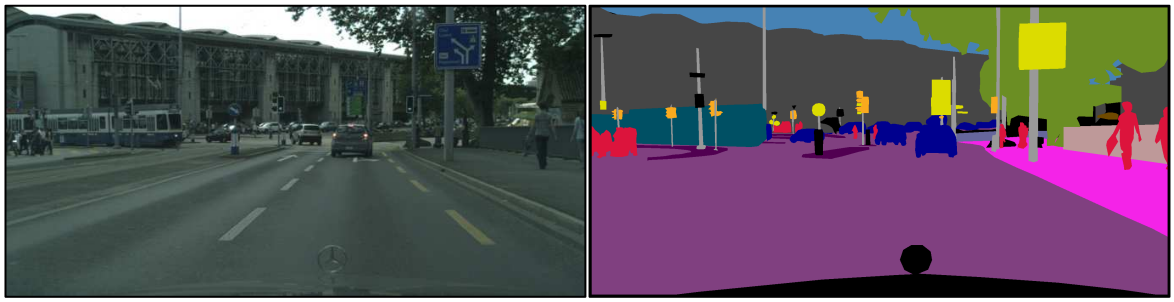


Slika 1.4.3 - Maksimalno i prosječno sažimanje [13]

Maksimalno sažimanje jest postupak u kojem s novom jezgrom proizvoljno malih dimenzija prolazimo po značajkama i uzimamo samo najveću vrijednost za svako nepreklapajuće područje, dok u prosječnom sažimanju uzimamo aritmetičku sredinu tih vrijednosti *(Slika 1.4.3)*. Osim što sažimanjem smanjujemo dimenzije značajki, također uklanjamo i šum stvoren pri kreiranju značajki.

2. Semantička segmentacija

Semantička segmentacija [14] jest zadatak računalnog vida u kojem je cilj svaki piksel slike pridružiti nekoj klasi odnosno objektu. Cilj je stvoriti segmentacijsku masku slike na temelju piksela. U procesu semantičke segmentacije u dubokom učenju i računalnom vidu, ulaz u model je nekakva slika, dok je izlaz modela upravo spomenuta segmentacijska mapa.

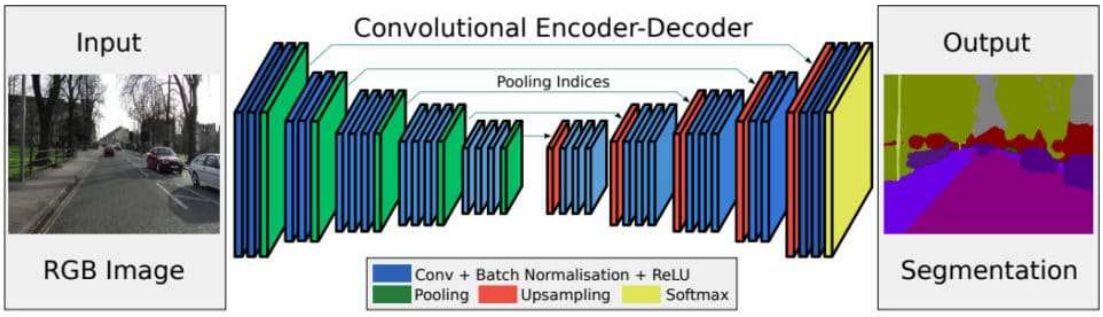


Slika 1.4.1 - Primjer slike i njene segmentacijske maske

Najčešće primjene ovog koncepta su sustavi za autonomnu vožnju, medicinska snimanja te industrijska inspekcija.

2.1. Arhitektura

Arhitektura korištena za zadatke semantičke segmentacije najčešće se sastoji od dvije glavne komponente, enkodera i dekodera *(Slika 2.1.1)*. Glavna zadaća enkodera jest ekstrakcija značajki visokih razina iz ulazne slike.

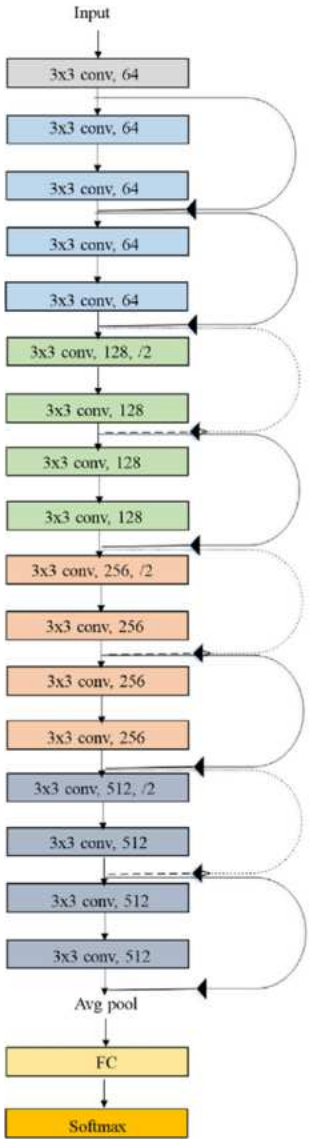


Slika 2.1.1 - Arhitektura modela za semantičku segmentaciju [15]

On najčešće u sebi sadrži neku konvolucijsku arhitekturu, npr. ResNet, MobileNet ili VGG. S druge strane, zadaća dekodera je generirati klasifikacijsku mapu na razini piksela kreiranu na temelju značajki visokih razina. Dekoder se sastoji od dekonvolucijskih slojeva i slojeva povećavanja rezolucije te preskočnih veza. Uloga dekonvolucijskih slojeva je u efektivno obrnut proces od konvolucije, gdje se nastoji podići rezolucija ulaznih značajki, bilo umetanjem nula između piksela ili pomičući nekakav filter po ulaznim značajkama na način na koji bi to povećalo rezoluciju. S druge strane, slojevi povećavanja rezolucije rezoluciju povećavaju na fiksnim interpolacijskim metodama kao što su metoda najbližih susjeda, bilinearna interpolacija ili bikubična interpolacija. Metodom najbližih susjeda se replicira vrijednost najbližeg piksela, bilinearnom interpolacijom uzimamo težinski prosjek najbliža 4 piksela, dok se bikubičnom interpolacijom uzima težinski prosjek najbližih 16 piksela kako bi se dobili glađi rezultati. Glavna razlika između dekonvolucijskih slojeva i slojeva povećanja rezolucije je činjenica da su dekonvolucijski slojevi sposobni naučiti, a računski su dosta zahtjevniji, dok slojevi povećanja rezolucije nisu toliko zahtjevni za izvesti no nemaju nikakvu sposobnost učenja. Uloga preskočnih veza dekodera jest da značajke iz određene razine enkodera kombinira direktno sa značajkama u korespondirajućoj razini dekodera koristeći konkatenaciju ili sumiranje vrijednosti po elementima.

2.2. SwiftNet

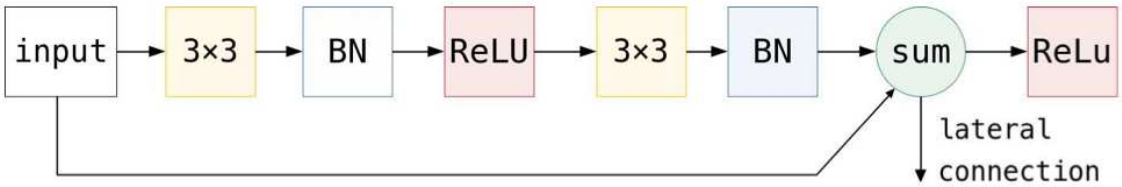
U našim eksperimentima korištena je arhitektura SwiftNet [18] koja je razvijena upravo na FER-u. Ona se sastoji od tri glavne građevne komponente, enkodera, dekodera i modula za povećanje receptivnog polja. Ulogu enkodera u ovoj arhitekturi jest poznata konvolucijska arhitektura ResNet-18 [16], koja se sastoji od 18 konvolucijskih slojeva *(Slika 2.2.1)*.



Slika 2.2.1 - Struktura konvolucijske arhitekture ResNet-18 [17]

Dekoder je nešto jednostavnije strukture, ostvaren kao niz modula za naduzorkovanje povezanih lateralnim (preskočnim) vezama. Predložena struktura prima dva ulaza, značajke male rezolucije te lateralne značajke iz ranijih korespondirajućih razina enkodera. Značajke niskih rezolucija su najprije naduzorkovane do rezolucije lateralnih značajki koristeći

bilinearnu interpolaciju te su one zatim međusobno posumirane po elementima. Nakon toga ih miješamo konvolucijom dimenzija 3x3.



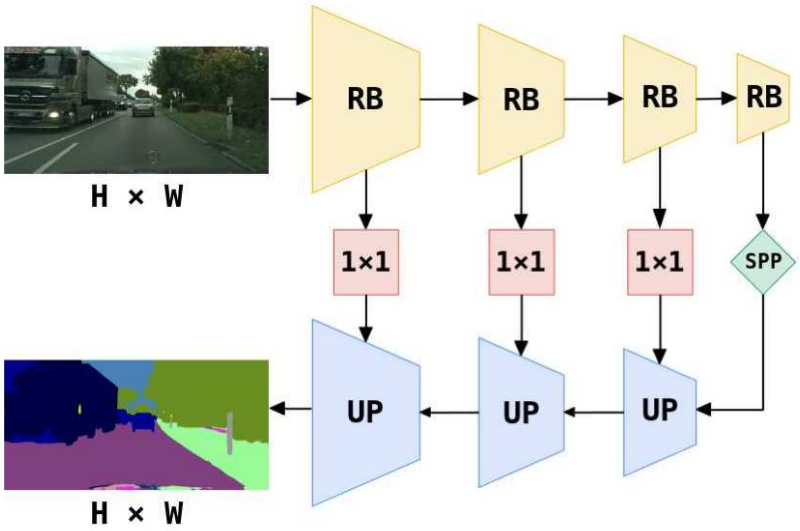
Slika 2.2.2 - Strukturni dijagram zadnje rezidualne jedinice u konvolucijskom bloku. Izlaz iz zglobnice je dalje proslijeđen sljedećem rezidualnom bloku. [18]

Ideja preskočnih veza u našoj predloženoj arhitekturi dekodera je uzimanje lateralnih značajki iz sume po elementima unutar odgovarajuće razine poduzorkovanja te slanje iste odgovarajućoj razini u dekoderu *(Slika 2.2.2)*.

Moduli za povećavanje receptivnog polja nam omogućavaju obavljanje zadataka povećanja receptivnog polja zadržavajući brzinu u stvarnom vremenu. Postoje dva načina za to, koristeći prostorno piramidalno sažimanje ili koristeći piramidalnu fuziju. U našem radu koristili smo prostorno piramidalno sažimanje *(Slika 2.2.)*.

Jednorazinski model (eng. Single scale model)

Predloženi jednorazinski model transformira ulaznu sliku u guste semantičke predikcije kroz poduzorkovanje enkoderom i naduzorkovanje dekoderom kao što je prikazano na *Slika 2.2.*. Žuti trapezi označavaju konvolucijske grupe. To su grupe koje operiraju na istoj prostornoj rezoluciji. Naš model se sastoji od 4 takve grupe, gdje prva grupa generira značajke razina H/4 × W/4, a svaki sljedeći poduzorkuje faktorom 2. Znači, posljednja grupa će generirati značajke dimenzija H/32 × W/32.



Slika 2.2.3 - Strukturni dijagram jednorazinskog modela. Žuti trapezi označavaju konvolucijske grupe enkodera koji mogu biti predtrenirani na ImageNet skupu. Zeleni dijamant predstavlja sloj prostornog piramidalnog sažimanja, crveni kvadrati predstavljaju takozvane „bottleneck“ slojeve, dok plavi trapezi označavaju dekodere. [18]

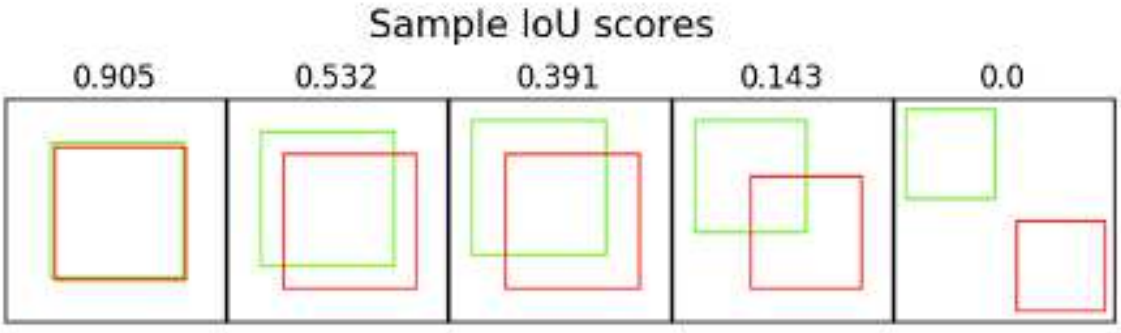
Tim značajkama „hranimo“ sloj prostornog piramidalnog sažimanja kako bi povećali receptivno polje našeg modela. Konačno, tenzor koji dobijemo kao rezultat te operacije prosljeđujemo dekoderima koji su prikazani plavom bojom.

2.3. Metrike

Za razliku od klasičnih klasifikacijskih metrika koje govore je li naš model točno predvidio klasu ili nije, u domeni semantičke segmentacije to nije ispravan pristup. Razlog tome je što je vrlo neizgledno da će naš model generirati masku koja se u potpunosti podudara s očekivanom maskom. Također, ako bi gledali na razini svakog piksela zasebno pri evaluaciji modela, time ne bismo direktno mjerili prostornu točnost i performanse lokaliziranja što su zapravo i najbitniji aspekti semantičke segmentacije. Iz tog razloga uvodimo novu metriku, presjek kroz unija *(eng. intersect over union, skraćeno IoU)*:

$$IoU = \frac{|A \cap B|}{|A \cup B|} \tag{2.3.1}$$

gdje nam A i B predstavljaju predviđenu odnosno točnu segmentacijsku masku *(Slika 2.3.1)*.

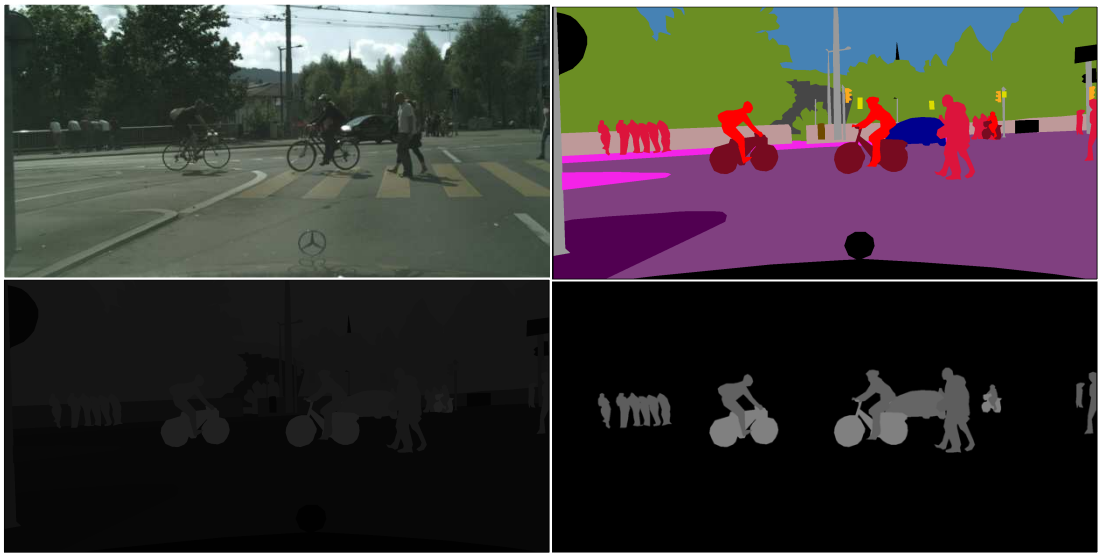


Slika 2.3.1 - Vizualna reprezentacija IoU metrike [19]

Taj postupak provodimo za svaku klasu u našem skupu zasebno te na temelju toga računamo prosječnu vrijednost *(eng. mean intersect over union, skraćeno mIoU)* kao konačnu vrijednost.

3. Cityscapes

Skup podataka Cityscapes [1] je skup koji se fokusira na semantičko razumijevanje urbanih uličnih scena. Skup se sastoji od 5000 precizno označenih i 20000 grubo označenih slika koje su prikupljene u 50 različitih gradova njemačkog govornog područja. Fotografije su slikane u različitim godišnjim dobima u različita doba dana te u različitim vremenskim uvjetima. Objekti su klasificirani u 30 kategorija, no u našem eksperimentu koristit ćemo samo 19 klasa, i to samo one slike koje su precizno označene.



Slika 3.1.1 - Primjer slike iz skupa Cityscapes. Originalna slika (gore lijevo), segmentacijska maska (gore desno), identifikatori klasa korištenih za treniranje (dolje lijevo), identifikatori objekata koji mogu imati više instanci (dolje desno) [1]

Skup podataka se podijeljen u tri podskupa, skup za treniranje, skup za unakrsnu validaciju te skup za testiranje. Naime, skup za testiranje nije javno dostupan, stoga su testiranja obavljena na skupu za validaciju.

4. Implementacija

Programska implementacija našeg rješenja izvedena je u programskom jeziku Python i njegovim raznim bibliotekama koje nude bogatu podršku za područje duboko učenje i računalnog vida. Biblioteke na kojima se ovaj rad temelji su PyTorch[1], NumPy[2] te OpenCV[3]. Dio eksperimenata obavljen je putem Google Colaba[4] koristeći Nvidia T4 GPU, a ostatak je obavljen na serverima Zavoda za elektroniku, mikroelektroniku, računalne i inteligentne sustave smještenog na FER-u. Odabrana arhitektura je već spomenuti SwiftNet temeljen na ResNet-18 arhitekturi.

4.1. PyTorch

PyTorch je radni okvir prilagođen dubokom učenju razvijen od strane kompanije Meta, koji je danas postao vodeći alat za razvoj i treniranje neuronskih mreža, a temelji se na objektima koje nazivamo tenzorima. Razlog zašto je ova biblioteka toliko popularna jest činjenica da ima implementirane dinamičke komputacijske grafove. To znači da se grafovi grade u hodu kako se operacije izvršavaju što ovu biblioteku čini jako intuitivnom za korištenje. Također, ova biblioteka ima implementaciju alata za automatsko diferenciranje imena „Autograd“ što nam računa gradijente automatski. Ta nam mogućnost uvelike olakšava i ubrzava algoritam prolaska unatrag i algoritme optimizacije. PyTorch također nudi mnoge opcije sa manipulaciju slikama što je jako pogodno za područje računalnog vida.

4.2. NumPy

NumPy je fundamentalna biblioteka za znanstveno računanje u Pythonu. Ova biblioteka nam nudi vrlo široku podršku za efikasne operacije nad poljima i matricama te mnoge druge matematičke funkcije. Ima bogatu podršku za linearnu algebru, generiranje nasumičnih brojeva, a svoju brzinu zahvaljuje činjenici da je njena izvedba napisana u programskom jeziku C.

[1] https://pytorch.org
[2] https://numpy.org
[3] https://opencv.org
[4] https://colab.research.google.com

4.3. OpenCV

OpenCV je moćna biblioteka za računalni vid i strojno učenje. Razvijena je s ciljem da računalni vid postane praktično korišten u stvarnim aplikacijama i pristupačan za širu publiku. Nudi širok spektar alata za različite zadatke računalnog vida, kao što su obrada slika, prepoznavanje uzoraka, objekata i lica, detekcija pokreta, kalibracija kamere i 3D rekonstrukcija te je vrlo integrabilan s drugim bibliotekama kao što su PyTorch i TensorFlow.

4.4. Matplotlib

Matplotlib[5] je sveobuhvatna biblioteka za stvaranje statičnih, animiranih i interaktivnih vizualizacija u Pythonu. Široko se koristi u znanosti o podacima, a njegovi ključni aspekti su bogate mogućnosti grafičkog prikaza. Ova biblioteka nudi širok raspon funkcija za crtanje, omogućujući korisnicima stvaranje raznih vrsta grafova, uključujući linijske grafove, raspršene grafove, stupčaste dijagrame, histograme, tortne dijagrame i druge, što uvelike olakšava vizualizaciju raznih podataka i rezultata. Također, jedna velika pogodnost Matplotliba jest odlična kompatibilnost s ostalim bibliotekama, kao što su već spomenuti PyTorch i NumPy.

[5] https://matplotlib.org

5. Eksperimenti

Eksperimente u ovom radu provedeni su u svrhu ispitivanja i uspoređivanja različitih metoda i pristupa treniranju i učenju dubokih konvolucijskih modela. Svi eksperimenti su provedeni na skupu Cityscapes na pola rezolucije (1024x512). Rezultate koje ćemo uspoređivati su rezultati dobiveni nadziranim učenjem, polunadziranim učenjem s 50% označenih podataka te polunadziranim učenjem s 25% označenih podataka.

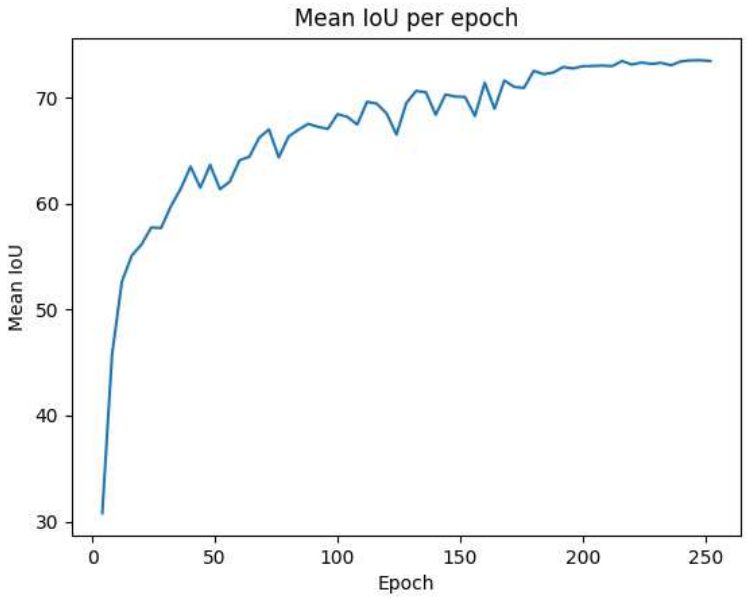
5.1. Hiperparametri

Kao što je već spomenuto, treniranje je obavljeno na pola rezolucije ulaznih slika, no na pola rezolucije smo postavili i nasumične izreske iz ulazne slike i njene maske (448x448). Treniranje je trajalo 250 epoha za nadzirano učenje, a za polunadzirano je model najprije istreniran na 50% označenih slika na pola epoha (125), dok je nakon generiranih oznaka ponovno istreniran na 250 epoha. Za slučaj polunadziranog učenja gdje je bilo samo 25% označenih podataka odlučili smo staviti svih 250 epoha zato što je mIoU konstantno rastao. Veličina grupa *(eng. batch size)* je bila postavljena na 14 slika. Stopa učenja postavljena je na 4×10^{-4}, dok je minimalna stopa učenja postavljena na 1×10^{-6}. Validacija se obavljala svake 4 epohe. Kao funkcija gubitka korištena je već spomenuta funkcija unakrsne entropije, dok je korišteni optimizacijski model *Adam*[6], dok je platforma za računanje bila platforma tvrtke Nvidie pod nazivom *Cuda*[7].

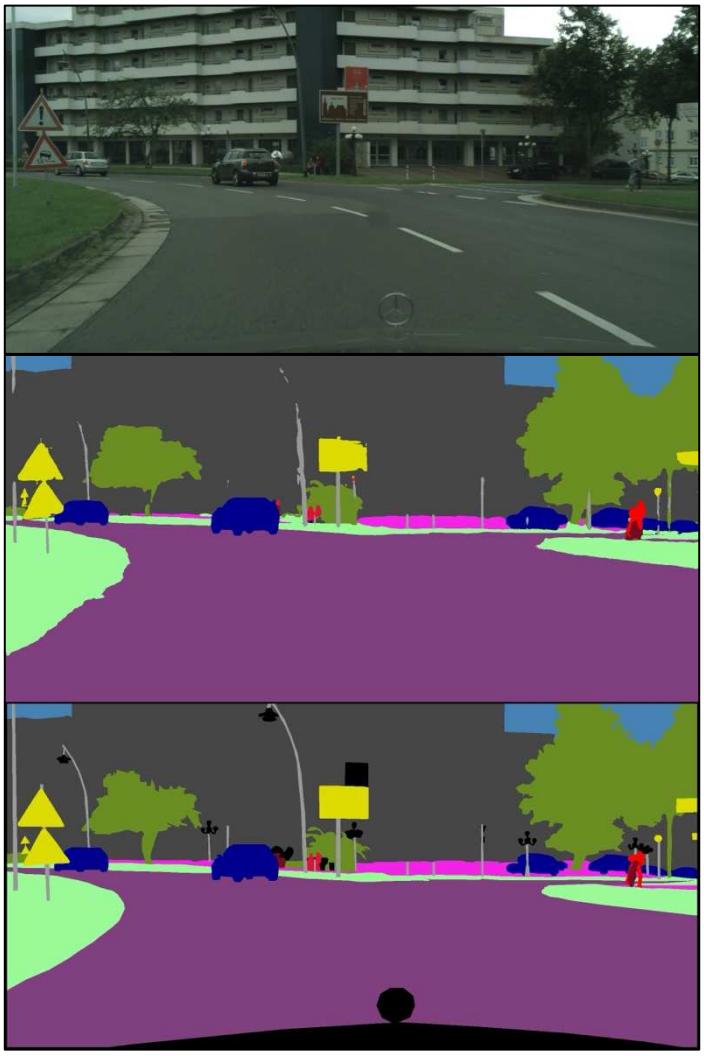
5.2. Rezultati nadziranog učenja

Nakon provedenih 250 epoha treniranja, naš model postiže mIoU metriku u vrijednosti 73.49%, što je izvrstan rezultat s obzirom na to da je model treniran na samo pola rezolucije. Treniranje obavljeno na punoj rezoluciji bi očekivano dalo i veće rezultate. Kretanje mIoU metrike kroz epohe za vrijeme treniranja prikazano je na *(Slika 5.2.1* - Mijenjanje mIoU vrijednosti kroz epohe.

[6] https://pytorch.org/docs/stable/generated/torch.optim.Adam.html
[7] https://developer.nvidia.com/cuda-toolkit



Slika 5.2.1 - Mijenjanje mIoU vrijednosti kroz epohe



Slika 5.2.2 - Predikcija maske slike iz skupa Cityscapes koristeći Swiftnet. Odozgo prema dolje: originalna slika, predviđena segmentacijska maska, stvarna segmentacijska maska

Ako malo temeljitije proučimo rezultate iz *Tablica 1*, možemo primijetiti da su točnosti za klase čiji su objekti obično površinom veći i jednostavnijeg oblika veće (npr. cesta, nebo), nego za one objekte čije su strukture kompleksnije i manje (npr. stup, ograda, semafor). To je rezultat činjenice da je veće i pravilnije objekte puno lakše „pogoditi“ dok bi nam za takvu preciznost manjih i složenijih objekata trebalo puno duže treniranje.

Tablica 1 - IoU vrijednost za svaku klasu skupa Cityscapes dobiveni na temelju nadziranog učenja

Klasa	IoU (%)
Cesta (*eng. Road*)	97.54
Pločnik (*eng. Sidewalk*)	81.83
Zgrada (*eng. Building*)	91.38
Zid (*eng. Wall*)	51.61
Ograda (*eng. Fence*)	55.43
Stup (*eng. Pole*)	60.46
Semafor (*eng. Trafficlight*)	65.63
Prometni znak (*eng. Trafficsign*)	75.51
Vegetacija (*eng. Vegetation*)	91.96
Prirodni teren (*eng. Terrain*)	63.79
Nebo (*eng. Sky*)	94.24
Osoba (*eng. Person*)	79.94
Vozač (*eng. Rider*)	55.87
Automobil (*eng. Car*)	93.81
Kamion (*eng. Truck*)	63.7
Autobus (*eng. Bus*)	82.17
Vlak (*eng. Train*)	68.05
Motocikl (*eng. Motorcycle*)	49.71
Bicikl (*eng. Bicycle*)	75.0

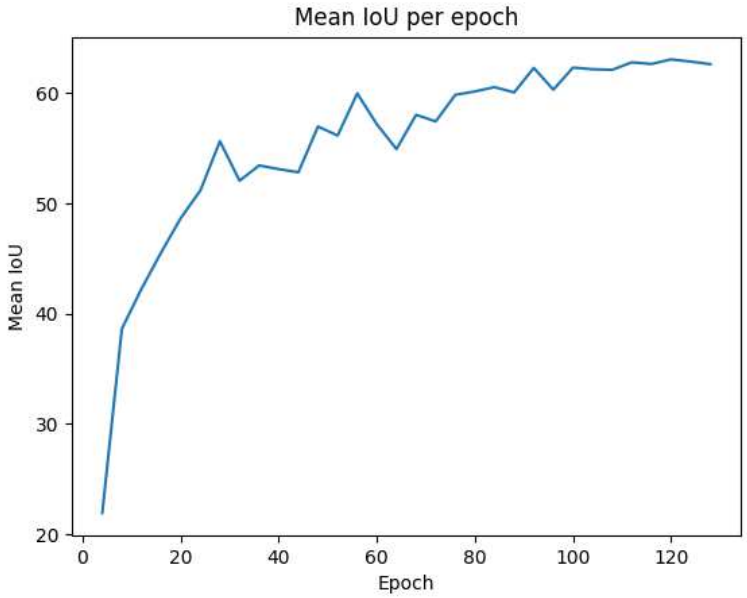
5.3. Rezultati polunadziranog učenja

Za potrebe ovog eksperimenta, uzeli smo određenih postotak označenih podataka iz skupa za treniranje te smo za ostatak tog skupa oznake ignorirali. Prvo smo model istrenirali samo na označenim podacima te smo ga evaluirali, generirali nove labele i zabilježili rezultate. Zatim smo uzeli taj predtrenirani model, te ga istrenirali na potpunom, sada označenom skupu podataka

5.3.1. 50% označenih podataka

Ovaj eksperiment smo proveli na 50% označenih podataka i na pola inicijalno određenih epoha za predtreniranje. Razlog zašto smo se odlučili za pola epoha je zato što je eksperiment proveden na Google Colabu, stoga smo imali ograničene resurse, no rezultati su se i dalje ispostavili dovoljno točni.

Nakon što je provedena prva faza treniranja, dobili smo mIoU u iznosu od 63.06%. Na *(Slika 5.3.1)* prikazan je tijek mijenjanja te metrike kroz treniranje, dok su *Tablica 2* prikazane mIoU vrijednosti za svaku klasu.



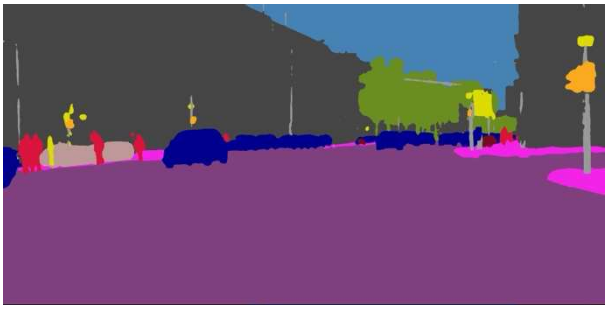
Slika 5.3.1 - Mijenjanje IoU kroz epohe prve faze treniranja na 50% označenog skupa

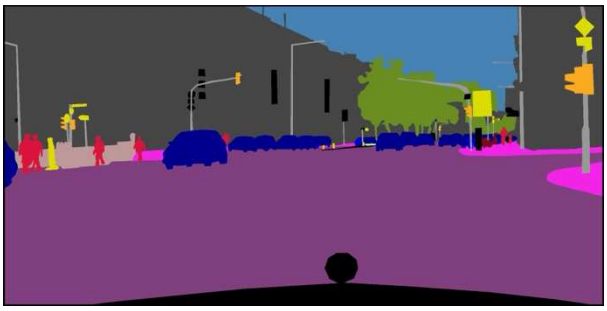
Tablica 2 - Vrijednosti IoU za svaku klasu nakon prve faze treniranja

Klasa	IoU (%)
Cesta (*eng. Road*)	96.65
Pločnik (*eng. Sidewalk*)	74.87
Zgrada (*eng. Building*)	86.95
Zid (*eng. Wall*)	42.59
Ograda (*eng. Fence*)	40.43
Stup (*eng. Pole*)	56.83
Semafor (*eng. Trafficlight*)	53.69
Prometni znak (*eng. Trafficsign*)	67.11
Vegetacija (*eng. Vegetation*)	89.96
Prirodni teren (*eng. Terrain*)	44.49
Nebo (*eng. Sky*)	90.5
Osoba (*eng. Person*)	77.75
Vozač (*eng. Rider*)	39.21
Automobil (*eng. Car*)	91.33
Kamion (*eng. Truck*)	57.99
Autobus (*eng. Bus*)	57.97
Vlak (*eng. Train*)	19.22
Motocikl (*eng. Motorcycle*)	41.18
Bicikl (*eng. Bicycle*)	69.41

Naime, iako se na prvu čini kako labele generirane sa samo 63.06% mIoU nisu dovoljno točne da bi se koristile za treniranje, one su ipak dovoljne sličnosti *(Slika 5.3.2)*.

Ulazna slika



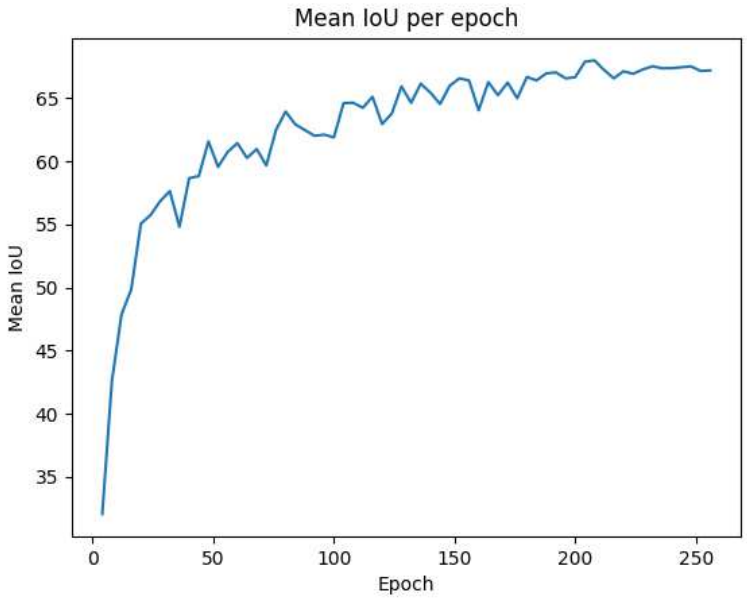


Predviđena segmentacijska maska

Točna segmentacijska maska

Slika 5.3.2 - Primjer segmentacijske maske
korištene za drugu fazu treniranja

Nakon obavljene i druge faze treniranja modela u ovom eksperimenta, rezultati su sljedeći:



Slika 5.3.3 - Kretanje mIoU vrijednosti u drugoj fazi treniranja na 50% označenog skupa

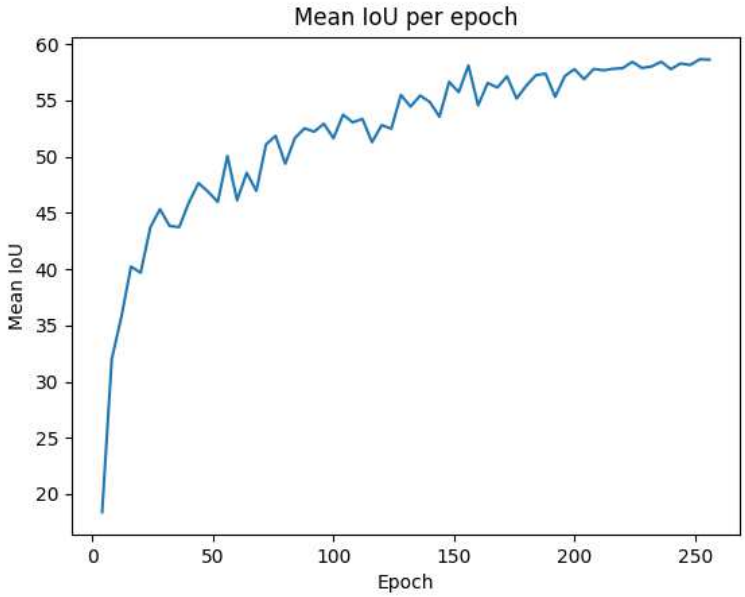
Konačna vrijednost mIoU nakon treniranja iznosi 67.98% *(Slika 5.3.3)*, što je samo 5.51% manje nego što je postignuto nadziranim učenjem. Također, u odnosu na predtrenirani model vidimo porast od čak 4.92%.

Tablica 3 - Vrijednosti IoU za svaku klasu nakon druge faze treniranja

Klasa	IoU (%)
Cesta (*eng. Road*)	96.48
Pločnik (*eng. Sidewalk*)	75.97
Zgrada (*eng. Building*)	89.96
Zid (*eng. Wall*)	48.43
Ograda (*eng. Fence*)	46.37
Stup (*eng. Pole*)	58.72
Semafor (*eng. Trafficlight*)	60.85
Prometni znak (*eng. Trafficsign*)	72.95
Vegetacija (*eng. Vegetation*)	91.64
Prirodni teren (*eng. Terrain*)	61.34
Nebo (*eng. Sky*)	92.98
Osoba (*eng. Person*)	78.17
Vozač (*eng. Rider*)	48.94
Automobil (*eng. Car*)	92.94
Kamion (*eng. Truck*)	52.36
Autobus (*eng. Bus*)	70.72
Vlak (*eng. Train*)	31.48
Motocikl (*eng. Motorcycle*)	47.29
Bicikl (*eng. Bicycle*)	73.95

5.3.2. 25% označenih podataka

Za izvedbu ovog eksperiment, uzeli smo samo 25% svih označenih podataka originalnog skupa i njih tretirali kao označene. Ovaj eksperiment nam je puno zanimljiviji od prethodnog, iz razloga što je ovdje čak duplo manje podataka označeno, što je u odnosu na cijeli skup relativno malo. Prvu fazu ovog eksperimenta odnosno predtreniranje modela na 25% označenih podataka provedeno je na 250 epoha kako bi postigli što veću inicijalnu točnost modela uzevši u obzir tako malu količinu podataka. Rezultati provedbe prve faze eksperimenta prikazani su *Slika 5.3.4* i *Tablica 4.*



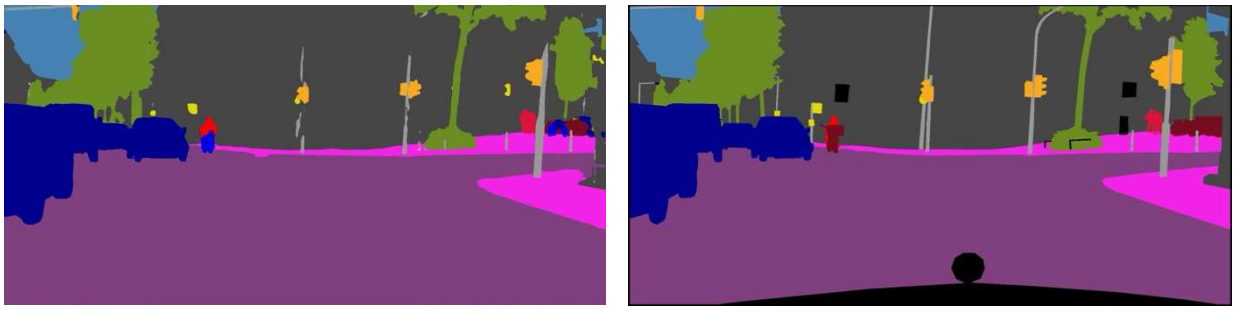
Slika 5.3.4 - Kretanje mIoU vrijednosti u prvoj fazi treniranja na 25% skupa

Na grafu prikazanom na *Slika 5.3.4* možemo vidjeti kako konvergencija učenja ne samo da je znatno sporija nego u ostalim eksperimentima, nego je i puno šumovitija, a sama vrijednost metrike mIoU dostiže vrijednost od 58.66%.

Tablica 4 - Vrijednosti IoU za svaku klasu nakon prve faze treniranja

Klasa	IoU (%)
Cesta (*eng. Road*)	94.53
Pločnik (*eng. Sidewalk*)	66.64
Zgrada (*eng. Building*)	83.64
Zid (*eng. Wall*)	34.05
Ograda (*eng. Fence*)	27.49
Stup (*eng. Pole*)	57.04
Semafor (*eng. Trafficlight*)	55.1
Prometni znak (*eng. Trafficsign*)	65.71
Vegetacija (*eng. Vegetation*)	89.37
Prirodni teren (*eng. Terrain*)	55.62
Nebo (*eng. Sky*)	87.5
Osoba (*eng. Person*)	73.65
Vozač (*eng. Rider*)	36.85
Automobil (*eng. Car*)	84.15
Kamion (*eng. Truck*)	35.83
Autobus (*eng. Bus*)	26.6
Vlak (*eng. Train*)	39.34
Motocikl (*eng. Motorcycle*)	34.64
Bicikl (*eng. Bicycle*)	66.85

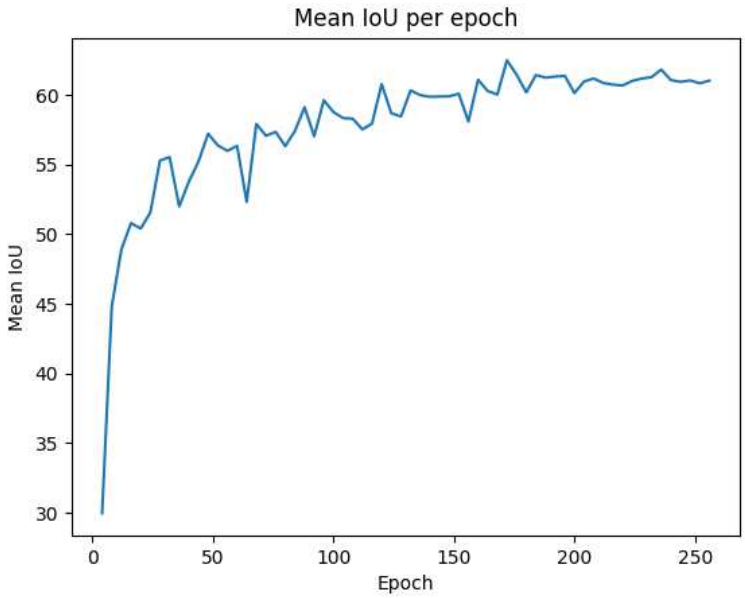
S druge strane, ako malo pomnije promotrimo *Tablica 4*, primijetit ćemo kako je točnost predikcija za već spomenute jednostavne i velike objekte relativno visoka. Iz toga možemo zaključiti da najveći problem u ovom pristupu predstavljaju složeni objekti kompleksnijih oblika ili vrlo precizno definirani objekti, čije su oznake generirane predtreniranim modelom bile neprecizne. Pogledajmo primjer jednog takvog slučaja prikazanog *(Slika 5.3.5)*.



Predviđena segmentacijska mapa Točna segmentacijska maska

Slika 5.3.5 - Prikaz gubitka preciznosti oznaka složenih i malih objekata

Nakon provedbe druge faze eksperimenta, zabilježili smo sljedeće rezultate:



Slika 5.3.6 - Kretanje mIoU vrijednosti u drugoj fazi treniranja na 25% skupa

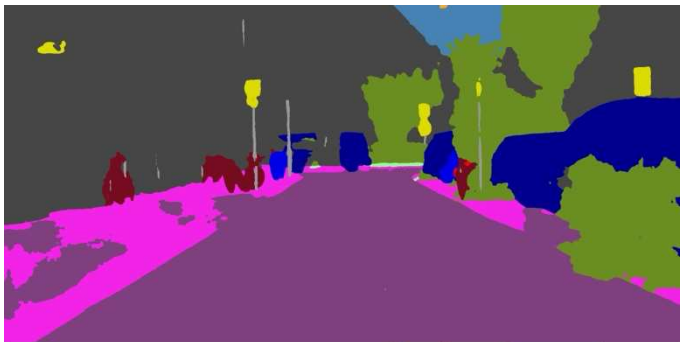
Kao što vidimo na grafu na *Slika 5.3.6*, konvergencija je idalje poprilično šumovita te ima velik broj naglih i strmih rastova i padova, no možemo vidjeti da u usporedbi s prvom fazom ovog eksperimenta puno brže konvergira s konačnim mIoU u vrijednosti od 62.48%.

Klasa	IoU (%)
Cesta (*eng. Road*)	94.46
Pločnik (*eng. Sidewalk*)	66.99
Zgrada (*eng. Building*)	87.66
Zid (*eng. Wall*)	33.32
Ograda (*eng. Fence*)	27.28
Stup (*eng. Pole*)	55.94
Semafor (*eng. Trafficlight*)	59.37
Prometni znak (*eng. Trafficsign*)	70.84
Vegetacija (*eng. Vegetation*)	90.99
Prirodni teren (*eng. Terrain*)	57.49
Nebo (*eng. Sky*)	91.79
Osoba (*eng. Person*)	76.0
Vozač (*eng. Rider*)	50.34
Automobil (*eng. Car*)	90.99
Kamion (*eng. Truck*)	24.23
Autobus (*eng. Bus*)	56.62
Vlak (*eng. Train*)	33.68
Motocikl (*eng. Motorcycle*)	46.84
Bicikl (*eng. Bicycle*)	72.21

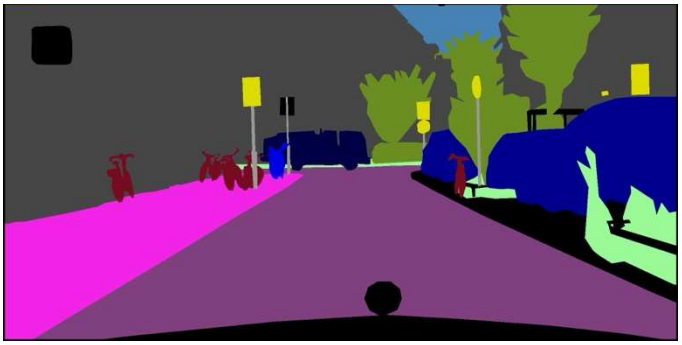
Tablica 5 - Vrijednosti IoU za svaku klasu nakon druge faze treniranja

Primjer predikcije ovog modela nakon treniranja na potpunom skupu za učenje prikazan je *(Slika 5.3.7)*.

Ulazna slika



Predviđena segmentacijska maska



Točna segmentacijska maska

Slika 5.3.7 - Primjer segmentacijske maske korištene za drugu fazu treniranja modela predtreniranog na 25% skupa

6. Zaključak

U ovom radu istražili smo polunadzirano učenje za zadatak semantičke segmentacije, gdje je većina dostupnih podataka neoznačena. Cilj je bio poboljšati generalizacijsku sposobnost modela koristeći ograničene resurse označenih slika.

Proučene su i implementirane različite arhitekture temeljene na konvolucijskim mrežama i semantičkoj segmentaciji, koje su zatim testirane na potpuno označenim podacima kako bi se uspostavila osnovna razina performansi. U scenariju polunadziranog učenja, koristili smo samo određeni postotak označenih slika, za što smo dobili iznenađujuće rezultate.

Validacija hiperparametara i vrednovanje generalizacijske izvedbe ukazali su na značajna poboljšanja u usporedbi s tradicionalnim metodama učenja. Postignuti rezultati pokazuju da polunadzirano učenje može učinkovito iskoristiti neoznačene podatke, što je ključno za primjene u stvarnim situacijama gdje je označavanje podataka skupo ili teško. Također, u kombinaciji s nekim drugim metodama kao što su primjena pažnje bi mogli dodatno poboljšati performanse ovog pristupa te bi ovaj pristup kroz par godina mogao znatnije konkurirati potpuno nadziranom učenju.

Ukupno gledano, ovaj rad doprinosi razumijevanju i primjeni polunadziranog učenja u području semantičke segmentacije, te nudi smjernice za daljnja istraživanja i primjene u drugim domenama računalnog vida.

Literatura

[1] Cityscapes skup podataka, https://www.cityscapes-dataset.com

[2] Couturier, Raphaël & Dionis, Etienne & Guérin, S. & Guyeux, Christophe & Sugny, Dominique. (2022). Characterization of a Driven Two-level Quantum System by Supervised Learning. 10.20944/preprints202212.0433.v1.

[3] Semi-Supervised Learning, Explained with Examples https://www.altexsoft.com/blog/semi-supervised-learning/

[4] Grubišić I, Oršić M, Šegvić S. Revisiting Consistency for Semi-Supervised Semantic Segmentation. Sensors. 2023; 23(2):940. https://doi.org/10.3390/s23020940

[5] Petra Bevandić, Marin Oršić, Ivan Grubišić, Josip Šarić i Siniša Šegvić, Nulta laboratorijska vježba iz Dubokog učenja https://www.zemris.fer.hr/~ssegvic/du/lab0.shtml

[6] Li, PhD, H. (2022). Numerical Differentiation and Integration. In: Numerical . Methods Using Java. Apress, Berkeley, CA https://doi.org/10.1007/978-1-4842-6797-4_6

[7] Ian Goodfellow, Yoshua Bengio and Aaron Courville. Deep Learning. MIT Press http://www.deeplearningbook.org

[8] Petra Bevandić, Marin Oršić, Ivan Grubišić, Josip Šarić i Siniša Šegvić, Prva laboratorijska vježba iz Dubokog učenja https://www.zemris.fer.hr/~ssegvic/du/lab1.shtml

[9] Precision, Recall & F1-Score, https://r170395.medium.com/precision-recall-f1-score

[10] Li, Z., Liu, F., Yang, W., Peng, S. and Zhou, J., 2021. A survey of convolutional neural networks: analysis, applications, and prospects. IEEE transactions on neural networks and learning systems, 33(12), pp.6999-7019.

[11] Naebi, Ahmad, and Zuren Feng. 2023. "The Performance of a Lip-Sync Imagery Model, New Combinations of Signals, a Supplemental Bond Graph Classifier, and Deep Formula Detection as an Extraction and Root Classifier for Electroencephalograms and Brain–Computer Interfaces" Applied Sciences 13, no. 21: 11787. https://doi.org/10.3390/app132111787

[12] Elyasi, N. & Moghadam, Mehdi. (2020). Tda in classification alongside with neural nets.

[13] Yani, Muhamad & Irawan, S, & Setianingsih, Casi. (2019). Application of Transfer Learning Using Convolutional Neural Network Method for Early Detection of Terry's Nail. Journal of Physics: Conference Series. 1201. 012052. 10.1088/1742-6596/1201/1/012052.

[14] Hao, S., Zhou, Y. and Guo, Y., 2020. A brief survey on semantic segmentation with deep learning. Neurocomputing, 406, pp.302-321.

[15] Ouassit, Youssef & Ardchir, Soufiane & Ghoumari, Mohammed & Azouazi, Mohamed. (2022). A Brief Survey on Weakly Supervised Semantic Segmentation. International Journal of Online and Biomedical Engineering (iJOE). 18. 83-113. 10.3991/ijoe.v18i10.31531.

[16] ResNet-18 https://pytorch.org/vision/models/resnet18.html

[17] Bonny, T., Al-Ali, A., Al-Ali, M. et al. Dental bitewing radiographs segmentation using deep learning-based convolutional neural network algorithms. Oral Radiol 40, 165–177 (2024). https://doi.org/10.1007/s11282-023-00717-3

[18] Orsic, M., Kreso, I., Bevandic, P. and Segvic, S., 2019. In defense of pre-trained imagenet architectures for real-time semantic segmentation of road-driving images. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 12607-12616).

[19] Ronny Restrepo, Intersect over Union: http://ronny.rest/tutorials/module/localization_001/iou/

Polunadzirano učenje semantičke segmentacije

Sažetak

Semantička segmentacija danas je vrlo unosno područje s mnogim primjenama, no nedostatak označenih slika je i dalje neriješen problem računalnog vida. U ovom radu isprobavamo potpuno nadzirani i polunadzirani pristup kako bismo ustanovili i ocijenili postignute točnosti. U radu su opisani osnovni koncepti dubokog učenja i metoda kojima smo se služili te korištena arhitektura. Korišteni skup podataka jest Cityscapes dok je korištena arhitektura SwiftNet temeljen na ResNet-18 arhitekturi enkodera. Programski kod je napisan u programskom jeziku Python koristeći biblioteke PyTorch, NumPy te OpenCV, dok su rezultati vizualizirani bibliotekom Matplotlib. Kod je izvršen djelomično na platformi Google Colab, a djelomično na udaljenom serveru ZEMRIS-a na FER-u.

Ključne riječi: Duboko učenje, računalni vid, nadzirano učenje, polunadzirano učenje, konvolucijske neuronske mreže, semantička segmentacija

Semi-supervised learning of semantic segmentation

Summary

Semantic segmentation is currently a highly lucrative field with numerous applications, yet the lack of labeled images remains an unresolved problem in computer vision. In this paper, we explore fully supervised and semi-supervised approaches to establish and evaluate the achieved accuracies. The paper describes the basic concepts of deep learning and the methods we employed, as well as the utilized architecture. The dataset used is Cityscapes, and the architecture used is SwiftNet, based on the ResNet-18 encoder architecture. The code is written in the Python programming language, utilizing libraries such as Python, NumPy and OpenCV, while the results are visualized using the Matplotlib library. The code execution was partially carried out on the Google Colab platform and partially on a remote server at ZEMRIS, FER.

Keywords: Deep learning, computer vision, supervised learning, semi-supervised learning, convolutional neural networks, semantic segmentation