SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 6987

Semantička segmentacija kolničkih trakova

Jelena Bratulić

Zagreb, srpanj 2020.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 13. ožujka 2020.

ZAVRŠNI ZADATAK br. 6987

Pristupnica:	**Jelena Bratulić (0036506909)**
Studij:	Računarstvo
Modul:	Računarska znanost
Mentor:	prof. dr. sc. Siniša Šegvić

Zadatak: **Semantička segmentacija kolničkih trakova**

Opis zadatka:

Semantička segmentacija prirodnih scena važan je zadatak računalnog vida s mnogim zanimljivim primjenama. U posljednje vrijeme vrlo zanimljive rezultate na tom zadatku postižu konvolucijski modeli s ljestvičastim naduzorkovanjem. Zbog velikog broja komercijalnih primjena, posebno je zanimljiva segmentacija kolničkih trakova iz perspektive vozača. U okviru rada, potrebno je odabrati okvir za automatsku diferencijaciju te upoznati biblioteke za rukovanje matricama i slikama. Proučiti i ukratko opisati postojeće pristupe za semantičku segmentaciju. Predložiti arhitekturu dubokog modela koja bi bila prikladna za slučaj segmentacije rubova prometnih trakova. Uhodati postupke učenja modela i validiranja hiperparametara. Primijeniti naučene modele na snimkama iz perspektive vozača. Prikazati i ocijeniti ostvarene rezultate.. Radu priložiti izvorni i izvršni kod razvijenih postupaka, ispitne slijedove i rezultate, uz potrebna objašnjenja i dokumentaciju. Citirati korištenu literaturu i navesti dobivenu pomoć.

Rok za predaju rada: 12. lipnja 2020.

Zahvaljujem mentoru prof. dr. sc. Siniši Šegviću na pomoći, savjetima, strpljenju i prenesom znanju tijekom pisanja ovog rada.

Veliko hvala Josipu Šariću na nesebičnoj pomoći i savjetima.

SADRŽAJ

1. Uvod

Računalni vid je grana umjetne inteligencije čiji je zadatak razumjeti i prepoznati sadržaj pojedine slike. Razumijevanje sadržaja slike moguće je postići procesom klasifikacije čime se slici pridjeljuje odgovarajuća oznaka ili kategorija. Nadalje, klasifikaciju je moguće promatrati i u kontekstu guste predikcije. Upravo je to zadatak semantičke segmentacije koja pojedinom pikselu pridjeljuje semantičku oznaku, odnosno labelu.

Semantička segmentacija ima široki raspon područja primjene, no u posljednje vrijeme sve je atraktivnije područje autonomne vožnje. U rješavanju tog zadatka vrlo zanimljive rezultate pokazuju duboki konvolucijski modeli, posebice modeli s ljestvičastim naduzorkovanjem.

Duboki konvolucijski modeli su modeli strojnog učenja koji se sastoje od više uzastopnih nelinearnih transformacija od kojih su većina konvolucije. Modeli strojnog učenja su prikladan formalizam za rješavanje semantičke segmentacije jer imaju mogućnost učenja iz podataka te modeliranja složenih funkcija.

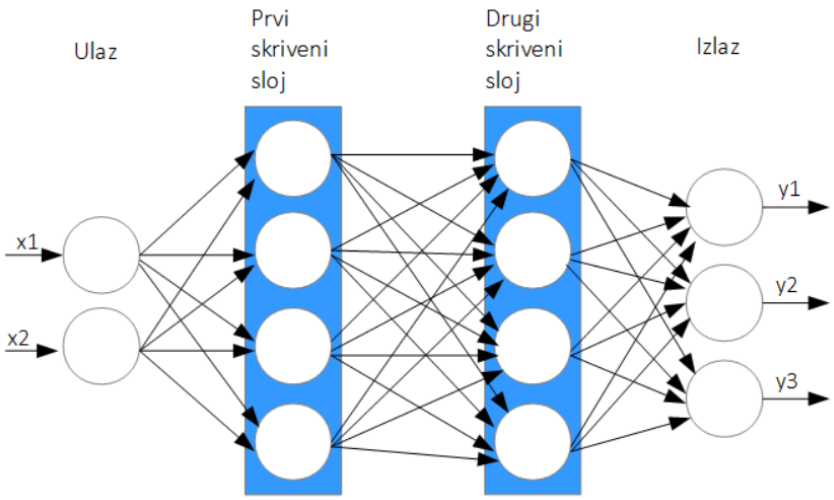
U ovom radu ukratko ću opisati osnove arhitekture i svojstva neuronske mreže te dubokih konvolucijskih modela. Predstavit ću model rezidualnih mreža te mreža s ljestvičastim naduzorkovanjem. Središnji dio ovog rada je problem semantičke segmentacije te njezine primjene u svakodnevnom životu, konkretnije u razvitku autonomne vožnje. Implementirat ću sustav za semantičku segmentaciju kolničkih trakova baziranu na konvolucijskom modelu s ljestvičastim naduzorkovanjem te ću isti model testirati na skupu podataka LLAMAS. Prikazat ću rezultate testiranja i osvrnuti se na implementirani model.

2. Duboko učenje

Duboko učenje (eng. *Deep learning*) je oblik strojnog učenja (eng. *Machine learning*), a strojno učenje je komponenta umjetne inteligencije (eng. *Artificial inteligence*). Duboki modeli su modeli strojnog učenja koji se sastoje od više uzastopnih nelinearnih transformacija. U sljedećem poglavlju osvrnut ću se na dva modela strojnog učenja - umjetne neuronske mreže i konvolucijske modele.

2.1. Umjetne neuronske mreže

Umjetna neuronska mreža (eng. *artificial neural network*) je skup međusobno povezanih jednostavnih procesnih elemenata čija se funkcionalnost temelji na biološkom neuronu. Pri tome je obradbena moć mreže pohranjena u snazi veze između pojedinih procesnih elemenata koji se nazivaju neuroni [2]. Neuronska mreža je model strojnog učenja na kojem je zasnovano duboko učenje.



Slika 2.1: Arhitekturu prikazane četveroslojne neuronske mreže možemo zapisati kao 2x4x4x3 što označava ulazni sloj od 2 neurona, dva skrivena sloja od 4 neurona i izlazni sloj od 3 neurona.

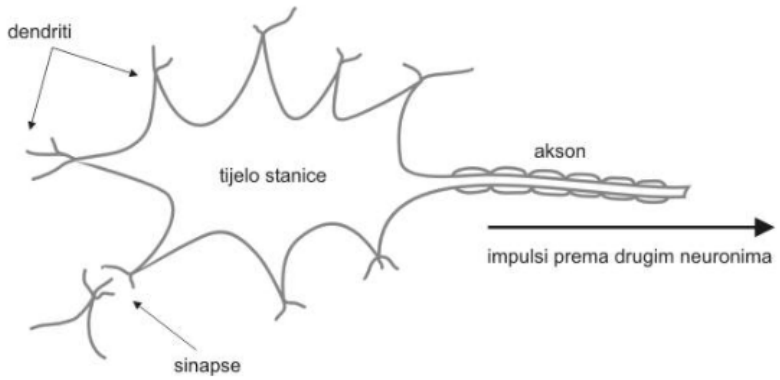
Neuronska mreža sastoji se od više slojeva pri čemu su ulazni i izlazni sloj vidljivi korisniku, a unutarnji nisu, stoga se zovu skriveni slojevi. Na slici 2.1 prikazana je arhitektura umjetne neuronske mreže. Izlazni neuroni izračunavaju se težinskim zbrajanjem neurona pojedinog sloja. Ako su svi neuroni sloja k povezani sa svim neuronima sloja $k + 1$, govorimo o potpuno povezanom sloju.

Mreža s 2.1 je unaprijedna neuronska mreža (eng. *feed-forward neural network*) što znači da unutar arhitekture mreže nema ciklusa između pojedinih slojeva. Kod unaprijedne mreže ulaz u pojedini neuron je izričito izlaz iz prethodnog sloja. Nadalje, uz unaprijedne mreže, postoje još i mreže s povratnom vezom (eng. *reccurent networks*), lateralno povezane mreže (eng. *ladder networks*) i hibridne mreže [2].

Važno svojstvo neuronske mreže je mogućnost učenja iz podataka što omogućava rješavanje problema računalnog vida poput klasifikacije i regresije podataka.

2.1.1. Umjetni neuron

Umjetna neuronska mreža sastoji se od velikog broja umjetnih neurona. Umjetni neuron prvi je put definiran 1943. godine kada su McCullan i Pitts predstavili model TLU-perceptrona (eng. *Threshold Logic Unit*). Definicija neurona proizašla je iz građe i načina rada biološkog neurona. Naime, živčani sustav živih bića se sastoji od velikog broja neurona koji su međusobno povezani kako bi prenosili podražaje i informacije, odnosno znanje.



Slika 2.2: Građa neurona s označenim dijelovima [2].

Biološki neuron, prikazan na 2.2, podražaje prima preko dendrita koji su sinapsama povezani s drugim neuronima te podražaje propagira aksiomom i završnim člancima dalje. Ulaz umjetnog neurona (slika 2.3) predstavljen je ulazima x_i koji su zapravo izlazi iz prethodnih neurona. Pojedini ulaz x_i množi se odgovarajućom težinom w_i.

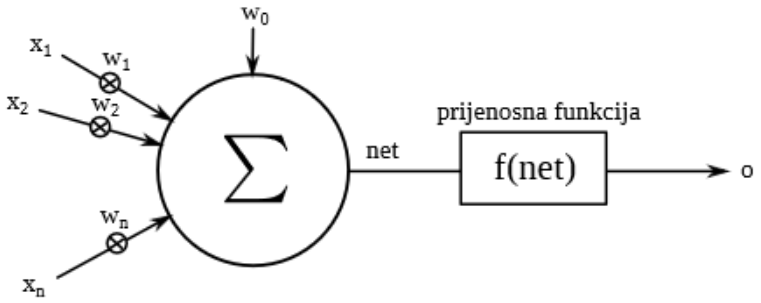
Težine (eng. *weights*) w_i su u početku nasumično generirane vrijednosti koje se procesom učenja korigiraju, no o tome ću detaljnije pisati u poglavlju Učenje modela. Nadalje, težinski pomnoženi ulazi se zbrajaju s pragom neurona (eng. *bias*) θ. Prag neurona se često definira i kao $w_0 = 1$ kako bi se sumacija ulaza i praga mogla lakše prikazati kao

$$net = \sum_{i=0}^{n} w_i x_i, \tag{2.1}$$

što je analogno

$$net = \sum_{i=1}^{n} w_i x_i + \theta \tag{2.2}$$

Vrijednost net naziva se akumulirana vrijednost neurona.



Slika 2.3: Umjetni neuron [2].

Konačno, izlaz iz neurona (o) dobivamo primjenjivanjem prijenose funkcije (eng. *activation function*) f na akumuliranu vrijednost neurona. Odabir prijenosne funkcije uvelike ovisi o krajnjem rezultatu te namjeni mreže. Izlaz neurona možemo dakle zapisati kao

$$o = f(\sum_{i=0}^{n} w_n x_x) \tag{2.3}$$

Ukoliko vrijednosti ulaza i težina prikažemo vektorski, izlaz neurona možemo prikazati kao

$$o = f(\sum w^T x) \tag{2.4}$$

2.1.2. Aktivacijske funkcije

Aktivacijska funkcija prilagođava se arhitekturi mreže te određuje ograničenja mreže. Korištenjem linearnih aktivacijskih funkcija uvelike ograničavamo mogućnosti mreže. Linearna kombinacija linearne kombinacije je ponovno linearna kombinacija čime nije moguće riješiti problem klasifikacije linearno-nezavisnih razreda. Linearno-nezavisni razred je razred čiju pripadnost nije moguće odrediti decizijskom granicom - pravcem oko kojeg se mijenja mišljenje klasifikacije uzorka [2].

Općenito vrijedi da su dobre prijenosne funkcije nelinearane i monotono rastuće. Neke od najčešćih prijenosnih funkcija su funkcija identiteta, funkcija skoka (eng. *step function*), sigmoida, tangens hiperbolni, zglobnica (ReLu, eng. Rectified Linear Unit) te propusna zglobnica (LReLU, eng. *Leaky Rectified Linear Unit*).

Identitet je linearna funkcija koja se definira kao

$$f(x) = x \tag{2.5}$$

Linearna funkcija ovakva oblika koristit će se samo u jednostavnijim modelima koji se lako uče i koji nisu jako duboki, odnosno nemaju puno slojeva. Neuron čija je prijenosna funkcija funkcija identiteta naziva se ADELINE neuron.

Funkcija skoka je funkcija koja za negativne vrijednosti ulaza poprima 0, a za pozitivne vrijednosti ulaza poprima 1, odnosno

$$f(x) = \begin{cases} 0, x < 0, \\ 1, x \geqslant 0 \end{cases} \tag{2.6}$$

Funkcija skoka može se koristiti kao prijenosna funkcija u mreži koja provodi binarnu klasifikaciju ulaza. Mogući izlaz mreže će primjenom prijenosne funkcije na akumuliranu vrijednost poprimiti vrijednost 1 ako je iznos akumulirane vrijednosti veći od nule, odnosno 0 u slučaju negativne vrijednosti. Vrijednost 0 na izlazu označavat će pripadnost prvoj klasi, a vrijednost 1 pripadnost drugoj klasi.

Funkcija skoka se koristi u još jednoj verziji gdje se izlaz preslikava u vrijednosti -1 i 1. Ako je ulazna vrijednost negativna, izlaz će poprimiti vrijednost -1, a ako je ulazna vrijednost pozitivna, izlaz će biti 1.

Neuron čija je prijenosna funkcija step funkciju naziva se TLU-perceptron i upravo je tim modelom definiran pojam umjetnog neurona 1943.godine.

Sigmoida je funkcija čija je karakteristika preslikavanje broja na raspon [0,1]. Funkciju definiramo kao

$$f(x) = \frac{1}{1 - e^{-x}} \tag{2.7}$$

Sigmoida velike pozitivne brojeve preslikava u 1, a velike negativne brojeve u 0. Sigmoidu možemo shvatiti i kao olabavljenu funkciju skoka jer promjena po y-osi nije trenutačna, već je postepena.

Sigmoida se u posljednje vrijeme rjeđe koristi u dubokim modelima, ali je zato i dalje zastupljena kod modela s povratnom vezom. Korištenjem sigmoide pojavljuju se dva problema.

Problem nestajućih gradijenata (eng. *vanishing gradient problem*) javlja se kada sigmoida poprima zasićenu vrijednost (0 ili 1). Tada je vrijednost gradijenta 0 ili vrlo blizu 0 čime se u potpunosti blokira prolaz podataka prilikom procesa učenja, odnosno unazadne propagacije (detaljnije opisano u poglavlju Učenje modela). Ovaj problem se javlja i ako su početne težine neurona postavljenje na jako velike brojeve jer to također dovodi do zasićenja neurona.

Nadalje, središnje vrijednosti sigmoide nisu centrirane oko nule što također remeti izračunavanje gradijenta i samog procesa učenja. Međutim problem centriranja moguće je riješiti primjenom funkcije tangensa hiperbolnog.

Tangens hiperbolni je funkcija koju možemo smatrati kao skaliranu sigmoidu na raspon [-1, 1] po ordinati. Također ju možemo promatrati i kao olabavljenu funkciju skoka koja ima razine -1 i 1. Tangens hiperbolni definiramo kao

$$f(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}} \tag{2.8}$$

Iako tangens hiperbolni rješava problem centriranja vrijednosti oko nule, i dalje je prisutan problem nestajućih gradijenata.

Zglobnica je funkcija koju definiramo kao

$$f(x) = max(0, x) \tag{2.9}$$

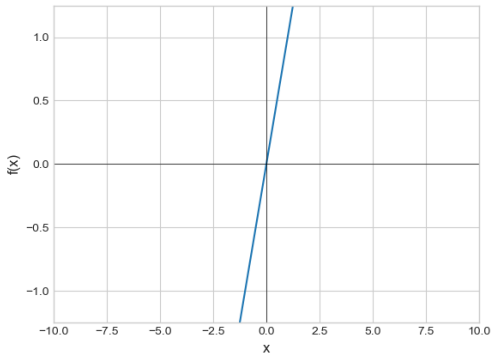
Ona propušta pozitivne vrijednosti, a sve negativne guši. Zglobnica je jednostavna operacija i brzo izračunljiva pa se u posljednje vrijeme najčešće koristi u dubokim modelima. Također, njezinom primjenom omogućujemo i brzu konvergenciju stohastičkog gradijenta što je iznimno važno prilikom procesa učenja.

Međutim, kod zglobnice se pojavljuje problem umirućih neurona koji nastaje kada izlaz poprimi vrijednost 0 za bilo koji primljeni ulaz. Umrli neuroni tada blokiraju tok podataka i nije moguće provesti korekciju prilikom unazadnog prolaska.

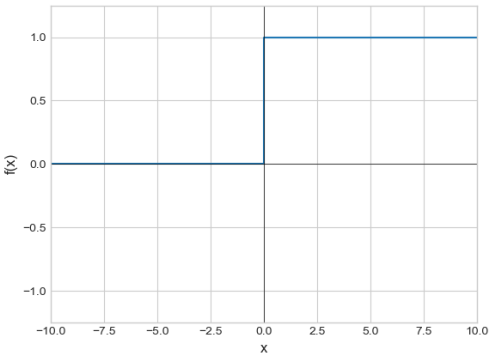
Rješenje, odnosno ublaživanje tog problema nudi funkcija propusne zglobnice koju definiramo kao

$$f(x) = \begin{cases} x, n \geqslant 0, \\ \alpha x, x < 0 \end{cases} \tag{2.10}$$

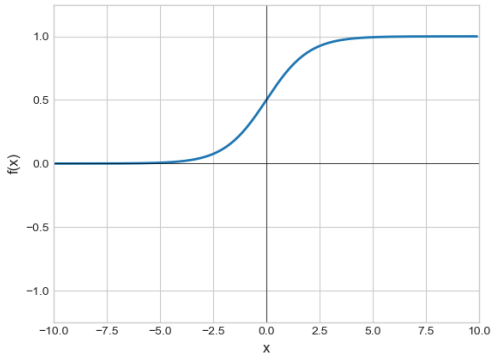
Propusna zglobnica ne guši u potpunosti negativne vrijednosti već ih propušta pomnožene s konstantom izrazito malog iznosa, primjerice 0.01 (prikazano na 2.4f).



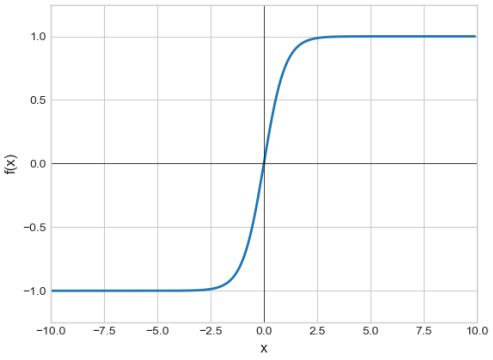
(a) Funkcija identiteta.



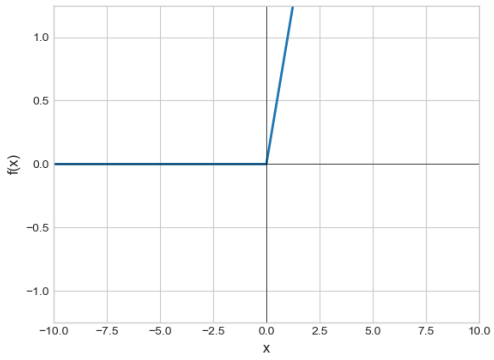
(b) Funkcija skoka.



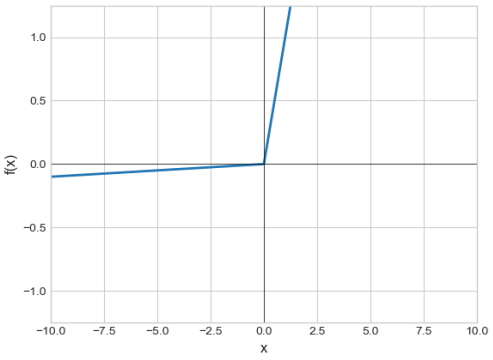
(c) Sigmoida.



(d) Tangens hiperbolni.



(e) Funkcija zglobnica.



(f) Funkcija propusne zglobnice za $\alpha = 0.01$.

Slika 2.4: Prijenosne funkcije prikazane na [-10, 10].

2.2. Učenje modela

Najvažnije svojstvo neuronskih mreža jest mogućnost učenja iz podataka. Razmatrat ću učenje nadziranih modela. Kod nadziranog učenja (eng. *supervised learning*) pretpostavlja se postojanje ulaznih i izlaznih (stvarnih vrijednosti) podataka.

Prilikom procesa učenja nastoji se prilagoditi težine pojedinog sloja kako bi u sljedećem prolazu kroz slojeve izlaz iz mreže, odnosno predikcija bila što sličnija očekivanoj vrijednosti. U procesu učenja, prilagođavanje težina provodi se optimiranjem funkcije gubitka na skupu podataka za učenje. U nastavku ću opisati često korištene metode izračuna gubitka i gradijentnog spusta te ću ukratko opisati sam proces učenja modela.

2.2.1. Funkcija gubitka

Gubitak je mjera pogreške određivanja predikcije, odnosno on bilježi koliko je odstupanje između izlaza iz mreže i očekivane vrijednosti. Gubitak će biti veći što je odstupanje veće. Kako bi se izračunao gubitak, koriste se različite funkcije gubitka ovisno o tome provodi li mreža klasifikaciju ili regresiju.

Srednja kvadratna pogreška (MSE, eng. *Mean Square Error*) je najčešća funkcija gubitka koja se koristi kod problema regresije. Srednju kvadratnu pogrešku možemo definirati kao

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2, \tag{2.11}$$

pri čemu je n umnožak uzoraka i izlaza modela, y_i stvarna vrijednost izlaza,a $\hat{y}_i$ procjena izlaza modela.

Unakrsna entropija je funkcija gubitka koja je najčešće korištena prilikom klasifikacije podataka. Unakrsnu entropiju moguće je koristiti i za binarnu i za višeklasnu klasifikaciju, a dobre rezultate pokazuje i kod neravnomjerne zastupljenosti klasa pri čemu se dodatno koriste težine koje predstavljaju udio zastupljenosti klasa.

Samu operaciju unakrsne entropije možemo definirati kao

$$E(p, q) = -\sum_{x} p(x) \log q(x), \tag{2.12}$$

pri čemu su funkcije p i q vjerojatnosne funkcije.

Ako promatramo slučaj binarne klasifikacije pri čemu je izlazna aktivacijska funkcija logistička funkcija poput sigmoide, binarnu unakrsnu entropiju, uz oznake uvedene kod MSE, definiramo kao

$$E = \frac{-1}{n} \sum_{i=1}^{n} y_i \log \hat{y}_i + (1 - y_i) \log (1 - \hat{y}_i) \tag{2.13}$$

Dodatno, y_i ovdje označava stvarnu oznaku klase koja odgovara ulaznom podatku, a $\hat{y}_i$ predstavlja procjenu izlaza za redni broj i.

Nadalje, unakrsnu entropiju je moguće svesti i na negativni logaritam izglednosti (eng. *negative log likelihood*) [5]. Ako izlazna aktivacijska funkcija određuje izlaz takav da je on razdioba vjerojatnosti razreda te ako stvarna razdioba izlaza za jednu klasu poprima vrijednost 1, a za ostale 0, tada unakrsnu entropiju možemo definirati kao

$$E = \frac{1}{n} \sum_{i} \log f(x_i)[y_i], \tag{2.14}$$

što odgovara izrazu negativnog logaritma izglednosti. Vrijednost x_i odgovara i-tom podatku za učenje, a vrijednost y_i indeksu točne klase tog podatka.

2.2.2. Propagacija pogreške unatrag

Propagacija pogreške unatrag (eng. *Error backpropagation*) je postupak učenja neuronskih mreža koji se temelji na učinkovitom izračunu svih parcijalnih derivacija u njihovoj primjeni na određivanju iznosa kojim korigiramo svaku od težina [2].

Ukoliko pretpostavimo izlaznu aktivacijsku funkciju funkciju sigmoidu te da se učenje provodi pojedinačno, sam algoritam možemo predočiti pseudokodom prikazanim na 1.

Dodatno još valja napomenuti kako *Downstream* neuroni označavaju sve one neurone sljedećeg sloja koje trenutni neuron pobuđuje. Algoritam je jednostavan i brz jer koristi jednostavne matematičke funkcije za izračun, no valja biti oprezan prilikom odabira prijenosne funkcije. Ukoliko se kao prijenosnu funkciju koristi funkciju koja vrlo lako poprima vrijednosti oko nule, može doći do problema umirućih neurona koji sam već opisala u jednom od prethodnih poglavlja.

Algoritam 1: Propagacija pogreške unazad za klasične neuronske modele sa sigmoidnom aktivacijom [2].

sve težine postavi na slučajne male vrijednosti

dok *nije zadovoljen uvjet zaustavljanja* **čini**

  za svaki *uzorak* $s : (x_{s,1}, ..., x_{s,n}) \leftarrow (t_{s,1}, ..., t_{s,n})$ **čini**

    postavi podatak $(x_{s,1}, ..., x_{s,2})$ na ulaz mreže

    izračunaj izlaz svih neurona u svim slojevima pri čemu je izlaz zadnjeg sloja definiran kao $(o_{s,1}, ..., o_{s,n})$

    odredi gradijent gubitka neurona izlaznog sloja:
    $\delta_i^K = o_{s,i} \cdot (1 - o_{s,i}) \cdot (t_{s,i} - o - s, i)$

    vraćaj se sloj po sloj unatrag i za svaki neuron izračunaj gradijent gubitka, za i-ti neuron u k-tom sloju, gradijent gubitka računaj kao:
    $\delta_i^{(k)} = y_i^{(k)} \cdot (1 - y_i^{(k)}) \cdot \sum_{d \ell Downstream} w_{i,d} \cdot \delta_d^{(k+1)}$

    napravi korekciju svih težina w i pragova b kao
    $w_{i,j}^{(k)} \leftarrow w_{i,j}^{(k)} + \eta \cdot y_i^{(k)} \cdot \delta_j^{(k+1)}$ i $b_j^{(k)} \leftarrow b_j^{(k)} + \eta \cdot \delta_j^{(k+1)}$

  kraj

kraj

2.2.3. Gradijentni spust

Budući da želimo minimizirati gubitak, jer manji gubitak znači manju grešku predikcije naspram stvarne vrijednosti, težine slojeva moramo podesiti tako da se gubitak smanjuje. Kako bismo ostvarili pomicanje vrijednosti gubitka u smjeru minimuma, koristimo metodu gradijentnog spusta prilikom učenja modela.

Dakle, ukoliko $f(x)$ predstavlja funkciju gubitka koju želimo minimizirati, tada $\nabla f(x)$ označava pripadni gradijent toj funkciji. Aproksimiramo li funkciju $f(x)$ Taylorovim razvojem prvog reda, slijedi :

$$f(x + \Delta x) \approx f(x) + \frac{df(x)}{dx}\Delta x = f(x) + \nabla f(x)^T \Delta x \tag{2.15}$$

Uvrstimo li u aproksimaciju pomak u smjeru negativnog gradijenta, dobivamo :

$$\Delta x = -\eta \nabla f(x) \tag{2.16}$$

Ako vrijedi Taylorova aproksimacija, tada vrijednost funkcije $f(x)$ opada, a upravo to i želimo postići, odnosno minimizirati gubitak. Varijabla η naziva se stopa učenja (eng. *learning rate*) i ona regulira u kojoj mjeri će se težina neurona ažurirati. Stopa učenja je obično male, ali pozitivne vrijednosti. Ako je stopa učenja prevelika, učenje

će divergirati, no ako je premalena proces učenja će sporo napredovati. Uobičajena vrijednost kreće se u rasponu od 0.001 do 0.5.

Budući da je svaki neuron određen težinom w_i i pragom w_0 ili b_i, u procesu učenja upravo ćemo težine i pragove neurona nastojati optimizirati. Uzimajući u obzir prethodno uvedeni izraz za promjenu, možemo definirati izraz korekcije za težine, odnosno pragove.

$$w_i \leftarrow w_i - \eta \frac{\partial f}{\partial w_i} \tag{2.17}$$

$$b_i \leftarrow b_i - \eta \frac{\partial f}{\partial b_i} \tag{2.18}$$

Međutim, kako bi se ubrzao gradijentni spust i time ubrzalo računanje prethodnog izraza, često se koristi stohastički gradijentni spust (SGD, eng. *Stochastic Gradient Descent*). Dodatno je uveden i ADAM koji optimizira gradijentni spust te pokazuje bolja svojstva od SGD-a.

Stohastički gradijentni spust računa gradijentni spust samo manje grupe ulaznih podataka. Time se ubrzava izračunavanje pogreške, a također se i izbjegava problem lokalnih minimuma prema kojem gradijent nastoji minimizirati funkciju gubitka prema njegovom lokalnom, a ne globalnom minimumu.

Adam (eng. Adaptive Moment Estimation) je jedan od mnogobrojnih optimizatora koji se koriste prilikom izračunavanja gradijentnog spusta [13]. Adam možemo definirati kao

$$m = \beta_1 m + (1 - \beta_1) dx, \tag{2.19}$$

$$v = \beta_2 v + (1 - \beta_2) dx^2, \tag{2.20}$$

pri čemu m predstavlja procjenu prvog momenta koja je srednja vrijednost, a v predstavlja procjenu drugog momenta, odnosno necentriranu varijancu koju dobivamo ako ne oduzimamo srednju vrijednost prilikom izračuna. Korekciju težina neurona sada možemo opisati izrazom

$$w_i \leftarrow w_i - \frac{\eta}{\sqrt{v} + \epsilon} m \tag{2.21}$$

Parametri β_1 i β_2 su postavljeni u vrijednosti vrlo blizu 1 kako bi se dobilo najbolje rezultate. Preporučeno je koristiti sljedeće vrijednosti parametara: $\beta_1 = 0.9, \beta_2 = 0.99, \epsilon = 10^{-8}$.

2.2.4. Metode regularizacije

Procesom učenja modela nastoji se minimizirati gubitak podešavanjem težina pojedinog neurona. Već sam opisala kako se za taj postupak koristi funkcija gubitka i gradijenti spust koji osigurava podešavanje težina tako da se funkcija gubitka kreće prema njezinom minimumu. Međutim, često se koriste dodatne metode regularizacije kako bi se osigurali od moguće pojave prenaučenosti modela.

L1 i L2 regularizacije su metode regularizacije koje se koriste kako bi se izbjegla pojava prenaučenosti modela. Metode možemo definirati kao

$$L1 = \sum_{i=0}^{n} |w_i|, \tag{2.22}$$

$$L2 = \sum_{i=0}^{n} w_i^2 \tag{2.23}$$

Primjenom neke od metoda regularizacije, funkcija gubitka postat će

$$E_{regularizirano} = E + \lambda L, \tag{2.24}$$

pri čemu $E_{regularizirano}$ označava funkciju gubitka uz korištenje metode regularizacije, E funkciju gubitka, λ parametar utjecaja regularizacije te L određenu vrstu norme regularizacije.

Normalizacija po grupi (eng. *batch norm*) je metoda regularizacije koja se temelji na normalizaciji izlaza prije primjene nelinearne aktivacijske funkcije. Dakle, distribucija izlaza transformira se u distribuciju sa srednjom vrijednošću 0 i varijancom 1.

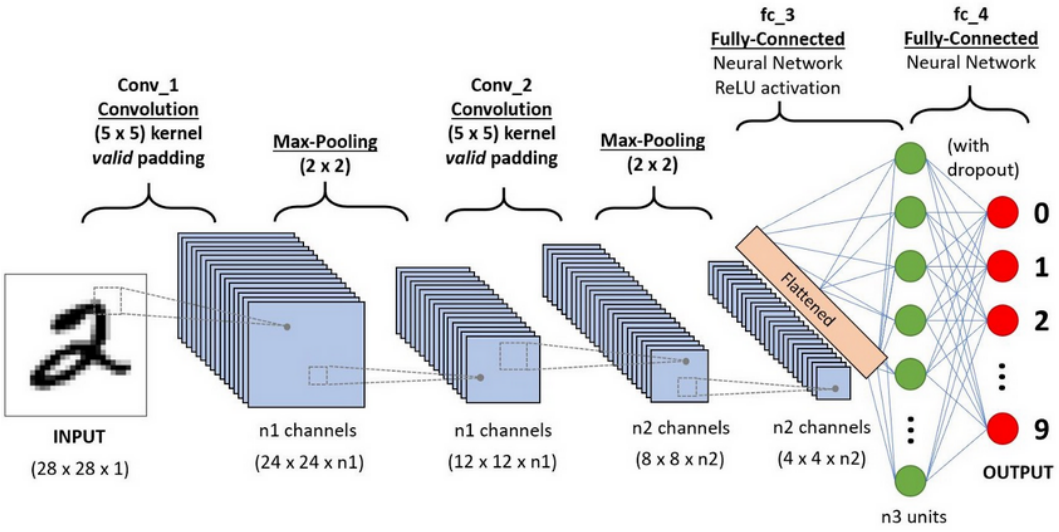
Korištenjem normalizacije grupa rješava se problem koji se pojavljuje u dubljim modelima kada se u svakom sloju mijenja distribucija podataka ovisno o postavljenim hiperparametrima sloja. Promjenom distribucije, usporava se sam proces učenja što nije nimalo poželjna pojava.

Preduvjet korištenja normalizacije grupa jest treniranje modela na manjim grupama (eng. *mini-batches*). Model može biti treniran na tri načina - pojedinačno (eng. *online*), po mini grupama ili po grupama (eng. *batches*). Preporučuje se i najčešće se koristi grupno treniranje budući da se time ubrzava proces učenja modela.

2.3. Duboki konvolucijski modeli

Skriveni slojevi neuronskih mreža u prošlim poglavljima bili su potpuno povezani slojevi. Takve mreže karakterizira veliki broj parametara mreže zbog velikog broja veza između slojeva. Potpuno povezani modeli prilikom učenja na podacima velike dimenzije, npr. slika visoke rezolucije, mogu dovesti do problema sporog učenja ili prenaučenosti (eng. *overfitting*) pri čemu se gubi svojstvo generalnosti modela.

Kao rješenje na spomenute probleme, pojavili su se konvolucijski modeli koje definiramo kao neuronske mreže s barem jednim konvolucijskim slojem. Konvolucijski modeli odlikuju se izrazito manjim brojem parametara u odnosu na potpuno povezane modele.



Slika 2.5: Primjer arhitekture konvolucijske mreže za klasifikaciju rukom pisanih znamenki. Na ulaz mreže dovodi se slika veličine 28x28 piksela koja se provlači kroz skrivene konvolucijske slojeve, slojeve sažimanja te potpuno povezani sloj. Izlazni sloj mreže sastoji se od 10 neurona koji odgovaraju 10 klasa znamenke. Izlazni neuroni su jednojedinično kodirani što znači da svaki neuron bilježi mjeru podudaranja ulazne slike s tom klasom [19].

Uobičajeni model konvolucijske mreže sastoji se od 3 dijela - konvolucijskih slojeva, slojeva sažimanja i potpuno povezanih slojeva. Primjer jednog konvolucijskog modela za klasifikaciju je prikazan na 2.5. Međutim, moderni konvolucijski modeli za klasifikaciju nemaju izravnavanje (eng. *flatten*), već koriste globalno sažimanje. Potpuno povezane slojeve sam već opisala u prethodnim poglavljima pa ću u nastavku detaljnije opisati konvolucijski sloj i sloj sažimanja.

No, prije toga moram uvesti pojam mape značajki (eng. *features map*) koja pred-

stavlja matrični zapis podataka oblika $H \times W \times C$ pri čemu su H i W prostorne dimenzije mape koje su određene njezinom širinom i visom, a C predstavlja semantičku dimenziju, odnosno dubinu. Ulaz i izlaz svakog konvolucijskog sloja jest upravo mapa značajki koja se mijenja ovisno o postavljenim hiperparametrima sloja.

Uzmemo li primjer ulazne slike s 2.5, dimenzije mape značajki koje bi odgovarale toj slici bile bi $28 \times 28 \times 1$ pri čemu 28×28 označava 28 piksela po širini te 28 piksela po dužini, a 1 označava 1 kanal, odnosno da se radi o crno-bijeloj slici u ovom slučaju.

2.3.1. Konvolucijski sloj

Konvolucijski sloj je najvažniji sloj konvolucijskog modela. Konvolucija je matematička operacija koja se provodi nad dvije funkcije f i g stvarajući treću funkciju koja izražava kako oblik jedne modificira drugu. Konvolucija je definirana izrazom

$$(f * g)(t) = \int_{-\infty}^{+\infty} f(\tau)g(t-\tau)d\tau \tag{2.25}$$

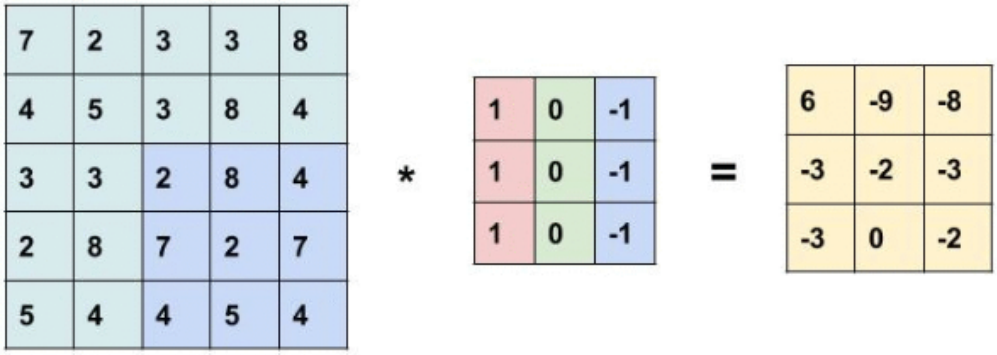
Međutim, konvolucijski sloj provodi diskretni oblik operacije konvolucije koji definiramo kao

$$I * K(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n), \tag{2.26}$$

pri čemu I označava sliku ili ulaznu mapu značajki, a K jezgru (eng. *kernel*) konvolucijskog sloja. Samu operaciju možemo lakše objasniti referencirajući se na sliku 2.6. Naime, izlaz konvolucijskog sloja je ponovno mapa značajki koja nastaje tako da se posmični prozor (tamnoplava 3×3 matrica unutar 5×5 matrice) kreće po vrijednostima ulaza te provodi operaciju množenja jezgre i određenog dijela ulazne mape značajki.

Potrebno je proći sve kombinacije pomaka po ulazu te za svaku kombinaciju pomnožiti vrijednosti ulaza (plava matrica) i jezgre (RGB matrica) te zbrojiti i zbrojenu vrijednost upisati na pripadno mjesto u izlaznoj mapi značajki (žuta tablica, pozicija x_{33}). Možemo primijetiti kako se ovom operacijom smanjuje prostorna dimenzija mape značajki, odnosno smanjuje se širina i visina početne mape značajki (ulazna je određena širinom i visinom slike).

Jezgra konvolucijskog sloja je male dimenzije i najčešće neparne (3, 5 ili 7), a određena je četirima parametrima - dubinama (eng. *depth*) ulazne i izlazne mape značajki, korakom (eng. *stride*) i nadopunjavanjem nulama (eng. *zero-padding*). Dubina jezgre definira semantičku dimenziju izlazne mape značajki. Korak jezgre definira pomak pri sljedećem pomaku posmičnog prozora po ulaznoj mapi značajki.



Slika 2.6: Primjer konvolucije nad matricom dimenzije 5x5 filtrom 3x3. [4]

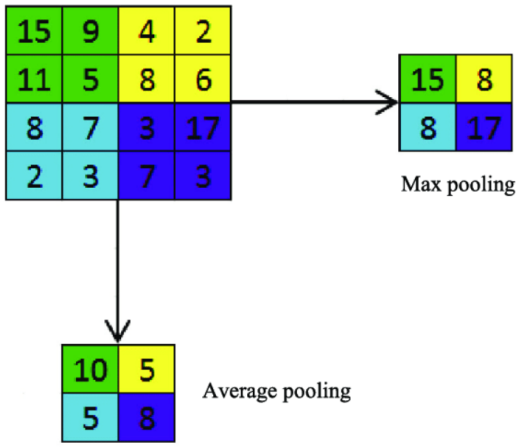
Veličina koraka utječe na izlaznu prostornu dimenziju - što je korak veći, to je izlazna prostorna dimenzija manja. Prostorna dimenzija se dodatno smanjuje i samom operacijom konvolucije. Kako bismo zadržali izvornu prostornu dimenziju mape značajki, moguće je provesti nadopunjavanje nulama koje se regulira posljednjim hiperparametrom. Veličina postavljenog parametra govori za koliko će se proširiti ulazna mapa značajki. Novoproširene značajke imat će vrijednost 0.

2.3.2. Sloj sažimanja

Mapi značajki se dodatno smanjuju dimenzije prolaskom kroz sloj sažimanja (eng. *Pooling*). Smanjenjem dimenzija smanjuje se broj parametara mreže, ubrzava računanje, ali i izbjegava problem prenaučenosti modela.

Smanjenje se provodi postupkom koji je sličan konvoluciji. Svaki sloj sažimanja ima definiranu vrijednost veličine jezgre koja se slijedno kreće po ulaznoj mapi značajki i primjenjuje jedan od kriterija izračuna nove vrijednosti mape značajki te konačno zapisuje novu vrijednost u izlaznu mapu značajki. Smanjenje se provodi analogno po svim dubinama nezavisno jednoj od druge te se broj dubina mape značajki ne mijenja.

Sloj sažimanja izračun novih vrijednosti može računati na temelju više kriterija, no najčešće zastupljen kriterij je odabir maksimalne vrijednosti (eng. *Max pooling*). Još jedan, ali manje zastupljen kriterij je odabir srednje vrijednosti (eng. *Average pooling*). Usporedni prikaz razlika oba sloja sažimanja prikazan je na slici 2.7.



Slika 2.7: Rezultat prolaska jezgre veličine 2×2 sloja sažimanja preko mape značajki dimenzije $4 \times 4 \times 1$ po kriteriju najvećeg elementa (desno) i srednje vrijednosti (dolje) [4].

2.3.3. Rezidualni modeli mreža

Razvitkom arhitekture neuronske mreže počeli su se graditi sve dublji i dublji modeli vjerujući kako će uspješnost modela biti bolja. Međutim, pokazalo se kako uspješnost mreža raste do određenog broja slojeva te tada počinje padati.

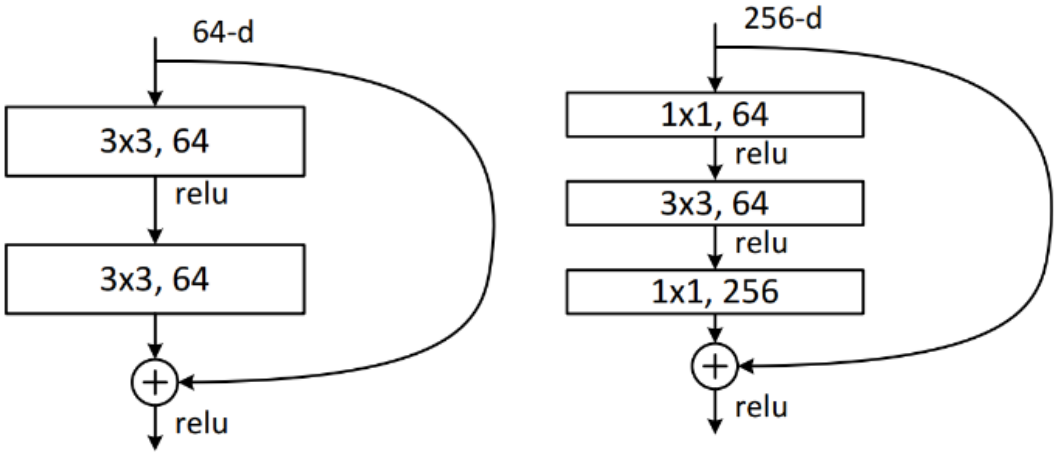
Kako bi se izbjegnuo navedeni problem, konvolucijski slojevi su nadograđeni dodavanjem rezidualnih veza (eng. *skip-connections*). Rezidualna veza spaja, odnosno pribraja ulaz prvog konvolucijskog sloja izlazu posljednjeg konvolucijskog sloja (prije prolaska kod prijenosnu funkciju) u bloku. Rezidualna veza je zapravo funkcija identiteta što znači da ne provodi nikakvu transformaciju nad primljenim podacima. Funkcija identiteta opisana je u Funkcija identiteta.

Ukoliko zbrajanje nije moguće zbog nepodudaranja u dimenzijama mape značajki, dodatno se provodi povećanje broja značajki pomoću konvolucije s veličinom jezgre 1x1 čiji izlazni korak se po potrebi postavlja na 2.

Na slici 2.8a prikazana je osnovna rezidualna jedinica (eng. *baseline unit*) kakva je prethodno opisana. Takva jedinica koristi se kod umjereno dubokih modela koji imaju do 30 slojeva, primjerice arhitektura ResNet-18 pri čemu 18 označava broj slojeva, dakle 18.

Međutim, kod izrazito dubokih modela moguća je pojava prevelikog broja parametara čime se javlja i mogućnost prenaučenosti modela. Kako bi se izbjegli ti problemi, uvedena je i rezidualna jedinica s uskim grlom koji u svakom bloku smanjuje broj parametara smanjenjem mapa značajki.

Primjer takve jedinice prikazan je na 2.8b. Prva konvolucija veličine jezgre 1 koristi se upravo za smanjenje mape značajki, dok se druga konvolucija veličine jezgre 1



(a) Osnovna rezidualna jedinica.

(b) Rezidualna jedinica s uskim grlom (eng. *bottleneck*).

Slika 2.8: *Baseline* i *Bottleneck* modeli rezidualne jedinice. Uvođenjem jedinice s uskim grlom reguliran je problem povećanja broja parametara, kao i prenaučenosti modela [8].

koristi za proširivanje mapa značajki na isti broj kao pri ulazu u blok [7].

Pokazalo se kako rezidualni modeli postižu bolju uspješnost od ostalih modela upravo zbog rezidualnih blokova koji omogućuju lakšu propaganciju gradijenta prilikom unazadnog prolaza. Danas se arhitektura ResNet koristi za rješavanje različitih problema računalnog vida.

2.3.4. Modeli mreža s ljestvičastim naduzorkovanjem

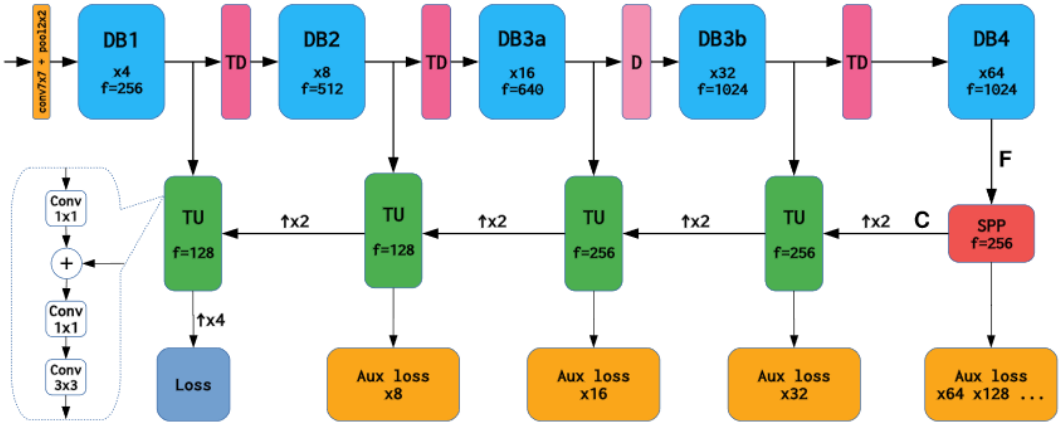
Iako su prethodno predstavljeni rezidualni modeli u posljednje vrijeme često korišteni, problem koji nije njihovom arhitekturom riješen jest učenje za gustu predikciju. Rezidualni modeli sastoje se od 4 bloka pri čemu se izlaz iz svakog bloka smanji za faktor 2 [7].

To znači da podatak pri izlazu iz mreže ima 32 puta manju rezoluciju nego što ju je izvorno imao. Drastično smanjenje rezolucije i učenje na smanjenoj rezoluciji dovodi do nemogućnosti učenja modela o detaljima koji se izgube prilikom smanjenja.

Kako bi se spriječio navedeni problem, koristi se ljestvičasto naduzorkovanje [14]. Arhitektura mreže s ljestvičastim uzorkovanjem prikazana je na 2.9 te će nam poslužiti za lakše objašnjenje ideje naduzorkovanja.

Ljestvičasto naduzorkovanje ostvaruje se uzastopnim pribrajanjem aktivacijskih značajki višeg i nižeg sloja. Kako bismo mogli zbrojiti mape značajki, dimenzije pojedinih mapa moraju odgovarati. Budući da su aktivacijske značajke višeg sloja uvijek

umanjene za faktor 2 s obzirom na aktivacijske značajke nižeg sloja, potrebno je značajke višeg sloja bilinearano naduzorkovati za faktor 2.



Slika 2.9: Arhitektura mreže s ljestvičastem naduzorkovanjem [14].

Nadalje, potrebno je izjednačiti i broj značajki što se postiže primjenom konvolucije s veličinom jezgre 1 na izlaz nižeg sloja. Značajke je sada moguće zbrojiti, a nakon zbrajanja ako je potrebno provede se još konvolucija s veličinom jezgre 1 kako bi se smanjila dimenzionalnost mape značajki te konvolucija s veličinom jezgre 3 čija je zadaća zaglađivanje značajki. Zaglađene značajke se propagiraju dalje u mrežu pri čemu one sada predstavljaju izlaz višeg sloja [14].

Navedeni postupak provodi se dok se ne zbroje sve vrijednosti izlaza iz blokova. Dodatno je moguće i na izlaz mreže propagirati i pomoćne gubitke koji se izračunavaju na pojedinom sloju naduzorkovanja. Pokazalo se kako pribrajanje pomoćnog gubitka u izrazito dubokim modelima (npr. DenseNet-121) pridonosi uspješnosti modela [14].

2.4. Primjene modela dubokog učenja

Već sam spomenula kako je važno svojstvo neuronskih mreža mogućnost učenja iz podataka. Učenjem iz podataka moguće je riješiti veliki broj problema računalnog vida. Neuronske mreže nude rješenje na problem klasifikacije i regresije podataka.

Regresiju provodimo u slučaju kontinuiranih predikcija, dok klasifikaciju provodimo u slučaju kategoričkih predikcija. Klasifikacija podataka je postupak pridjeljivanja oznaka (eng. *labels*) uzorcima na temelju značajki uzorka (boje, težina, oblik, ...) [2].

Nadalje, klasifikaciju je moguće provesti nad, primjerice, cijelom slikom pa je izlaz iz mreže jednojedinično kodiran pri čemu se za cijelu sliku određuje kojoj klasi

pripada, no moguće ju je provesti i nad manjim dijelovima slike. Klasifikacija pojedinačnog dijela slike, primjerice pojedinog piksela naziva se segmentacija.

Klasifikacija neuronskim mrežama ima mnogo različitih područja primjene. U području medicine [24], neuronske mreže uče prepoznavati različite oblike zloćudnih bolesti s magnetskih slika tijela. Mogućnost rane detekcije bolesti u većini slučajeva može spasiti čovjekov život.

Same neuronske mreže koriste se i kod obrade prirodnog jezika gdje se neuronska mreža koristi kako bi se razumio tekst ili generirao prijevod teksta na drugom jeziku i slično. Virtualni pomoćnici ili aplikacije za prevođenje jezika ne bi bili ostvarili bez uporabe dubokih modela. Nadalje, vrlo je česta primjena i u području prepoznavanja govora kada se neuronska mreža koristi kako bi ulazne podatke koji se sastoje od glasovnog zapisa pretvorila u tekst.

Konačno, izrazito važno područje primjene je i autonomna vožnja [10] koja je u posljednje vrijeme sve popularnija te se sve više razvija. Autonomna vožnja izrazito je složen proces koji iziskuje visoku točnost prepoznavanja objekata i podražaja iz okoline. Duboki modeli koriste se kako bi se raspoznala okolina u kojoj se autonomno vozilo nalazi, ali i kako bi ono znalo prepoznati ograničenja i mogućnosti svog kretanja.

3. Semantička segmentacija i njezine primjene

Semantička segmentacija je jedna od metoda računalnog vida čiji je cilj dodjeljivanje oznake klase svakom pikselu sa slike. Iako je samu segmentaciju moguće postići promatrajući bliska područja, promjenu boje, uspoređujući piksele i superpiksele te ostalim sličnim metodama [21], te se metode ne koriste toliko često jer se pokazalo kako ne daju veliku točnost. Međutim, veliki uspjeh u rješavanju problema semantičke segmentacije dogodio se uvođenjem dubokih konvolucijskih modela u okviru dubokog učenja. U posljednje se vrijeme koriste konvolucijski modeli s ljestvičastim naduzorkovanjem koji pokazuju iznimno dobre rezultate kod detekcije objekata vrlo malih dimenzija.

Duboki konvolucijski model namijenjen segmentaciji razlikuje se od standardnog modela za klasifikaciju slike budući da je izlaz iz modela za segmentaciju segmentacijska mapa, odnosno mapa istih dimenzija (rezolucije) kao i ulazna slika, ali elementi mape odgovaraju nekoj od labela klasa koja se nalazi na slici (primjer 3.1).

(a) Ulazna slika za segmentaciju. **(b)** Labele koje određuju klase na slici.

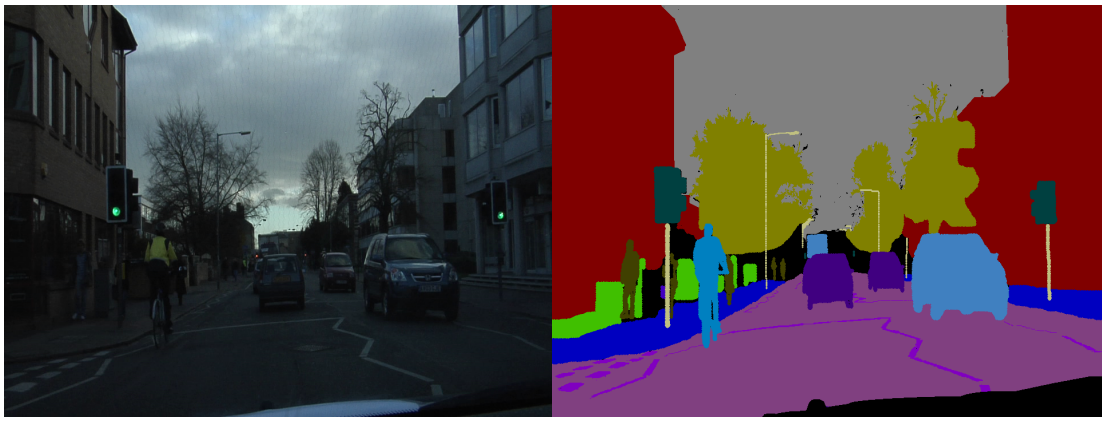
Slika 3.1: Primjer ulazne slike i maske s označenim labelama. Labele se sastoje od jedinstevnih cijelobrojnih vrijednosti pri čemu svakoj vrijednosti odgovara drukčija semantička cjelina. 1 - osoba, 2 - modni dodaci, 3 - biljke, 4 - pločnik, 5 - zgrade [12].

Pojedina klasa može predstavljati objekt veličine i do svega nekoliko piksela, stoga je potrebno koristiti model koji neće prilikom učenja gubiti informacije o detaljima na slici, odnosno objektima manjih dimenzija. Modeli s ljestvičastim uzorkovanjem sprječavaju gubitak informacije te osiguravaju naduzorkovanje na početnu rezoluciju koja je potrebna kako bi se mogli provesti daljnji algoritmi strojnog učenja za samo učenje modela.

Semantička segmentacija ima mnogo različitih područja primjene, od klasične obrade slike [22] do medicinskih snimaka [9], satelitskih snimaka [20], poljoprivrede [23] te autonomne vožnje i navigacije [18]. Autonomna vožnja jedna je od popularnijih područja primjene semantičke segmentacije u posljednje vrijeme, no upravo ona predstavlja poseban izazov.

3.1. Autonomna vožnja

Ideja autonomne vožnje uvelike se počela razvijati poticajem američke vladine agencije DARPA-e početkom ovog stoljeća. U zadnjih petnaestak godina dogodila su se revolucionarna otkrića u tom području, no kako bi sustavi autonomne vožnje bili efikasni i što više zastupljeniji u svakodnevnoj uporabi, potrebno je dodatno poboljšati točnost prepoznavanja mogućih opasnosti na cesti, pješaka, vozila i ostalih elemenata.



(a) Stvarna slika snimljena iz perspektive vozača. **(b)** Slika s označenim klasama.

Slika 3.2: Jedan primjer kadra iz skupa CamVid [3] koji se koristi za semantičku segmentaciju prirodne scene iz perspektive vozača. U ovom slučaju, svaki objekt se klasificira u jednu od 11 klasa - nebo, cesta, zgrada, drveće, auto, biciklisti, pješaci, pločnik, kolnički trakovi.

Sam proces autonomne vožnje sastoji se od nekoliko koraka, točnije od razumijevanja prirodne scene vozača, detekcije te aproksimacije kolničkih trakova, a konačno i odluke o sljedećoj akciji vozila [6]. Semantička segmentacija nudi rješenje problema razumijevanja prirodne scene vozača, kao i prepoznavanja, odnosno detekcije kolničkih trakova.

Semantičku segmentaciju možemo u ovom kontekstu razumjeti kao primarni korak pri realizaciji autonomne vožnje. Primjena semantičke segmentacije za razumijevanje prirodne scene vozača prikazana je na 3.2. Iz primjera je vidljivo kako semantička segmentacija omogućuje shvaćanje scene i prepoznavanje okoline oko vozača, razlikovanje ceste od ostalih vozila, pješaka ili prepreka te upravljačkom algoritmu daje predodžbu kuda bi se trebao kretati.

Nadalje, iako se razumijevanjem scene daje aproksimativno područje kretanja (razlikovanje ceste od pločnika), kako bi se razvio potpuno siguran i efikasan sustav autonomne vožnje, potrebno je moći prepoznati i prometne znakove, potencijalne opasnosti i kolničke trakove koji striktno propisuju kuda se vozilo smije kretati, drugim riječima koji propisuju njegovu putanju kretanja.

Problem prepoznavanja kolničkih trakova moguće je riješiti semantičkom segmentacijom, a upravo time ću se i baviti u daljnjem radu i eksperimentima. Ovaj problem je izrazito izazovan budući da su kolnički trakovi širine svega 12 cm u stvarnoj veličini, a na slici pojedini kolnički trak može predstavljati i samo nekoliko piksela ako se radi o udaljenom dijelu scene. Nadalje, potrebno je omogućiti prepoznavanje trakova u različitim, vrlo često nepogodnim uvjetima i to s velikom točnošću.

U daljnjoj izradi ovog rada bavit ću se semantičkom segmentacijom s primjenom na autonomnim vozilima budući da ću pokušati implementirati segmentaciju kolničkih trakova.

4. Programska implementacija

4.1. Korištene tehnologije

Postupak učenja modela napisan je u programskom jeziku Python[1], uz korištenje biblioteke otvorenog koda PyTorch[2] za implementaciju dubokih modela i duboko učenje. Prilikom same implementacije korištena je i biblioteka Pillow[3] s modulom Image za baratanje i manipulaciju slikama te biblioteka Numpy[4] za manipulaciju vektora, matrica i tenzora.

Kako bih mogla vremenski i memorijski efikasno evaluirati uspješnost modela, implementirala sam metodu *metrics* čija je namjena izračunavanje točnosti, preciznosti i odziva modela. Također je korištena i biblioteka scikit-learn[5].

Nadalje, kako bi se implementirale potrebne strukture za rad s dubokim modelima, korišteni su moduli knjižice biblioteke koji omogućuju implementaciju personalizirane strukture za manipulaciju skupom podataka (moduli DataSet i DataLoader). Implementiran je razred LlamasDataset koji je baziran na strukturi Dataset modula, ali je dorađen kako bi odgovarao korištenom skupu podataka i njegovim karakteristikama.

Korištene su i dostupne metode optimizacije i funkcije gubitka iz PyTorcha. Kao optimizator korišten je Adam s pretpostavljenim vrijednostima parametara dok je kao funkcija gubitka uzeta metoda unakrsne entropije s odgovarajućim težinama za pojedinu klasu (svaka težina odgovara udjelu zastupljenosti piksela te klase u cijeloj slici).

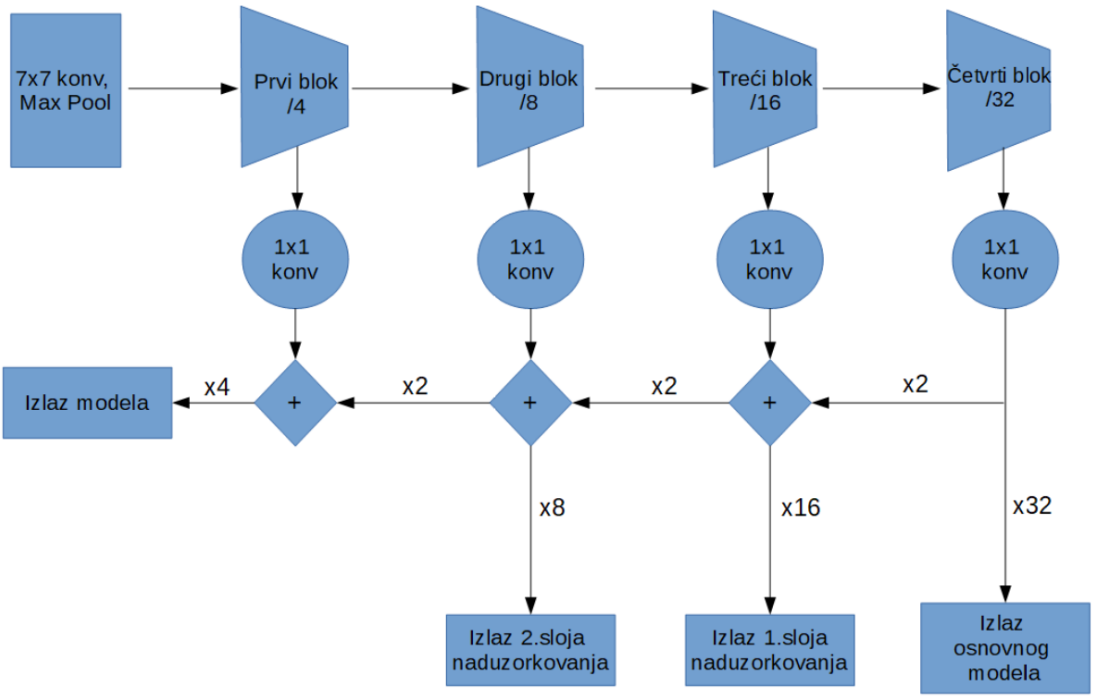
Sam proces učenja i testiranja proveden je na besplatnoj platformi Google Colaboratory[6] koja nudi mogućnost korištenja grafičke procesne jedinice NVIDIA TESLA K80 te podržava rad s udaljenim repozitorijima na gitu, kao i učitavanje podataka s besplatnog oblaka (Google Disc).

[1] `https://docs.python.org/3`
[2] `https://pytorch.org`
[3] `https://pillow.readthedocs.io/en/stable`
[4] `http://www.numpy.org`
[5] `https://scikit-learn.org/stable`
[6] `https://colab.research.google.com`

4.2. Arhitektura mreže

Model mreže zasnovan je na pretreniranom modelu ResNet18 [7] koji koristi rezidualne blokove, a sastoji se od 18 slojeva. Model mreže velikim dijelom odgovara izvornom modelu, no za potrebe provedbe semantičke segmentacije, kao i za prilagodbu modela skupu podataka, napravila sam manje preinake po uzoru na predloženu arhitekturu modela iz [17].

Kraj mreže izvornog modela sastoji se od sloja globalnog sažimanja (eng. *Average Pool*) i potpuno povezanog sloja, no ja sam zamijenila te slojeve konvolucijskim slojem s korakom i filterom veličine 1. Broj filtera odgovara broju klasa, odnosno 2 za slučaj binarne segmentacije i 5 za slučaj višeklasne segmentacije. Nadalje, implementirala sam i ljestvičasto naduzorkovanje kroz 3 sloja u modelu.



Slika 4.1: Prikaz arhitekture modela mreže. Dodatno je implementiran i model u kojem se pri izlasku iz modela, prosljeđuju i izlazi sa slojeva unutarnjih predikcija (*izlaz 1.sloja i 2.sloja naduzorkovanja*).

Ljestvičasto naduzorkovanje sam ostvarila tako da sam viši sloj bilinearano naduzorkovala za faktor 2, a nižem sloju sam smanjila broj značajki pomoću konvolucije s korakom i filterom veličine 1. Slojeve sam zbrojila i dodatno primijenila konvoluciju s korakom 1 i veličinom filtera 3 te proslijedila dublje u mrežu na sljedeći sloj. Potpuni prikaz mreže prikazan je na 4.1.

Kako bih model mreže prilagodila korištenom skupu podataka, izmijenila sam graf mreže tako da sam zbrojila težine postavljene za ulazni konvolucijski sloj. Time sam osigurala da mreža pri ulazu prima jednokanalne, odnosno crno-bijele slike. Također, pri samom izlasku iz mreže, dodala sam i bilinearno naduzorkovanje na početnu rezoluciju kako bih mogla provesti usporedbu rezultata za pojedini piksel (dimenzije predikcije i labele moraju odgovarati).

Dodatnu regularizaciju nisam implementirala, stoga se mreža oslanja na regularizacijski efekt normalizacije grupa zbog čega je iznimno važno koristiti grupe za učenje (eng. *batch*). Prilikom učenja modela, korištena je veličina grupe od 12 slika.

5. Skup podataka LLAMAS

U procesu učenja i testiranja modela mreže korišten je skup podataka LLAMAS [1]. LLAMAS je skup podataka koji je namijenjen semantičkoj segmentaciji prometnih trakova. Kod segmentacije kolničkih trakova, skup podataka predstavlja dodatan izazov budući da je sama kolnička traka u prirodi veličine svega 12 centimetara, što na slici s velike udaljenosti može značiti tek nekoliko piksela.

Skup podataka LLAMAS razvijen je od strane njemačke tvrtke Bosch, a predstavljen je prošle godine. Radi se o skupu podataka skupljenom iz snimaka s 14 različitih autocesta ukupne duljine 350 kilometara. LLAMAS je namijenjen za razvoj autonomnih vozila budući da omogućuje provedbu segmentacije te aproksimacije kolničkih trakova.

Sam skup podataka broji preko 100,000 slika te je već unaprijed podijeljen u podskupove za učenje, validiranje i testiranje. Skup za učenje sadrži gotovo 60,000 slika, skup za validaciju 20,000 slika, a skup za testiranje 20,000 slika.

(a) Primjerak slike iz skupa za učenje. **(b)** Maska s označenim labelama.

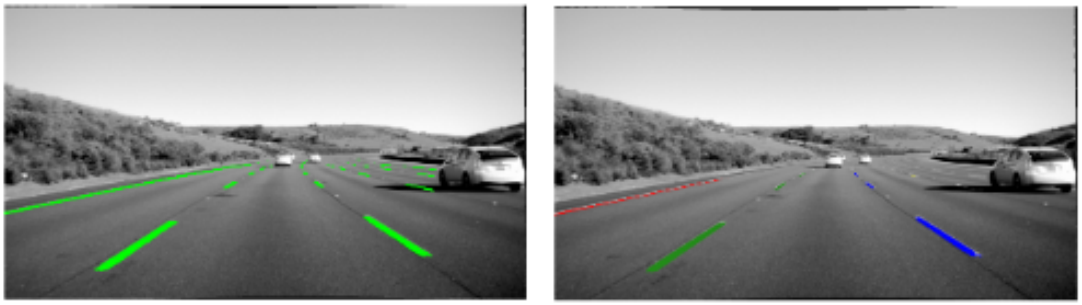
Slika 5.1: Primjer kadra iz skupa podataka LLAMAS.

Kod skupa za testiranje još uvijek nisu dostupne labele, pa kako bih izbjegla taj problem, skup za validaciju sam podijelila na dva podskupa, 10,000 slika u skup za validaciju, a preostalih 10,000 slika u skup za testiranje. Nadalje, dodatno sam napravila i podrezani skup za učenje koji se sastoji od 2,400 slika. Iz svakog podskupa za učenje (ukupno ih je 8) je uzeto 300 slika te je tako napravljen reprezentativni uzorak manjeg

obujma kako bi se ubrzalo učenje modela.

Svaka slika iz skupa popraćena je svojom JSON (eng. *JavaScript Object Notation*) datotekom u kojoj su definirani markeri pojedine labele. Marker je predstavljen podacima za automatsko generiranje labela koje se provodi postupkom sličnim GraphSLAM-u [11]. Svaki marker sadrži informaciju o vrsti/nazivu traka, početnoj i krajnjoj točki te pikselu traka. JSON datoteke potrebno je provesti kroz službene skripte za generiranje labela koje su dostupne na službenom git repozitoriju [1].

Slike u LLAMAS-u dolaze u crno-bijeloj (eng. *grayscale*) ili trobojnoj varijanti. Jednokanalna (crno-bijela) verzija je manje memorijske veličine pa sam nju koristila. Svaka slika rezolucije je 1276 x 717 piksela. Iz 5.1 vidljivo je kako u gornjem dijelu slike nema korisnih informacija o prometnim trakovima, a kako bih izvukla maksimalnu količinu informacija iz pojedine slike, svakoj slici sam odrezala gornju polovicu piksela te sam ih dodatno smanjila kako bih ubrzala proces učenja.



(a) Binarna segmentacija kolničkih trakova. **(b)** Višeklasna segmentacija kolničkih trakova.

Slika 5.2: Primjer različitih vrsta segmentacije na jednom kadru iz skupa za učenje.

Micanjem gornjeg dijela slike, ublažila sam problem disbalansa među klasama koji je vrijedio na cijeloj slici. Disbalans klasa (pozadina i trakovi) i dalje postoji, ali manji je nego prije micanja gornjeg dijela slike. U prosjeku su na jedan piksel kolničkog traka dolazili čak 85 piksela pozadinskih, nepotrebnih piksela, dok nakon transformacije dolazi 30 piksela.

Završna rezolucija slika za binarne eksperimente je 638 x 200 piksela, dok je za višeklasne eksperimente 960 x 288 piksela. Kod binarnih eksperimenata, maska pojedine slike sastoji se samo od labela 0 - pozadina i 1 - trak, dok se kod višeklasnih eksperimenata maska sastoji od labela 0 - pozadina, 1 - daljnji lijevi trak (l_0), 2 - bliži lijevi trak (l_1), 3 - bliži desni trak (l_2), 4 - daljnji desni trak (l_3). Različite klase označavaju različit položaj, odnosno udaljenost trakova od vozila/kamere (slika 5.2).

[1] `https://github.com/karstenBehrendt/unsupervised_llamas`

Ukupan broj klasa	5
Broj korištenih klasa	2, 5
Ukupan broj slika	79,113
Skup za učenje	58,269
Podrezani skup za učenje	2,400
Skup za validaciju	10,029
Skup za testiranje	10,815
Izvorna rezolucija	1276 x 717
Korištena rezolucija	638 x 200, 960 x 288

Tablica 5.1: Osnovni podaci za skup podataka LLAMAS.

6. Eksperimentalni rezultati

Kako bih mogla evaluirati model i provjeriti njegovu uspješnost te ga usporediti s drugim modelima, potrebno je koristiti neku od evaluacijskih mjera, odnosno metrika koje su opisane u nastavku.

6.1. Metrike

Budući da ću provesti semantičku segmentaciju koja se odražava kao klasifikacija svakog piksela, odnosno za svaki piksel se daje odgovor za pojedinu klasu pripada li joj (DA) ili ne pripada (NE), promatrat ću 4 moguća ishoda za pojedini piksel:

- TP (eng. *True Positive*) - točno pozitivno klasificirani pikseli
- TN (eng. *True Negative*) - točno negativno klasificirani pikseli
- FP (eng. *False Positive*) - netočno pozitivno klasificirani pikseli
- FN (eng. *False Negative*) - netočno negativno klasificirani pikseli

Uspješnost modela moguće je mjeriti točnošću (eng. *Accuracy*), odnosno udjelom točno klasificiranih piksela u ukupnom broju piksela:

$$A_{cc} = \frac{TP + TN}{TP + TN + FP + FN} \tag{6.1}$$

Međutim, ja nisam evaluirala točnost mog postupka na ovaj način. Računanje točnosti na prethodno opisani način je moguće samo ako ne postoji disbalans u zastupljenosti klasa [15], a kod skupa podataka LLAMAS postoji disbalans u zastupljenosti klasa.

Nadalje, kako bih metriku točnosti prilagodila disbalansu klasa, uvest ću preciznost (eng. *Precision*) i odziv (eng. *Recall*). Preciznost se definira kao

$$P = \frac{TP}{TP + FP} \tag{6.2}$$

Preciznost predstavlja udio ispravnih pozitivno detektiranih piksela u ukupnom broju pozitivno detektiranih piksela. S druge strane, odziv se definira kao

$$R = \frac{TP}{TP + FN} \tag{6.3}$$

Odziv predstavlja udio ispravnih pozitivno detektiranih piksela u ukupnom broju pozitivnih piksela [15]. Konačno, moguće je definirati i metriku F1 kao

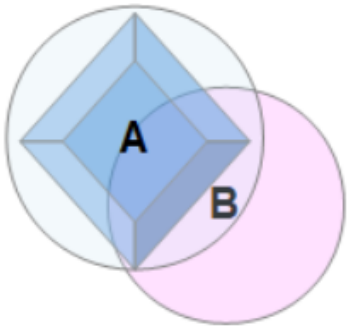
$$F1 = \frac{2PR}{P + R} \tag{6.4}$$

Metrika F1 daje dobre rezultate za procjenu uspješnosti modela kod disbalansa klasa pa sam nju i koristila.

Dodatno sam još koristila i Jaccardov indeks (eng. *Jaccard index, Intersection over Union*) koji je česta, mjera evaluacije modela u semantičkoj segmentaciji kada promatra sličnost uzoraka. Formalno ga definiramo kao omjer presjeka i unije predikcija i oznaka. Ako koristimo prethodno uvedene pojmove TP, TN, FP i FN, Jaccardov indeks možemo izraziti kao

$$IoU = \frac{TP}{TP + FN + FP} \tag{6.5}$$

Presjek predikcije i oznake bit će vrijednost ispravno detektiranih pozitivnih piksela, dakle TP. No, unija predikcije i labela sastojat će se od ponovno točnih pozitivno detektiranih piksela, ali i od netočnih pozitivno i negativno klasificiranih piksela, što je vidljivo i sa 6.1.



Slika 6.1: Plavi romb predstavlja objekt koji se detektira. Skup A označava labelu (stvarne pozitivne piksele), a skup B predikciju. Presjek skupova A i B bit će upravo točno detektirani pozitivni pikseli, dok će unija skupova A i B biti točno detektirani pozitivni pikseli, ali i netočno detektirani pikseli.

Uvest ću još i mjeru prosječne preciznosti (eng. *Average Precision*) koja se definira kao

$$AP = \int_0^1 p(r)dr, \tag{6.6}$$

pri čemu r predstavlja krivulju preciznost-odziv. Dakle, AP definiramo kao površinu ispod krivulje preciznost-odziv koju određujemo računajući preciznost za različite vrijednosti odziva. Budući da se vrijednosti odziva i preciznosti kreću u rasponu od 0 do 1, vrijednost prosječne preciznosti će se također kretati u rasponu od 0 do 1. Važno je spomenuti kako se mjera prosječne preciznosti može koristiti samo kod binarne klasifikacije.

Srednja prosječna preciznost (mAP, eng. *Mean Average Precision*) predstavlja aritmetičku sredinu prosječne preciznosti, a koristi se u slučaju višeatributne klasifikacije (eng. *multi-label classification*) kada imamo više binarnih glava.

6.2. Rezultati

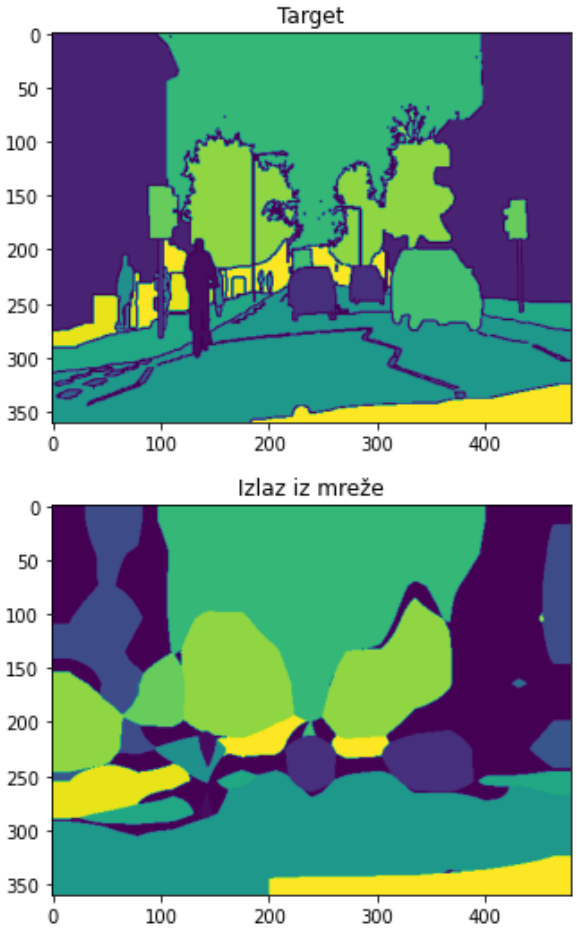
6.2.1. Eksperiment na skupu podataka CamVid

Preliminarni eksperiment sam provela nad skupom podataka CamVid [3] kako bih provjerila arhitekturu mreže s obzirom na rješavanje problema semantičke segmentacije. Semantička segmentacija kolničkih oznaka je složeniji zadatak od segmentacije scene iz perspektive vozača pa sam ovim eksperimentom željela prvo evaluirati uspješnost modela na jednostavnijem problemu.

Skup podataka CamVid sastoji se od skupa za učenje veličine 367 slika, validacijskog skupa veličine 101 slika i ispitnog skupa veličine 233 slike. Sve slike su izvorne rezolucije od 960 x 720 piksela, no ja sam provela nadzirano učenje i testiranje na smanjenoj rezoluciji od 480 x 360 piksela. Sam skup podataka broji 32 razreda, no ja sam koristila sažetiju verziju koja se sastoji od 11 razreda - nebo, cesta, zgrada, drveće, auto, biciklisti, pješaci, pločnik, kolnički trakovi.

Učenje je provedeno na 80 epoha uz veličinu grupe za učenje od 16 slika na platformi Google Colaboratory uz korištenje GPU NVIDIA TESLA K80. Jedna epoha učenja trajala je 10 sekundi. Kao funkcija gubitka korištena je unakrsna entropija, a kao optimizator je korišten ADAM s pretpostavljenim parametrima ($lr = 0.001, \beta_1 = 0.9, \beta_2 = 0.999$).

Uspješnost modela evaluirana je metrikom točnosti (A_{cc}, eng. *accuracy*) koja predstavlja udio točno klasificiranih piksela u ukupnom broju klasificiranih piksela. Pos-



Slika 6.2: Očekivani i dobiveni izlaz iz modela za jednu scenu iz skupa podataka CamVid.

tignuta je točnost od 94% na skupu za učenje te 85% na skupu za testiranje što je zadovoljavajuće.

6.2.2. Eksperimenti na skupu podataka LLAMAS

Eksperimenti na skupu podataka LLAMAS su također provedeni na platformi Google Colaboratory uz korištenje GPU NVIDIA TESLA K80. Provedeno je nadzirano učenje koje podrazumijeva postojanje ulaznih podataka i oznaka pri čemu su skupovi za učenje, validaciju i testiranje bili zastupljeni s 2,400, 10,029 i 10,815 slika. Samo učenje je provedeno na 80 epoha uz veličine grupe za učenje (eng. *batch size*) od 12 slika. Jedna epoha učenja trajala je 15 sekundi.

U svim eksperimentima kao funkcija gubitka korištena je unakrsna entropija s podešenim težinama. Prilikom binarnih eksperimenata, težine su iznosile 0.4 za pozadinu i 1.0 za kolničke trakove, dok su prilikom višeklasnih eksperimenata iznosile 0.4 za pozadinu i 1.0 za pojedinu klasu kolničkih oznaka. Nadalje, kao optimizator je korišten

ADAM s pretpostavljenim parametrima ($lr = 0.001, \beta_1 = 0.9, \beta_2 = 0.999$).

Rezultate uspješnosti modela sam uspoređivala s rezultatima iz konferencijskog članka o primjeni arhitekture mreža Student-Učitelj [10]. Dodatno sam bilježila i kretanje brojčanih pokazatelja u ovisnosti o broju slojeva ljestvičastog naduzorkovanja, kao i utjecaj aux gubitka na finalni model mreže.

Vrsta modela	Mode	Gubitak	F1	IoU
Osnovni model	učenje	0.102	0.388	0.242
	validiranje	0.104	0.368	0.226
	testiranje	0.133	0.275	0.161
Jedan sloj naduzorkovanja	učenje	0.074	0.521	0.341
	validiranje	0.088	0.519	0.350
	testiranje	0.101	0.398	0.255
Dva sloja naduzorkovanja	učenje	0.054	0.640	0.471
	validiranje	0.079	0.610	0.431
	testiranje	0.090	0.434	0.298
Tri sloja naduzorkovanja	učenje	0.044	0.678	0.514
	validiranje	0.070	0.641	0.468
	testiranje	0.080	0.499	0.341

Tablica 6.1: Brojčani pokazatelji za skup podataka LLAMAS na skupovima za učenje, validaciju i testiranje po slojevima ljestvičastog naduzorkovanja.

U tablici 6.1 prikazani su brojčani pokazatelji ovisno o korištenoj arhitekturi modela. Inicijalno sam implementirala model mreže bez ljestvičastog naduzorkovanja, no brojčani pokazatelji nisu bili dobri. Pojedini piksel se pri izlazu bilinearano naduzorkovao 32 puta što nikako nije moglo dati precizno određene rubove kolničkih oznaka. Kako bih otklonila taj problem, dodavala sam sloj po sloj ljestvičastog naduzorkovanja i bilježila promjene.

Vodeći se rezultatima iz [14] očekivala sam kako će svakim novododanim slojem ljestvičastog naduzorkovanja, model bolje uočavati detalje sa slike što bi se trebalo pokazati i na porastu točnosti te IoU-a. Pretpostavljeni rezultati su naposljetku i dobiveni te je finalni model mreže, koji ima 3 sloja naduzorkovanja, dao najbolje brojčane pokazatelje.

Nakon određenog broja epoha, pojavio se problem prenaučenosti modela. Prenaučenost modela odrazila se kao smanjenje točnosti te kao povećanje gubitka na skupu za validaciju, dok na skupu za učenje to nije vrijedilo. Budući da nisam implementirala

nikakvu dodatnu regularizaciju, moj model se u potpunosti oslanja na već implementiranu ResNet-ovu normalizaciju po grupama koja nije dovoljno snažna da spriječi pojavu prenaučenosti.

Na slikama 6.3 i 6.4 prikazani su primjeri dobre i loše segmentacije trakova. Problem, čak i nakon dodanog ljestvičastog naduzorkovanja, su udaljeniji trakovi koji se ne prepoznaju dobro, posebice ako se radi o traku isprekidane linije (trak za pretjecanje).

Nadalje, u 6.2 su prikazani rezultati eksperimenta u kojem sam ispitivala ovisnost aux gubitka na finalni model. Pokazalo se kako pribrajanje aux gubitka doprinosi poboljšanju uspješnosti modela, posebice kod dubokih modela. Učinak pribrajanja aux gubitka najbolje se primjećuje kod dubljih modela od mojega, dok kod umjereno dubokih nema velikog utjecaja.

Izlaz iz modela	Mode	Gubitak	F1	IoU
Osnovni model	učenje	0.098	0.385	0.239
	validiranje	0.103	0.385	0.238
	testiranje	0.113	0.280	0.164
Jedan sloj naduzorkovanja	učenje	0.086	0.534	0.364
	validiranje	0.093	0.502	0.336
	testiranje	0.110	0.369	0.227
Dva sloja naduzorkovanja	učenje	0.072	0.659	0.493
	validiranje	0.086	0.603	0.432
	testiranje	0.105	0.467	0.305
Tri sloja naduzorkovanja	učenje	0.068	0.706	0.546
	validiranje	0.079	0.639	0.471
	testiranje	0.098	0.506	0.335

Tablica 6.2: Brojčani pokazatelji za nezavisni model koji ima izlaze na različitim slojevima naduzorkovanja.

Model mreže je za ovaj eksperiment dodatno izmijenjen tako da se pri izlazu iz mreže propagiraju i izlazi s unutrašnjih slojeva naduzorkovanja, a ukupnom gubitku se dodaje gubitak pojedinog sloja pomnožen odgovarajućim težinskim faktorom. Gubitci koji su iz dubljeg sloja imaju manji težinski faktor. Težinski faktori (w_i) za pojedini sloj i određeni su sljedećom formulom:

$$w_i = 2^{-i} \tag{6.7}$$

Indeksi dubine (i) se gledaju od izlaza iz mreže prema unutrašnjosti, odnosno $i_{izlaz} = 0$. Konkretno, za prethodno opisanu i korištenu arhitekturu, odgovarajući faktori su navedeni u tablici 6.3.

Izlaz iz modela	Indeks težine (i)	Vrijednost težine (w_i)
Tri sloja naduzorkovanja	0	1.0
Dva sloja naduzorkovanja	1	0.5
Jedan sloj naduzorkovanja	2	0.25
Osnovni model	3	0.125

Tablica 6.3: Težine pribrajanja aux gubitka za pojedini sloj naduzorkovanja

Na 6.5 vizualizirane su predikcije s pojedinog sloja naduzorkovanja. Efekt uporabe ljestvičastog naduzorkovanja primjećuje se u razini detalja koji su detektirani, odnosno debljini kolničkog traka koji je prediktiran.

Dodatno sam još provela i eksperiment u kojem je model treniran na cijelom skupu za učenje, dakle na 58,269 slika smanjenje rezolucije 638 x 200. Učenje je u ovom slučaju provedeno kroz 20 epoha zbog iznimno dugog trajanja jedne epohe učenja. Usporedba dobivenih rezultata s rezultatima iz 6.1 prikazana je u 6.4.

Veličina skupa za učenje	Vrsta skupa	Gubitak	F1	IoU
2,400	Učenje	0.044	0.678	0.514
	Validacija	0.070	0.641	0.468
	Testiranje	0.080	0.499	0.341
58,269	Učenje	0.045	0.656	0.489
	Validacija	0.058	0.671	0.504
	Testiranje	0.056	0.563	0.393

Tablica 6.4: Usporedba uspješnosti modela ovisno o različitoj veličini skupa za učenje.

Korištenjem cijelog skupa za učenje, model uspijeva bolje generalizirati neviđene podatke. Koristeći veći broj slika, model uči na većem broju različitih situacija kada je potrebno prepoznati kolničke oznake pa su rezultati samim time i bolji za čak 6% na skupu za testiranje te za 3% na skupu za validaciju.

Konačno, u tablici 6.5 prikazana je uspješnost različitih modela na skupu podataka LLAMAS. Rezultati su dobiveni za višeklasnu segmentaciju od 5 klasa. Četiri klase označavaju različite kolničke trakove na slici ovisno o udaljenosti od auta. Evaluacija je provedena na rezoluciji od 960x288 piksela. Samo učenje provedeno je na pola

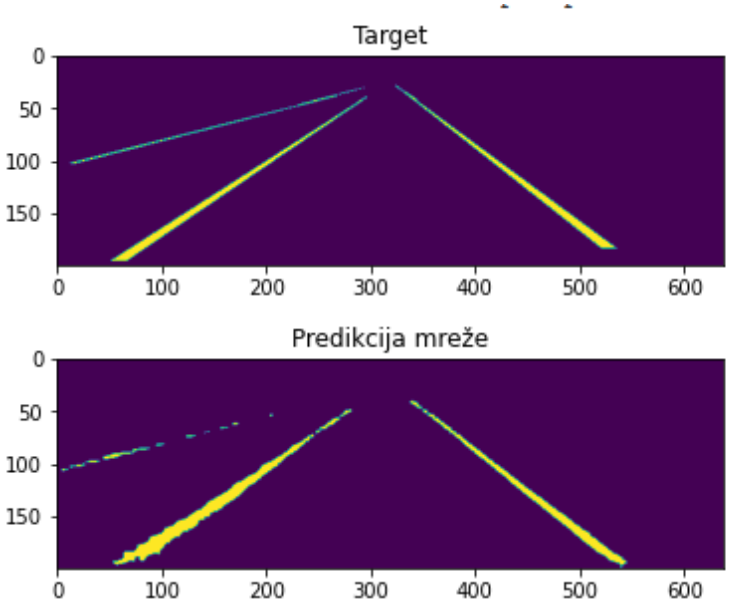
Model mreže	mAP
ERFNet-IntRA-KD [10]	0.598
ResNet-101 [7]	0.607
ResNet-50 [7]	0.579
UNet-ResNet-34 [16]	0.592
ResNet-18 (moj)	0.389

Tablica 6.5: Usporedba uspješnosti modela na skupu podataka LLAMAS.

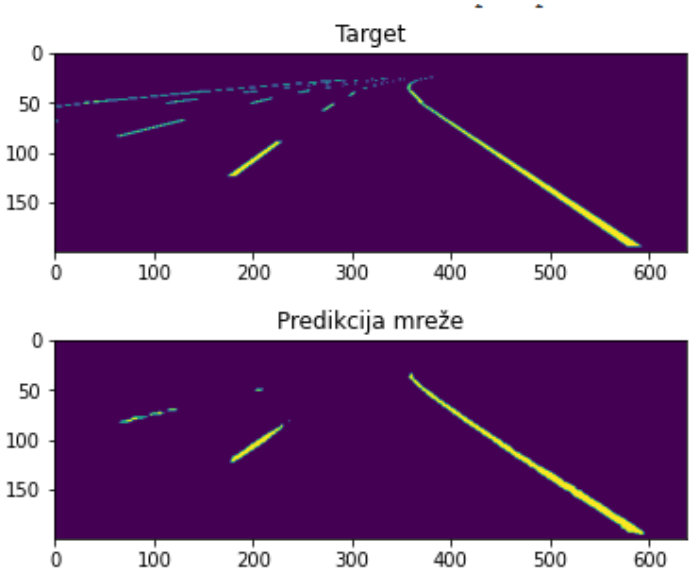
rezolucije, odnosno u istim uvjetima kao što su i prethodni eksperimenti provedeni, ali su se podaci prije izlaska iz mreže bilinearano naduzorkovali na rezoluciju od 960 x 288 piksela budući da je takva rezolucija korištena i u [10].

Mjera srednje prosječne preciznosti dobivena je kao aritmetička sredina prosječne preciznosti pojedine klase, odnosno labele. Dobiveni rezultat je zadovoljavajući iako je lošiji od rezultata dobivenih u [10] no to se može objasniti različitom arhitekturom korištenog modela.

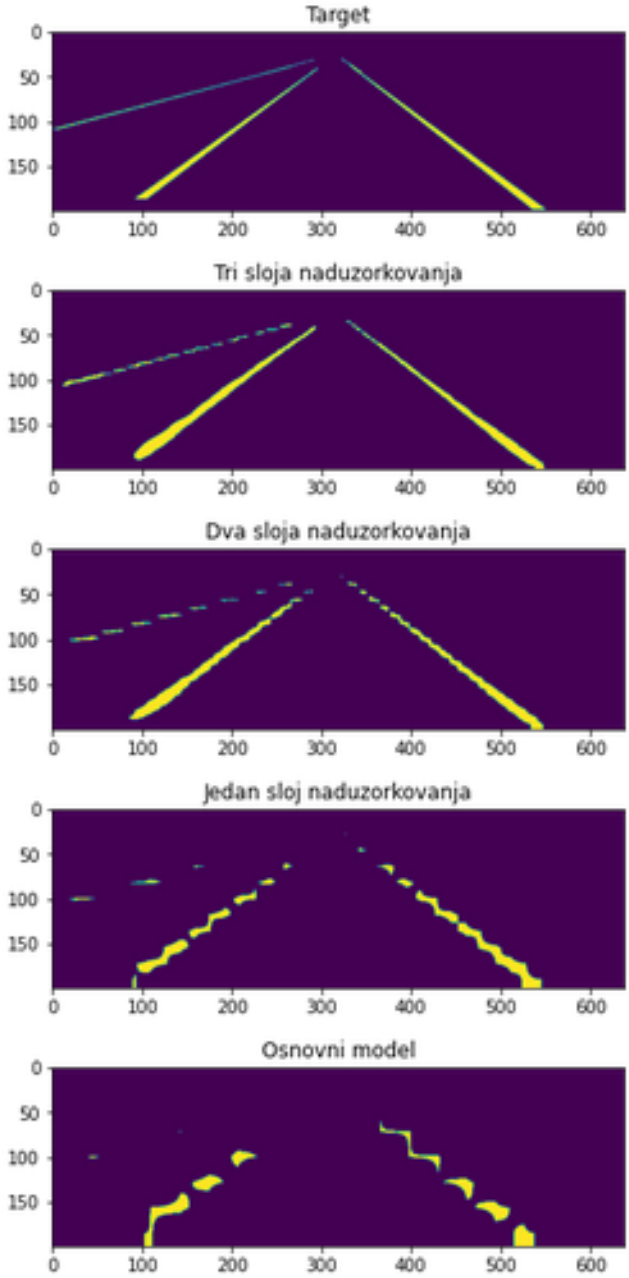
Finalni model binarne i višeklasne segmentacije dao je dobre rezultate za većinu ispitnih slučajeva. Međutim, dobri rezultati nisu dobiveni ako je bilo potrebno detektirati udaljene isprekidane trakove (trak za pretjecanje). Taj problem mogao bi se riješiti modificiranjem modela, korištenjem više dodatnih skrivenih slojeva, ali i drukčijom kombinacijom funkcije gubitka i hiperparametara. Mora se uzeti u obzir i činjenica kako u skupu podataka vrijedi veliki disbalans u zastupljenosti klasa što dodatno otežava rješavanje problema segmentacije.



Slika 6.3: Primjer dobre segmentacije trakova.



Slika 6.4: Primjer loše segmentacije trakova i usporedba s očekivanim izlazom. Vidljivo je kako se udaljeniji trakovi ne detektiraju dobro.



Slika 6.5: Predikcije mreže s različitih izlaza iz mreže i usporedba s labelom. Izlazi iz dubljih slojeva su neprecizniji što se očituje u debljoj kolničkoj traki jer jedan izlazni piksel zapravo predstavlja blok od 32 piksela (osnovni model) ili 16 piksela (1.sloj naduzorkovanja). Također, razina detalja koja se uspije detektirati u dubljim slojevima je puno lošija u odnosu na slojeve bliže izlazu iz modela.

7. Zaključak

U ovom radu ukratko su opisana osnovna načela rada i svojstva dubokih modela kao i proces njihovog učenja. Nadalje, opisana su i svojstva konvolucijskih mreža te su predstavljene rezidualne konvolucijske mreže i modeli za gustu predikciju. Prikazan je primjer semantičke segmentacije kao jedne od područja primjene konvolucijskih mreža.

Nadalje, implementiran je sustav za semantičku segmentaciju koji se temelji na dubokom konvolucijskim modelu s ljestvičastim naduzorkovanjem. Model, kao i učenje te testiranje, izvedeni su korištenjem biblioteke PyTorch koja omogućuje konfiguriranje arhitekture neuronske mreže, korištenje već gotovih pretreniranih modela te efikasno izvršavanje koda na grafičkim karticama. Implementacija sustava je provedena na besplatnoj platformi Google Colaboratory koja nudi mogućnost korištenja grafičke kartice NVIDIA TESLA K80.

Model mreže implementiran je po uzoru na arhitekturu predstavljenu u [14] i [17], a sam model prikazan je na 4.1. Polazni model bio je pretrenirani model ResNet-18 koji je dodatno nadograđen slojevima ljestvičastog naduzorkovanja. Gotov model mreže sastojao se od tri sloja naduzorkovanja.

Implementirani model ispitan je na skupu podataka LLAMAS koji je namijenjen detekciji kolničkih trakova. Učenje i eksperimenti su provedeni na smanjenoj, odnosno polovičnoj rezoluciji osim eksperimenta usporedbe s [10] koji je proveden na rezoluciji od 960 x 288 piksela.

Dobiveni rezultati su zadovoljavajući, no svakako je moguće dodatno poboljšati model. Pokazalo se da model ne raspoznaje dobro kolničke trakove koji su udaljeniji, posebice kolničke trakove crte za pretjecanje. Rješenje tog problema moguće je postići korištenjem dubljeg modela te kombiniranjem druge funkcije gubitka i hiperparametara.

LITERATURA

[1] Karsten Behrendt i Ryan Soussan. Unsupervised labeled lane marker dataset generation using maps. U *Proceedings of the IEEE International Conference on Computer Vision*, 2019.

[2] Jan Šnajder Bojana Dalbelo-Bašić, Marko Čupić. *Umjetne neuronske mreže*. URL `https://www.fer.unizg.hr/_download/repository/UmjetneNeuronskeMreze.pdf`.

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

[4] Sanket Doshi. Convolutional neural network: Learn and apply, 2019. URL `https://medium.com/@sdoshi579/convolutional-neural-network-learn-and-apply-3dac9acfe2b6`.

[5] Ian Goodfellow, Yoshua Bengio, i Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[6] Mohammad Hajizadeh Saffar, Mohsen Fayyaz, Mohammad Sabokrou, i Mahmood Fathy. Semantic Video Segmentation: A Review on Recent Approaches. *arXiv e-prints: 1806.06172*.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, i Jian Sun. Deep Residual Learning for Image Recognition. *IEEE Conference on Computer Vision and Pattern Recognition*, .

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, i Jian Sun. Identity Mappings in Deep Residual Networks. *ECCV 2016*, .

[9] Kelei He, Chunfeng Lian, Ehsan Adeli, Jing Huo, Yinghuan Shi, Yang Gao, Bing Zhang, Junfeng Zhang, i Dinggang Shen. TripletUNet: Multi-Task U-Net with

Online Voxel-Wise Learning for Precise CT Prostate Segmentation. *arXiv e-prints: 2005.07462*, .

[10] Yuenan Hou, Zheng Ma, Chunxiao Liu, Tak-Wai Hui, i Chen Change Loy. Inter-Region Affinity Distillation for Road Marking Segmentation. *CVPR 2020*.

[11] S. Thrun J. Levinson. Robust vehicle localization inurban environments using probabilistic maps. *Roboticsand Automation (ICRA), 2010 IEEE International Conference*, 2010.

[12] Jeremy Jordan. An overview of semantic image segmentation., 2018. URL `https://www.jeremyjordan.me/semantic-segmentation/`.

[13] Diederik P. Kingma i Jimmy Ba. Adam: A Method for Stochastic Optimization. *3rd International Conference for Learning Representations 2015*.

[14] Ivan Krešo, Josip Krapac, i Siniša Šegvić. Efficient Ladder-style DenseNets for Semantic Segmentation of Large Images. *arXiv e-prints: 1905.05661*.

[15] Jan Šnajder. *Strojno učenje 21: Vredonovanje modela*. URL `https://www.fer.unizg.hr/_download/repository/SU-2019-21-VrednovanjeModela.pdf`.

[16] Philipp Fischer i Thomas Brox Olaf Ronneberger. U-net: Convolutional networks for biomedical image segmen-tation. *MICAI 2015*.

[17] Marin Oršić, Ivan Krešo, Petra Bevandić, i Siniša Šegvić. In Defense of Pre-trained ImageNet Architectures for Real-time Semantic Segmentation of Road-driving Images. *CVPR 2019: 12607-12616*.

[18] Miho Adachi Takeshi Nakajima Hiroki Ishida Kazuya Kojima Risako Aoki Takuro Oki Shingo Kobayashi Ryusuke Miyamoto, Yuta Nakamura. Vision-based road-following using results of semantic segmentation for autonomous navigation. *2019 IEEE 9th International on Consumer Electronics (ICCE-Berlin)*.

[19] Sumit Saha. A comprehensive guide to convolutional neural networks — the eli5 way, 2018. URL `https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-w`

[20] Onur Tasar, Alain Giros, Yuliya Tarabalka, Pierre Alliez, i Sébastien Clerc. DAugNet: Unsupervised, Multi-source, Multi-target, and Life-long Domain Adaptation for Semantic Segmentation of Satellite Images. *arXiv e-prints: 2005.06216*.

[21] Tingwu Wang. *Semantic Segmentation*. URL https://www.cs.toronto.edu/~tingwuwang/semantic_segmentation.pdf.

[22] S. Paris M. Pollefeys Y. Aksoy, T. Oh i W. Matusik. Semantic soft segmentation. *ACM Trans. Graph.*, 2018.

[23] Laura Zabawa, Anna Kicherer, Lasse Klingbeil, Reinhard Töpfer, Heiner Kuhlmann, i Ribana Roscher. Counting of grapevine berries in images via semantic segmentation using convolutional neural networks. *ISPRS Journal of Photogrammetry and Remote Sensing*, 164:73–83, Lipanj 2020. doi: 10.1016/j.isprsjprs.2020.04.002.

[24] Eshal Zahra, Bostan Ali, i Wajahat Siddique. Medical Image Segmentation Using a U-Net type of Architecture. *arXiv e-prints: 2005.05218*.

Semantička segmentacija kolničkih trakova

Sažetak

Semantička segmentacija je jedan od problema kojim se bavi računalni vid. Semantička segmentacija ima širok spektar primjene, od medicine, poljoprivrede do u posljednje vrijeme sve popularnijeg područja autonomne vožnje. Autonomna vožnja nameće poseban izazov semantičkoj segmentaciji, posebice ako se radi o segmentaciji kolničkih trakova budući da su sami trakovi izrazito uski i teško primjetni na slikama. Međutim, dobre rezultate u rješavanju tog problema postižu duboki konvolucijski modeli s ljestvičastim naduzorkovanjem. U ovom radu implementiran je jedan takav model te su provedeni i prikazani eksperimentalni rezultati na skupu podataka LLAMAS.

Ključne riječi: semantička segmentacija, računalni vid, autonomna vozila, duboki konvolucijski modeli, ljestvičasto naduzorkovanje, LLAMAS

Semantic segmentation of road lanes

Abstract

Semantic segmentation is one of the problems that computer vision deals with. Semantic segmentation has a wide range of applications, from medicine, agriculture to the recently increasingly popular field of autonomous driving. Autonomous driving imposes a special challenge on semantic segmentation, especially when it comes to lane segmentation since the lanes themselves are extremely narrow and difficult to notice in images. However, deep convolutional models with ladder upsampling achieve good results in solving this problem. In this paper, one such model was implemented and experimental results were performed and presented on the LLAMAS data set.

Keywords: semantic segmentation, computer vision, autonomous driving, deep convolutional models, ladder upsampling, LLAMAS