SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 1729

Učenje dubokih korespondencijskih metrika trojnim gubitkom

Marin Kostelac

Zagreb, srpanj 2018.

Umjesto ove stranice umetnite izvornik Vašeg rada.
Da bi ste uklonili ovu stranicu obrišite naredbu `\izvornik`.

SADRŽAJ

1. Uvod

Računalni vid je područje računalne znanosti koje se bavi obradom slikovnih podataka. U ovom radu bavimo se problemom prepoznavanja dubine u sceni. Sustav na ulaz dobiva slike scene iz dva kuta, a na izlazu postavlja udaljenosti pojedinih piksela pomoću kojih je moguće odrediti udaljenost objekta od kamere.

Za obradu slika u današnje vrijeme se često koriste neuronske mreže, točnije konvolucijske neuronske mreže. Takve mreže su svoju uporabu u području računalnog vida našle zbog invarijantnosti na translaciju.

Korištenjem korespondencijskih metrika u računalnom vidu, visokodimenzijske podatke, tj. slike, preslikavamo u vektorski prostor manjih dimenzija koji računalu olakšava rješavanje zadatka. Na primjeru problema kojim se bavi ovaj rad, korespondencijske metrike koriste se kako bi se pronašli odgovarajući pikseli u obje slike te se prema njihovim položajima na slici određuje udaljenost točke koju pikseli prikazuju.

Trojni gubitak omogućava definiranje željenog odnosa među podacima i učenje željenog svojstva korespondencijskih metrika. Takav pristup olakšava rješavanje problema stereoskopske rekonstrukcije jer je u takvom problemu osjetno lakše modelirati problem kao odnos među podacima nego kao klasifikaciju ili regresiju.

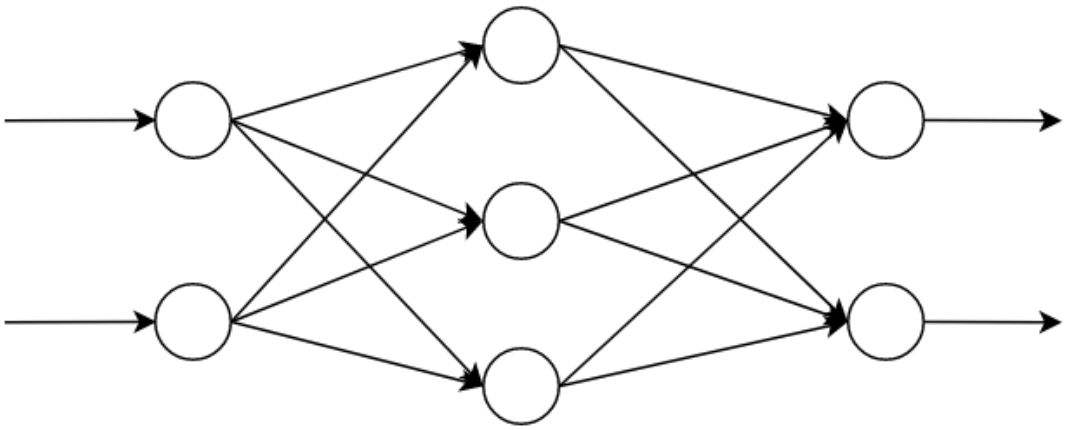
U ovom radu opisane su neuronske mreže i konvolucijske neuronske mreže. Opisana je primjena korespondencijskih metrika i trojnog gubitka na problem stereoskopske rekonstrukcije. Opisana je arhitektura primjenjena na problem i izneseni su rezultati učenja modela.

2. Neuronske mreže

U ovom poglavlju bit će objašnjene umjetne neuronske mreže i konvolucijske neuronske mreže. Pojasnit će se njihova struktura i način rada.

2.1. Umjetne neuronske mreže

Umjetne neuronske mreže su skup međusobno povezanih neurona. Primjer neuronske mreže prikazan je na slici 2.1.



Slika 2.1: Prikaz jednostavne neuronske mreže

Umjetne neuronske mreže su skup međusobno povezanih neurona. Neuroni su gradivne jedinice neuronskih mreža koji najlakše mogu biti opisani kao matematičke funkcije koje primaju vektor brojeva i na izlazu daju skalar. Najjednostavniji tip neurona prikazan je jednadžbom 2.1.

$$f(\vec{x}) = \vec{x} \cdot \vec{w} - b \tag{2.1}$$

gdje je $\vec{x}$ ulazni vektor, $\vec{w}$ vektor težina, a b prag neurona. Nedostatak ovakvog modela je njegova linearnost, čime se onemogućuje modeliranje nelinearnih funkcija.

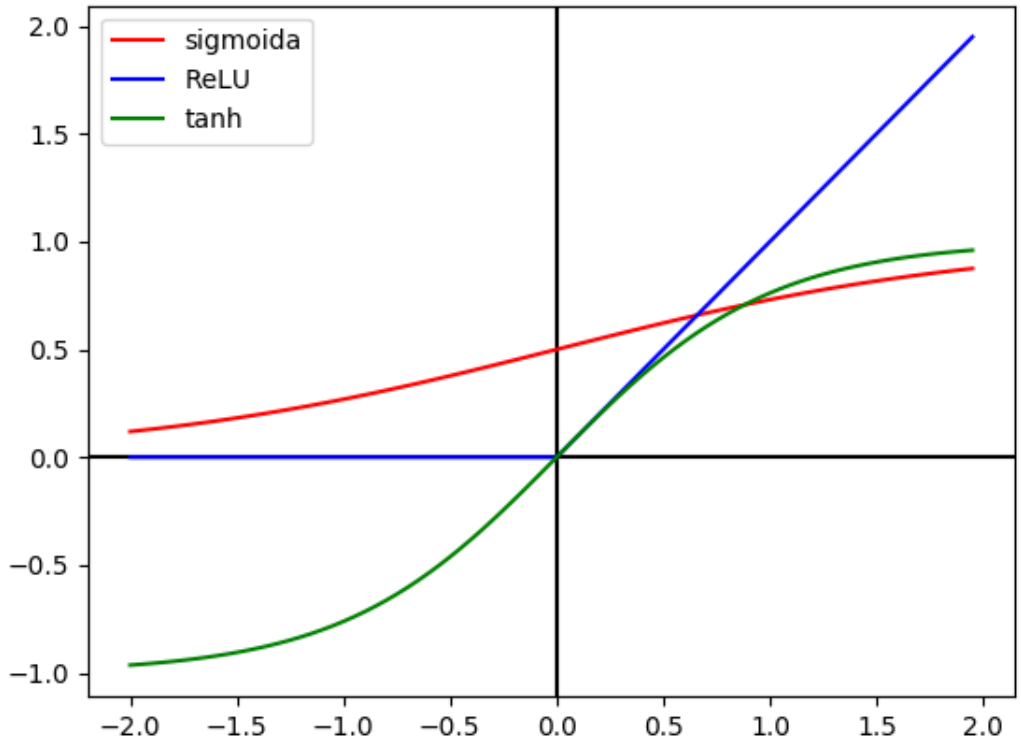
Taj problem se rješava korištenjem nelinearne aktivacijske funkcije neurona $g(x)$ kako je prikazano jednadžbom 2.2.

$$f(\vec{x}) = g(\vec{x} \cdot \vec{w} - b) \tag{2.2}$$

Neke od često korištenih aktivacijskih funkcija su sigmoidalna aktivacijska funkcija, zglobnica (engl. *rectified linear unit, ReLU*) i tangens hiperbolni. Izraz za sigmoidalnu aktivacijsku funkciju prikazan je u 2.3, a za ReLU u 2.4. Grafovi za navedene aktivacijske funkcije u rasponu $[-2, 2]$ prikazani su na slici 2.2.

$$\sigma(x) = \frac{1}{1 - e^{-x}} \tag{2.3}$$

$$relu(x) = max(0, x) \tag{2.4}$$



Slika 2.2: Prikaz aktivacijskih funkcija

Pretpostavimo da radimo s neuronskom mrežom prikazanom na slici 2.1. Nazovimo ulaze x_1 i x_2, a izlaze y_1 i y_2. Prvi sloj mreže na slici je ulazni sloj koji primljene podatke samo prosljeđuje u idući sloj mreže – skriveni sloj. Račun u matričnom obliku prikazan je u nastavku.

$$X = \begin{bmatrix} x_1 & x_2 \end{bmatrix}^T \tag{2.5}$$

$$W_1 = \begin{bmatrix} w^1_{1,1} & w^1_{1,2} \\ w^1_{2,1} & w^1_{2,2} \\ w^1_{3,1} & w^1_{3,2} \end{bmatrix} \tag{2.6}$$

$$H = W_1 X \tag{2.7}$$

Broj skrivenih slojeva može se proizvoljno mijenjati. Rezultat H dobiven u izrazu 2.7 predstavlja izlaz skrivenog sloja koji se prosljeđuje na ulaz idućeg sloja. U slučaju mreže sa slike 2.1 to je izlazni sloj koji funkcionira na identičan način, što je prikazano u izrazima 2.8 i 2.9.

$$W_2 = \begin{bmatrix} w^2_{1,1} & w^2_{1,2} & w^2_{1,3} \\ w^2_{2,1} & w^2_{2,2} & w^2_{2,3} \end{bmatrix} \tag{2.8}$$

$$Y = W_2 H = \begin{bmatrix} y_1 & y_2 \end{bmatrix}^T \tag{2.9}$$

2.2. Učenje neuronskih mreža

Neuronske mreže svoje znanje čuvaju u parametrima neurona. One imaju jaku sposobnost učenja i proizvoljan kapacitet iz razloga što mogu imati mnogo parametara koje prilagođavaju podacima pri treniranju. Najčešće korišten način za učenje neuronskih mreža je algoritam propagacije pogreške unatrag (engl. *backpropagation*). Algoritam u obliku pseudokoda prikazan je u algoritmu 1. U pseudokodu funkcija $f(\mathbf{x}^{(i)}; \boldsymbol{\theta})$ predstavlja prolaz unaprijed, tj izlaz mreže za ulaz $\mathbf{x}^{(i)}$ uz parametre $\boldsymbol{\theta}$. Funkcija $L(\hat{\mathbf{y}}^{(i)}, \mathbf{y}^{(i)})$ predstavlja gubitak. Operator $\nabla_{\boldsymbol{\theta}}$ označava gradijent funkcije po $\boldsymbol{\theta}$.

Algoritam propagacije pogreške unatrag je iterativan, ponavlja se dok se ne ispuni proizvoljni kriterij zaustavljanja. Neki od češće korištenih kriterija zaustavljanja su zaustavljanje nakon fiksnog broja epoha i zaustavljanje nakon što greška na skupu podataka za validaciju počne rasti.

Za algoritam je bitno imati što točniji gradijent kako bi se težine kretale u pravom smjeru. U pravilu, što više podataka prođe kroz mrežu prije računanja prosječnog gradijenta, to će rezultat biti točniji. Kako bi se dobio bolji gradijent, moguće je u svakom koraku koristiti manju grupu primjera za učenje (engl. *minibatch*) ili čak cijeli skup za učenje (engl. *batch*). Najčešće se za računanje gradijenta koristi manja grupa

primjera iz skupa za učenje. Takva varijanta algoritma naziva se stohastički gradijentni spust (engl. *stochastic gradient descent, SGD*).

Algoritam 1: Propagacija pogreške unatrag

Ulaz: $\mathbf{X}$ – ulazni podatci
Ulaz: $\boldsymbol{\theta}$ – težine
Ulaz: $\mathbf{Y}$ – očekivani izlaz
Ulaz: μ – stopa učenja
dok *kriterij zaustavljanja nije ispunjen* **čini**
  za svaki $\mathbf{x}^{(i)}$ *iz* $\mathbf{X}$ **čini**
    $\mathbf{g} \leftarrow \nabla_{\boldsymbol{\theta}} L(f(\mathbf{x}^{(i)}; \boldsymbol{\theta}), \mathbf{y}^{(i)})$;
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \mu * \mathbf{g}$;
  kraj
kraj

2.3. Adam

Kako bi se ubrzao postupak učenja neuronskih mreža, može se koristiti neka od varijanti gradijentnog spusta kao što su dodavanje momenta, AdaGrad, RMSProp ili Adam. Jedan od češće korištenih algoritama je Adam [6].

Algoritam prilikom ažuriranja parametara uzima u obzir prosjek gradijenata kroz korake umjesto trenutnog gradijenta kako bi smanjio utjecaj šuma. Šum u funkciji pogreške nastaje zbog slučajno odabranog podskupa primjera iz skupa za učenje ili proizlazi iz šuma u podacima. Adam pri izračunu koristi prosjek gradijenata i prosjek kvadrata gradijenata. Oba prosjeka su eksponencijalno prigušeni. Prvi se koristi kao moment koji usmjerava kretanje parametara u smjeru iz prošlih koraka, a drugi kako bi spriječio naglo pomicanje parametara koji se brzo kreću i potaknuo veće pomake parametara s manjim gradijentom. Pseudokod je prikazan u algoritmu 2.

Algoritam 2: Adam

Ulaz: α – stopa učenja

Ulaz: β_1 – stopa prigušenja za prvi moment (predložena vrijednost 0.9)

Ulaz: β_2 – stopa prigušenja za drugi moment (predložena vrijednost 0.999)

Ulaz: ϵ – konstanta za numeričku stabilizaciju (predložena vrijednost 10^{-8})

Ulaz: $\boldsymbol{\theta}$ – početni parametri modela

$\mathbf{m} \leftarrow \mathbf{0}$;

$\mathbf{v} \leftarrow \mathbf{0}$;

$t \leftarrow 0$;

dok *kriterij zaustavljanja nije ispunjen* **čini**

$\mathbf{x}^{(i)}, \mathbf{y}^{(i)} \leftarrow izaberi\_minibatch(velicina = s)$;

$\mathbf{g} \leftarrow \frac{1}{s} \nabla_{\boldsymbol{\theta}} \sum_i L(f(\mathbf{x}^{(i)}; \boldsymbol{\theta}), \mathbf{y}^{(i)})$;

$t \leftarrow t + 1$;

$\mathbf{m} \leftarrow \beta_1 \mathbf{m} + (1 - \beta_1)\mathbf{g}$;

$\mathbf{v} \leftarrow \beta_2 \mathbf{v} + (1 - \beta_2)\mathbf{g} \odot \mathbf{g}$;

$\hat{\mathbf{m}} \leftarrow \frac{\mathbf{m}}{(1-\beta_1^t)}$;

$\hat{\mathbf{v}} \leftarrow \frac{\mathbf{v}}{(1-\beta_2^t)}$;

$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \frac{\hat{\mathbf{m}}}{\sqrt{\hat{\mathbf{v}}}+\delta}$;

kraj

3. Konvolucijske neuronske mreže

Konvolucijske neuronske mreže [2] su tip neuronskih mreža koji je prilagođen radu s višedimenzijskim podacima. Često se koriste za rad sa slikama. Za razliku od potpuno povezanih neuronskih mreža, konvolucijske neuronske mreže su invarijantne na translaciju.

3.1. Konvolucija

Konvolucija [2] je matematička operacija nad dvjema funkcijama. Općeniti izraz za jednodimenzijsku konvoluciju naveden je u jednadžbi 3.1.

$$s(t) = (x * w)(t) = \int x(a)w(t-a)da \tag{3.1}$$

U konvolucijskim neuronskim mrežama uglavnom koristimo konvoluciju diskretnih funkcija u dvije dimenzije. To znači da u jednadžbu umjesto integrala dolazi sumacija i da funkcije primaju po dva argumenta. Izraz za takav tip konvolucije prikazan je u 3.2. U ovom zapisu izmijenjena je notacija kako bi bila bliža primjeni konvolucije: I predstavlja sliku (engl. *image*), a K predstavlja jezgru (engl. *kernel*).

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n) \tag{3.2}$$

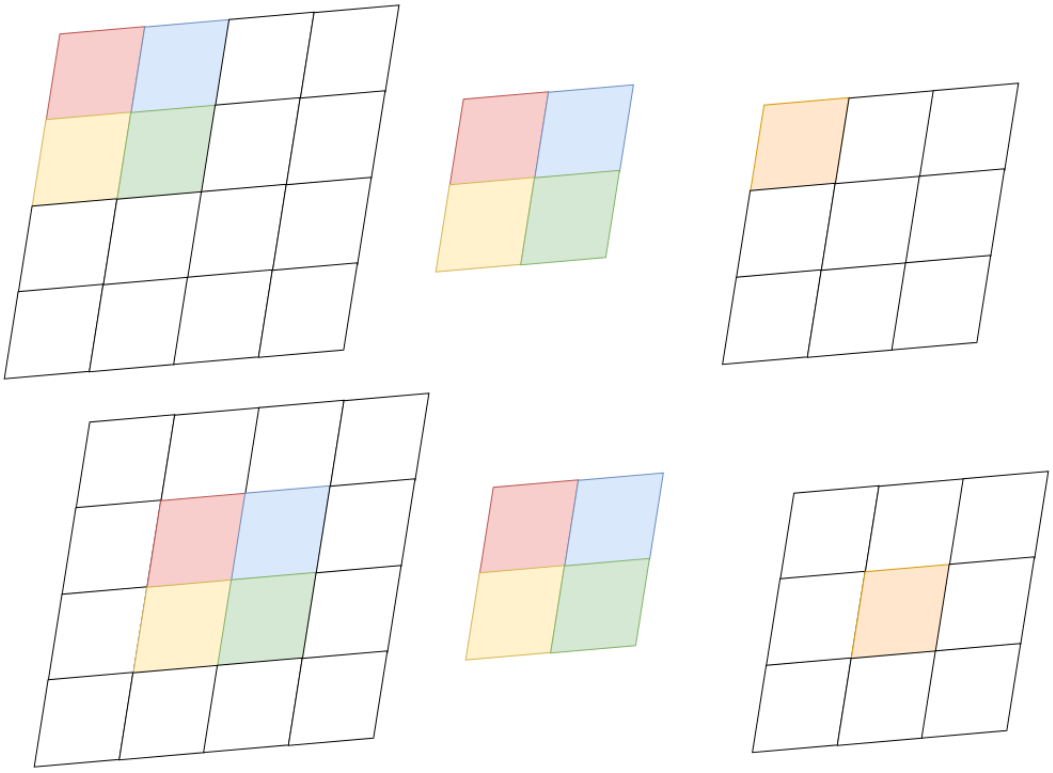
Konvolucija ima svojstvo komutativnosti, što znači da je moguće izraz napisati na način prikazan jednadžbom 3.3. Takav zapis omogućava jednostavniju implementaciju zato što su jezgre u konvoluciji manje od ulazne slike pa je manje mogućih vrijednosti za indekse m i n na kojima je rezultat definiran.

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n) \tag{3.3}$$

Iz jednadžbe 3.3 može se primjetiti da će jezgra prilikom konvolucije biti zrcaljena. To osigurava svojstvo komutativnosti koje u praksi nije potrebno pa se u bibliotekama

za strojno učenje često koristi kros korelacija. U praksi, jedina razlika u odnosu na konvoluciju je činjenica da će parametri jezgre biti naučeni zrcaljeno. Kros korelacija je prikazana jednadžbom 3.4. Slikovni prikaz ove operacije može se vidjeti na slici 3.1.

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i+m, j+n) K(m,n) \tag{3.4}$$



Slika 3.1: Konvolucija

3.2. Slojevi konvolucijskih neuronskih mreža

U konvolucijskim neuronskim mrežama javljaju se dva karakteristična tipa slojeva: konvolucijski slojevi (engl. *convolution layers*) i slojevi sažimanja (engl. *pooling layers*).

3.2.1. Konvolucijski slojevi

Konvolucijski slojevi na ulaz dobivaju sliku i koristeći konvoluciju obrađuju podatke jezgrama unutar sloja. Na izlazu sloj generira po jednu mapu značajki za svaku jezgru. Operacija konvolucije je detaljnije objašnjena u poglavlju 3.1.

Jezgre se koriste kako bi reagirale na određene elemente slike, što se pokušava naučiti kako bi se riješio zadatak. Kako se jezgre pomiču po ulaznoj slici, one će reagirati na element kojem odgovaraju bez obzira na njegovu poziciju. Time se postiže važno svojstvo konvolucijskih neuronskih mreža: invarijantnost na translaciju.

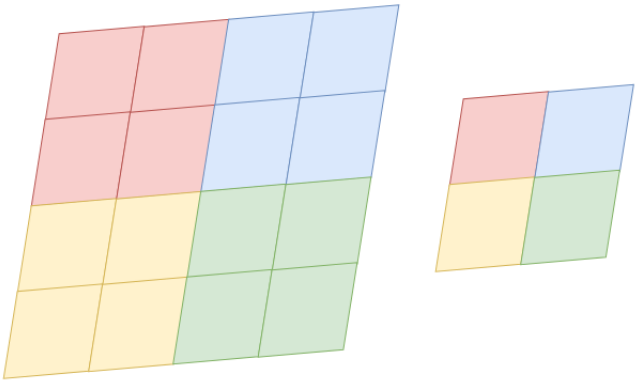
Dodatni parametar koji se javlja u konvolucijskim slojevima je korak konvolucije (engl. *stride*). Njime se određuje za koliko mjesta se jezgra pomiče po ulaznoj slici u svakom koraku, tj. za koju vrijednost uvećavamo parametre m i n. U poglavlju 3.1 pretpostavlja se da je korak konvolucije uvijek 1.

U konvolucijskim neuronskim mrežama najčešće se koriste dva tipa konvolucije: konvolucija bez nadopunjavanja (engl. *valid*) i konvolucija uz nadopunjavanje (engl. *same*). Konvolucija bez nadopunjavanja znači da se jezgra pomiče po ulaznoj slici za definirani korak dok ne izađe van okvira slike.

Ako je korak konvolucije veći od 1, to znači da rubovi slike neće biti zahvaćeni konvolucijom te neće utjecati na krajnji rezultat, što znači potencijalni gubitak informacija. Kako bi se to spriječilo, koristi se konvolucija uz nadopunjavanje koja nadopunjuje rubove ulazne slike jednoliko s obje strane kako bi svi elementi ulazne slike utjecali na izlaz. Nedostatak takvog pristupa je potencijalni utjecaj nadopune na rubovima izlaznih mapa značajki.

3.2.2. Slojevi sažimanja

Slojevi sažimanja rade slično kao konvolucijski slojevi, samo se umjesto jezgre po ulaznim podacima pomiče pomični prozor. Element izlaza je funkcija elemenata unutar pomičnog prozora, najčešće maksimum ili srednja vrijednost. Ilustracija sažimanja prikazana je na slici 3.2.



Slika 3.2: Sažimanje

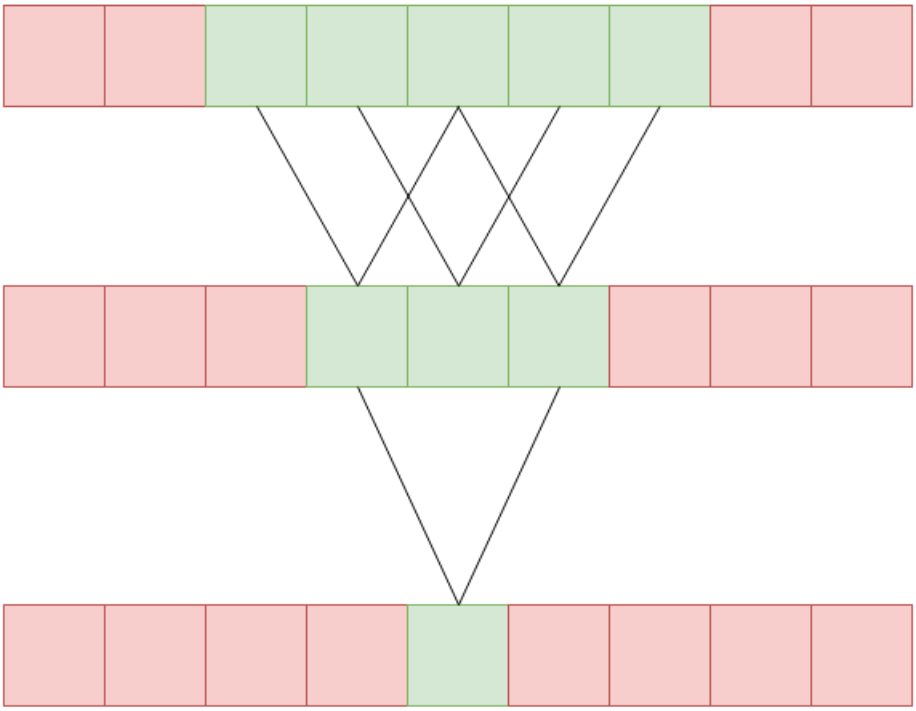
Za razliku od konvolucijskog sloja, sloj sažimanja ne stvara mijenja broj mapa značajki, ali smanjuje njihove dimenzije. U slojevima sažimanja se korak konvolucije obično namješta tako da nema preklapanja u položajima pomičnog prozora.

Slojevi sažimanja se koriste kako bi se smanjila dimenzionalnost podataka čime se smanjuje računska složenost uz potencijalni gubitak dijela informacija. Također, povećavaju prostornu invarijantnost mreže smanjujući razmak među aktivacijama.

3.3. Receptivno polje

Receptivno polje u konvolucijskim neuronskim mrežama predstavlja sve elemente ulazne slike koji utječu na određeni element izlazne mape značajki nekog sloja [2, 8]. Za zadatke s kojima se susrećemo u računalnom vidu važno je da je receptivno polje dovoljno veliko kako bi se mogle prepoznati potrebne značajke.

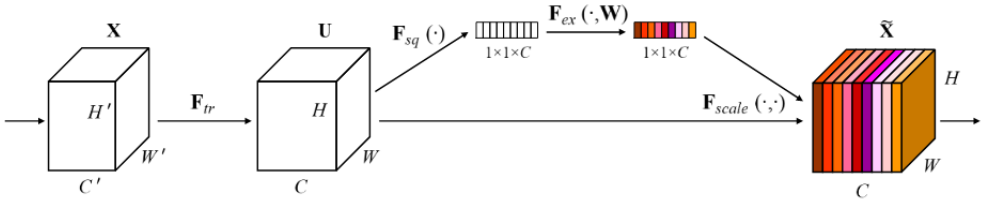
Na veličinu receptivnog polja djeluje arhitektura mreže. Receptivna polja elemenata se povećavaju što je sloj dublje u mreži. Svakim konvolucijskim slojem i slojem sažimanja povećava se receptivno polje izlaza. Ilustracija širenja receptivnog polja prikazana je na slici 3.3.



Slika 3.3: Receptivno polje

3.4. Squeeze and Excite blok

Squeeze and Excite blok [5] je sloj u konvolucijskim neuronskim mrežama koji se koristi kako bi se poboljšalo učenje mreže. Ideja je da se informacije na ulazu u sloj sažmu (engl. *squeeze*), uzbude (engl. *excite*) i dodaju originalnom ulazu u sloj kako bi se naglasile mape značajki koje sadrže bitnije informacije, a potisnule one koje sadrže manje bitne informacije. Idejni prikaz bloka vidi se na slici 3.4.



Slika 3.4: Squeeze and Excite blok [5]

3.4.1. Faza sažimanja

U fazi sažimanja koristi se globalno sažimanje usrednjavanjem (engl. *global average pool*) čime se za svaku mapu značajki na ulazu dobiva po jedna vrijednost koja predstavlja deskriptor te mape značajki. Time se dobiva deskriptor mape značajki u kojem je sadržana informacija koju blok pokušava iskoristiti.

3.4.2. Faza uzbuđivanja

U fazi uzbuđivanja potrebno je pomoću deskriptora odlučiti koje mape značajki će se naglasiti, a koje potisnuti. Za operaciju koja se provodi je bitno da može naučiti koje aktivacije su bitnije od drugih i da može naglašavati i potiskivati više mapa značajki. Operacija koja se koristi sastoji se od dva potpuno povezana sloja sa aktivacijskim funkcijama ReLU i sigmoidom. Operacija je prikazana u jednadžbi 3.5.

$$\mathbf{s} = \sigma(\mathbf{W}_2 * relu(\mathbf{W}_1 * \mathbf{z})) \tag{3.5}$$

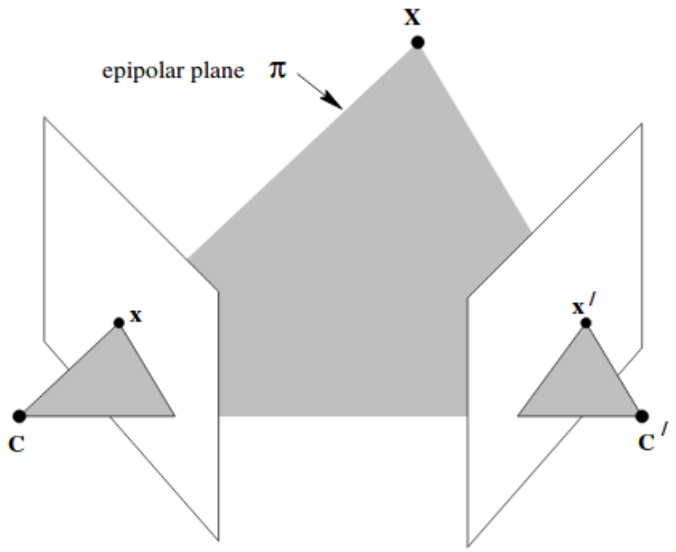
U jednadžbi $\mathbf{s}$ predstavlja konačne težinske faktore kojima se mreža skalira, a $\mathbf{z}$ deskriptore mapa značajki dobivenih u fazi sažimanja. Zadnji korak Squeeze and Excite bloka je množenje mapa značajki sa ulaza u blok dobivenim težinskim faktorima.

4. Stereoskopska rekonstrukcija

U ovom poglavlju bit će opisan problem kojim se rad bavi. Stereoskopska rekonstrukcija je postupak kojim se iz dviju slika iste scene određuje udaljenost točke od kamere. Kako bi to bilo moguće, potrebno je naći odgovarajuće piksele na obje slike.

4.1. Epipolarna geometrija

Epipolarna geometrija [3] je geometrija dvaju pogleda. Pretpostavimo da promatramo točku $\mathbf{X}$ u sceni s dvije kamere. Nazovimo centre kamera $\mathbf{C}$ i $\mathbf{C}'$, a projekcije promatrane točke iz kamera $\mathbf{x}$ i $\mathbf{x}'$. Centri kamera, promatrana točka i njezine projekcije leže u istoj ravnini koju zovemo epipolarna ravnina (slika 4.1).



Slika 4.1: Epipolarna ravnina [3]

Bitni pojmovi vezani uz epipolarnu geometriju su:

– epipol - točka u kojoj pravac koji povezuje centre kamera siječe slikovnu ravninu

– epipolarna ravnina - ravnina na kojoj se nalazi pravac koji povezuje centre kamera

– epipolarni pravac - sjecište epipolarne ravnine i slikovne ravnine, povezuje odgovarajuće piksele na projekcijama scene

Iz epipolarne geometrije proizlazi fundamentalna matrica koja određuje relativan odnos projekcija točke iz scene na slikovne ravnine. Taj odnos je opisan epipolarnim ograničenjem koje je prikazano jednadžbom 4.1.

$$\mathbf{x'^T F x} = 0 \tag{4.1}$$

Esencijalna matrica je poseban oblik fundamentalne matrice koji ima manje stupnjeva slobode. To je čini jednostavnijom za estimaciju.

4.2. Rektifikacija slika

Rektifikacija [8] je postupak kojim se par slika iste scene transformira tako da epipolarni pravci budu paralelni s x-osi. Tim postupkom se osigurava da se korespondentni pikseli na slikama nalaze na istoj y vrijednosti. To znači da je korespondentne piksele dovoljno tražiti u istom retku druge slike čime se značajno smanjuje prostor pretraživanja u stereoskopskoj rekonstrukciji.

Postupak počinje pronalaskom dovoljnog broja parova korespondentnih piksela kako bi se moglo estimirati esencijalnu matricu i epipolove $\mathbf{e}$ i $\mathbf{e}'$. Tada se pronalazi matrica $\mathbf{H}'$ koja epipol $\mathbf{e}'$ preslikava u točku u beskonačnosti $(1, 0, 0)^T$. Nakon toga se pronalazi matrica $\mathbf{H}$ koja minimizira udaljenost $\sum_i d(\mathbf{H}\mathbf{x}_i, \mathbf{H}'\mathbf{x}'_i)$. Na kraju, potrebno je lijevu i desnu sliku transformirati koristeći dobivene matrice.

4.3. Određivanje dubine

Dubina, tj. udaljenost objekta od kamere određuje se pomoću dispariteta [9]. Disparitet je udaljenost odgovarajućih piksela na slikama s dviju kamera koje promatraju scenu. U slučaju rektificiranih slika, disparitet odgovara razlici u horizontalnom položaju. Objekt koji se na lijevoj slici nalazi na položaju (x, y) na desnoj slici nalazi se na položaju $(x - d, y)$. Dubina z može se izračunati prema jednadžbi 4.2. U jednadžbi f predstavlja fokalnu duljinu kamere, a B je udaljenost između centara kamera.

$$z = \frac{fB}{d} \tag{4.2}$$

Može se primjetiti da je udaljenost objekta od kamere obrnuto proporcionalno disparitetu. Ako je disparitet jednak 0, objekt se nalazi u beskonačnosti, a što je disparitet veći, to je objekt bliže kameri.

Rektifikacija slika nije nužna, ali osigurava da se odgovarajući pikseli naalaze na istoj visini na slikama. Time se prostor pretraživanja odgovarajućeg piksela na drugoj slici značajno smanjuje.

4.4. Primjena korespondencijskih metrika

Korespondencijske metrike su značajke izlučene iz podataka kojima se može utvrditi sličnost po nekom kriteriju [1]. Primjenjeno na problem stereoskopske rekonstrukcije, koristeći piksel i njegovo susjedstvo moguće je konstruirati metriku kojom bi se mogli prepoznati pikseli koji odgovaraju istom objektu. Takvu metriku moguće je naučiti koristeći konvolucijske neuronske mreže.

4.5. Trojni gubitak

Pri učenju korespondencijskih metrika konvolucijskim neuronskim mrežama isprva nije jasno kako definirati gubitak jer željeni rezultati ne definiraju izlaze mreže, već odnose među njima za različite ulaze. U takvim slučajevima moguće je primjeniti trojni gubitak [4].

Trojni gubitak se formira usporedbom izlaza iz mreže za tri različita primjera za učenje. Primjeri su izabrani tako da se uz proizvoljni primjer $\mathbf{x}$ izabere pozitivni primjer $\mathbf{x}^+$ i negativni primjer $\mathbf{x}^-$. Ideja je da pozitivni primjer bude što bliže referentnom primjeru, a negativni primjer što dalje.

Gubitak također definira i marginu m koja služi tome da metrika jasnije odvaja pozitivne od negativnih primjera. Time se smanjuje mogućnost greške i pospješuje generalizacija. Izraz za trojni gubitak prikazan je u jednadžbi 4.3.

$$L(\mathbf{x}, \mathbf{x}^+, \mathbf{x}^-) = max(d(\mathbf{x}, \mathbf{x}^+) - d(\mathbf{x}, \mathbf{x}^-) + m, 0) \tag{4.3}$$

Nedostatak trojnog gubitka je što nije direktno vezan uz zadatak mreže pa manji gubitak ne mora značiti bolje rezultate. Ta činjenica može negativno utjecati na proces učenja zato što se u gradijentnom spustu minimizira gubitak čiji minimum se ne mora poklapati s najboljim parametrima za rad modela.

5. Programska izvedba

U ovom poglavlju bit će opisano programsko rješenje, korištene biblioteke i dobiveni rezultati. Također, bit će navedeni problemi koji su se pojavili za vrijeme treniranja modela.

5.1. Korištene biblioteke

Programsko rješenje pisano je u programskom jeziku Python 3.5. U kodu su korištene njegove standardne biblioteke: os, PIL i datetime. Za definiranje arhitekture i treniranje modela korišten je razvojni okvir PyTorch. Za olakšavanje rada s poljima korištena je biblioteka NumPy.

5.1.1. PyTorch

PyTorch je razvojni okvir za rad s tenzorima i dubokim neuronskim mrežama. PyTorch omogućava korištenje grafičke kartice korištenjem tehnologije CUDA kako bi se računske operacije paralelizirale i odradile znatno brže nego na procesoru. Od velike pomoći je i njegova podrška za automatsku diferencijaciju što omogućava vrlo jednostavno računanje gradijenata. Također, PyTorch nudi gotove implementacije slojeva za neuronske mreže, aktivacijskih funkcija, postupaka za računanje gubitka i optimizacijskih postupaka.

5.1.2. Numpy

NumPy je biblioteka za Python koja pomaže pri radu s velikim poljima. Pisan je u programskom jeziku C te koristi biblioteku BLAS (Basic Linear Algebra Subprograms) kako bi bio što efikasniji. Prilikom izračuna koristi se paralelizacija na procesoru. NumPy nudi gotove implementacije računskih operacija nad poljima kao što su skalarni i vektorski umnožak, matrično množenje i računanje normi vektora i matrica.

5.2. Korištene arhitekture

Arhitektura korištena u ovom radu preuzeta je iz rada [8]. Model sadrži 4 konvolucijska sloja sa 64 jezgre veličine 3×3. Na izlazu posljednjeg sloja dobiva se 64-dimenzionalni vektor. Zbog korištenja konvolucijskih slojeva moguće je izvesti ugrađivanje cijele ulazne slike u jednom prolazu. Nakon posljednjeg konvolucijskog sloja svaki 64-dimenzionalni vektor normira se na jediničnu dužinu. Kroz ovaj proces prolaze dvije slike čija se ugrađivanja zatim uspoređuju redak po redak. Za svaki piksel lijeve slike bira se piksel desne slike koji mu je najsličniji prema naučenoj metrici. Disparitet se dobiva kao razlika položaja odgovarajućih piksela na x-osi.

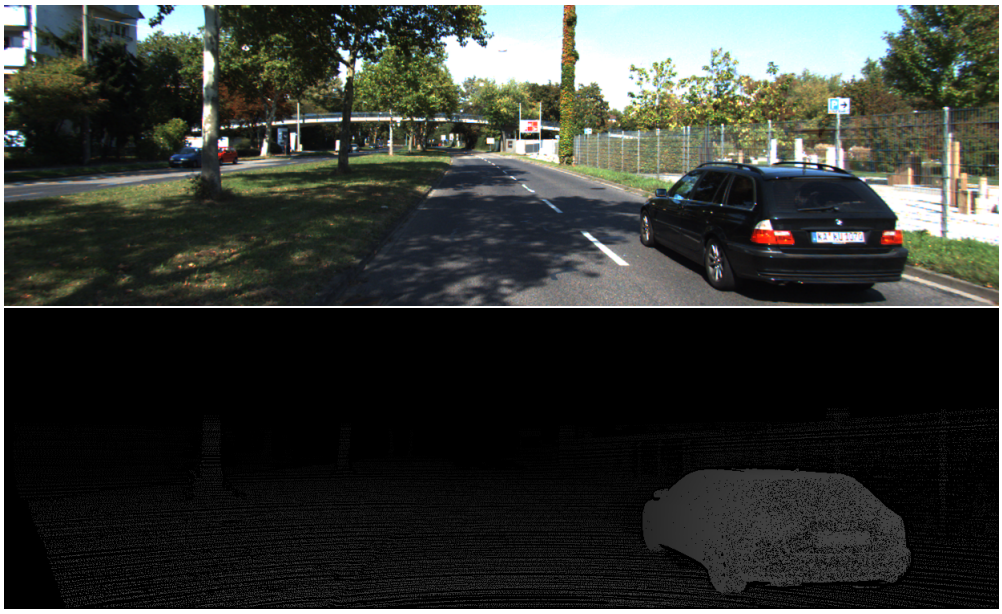
Zadatak modela je naučiti metriku tako da odgovarajući pikseli nakon ugrađivanja budu što sličniji. Pri učenju koristi se trojni gubitak opisan u poglavlju 4.5. Kako bi gubitak bio primjenjiv, kroz konvolucijske slojeve je potrebno provući tri primjera. Jedan od tih primjera je referentni primjer, drugi je njegov odgovarajući (pozitivni) primjer, a treći je negativni primjer. Nakon ugrađivanja primjera gubitak se računa prema jednadžbi 4.3.

Uz opisanu arhitekturu, korištene su i dvije varijacije. U jednoj je promijenjen način usporedbe pikesla nakon ugrađivanja. U originalnom modelu ugrađivanja se uspoređuju iterativno rotacijama stupaca desne slike i usporedbom stupaca koji se trenutno preklapaju. Pri tome se događa da se uspoređuju ugrađivanja sa suprotnih strana slika koji izlaze van okvira maksimalnog dispariteta i, u slučaju visoke sličnosti, uzrokuju pogreške u evaluaciji. Varijacija arhitekture koja pokušava riješiti taj problem umjesto rotacije stupaca desne slike koristi logički posmak (engl. *logical shift*) pri čemu praznina koja nastane posmakom stupaca neće biti popunjena stupcem koji je "ispao" nego će tenzor biti nadopunjen nulama. Druga varijacija arhitekture sadrži promjenu iz prve varijacije, ali dodatno u proces ugrađivanja prije posljednje konvolucije ubacuje Squeeze and Excite blok opisan u poglavlju 3.4. Time se pokušalo olakšati učenje metrike mogućnošću isticanja važnijih značajki i dodatnom fleksibilnošću zbog većeg broja parametara.

5.3. Podatkovni skup

Za učenje je korišten podatkovni skup KITTI 2015 [7]. Skup sadrži 200 scena za treniranje koje uključuju par slika i mapu dispariteta. Slike prikazuju scene vožnje u gradu. Dispariteti su mjereni laserskim uređajem, a mape dispariteta su dodatno obogaćene koristeći 3D modele automobila. Stereo parovi su kalibrirani i rektificirani. Zbog fo-

kusa na automobilima i scena iz vožnje, podatkovni skup je pogodan za učenje modela koji bi se koristili u autonomnoj vožnji. Mape dispariteta nisu potpune, na mjestima gdje disparitet nije poznat nalazi se nula. Isti disparitet odgovara i točkama u beskonačnosti, tako da skup podataka nije prigodan za učenje stereoskopske rekonstrukcije na nebu ili vrlo udaljenim objektima. Jedan primjer iz skupa prikazan je na slici 5.1.



Slika 5.1: Primjer iz podatkovnog skupa KITTI 2015. Gore je prikazana lijeva slika para, a ispod mapa dispariteta.

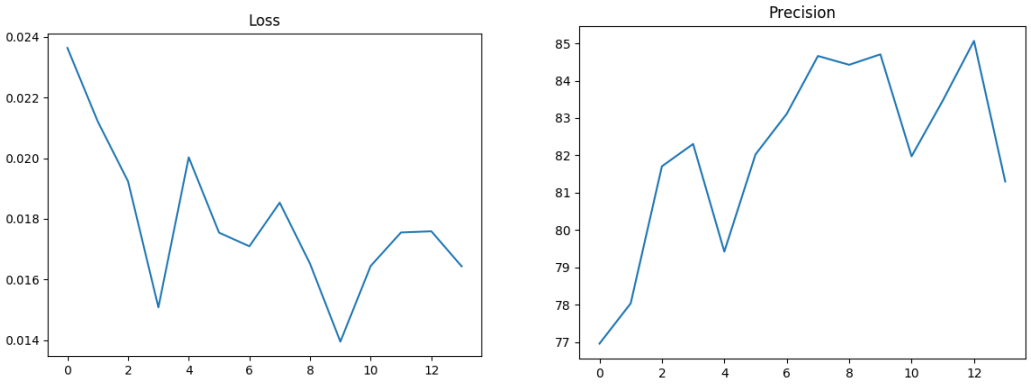
5.4. Rezultati učenja

U sklopu rada ukupno je učeno 7 modela. Postignute preciznosti svih modela vide se u tablici 5.1. Korištena stopa učenja je $1e^{-3}$ i margina u trojnom gubitku 0.1, osim u modelima gdje je navedeno drugačije. Korišteno je po 20 slika u skupu za učenje i skupu za testiranje.

Za početak, naučen je osnovni model. Dobiveni rezultati vide se na slici 5.2. Iz grafova se primjećuje da gubitak i preciznost poskakuju što vodi na preveliku stopu učenja ili premali skup za učenje, ali model daje relativno dobre rezultate.

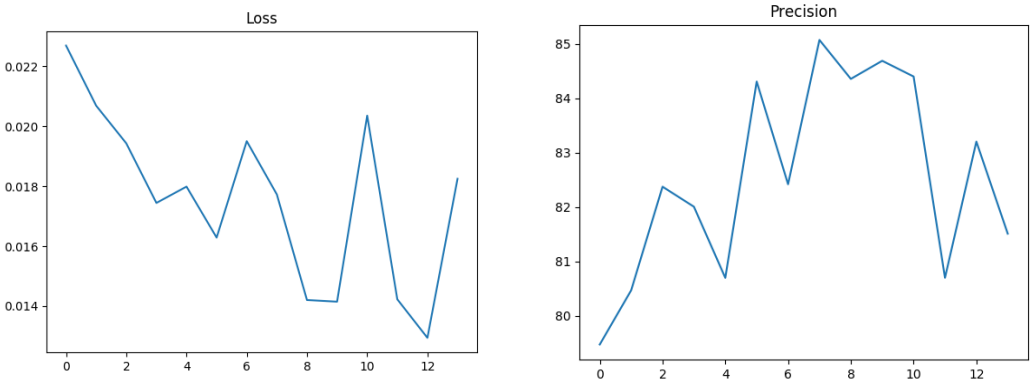
Nakon toga pokrenuto je učenje na arhitekturi s izmijenjenim uspoređivanjem piksela. Očekivani su nešto bolji rezultati nego na osnovnom modelu. Kao što se vidi sa slike 5.3 i iz tablice, razlika je zanemariva.

Model s uključenim Squeeze and Excite blokom također ne postiže bolje rezultate, što je vidljivo na slici 5.4. Primjećuje se još veće poskakivanje u gubitku nego na



(a) **(b)**

Slika 5.2: Rezultati na osnovnom modelu: (a) gubitak na skupu za testiranje, (b) preciznost na skupu za testiranje



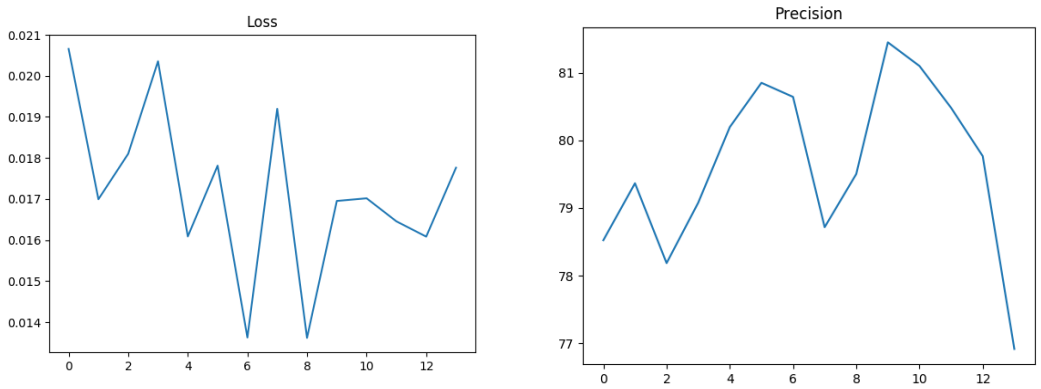
(a) **(b)**

Slika 5.3: Rezultati na modelu s izmjenjenim uspoređivanjem piksela: (a) gubitak na skupu za testiranje, (b) preciznost na skupu za testiranje

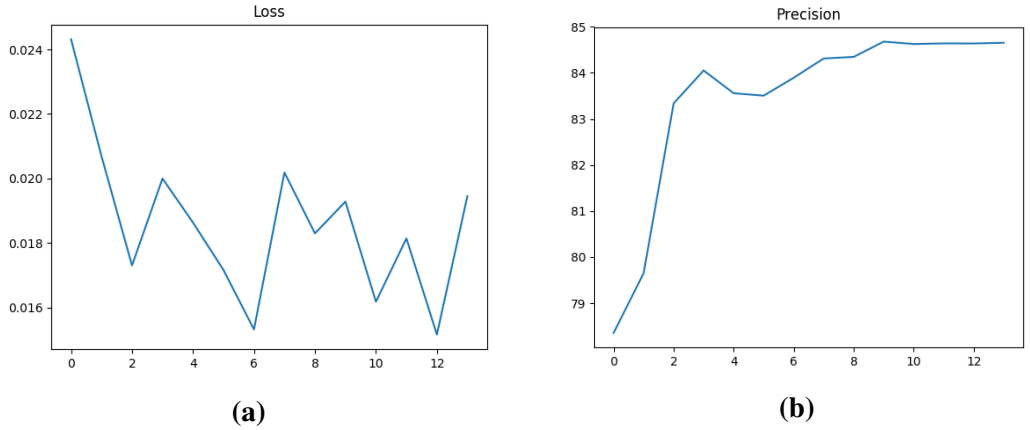
osnovnom modelu što se može objasniti većom ekspresivnošću modela zbog dodatnih slojeva.

Pokušano je opet učiti model s izmijenjenim uspoređivanjem piksela, ovaj put uz promjenu stope učenja na $1e^{-5}$ na sedmoj epohi. Grafovi se mogu vidjeti na slici 5.5 Na preciznosti se primjećuje ljepše ponašanje, ali gubitak je nestabilan.

Pokušano je mijenjati veličinu margine u trojnom gubitku. Margina je povećana na 0.5 i smanjena na 0.05 kako bi se promatrao utjecaj na učenje modela. Grafovi za model sa povećanom marginom vide se na slici 5.6, a za model sa smanjenom marginom na 5.7. Može se primjetiti da je veličina margine utjecala na gubitak, što je i očekivano. Primjećuje se razlika u ponašanju preciznosti, ali razlika najboljih epoha je minimalna.



Slika 5.4: Rezultati na modelu sa Squeeze and Excite blokom: (a) gubitak na skupu za testiranje, (b) preciznost na skupu za testiranje

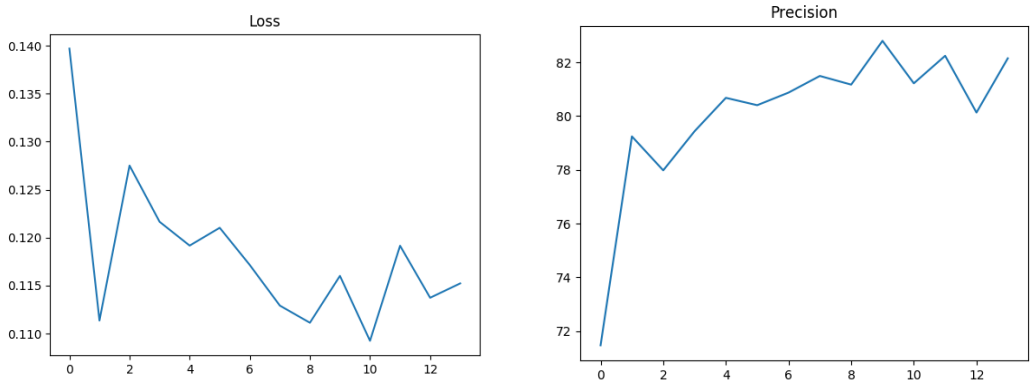


Slika 5.5: Rezultati na modelu sa promijenjenom stopom učenja: (a) gubitak na skupu za testiranje, (b) preciznost na skupu za testiranje

Konačno, pokrenut je model s izmijenjenim uspoređivanjem piksela s dvostruko većim skupom podataka za učenje kako bi se vidio utjecaj veće količine podataka na proces učenja i konačan rezultat. Grafovi su prikazani na slici 5.8. Može se primjetiti da je preciznost malo iznad prijašnjih rezultata, što pokazuje da bi imalo smisla dalje povećavati skup za učenje.

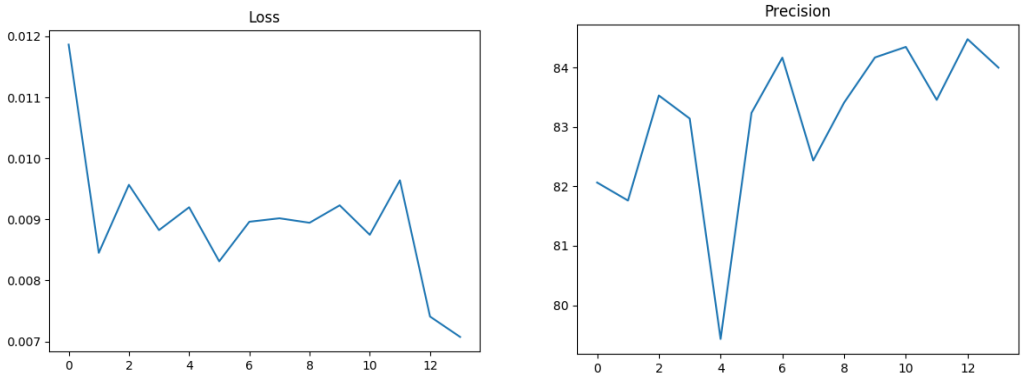
5.5. Utjecaj veličine skupa podataka za treniranje

Kao jedan od eksperimenata, model je učen na različitim veličinama skupova podataka. Korišten je model s izmijenjenim uspoređivanjem piksela i skupovi podataka veličina 1, 2, 5, 10 i 20 slika. Svaki veći skup primjera za učenje u sebi je sadržavao onaj



(a) **(b)**

Slika 5.6: Rezultati na modelu sa povećanom marginom trojnog gubitka: (a) gubitak na skupu za testiranje, (b) preciznost na skupu za testiranje



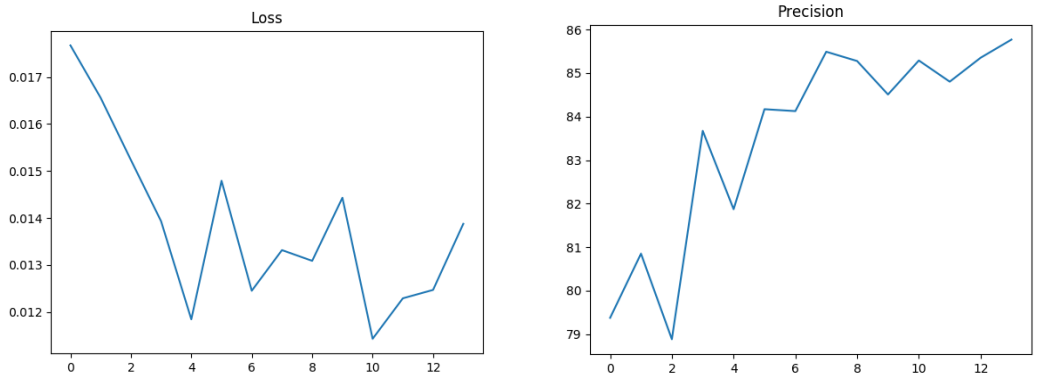
(a) **(b)**

Slika 5.7: Rezultati na modelu sa smanjenom marginom trojnog gubitka: (a) gubitak na skupu za testiranje, (b) preciznost na skupu za testiranje

manji od njega, a evaluacija modela odvijala se na istom skupu podataka za testiranje. Korištena stopa učenja je $1e^{-5}$ Rezultati su prikazani u tablici 5.2. Ovisnost preciznosti o veličini skupa podataka za učenje prikazana je na slici 5.9. Kretanje preciznosti po epohama učenja prikazano je na slici 5.10. Na slikama je vidljivo kako poboljšanje uz povećanje skupa podataka za učenje raste sublinearno.

5.6. Preciznost u odnosu na točni disparitet i semantičku kategoriju piksela

U nastavku su opisani rezultati eksperimenata u kojima se provjeravala ovisnost preciznosti modela o točnom disparitetu i semantičkoj kategoriji piksela. Pri provjeri je



(a) **(b)**

Slika 5.8: Rezultati na modelu s većim skupom podataka za učenje: (a) gubitak na skupu za testiranje, (b) preciznost na skupu za testiranje

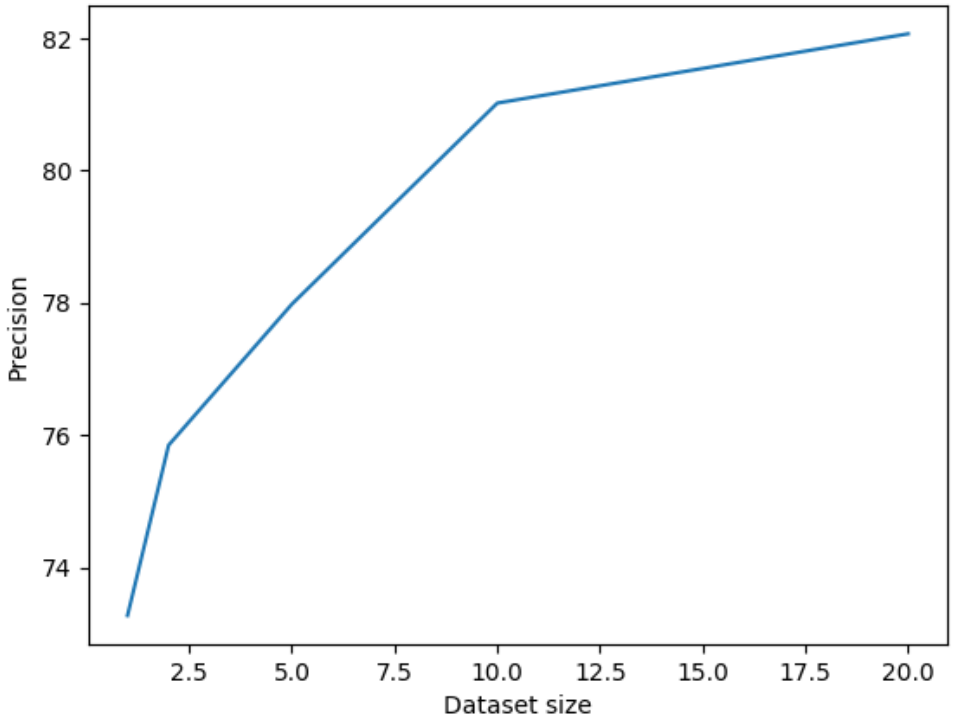
Tablica 5.1: Rezultati modela

Model	**Preciznost**
Osnovni model	85.065%
Model s izmijenjenim uspoređivanjem piksela	85.076%
Model sa Squeeze and Excite blokom	81.446%
Model s izmijenjenom stopom učenja	84.679%
Model sa smanjenom marginom u gubitku	84.475%
Model sa povećanom marginom u gubitku	84.799%
Model s povećanim podatkovnim skupom	85.771%

za testiranje korišten skup podataka od 180 slika, a preciznost je provjerena na modelu iz pokusa sa veličinama skupova podataka koji je učen na 20 slika.

Na slici 5.11 prikazana je ovisnost preciznosti o točnom disparitetu. Može se primjetiti da model radi vrlo dobro u rasponu dispariteta od 10 do 60. Za vrlo male disparitete radi loše, a za disparitete iznad 60 preciznost počinje naglo opadati. Može se pretpostaviti da se model tako ponaša zbog nedostatka podataka u skupu za treniranje koji sadrže objekte koji su vrlo bliski kamerama i ibjekte koji su vrlo daleko od kamera.

Na slici 5.12 prikazana je ovisnost preciznosti o semantičkoj kategoriji piksela. Može se vidjeti da model radi podjednako dobro za većinu kategorija, a ističu se građevine, nebo i vozila. U sva tri slučaja može se pretpostaviti da se radi o jednolikim ili refleksivnim površinama za koje je teško odrediti točan odgovarajući piksel uz korištene tehnike. Rezultat bi se mogao popraviti korištenjem zaglađivanja dispariteta u



Slika 5.9: Ovisnost preciznosti o veličini skupa podataka za učenje

odnosu na okolne disparitete. Na taj način bi se postigli bolji rezultati na jednolikim površinama.

5.7. Izazovi kod treniranja modela

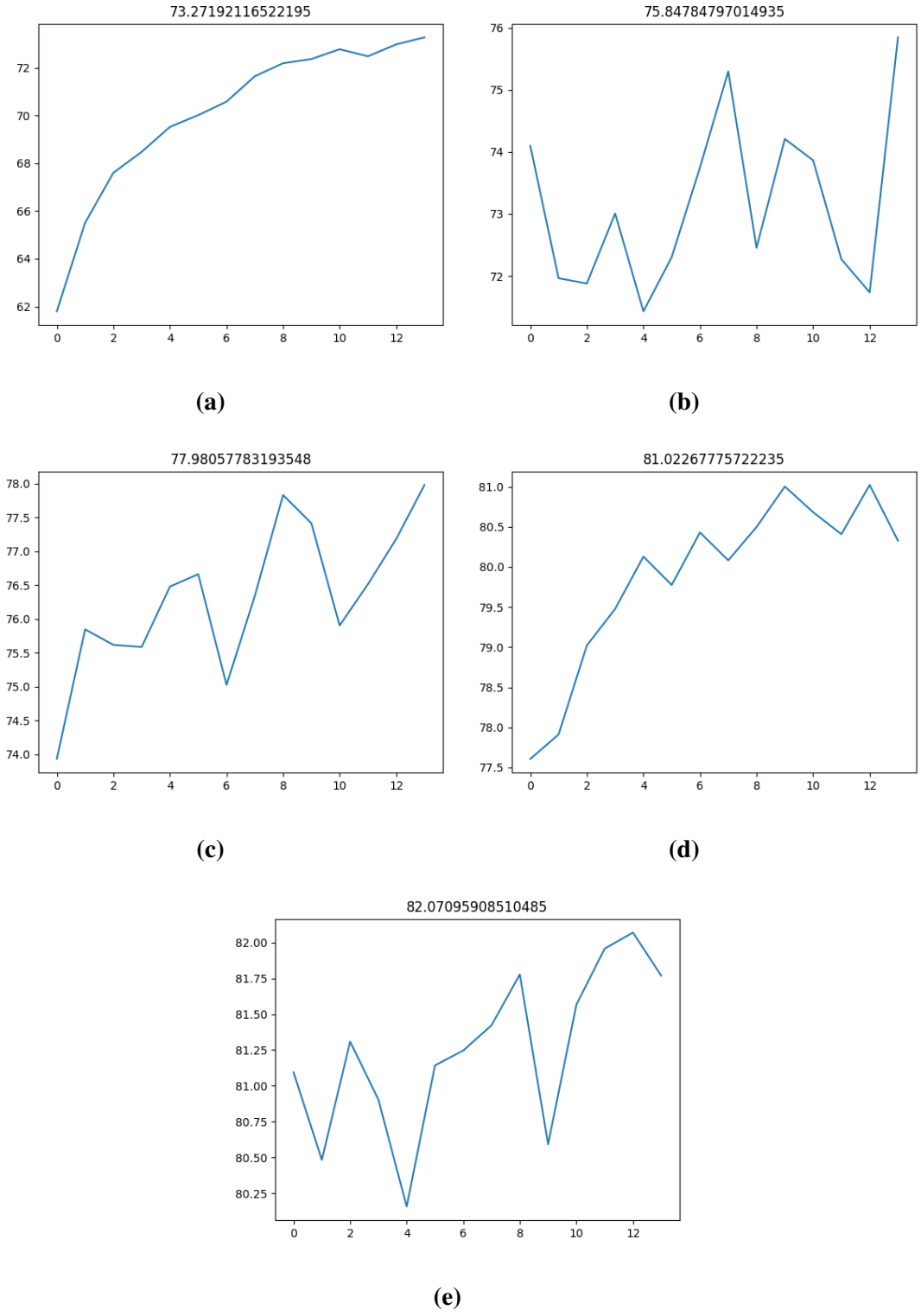
Prilikom učenja modela javilo se nekoliko problema koji su otežali rad. Prvi takav problem je trajanje postupka učenja na dostupnom *hardwareu*, čime je bila određena i veličina skupa za učenje. Na skupu za učenje od 20 scena, trajanje učenja je bilo

Tablica 5.2: Rezultati učenja s različitim veličinama skupa za učenje

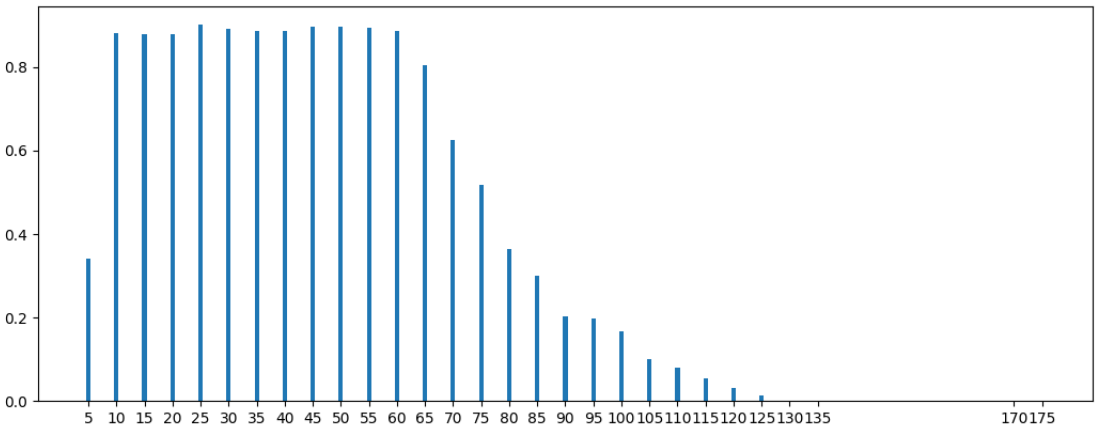
Veličina skupa za učenje	**Preciznost**
1	73.272%
2	75.848%
5	77.981%
10	81.023%
20	82.071%

između 7 i 8 sati. Korištenje većeg skupa za učenje na istom broju modela bilo bi vremenski vrlo skupo.

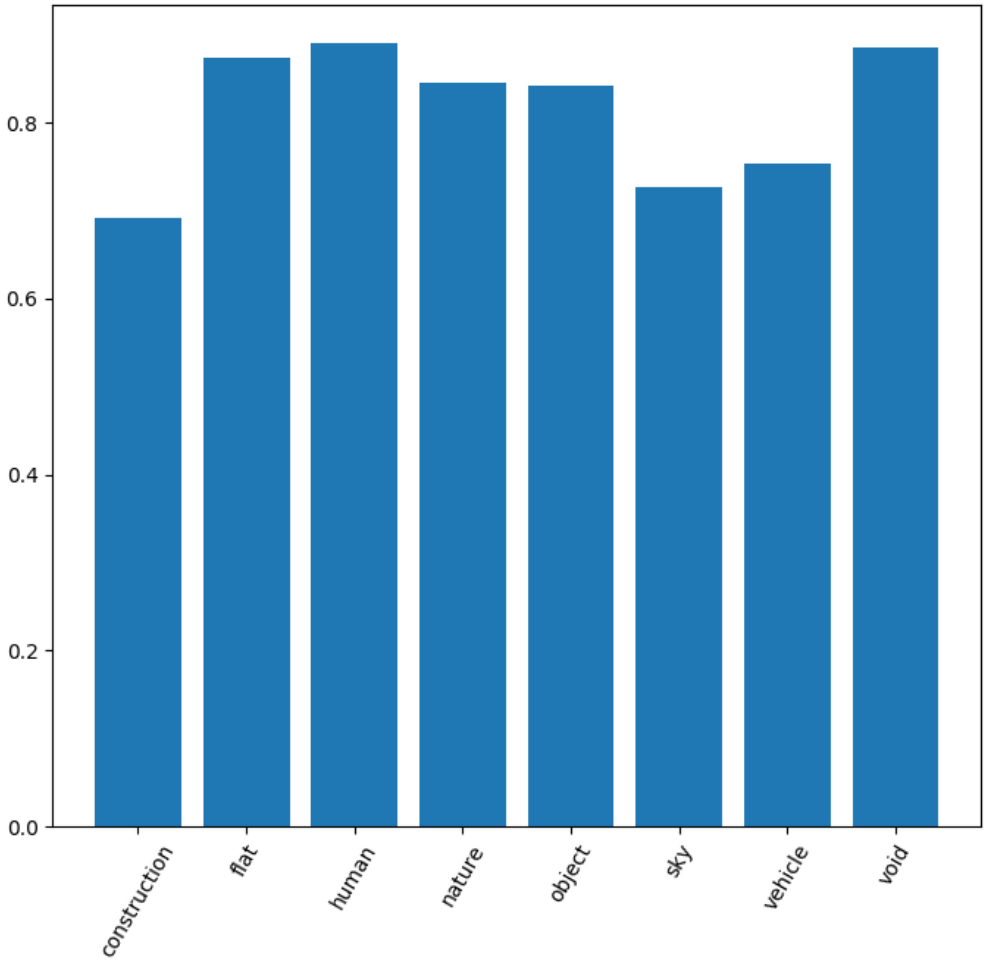
Drugi problem je memorijska složenost predobrade podataka u skupu za učenje. Proces uključuje traženje dispariteta u mapi koji se mogu iskoristiti kao primjeri za učenje i generiranje okana veličine 9×9 za odgovarajuće piksele na lijevoj i desnoj slici i jedan dodatni kao negativni primjer. Vrlo brzo se pojavi mnogo redundantnih podataka u oknima te skup za učenje višestruko naraste. Podataka je bilo previše da bi stali u radnu memoriju dostupnog računala, a zapisivanje u trajnu memoriju nije bilo moguće zbog prevelikog broja malih datoteka. Problem se zaobišao predobradom podataka po potrebi čime se uštedilo na memorijskim zahtjevima na račun procesorskog vremena, tako da su se podaci obrađivali po potrebi. Dodatni nedostatak ove odluke je bilo ograničavanje nasumičnog odabira primjera tijekom učenja jer su u svakom trenu dostupni bili samo primjeri za učenje nastali iz jedne scene.



Slika 5.10: Rezultati uz različite veličine skupova za testiranje: (a) 1 slika (b) 2 slike (c) 5 slika (d) 10 slika (e) 20 slika



Slika 5.11: Ovisnost preciznosti o točnom disparitetu



Slika 5.12: Ovisnost preciznosti o semantičkoj kategoriji piksela

6. Zaključak

Stereoskopska rekonstrukcija je jedan od izazovnijih problema u računalnom vidu. Rekonstrukcije 3D scene pomoću slika iz dvaju različitih kutova je proces s kojim se čovjek svakodnevno susreće i s lakoćom ga rješava, ali računalima to predstavlja mnogo teži zadatak. Pojavom dubokog učenja računalni vid se značajno razvio i danas duboki modeli rješavaju mnoge probleme u računalnom vidu.

Korištenje korespondencijskih metrika u računalnom vidu omogućava preslikavanje podataka u visokodimenzionalni vektorski prostor. Značajnost toga je u činjenici da je takvim preslikavanjem računalima znatno lakše uspoređivati podatke i učiti kako ih razlikovati.

Trojni gubitak se nadovezuje na korištenje korespondencijskih metrika tako da definira kakav odnos među podacima je potrebno naučiti. Time je moguće usmjeriti postupak učenja u željenom smjeru bez definiranja konkretnog izlaza koji očekujemo.

U radu je prikazan postupak primjene trojnog gubitka i korespondencijskih metrika u stereoskopskoj rekonstrukciji. Opisano je nekoliko različitih modela i prikazana je njihova mogućnost rješavanja problema stereoskopske rekonstrukcije. Čak i na maloj količini podataka za učenje, model je pokazao relativno visok stupanj preciznosti.

U budućem radu je moguće ispitati model na većem skupu podataka i ekstenzivnije ispitivanje utjecaja vrijednosti hiperparametara. Vrlo zanimljivo bi bilo proučiti utjecaj različitih arhitektura na efikasnost korespondencijske metrike. Također, preciznost modela mogla bi se poboljšati korištenjem neke od tehnika globalnog zaglađivanja dispariteta.

LITERATURA

[1] Qiong Cao, Yiming Ying, i Peng Li. Similarity metric learning for face recognition. U *Computer Vision (ICCV), 2013 IEEE International Conference on*, stranice 2408–2415. IEEE, 2013.

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

[3] R. I. Hartley i A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, 2004.

[4] Elad Hoffer i Nir Ailon. Deep metric learning using triplet network. *CoRR*, abs/1412.6622, 2014. URL `http://arxiv.org/abs/1412.6622`.

[5] Jie Hu, Li Shen, i Gang Sun. Squeeze-and-excitation networks. *CoRR*, abs/1709.01507, 2017. URL `http://arxiv.org/abs/1709.01507`.

[6] Diederik P. Kingma i Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL `http://arxiv.org/abs/1412.6980`.

[7] Moritz Menze i Andreas Geiger. Object scene flow for autonomous vehicles. U *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, stranice 3061–3070, 2015.

[8] Marin Oršić. Učenje korespondencijske metrike za gustu stereoskopsku rekonstrukciju. 2017. URL `http://www.zemris.fer.hr/~ssegvic/project/pubs/orsic17ms.pdf`.

[9] Jure Zbontar i Yann LeCun. Computing the stereo matching cost with a convolutional neural network. U *Proceedings of the IEEE conference on computer vision and pattern recognition*, stranice 1592–1599, 2015.

Učenje dubokih korespondencijskih metrika trojnim gubitkom

Sažetak

Stereoskopska rekonstrukcija je bitno polje računalnog vida. Najbitniji dio problema je pronalazak korespondentnih piksela u dvjema slikama iste scene. To je moguće ostvariti koristeći korespondencijske metrike. Pri učenju korespondencijskih metrika od velike pomoći je trojni gubitak. Definiranjem odnosa među podacima moguće je naučiti metriku koja olakšava pronalazak korespondentnih piksela. U okviru rada opisan je problem stereoskopske rekonstrukcije i primjena korespondencijskih metrika i trojnog gubitka na njega. Opisan je i testiran model za rješavanje opisanog problema.

Ključne riječi: stereoskopska rekonstrukcija, korespondencijske metrike, trojni gubitak, konvolucijske mreže

Learning deep correspondence metrics with triplet loss

Abstract

Stereoscopic reconstruction is an important field in computer vision. The most important part of the problem is finding the corresponding pixels in two images of the same scene. It is possible to achieve that using correspondence metrics. Triplet loss is a big help in training correspondence metrics. By defining the data relations it is possible to train a metric which makes finding corresponding pixels easier. in this thesis, the stereoscopic reconstruction problem is described. Application of correspondence metrics and triplet loss to it are explained. A model for solving the problem is described and tested.

Keywords: stereoscopic reconstruction, correspondence metrics, triplet loss, convolutional networks