SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 2793

Eksperimentalno vrednovanje metoda za monokularnu predikciju dubine

Dominik Špiljak

Zagreb, srpanj 2022.

Zahvaljujem se prof. dr. sc. Siniši Šegviću na pruženoj pomoći i savjetima tijekom izrade ovog rada. Također, zahvaljujem se Pauli, roditeljima, obitelji i prijateljima na podršci tijekom cijelog studija.

SADRŽAJ

1. Uvod

Razumijevanje scene važno je područje računalnog vida čiji je cilj iz ulazne slike izlučiti informacije o sceni koristeći metode kao što su: procjena dubine, procjena normala, semantička segmentacija, itd. Kroz navedene metode, moguće je stvoriti dodanu vrijednost kroz različite primjene u interakciji s tom scenom. Primjer takve primjene su autonomni dronovi kod kojih često nije moguće ugraditi sve potencijalno korisne senzore zbog ograničenja težine. U takvim slučajevima, performanse cijelog sustava su uvjetovane performansama metoda za razumijevanje scene računalnim vidom. Ovakvi sustavi često zahtijevaju zaključivanje u stvarnom vremenu. Zato je potrebno da metode razumijevanja scene uz točnost zadovoljavaju i zahtjeve brzine zaključivanja. Na primjer, ako autonomni dron ne donosi zaključke dovoljno brzo, može doći do oštećenja cijelog sustava uslijed sudara. Zato je razumijevanje scene vrlo atraktivno i brzo rastuće područje računalnog vida. Do nedavno, konvolucijski modeli dominirali su tim područjem te su doživjeli veliki napredak predstavljanjem preskočnih veza i U-Net arhitekture. Koristeći preskočne veze, postalo je vrlo lagano trenirati vrlo duboke modele, a koristeći U-Net arhitekturu, moguće je povećati receptivno polje i brzinu zaključivanja modela.

U posljednje vrijeme, područje je doživjelo veliki napredak zahvaljujući modelima koji se temelje na mehanizmu pažnje. Ta vrsta modela je originalno predstavljena u području obrade prirodnog jezika, ali u posljednje vrijeme pronalazi svoje mjesto u području računalnog vida i sve se više prilagođava tom problemu. Ti modeli imaju globalno receptivno polje i ne zahtijevaju smanjivanje rezolucije mape značajki kao što zahtijevaju konvolucijski modeli te tako minimaliziraju gubitak informacija pri unaprijednom prolasku kroz model. No, usprkos visokih performansi, takvi modeli zahtijevaju veliku količinu podataka tijekom treniranja kako bi pokazali bolje performanse od konvolucijskih modela i imaju veliki memorijski otisak.

U ovom radu razmatrat ćemo metode za monokularnu predikciju dubine. Predikcija dubine omogućuje rekonstrukciju trodimenzionalne strukture iz dvodimenzionalnog ulaza te tako pruža veliku količinu informacije koja doprinosi razumijevanju scene i

donošenju odluka cijelog sustava. Kao rezultat te metode, dobivamo RGBD sliku koja u jednoj dimenziji sadržava informaciju od dubini svakog piksela. Koristeći taj rezultat, moguće je vizualizirati oblak točaka u trodimenzionalnom prostoru i dobiti dobru ideju o strukturi scene.

Kroz ovaj rad, usporedit ćemo tri duboka modela s različitim svojstvima. Prvi model je MIMO-UNet [3] koji je originalno predstavljen za problem odmućivanja slika i koristi modificiranu U-Net arhitekturu s više ulaza i izlaza te tako obrađuje sliku na različitim rezolucijama. Drugi model je Ladder DenseNet [9] koji je originalno predstavljen za problem semantičke segmentacije i koristi predtrenirane DenseNet modele te ljestvičasto naduzorkovanje kako bi smanjio memorijski otisak i ubrzao učenje koristeći manji broj parametara. Treći model je DPT [17] koji je originalno predstavljen za problem semantičke segmentacije i predikcije dubine i koristi slojeve pažnje koji predstavljaju stanje tehnike u području računalnog vida, ali i ostalim područjima dubokog učenja.

Modeli su trenirani i evaluirani na skupu podataka NYU depth V2 [14]. Za svaki model evaluirat ćemo performanse modela, memorijski otisak pri učenju te brzinu zaključivanja.

2. Komponente dubokih modela

2.1. Potpuno povezani sloj

Potpuno povezani sloj je osnovni gradivni blok unaprijednih potpuno povezanih modela. U prvom koraku, na izlaze prethodnog sloja $k-1$ primjenjujemo afinu transformaciju

$$s_k = h_{k-1} W_k^T + b_k. \tag{2.1}$$

U drugom koraku na izlaz s_k primjenjujemo aktivacijsku funkciju $h_k = f(s_k)$. Neka je h_{k-1} izlazna matrica značajki prošlog sloja dimenzija $N \times H_{in}$, a željena izlazna dimenzija značajki H_{out}, gdje je N veličina mini-grupe i H_{in} izlazna dimenzija sloja $k-1$. Tada je W matrica dimenzija $H_{out} \times H_{in}$, a b vektor duljine H_{out}. Na rezultat s_k dimenzija $N \times H_{out}$ tada primjenjujemo neku nelinearnu aktivacijsku funkciju koje će biti objašnjene u 2.4.
Tijekom unatražnog prolaza, potrebno je odrediti gradijente potpuno povezanog sloja. Parcijalnu derivaciju po W_k računamo kao:

$$\frac{\partial \mathcal{L}}{\partial W_k} = \left[\frac{\partial \mathcal{L}_k}{\partial s_k}\right]^T h_{k-1}^T. \tag{2.2}$$

Parcijalna derivacija po b_k iznosi:

$$\frac{\partial \mathcal{L}}{\partial b_k} = \frac{\partial \mathcal{L}_k}{\partial s_k}. \tag{2.3}$$

Parcijalna derivacija po h_{k-1}:

$$\frac{\partial \mathcal{L}}{\partial h_{k-1}} = \frac{\partial \mathcal{L}_k}{\partial s_k} W_k. \tag{2.4}$$

Prednost potpuno povezanog sloja nad konvolucijskim slojem je globalno receptivno polje zato što svaki izlaz u izlaznoj matrici sloja h_k ovisi o svim ulazima h_{k-1}. U posljednje vrijeme, potpuno povezani sloj je većinom korišten u modelima koji se temelje na mehanizmu pažnje.

2.2. Konvolucijski sloj

Dva velika nedostatka potpuno povezanih slojeva su veliki broj parametara i neekvivarijantnost na translaciju. Ti nedostatci najviše dolaze do izražaja u području računalnog vida. Ulazne slike mogu biti velikih dimenzija, ponekad čak i 1024×1024 piksela. Ako želimo klasificirati RGB sliku u jednu od 10 klasa i koristimo samo jedan potpuno povezani sloj, već imamo $1024 \times 1024 \times 3 \times 10 = 31.5$ milijuna parametara koje je potrebno trenirati. Također, ako je objekt koji klasificiramo na ulaznoj slici translatiran, moguće je da težine za taj dio slike nisu prilagođene za klasifikaciju tog objekta jer se taj slučaj nije pojavljivao u skupu za treniranje i to će rezultirati pogrešnom klasifikacijom. Te probleme rješava konvolucijski sloj koji koristi manji broj parametra koji su dijeljeni čime ostvarujemo ekvivarijantnost na translaciju.

Konvolucijski sloj se sastoji od jezgre W_k male rezolucije, najčešće 3×3 ili 1×1, i pomaka b_k. Izlaz sloja se računa kao rezultat konvolucije između ulazne mape značajki i jezgre tako da se jezgra ponaša kao pomični prozor te se računa skalarni produkt jezgre i odgovarajućeg dijela ulazne mape značajki. U slučaju obrade 2D slike, jezgra W_k je tenzor 4. reda dimenzija $C_{out} \times C_{in} \times K_1 \times K_2$ gdje su C_{out} izlazni broj kanala, C_{in} izlazni broj kanala prethodnog sloja $k-1$ te $K_1 \times K_2$ rezolucija jezgre, a b vektor duljine C_{out}. Uzmimo slučaj kada se ulazna slika nadopunjuje s $\lfloor \frac{K_1}{2} \rfloor$ nula s lijeve i desne strane te s $\lfloor \frac{K_2}{2} \rfloor$ nula s gornje i donje strane kako bi izlazna mapa značajki bila istih dimenzija te neka je $C_{in} = 1$. Jednu vrijednost izlazne mape značajki q konvolucijskog sloja računamo kao:

$$q_{ij} = b + \sum_{u=1}^{K_1} \sum_{v=1}^{K_2} p_{i-o_{k_1}+u, j-o_{k_2}+v} \times w_{u,v} \tag{2.5}$$

gdje su p ulazna mapa značajki i $o_{k_l} = \lfloor \frac{K_l}{2} \rfloor + 1$.

Prilikom računanja gradijenata ulaza i težina potrebno je primijetiti da ne utječu svi ulazi na sve izlaze kao u slučaju potpuno povezanog sloja. Zato će se gradijenti po ulazu za jednu vrijednost računati koristeći jezgru, pomak i samo dio gradijenata po izlazu. Točnije, prilikom unaprijednog prolaza, ulazna vrijednost p_{ij} će utjecati samo na izlazne vrijednosti $q_{i+u,j+v}$ gdje je $u \in -\lfloor \frac{k_1}{2} \rfloor ... \lfloor \frac{k_1}{2} \rfloor$, a $v \in -\lfloor \frac{k_2}{2} \rfloor ... \lfloor \frac{k_2}{2} \rfloor$. Koristeći tu informaciju lagano je odrediti da vrijedi:

$$\frac{\partial q_{i+u,j+v}}{\partial p_{ij}} = w_{o_{k_1}-u, o_{k_2}-v} \tag{2.6}$$

gdje je izlazna mapa značajki q nadopunjena s $\lfloor \frac{K_1}{2} \rfloor$ nula s lijeve i desne strane te s $\lfloor \frac{K_2}{2} \rfloor$ nula s gornje i donje strane. Ako fiksiramo vrijednosti u i v te pogledamo kako

jedan element jezgre utječe na izlaznu mapu značajki, dobit ćemo:

$$q_{ij} = p_{i-o_{k_1}+u,j-o_{k_2}+v} \times w_{u,v}, \forall i, j \tag{2.7}$$

Sada je lagano odrediti $\frac{\partial q_{i,j}}{\partial w_{u,v}}$:

$$\frac{\partial q_{i,j}}{\partial w_{u,v}} = p_{i-o_{k_1}+u,j-o_{k_2}+v} \tag{2.8}$$

U radu se također koristi transponirana konvolucija. Obična konvolucija često smanjuje rezoluciju značajki te povećava broj kanala kako bi se povećalo receptivno polje modela. Pri predikciji dubine, često uspoređujemo odnose objekata u sceni, to jest, objekti koji su bliži nam mogu biti referenca o daljini nekog drugog objekta. Ta informacija je pogotovo bitna ako su objekti iste veličine u stvarnosti pa je usporedbom veličine oba objekta moguće odrediti njihovu udaljenost. Zato nam veće receptivno polje konvolucija može značajno poboljšati kvalitetu predikcije. Transponirana konvolucija radi suprotno, povećava rezoluciju mapa značajki te smanjuje njihov broj, što se često radi kod modela za gustu predikciju čiji izlaz treba biti iste rezolucije kao i ulaz. Operacija transponirane konvolucije je ista operacija koja se obavlja prilikom računanja gradijenta obične konvolucije s obzirom na ulaznu mapu značajki.

2.3. Sloj sažimanja

Smanjujući rezoluciju, povećava se receptivno polje modela te se smanjuje memorijski otisak tijekom učenja. Za razliku od konvolucije koja često i povećava broj mapa značajki i ima svoje težine, nakon sloja sažimanja broj mapa značajki se ne mijenja i sloj nema težine. Najčešće korištene verzije slojeva sažimanja su sažimanje maksimalnom vrijednosti i sažimanje prosječnom vrijednosti. Neka su sažimanja dvodimenzionalna i neka se obavljaju nad isječcima mapa značajki veličine $K_1 \times K_2$ te neka je ulazna mapa značajki h. Tada se jedna izlazna vrijednost sloja sažimanja maksimalnom vrijednosti računa kao:

$$s_{i,j} = \max(h_{K_1 \cdot i, K_2 \cdot j}, h_{K_1 \cdot i, K_2 \cdot j+1}, \ldots, h_{K_1 \cdot i+K_1-1, K_2 \cdot j+K_2-1}), \tag{2.9}$$

ako je maksimalna vrijednost $h_{m,n}$, tada je gradijent s obzirom na ulaz:

$$\frac{\partial s_{i,j}}{\partial h_{k,l}} = \begin{cases} 1 & \text{ako je } k = m \text{ i } l = n, \\ 0 & \text{inače.} \end{cases} \tag{2.10}$$

Jedna izlazna vrijednost sloja sažimanja srednjom vrijednosti računa se kao:

$$s_{i,j} = \frac{\sum_{k=0}^{K_1-1} \sum_{l=0}^{K_2-1} h_{K_1 \cdot i+k, K_2 \cdot j+l}}{K_1 \times K_2}. \tag{2.11}$$

a gradijent s obzirom na ulaz se računa kao:

$$\frac{\partial s_{i,j}}{\partial h_{k,l}} = \begin{cases} \frac{1}{K_1 \cdot K_2} & \text{ako je } k \in [K_1 \cdot i, K_1 \cdot i + K_1 - 1] \\ & \text{i } l \in [K_2 \cdot j, K_2 \cdot j + K_2 - 1], \\ 0 & \text{inače.} \end{cases} \tag{2.12}$$

2.4. Aktivacijske funkcije

Aktivacijske funkcije se koriste kako bi se postigla nelinearnost u modelu ili kako bi se izlazi modela mogli interpretirati kao vjerojatnosti. Koristeći nelinearnost model može naučiti kompleksnije odnose među značajkama. Kada model ne bi koristio nelinearne funkcije, moguće je za bilo koji model naći ekvivalentan bez skrivenih slojeva [2, p. 229]. U ovom radu korištene su tri aktivacijske funkcije: sigmoida, ReLU te GELU. Izlaz sigmoide se računa kao:

$$\sigma(x) = \frac{1}{1 + e^{-x}}. \tag{2.13}$$

Gradijent s obzirom na ulaz se računa kao:

$$\frac{\partial \sigma}{\partial x} = \sigma(x)(1 - \sigma(x)). \tag{2.14}$$

Najveći nedostatak sigmoide je kada je u zasićenju, vrlo je slična konstantnoj funkciji, kako je vidljivo na slici 2.1, i time "guši" gradijent. Koristila se u skrivenim slojevima modela prije nego što ju je zamijenila ReLU funkcija te se sada najčešće koristi samo na izlazu modela. Izlaz ReLU funkcije računa se kao:

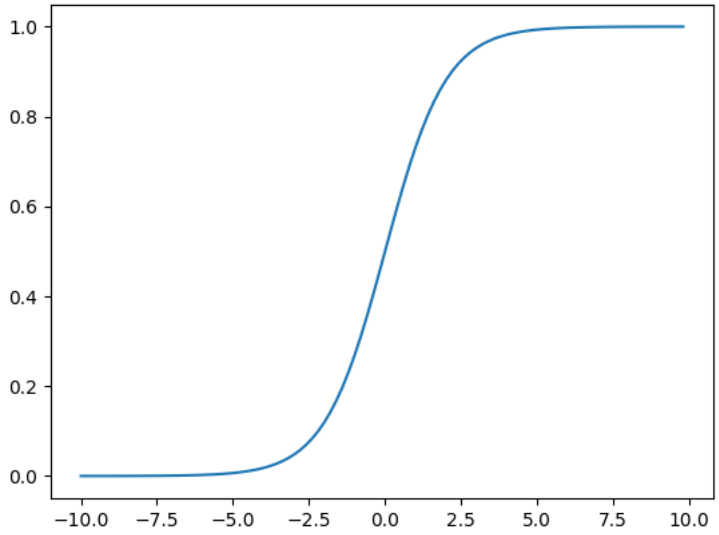
$$ReLU(x) = \begin{cases} x & \text{ako je } x > 0 \\ 0 & \text{inače.} \end{cases} \tag{2.15}$$

Gradijent s obzirom na ulaz ReLU funkcije je moguće lagano izračunati:

$$\frac{\partial ReLU}{\partial x} = \begin{cases} 1 & \text{ako je } x > 0 \\ 0 & \text{inače.} \end{cases} \tag{2.16}$$

Jedan od nedostataka ReLU funkcije je to što je moguće da svi izlazi nekog sloja budu manji od 0 pa svi izlazi ReLU funkcije budu 0 i gubi se cijela mapa značajki. Zato se ReLU sloj često uparuje sa slojem normalizacije po grupi koji postavlja srednju vrijednost aktivacija 0 te varijancu na 1. Tako se ostvaruje najveća efikasnost sloja jer će 50% izlaza biti propušteno.

U nekim slučajevima, bolja verzija ReLU funkcije je GELU [7]. GELU funkcija se često koristi u modelima koji se temelje na mehanizmu pažnje u području računalnog vida. Usporedba GELU i ReLU funkcije je prikazana na slici 2.2.



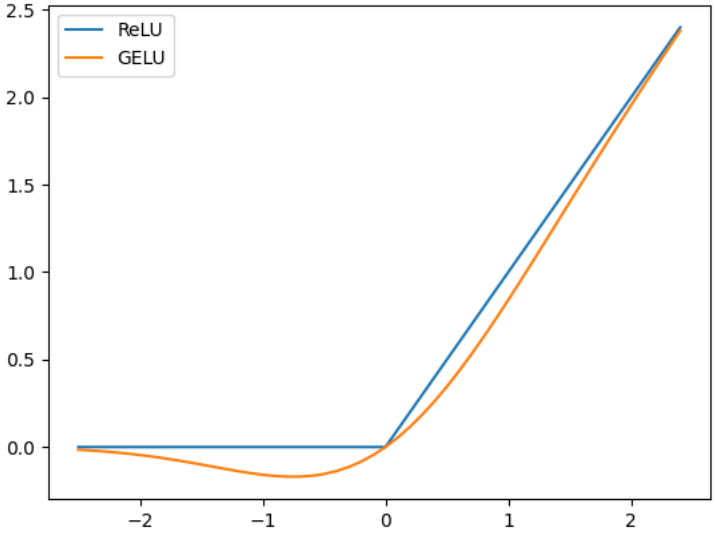
Slika 2.1: Prikaz sigmoidalne funkcije.

2.5. Preskočne veze

Poznato je da povećavanje dubine modela povećava i kapacitet modela. Povećavanjem kapaciteta, moguće je riješiti sve kompleksnije probleme. Gusta predikcija često zahtjeva vrlo velik kapacitet modela koji može procesirati velike mape značajki. No, što je dublji model, to je kompleksnija funkcija gubitka i teže ga je trenirati. Jedno od rješenja je koristiti sporedne gubitke predikcija iz raznih slojeva modela i tako omogućiti lakšu propagaciju gradijenata kroz cijeli model. No, koristeći preskočne veze, više nije potrebno koristiti sporedne gubitke jer funkcija gubitka u tom slučaju postaje glađa [11]. Na slici 2.3 prikazan je gradivni blok ResNet-50 modela koji sadržava preskočnu vezu [6]. Ulazna vrijednost prolazi kroz više slojeva konvolucija, normalizacije po grupi i ReLU funkcija nakon čega se izlaz zbraja s ulazom. Zbog rezidualne veze koja slobodno propušta gradijent iz izlaza na ulaz, treniranje vrlo dubokih modela postaje stabilnije i puno lakše.

2.6. ResNet-50

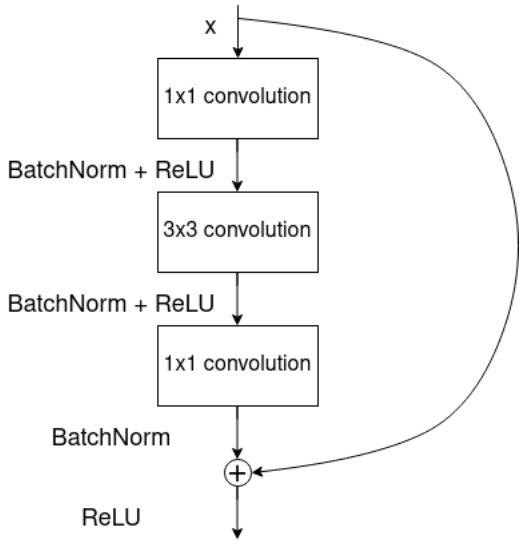
U ovom radu korišten je ResNet-50 u kombinaciji sa slojevima transformera koji će biti detaljnije opisani u 2.10. Osnovni gradivni blokovi ResNet-50 modela su rezidualne jedinice koji se sastoje od konvolucijskog sloja s veličinom jezgre 1×1, konvolucijskog sloja s veličinom jezgre 3×3 te još jednog konvolucijskog sloja s veličinom jezgre 1×1. Nakon svakog konvolucijskog sloja u bloku koriste se sloj normalizacije po



Slika 2.2: Usporedba ReLU i GELU funkcije.

grupi i ReLU funkcija osim zadnjeg konvolucijskog sloja čiji se izlaz provodi kroz sloj normalizacije po grupi i zbraja s ulazom jedinice prije ReLU funkcije. Svaka rezidualna jedinica ima hiperparametar koji određuje unutarnji broj mapi značajki w. Prvi i drugi konvolucijski sloj stvaraju izlaze s w mapa značajki dok treći konvolucijski sloj stvara izlaz s $4 \times w$ mapa značajki. Na slici 2.3 prikazana je opisana rezidualna jedinica.

Ulaz dimenzija $C \times H \times W$ u ResNet-50 modelu prolazi kroz konvoluciju s veličinom jezgre 7×7, korakom veličine 2 i 64 izlazne mape značajki. Tako dobiveni tenzor značajki prolazi kroz sloj normalizacije po grupi i ReLU funkciju te sloj sažimanja maksimalnom vrijednosti s veličinom jezgre 3×3 i korakom 2 kako bi se dobio izlaz dimenzija $64 \times \frac{H}{4} \times \frac{W}{4}$. Dobiveni tenzor slijedno prolazi kroz 4 rezidualna bloka. Prvi rezidualni blok sastoji se od 3 konvolucijske jedinice s $w = 64$ i stvara izlaz dimenzija $256 \times \frac{H}{4} \times \frac{W}{4}$. Drugi rezidualni blok sastoji se od 4 jedinice s $w = 128$ i stvara izlaz dimenzija $512 \times \frac{H}{8} \times \frac{W}{8}$. Treći rezidualni blok sastoji se od 6 jedinica s $w = 256$ i stvara izlaz dimenzija $1024 \times \frac{H}{16} \times \frac{W}{16}$. Četvrti rezidualni blok sastoji se od 3 jedinice s $w = 512$ i stvara izlaz dimenzija $2048 \times \frac{H}{32} \times \frac{W}{32}$. U prvoj jedinici svakog bloka, preskočna veza se provodi kroz konvolucijski sloj s veličinom jezgre 1×1 i izlaznim brojem značajki $w \times 4$ u prvom rezidualnom bloku i $w \times 2$ u ostalima te sloj normalizacije po grupi kako bi broj mapi značajki odgovarao izlaznom. Također, u prvoj jedinici 2., 3. i 4. bloka, konvolucijski sloj s veličinom jezgre 3×3 i konvolucijski sloj kroz koji se provodi rezidualna veza imaju veličinu koraka 2 čime smanjuju rezoluciju mape značajki dva



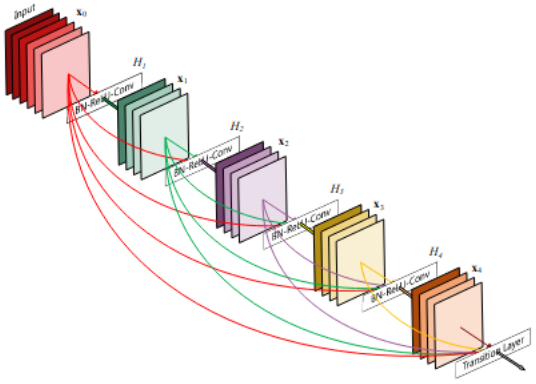
Slika 2.3: Prikaz gradivnog bloka modela ResNet-50 s preskočnom vezom koji se naziva rezidualna jedinica. Ulaz prolazi kroz niz konvolucijskih slojeva, slojeva normalizacije po mini-grupi i ReLU funkcija. Izlaz zadnjeg sloja normalizacije po mini-grupi se zbraja s ulazom u jedinicu prije nego što se provede kroz nelinearnost ReLU.

puta. Izlaz zadnjeg rezidualnog bloka zatim prolazi kroz globalno sažimanje srednjom vrijednosti te potpuno povezani sloj s izlaznom dimenzionalnosti 1000 što odgovara broju klasa za ImageNet klasifikaciju.

2.7. DenseNet

Arhitektura DenseNet [8] inspirirana je uspjehom arhitekture ResNet. U ovom radu koristimo više modela iz obitelji DenseNet: DenseNet-121, DenseNet-169, DenseNet-201 te DenseNet-161. Te modele koristili smo kao okosnice modela Ladder DenseNet. Svaki od DenseNet modela ima hiperparametar k koji se naziva korak rasta i označava broj izlaznih mapa značajki svake konvolucijske jedinice. Svi navedeni modeli imaju stopu rasta $k = 32$ osim DenseNet-161 modela koji ima stopu rasta od $k = 48$. Osnovni gradivni blokovi DenseNet modela su gusto povezani blokovi koji se sastoje od konvolucijskih jedinica i tranzicijskih slojeva. U svakoj konvolucijskoj jedinici, ulaz se provodi kroz konvoluciju s veličinom jezgre 1×1 i $4 \times k$ izlaznih mapa značajki te konvoluciju s veličinom jezgre 3×3 gdje prije svake konvolucije ulaz prolazi kroz sloj normalizacije po grupi i ReLU funkciju. Gusto povezani blok se sastoji od određenog broja konvolucijskih jedinica gdje svaka jedinica prima spojene izlaze prošlih jedinica

unutar tog bloka zajedno s ulazom bloka. Tako se postiže gusta povezanost konvolucijskih jedinica unutar bloka i pospješuje se protok gradijenata kroz model. Na slici 2.4 prikazan je primjer gusto povezanog bloka. Tranzicijski slojevi se sastoje od slo-



Slika 2.4: Prikaz gusto povezanog bloka s 5 slojeva i stopom rasta $k = 4$ preuzet iz rada [8]. Ulaz se sastoji od 6 mapi i svaki konvolucijski sloj stvara izlaz s 4 mape značajki. Svakoj konvolucijskoj jedinici se dovode izlazne mape prošlih slojeva zajedno s ulazom bloka. Na kraju, sve mape značajki se dovode na ulaz tranzicijskog sloja.

jeva normalizacije po grupi, ReLU funkcije, konvolucijskog sloja s veličinom jezgre 1×1 koji smanjuje broj značajki dva puta te sloja sažimanja srednjom vrijednosti s veličinom jezgre 2×2 i veličinom koraka 2.
Svim modelima ulaz prolazi kroz konvolucijski sloj s veličinom jezgre 7×7, veličinom koraka 2 i izlaznim brojem značajki 64, osim DenseNet-161 modela kod kojeg korijenska konvolucija proizvodi 96 mapa značajki. Nakon toga, izlazi prolaze kroz gusto povezane blokove s različitim brojem konvolucijskih jedinica prikazanih na tablici 2.1. Između svaka dva susjedna gusto povezana bloka nalazi se tranzicijski sloj. Na kraju, izlazi prolaze kroz globalni sloj sažimanja srednjom vrijednosti te potpuno povezanim slojem s izlaznom dimenzionalnosti 1000 što odgovara broju klasa za ImageNet klasifikaciju.

2.8. U-Net struktura

U arhitekturama poput ResNet ili DenseNet modela, princip obrade je ulaz postupno smanjivati i obrađivati na sve manjim rezolucijama te na kraju proizvesti klasifikaciju. Konvolucijski slojevi najčešće imaju vrlo male rezolucije jezgre te zbog toga na jednu vrijednost izlazne mape značajki utječe samo mali dio ulazne mape značajki. Sma-

Tablica 2.1: Broj konvolucijskih jedinica u gusto povezanim blokovima za svaki model.

	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-161
DenseNet blok 1	6	6	6	6
DenseNet blok 2	12	12	12	12
DenseNet blok 3	24	32	48	36
DenseNet blok 4	16	32	32	24

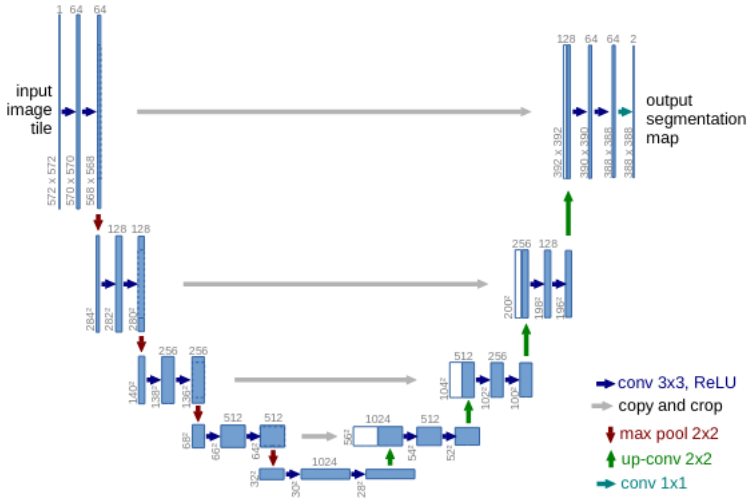
njivanjem rezolucije povećavamo receptivno polje konvolucija kasnije u modelu što dovodi do boljeg razumijevanja cjelokupnog konteksta ulaza. Slojevi postupno smanjuju rezoluciju mape značajki i povećavaju broj mapi značajki kako ne bi došlo do gubitka informacija.

U zadacima poput guste predikcije, potrebno je stvoriti izlaz iste rezolucije kao ulaz. Naime, također je potrebno što više povećati receptivno polje konvolucija jer cijeli kontekst slike može utjecati na rezultat guste predikcije, pogotovo u zadatku procjene dubine. Zato je potrebno koristiti dio mreže koji će smanjivati rezoluciju mape značajki kako bi se obrađivale konvolucijskim slojevima koji imaju veliko receptivno polje, ali i dio koji će postupno povećavati mapu značajki kako bi se postigla ista rezolucija na izlazu kao i na ulazu.

Jedan od najpoznatijih modela s takvom arhitekturom je U-Net model predstavljen u radu [18]. Model je originalno predstavljen za zadatak segmentacije medicinskih slika, ali je brzo pronašao primjenu u cijelom području guste predikcije. Arhitektura je prikazana na slici 2.5 i sastoji se od tri dijela: dijela koji smanjuje rezoluciju reprezentacije i često se naziva enkoder, uskog grla, i dijela koji povećava rezoluciju reprezentacije i često se naziva dekoder. Svaki blok dekodera uz dosad generirani kontekst dobiva i izlaz enkodera na odgovarajućoj razini koje spaja i obrađuje. U enkoderskom dijelu, mapa značajki se postupno smanjuje koristeći slojeve sažimanja maksimalnom vrijednosti dok se u dekoderskom dijelu mapa značajki postupno povećava koristeći transponiranu konvoluciju. U ovom radu je korišten MIMO-UNet model koji koristi modificiranu U-Net arhitekturu koja prima više ulaznih slika i generira više izlaznih slika.

2.9. Sloj pažnje i sloj pažnje s više glava

Mnogi recentni modeli zasnivaju se na sloju pažnje [19]. Velika prednost ovog sloja je što može primiti ulaze varijabilnih duljina i obraditi ih odjednom za razliku od LSTM

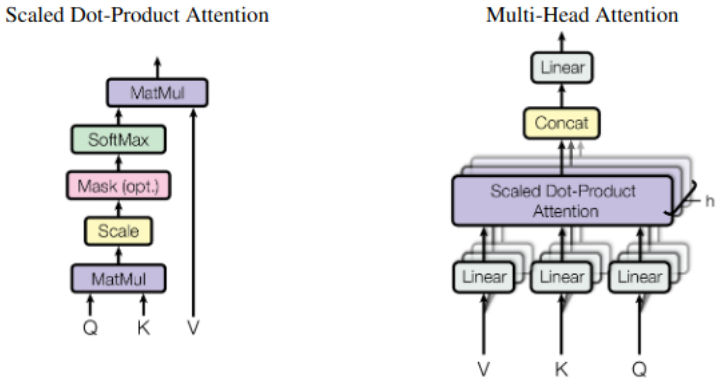


Slika 2.5: Prikaz U-Net arhitekture preuzet iz rada [18] s vidljivim gradivnim blokovima: enkoder, usko grlo i dekoder. Također, prikazane su i preskočne veze iz enkodera prema dekoderu.

slojeva koji obrađuju element po element slijeda.

Slika 2.6 prikazuje da sloj pažnje prima tri ulazna vektora: Q (ili $query$) dimenzija $B \times N_t \times D$ te K (ili key) i V (ili $value$) dimenzija $B \times N_s \times D$ gdje je B veličina minigrupe, N_s i N_t duljine slijeda i D dimenzionalnost vrijednosti. Nad vektorima Q i K vrši se operacija matričnog množenja nakon čega se dobiva QK matrica dimenzija $B \times N_t \times N_s$ nad kojom se obavlja dijeljenje elemenata korijenom dimenzionalnosti D radi stabilnosti treniranja i problema nestajućih gradijenata. Nakon dijeljenja, u maskiranoj verziji sloja pažnje koja se koristi u slojevima dekodera, koristi se maska kako bi se spriječio protok informacija na pozicijama iz budućnosti. Vrijednost tih elemenata se postavlja na $-\infty$ i taj će proces biti detaljnije opisan u 2.10. Nakon toga, skalirana matrica se provodi kroz softmax funkciju i *dropout* sloj koji sprječava prenaučenost tako da u fazi učenja nasumično postavi određeni postotak aktivacija na 0. Na kraju, izlaz *dropout* sloja se kombinira s V vektorom koristeći operaciju matričnog množenja i dobivamo rezultat sloja pažnje dimenzija $B \times N_t \times D$.

Navedeni sloj pažnje se često radi više puta u paraleli u sklopu sloja koji se naziva sloj pažnje s više glava prikazan na slici 2.6. Svaki od ulaznih Q, K i V vektora prolaze kroz h zasebnih potpuno povezanih slojeva, kako bi se dobilo h različitih Q', K' i V' vektora. Nakon toga, ti vektori prolaze kroz h slojeva pažnje nakon čega su spojeni i provedeni kroz još jedan potpuno povezani sloj kako bi se promijenila dimenzionalnost. Svako od h paralelnih izvršavanja naziva se glava i autori rada koriste



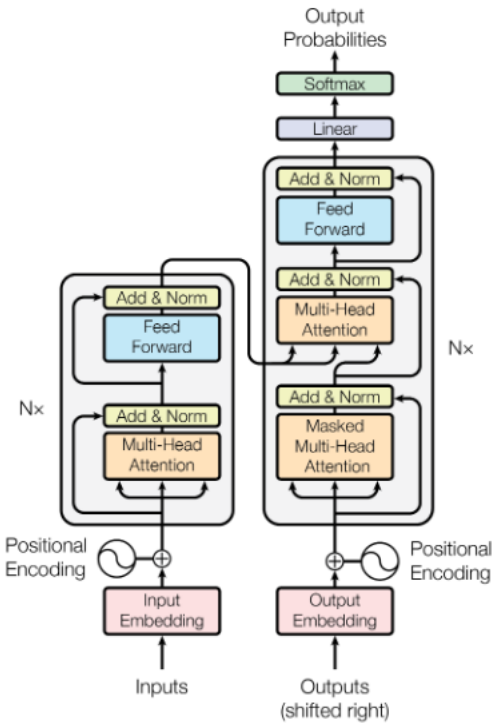
Slika 2.6: Prikaz slojeva pažnje predstavljenih u radu [19]. Na lijevoj slici prikazana je operacija pažnje nad Q, K i V vektorima koja se koristi u sloju pažnje s više glava prikazanom na desnoj slici. Na desnoj slici prikazane su h glava koje obavljaju operaciju pažnje u paraleli.

8 glava gdje svaka glava ima dimenzionalnost $D = \frac{D_{model}}{h}$.

2.10. Arhitektura transformera

Arhitektura transformera predstavljena u radu [19] prikazana je na slici 2.7. Ovaj model je originalno predstavljen kao metoda stanja tehnike za područje obrade prirodnog jezika, ali moguće ga je koristiti u drugim područjima poput računalnog vida uz male promjene gdje će i dalje imati vrlo dobre performanse. Na ulaz transformera dovode se slijedne podatkovne reprezentacije varijabilnih duljina koje se sastoje od *tokena*. Slijedovi unutar mini-grupe se nadopunjavaju <*PAD*> *tokenima* na duljinu najdužeg slijeda kako bi se dobio ulaz $B \times N$ gdje je B veličina mini-grupe, a N duljina najduljeg slijeda u mini-grupi. Unaprijednim prolazom kroz enkoder stvara se izlaz koji se često naziva i memorija. Dekoder generira izlaz *token* po *token* tako da na ulaz prima dosad generirane izlazne *tokene*. Tijekom prvog prolaza, na ulaz dekodera se postavlja poseban token za početak slijeda <*SOS*>. Dekoder generira *tokene* dok ne dođe do *tokena* koji označava završetak slijeda <*EOS*>. Memorija koju je generirao enkoder koristi se u svakom dekoderskom bloku, u sloju enkoder-dekoder pažnje.

Ulaz prvo prolazi kroz sloj koji računa linearno ugrađivanje riječi. Autori rada su postavili dimenziju ugrađivanja na $D = 512$. Ovaj hiperparametar se često naziva i dimenzionalnost modela. Izlaz sloja je dimenzija $B \times N \times D$. Zbog toga što sloj pažnje obrađuje ulaz varijabilne duljine odjednom, gubi se informacija o poziciji svakog *tokena*. Zato se na izlaze sloja koji računa linearno ugrađivanje riječi pribraja pozicijsko



Slika 2.7: Prikaz arhitekture transformera predstavljene u radu [19]. Na slici se jasno vide enkoderski dio koji je građen od određenog broja enkoderskih blokova te dekoderski dio koji je građen od određeng broja dekoderskih blokova.

kodiranje. Kada bismo konkatenirali pozicijsko kodiranje, dimenzionalnost tenzora bi bila $B \times N \times (D_w + D_p)$, gdje su D_p dimenzionalnost pozicijske, a D_w dimenzionalnost semantičke informacije dobivene iz sloja koji računa linearno ugrađivanje riječi. Ako ostavimo dimenzionalnost *tokena* istom, $D = D_w + D_p$, smanjujemo količinu semantičke informacije što može dovesti do lošijih performansa modela, a ako povećamo dimenzionalnost *tokena* na $B \times N \times (D + D_p)$, gdje je $D = D_w$, broj parametara se povećava i dodatno pogoršava memorijski otisak modela. Pribrajanje pozicijskog kodiranja eksperimentalno pokazuje dobre rezultate i tako maksimalno iskorištavamo semantičku informaciju. Pozicijsko kodiranje se računa kao:

$$\begin{aligned} PE_{(pos,2i)} &= sin(pos/10000^{2i/d_{model}}) \\ PE_{(pos,2i+1)} &= cos(pos/10000^{2i/d_{model}}). \end{aligned} \tag{2.17}$$

gdje je pos pozicija *tokena* u slijedu, a i indeks dimenzije. Na parnim dimenzijama primjenjuje se sinus funkcija dok se na neparnim dimenzijama primjenjuje kosinus funkcija. Tako sve pozicije imaju vrijednosti između -1 i 1 te nijedan *token* neće imati iste pozicijske vrijednosti. Izlaz pozicijskog kodiranja se provodi kroz enkoder

kako bi se generirala memorija.
Enkoder se sastoji od više enkoderskih blokova (originalno 6 [19]). Ulaz u enkoderski blok se kopira u tri vektora Q, K i V koji se provode kroz sloj pažnje s više glava koji se naziva *self-attention* slojem. Izlaz sloja pažnje s više glava se provodi kroz *dropout* sloj čiji se izlaz preskočnom vezom zbraja s ulazom sloja pažnje s više glava kako bi dobili mapu značajki dimenzija $B \times N \times D$ koja se normalizira po sloju. Normalizacija po sloju funkcionira slično kao i normalizacija po grupi. Jedina razlika je što se normalizacija po sloju provodi po dimenziji D, to jest, normaliziraju se vrijednosti svakog od N *token*-a u svakom primjeru unutar grupe. Nakon toga, izlaz sloja normalizacije prolazi kroz dva potpuno povezana sloja s ReLU funkcijom i *dropout* slojem između. Ti slojevi se često zajedno nazivaju unaprijedna mreža s obzirom na poziciju (*FFN*). Prvi potpuno povezani sloj pretvara dimenzije ulaza s D na D_{in} dok drugi potpuno povezani sloj vraća dimenzionalnost izlaza na D (originalno $D = 512$ i $D_{in} = 2048$ [19]). Izlaz potpuno povezanih slojeva se ponovno provodi kroz *dropout* sloj čiji se izlaz preskočnom vezom zbraja s ulazom *FFN*-a te ponovno normalizira normalizacijom po sloju.
Dekoder se sastoji od više dekoderskih blokova (originalno 6 [19]). Dosad generirani *tokeni* koji ulaze u dekoder također prolaze kroz *embedding* sloj s dimenzionalnosti D i pozicijsko kodiranje. Ulaz u dekoderski blok prolazi kroz maskirani *self-attention* sloj. Kada se generira izlaz y_i, on može vidjeti samo izlaze $y_j, j < i$ pa je potrebno te vrijednosti maskirati tako da se vrijednosti u QK matrici za buduće *token*-e postavljaju na $-\infty$. Izlaz maskiranog *self-attention* sloja se provodi kroz *dropout* sloj čiji se izlaz preskočnom vezom zbraja s ulazom maskiranog *self-attention* sloja i normalizira normalizacijom po sloju. Izlaz normalizacije po sloju ulazi u sloj pažnje s više glava kao Q vektor dok se kao K i V vektori uzima memorija koju je generirao enkoder te se taj sloj često naziva sloj enkoder-dekoder pažnje. Na taj način, dekoder može, uz pomoć mehanizma pažnje, težinski uzimati informacije iz ulaza u transformer ovisno o dosad generiranim izlazima. Izlaz enkoder-dekoder pažnje prolazi kroz *dropout* sloj čiji se izlaz preskočnom vezom zbraja s ulazom enkoder-dekoder pažnje i normalizira normalizacijom po sloju. Izlaz normalizacije prolazi kroz *FFN* koji je također korišten u enkoderskim blokovima te se izlaz provodi kroz *dropout* sloj. Izlazu *dropout* sloja se preskočnom vezom pribraja ulaz u *FFN* preskočnom vezom te se zbroj normalizira normalizacijom po sloju. Izlaz dekodera se provodi kroz potpuno povezani sloj kako bi se postigao željeni broj klasa te se na izlaz potpuno povezanog sloja primjenjuje operacija *softmaxa* kako bi se dobile vjerojatnosti za svaku klasu.

3. Skup podataka

3.1. NYU Depth V2

Za treniranje i testiranje, izabran je podskup skupa podataka NYU Depth V2 [14]. Skup je dobiven snimanjem scena zatvorenih prostora koristeći Microsoft Kinect. Kinect koristi običnu RGB kameru uparenu s infracrvenom kamerom koje snimaju na frekvencijama od 9 do 30 Hz, ovisno o rezoluciji. Infracrvena kamera koristi strukturirani i nevidljivi uzorak koji projicira na površinu koju snima. Prema deformaciji uzorka, moguće je odrediti dubinu scene u svakom pikselu.
Skup se sastoji od 1449 gusto označenih parova RGB slika i dubinskih mapa te 407024 neoznačenih parova iz 464 scena. Gusto označeni parovi su dobiveni sinkroniziranjem slika s metapodacima nakon čega su, uz podatke o dubini, anotirane i za zadatak semantičke segmentaciju. Pošto je cijeli skup podataka velik otprilike 430 GB, u eksperimentima je korišten manji podskup od 50000 parova RGB slika i dubinskih mapa za treniranje te skup od 654 parova RGB slika i dubinskih mapa za testiranje [5]. Slike su dimenzija 640×480 i vrijednosti dubinskih mapa su od 0 do 10 metara. Kako bi se popunile nedostajuće vrijednosti dubinskih mapa, korištena je metoda optimizacije [10].

3.2. Pretprocesiranje ulaznih slika

Prije ulaska u model, na slike primjenjujemo niz transformacija.

Izrezivanje središta

Prva transformacija je izrezivanje središta. Dubinske mape koje Kinect snima sadrže bijeli rub pa je potrebno od ulazne slike i dubinske mape rezolucija 640×480 izrezati središte rezolucije 608×448 prilikom treniranja. Slike iz skupa za testiranje koriste

Slika 3.1: Primjer slike iz skupa podataka NYU Depth V2. Plavi pikseli označavaju dijelove slike koji su bliže, crveni dijelove slike koji su dalje, a crni nedostajuće vrijednosti. Slika je preuzeta iz rada [14]. U ovom radu su korištene drugačije mape boje za prikaz dubinskih mapa. U ovom radu, žuta boja označava dijelove slike koji su bliže, a plava dijelove slike koji su dalje.

predefinirane parametre za izrezivanje središta [5].

Pretvaranje u Torch tenzor

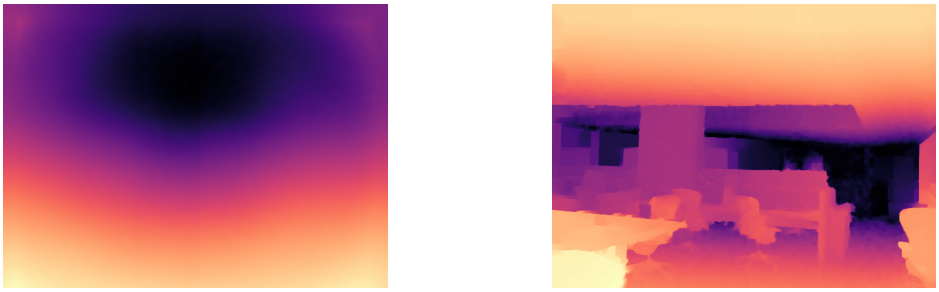
Nakon izrezivanja središta, RGB slika i dubinska mapa se pretvaraju u *Torch* tenzor koristeći transformaciju *torchvision.transforms.ToTensor*.

Promjena veličine slike

Rezolucija tenzora se smanjuje sa 608×448 na 320×240 koristeći transformaciju *torchvision.transforms.Resize* zbog ograničenja memorije tijekom treniranja.

Normalizacija

Nakon promjene veličine, tijekom faze treniranja primjenjujemo augmentacije koje će biti objašnjene u 3.3.
Kod normalizacije, RGB slika se normalizira tako da se od svakog kanala slike oduzima vrijednost μ_c te se dijeli s vrijednosti σ_c. U slučaju treniranja MIMO-UNet i DPT modela vrijednosti (μ_R, μ_G, μ_B) i $(\sigma_R, \sigma_G, \sigma_B)$ iznose (0.5, 0.5, 0.5) kako bi raspon vrijednosti piksela ulazne slike pretvorili iz $[0, 1]$ u $[-1, 1]$. U slučaju treniranja Ladder DenseNet modela, vrijednosti (μ_R, μ_G, μ_B) iznose (0.485, 0.456, 0.406), a vrijednosti $(\sigma_R, \sigma_G, \sigma_B)$ iznose (0.229, 0.224, 0.225). Te su vrijednosti dobivene računanjem srednjih vrijednosti i standardnih devijacija kanala slika iz ImageNet skupa. Okosnica Ladder DenseNet modela je predtrenirana na ImageNet skupu pa, kako bi maksimalno



Slika 3.2: Na prvoj slici je prikaz uprosječenih dubinskih mapa iz skupa za treniranje, a na drugoj slici *outlier* primjer koji iskače iz distribucije.

iskoristili predtrenirane parametre, prilagođavamo vrijednosti normalizacije korištenima u procesu predtreniranja.

3.3. Augmentacije

Kako bi se povećao skup podataka za treniranje, smanjio utjecaj prenaučenosti te povećala robusnost modela, koriste se augmentacije podataka tijekom faze treniranja. Sve navedene augmentacije se primjenjuju s vjerojatnošću 0.2. Rezultati svih augmentacija su prikazani na slici 3.3.

Zamućivanje Gaussovom jezgrom

Prilikom slikanja, može doći do zamućivanja slike zbog raznih faktora kao što su pomicanje kamere ili loš fokus kamere. Kako bi model radio dobro i na mutnim slikama, na slike se primjenjuje zamućivanje Gaussovom jezgrom s nasumično odabranim vrijednostima veličine jezgre iz raspona $[5, 9]$ i standardne devijacije iz raspona $[0.1, 5]$.

Vodoravno i vertikalno zrcaljene

Jedna od najčešće korištenih augmentacija za zadatak monokularne predikcije dubine je vodoravno zrcaljenje slike i dubinske mape jer je to najjednostavniji i najbolji način da se proširi skup za treniranje. Augmentirane slike su identičnog stila kao i ostale slike iz skupa za treniranje pa mogu imati veliki utjecaj u sprječavanju prenaučenosti. Na slici 3.2 je prikazana srednja vrijednost svih dubinskih mapa u skupu za treniranje. Zbog takve distribucije dubine, moguće je da se model prenauči te u slučajevima kao što je prikazani *outlier* daje krive predikcije. Zato se koristi augmentacija vertikalnog zrcaljenja slike i dubinske mape.

Promjena gama vrijednosti, nijansa i kontrasta

Promjena nekih parametara slike kao što su gama vrijednost, nijansa ili kontrast može poboljšati generalizaciju modela i time smanjiti prenaučenost. Promjena gama vrijednosti mijenja osvjetljenje slike pa koristeći ovu augmentaciju poboljšavaju se performanse modela na tamnijim i svjetlijim slikama. Za ulaznu sliku I_{in}, izlazna slika I_{out} se računa kao:

$$I_{out} = 255 \times \left(\frac{I_{in}}{255}\right)^{\gamma}, \tag{3.1}$$

gdje je γ vrijednost nasumično odabrana iz raspona $[\gamma_{min}, \gamma_{max}]$. U ovom radu, vrijednosti su $\gamma_{min} = 0.4$ i $\gamma_{max} = 1.6$. Promjena nijansa se vrši u HSV formatu slike mijenjanjem H vrijednosti. Promjenom H vrijednosti, mijenja se paleta boja u slici. Zato je prvo potrebno sliku pretvoriti iz RGB formata u HSV. Zatim se vrijednostima H kanala pribraja faktor pomaka δ_{hue} koji mora biti u rasponu $[-0.5, 0.5]$. U ovom radu, vrijednost δ_{hue} se bira nasumično iz tog raspona.
Kontrast označava razliku u osvjetljenju ili boji na slici. Povećavanjem faktora kontrasta $\delta_{contrast}$ boje postaju istaknutije dok smanjenjem te vrijednosti boje blijede. Prvo se računa Y komponenta slike za svaki piksel koja se naziva i osvjetljenje. Ta se vrijednost računa po formuli:

$$Y = 0.2989 \times R + 0.587 \times G + 0.114 \times B, \tag{3.2}$$

gdje su R, G i B vrijednosti crvene, zelene i plave boje za piksel. Nakon toga se računa srednja vrijednost svih Y komponenta, Y_{mean}. Izlazna slika se tada računa kao:

$$I_{out} = max(0, min(255, \delta_{contrast} \times I_{in} + (1 - \delta_{contrast}) \times Y_{mean}), \tag{3.3}$$

gdje su I_{in} ulazna, I_{out} izlazna slika, a $\delta_{contrast}$ nasumično odabrana vrijednost iz raspona $[0.4, 1.6]$.

Slika 3.3: Na lijevoj slici je prikazana slika iz skupa za učenje. Na desnoj slici su prikazani rezultati svake od korištenih augmentacija nad lijevom slikom. U prvom redu prikazani su rezultati zamućivanja Gaussovom jezgrom i vodoravnog zrcaljenja. U drugom redu prikazani su rezultati vertikalnog zrcaljenja i promjene gama vrijednosti. U trećem redu, prikazani su rezultati promjene nijanse i promjene kontrasta.

4. Metrike za procjenu kvalitete predikcija

Kako bismo lakše evaluirali i uspoređivali modeli, preuzeli smo metrike koje se često koriste za zadatak monokularne estimacije dubine i opisali ih u nastavku poglavlja. Metrike se računaju nad predviđenom dubinskom mapom P i točnom dubinskom mapom T dimenzija $w \times h$.

Log10 pogreška (Log10)

Log10 pogreška je metrika koja se koristi kako bi se usporedila apsolutna vrijednost razlike između logaritmiranih vrijednosti predviđene i točne dubinske mape i računa se kao:

$$Log_{10} = \frac{\sum_{i=0}^{h} \sum_{j=0}^{w} |log_{10}(P(i,j)) - log_{10}(T(i,j))|}{h \times w}. \tag{4.1}$$

Korijen srednje kvadratne pogreške (RMSE) i logaritmirani korijen srednje kvadratne pogreške (LRMSE)

Korijeni srednje vrijednosti kvadratne pogreške računaju se koristeći kvadriranu razliku između odgovarajućih vrijednosti dubinskih mapa ili, u slučaju logaritamske verzije, kvadriranu razliku između prirodnih logaritama tih vrijednosti te se računa korijen te razlike. Rezultat korijena srednje kvadratne pogreške se tada računa kao:

$$\text{RMSE} = \sqrt{\frac{\sum_{i=0}^{h} \sum_{j=0}^{w} (P(i,j) - T(i,j))^2}{h \times w}}, \tag{4.2}$$

a rezultat logaritmiranog korijena srednje kvadratne pogreške kao:

$$\text{LRMSE} = \sqrt{\frac{\sum_{i=0}^{h} \sum_{j=0}^{w} (ln(P(i,j)) - ln(T(i,j)))^2}{h \times w}}, \tag{4.3}$$

Relativna apsolutna pogreška (RAE) i relativna kvadratna pogreška (RSE)

Relativne greške se računaju kao apsolutna ili kvadrirana razlika dubinskih mapa podijeljena s vrijednostima točne dubinske mape značajki. Rezultat relativne apsolutne pogreške se računa kao:

$$\text{RAE} = \frac{\sum_{i=0}^{h} \sum_{j=0}^{w} \frac{|P(i,j) - T(i,j)|}{T(i,j)}}{h \times w}, \tag{4.4}$$

a rezultat relativne kvadratne pogreške kao:

$$\text{RSE} = \frac{\sum_{i=0}^{h} \sum_{j=0}^{w} \frac{(P(i,j) - T(i,j))^2}{T(i,j)}}{h \times w}, \tag{4.5}$$

Delta pogreške ($\delta < 1.25^{\epsilon}$)

Delta pogreške su najlakša metrika za interpretirati, efektivno predstavljajući točnost modela za različite vrijednosti relaksacije metrike ϵ. Delta pogreške se računaju kao omjer odgovarajućih vrijednosti dubinskih mapa te računanje postotka vrijednosti koje su manje od nekog broja. Rezultat delta pogreške se računa kao:

$$\tau(i,j) = \begin{cases} 1 & \text{ako } max(\frac{P(i,j)}{T(i,j)}, \frac{T(i,j)}{P(i,j)}) < 1.25^{\epsilon}, \\ 0 & \text{inače} \end{cases}$$

$$\delta < 1.25^{\epsilon} = \frac{\sum_{i=0}^{h} \sum_{j=0}^{w} \tau(i,j)}{h \times w}. \tag{4.6}$$

U ovom radu su korištene vrijednosti $\epsilon = \{1, 2, 3\}$.

5. Okruženje za eksperimente

Za eksperimente u ovom radu, korišten je stroj s Intel Core i5-7300HQ procesorom, 8 GB RAM memorije i GeForce GTX 1050 grafičkom karticom s 4 GB memorije.

5.1. Korištene knjižnice

NumPy

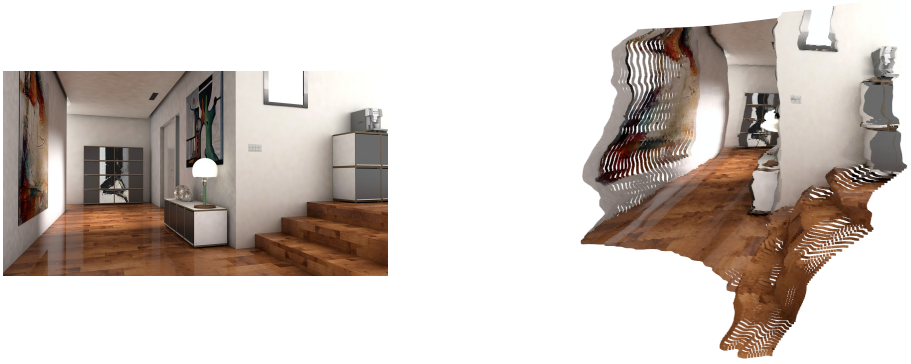
NumPy je popularna knjižnica za rad s višedimenzionalnim podacima te nudi mnoge operacije nad njima kao što su matrične operacije i generiranje nasumičnih brojeva. U ovom radu, NumPy je korišten za operacije nad matricama nakon učitavanja slike kao što je izrezivanje središta te za vizualizaciju rezultata modela.

OpenCV i Pillow

OpenCV i Pillow su najpoznatije knjižnice za učitavanje slika i operacije nad njima. U ovom radu, slike su učitane funkcijama iz Pillow knjižnice te kasnije pretvarane u NumPy matrice. Iz OpenCV knjižnice, koristi se funkcija *applyColorMap* koja za neki ulaz stvara izlaznu mapu boja ovisno o vrijednostima ulaza i tako su vizualizirane predikcije modela.

PyTorch

PyTorch je jedna od najpoznatijih knjižnica za automatsku diferencijaciju te definiranje i treniranje dubokih modela. Nudi vrlo bogat izbor funkcija koje se koriste u dubokim modelima i time pruža veliku fleksibilnost. U ovom radu je PyTorch korišten za definiranje modela, optimizatora, funkcije gubitka, rad s podacima prije ulaska u model te sam proces treniranja i evaluacije.



Slika 5.1: Na prvoj slici je prikazana ulazna slika interijera, a na drugoj slici je oblak točaka koji je generiran iz izlaza modela Ladder DenseNet-161 koristeći knjižnicu Open3D.

Torchvision i timm

Torchvision nudi mnoge operacije za transformaciju i augmentaciju slika te sadrži definicije poznatijih modela korištenih u području računalnog vida. Timm je skraćeni naziv za knjižnicu PyTorch Image Models koja sadrži dodatne definicije modela koje ne sadrži Torchvision. Sve transformacije i augmentacije ostvarene su funkcijama iz Torchvision knjižnice kao i definicija DenseNet modela s predtreniranim težinama koji se koristi u Ladder DenseNet modelu. Iz timm knjižnice, preuzeta je definicija ViT-Hybrid modela kako bi implementacija odgovarala korištenoj u službenoj implementaciji DPT modela.

PyTorch Lightning

PyTorch Lightning je knjižnica koja omogućava laganu nadogradnju i brzo definiranje procedura za treniranje i testiranje modela. Neke od tih procedura su automatsko pozivanje funkcija za računanje gradijenata i ažuriranje težina, automatsko pomicanje tenzora na *CUDA* uređaje, automatsko spremanje najboljih težina modela. Također, nudi gotova rješenja za naprednije procedure kao što su distribuirano treniranje, automatsko traženje najbolje stope učenja, učenje s automatsko miješanom preciznosti te pronalaženje najveće moguće veličine mini-grupe.

Open3D

Open3D je knjižnica za vizualizaciju i operacije nad 3D podacima. U ovom radu su korištene funkcije za vizualizaciju oblaka točaka kako bi se lakše interpretirali i vizualizirali rezultati modela. Primjer 3D vizualizacije je prikazan na slici 5.1.

6. MIMO-UNet

6.1. Arhitektura modela

U radu [3] predstavljen je model MIMO-UNet koji odmućuje sliku na više razina. Ulazna slika se skalira na različite rezolucije nakon čega se te reprezentacije istovremeno provode kroz model koji stvara jednak broj izlaza. Autori rada navođenjem drugih radova argumentiraju dobre performanse modela koji odmućuju sliku na više razina. Cilj modela je preuzeti ideju provođenja višerazinske obrade ulaza, ali u isto vrijeme riješiti problem računske složenosti takvih modela. Arhitektura modela je prikazana na slici 6.1. Ulazna slika skalira se na 3 razine: originalna veličina, 50% originalne veličine te 25% originalne. SCM (*shallow convolutional layer*, prikazan u slici 6.2(a)) se koristi kako bi se izvukle značajke iz umanjenih slika. Sastoji se od 4 konvolucijska sloja s naizmjenično 3×3 i 1×1 veličinom jezgre. Izlaz niza konvolucija se spaja s ulaznom slikom te se provodi kroz još jednu konvoluciju s 1×1 veličinom jezgre. Nad slikom originalne veličine primjenjuje se enkoderski blok EB_1. Enkoderski blok EB_1 sastoji se od konvolucijskog sloja koji mijenja dimenzionalnost mapi značajki na $w \times h \times 32$ te niza rezidualnih blokova. Izlaz enkoderskog bloka, EB_1^{out}, se tada daje kao ulaz enkoderskom bloku EB_2. Enkoderski blokovi EB_2 i EB_3 imaju istu strukturu. Prvo konvolucijski sloj s korakom 2 smanjuje rezoluciju mape značajki dva puta i udvostručuje broj mapa značajki. Nakon toga, koristeći *feature attention module* prikazan u slici 6.2(b) izlaz konvolucijske mreže $(\text{EB}_{k-1}^{out})^{\downarrow}$ se množi s izlazom SCM sloja te razine, SCM_k^{out}, nakon čega taj rezultat prolazi kroz konvolucijski sloj i pribraja se $(\text{EB}_{k-1}^{out})^{\downarrow}$. Na kraju, izlaz *feature attention modula* FAM_k^{out} prolazi kroz niz rezidualnih blokova. Koristeći *feature attention module*, reprezentacija niže razine koja sadrži informaciju o slici na manjoj rezoluciji se ugrađuje u model gdje se kombinira s dotad generiranim kontekstom.

Izlazi EB blokova ulaze u AFF (asymmetric feature fusion) sloj prikazan na slici 6.2(c). U sloju AFF_1 visina i širina mapa značajki EB_2^{out} i EB_3^{out} se mijenjaju kako bi odgovarale visini i širini mape značajki EB_1^{out}. U sloju AFF_2 visina i širina mapa značajki

EB_1^{out} i EB_3^{out} se mijenjaju kako bi odgovarale visini i širini mape značajki EB_2^{out}. Nakon promjene veličine, mape značajki se spajaju i provode kroz jedan konvolucijski sloj i generiraju izlazne mape značajki AFF_k^{out} to jest, AFF_1^{out} za AFF_1 te AFF_2^{out} za AFF_2.

Nakon toga, mapa značajki EB_3^{out} se provodi kroz dekoderski blok DB_3 koji se sastoji od niza rezidualnih blokova i sloja transponirane konvolucije koja povećava visinu i širinu mape značajki na $\frac{w}{2} \times \frac{h}{2}$, smanjuje broj mapa značajki na 64 te se generira izlaz DB_3^{out}. Izlaz rezidualnih blokova se također provodi kroz jedan konvolucijski sloj koji smanjuje broj značajki izlaza na 3. Tada se toj mapi značajki dodaje ulazna slika smanjena na 25% veličine preskočnom vezom te se time generira prvi izlaz $\hat{S}_3$.

Izlaz DB_3^{out} se dalje spaja s izlazom AFF_2^{out} i ulazi u dekoderski blok DB_2. DB_2 se sastoji od konvolucijskog sloja koji smanjuje broj mapa značajki na 64 nakon kojeg slijedi niz rezidualnih blokova i transponirana konvolucija koja povećava visinu i širinu mape značajki na $w \times h$, smanjuje broj mapa značajki na 32 i generira izlaz DB_2^{out}. Izlaz rezidualnih blokova se također provodi kroz jedan konvolucijski sloj koji smanjuje broj značajki izlaza na 3. Tada se toj mapi značajki dodaje ulazna slika smanjena na 50% veličine preskočnom vezom te se time generira drugi izlaz $\hat{S}_2$.

Izlaz DB_2^{out} se dalje spaja s izlazom AFF_1^{out} i ulazi u dekoderski blok DB_1. DB_1 se sastoji od konvolucijskog sloja koji smanjuje broj mapa značajki na 32 nakon kojeg slijedi niz rezidualnih blokova i još jedan konvolucijski sloj koji smanjuje broj mapa značajki na 3. Izlaznoj mapi značajki se pridodaje ulazna slika preskočnom vezom te se generira izlaz $\hat{S}_1$.

6.2. Treniranje modela

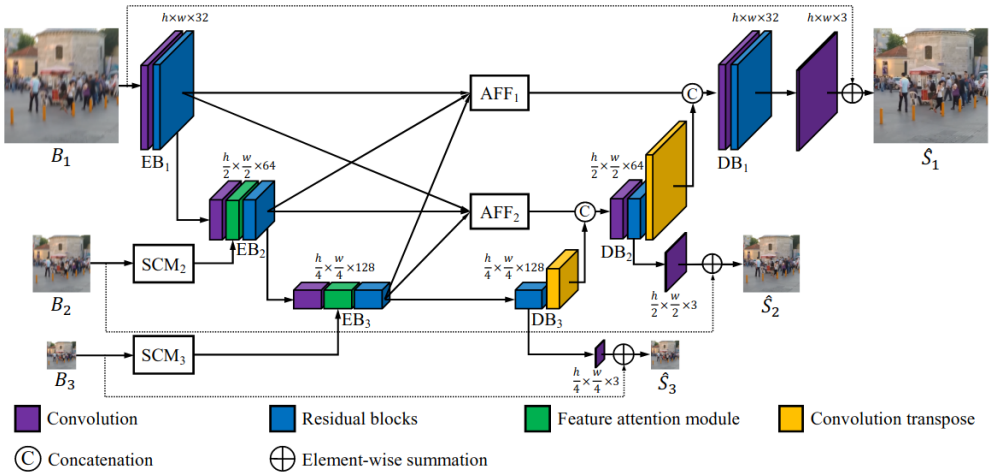
6.2.1. Gubitak

Gubitak korišten tijekom treniranja je skalirani logaritamski gubitak invarijantan na skalu (SILog) predstavljen u radu [5]. Prvo je potrebno izračunati razliku logaritama izlazne dubinske mape i točne dubinske mape:

$$\Delta d_i = \log \hat{d}_i - \log d_i^* \tag{6.1}$$

gdje je $\hat{d}_i$ vrijednost točne dubinske mape, a d_i^* vrijednost izlazne dubinske mape. Tada je moguće izračunati gubitak kao:

$$\mathcal{L} = \alpha \sqrt{\frac{1}{K} \sum_i \Delta d_i^2 - \frac{\lambda}{K^2} (\sum_i \Delta d_i)^2} \tag{6.2}$$



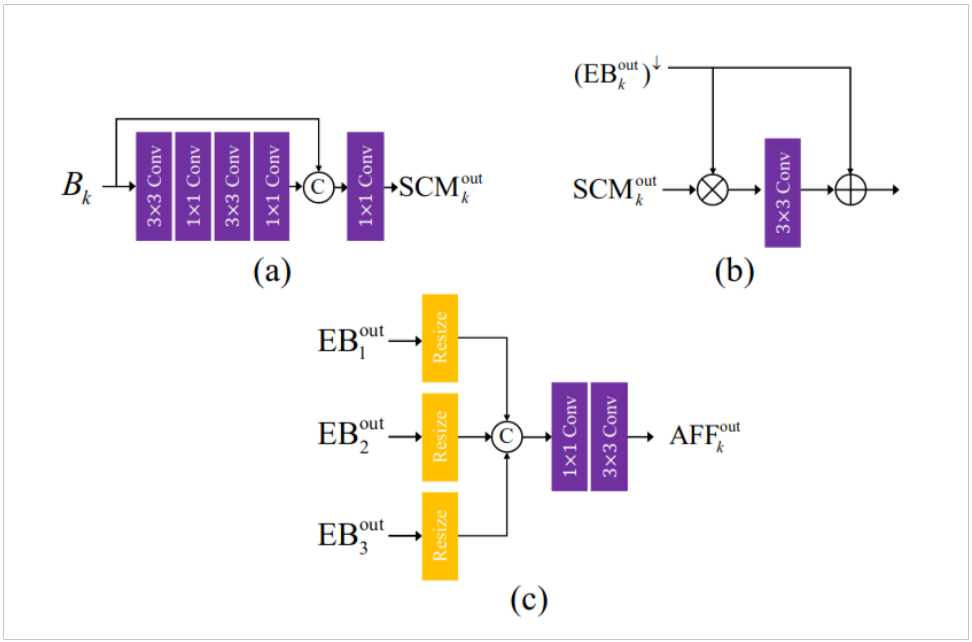
Slika 6.1: Arhitektura MIMO-UNet modela predstavljena u radu [3]. Ulazna slika na 50% i 25% rezolucije se obrađuju SCM blokovima prije nego što se ugrade u mrežu koristeći *feature attention module* blokove u enkoderskim blokovima. Koristeći AFF blokove, izlazi enkoderskih blokova se zajedno obrađuju prije ugrađivanja u dekoderske blokove koji generiraju izlazne mape.

gdje su eksperimentalno postavljene vrijednosti $\alpha = 10$ i $\lambda = 0.15$. Ovaj gubitak se koristi u mnogim *state-of-the-art* metodama za predikciju dubine kao [20, 1, 12]. Prednosti ovog gubitka je što će vrijednost gubitka biti jednaka ako pomnožimo $\hat{d}_i$ i d_i^* nekom konstantom. Zbog toga će gubitak ravnomjerno prioritizirati ažuriranje težina na svim dubinama i posljedično će učenje biti stabilnije i brže.

6.2.2. Promjene i prilagođavanje originalnog modela zadatku monokularne predikcije dubine

U zadatku predikcije dubine, za svaku izlaznu vrijednost dubinske mape vrlo je bitna lokalna, ali i globalna informacija u slici. Zato su uklonjene rezidualne veze koje povezuju ulazne slike i izlazne dubinske mape značajki na odgovarajućim razinama prikazane isprekidanim crtama na slici 6.1. Pošto sama ulazna vrijednost RGB piksela nema direktan utjecan na odgovarajući piksel u izlaznoj dubinskoj mapi, ova promjena povećava točnost modela te brzinu treniranja.

Također, na svaki od izlaza, primijenjena je sigmoidalna funkcija čiji je izlaz dodatno pomnožen s 10 što je najveća dubina u NYU Depth V2 skupu podataka. Tu metodu koriste većina radova koji koriste i SILog gubitak. [20, 12]. Koristeći sigmoidu, izlaz modela je ograničen na vrijednosti od 0 do 1 koje su množenjem s 10 skalirane na vri-

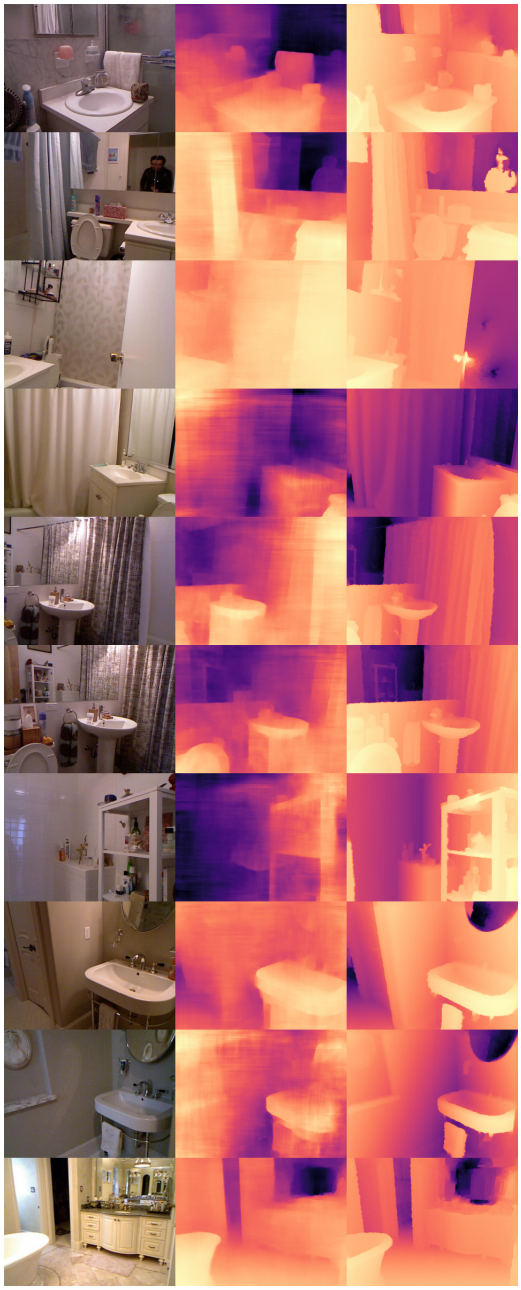


Slika 6.2: Detaljniji prikaz struktura gradivnih blokova korištenih u MIMO-UNet modelu. a) SCM (*shallow convolutional module*), b) *feature attention module*, c) AFF (*asymmetric feature fusion*). Slika je preuzeta iz rada [3].

jednosti od 0 do 10. Ta metoda uvodi stabilnost u treniranje pošto je mreža na početku nasumično inicijalizirana te se u izlaznoj dubinskoj mapi mogu pojaviti negativne vrijednosti, a logaritamska funkcija koja se koristi u SILog gubitku nije definirana za negativne vrijednosti. Također, ograničavanjem izlaza s $(-\infty, \infty)$ na očekivanih $(0, 10)$ pridonosi točnosti modela te stabilnosti i brzini učenja.

6.2.3. Rezultati eksperimenata

Model je treniran nad 350000 slika s veličinom mini-grupe 2 i početnom stopom učenja 0.0001 koristeći Adam optimizator. Nakon 150000 slika, stopa učenja je smanjena na 0.00005 te je nakon 230000 slika smanjena na 0.00001. Treniranje je trajalo 35 sati. Rezultati modela na skupu za testiranje su prikazani u tablici 9.1, a izlazne dubinske mape su vizualizirane na slici 6.3.



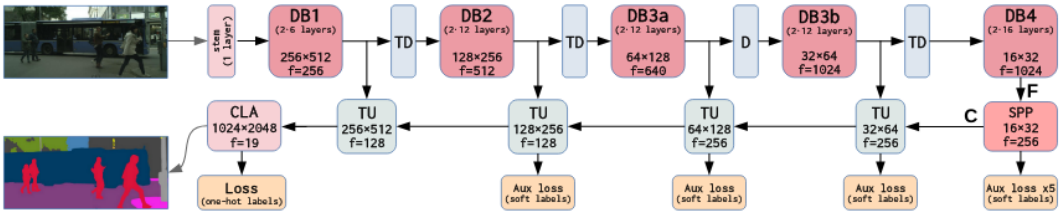
Slika 6.3: Rezultati MIMO-UNet modela na skupu za testiranje. U prvom stupcu su ulazne RGB slike, u drugom su dubinske mape na izlazu MIMO-UNet-a, a u trećem točne dubinske mape. Žuta boja označava dijelove slike koji su bliže, a plava dijelove slike koji su dalje.

7. Ladder DenseNet

7.1. Arhitektura modela

Ladder DenseNet pokazuje prednosti korištenja arhitekture ljestva i DenseNet model kao ekstraktor značajki za efikasno treniranje, veliku brzinu zaključivanja i visoku točnost za zadatak semantičke segmentacije [9]. Arhitektura modela je prikazana na slici 7.1 koja je preuzeta iz rada [9]. U ovom radu, korištene su 4 verzije Ladder DenseNet modela ovisno o korištenim ekstraktorima značajki: DenseNet-121, DenseNet-169, DenseNet-201 i DenseNet-161. DenseNet modeli su modificirani tako da su maknuti izlazni sloj globalnog sažimanja srednjom vrijednosti i potpuno povezani sloj. Također, treći DenseNet blok je podijeljen na blokove 3a koji sadrži prvu polovicu slojeva i 3b koji sadrži drugu polovicu slojeva trećeg DenseNet bloka. Između blokova postavljen je jedan sloj sažimanja srednjom vrijednosti s veličinom prozora 2×2 i veličinom koraka 2. Izlaz DenseNet modela ulazi u *spatial pyramid pooling* (SPP) modul prikazan na slici 7.2. Ulaz u SPP modul prvo prolazi kroz konvolucijski sloj s veličinom jezgre 1×1 koja smanjuje ulazni broj mapi značajki 2 puta. Izlaz konvolucije se dodatno kopira 4 puta i prolazi kroz 4 slojeva sažimanja srednjom vrijednosti s različitim veličinama kvadratnih jezgri i veličinom koraka. Neka je ulaz u slojeve sažimanja dimenzija $B \times \frac{D}{2} \times h \times w$ gdje je D broj mapi značajki na ulazu u SPP modul. Slojevi sažimanja stvaraju izlazne mape značajki s različitim veličinama redova s: 1, 2, 4 i 8. Svaki od slojeva sažimanja ima veličinu jezgre $k = \frac{h}{s}$ i veličinu koraka $\frac{h}{s}$ te stvara izlaznu mapu značajki dimenzija $B \times D \times s \times s \cdot \frac{w}{h}$. Nakon svakog sloja sažimanja slijede konvolucijski slojevi s veličinom jezgre 1×1 koji smanjuju ulazni broj mapi značajki 4 puta. Izlazi konvolucijskih slojeva prolaze kroz interpolacijske slojeve koji povećavaju rezolucije mapi značajki $\frac{h}{s}$ puta i stvaraju izlaze veličine $B \times \frac{D}{8} \times h \times w$. Izlazi svih interpolacijskih slojeva i izlaz prve konvolucijske mreže se spajaju i stvaraju izlaznu mapu značajki dimenzija $B \times D \times h \times w$ koja prolazi kroz još jedan konvolucijski sloj s veličinom jezgre 1×1 koji smanjuje broj mapi značajki 4 puta i generira izlaz iz SPP modula dimenzija $B \times \frac{D}{4} \times h \times w$. Izlaz SPP modula i DenseNet bloka 3b ulaze u

transition up blok TU prikazan na slici 7.2. Izlaz DenseNet bloka prvo prolazi kroz jedan konvolucijski sloj veličine jezgre 1×1 koji mijenja broj mapi značajki kako bi oba tenzora imala isti broj mapi značajki. Izlaz SPP modula prolazi kroz interpolacijski sloj koju povećava rezolucije mapi značajki 2 puta čiji se izlaz zbraja s izlazom konvolucije. Rezultat zbroja slijedno prolazi kroz jedan konvolucijski sloj s veličinom jezgre 1×1 koji mijenja broj mapi značajki na 128 i jedan konvolucijski sloj s veličinom jezgre 3×3 koji povećava broj mapi značajki na 256. Izlaz prvog TU bloka, zajedno s izlazom DenseNet bloka 3b, provodi se kroz još jedan TU blok koji također generira izlaz s 256 mapi značajki. Izlaz drugog TU bloka, zajedno s izlazom DenseNet bloka 2, provodi se kroz još treći TU blok gdje zadnji konvolucijski sloj s veličinom jezgre 3×3 generira izlaz sa 128 mapi značajki. Izlaz trećeg TU sloja, zajedno s izlazom DenseNet bloka 1, se provodi kroz još jedan TU sloj koji također generira izlaz s 256 mapi značajki. Izlaz zadnjeg TU bloka prolazi kroz CLA modul koji se sastoji od sloja normalizacije po mini-grupi, ReLU funkcije, konvolucijskog sloja s veličinom jezgre 1×1 koji mijenja broj mapi značajki da bude jednak broju klasa za zadatak semantičke segmentacije i jednog interpolacijskog sloja koji povećava rezolucije mapi značajki 4 puta.

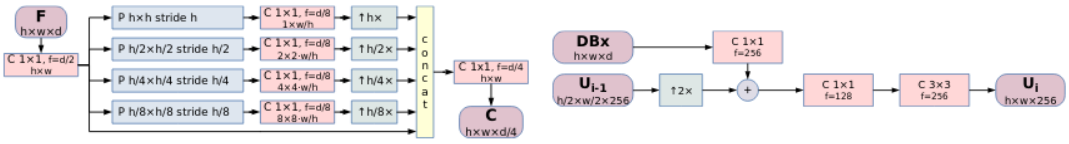


Slika 7.1: Prikaz arhitekture Ladder DenseNet modela predstavljen u radu [9] koji koristi DenseNet-121 model kao ekstraktor značajki za zadatak semantičke segmentacije.

7.2. Treniranje modela

7.2.1. Promjene i prilagođavanje originalnog modela zadatku monokularne predikcije dubine

Kako bi model generirao izlaze koji predstavljaju dubinske mape, potrebno je zamijeniti CLA modul s konvolucijskim slojem s veličinom jezgre 1×1 i jednom izlaznom mapom značajki čiji se izlaz provodi kroz interpolacijski sloj koji povećava rezoluciju



Slika 7.2: Na prvoj slici je prikazana arhitektura *spatial pyramid pooling* (SPP) modula koji paralelno obrađuje mape značajki različitih rezolucija prije spajanja. Na drugoj slici je prikazana arhitektura *transition up* (TU) bloka kojim se postupno povećava rezolucija mapi značajki i generira izlazna dubinska mapa.

Tablica 7.1: Veličine mini-grupa svakog od Ladder DenseNet modela tijekom treniranja. U drugom redu je prikazana verzija modela Ladder DenseNet-121 koja koristi metodu *checkpointinga* gradijenata unutar DenseNet modela kako bi se postigla veća veličina mini-grupe. *Korištenjem metode *checkpointinga*, moguće je povećati veličinu mini-grupe Ladder DenseNet-161 modela preko 4 puta [9].

Model	Veličina mini-grupe
Ladder DenseNet-121	16
Ladder DenseNet-121 (*checkpointing*)	28
Ladder DenseNet-169	14
Ladder DenseNet-201	6
Ladder DenseNet-161	6*

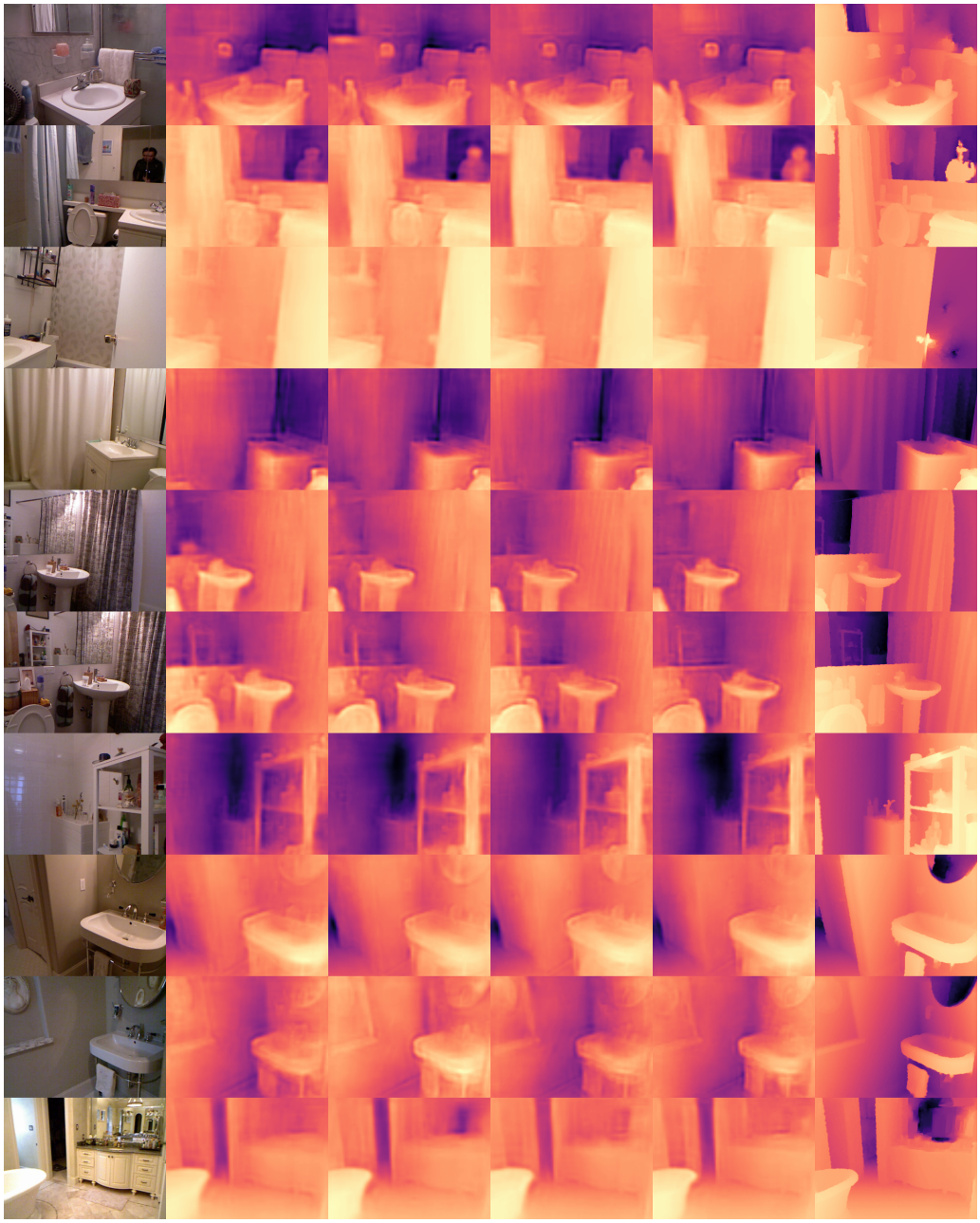
izlazne dubinske mape 4 puta. Prilikom računanja gubitka, uz izlaz modela, koriste se i izlazi SPP modula i prva 3 TU bloka. Na svaki od izlaza primijenjena je sigmoidalna funkcija kako bi se raspon vrijednosti doveo na $[0, 1]$. Izlazi sigmoidalnih funkcija su dodatno pomnoženi s konstantom 10 kako bi izlazne vrijednosti dubine odgovarale onima u skupu podataka.

7.2.2. Gubitak

Korišteni gubitak je SILog gubitak objašnjen u 6.2.1. Prije računanja gubitka izlaza SPP modula i prva 3 TU bloka, točna mapa značajki se skalira na rezolucije tih izlaza. Rezultati gubitaka prva 3 TU bloka se zbrajaju s rezultatom izlaza SPP modula pomnoženim s 5 te se rezultat zbrajanja dijeli s 8. Konačan gubitak se računa tako da se gubitak glavnog izlaza modela zbraja s pomoćnim gubitkom pomnoženim s faktorom $\alpha = 0.1$.

7.2.3. Rezultati eksperimenata

DenseNet dio modela je inicijaliziran s težinama predtreniranim na ImageNet skupu podataka. Svaki od modela je treniran je kroz 26 epoha s veličinama mini-grupe prikazanim na tablici 7.1 i početnom stopom učenja 0.0001 koristeći Adam optimizator. Korišten je *scheduler* koji množi stopu učenja s faktorom 0.5 ako je učenje postiglo plato s minimalnom stopom učenja 10^{-5}.
Rezultati modela na skupu za testiranje su prikazani u tablici 9.1, a izlazne dubinske mape su vizualizirane na slici 7.3.



Slika 7.3: Rezultati Ladder DenseNet modela na skupu za testiranje. U prvom stupcu su ulazne RGB slike, u drugom su dubinske mape na izlazu Ladder DenseNet-121 modela, u trećem su dubinske mape na izlazu Ladder DenseNet-169 modela, u četvrtom su dubinske mape na izlazu Ladder DenseNet-201 modela, u petom su dubinske mape na izlazu Ladder DenseNet-161 modela, a u šestom točne dubinske mape. Žuta boja označava dijelove slike koji su bliže, a plava dijelove slike koji su dalje.

8. Dense Prediction Transformer

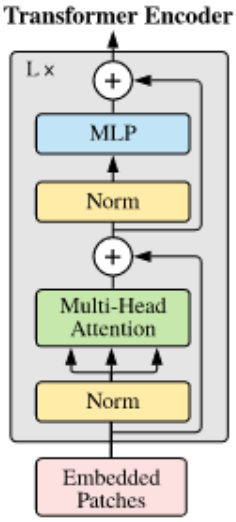
8.1. Hibridni Vision Transformer (ViT-Hybrid)

Prije objašnjenja arhitekture DPT modela, potrebno je objasniti arhitekturu Vision Transformer (ViT) modela koji je predstavljen u radu [4] i koristi se za ekstrakciju značajki iz ulazne slike u DPT modelu. U ovom radu korištena je verzija DPT modela s ViT-Hybrid modelom.

ViT-Hybrid se sastoji od modificiranog ResNet-50 i ViT-Base modela. Modificirani ResNet-50 sadrži iste ulazne slojeve i prva dva bloka kao i originalni ResNet-50. Treći blok ima istu širinu kao i originalna, ali je broj rezidualnih jedinica povećan sa 6 na 9. Također, maknuti su četvrti blok i izlazni slojevi iz modela. Povećavanjem broja jedinica u trećem bloku, ostaje isti ukupan broj jedinica kao i u originalnom ResNet-50 modelu. Na izlaz modificiranog ResNet-50 modela, postavljen je jedan konvolucijski sloj s veličinom jezgre 1×1 kako bi se promijenio broj mapi značajki na dimenzionalnost ViT-Base modela, D. U ovom slučaju, taj konvolucijski sloj smanjuje broj mapi značajki s 1024 na 768.

Nakon konvolucijskog sloja, izlazna mapa značajki dimenzija $B \times D \times H \times W$ se izravna tako da se prostorne dimenzije H i W spoje u jednu $N = H \times W$ kako bi se dobio ulaz u ViT-Base dimenzija $B \times N \times D$ koje odgovaraju očekivanim dimenzijama na ulazu u transformer modele gdje N zapravo označava broj *tokena* na ulazu u transformer. Dodatno, na početak tih N *tokena*, dodaje se još jedan, *CLS token* koji je inicijaliziran s nulama i tijekom treniranja model ažurira njegove vrijednosti. Taj *token* se često koristi za klasifikacijske zadatke i služi kao *token* koji sadrži reprezentaciju slike. Iako je primarno korišten za zadatak klasifikacije, ovaj *token* se koristi i u DPT modelu kao što je objašnjeno u 8.2. ViT-Base model je sličan arhitekturi enkoderskog dijela transformera uz male promjene u gradivnim blokovima. Prije nego što se ulazi provode kroz ViT blokove, isto kao i u enkoderu transformera, *tokenima* se pribraja pozicijsko kodiranje. Ulaz u ViT blok prikazan na slici 8.1 prvo prolazi kroz sloj normalizacije po sloju nakon čega ulazi u *self-attention* sloj. U slučaju ViT-Hybrid

modela, broj glava *self-attention* sloja iznosi 12. Izlaz *self-attention* sloja prolazi kroz *dropout* sloj čiji se izlaz zbraja s ulazom u sloj normalizacije po sloju preskočnom vezom. Nakon zbrajanja, izlaz prolazi kroz još jedan sloj normalizacije po sloju prije ulaska u *FFN*. Za razliku od originalnog transformera, umjesto ReLU korištena je aktivacijska funkcija GELU i unutarnja dimenzionalnost unaprijedne mreže prema poziciji je povećana s $D_{in} = 2048$ na $D_{in} = 3072$. Izlaz *FFN*-a prolazi kroz još jedan *dropout* sloj i zbraja se s ulazom u sloj normalizacije po sloju i generira se izlaz ViT bloka. ViT-Base se sastoji od 12 takvih blokova.



Slika 8.1: Prikaz ViT bloka, gradivnog bloka ViT modela koji je korišten u DPT modelu kao ekstraktor značajki. Slika je preuzeta iz rada [4].

8.2. Arhitektura modela

U radu [17] predstavljen je Dense Prediction Transformer (DPT) model na bazi transformera koji služi za razne zadatke guste predikcije. Arhitektura modela je prikazana na slici 8.2 zajedno s dva nova sloja predstavljena u radu, sloj ponovnog sastavljanja i sloj fuzije. Sloj ponovnog sastavljanja prima izlaze raznih slojeva ViT-Hybrid modela i ugrađuje ih u postupak dekodiranja izlazne mape.

Sloj ponovnog sastavljanja se sastoji od tri slijedne operacije: operacija čitanja, operacija konkatenacije i operacija ponovnog uzrokovanja. Korištenjem operacije čitanja, model ugrađuje informaciju *CLS tokena* u ostale *tokene*. Operacija čitanja prima ulaz dimenzija $B \times (N+1) \times D$ iz kojeg uzima *CLS token* i spaja ga sa svakim tokenom na

zadnjoj dimenziji kako bi se dobio izlaz dimenzija $B \times N \times 2D$. Taj izlaz se provodi kroz potpuno povezani sloj i GELU funkciju koja stvara izlaz dimenzija $B \times N \times D$. Korištenjem operacije konkatenacije, model priprema mape značajki za obradu konvolucijama pretvarajući ih u tenzore 4. reda. Operacija konkatenacije pretvara dimenzije mape značajki u $B \times \frac{H}{16} \times \frac{W}{16} \times D$ gdje su H i W visina i širina ulazne slike. Operacija ponovnog uzrokovanja mjenja rezoluciju i broj mapa značajki kako bi se mogle ugraditi u postupak dekodiranja. Izlaz operacije konkatenacije se provodi kroz operaciju ponovnog uzrokovanja koji sadrži jednu konvoluciju s veličinom jezgre 3×3 s istim ulaznim i izlaznim brojem značajki D. Nakon toga, koristi se još jedna konvolucija s veličinom jezgre 3×3 i odgovarajućom veličinom koraka kako bi se dimenzije mape značajki promijenile na $B \times \frac{H}{s} \times \frac{W}{s} \times D$ gdje je s hiperparametar sloja ponovnog sastavljanja. Na posljetku, koristi se konvolucija s veličinom jezgre 3×3 koja smanujuje dimenzinalnost mape značajki na $B \times \frac{H}{s} \times \frac{W}{s} \times \hat{D}$ gdje je $\hat{D} = 256$. U radu se navodi varijanta operacije ponovnog uzrokovanja koja koristi konvoluciju s veličinom jezgre 1×1 kako bi smanjila dimenzionalnost na $\hat{D} = 256$ te konvoluciju veličinom jezgre 3×3 i odgovarajućom veličinom koraka kako bi promjenila rezoluciju mape značajki na $B \times \frac{H}{s} \times \frac{W}{s} \times \hat{D}$ [17].

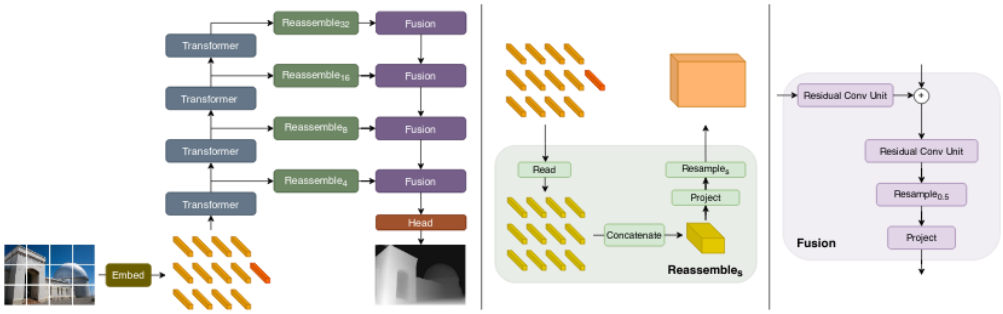
Koristeći sloj fuzije, model spaja dosad generirani kontekst s izlazima slojeva ponovnog sastavljanja. Sloj fuzije se sastoji od dvije rezidualne jedinice, interpolacijskog sloja i jedne izlazne konvolucije. Rezidualna jedinica se sastoji od dvije konvolucije s veličinom jezgre 3×3 s ulaznom i izlaznom dimenzijom veličine 256, ReLU funkcijom prije svake konvolucije i preskočnom vezom koja zbraja ulaz rezidualne jedinice s njezinim izlazom. Sloj fuzije prima izlaz sloja ponovnog sastavljanja koji provodi kroz jednu rezidualnu jedinicu čiji izlaz zbraja s trenutno generiranim kontekstom. Zbroj se provodi kroz drugu rezidualnu jedinicu čiji se izlaz provodi kroz interpolacijski sloj koji povećava rezoluciju ulaznih mapi značajki s $B \times N \times h \times w$ na $B \times N \times 2h \times 2w$. Izlaz interpolacijskog sloja na kraju prolazi kroz izlaznu konvoluciju s veličinom jezgre 1×1 i ulaznom i izlaznom dimenzijom veličine 256. U slučaju prvog sloja fuzije koji prima samo izlaz sloja ponovnog sastavljanja, ulaz se provodi kroz jednu rezidualnu jedinicu prije prolaska kroz interpolacijski sloj i izlaznu konvoluciju.

Nakon što je ulazna slika provedena kroz ViT-Hybrid model, uzimaju se izlazi na 4 različite razine, na izlazima prvog i drugog bloka modificiranog ResNet-50 modela te na izlazu 9. i 12. bloka ViT-Base modela. Na izlaze ResNet-50 modela, primjenjuje se samo operacija ponovnog uzrokovanja iz sloja ponovnog sastavljanja dok se na izlaze ViT-Base modela primjenjuju sve operacije. Točnije, na izlaz prvog bloka modificiranog ResNet-50 modela, primjenjuje se konvolucijski sloj s veličinom jezgre 1×1 koji

ima ulaznu i izlaznu dimenzionalnost 256 stvarajući izlaz dimenzija $B \times 256 \times \frac{H}{4} \times \frac{W}{4}$. Na izlaz drugog bloka modificiranog ResNet-50 modela, primjenjuje se konvolucijski sloj s veličinom jezgre 1×1 koji smanjuje broj mapi značajki s 512 na 256 stvarajući izlaz dimenzija $B \times 256 \times \frac{H}{8} \times \frac{W}{8}$. Na izlaz 9. bloka ViT-Base modela, primjenjuje se sloj ponovnog sastavljanja s hiperparametrom $s = 16$. Točnije, nakon operacija čitanja i konkatenacije, primjenjuje operacija ponovnog uzrokovanja koji se sastoji od jednog konvolucijskog sloja s veličinom jezgre 1×1 koja smanjuje broj mapi značajki sa 768 na 256 stvarajući izlaz dimenzija $B \times 256 \times \frac{H}{16} \times \frac{W}{16}$. Na izlaz 12. bloka ViT-Base modela, primjenjuje se sloj ponovnog sastavljanja s hiperparametrom $s = 32$ gdje se operacija ponovnog uzrokovanja sastoji od jednog konvolucijskog sloja s veličinom jezgre 1×1 koja smanjuje broj mapi značajki sa 768 na 256 i jednog konvolucijskog sloja s veličinom jezgre 3×3 i korakom veličine 2 koja smanjuje rezoluciju mape značajki 2 puta stvarajući izlaz dimenzija $B \times 256 \times \frac{H}{32} \times \frac{W}{32}$. Nakon toga, izlaz sloja ponovnog sastavljanja nad izlazom 12. bloka ViT-Base prolazi kroz prvi sloj fuzije koji stvara izlaz dimenzija $B \times 256 \times \frac{H}{16} \times \frac{W}{16}$. Izlaz prvog sloja fuzije i izlaz sloja ponovnog sastavljanja nad izlazom 9. bloka ViT-Base modela ulaze u drugi sloj fuzije koji stvara izlaz dimenzija $B \times 256 \times \frac{H}{8} \times \frac{W}{8}$. Izlaz drugog sloja fuzije i izlaz sloja ponovnog sastavljanja nad izlazom druge skupine modificiranog ResNet-50 modela ulaze u treći sloj fuzije koji stvara izlaz dimenzija $B \times 256 \times \frac{H}{4} \times \frac{W}{4}$. Izlaz trećeg sloja fuzije i izlaz sloja ponovnog sastavljanja nad izlazom prve skupine modificiranog ResNet-50 modela ulaze u treći sloj fuzije koji stvara izlaz dimenzija $B \times 256 \times \frac{H}{2} \times \frac{W}{2}$. Kako bi se generirala izlazna dubinska mapa, izlaz četvrtog sloja fuzije prolazi izlazni blok koji se sastoji od jednog konvolucijskog sloja s veličinom jezgre 3×3 koji smanjuje broj mapi značajki s 256 na 128, interpolacijskog sloja koji povećava rezolucije mapi značajki dva puta i tako stvara mape značajki iste rezolucije kao i ulazna slika, jednog konvolucijskog sloja s veličinom jezgre 3×3 koji smanjuje broj mapi značajki sa 128 na 32, ReLU funkcije, jednog konvolucijskog sloja s veličinom jezgre 3×3 koji smanjuje broj mapi značajki s 32 na 1 i konačno još jedne ReLU funkcije.

8.3. Treniranje modela

Model je inicijaliziran s težinama predtreniranim na *Mix 6* skupu podataka koji sadrži 1.4 milijuna slika. Taj skup podataka sadrži drugačije raspone dubine pa kako bi se model lakše prilagodio na NYU Depth V2 skup podataka potrebno je skalirati i primjeniti pomak na izlaze modela. Također, izlazi predtreniranog modela su recipročne vrijednosti dubine kako bi se mogla modelirati beskonačna dubina koja se pojavljuje u



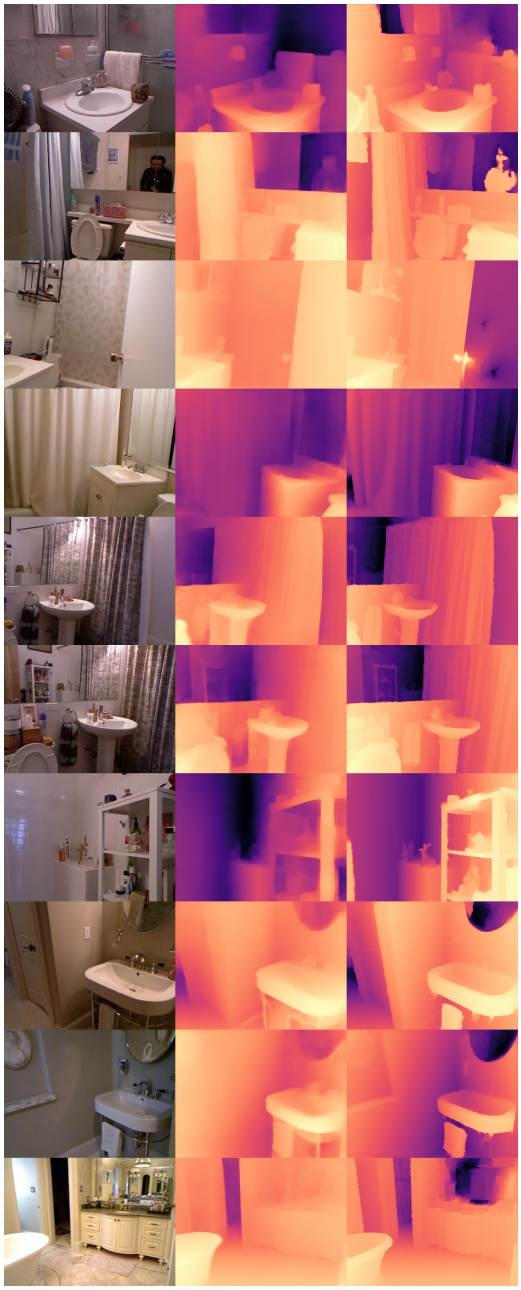
Slika 8.2: Na lijevoj slici je prikazana arhitektura DPT modela koji koriste ViT-Base ili ViT-Large kao ekstraktor značajki. Za razliku od ViT-Hybrid modela koji je korišten u ovom radu, ViT-Base i ViT-Large ne koriste ResNet-50 model već koriste nepreklapajuće isječke ulazne slike rezolucije 16×16 kako je to prikazano na slici. Kako bi se ti isječci mogli provesti kroz ViT model, prvo ih je potrebno izravnati u vektore duljine 16×16. Kada se koristi ViT-Hybrid kao ekstraktor značajki, slika se provodi kroz modificirani ResNet-50 čiji se izlaz može provesti kroz ViT model nakon promjene dimenzionalnosti značajki na dimenzionalnost ViT modela D koristeći jedan konvolucijski sloj. Na srednjoj slici je prikazan sloj ponovnog sastavljanja s ulaznim parametrom s koji obrađuje izlaze na različitim razinama ViT modela. Na desnoj slici je prikazan sloj fuzije koji sjedinjuje izlaze prijašnjeg sloja fuzije i sloja ponovnog sastavljanja.

Mix 6 skupu podataka. Izlazi se tada računaju kao:

$$Y = \frac{1}{Y' \times 0.000305 + 0.1378}, \tag{8.1}$$

gdje je Y' izlaz DPT modela.

Model je fino podešen nad 350000 slika s veličinom mini-grupe 1 i početnom stopom učenja 0.0001 koristeći Adam optimizator. Korišteni gubitak je SILog gubitak objašnjen u 6.2.1. Korišten je *scheduler* koji množi stopu učenja s faktorom 0.5 ako je učenje postiglo plato s minimalnom stopom učenja 10^{-5}. Treniranje je trajalo 80 sati. Rezultati modela na skupu za testiranje su prikazani u tablici 9.1, a izlazne dubinske mape su vizualizirane na slici 8.3.



Slika 8.3: Rezultati DPT modela na skupu za testiranje. U prvom stupcu su ulazne RGB slike, u drugom su dubinske mape na izlazu DPT-a, a u trećem točne dubinske mape. Žuta boja označava dijelove slike koji su bliže, a plava dijelove slike koji su dalje.

9. Usporedba modela i metoda

9.1. Usporedba metrika

Iz tablice 9.1, najbolje rezultate Log10, LRMSE, RMSE i $\delta < 1.25^1$ metrika ima DPT-Hybrid model dok Ladder-DenseNet-201 ima najbolje rezultate RSE, $\delta < 1.25^2$ i $\delta < 1.25^3$ metrika. Iako je DPT model najbolji u nekim metrikama, rezultati odstupaju od onih u radu. Moguć razlog je premali skup podataka za fino podešavanje iako je model predtreniran na *Mix 6* skupu podataka. U ovom radu, korišten je podskup NYU Depth V2 skupa podataka koji se sastoji od 50000 parova RGB slika i dubinskih mapa dok je u originalnom radu model fino podešen na cijelom skupu podataka koji se sastoji od 407024 neobrađenih slika. Modeli koji se temelje na mehanizmu pažnje zahtjeva puno više podataka kako bi postigao bolje rezultate od konvolucijskih modela. Osim količine podataka, drugi nedostatak takvih modela je veliki memorijski otisak tijekom učenja. Zato je još jedan mogući razlog slabijih rezultata prekratko učenje i nedovoljna optimizacija hiperparametara zbog velikih ograničenja memorije.
Ladder DenseNet modeli pokazuju usporedive rezultate metrika s DPT modelom uz puno veću brzinu i puno manji memorijski otisak tijekom učenja kako je pokazano u 9.2. Performanse Ladder DenseNet modela rastu s povećavanjem broja parametara modela do Ladder DenseNet-161 modela koji pokazuje skoro iste rezultate kao i Ladder DenseNet-201 model.

9.2. Usporedba brzine zaključivanja i memorijskog otiska modela

Rezultati brzine zaključivanja i memorijskog otiska modela su prikazani na tablici 9.2. Kao mjera brzine zaključivanja, koristi se broj mini-grupa veličine 1 koje mogu proći kroz model u sekundi (FPS) po uzoru na algoritam korišten u [16]. Cijeli proces pro-

Tablica 9.1: Vrijednosti metrika modela na skupu za testiranje. ↓ označava da je niža vrijednost bolja, a ↑ označava da je viša vrijednost bolja. Podebljane vrijednosti označavaju najbolje rezultate za neku metriku modela koji su uspoređivani u ovom radu. U zadnjem redu su prikazani rezultati trenutne *state-of-the-art* arhitekture BinsFormera na istom skupu za testiranje.

Model	Log10↓	LRMSE↓	RAE↓	RSE↓	RMSE↓	$\delta < 1.25^1 \uparrow$	$\delta < 1.25^2 \uparrow$	$\delta < 1.25^3 \uparrow$
MIMO-UNet	0.079	0.228	0.192	0.168	0.632	0.707	0.927	0.981
Ladder DenseNet-121	0.059	0.172	0.136	0.094	0.495	0.823	0.967	0.992
Ladder DenseNet-121 (*checkpointing*)	0.059	0.172	0.136	0.093	0.495	0.825	0.968	**0.993**
Ladder DenseNet-169	0.058	0.170	0.133	0.091	0.491	0.830	**0.970**	**0.993**
Ladder DenseNet-201	**0.056**	**0.166**	**0.129**	**0.088**	0.480	0.837	**0.970**	**0.993**
Ladder DenseNet-161	**0.056**	**0.166**	0.130	0.090	0.484	0.838	0.968	0.992
DPT-Hybrid	**0.056**	**0.166**	0.131	0.093	**0.476**	**0.841**	0.967	0.990
BinsFormer	0.040	N/A	0.094	N/A	0.330	0.925	0.989	0.997

Tablica 9.2: Metrike brzine zaključivanja i memorijskog otiska za svaki model. FPS predstavlja broj mini-grupa veličine 1 koje je moguće provesti kroz model u sekundi u fazi testiranja i računa se koristeći algoritam korišten u [16].

Model	Broj parametara	FPS @ (320×240)	Max. veličina mini-grupe tijekom treniranja na GeForce GTX 1050
MIMO-UNet	6.8M	9.30	2
Ladder DenseNet-121	9.5M	51.85	16
Ladder DenseNet-169	17.3M	38.72	12
Ladder DenseNet-201	24.3M	31.21	6
Ladder DenseNet-161	34.5M	24.81	6
DPT-Hybrid	123M	9.24	1

laska mini-grupe kroz model se sastoji od kopiranja mini-grupe na GPU, unaprijednog prolaska kroz model te kopiranje rezultata natrag na CPU. Mini-grupe se sastoje od tenzora dimenzija $3 \times 320 \times 240$ koje odgovaraju dimenzijama RGB slika u fazi treniranja. Proces se ponavlja 400 puta te se uzima srednja vrijednost. Sve vrijednosti su računate na istom stroju na kojem su i trenirane. Ladder DenseNet modeli pokazuju puno veći broj FPS-a od MIMO-UNet i DPT modela. Svi Ladder DenseNet modeli pokazuju mogućnost korištenja modela za predikciju u stvarnom vremenu na testiranom stroju.

Kao mjera memorijskog otiska, koristi se maksimalna veličina mini-grupe tijekom testiranja. Kao i u slučaju brzine zaključivanja, Ladder DenseNet pokazuje manji memorijski otisak od MIMO-UNet i DPT modela. Koristeći veće vrijednosti mini-grupa, bolje se iskorištava sloj normalizacije po grupi što rezultira i boljim rezultatima modela.

10. Zaključak

Rezultati metrika potvrđuju hipotezu o visokim performansama modela koji se temelje na mehanizmu pažnje. DPT postiže najbolje vrijednosti nekih metrika, iako su vrijednosti tih metrika usporedive s rezultatima Ladder DenseNet-201 i Ladder DenseNet-161 modela, a MIMO-UNet postiže najslabije rezultate svih metrika. Rezultati DPT modela su niži nego u originalnom radu [17]. Jedno od mogućih objašnjenja, uz manju rezoluciju ulaznih slika, je premali broj podataka za treniranje DPT modela što bi potvrdilo hipotezu o potrebi za velikim količinama podataka modela koji se temelje na mehanizmu pažnje. Uspoređene su i brzine zaključivanja svih modela gdje Ladder DenseNet modeli pokazuju mogućnosti obrade slika u realnom vremenu dok MIMO-UNet i DPT modeli pokazuju puno manje brzine zaključivanja. Također, uspoređene su i maksimalne veličine mini-grupa tijekom treniranja gdje Ladder DenseNet modeli pokazuju puno veću memorijsku efikasnost od MIMO-UNet i DPT modela čime se potvrđuje hipoteza o velikom memorijskom otisku modela koji se temelje na mehanizmu pažnje. MIMO-UNet je predstavljen za zadatak odmućivanja slika i pokazuje se kako takva arhitektura nije idealna za zadatak predikcije dubine. Koristeći mape značajki malih rezolucija, Ladder DenseNet modeli postižu vrlo dobre brzine s visokim performansama na manjim skupovima podataka čime potvrđuju svoju efikasnost. DPT model ima veliki kapacitet i, iako pokazuje visoke performanse, zahtjeva puno više podataka za treniranje kako bi postigao maksimalne performanse.

Za budući rad, modeli bi se mogli trenirati na originalnim rezolucijama kako bi se usporedio utjecaj povećavanja rezolucije na različite modele. U drugoj iteraciji eksperimenata, modeli bi se mogli trenirati na potpunom NYU Depth V2 skupu podataka koji sadržava 8 puta više podataka nego podskup korišten u ovom radu. Mala količina podataka se pokazala kao veliki problem tijekom treniranja DPT modela i taj bi se problem mogao riješiti korištenjem potpunog skupa podataka. Na taj način, uz performanse modela na velikim skupovima podataka, mogla bi se usporediti i efikasnosti modela ovisno o količini podataka. Također, modeli bi se mogli trenirati na strojevima koji imaju veću računalnu moć kako bi se ubrzalo učenje i povećala veličina

mini-grupa tijekom treniranja. Kada bi se ubrzalo učenje, bilo bi lakše optimizirati hiperparametre modela i postići maksimalne performanse za svaku arhitekturu. Povećavanjem mini-grupa, povećala bi se efikasnost slojeva normalizacije po grupi te bi se poboljšala konvergencija optimizacijskog algoritma. Bilo bi zanimljivo usporediti različite varijante opisanih modela poput DPT-Hybrid s modernijim okosnicama kao što su ConvNeXt modeli [13], ali i nove arhitekture poput piramidalnih modela s i bez dijeljenja parametara [15].

LITERATURA

[1] Shariq Farooq Bhat, Ibraheem Alhashim, i Peter Wonka. Adabins: Depth estimation using adaptive bins. *CoRR*, abs/2011.14141, 2020. URL https://arxiv.org/abs/2011.14141.

[2] Christopher M Bishop i Nasser M Nasrabadi. *Pattern recognition and machine learning*, svezak 4. Springer, 2006.

[3] Sung-Jin Cho, Seo-Won Ji, Jun-Pyo Hong, Seung-Won Jung, i Sung-Jea Ko. Rethinking coarse-to-fine approach in single image deblurring. *CoRR*, abs/2108.05054, 2021. URL https://arxiv.org/abs/2108.05054.

[4] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, i Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *CoRR*, abs/2010.11929, 2020. URL https://arxiv.org/abs/2010.11929.

[5] David Eigen, Christian Puhrsch, i Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. *CoRR*, abs/1406.2283, 2014. URL http://arxiv.org/abs/1406.2283.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, i Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL http://arxiv.org/abs/1512.03385.

[7] Dan Hendrycks i Kevin Gimpel. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *CoRR*, abs/1606.08415, 2016. URL http://arxiv.org/abs/1606.08415.

[8] Gao Huang, Zhuang Liu, i Kilian Q. Weinberger. Densely connected convolutional networks. *CoRR*, abs/1608.06993, 2016. URL http://arxiv.org/abs/1608.06993.

[9] Ivan Krešo, Josip Krapac, i Siniša Šegvić. Efficient ladder-style densenets for semantic segmentation of large images, 2019. URL https://arxiv.org/abs/1905.05661.

[10] Anat Levin, Dani Lischinski, i Yair Weiss. Colorization using optimization. U *ACM SIGGRAPH 2004 Papers*, stranice 689–694. 2004.

[11] Hao Li, Zheng Xu, Gavin Taylor, i Tom Goldstein. Visualizing the loss landscape of neural nets. *CoRR*, abs/1712.09913, 2017. URL http://arxiv.org/abs/1712.09913.

[12] Zhenyu Li, Zehui Chen, Xianming Liu, i Junjun Jiang. Depthformer: Exploiting long-range correlation and local information for accurate monocular depth estimation, 2022. URL https://arxiv.org/abs/2203.14211.

[13] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, i Saining Xie. A convnet for the 2020s. *CoRR*, abs/2201.03545, 2022. URL https://arxiv.org/abs/2201.03545.

[14] Pushmeet Kohli Nathan Silberman, Derek Hoiem i Rob Fergus. Indoor segmentation and support inference from rgbd images. U *ECCV*, 2012.

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

[16] Marin Orsic, Ivan Kreso, Petra Bevandic, i Sinisa Segvic. In defense of pre-trained imagenet architectures for real-time semantic segmentation of road-driving images. *CoRR*, abs/1903.08469, 2019. URL http://arxiv.org/abs/1903.08469.

[17] René Ranftl, Alexey Bochkovskiy, i Vladlen Koltun. Vision transformers for dense prediction. *CoRR*, abs/2103.13413, 2021. URL https://arxiv.org/abs/2103.13413.

[18] Olaf Ronneberger, Philipp Fischer, i Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015. URL http://arxiv.org/abs/1505.04597.

[19] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, i Illia Polosukhin. Attention is all you

need. *CoRR*, abs/1706.03762, 2017. URL http://arxiv.org/abs/1706.03762.

[20] Weihao Yuan, Xiaodong Gu, Zuozhuo Dai, Siyu Zhu, i Ping Tan. New crfs: Neural window fully-connected crfs for monocular depth estimation, 2022. URL https://arxiv.org/abs/2203.01502.

Eksperimentalno vrednovanje metoda za monokularnu predikciju dubine

Sažetak

Do nedavno, konvolucijski modeli su dominirali područjem računalnog vida koristeći preskočne veze i U-net strukturu. U području obrade prirodnog jezika je predstavljen transformer, model koji se temelji na mehanizmu pažnje te ostvaruje rezultate stanja tehnike. U području računalnog vida, inspiriran transformer modelom, predstavljen je vision transformer model koji ostvaruje rezultate stanja tehnike tom području. Najveći nedostaci modela koji se temelje na mehanizmu pažnje su potreba za vrlo velikom količinom podataka kako bi postigli maksimalne performanse te veliki memorijski otisak tijekom učenja. Kroz zadatak monokularne predikcije dubine, uspoređuju se dvije vrste konvolucijskih modela, MIMO-Unet i Ladder DenseNet, te jedan model koji se temelji na mehanizmu pažnje, DPT-Hybrid. Modeli su trenirani na podskupu skupa NYU Depth V2 koji se sastoji od 50000 parova RGB slika i dubinskih mapa za treniranje te 654 parova za testiranje. U radu se uspoređuju vrijednosti 8 različitih metrika za performanse modela na testnom skupu te brzine zaključivanja i memorijski otisak svakog od modela.

Ključne riječi: Računalni vid, gusta predikcija, monokularna predikcija dubine, konvolucijski modeli, preskočne veze, U-net struktura, modeli koji se temelje na mehanizmu pažnje, MIMO-UNet, Ladder DenseNet, DPT-Hybrid, NYU Depth V2.

Experimental evaluation of methods for monocular depth prediction

Abstract

Until recently, convolutional models dominated the field of computer visions using residual connections and U-net structure. In the field of natural language processing, a transformer, model based on the attention mechanism, was introduced and achieved state-of-the-art results. In the field of computer vision, inspired by the transformer model, a vision transformer model was presented that achieves state-of-the-art results in that field. The biggest disadvantages of attention-based models are the need for a very large amount of data to achieve maximum performance and big memory footprint during training. Through the task of monocular depth estimation, two types of convolutional models are compared, MIMO-Unet and Ladder DenseNet, and one attention-based model, DPT-Hybrid. The models were trained on a subset of the NYU Depth V2 dataset consisting of 50000 pairs of RGB images and depth maps for training and 654 pairs for testing. This thesis compares the values of 8 different metrics for the performance of models on the test set and the speed of inference and memory footprint for each of the models.

Keywords: Computer vision, dense prediction, monocular depth estimation, convolutional models, residual connections, U-net structure, attention-based models, MIMO-UNet, Ladder DenseNet, DPT-Hybrid, NYU Depth V2.