SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 6346

Konvolucijski modeli za jednooku predikciju dubine scene

Filip Oreč

Zagreb, lipanj 2019.

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

Hvala

SADRŽAJ

1. Uvod

Predikcija dubine scene je još davno poznati problem. Konvencionalni prikazi kao što su slika ili video prikazuju trodimenzionalni svijet u dvije dimenzije. Naime, time gubimo informaciju o trećoj dimenziji koja sadrži informaciju o dubini scene. Iako je dvodimenzionalna reprezentacija dovoljna u većini primjena, nekad nam je potrebna trodimenzionalna reprezentacija. Percepcija dubine proizlazi iz raznih znakova ili naznaka o dubini. Obično se dijele na binokularne znakove koji sadrže informacije u tri dimenzije i mogu se vidjeti s dva oka i monokularne znakove koji sadrže informacije u dvije dimenzije i mogu se vidjeti s jednim okom.

Tijekom godina razvile su se razne tehnike za predikciju dubine scene poput stereoskopije koja se oslanja na binokularne znakove. Korištenjem dvije slike iste scene koje su dobivene iz malo drugačijih kuteva, moguće je izračunati udaljenost od objekta. Aplikacijama koje uključuju razumijevanje scene, 3D modeliranje i slično jako je bitna informacija o dubini kad gore navedena organizacija podataka i tehnike nisu dostupni. U tom slučaju koristi se jednooka predikcija dubine scene koja predstavlja loše postavljen (engl. *ill-posed*) problem, jer iz jedne dvodimenzionalne slike može nastati beskonačno mnogo različitih trodimenzionalnih scena. Za ljude ovo ne predstavlja veliki problem, jer možemo jako dobro iskoristiti monokularne znakove, ali za računala predstavlja ogroman problem za riješiti s velikom preciznosti i malom uporabom resursa. Zbog navedenog razloga te jednostavnosti primjene, u zadnje vrijeme se sve češće upotrebljavaju konvolucijske neuronske mreže koji uče odnos između piksela boje i dubine, što je i predmet ovog rada. Detaljnije ću obraditi konvolucijsku neuronsku mrežu predloženu u [3], te ju implementirati i primijeniti na problem jednooke predikcije dubine scene.

U drugom poglavlju objašnjeni su osnovni pojmovi vezani uz umjetne neuronske mreže, te osnovni algoritmi koji se koriste prilikom njihovog učenja. Treće poglavlje opisuje konvolucijske neuronske mreže te zašto su one bitne. U sklopu trećeg poglavlja još se opisuju rezidualne neuronske mreže i koje probleme one rješavaju. Četvrto poglavlje bavi se detaljima implementacije predložene arhitekture, te se ukratko opi-

suje korišteni programski okvir PyTorch. U petom poglavlju su opisani rezultati koje je model ostvario.

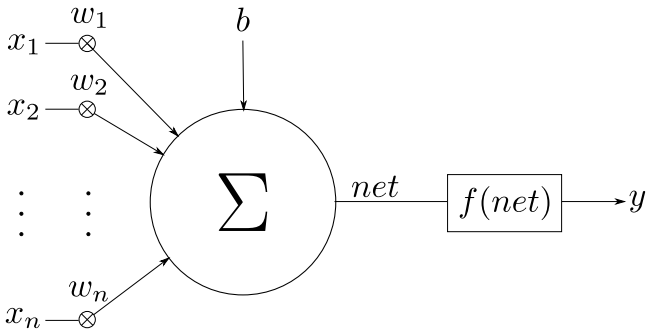
2. Umjetna neuronska mreža

Nastanak umjetnih neuronskih mreža u početku je bio inspiriran biološkim neuronima i neuronskim mrežama, ali daljnjim razvojem su se odvojile od biološke povezanosti i postale stvar inženjerstva. Definiraju se procesne jedinice zvane umjetni neuroni koji se međusobno povezuju te grade umjetne neuronske mreže.

Ovaj rad se bavi samo unaprijednim neuronskim mrežama (engl. *feedforward neural networks*). Cilj unaprijedne neuronske mreže je aproksimirati neku funkciju f^*, tako da definira preslikavanje $\mathbf{y} = f(\mathbf{x}; \boldsymbol{\theta})$ i nauči vrijednost parametara $\boldsymbol{\theta}$ kako bi najbolje aproksimirala ciljanu funkciju f^*. Ovakvi modeli zovu se unaprijedni jer nema veza koje izlaze iz modela vraćaju nazad na ulaze.

2.1. Umjetni neuron

Warren McCulloch i Walter Pitts 1943. godine konstruirali su matematički model neurona kakav je prikazan na slici 2.1.



Slika 2.1: Umjetni neuron

Umjetni neuron prima više ulaza $x_1, x_2, ..., x_n$ koji tvore ulazni vektor $\mathbf{x}$. Taj ulazni vektor može biti stvarni ulaz ili izlaz iz nekog prethodnog neurona. Za svaku vrijednost

x_i ulaznog vektora $\mathbf{x}$ postoji vrijednost w_i koju nazivamo težina (eng. *weight*). To je vrijednost koja predstavlja utjecaj ulaza x_i na neuron. Svaki ulaz x_i množi se s odgovarajućom težinom w_i što daje umnožak $x_i \cdot w_i$. Vrijednosti $w_1, w_2, ..., w_n$ tvore vektor težina $\mathbf{w}$. Još se definira i vrijednost b koja označava pomak (engl. *bias*). Sve primljene vrijednosti se sumiraju prema izrazu 2.1.

$$net = \sum_{i=1}^{n} x_i \cdot w_i + b \tag{2.1}$$

Izraz 2.1 može se zapisati u matričnom obliku:

$$net = \mathbf{w}^\top \cdot \mathbf{x} + b \tag{2.2}$$

Pri tome ulazni vektor $\mathbf{x}$ i vektor težina $\mathbf{w}$ imaju dimenzije N$\times$1, gdje N predstavlja broj ulaza u neuron, dok je pomak b skalar. Rezultat sumiranja net dalje se predaje kao ulaz prijenosnoj funkciji koja određuje konačni izlaz o neurona prema izrazu 2.3.

$$y = f(net) = f(\mathbf{w}^\top \cdot \mathbf{x} + b) \tag{2.3}$$

Gdje f predstavlja prijenosnu funkciju.

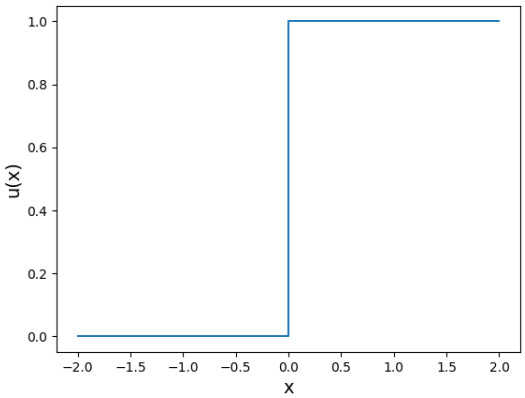
2.1.1. Prijenosne funkcije

Glavna zadaća prijenosne funkcije je pretvorba ulaznih vrijednosti u izlaznu vrijednost. Pri tome se mogu koristiti različite vrste funkcija ovisno o arhitekturi mreže. Danas postoji više prijenosnih funkcija koje se češće koriste.

Prva od njih je funkcija skoka i definirana je izrazom 2.4.

$$u(x) = \begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases} \tag{2.4}$$

Bitno svojstvo ove funkcije je prekid u točki $x = 0$, te zbog toga nije diferencijabilna u toj točki. Naime, to svojstvo onemogućava korištenje algoritama učenja koji se temelje na gradijentu. Izgled funkcije prikazan je na slici 2.2.

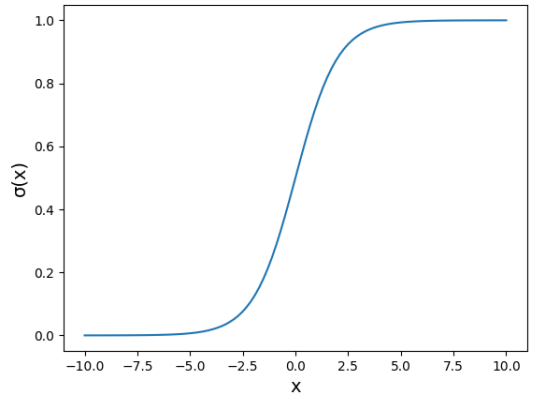


Slika 2.2: Funkcija skoka

Sigmoidalna funkcija ima jako dobra svojstva te se zbog toga često primjenjuje. Smješta bilo koji realni broj na interval $[0, 1]$. Definirana je izrazom 2.5.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.5}$$

Ova funkcija ima dosta sličnosti s funkcijom skoka. Naime, ako je x neki veliki pozitivan broj onda je $e^{-x} \approx 0$, i izlaz iz neurona je blizu 1. S druge strane ako je x neki veliki negativni broj onda $e^{-x} \to \infty$, i izlaz iz neurona je blizu 0. To se jako dobro može primijetiti na grafu funkcije prikazanom na slici 2.3.



Slika 2.3: Sigmoidalna funkcija

Velika prednost sigmoidalne funkcije u odnosu na funkciju skoka je činjenica da je derivabilna, što omogućava korištenje algoritama učenja koji se temelje na gradijentu. Derivacija ove funkcije dana je izrazom 2.6.

$$\frac{d\sigma(x)}{dx} = \sigma(x) \cdot (1 - \sigma(x)) \tag{2.6}$$

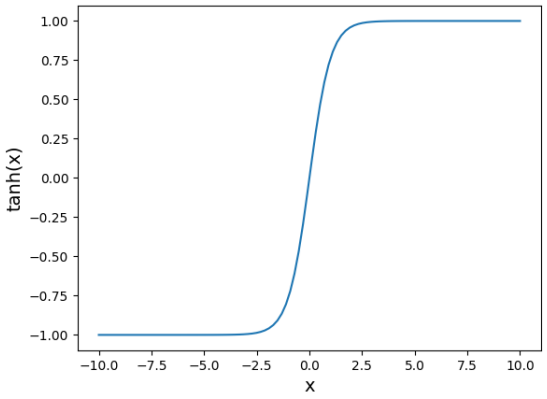
Na krajevima funkcija jako slabo reagira na promjene i zato će gradijent u tim dijelovima imati male vrijednosti. Ako mreža uči pomoću lokalnih gradijenata, u tom slučaju učenje staje, jer gradijenti ne mogu napraviti značajne promjene.

Funkciju tangens hiperbolni moguće je dobiti izravno iz sigmoidalne funkcije. Definicija je dana izrazom 2.7.

$$tanh(x) = 2\sigma(2x) - 1 \tag{2.7}$$

Ova prijenosna funkcija smješta bilo koji realni broj na interval $[-1, 1]$. Prednost je što je derivabilna, ali kao i sigmoidalna funkcija na krajevima jako slabo reagira na promjene, te ako je vrijednost ulaza u funkciju u tom području, učenje staje, što se može vidjeti na grafu funkcije 2.4. Derivacija funkcije dana je izrazom 2.8.

$$\frac{dtanh(x)}{dx} = 1 - tanh^2(x) \tag{2.8}$$



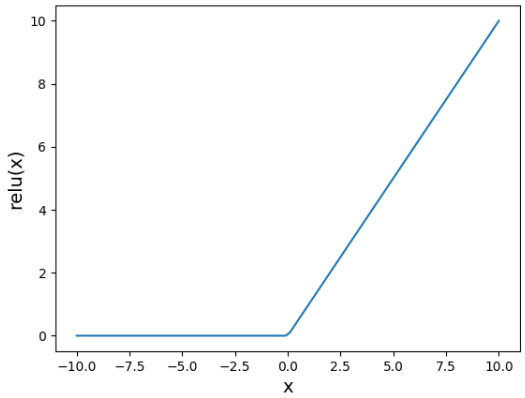
Slika 2.4: Tangens hiperbolni

Zadnjih nekoliko godina sve se češće koristi ReLU (*Recitified Linear Unit*) prijenosna funkcija. Određena je izrazom 2.9.

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

Ova prijenosna funkcija sve vrijednosti veće od 0 propušta, dok vrijednosti manje od 0 uopće ne propušta, što se može vidjeti na grafu funkcije 2.9.
Derivacije ove funkcije dana je izrazom 2.10.

$$\frac{drelu(x)}{dx} = \begin{cases} 1, & x > 0 \\ 0, & x \leq 0 \end{cases} \tag{2.10}$$

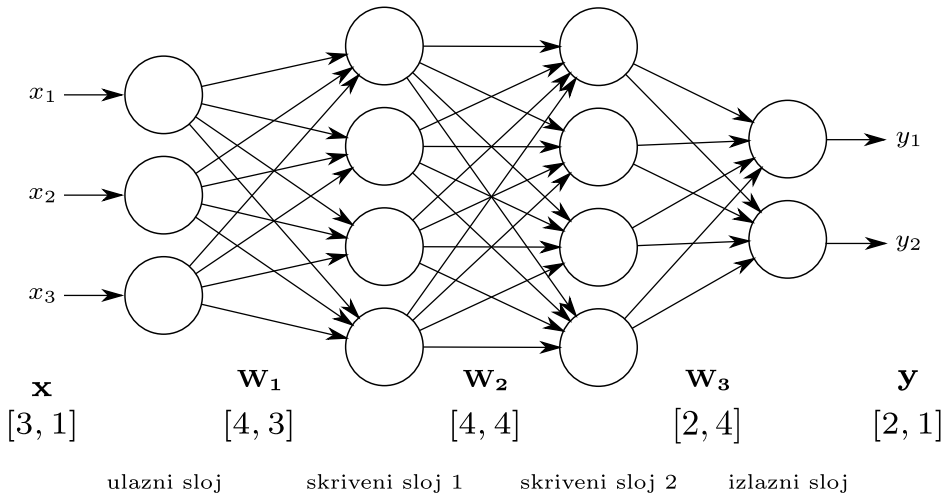


Slika 2.5: ReLU

Razlog zbog kojeg se sve češće primjenjuje je brzina računanja. Ali i ova prijenosna funkcija ima nedostatak. Iz izraza 2.10 se vidi da je za sve negativne brojeve derivacija jednaka 0, što predstavlja problem za algoritme učenja temeljene na gradijentu. Neuron kojemu je suma ulaza, odnosno net negativan, se zbog toga neće moći prilagođavati.

2.2. Arhitektura umjetne neuronske mreže

Umjetna neuronska mreža sastoji se od više umjetnih neurona koji su međusobno povezani i grupirani u slojeve. Na slici 2.6 je prikazana neuronska mreža koja ima četiri sloja. Prvi sloj je ulazni i sastoji se od tri neurona, a zadnji sloj je izlazni i sastoji se od dva neurona. Svaki sloj koji se nalazi između ulaznog i izlaznog zove se skriveni sloj.



Slika 2.6: Umjetna neuronska mreža

Ovakva mreža zove se još i potpuno povezana mreža, jer je izlaz iz svakog neurona u jednom sloju povezan sa svim neuronima u idućem sloju. Arhitektura neuronske mreže opisuje kompoziciju funkcija. Na primjer, mrežu sa slike 2.6 možemo opisati kompozicijom funkcija $f(\mathbf{x}) = f^{(3)}(f^{(2)}(f^{(1)}(\mathbf{x})))$, gdje $f^{(1)}$ predstavlja prvi skriveni sloj, $f^{(2)}$ drugi skriveni sloj, a $f^{(3)}$ izlazni sloj. Pri tome su funkcije $f^{(1)}$, $f^{(2)}$, $f^{(3)}$ definirane izrazima:

$$\mathbf{h_1} = f^{(1)}(\mathbf{x}; \mathbf{W_1}, \mathbf{b_1}) = g_1(\mathbf{W_1} \cdot \mathbf{x} + \mathbf{b_1}) \tag{2.11}$$

$$\mathbf{h_2} = f^{(2)}(\mathbf{h_1}; \mathbf{W_2}, \mathbf{b_2}) = g_2(\mathbf{W_2} \cdot \mathbf{h_1} + \mathbf{b_2}) \tag{2.12}$$

$$\mathbf{y} = f^{(3)}(\mathbf{h_2}; \mathbf{W_3}, \mathbf{b_3}) = g_3(\mathbf{W_3} \cdot \mathbf{h_2} + \mathbf{b_3}) \tag{2.13}$$

Gdje g_1, g_2 i g_3 predstavljaju prijenosne funkcije u pojedinim slojevima, a matrice $\mathbf{W_1}$, $\mathbf{W_2}$ i $\mathbf{W_3}$ predstavljaju matrice težina pojedinih slojeva. Vektori $\mathbf{b_1}$, $\mathbf{b_2}$ i $\mathbf{b_3}$ predstavljaju pomak za svaki sloj.

2.3. Učenje umjetne neuronske mreže

Učenje neuronske mreže je proces nalaženja optimalnih parametara $\boldsymbol{\theta}$, kako bi neuronska mreža što bolje aproksimirala ciljnu funkciju. Ciljna funkcija se zaključuje na osnovu uzoraka iz skupa za učenje koji se predočavaju neuronskoj mreži. Svaki uzorak je par koji se sastoji od ulaza u mrežu i željenog izlaza. Ovakav način učenja se zove nadzirano učenje (engl. *supervised learning*).

Kako bi odredili koliko dobro mreža aproksimira ciljnu funkciju koristimo funkciju gubitka. Optimiziranjem funkcije gubitka neuronska mreža se uči.

2.3.1. Funkcija gubitka

Kako bi se moglo ostvariti učenje, potrebno je definirati funkciju gubitka, koja ovisi o parametrima $\boldsymbol{\theta}$, tj. o težinama $\mathbf{W}$ i pomacima $\mathbf{b}$. Kao i prijenosna funkcija, funkcija gubitka se može definirati na više načina.

Najčešće funkcije gubitka za regresijske probleme su $\mathcal{L}_1$ i $\mathcal{L}_2$ funkcije gubitka. Definirane su sljedećim izrazima:

$$\mathcal{L}_1 = \sum_{i=0}^{n} |y_i - \hat{y}_i| \tag{2.14}$$

$$\mathcal{L}_2 = \sum_{i=0}^{n} (y_i - \hat{y}_i)^2 \tag{2.15}$$

gdje y_i predstavlja ispravnu vrijednost, $\hat{y}_i = f(\mathbf{x})$ procijenjenu vrijednost, a n broj podataka.

Za probleme klasifikacije se $\mathcal{L}_1$ i $\mathcal{L}_2$ funkcije gubitka ne koriste toliko često. Za ovaj problem se pretežito koristi unakrsna entropija (engl. *cross- entropy loss*). Definirana je izrazom 2.16.

$$\mathcal{L}_{CE} = -\sum_{i=0}^{n} y_i \cdot log(\hat{y}_i) \tag{2.16}$$

2.3.2. Gradijentni spust

Optimiziranjem funkcije gubitka parametri neuronske mreže se "štimaju" i mreža uči. Za optimiziranje funkcije gubitka koristi se gradijentni spust. To je iterativni optimizacijski algoritam za pronalaženje minimuma funkcije.

Ako je funkcija $y = f(x)$, gdje su x i y realni brojevi, funkcija koja se optimizira, onda je derivacija te funkcije $f'(x) = \frac{dy}{dx}$. Tada je aproksimacija funkcije $f(x)$ Taylorovim razvojem prvog reda:

$$f(x + \Delta x) \approx f(x) + f'(x) \cdot \Delta x \tag{2.17}$$

Ako se u tu aproksimaciju uvrsti pomak u smjeru gdje funkcija pada, odnosno u smjeru negativne derivacije dobiva se:

$$f(x - \epsilon \cdot sign(f'(x))) \approx f(x) - f'(x) \cdot \epsilon \cdot sign(f'(x)) \tag{2.18}$$

Vidi se da je $f(x - \epsilon \cdot sign(f'(x)))$ manje od $f(x)$ za dovoljno mali ϵ. Dakle, ako se x iterativno pomiče malim koracima sa suprotnim predznakom derivacije, može se doći do minimuma funkcije.

Za funkcije više varijabli koriste se parcijalne derivacije i gradijent funkcije. Parcijalna derivacija $\frac{\partial}{\partial x_i} f(\mathbf{x})$ određuje koliko se f promijeni ako se jedino x_i poveća u točki $\mathbf{x}$. Gradijent funkcije $\nabla f(\mathbf{x})$ je vektor koji sadrži sve parcijalne derivacije funkcije. Dan je izrazom 2.19.

$$\nabla f(\mathbf{x}) = \left[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \cdots, \frac{\partial f}{\partial x_n} \right] \tag{2.19}$$

Aproksimacija funkcije više varijabli Taylorovim razvojem prvog reda je dana sljedećim izrazom:

$$f(\mathbf{x} + \Delta \mathbf{x}) \approx f(\mathbf{x}) + \nabla f(\mathbf{x})^\top \Delta \mathbf{x} \tag{2.20}$$

Ako se u danu aproksimaciju uvrsti pomak u smjeru najbržeg pada funkcije, odnosno u smjeru negativnog gradijenta funkcije dobiva se:

$$f(\mathbf{x} - \epsilon \cdot \nabla f(\mathbf{x})) \approx f(\mathbf{x}) - \epsilon \cdot \nabla f(\mathbf{x})^\top \nabla f(\mathbf{x}) \tag{2.21}$$

gdje ϵ predstavlja mali pozitivan parametar koji se češće zove korak učenja. Kako je $\nabla f(\mathbf{x})^\top \nabla f(\mathbf{x}) \geq 0$, funkcija $f(\mathbf{x})$ će se smanjivati sve dok ne dođe do minimuma gdje je $\nabla f(\mathbf{x}) = \mathbf{0}$. Za sljedeću iteraciju $\mathbf{x}$ se ažurira izrazom 2.22 i postupak se ponavlja za točku $\mathbf{x}'$.

$$\mathbf{x}' = \mathbf{x} - \epsilon \cdot \nabla f(\mathbf{x}) \tag{2.22}$$

Ako funkcija f predstavlja funkciju gubitka $\mathcal{L}$ onda se svakim korakom gradijentnog spusta ažuriraju se parametri neuronske mreže izrazima:

$$\mathbf{w}' = \mathbf{w} - \epsilon \cdot \nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}, \mathbf{b}) \tag{2.23}$$

$$\mathbf{b}' = \mathbf{b} - \epsilon \cdot \nabla_{\mathbf{b}} \mathcal{L}(\mathbf{w}, \mathbf{b}) \tag{2.24}$$

Ako je skup podatak za učenje velik, računanje gradijenta za sve podatke odjednom može biti sporo. Postupak se može ubrzati korištenjem stohastičkog gradijentnog spusta. Korak gradijentnog spusta se računa samo za dio skupa za učenje (engl. *batch*). Prolazak kroz cijeli skup za učenje naziva se epoha. Ovaj način također omogućava i bolje izbjegavanje lokalnih minimuma.

2.3.3. Optimizacija

Učenje neuronske mreže gradijentnim spustom može se ubrzati raznim optimizacijskim metodama.

Jedna od tih metoda je momentum. Definira se novi vektor $\mathbf{v}$, te se mijenja jednadžba za ažuriranje parametara gradijentnog spusta na sljedeći način:

$$\begin{aligned} \mathbf{v}' &= \alpha \mathbf{v} - \epsilon \nabla \mathcal{L} \\ \mathbf{w}' &= \mathbf{w} + \mathbf{v}' \end{aligned} \tag{2.25}$$

gdje je $\alpha \in [0, 1)$ hiperparametar koji modelira prigušenje koje omogućuje zaustavljanje u minimumu. Najčešće se inicijalizira na 0.9. Vektor $\mathbf{v}$ predstavlja brzinu kretanja prema minimumu, što je gradijent veći, brzina će se sve više povećavati.

2.3.4. Regularizacija

Neuronska mreža mora imati dobre rezultate izvođenja ne samo na skupu podataka za učenje, nego i na novim podatcima. Razvijene su razne metode kako bi se smanjila pogreška na novim podatcima, pa čak i ako se poveća pogreška na skupu za učenje. Ove metode poznate su kao regularizacija.

Jedna od metoda je dodavanje funkciji gubitka dodatni član:

$$\mathcal{L}_r = \mathcal{L} + \lambda N(w) \tag{2.26}$$

gdje je $\mathcal{L}$ gubitak, a λ je hiperparametar koji određuje utjecaj regularizacijskog člana. Ako regularizacijski član odgovara L^1-normi ili L^2-normi, onda se ova metoda zove L^1-regularizacija, odnosno L^2-regularizacija. U tom slučaju funkcija gubitka se može ovako napisati:

$$\mathcal{L}_r = \mathcal{L} + \lambda \sum_i |w_i| \tag{2.27}$$

$$\mathcal{L}_r = \mathcal{L} + \lambda \sum_i w_i^2 \tag{2.28}$$

gdje se jednadžba 2.27 odnosi na L^1-regularizaciju, a 2.28 na L^2 regularizaciju. Posljedica dodavanja regularizacijskog člana je preferiranje učenja manjih težina.

Sljedeća metoda je dropout, koja se dosta razlikuje od L^1 i L^2 regularizacije. Ova metoda se ne oslanja na izmjenjivanje funkcije gubitka nego na izmjenjivanje same mreže. Svaki neuron u skrivenom sloju s određenom vjerojatnošću privremeno postaje isključen tijekom učenja. Tako se sprječava da utjecaj nekih neurona postane prevelik i tako uzrokuje prenaučenost.

Povećavanje skupa za učenje je još jedan način kako smanjiti prenaučenost. Dobivanje novih podataka za učenje može biti skupo i nije uvijek moguće. Zato se postojeći skup podataka za učenje može proširiti raznim transformacijama. To mogu biti razne transformacije poput rotacije, zrcaljenja, skaliranja, izmjene svjetline i kontrasta.

Još jedan oblik regularizacije je normalizacija nad grupama (engl. *batch-normalization*). Izlaz iz svakog sloja se normalizira. Ako je μ_B aritmetička sredina, a σ_B^2 varijanca podataka mini grupe B, tada za sloj s ulazom $x = (x^{(1)}, ..., x^{(d)})$ se primjenjuje normalizacija:

$$\hat{x}^{(k)} = \frac{x^{(k)} - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \tag{2.29}$$

Bez normalizacije male promjene u prvim slojevima će se povećati kako se propagiraju kroz mrežu, što rezultira velikim promjenama u dubljim slojevima. Metoda normalizacije nad grupama smanjiva ovakve neželjene promjene i tako ubrzava proces treniranja. Kako ova metoda koristi srednju vrijednost i varijancu grupe koje su svaku epohu različite, model ne trenira na istim podatcima i tako se postiže regularizacija.

2.3.5. Propagacija pogreške unatrag

Propagacija pogreške unatrag (engl. *back-propagation*) je algoritam za učinkovito računanje gradijenata funkcije putem rekurzivne primjene pravila ulančavanja. Dakle, za neku funkciju $f(\mathbf{x})$ gdje je $\mathbf{x}$ ulazni vektor, algoritam propagacije pogreške unatrag

računa gradijent funkcije f u točki $\mathbf{x}$, tj. $\nabla f(\mathbf{x})$. Za slučaj učenja neuronske mreže gradijentnim spustom, funkcija f će odgovarati funkciji gubitka $\mathcal{L}$, a ulazni vektor $\mathbf{x}$ će se sastojati od podataka za učenje, težina $\mathbf{W}$ i pomaka $\mathbf{b}$ neuronske mreže.

U konačnici, to znači računanje parcijalnih derivacija $\partial\mathcal{L}/\partial w_{jk}^l$ i $\partial\mathcal{L}/\partial b_j^l$, gdje w_{jk}^l označava težinu koja povezuje k-ti neuron u $(l-1)$-om sloju s j-tim neuronom u l-tom sloju. Slično, b_j^l označava pomak j-tog neurona u l-tom sloju. Za računanje ovih parcijalnih derivacija uvodi se pogreška δ_j^l neurona j u sloju l:

$$\delta_j^l = \frac{\partial\mathcal{L}}{\partial net_j^l} \tag{2.30}$$

gdje net_j^l predstavlja izlaz iz neurona prije primjene aktivacijske funkcije definiran jednadžbom 2.1. Za pogrešku neurona δ_j^L u izlaznom sloju vrijedi:

$$\delta_j^L = \frac{\partial\mathcal{L}}{\partial y_j^L}\frac{\partial y_j^L}{\partial net_j^L} = \frac{\partial\mathcal{L}}{\partial y_j^L}f'(net_j^L) \tag{2.31}$$

gdje y_j^L predstavlja izlaz iz neurona, a f prijenosnu funkciju definirane jednadžbom 2.3. Jednadžba 2.31 može se zapisati u matričnom obliku:

$$\delta^L = \nabla_y\mathcal{L} \odot f'(net^L) \tag{2.32}$$

gdje $\odot$ predstavlja umnožak matrica gdje se elementi na istim mjestima množe. Primjenom ove jednadžbe mogu se dobiti pogreške samo za neurone izlaznog sloja, ali poznavanjem pogrešaka nekog sloja mogu se dobiti pogreške prethodnog sloja. Pogreške skrivenih slojeva se izračunavaju prema izrazu:

$$\delta^l = ((w^{l+1})^\top\delta^{l+1}) \odot f'(net^l) \tag{2.33}$$

Jednadžbama 2.32 i 2.33 može se izračunati pogreška δ^l za svaki sloj mreže. Prvo se izračuna pogreška izlaznog sloja δ^L pomoću jednadžbe 2.32, zatim se pomoću jednadžbe 2.33 izračuna pogreška δ^{L-1}, zatim pogreška δ^{L-2} i sve tako do ulaznog sloja mreže.

Parcijalna derivacija s obzirom na pomak odgovara upravo pogrešci neurona:

$$\frac{\partial\mathcal{L}}{\partial b_j^l} = \delta_j^l \tag{2.34}$$

dok je parcijalna derivacija s obzirom na težinu neurona:

$$\frac{\partial\mathcal{L}}{\partial w_{jk}^l} = y_k^{l-1}\delta_j^l \tag{2.35}$$

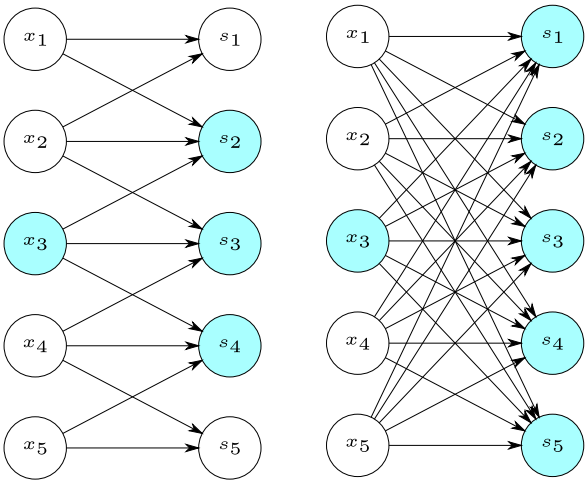
Jednadžbama 2.34 i 2.35 izračunava se ovisnost funkcije gubitka o pojedinim parametrima, što je i cilj algoritma propagacije pogreške unatrag.

3. Konvolucijska neuronska mreža

Konvolucijske neuronske mreže (engl. *convolutional neural networks*) su posebna vrsta neuronskih mreža za obradu podataka koji imaju rešetkastu (engl. *grid-like*) topologiju. Najbolji primjer su slikovni podatci, koji se mogu smatrati kao 2D rešetka piksela.

Kod tradicionalnih potpuno povezanih neuronskih mreža izlaz iz svakog neurona u jednom sloju je povezan sa svakim neuronom u sljedećem sloju. Za slike dimenzija HxWxC ulaz u mrežu je dimenzija $H \cdot W \cdot C$ što znači da će samo jedan neuron u prvom skrivenom sloju imati $H \cdot W \cdot C$ težina. Neuronska mreža u većini slučajeva ima više od jednog neurona u skrivenom sloju te će doći do pojave previše parametara.

Konvolucijske neuronske mreže imaju prorijeđenu interakciju između neurona, što znači da je potrebno pohraniti manje parametara i da izračunavanje izlaza zahtjeva manje operacija. Na slici 3.1 je označen neuron x_3 i izlazni neuroni s koji su povezani s tim neuronom. Lijevo je prikazana prorijeđena interakcija, a desno potpuno povezana.

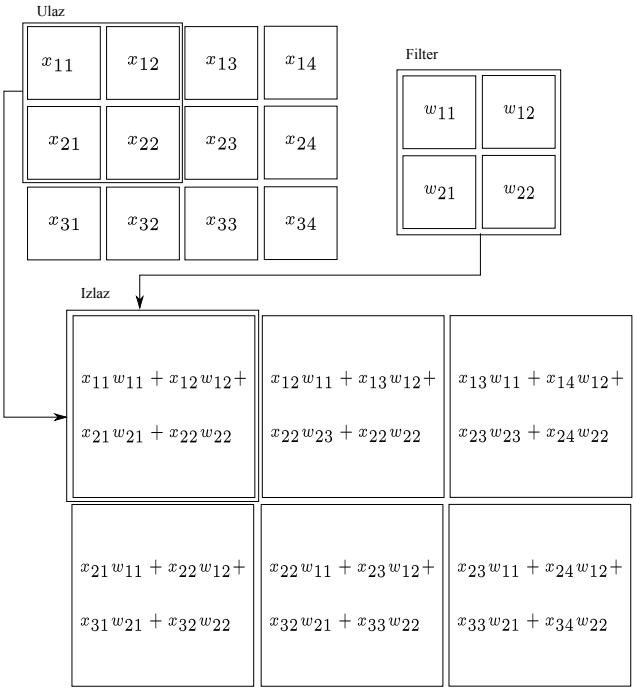


Slika 3.1: Primjer prorijeđene interakcije neurona

Slojevi konvolucijske neuronske mreže su organizirani u tri dimenzije: visina, dužina i dubina. Neuroni u pojedinom sloju su povezani samo s malim dijelom neurona

u prethodnom sloju, a ne sa svim neuronima kao kod potpuno povezanih slojeva. Konvolucijske neuronske mreže grade se od tri osnovna sloja: konvolucijski sloj, sloj sažimanja i potpuno povezani sloj.

Konvolucijski sloj je temeljni sloj konvolucijske neuronske mreže. Parametri ovog sloja sastoje se od skupa filtera koji se mogu naučiti. Filteri su prostorno dosta manjih dimenzija od ulaza u sloj. Izlazi se izračunavaju tako da se primjenjuje dvodimenzionalna konvolucija ulaza s filterom čime se dobiva dvodimenzijska mapa značajki. Na slici 3.2 filter je dimenzija 2x2, dok je ulaz dimenzija 3x4. Filter se pomjera po dužini i visini ulaza i izračunava se skalarni produkt između članova filtera s članovima ulaza na odgovarajućim pozicijama. Na kraju se dobiva mapa značajki s dimenzijama 2x3. Konvolucijski slojevi obično imaju više filtera, i svaki od njih će izračunati zasebnu



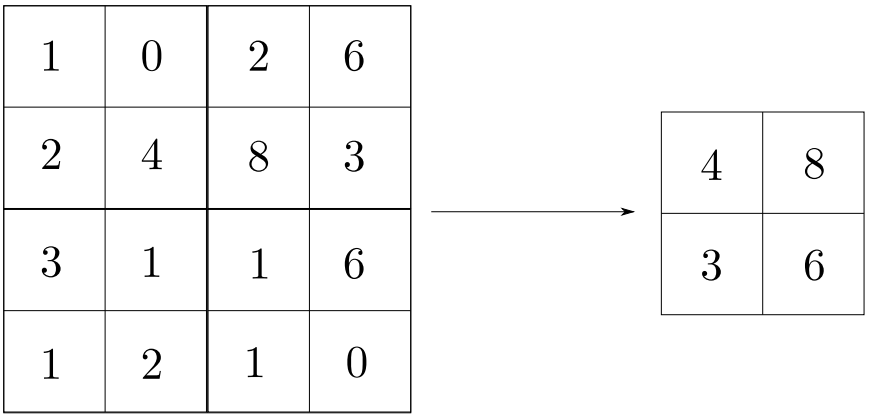
Slika 3.2: Konvolucija ulaza i filtera

mapu značajki koje se slažu po dubini i tako stvaraju izlaz. Na dobivene izlaze još se primjenjuje prijenosna funkcija. Broj neurona koji su povezani s neuronom u sljedećem sloju zove se receptivno polje neurona i odgovara upravo veličini filtera.

Kako bi izlazne mape značajki imale istu visinu i dužinu kao i ulaz ponekad je potrebno nadopuniti ulaz kako filter prilikom konvolucije ne bi prelazio rubove ulaza. Najčešće se koristi nadopunjavanje nulama. Filter se ne mora uvijek micati samo po jedan neurona, ali ako se miće po više neurona odjednom dimenzija izlaza će se smanjiti. Broj za koji se filter miće zove se korak (engl. *stride*).

Sloj sažimanja je sloj koji smanjiva prostorne dimenzije (visinu i dužinu) ulaza

kako bi se smanjio broj parametara neuronske mreže. Primjenjuje se nezavisno na svakoj od mapi značajki. Sažimanje se može ostvariti na više načina, ali se najčešće koristi sažimanje maksimalnom vrijednošću (engl. *max-pooling*). Odabire se veličina i korak filtera, te se primjenjuje funkcija sažimanja na područje koje odgovara trenutnoj poziciji filtera. Na slici 3.3 je prikazano sažimanje maksimalnom vrijednošću s veličinom filtera 2x2 i korakom 2. Ovaj sloj ne uvodi dodatne parametre u mrežu.



Slika 3.3: Sažimanje maksimalnom vrijednošću s filterom veličine 2x2 i korakom 2

3.1. Rezidualna neuronska mreža

Duboke konvolucijske neuronske mreže su se pokazale jako učinkovite pri rješavanju problema iz područja računalnog vida. Također se pokazalo da je dubina neuronske mreže jako važna. Ali nakon određenog broja slojeva dolazi do problema nestajućih i eksplodirajućih gradijenata što sprječava učenje od početka. Ovaj problem se u velikoj mjeri rješava normaliziranim inicijaliziranjem i normalizacijskim slojevima, što omogućava mrežama s desetak slojeva konvergirati. Međutim, pojavio se problem degradacije: kako se dubina mreže povećava točnost mreže počinje naglo opadati.

Kao rješenje navedenog problema razvile su se rezidualne neuronske mreže koje koriste rezidualne blokove definirane u [2]. Ovi blokovi uvode nove veze u slojevima zvane prečice koje preskaču jedan ili više slojeva i zbrajaju se s izlazom zadnjeg preskočenog sloja. Kako se izlaz iz prečice zbraja sa izlazom preskočenih slojeva dimenzije im se moraju podudarati.

Rezidualni blokovi rješavaju problem degradacije, te dodavanjem novih slojeva pospješuju se rezultati, ali se povećava broj parametara. Za mreže s 50 slojeva i više koriste se rezidualni blokovi s uskim grlom, kako bi smanjili broj parametara. Kod običnih rezidualnih blokova koriste se dva konvolucijska sloja s filterima veličine 3x3,

dok blok s uskim grlom koristi jednu takvu konvoluciju i dvije konvolucije s filterima veličine 1x1 ispred i iza nje. Prva konvolucija smanjiva broj filtera, dok zadnja povećava na početni broj kako bi se omogućilo zbrajanje s izlazom iz prečice.

4. Implementacija

Programska implementacija konvolucijskog modela za jednooku predikciju dubine scene ostvarena je po uzoru na rad [3] uz manje promjene.

4.1. Korišteni alati i tehnologije

Za izradu rada korišten je programski jezik Python[1], radno okruženje PyTorch[2], te biblioteke Numpy[3], h5py[4], matplotlib[5], sckit-image [6]. Biblioteka Numpy je korištena za efikasno izvršavanje različitih operacija nad matricama, dok je biblioteka sckit-image korištena za učitavanje slika. Biblioteka h5py korištena je za jednostavno spremanje i učitavanje slika. Za vizualizaciju podataka i rezultata korištena je biblioteka matplotlib.

Veći dio programske implementacije napisan je koristeći radno okruženje PyTorch. Verzija PyTorcha u kojoj je pisana programska implementacija je 1.0.1. Jedna od bitnijih karakteristika ovog radnog okruženja je njegov tensor koji je jako sličan Numpyovom polju s dodatkom da se može izvoditi na GPU koji podržava CUDA što omogućava veću brzinu izvođenja. Još jedno bitno svojstvo PyTorcha je to što omogućava jednostavnu implementaciju propagacije pogreške unatrag. Prilikom izvršavanja matematičkih operacija nad tensorima PyTorch sprema koje su se operacije izvodile i pomoću toga izračunava gradijente. Ova metoda zove se automatska diferencijacija i implementirana je u modulu *Autograd*.

Jednostavan primjer automatske diferencijacije u PyTorchu:

```
1 import torch
2
```

[1] https:// www.python.org
[2] https://pytorch.org
[3] https://www.numpy.org
[4] https:// www.h5py.org
[5] https://matplotlib.org
[6] https://scikit-image.org/

```
3 x = torch.ones(1, requires_grad=True) #tensor([1.])
4 y = x + 2 #tensor([3.])
5 z = y * y * 3 #tensor([27.])
6
7 z.backward() #racunanje gradijenata unatrag
8
9 print(x.grad) #dz/dx = 18
```

4.2. NYU Depth podaci

Neuronska mreža je trenirana na *NYU Depth v2* skupu podataka. Skup se sastoji od slijeda videa iz raznih scena iz zatvorenih prostora koje su snimane s RGB i dubinskom kamerom Microsoft Kinect. Ukupno ima 464 scene iz 3 različita grada, 1449 gusto označenih usklađenih parova RGB slika i dubinskih slika, te 407024 neusklađenih slika. Označeni podatci su prethodno obrađeni da popune nedostajuće dubinske oznake, dok su ostali podatci neobrađeni.

Slika 4.1: Primjer prethodno obrađenih podataka

Slika 4.2: Primjer neobrađenih podataka

4.3. Priprema podataka

Treniranje neuronske mreže se provodi nadziranim učenjem nad neobrađenim podatcima. Neobrađeni podatci se sastoje od 464 scene s podjelom na 249 scene za treniranje i 215 za testiranje. Za treniranje se međutim koristi mali podskup slika iz 242 scene za treniranje. Iz svake scene uzorkovano je 50 jednako udaljenih slika i odgovarajućih dubinskih slika, što je rezultiralo od ukupno 12100 parova RGB-D slika. Prilikom treniranja provodi se umjetno proširivanje skupa za učenje čime se dobiva blizu 95000 RGB-D parova.

Ulazne RGB slike imaju dimenzije 640x480, te se moraju svesti na dimenzije ulaza u neuronsku mrežu. Prvo se uzorkuju (engl. *down-sample*) na pola veličine, zatim se odrežu u sredini na dimenzije 304x228.

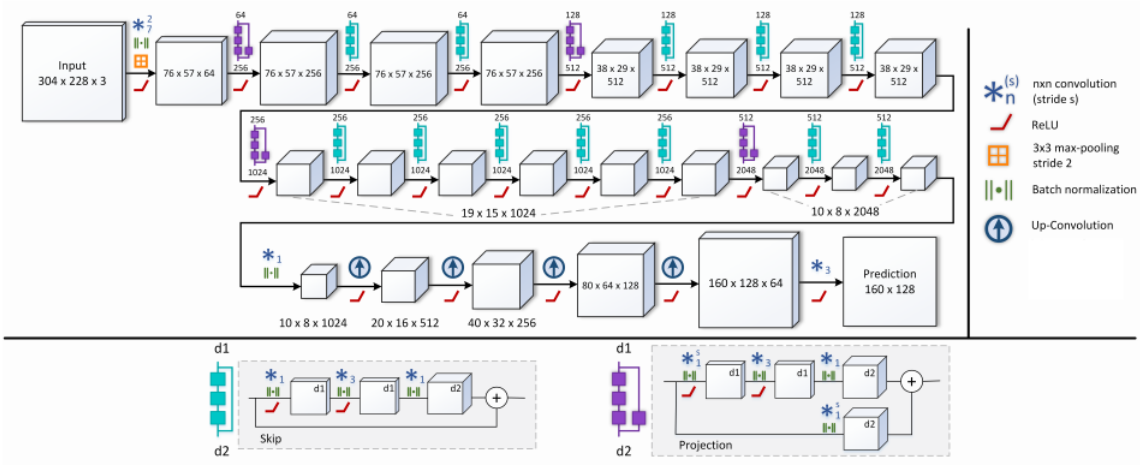
Nad tim ulaznim RGB slikama se primjenjuju transformacija za umjetno proširivanje skupa podataka. Transformacije koje se primjenjuju:

- Skaliranje: ulazne slike i dubinske mape se skaliraju sa $s \in [1, 1.5]$ i dubinske mape se dijele sa s.
- Rotacija: ulazne slike i dubinske mape se rotiriaju za $r \in [-5, 5]$ stupnjeva.
- Translacija: ulazne slike i dubinske mape su odrezane na nasumično odabranim mjestima kako bi im dimenzije odgovarale dimenzijama ulaza u mrežu.
- Boja: ulazne slike se množe s nasumičnom RGB vrijednosti $c \in [0.8, 1.2]^3$.
- Okretanje: ulazne slike i dubinske mape su vodoravno preokrenute s vjerojatnošću od 0.5.

Ovaj postupak se ponavlja 8 puta za svaki RGB-D par.

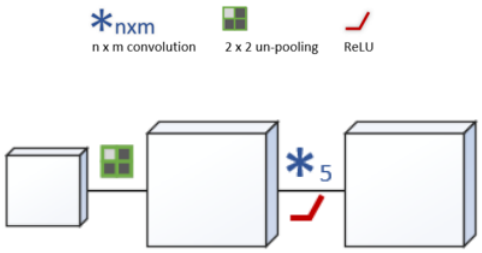
4.4. Arhitektura modela

Arhitektura neuronske mreže se nadograđuje na ResNet-50 model. Zamjenjuje se zadnji potpuno povezani sloj, koji je dio izvorne arhitekture, s novim blokovima za naduzorkovanje, čime se dobiva izlaz koji približno duplo manje dimenzije od ulaza. Zadnji konvolucijski sloj ResNet-50 modela proizvodi izlaz s 2048 mapi značajki s dimenzijama 10x8, zatim se blokovima za naduzorkovanje postiže izlaz dimenzija 160x128 piksela. Arhitektura modela je vidljiva na slici 4.3.



Slika 4.3: Arhitektura modela

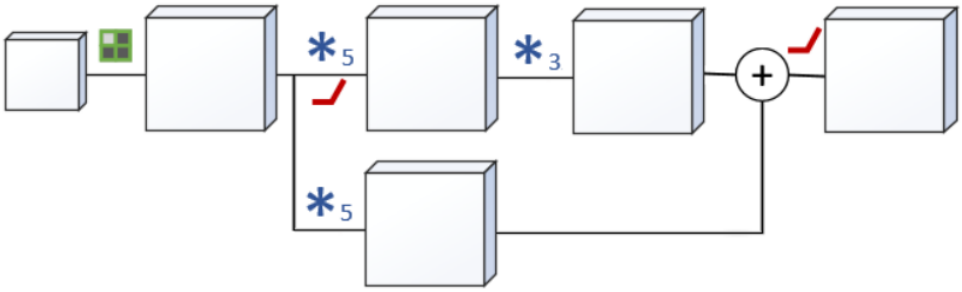
Kako bi se povećala rezolucija izlaza uvode se novi blokovi za naduzorkovanja zvano up-convolution blokovi. Cilj ovih blokova je povećati za duplo dimenzije ulaza, tako da preslikavaju svaki ulaz u gornji lijevi kut filtera veličine 2x2, a ostatak filtera se puni nulama. Nakon svakog takvog sloja slijedi konvolucijski sloj s filterom veličine 5x5 i zatim ReLU sloj. Up-convolution blok vidljiv je na slici 4.4.



Slika 4.4: Up-convolution blok

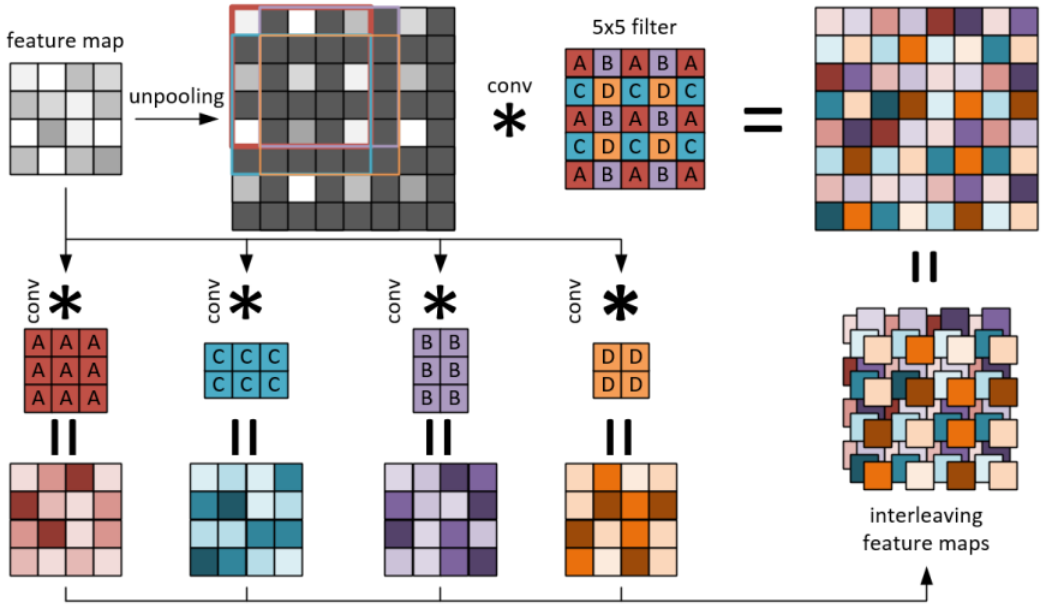
Ovaj blok se dalje proširuje kako bi se stvorio up-projection blok. Nakon up-convolution bloka dodaje se 3x3 konvolucija i rezultatu se dodaje projekcijska veza iz

mape značajki s manjom rezolucijom kako je prikazano na slici 4.5. Zbog razlike u dimenzijama, manje mape značajki trebaju se naduzorkovati s još jednim up-convolution blokom. Prva operacija up-convolution bloka se primjenjuje za obje grane zajedno, a zatim se 5x5 konvolucija primjenjuje odvojeno.



Slika 4.5: Up-projection blok

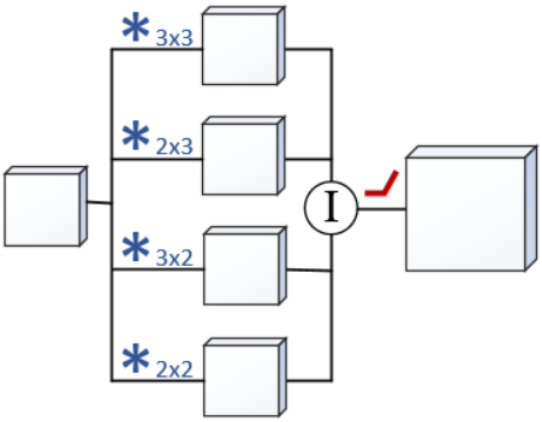
Kako bi se povećala učinkovitost up-convolution bloka uvodi se novi brzi up-convolution blok. Nakon primjene prve operacije u up-convolution bloku 75% brojeva u izlaznim mapama značajki su nule, prema tome 5x5 konvolucija većinom djeluje na nulama, što se može izbjeći. Ovo se može vidjeti na slici 4.6.



Slika 4.6: Način rada brzog up-convolution bloka

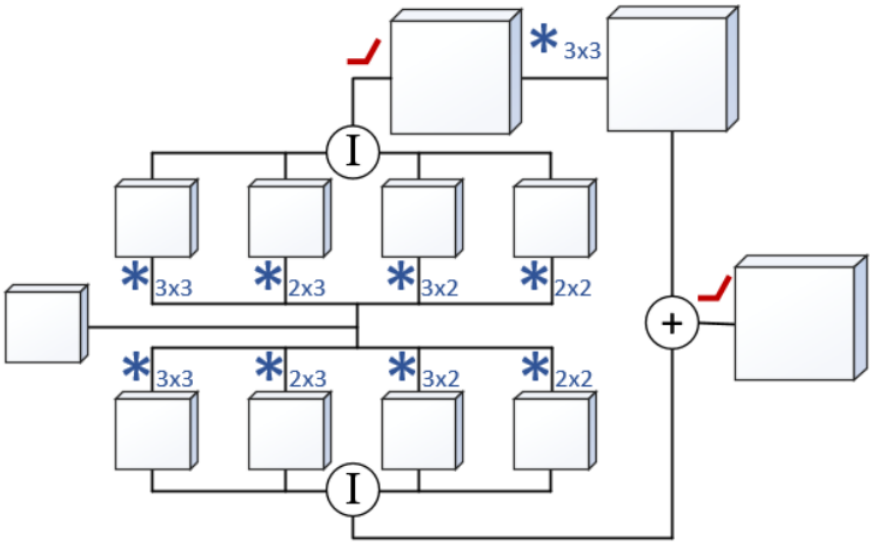
Nakon povećavanja dimenzije mapi značajki, te primjene 5x5 konvolucije, samo se određene težine množe s vrijednostima koje nisu nula. Te težine se dijele na četiri grupe koje se ne preklapaju, prikazane oznakama A,B,C,D i različitim bojama. Na

temelju te četiri grupe izvorna 5x5 konvolucija se zamjenjuje s četiri nova filtera dimenzija 3x3 (A), 3x2 (B), 2x3 (C), 2x2 (D). Isti izlaz se sad može ostvariti kao i kod izvornih operacija umetanjem elemenata iz četiri nove mape značajki kao što je prikazano na slici 4.6. Brzi up-convolution blok koji koristi ovu metodu prikazan je na slici 4.7.



Slika 4.7: Brzi up-convolution blok

Kako se kod običnog up-convolution bloka uvodi novi up-projection blok, tako se i za brzi up-convolution blok uvodi novi brzi up-projection blok. Njegova arhitektura prikazana je na slici 4.8.



Slika 4.8: Brzi up-projection blok

5. Rezultati

6. Zaključak

Zaključak.

LITERATURA

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

[2] 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.

[3] Iro Laina, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari, i Nassir Navab. Deeper depth prediction with fully convolutional residual networks. *CoRR*, abs/1606.00373, 2016. URL http://arxiv.org/abs/1606.00373.

[4] Marko Čupić. Umjetna inteligencija, umjetne neuronske mreže. http://java.zemris.fer.hr/nastava/ui/ann/ann-20180604.pdf.

Konvolucijski modeli za jednooku predikciju dubine scene

Sažetak

Sažetak na hrvatskom jeziku.

Ključne riječi: Ključne riječi, odvojene zarezima.

Title

Abstract

Abstract.

Keywords: Keywords.