Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

# Detekcija izbranih površinskih anomalij na odbojnih površinah z deflektometrijo

Lojze Žust

Delo je pripravljeno v skladu s Pravilnikom o podeljevanju Prešernovih nagrad študentom, pod mentorstvom izr. prof. dr. Mateja Kristana

Ljubljana, 2018

# Povzetek

**Naslov:** Detekcija izbranih površinskih anomalij na odbojnih površinah z deflektometrijo

**Avtor:** Lojze Žust

Predlagamo novo metodo za detekcijo anomalij na reflektivnih površinah z uporabo deflektometrije. Klasične metode deflektometrije zahtevajo 3D rekonstrukcijo opazovanega objekta, napake pa zaznajo z odstopanji od referenčnega modela objekta brez defektov. Te metode za delovanje potrebujejo natančno kalibracijo sistema in mirujoč objekt. Predlagamo novo metodo, ki je sposobna hitre detekcije na podlagi zgolj ene slike opazovane površine brez vmesne 3D rekonstrukcije. Metoda se uči na anotiranih primerih anomalij, zato pri inferenci ne potrebuje posebne kalibracije in referenčnih objektov. Razvita metoda predstavlja povsem novo paradigmo detekcije anomalij. Sestavljena je iz dveh delov. Prvi del metode temelji na semantični segmentaciji , ki za vsak piksel vrne verjetnost prisotnosti anomalije. Semantična segmentacija je implementirana s konvolucijskimi nevronskimi mrežami. Drugi del metode predstavlja robusten postopek za lokalizacijo detekcij iz segmentacijske maske, ki je sposoben zaznati tudi močno prekrivajoče se detekcije. S preliminarno analizo in eksperimentalno evalvacijo utemeljimo izbiro arhitekture ter hiperparametrov modela. Razvito metodo učimo in evalviramo na problemu detekcije udrtin v strehi avtomobila, ter pokažemo bistvene izboljšave v primerjavi s trenutno najboljšo komercialno metodo. Naša metoda je 17x hitrejša in skoraj 50 % uspešnejša od komercialne metode s priklicem, natančnostjo in F-mero kar 88%.

**Ključne besede:** konvolucija, nevronske mreže, deflektometrija, semantična segmentacija, strojno učenje.

# Abstract

**Title:** Deflectometry-based detection of specific reflective surface anomalies

**Author:** Lojze Žust

In this work we propose a new deflectometry-based anomaly detection approach applicable to reflective surfaces. Classic deflectometry methods detect surface anomalies by performing partial 3D surface reconstruction and differencing it with a pre-recorded reference model of the observed object. Most of these methods require accurate calibration between the pattern projector, camera and the inspected object. In contrast, our anomaly detection approach is capable of fast detection without the need for a 3D reconstruction. Since the proposed method can be trained on annotated anomaly examples, reference objects and accurate calibration are not required. Furthermore, the developed method brings forward an entirely new approach to anomaly detection and is made up of two parts. The first part is based on semantic segmentation and performs pixel-wise anomaly classification. We utilize the power of deep models for this purpose. The second part is a robust mechanism for anomaly localization from the segmentation mask, capable of extracting even largely overlapping detections. Preliminary analysis and experimental evaluation were performed to justify the architecture and hyperparameters of our deep semantic segmentation model. The final model was trained and evaluated on the problem of dent detection in car roofs, where a significant improvement over the state of the art commercial method has been shown. Our model is 17x faster and almost 50 % more accurate than the commercial method and achieves a precision, recall and F-score of 88 %.

**Keywords:** convolution, neural networks, deflectometry, semantic segmentation, machine learning.

# Kazalo

# Poglavje 1

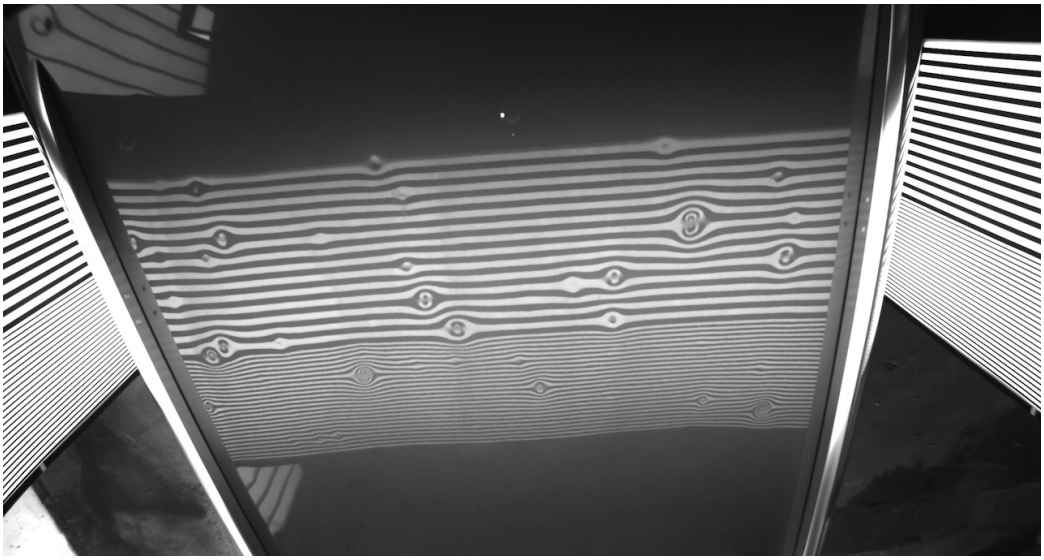
# Uvod

## 1.1 Motivacija

Deflekotmetrija [21] se ukvarja z opazovanjem in analiziranjem odboja svetlobnega vzorca na prosojnih in odbojnih površinah. Eno izmed področij deflektometrije je tudi modeliranje in analiza lastnosti odbojnih površin [11]. Obsežno se uporablja v industriji, saj omogoča nedestruktivno analizo kvalitete delov. Primer uporabe je detekcija anomalij na pločevini, kar se izkaže za zelo uporabno v avtomobilski industriji [15]. Avtomobilska pločevina je med svojo življensko dobo izpostavljena številnim zunanjim dejavnikom poškodb.

Tipičen primer poškodb so razne praske in vdolbine, posledica manjših nesreč in trkov. Še posebej neprijetne pa so poškodbe, ki jih na strehi nesrečnega lastnika pusti toča. Močan naliv lahko na pločevini pusti na stotine udrtin. V praksi njihovo označevanje in štetje poteka ročno. To je naporen in dolgotrajen postopek, ki zahteva veliko pozornosti, saj so udrtine s prostim očesom precej slabo vidne. Sistem, ki nadomešča ročno označevanje z metodami deflektometrije bi bil zato velik prispevek.

Tak sistem bi minimiziral pristranskost ocenjevalca, omogočil avtomatsko izpolnjevanje poročil in nenazadnje bistveno povečal hitrost obdelave posameznega avtomobila. Metode deflektometrije na odbojnih površinah uporabljajo posebne sisteme za zajem slike, ki na površino projicirajo svetlobni



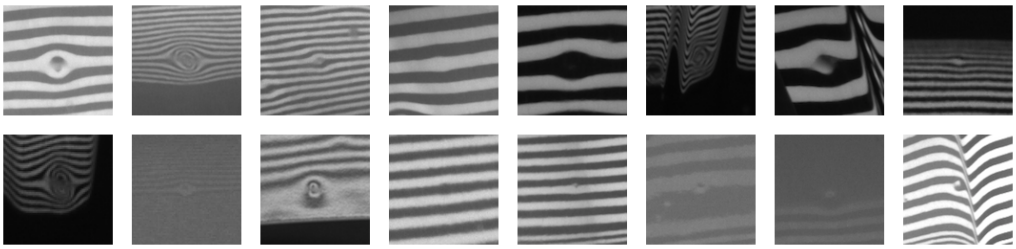
Slika 1.1: Primer slike zajete v sistemu.

vzorec. Človeško oko na slikah, zajetih s takim sistemom (glej Sliko 1.1), zlahka zazna anomalije. Pri implementaciji metode deflektometrije pa se pojavi vrsta težav.

Odbojne površine so si med seboj zelo različne. Njihova oblika se od objekta do objekta zelo razlikuje. Avtomobili so pogosto umazani, po strehah pa imajo razne nosilce in žlebove. Ker v praksi, v nasprotju z industrijskimi procesi, nimamo referenčne baze objetkov, te lastnosti metode deflektometrije težko ločijo od udrtin. Slike se zaradi različnih barv in značilnosti odbojnih površin zelo razlikujejo med seboj. Svetloba, ki pada na udrtine, pa pri odboju tvori najrazličnejše vzorce (glej Sliko 1.2).

## 1.2 Sorodna dela

Dosedanje metode detekcije anomalij na odbojnih površinah v veliki večini temeljijo na tehnikah deflektometrije, ki je dobro raziskano področje [11, 21, 17, 7, 18]. Osnovna ideja metod deflektometrije je opazovanje spremembe vnaprej določenega svetlobnega vzorca, ki ga projiciramo na analizirano površino. Običajno gre za tako imenovan resast vzorec, ki je sestavljen iz



Slika 1.2: Odboj svetlobe na udrtinah povzroča zelo različne vzorce.

ravnih svetlobnih črt. Na podlagi deformacij, do katerih pride pri odboju vzorca od ukrivljene površine, lahko sklepamo o 3D lastnostih opazovane površine. S frekvenčno analizo svetlosti vzorca, ki jo napravimo vzdolž preseka pravokotno na svetlobne črte, ocenimo nagib točke.

V [15] iz slik pridobljenih iz dveh kamer, ki snemata projiciran vzorec, rekonstruirajo 3D površino analiziranega kosa. Tega nato primerjajo s kosom v bazi in tako najdejo anomalije v površini. V [4] je opisan algoritem, ki z deflektometrijo in s svetlobnim vzorcem detektira napake v industrijskih kosih. S frekvenčno analizo vzorca in triangulacijo določijo višino točk na opazovanem kosu. Napake detektirajo tako, da rekonstruiran kos primerjajo s kosom v bazi. V obeh primerih so za 3D rekonstrukcijo potrebni posebni pogoji. V prvem potrebujemo stereo sistem kamer, ki na znani razdalji opazujeta objekt. V drugem pa potrebujemo sistem za kontroliran premik industrijskega kosa, saj višino ocenjujemo na podlagi spremembe vzorca pri znanem premiku. Poleg tega je za detekcijo anomalij potrebna referenčna baza opazovanih objektov. Narava problema, ki ga naslavljamo, ne daje možnosti gradnje omenjene referenčne baze. Število vseh opazovanih objektov je namreč preveliko, da bi bila takšna rešitev praktična.

V [10] se problema lotijo z drugačnega vidika. Detekcijo udrtin implementirajo kot razpoznavo objektov z uporabo Haarovih značilk. Učenje in testiranje izvajajo na umetno generiranih podatkih. Tak sistem ne zahteva referenčne baze kosov, saj metoda išče napake glede na njihov videz, ta pa je v primeru udrtin na različnih površinah dokaj konsistenten. Empirično

določijo tri tipe odbojev, do katerih lahko pride na anomalijah in za vsakega učijo ločen klasifikator. Pri našem problemu je število tipov odbojev težko določiti, saj se spreminjajo glede na lastnosti pločevine in velikost udrtine. Raznovrstnost tipov odbojev je razvidna na Sliki 1.2. Za delovanje njihove metode bi bilo potrebno empirično poiskati vse različne tipe udrtin, ki se pojavljajo in za vsakega izdelati učno množico. Poleg tega pri velikem številu klasifikatorjev lahko postane vprašljiva časovna učinkovitost.

Na detekcijo anomalij lahko gledamo tudi kot na označevanje pikslov, ki predstavljajo anomalijo. V tem kontekstu lahko detekcijo anomalij obravnavamo kot problem semantične segmentacije, ki na nivoju pikslov določi prisotnost udrtine. Področje semantične segmentacije v računalniškem vidu je zelo obširno. Širok pregled razvoja in obstoječih metod semantične segmentacije na različnih tipih podatkov (slike, 3D slike, volumetrični podatki) je na voljo v [5].

## 1.3 Prispevki

V tem delu je predstavljena nova metoda za detekcijo anomalij v deflektometriji. Metoda, za razliko od večine obstoječih rešitev, ne temelji na delni 3D rekonstrukciji opazovanega objekta in računanjem odstopanj, temveč za osnovo vzame podobnost med anomalijami. Ker se metoda uči karakteristik samih anomalij, lahko dobro posplošuje in lahko detektira sorodne anomalije tudi na povsem neznanih objektih, ki niso bili prisotni v učni množici. S tem pristopom je metoda sposobna zaznati anomalije tudi na površinsko kompleksnih objektih, kjer klasične metode zaradi kompleksnosti vzorca odpovejo. Vse to metoda dosega v zelo preprostem sistemu. Metoda se uči na posameznih slikah udrtin, detekcija pa se prav tako izvaja na posameznih slikah objektov, brez dodatnih kontrolnih sistemov in meritev. Izbran pristop prinaša tudi enostavno prenosljivost metode na sorodne domene.

## 1.4 Struktura naloge

Preostanek naloge sestavljajo štiri poglavja. V Poglavju 2 opišemo zgradbo in delovanje konvolucijskih nevronskih mrež, kjer prikažemo operacijo konvolucije, vrste slojev, ki se uporabljajo v mrežah, ter njihove sestavne dele. V Poglavju 3 predstavimo našo metodo za lokalizacijo udrtin. Opišemo arhitekturo mreže, metode predprocesiranja ter algoritem za lokalizacijo detekcij. Poglavje 4 vsebuje eksperimentalno evalvacijo. V poglavju predstavimo podrobnosti implementacije, nato pa analiziramo vpliv posameznih hiperparametrov na natančnost metode. Na zadnje še evalviramo končen model in ga primerjamo z osnovno metodo. V Poglavju 5 podamo kratek povzetek rezultatov in prispevkov naloge, ter predloge za izboljšavo sistema.

# Poglavje 2

# Konvolucijske nevronske mreže

Konvolucijske nevronske mreže so vrsta globokih nevronskih mrež, ki temeljijo na operaciji konvolucije. Uporabljajo se za izločanje informacij iz kompleksnih tipov podatkov (npr. slike [13, 28], zvok [8]), za kar v primerjavi s klasičnimi metodami potrebujejo zelo malo predprocesiranja. Sestavljene so iz zaporedja skupkov operacij, ki jih imenujemo sloji. To poglavje vsebuje pregled osnovnih gradnikov konvolucijskih nevronskih mrež. V Poglavju 2.1 opišemo operacijo konvolucije, v Poglavju 2.2 vrste in strukture slojev, ki pogosto sestavljajo te mreže, v Poglavju 2.3 pa predstavimo postopek učenja mreže.

## 2.1 Konvolucija

Konvolucija je matematična operacija med dvema funkcijama, ki proizvede novo funkcijo. Nova funkcija je spremenjena verzija prve funkcije, definirana kot integral

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau, \tag{2.1}$$

kjer je $(f * g)(t)$ funkcija, ki je rezultat konvolucije med funkcijama $f(t)$ in $g(t)$. Kadar imamo opravka z diskretnimi vrednostmi se integral spremeni v vsoto

$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m]g[n-m], \tag{2.2}$$

kjer sta $f[\cdot]$ in $g[\cdot]$ diskretni funkciji. Operacijo konvolucije lahko posplošimo na več dimenzij. Rezultat dvodimenzionalne diskretne konvolucije med dvodimenzionalnima funkcijama $f[\cdot,\cdot]$ in $g[\cdot,\cdot]$ dobimo po enačbi

$$(f * g)[n_1, n_2] = \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} f[k_1, k_2]g[n_1 - k_1, n_2 - k_2]. \tag{2.3}$$

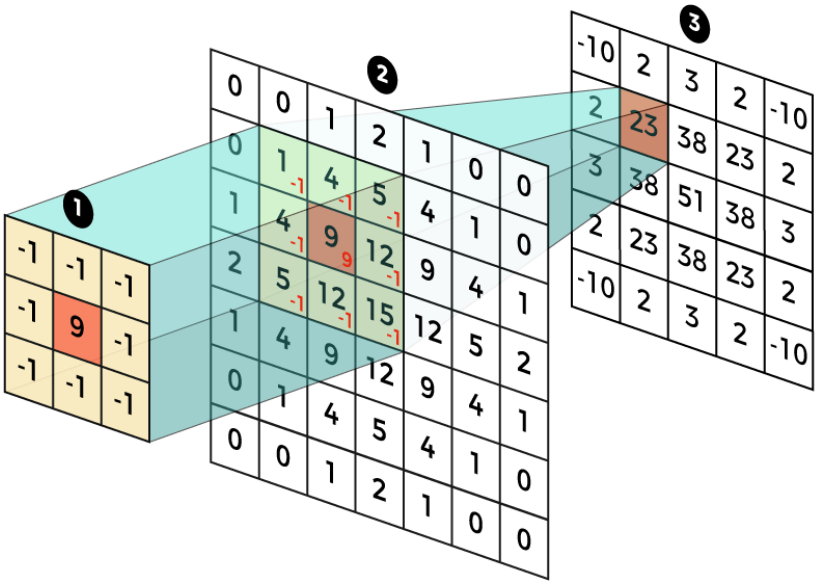
Operacijo dvodimenzionalne diskretne konvolucije je najlažje ponazoriti na praktičnem primeru. V našem primeru je osnovna oblika podatkov sivinska slika, ki jo lahko predstavimo kot matriko $\mathbf{X}$, katere vrednosti predstavljajo svetlost posameznega piksla. Za konvolucijo potrebujemo še konvolucijski filter oz. jedro. To je matrika, ki jo bomo označili z $\mathbf{W}$. Rezultat konvolucije je matrika, ki jo označimo kot $\mathbf{Y} = \mathbf{X} * \mathbf{W}$, posamezen element pa lahko izračunamo po (2.3). Upoštevati moramo, da naše matrike niso neskončne velikosti. Število elementov v vrstici matrike (širino) bomo označevali z $w_{\mathbf{M}}$, višino pa z $h_{\mathbf{M}}$, kjer je $\mathbf{M}$ poljubna matrika. Element matrike $\mathbf{Y}$ tako dobimo po enačbi

$$\mathbf{Y}[n_1, n_2] = \sum_{k_1=-\epsilon_y}^{\epsilon_y} \sum_{k_2=\epsilon_x}^{\epsilon_x} \mathbf{X}[k_1, k_2]\mathbf{W}[n_1 - k_1, n_2 - k_2], \tag{2.4}$$

kjer sta $\epsilon_x$ in $\epsilon_y$ x in y-okolici elementa, ki povesta, kako veliko okolico elementa vhoda v x in y smeri upoštevamo pri izračunu elementa izhoda. Dobimo ju iz velikosti filtra po enačbah

$$\begin{aligned} \epsilon_x &= \frac{w_{\mathbf{W}} - 1}{2}, \\ \epsilon_y &= \frac{h_{\mathbf{W}} - 1}{2}. \end{aligned} \tag{2.5}$$

Konvolucijo grafično ponazorimo s Sliko 2.1. Konvolucijski filter transponiramo in ga postavimo tako, da se središčni element filtra in element na vhodu pokrivata. Element na izhodu $\mathbf{Y}$ dobimo kot vsoto zmnožka po parih



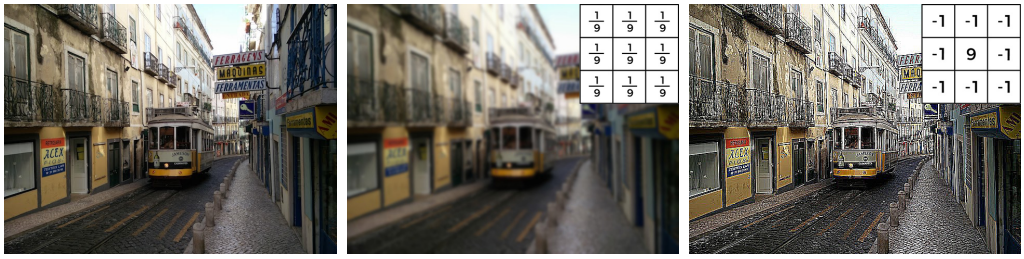

Slika 2.1: Prikaz delovanja konvolucije na primeru (1 - konvolucijski filter, 2 - vhodna matrika, 3 - izhodna matrika).

vseh elementov, ki jih jedro prekrije. Filter premikamo po vhodni matriki in postopek ponavljamo za vsak element. Če filtra pred operacijo ne transponiramo, potem namesto konvolucije govorimo o križni korealaciji. Velikost filtra določa velikost območja, ki jo konvolucija upošteva. Pri velikosti $3 \times 3$ se upoštevajo vsi sosednji elementi vhodnega elementa. Velikost $5 \times 5$ upošteva še sosede sosedov in tako dalje.

Konvolucija je zelo uporabna v računalniški grafiki in računalniškem vidu. Z njo lahko implementiramo mnoge uporabne preslikave kot na primer megljenje in ostrenje slik (glej Sliko 2.2). V računalniškem vidu se uporabljajo tudi pri detekciji robov in iskanju značilnih točk.

### 2.1.1 Konvolucija v konvolucijskih nevronskih mrežah

Konvolucija je glavna komponenta konvolucijskih nevronskih mrež. Mreže so sestavljene iz več konvolucijskih slojev, ki vsebujejo množico konvolucijskih




(a) Originalna slika. (b) Zamegljena slika. (c) Izostrena slika.

Slika 2.2: Primer uporabe konvolucije: ostrenje in megljenje slik. Prikazani so tudi konvolucijski filtri, ki so bili uporabljeni za efekt.

filtrov. V fazi učenja se vrednosti elementov konvolucijskih filtrov postopoma optimizirajo tako, da se izhod konvolucijske nevronske mreže čim bolj sklada z željenimi vrednostmi podatkov učne množice. Poleg konvolucijskih slojev se v konvolucijskih nevronskih mrežah običajno uporabljajo še sloji združevanja, ki zmanjšajo velikost slike. Razloge za to bomo navedli v nadaljevanju. Slika 2.3 prikazuje primer arhitekture konvolucijske nevronske mreže.

## 2.2 Sloji

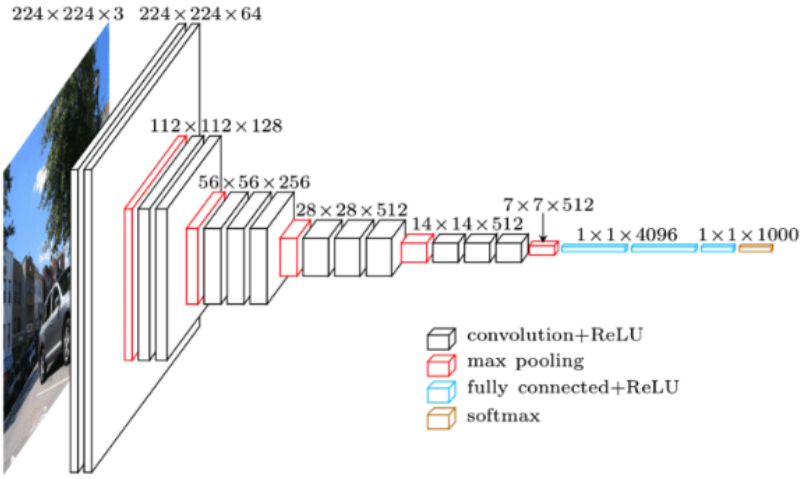
V tem delu bomo predstavili različne tipe slojev, ki sestavljajo konvolucijske mreže. Opisali bomo njihove sestavne dele, ter predstavili pomen njihove uporabe.

### 2.2.1 Konvolucijski sloj

Osnovni gradnik konvolucijske nevronske mreže je konvolucijski sloj. Predstavimo ga lahko z enačbo

$$\mathbf{Y} = f(\mathbf{X} * \mathbf{W} + \mathbf{b}), \tag{2.6}$$

kjer je $\mathbf{X}$ tenzor vhoda sloja, $\mathbf{Y}$ tenzor izhoda, $\mathbf{W}$ tenzor uteži oz. konvolucijskih filtrov, $\mathbf{b}$ vektor odmikov, $f(\cdot)$ pa aktivacijska funkcija sloja. V




Slika 2.3: Primer konvolucijske nevronske mreže. Na sliki je prikazana arhitektura mreže VGG16 [25]. Slika je povzeta po [14].

nadaljevanju opišemo konvolucijske filtre.

### Filtri

V Poglavju 2.1 smo predstavili konvolucijo na dvodimenzionalnih matrikah (2.4). Slike pa v splošnem vsebujejo še tretjo dimenzijo – kanale. Na vhodu imajo slike običajno enega (sivinske slike) ali tri (barvne slike) kanale. Število kanalov slike pa se v večini primerov skozi mrežo spreminja. Tudi konvolucijski filtri morajo zaradi tega biti tridimenzionalni. Število kanalov filtra mora biti enako številu kanalov vhoda. Konvolucija na večih kanalih deluje enako kot konvolucija na dvodimenzionalnih matrikah, le da seštejemo rezultate dvodimenzionalnih konvolucij po posameznih kanalih. Enačba konvolucije tako postane

$$Y[n_1, n_2] = \sum_{c=0}^{c_X} \sum_{k_1=-\epsilon_y}^{\epsilon_y} \sum_{k_2=\epsilon_x}^{\epsilon_x} X[k_1, k_2, c] W[n_1 - k_1, n_2 - k_2, c], \tag{2.7}$$

kjer uvedena notacija $c_{\mathbf{X}}$ označuje število kanalov tenzorja $\mathbf{X}$. Po taki operaciji konvolucije na izhodu dobimo tenzor z enim samim kanalom. Konvolucijski sloji običajno vsebujejo več filtrov. Rezultate konvolucij s posameznimi

filtri zložimo v izhod z več kanali. Število filtrov torej določa število kanalov izhoda $c_{\mathbf{Y}}$.

### Obdajanje z ničlami

Konvolucija na robovih ni definirana, saj za izračun potrebuje sosednje elemente, ki na robovih manjkajo (glej Sliko 2.1). Izhod mreže je zato nekoliko manjši od vhoda. Njegovo velikost dobimo po enačbah

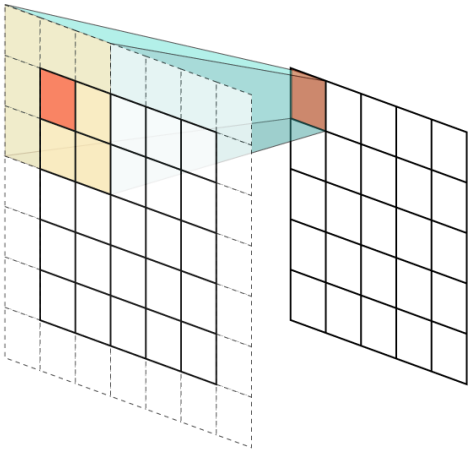
$$\begin{aligned} w_{\mathbf{Y}} &= w_{\mathbf{X}} - 2\epsilon_x, \\ h_{\mathbf{Y}} &= h_{\mathbf{X}} - 2\epsilon_y. \end{aligned} \tag{2.8}$$

Velikost filtrov torej vpliva na količino zmanjšanja velikosti izhoda. Pri načrtovanju arhitekture nam to lahko predstavlja problem. Če spremenimo velikost filtrov v nekem sloju, se spremeni tudi velikost izhoda tega sloja. Posledično se spremenijo velikosti vhodov in izhodov vseh nadaljnjih slojev, zato je potrebno mrežo pravilno popraviti, da ostane smiselna. Vsako spreminjanje velikosti filtra torej pomeni, da moramo ponovno preračunati in popraviti velikosti vseh nadaljnjih slojev.

Temu se lahko zelo enostavno izognemo, tako da vhod konvolucijskega sloja ob robovih ustrezno obdamo z ničlami. Temu postopku rečemo obdajanje z ničlami (angl. padding). Od tu dalje bomo za ta postopek uporabljali besedo *obdajanje*. Vhod razširimo na tako velikost, da bo po operaciji konvolucije izhod reduciran na prvotno velikost vhoda. Tako poskrbimo, da sta vhod in izhod sloja vedno enako velika, ne glede na velikost filtrov. Tako obdajanje imenujemo obdajanje za ohranitev velikosti (Slika 2.4). Od tu naprej bomo z vsakim omenjanjem obdajanja v mislih imeli to vrsto.

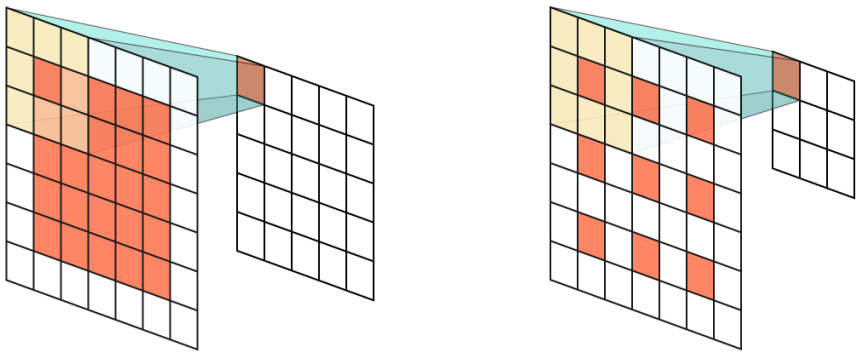
### Korak

Velikost izhoda lahko nadziramo tudi s parametrom korak (angl. stride). Ta narekuje, kako gosto pri konvoluciji jemljemo elemente iz vhoda. Njegova vrednost določa, za koliko elementov vhoda se premaknemo preden izračunamo naslednji element izhoda (glej Sliko 2.5). Korak lahko določimo individualno



Slika 2.4: Obdajanje za ohranitev velikosti. S črtano obrobo so označeni elementi, ki jih obdajanje doda.

za obe dimenziji, najpogosteje pa se uporablja enaka vrednost za obe dimenziji. Pri običajni konvoluciji po (2.7) je vrednost parametra korak enaka 1. Korak 2 pomeni, da konvolucija kot centralni element jemlje vsak drugi element, korak 3 vsak tretji, in tako naprej.



(a) Navadna konvolucija, korak = 1. (b) Konvolucija, korak = 2.

Slika 2.5: Izpuščanje elementov s parametrom korak. Oranžni elementi predstavljajo centre konvolucije.

Kombinacija dodajanja ničel in uporabe koraka daje izhode predvidljive velikosti. Če uporabimo korak 2, se širina in višina slike na izhodu razpolovita. V splošnem je velikost izhoda enaka

$$\begin{aligned} w_Y &= \frac{w_X}{s_x}, \\ h_Y &= \frac{h_X}{s_y}, \end{aligned} \tag{2.9}$$

kjer $s_x$ in $s_y$ označujeta parameter korak po $x$ in $y$ osi.

### Odmik in aktivacijska funkcija

Izhodu konvolucije dodamo odmik (angl. bias). To je vektor $\mathbf{b}$ učljivih vrednosti dolžine $c_{\mathbf{Y}}$ (ena vrednost na filter). Posamezno vrednost prištejemo vsem elementom tega kanala, t.j.,

$$\mathbf{Y}[i, j, k] = \mathbf{Y}[i, j, k] + \mathbf{b}[k]. \tag{2.10}$$

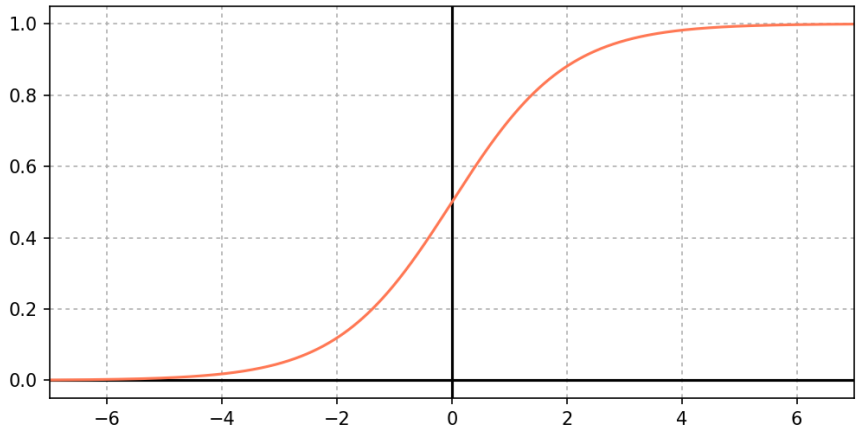
Naloga odmika je, da elemente izhoda konvolucije ustrezno zamakne, preden jih spustimo skozi aktivacijsko funkcijo. Aktivacijska funkcija je enaka za vse elemente in se proži pri istih vrednostih. Z odmikom zamaknemo vrednosti posameznih kanalov in s tem omogočimo, da se pri različnih kanalih aktivacijska funkcija proži pri različnih vrednostih ter tako povečamo fleksibilnost učenja.

Zamaknjene vrednosti nato spustimo skozi aktivacijsko funkcijo. Brez nelinearne aktivacijske funkcije je mreža le ena velika linearna funkcija, aktivacije pa omogočijo učenje nelinearnosti. Aktivacijska funkcija je v teoriji lahko katerakoli funkcija, v praksi pa se najpogosteje uporabljajo sigmoidna funkcija, funkcija tanh in funkcija ReLU, ki jih bomo opisali v nadaljevanju.

**Sigmoidna funkcija** S pojmom sigmoidna funkcija običajno označujemo funkcijo

$$\sigma(x) = \frac{1}{1 + e^{-x}}, \tag{2.11}$$

ki vhodne vrednosti $x$ stisne v izhodne vrednosti med 0 in 1 (glej Sliko 2.6). Ločnica med obema vrednostima je precej strma, zato je funkcija idealna za binarno klasifikacijo, kjer želimo napovedati ali primer pripada razredu ali ne. Pri $x = 0$ je funkcija najbolj strma, gradient pa največji, kar pomeni, da se tu najmočneje uči. Ob robovih je gradient zelo majhen, zato je učenje veliko šibkejše. Pri večjem številu slojev se hitro lahko pojavi problem izginjajočih gradientov (angl. vanishing gradient). Kot smo ugotovili so pri robovih gradienti majhni. Ko se gradienti pri vzvratnem propagiranju napake prenašajo navzgor po mreži, se pri vsaki sigmoidni funkciji še zmanjšajo, zaradi omejene natančnosti float števil pa lahko celo povsem izginejo. Zgornji sloji se tako učijo občutneje počasneje od spodnjih, ali pa se sploh ne učijo. Problem izginjajočih gradientov rešuje funkcija ReLU [19], ki jo bomo opisali v nadaljevanju.




Slika 2.6: Sigmoidna funkcija.

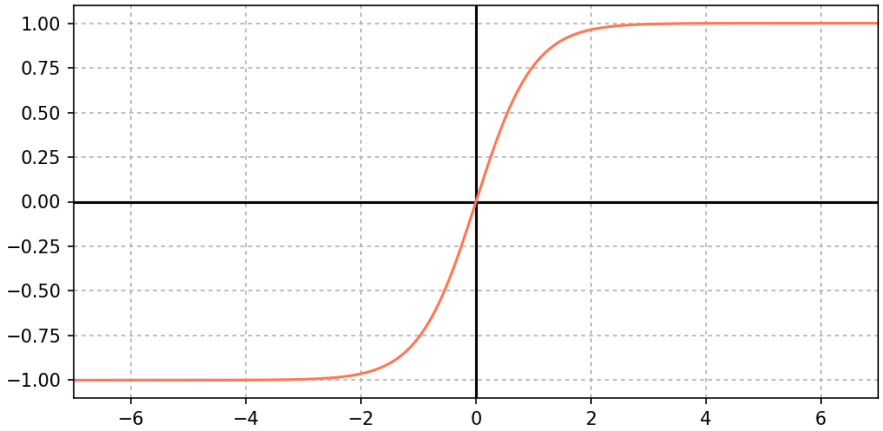
**Funkcija tanh** Funkcija $tanh(\cdot)$ (Slika 2.7) je definirana kot

$$\tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1} = \frac{2}{1 + e^{-2x}} - 1. \tag{2.12}$$

Funkcija tanh je pravzaprav skalirana sigmoidna funkcija, ki jo dobimo z

enačbo

$$\tanh(x) = 2\sigma(2x) - 1. \tag{2.13}$$




Slika 2.7: Funkcija $tanh(x)$.

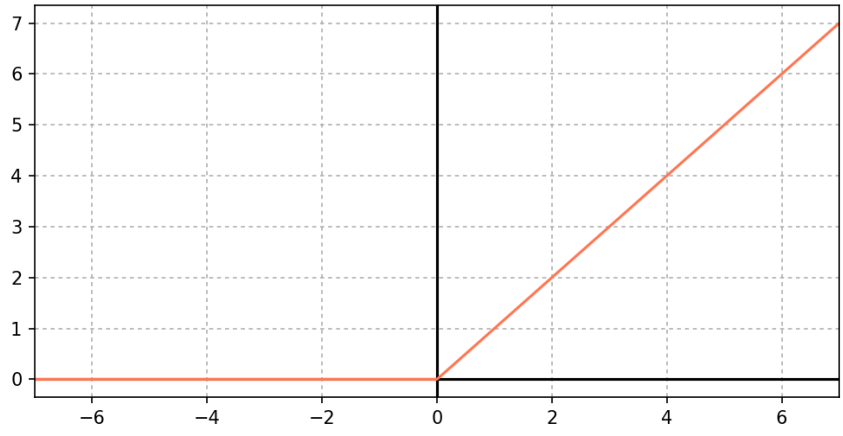
Posledično ima podobne lastnosti kot sigmoidna funkcija, le da je njeno definicijsko območje interval $(-1, 1)$. V $x = 0$ ima še večji gradient od sigmoidne funkcije, zato še ostreje ločuje vrednosti med -1 in 1.

**Funkcija ReLU** Funkcija ReLU (angl. rectified linear unit) označuje funkcijo, ki je definirana kot pozitivni del vhoda (glej Sliko 2.8). Zapišemo jo lahko kot

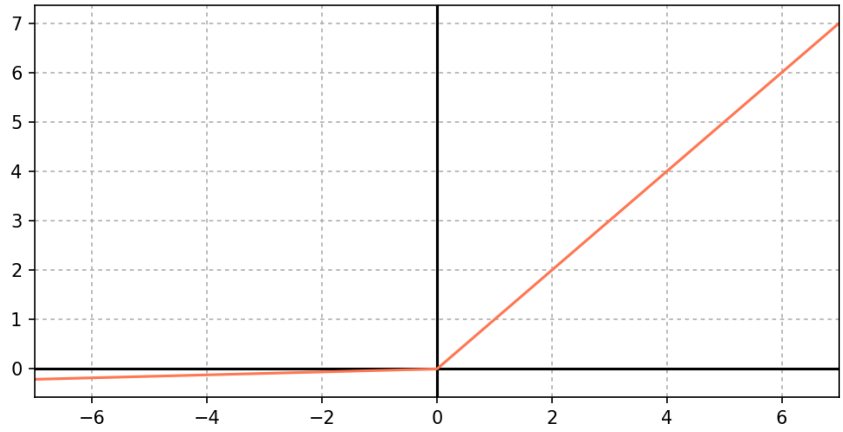
$$\text{ReLU}(x) = \max(0, x). \tag{2.14}$$

ReLU ima kar nekaj prednosti pred sigmoidnimi funkcijami. Pri naključno inicializirani mreži in naključnem vhodu bo kar 50% aktivacij neaktivnih (enakih nič), kar pomeni večjo hitrost mreže, saj pri neaktiviranih nevronih ni potrebno izvajati operacij množenja. Funkcija je precej bolj računsko enostavna kot sigmoidne funkcije, zato je bolj učinkovita. Prav tako ne

prihaja do izginjajočih gradientov ali gradientne eksplozije, saj je odvod pri $x > 0$ konstanten.




Slika 2.8: Funkcija ReLU.




Slika 2.9: Puščajoči ReLU.

Lahko pa se zgodi, da uteži nekega nevrona potisnemo v tak položaj, da bo nevron pri vseh vhodih neaktiviran, kar pomeni, da bo odvod vedno enak 0. Tak (t.i. mrtev) nevron je povsem neuporaben. Ta problem se

običajno pojavi, če je hitrost učenja nastavljena previsoko. Problemu se lahko izognemo s prilagoditvijo, ki jo imenujemo puščajoči (angl. leaky) ReLU (Slika 2.9). Namesto konstantne vrednosti 0, pri $x < 0$, uporabimo zelo položno linearno funkcijo

$$f(x) = max(\alpha x, x); \alpha << 1, \tag{2.15}$$

kjer je $\alpha$ parameter, ki določa nagib premice.

#### Število učljivih parametrov

V tem delu definiramo število učljivih parametrov, ki jih vsebuje konvolucijski sloj. Vhod konvolucijskega sloja je velikosti $w_{\mathbf{X}} \times h_{\mathbf{X}} \times c_{\mathbf{X}}$, izhod pa velikosti $w_{\mathbf{Y}} \times h_{\mathbf{Y}} \times c_{\mathbf{Y}}$. Konvolucijski sloj vsebuje $c_{\mathbf{Y}}$ filtrov, vsak izmed njih pa je tenzor velikosti $k \times k \times c_{\mathbf{X}}$, pri čemer je $k$ velikost filtra. Poleg filtrov so učljivi parametri še elementi odmika. Teh je enako številu kanalov izhoda $c_{\mathbf{Y}}$. Vseh parametrov posameznega konvolucijskega sloja je torej

$$N_{\text{par}} = c_{\mathbf{Y}}(k^2 c_{\mathbf{X}}) + c_{\mathbf{Y}}. \tag{2.16}$$

### 2.2.2 Sloji združevanja

V Poglavju 2.1 smo omenili, da velikost filtra določa širino okolice, ki jo konvolucija upošteva pri izračunu izhodne vrednosti. Če želimo povečati širino te okolice, da bi na sliki lahko zaznali večje strukture, lahko povečamo velikost filtrov. Vendar z večanjem velikosti filtrov hitro narašča tudi število učljivih parametrov, čemur pa se želimo izogniti. V konvolucijskih nevroskih mrežah problem običajno rešujemo s sloji združevanja. Ti elemente slike na vhodu z ustrezno operacijo zgostijo in sliko zmanjšajo na novo velikost. S tem zmanjšamo strukture na sliki in omogočimo, da jih nadaljnji konvolucijski sloji kljub majhnim filtrom zaznajo. Pravimo da povečujemo zaznavno polje kovolucijskih slojev. Le-to nam pove, kako velika je okolica na vhodu konvolucijske mreže, ki vpliva na rezultate tega sloja.

Sloji združevanja se običajno pojavijo vsakih nekaj konvolucijskih slojev in razpolovijo širino in višino slike. Združevanje deluje neodvisno na vsakem kanalu vhoda. Podobno kot pri konvoluciji gledamo elemente znotraj določene okolice (bloka) in iz njih izračunamo nov element, le da je operacija običajno fiksna (nima učljivih parametrov). Najpogosteje uporabljamo bloke velikosti $2 \times 2$ in korak 2. To pomeni, da po štiri elemente vhoda združimo v enega, slika pa se pri tem razpolovi po širini in višini.

Zmanjševanje slike ima več posledic. Omenili smo že povečevanja zaznavnega polja konvolucijskih slojev, ki sledijo sloju združevanja. Če sliko razpolovimo po širini in višini, en piksel vsebuje informacije štirih pikslov originalne slike. To se zgodi vsakič, ko v mreži uporabimo sloj združevanja. Širina okolice tako s številom slojev združevanja eksponento narašča, medtem ko velikost filtrov in z njo število parametrov ostaja enako. Zato lahko s sorazmerno majhnimi filtri v sliki detektiramo tako majhne, kot velike strukture.

Zaradi občutnega zmanjšanja števila aktivacij vseh nadaljnih slojev, do katerega pride zaradi zmanjševanja slike, se poveča tudi računska učinkovitost. Slabost združevanja pa je izguba informacije, ki se pojavi, ko združujemo več elementov v enega. Zato je še posebej pomembno na kakšen način agregiramo elemente bloka v novo vrednost. Poznamo več vrst združevanja, najpogosteje pa se uporabljata združevanje z maksimumom in združevanje s povprečenjem.

**Združevanje s povprečenjem** Združevanje s povprečenjem (angl. average pooling) pri združevanju nov element izračuna kot povprečje elementov v bloku. Operacija je identična dvodimenzionalni konvolucji, pri kateri so vsi elementi filtra enaki

$$\mathbf{W}[i, j] = \frac{1}{k^2}, \tag{2.17}$$

kjer je $k$ velikost filtra $\mathbf{W}$.

Lahko bi rekli, da združevanje s povprečenjem zgladi aktivacije in prikaže povprečno stanje bloka. Ta način je v uporabi prevladoval na začetku ra-

zvoja konvolucijskih nevronskih mrež, nato pa ga je kot najpogostejša metoda združevanja zamenjalo združevanje z maksimumom.

**Združevanje z maksimumom** Združevanje z maksimumom (angl. max pooling) za nov element vzame največjega izmed elementov v bloku. S tem poskrbimo, da se pomembne močne aktivacije prenesejo v naslednji sloj in se ne izgubijo v povprečju. Izgubimo pa informacijo o manjših aktivacijah znotraj bloka.

Združevanje z maksimumom se za potrebe združevanja zdi bolj intuitivno kot združevanje s povprečenjem. Združevanje z maksimumom na izhod prepiše najbolj močne aktivacije, ki vsebujejo veliko informacije, medtem kot jih združevanje s povprečenjem zgladi. Zato se v praksi združevanje z maksimumom uporablja zelo pogosto, sploh pri klasifikacijskih problemih, kjer želimo zgolj detektirati strukture na sliki, lokacija pa nam ni tako pomembna.

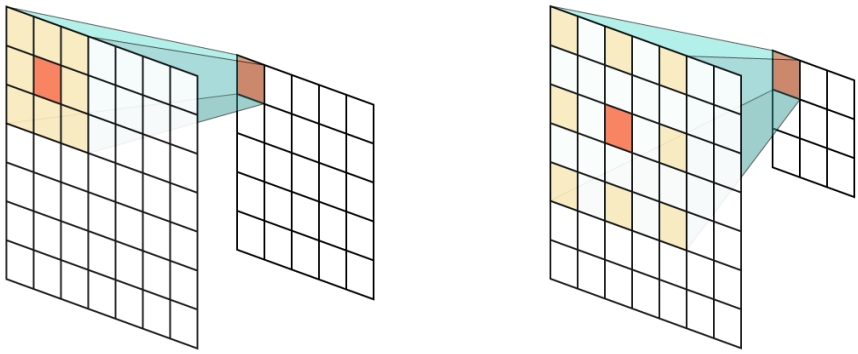
### Združevanje s konvolucijskimi sloji

Omenjeni sloji združevanja za agregacijo elementov uporabljajo fiksno funkcijo. Maksimalno združevanje je zelo uporabno pri detekciji struktur, če nam je pomembna tudi lokacija strukture, pa takšno združevanje zavrže pomembne informacije o lokaciji. Danes se za združevanje čedalje bolj uporabljajo kar konvolucijski sloji s korakom. Tako se mreža lahko nauči, katere informacije so pri napovedi pomembne in katere lahko zavrže. Zato je takšno združevanje splošnejše in lahko prinese izboljšave.

Springenberg et. al. [27] so z eksperimentalno analizo pokazali, da združevanje s konvolucijami ne zmanjšuje natančnosti mreže pri klasifikaciji, kjer se zdi vloga združevanja z maksimumumom še najbolj smiselna. Z uporabo učljivih konvolucij pa povečamo tudi število parametrov ter upočasnimo učenje. V primerih, kjer konvolucije niso potrebne, je torej bolje uporabiti fiksne sloje združevanja.

### Razširjena konvolucija

Razširjena (angl. dilated/atrous) konvolucija, nudi drugačen pristop k povečevanju zaznavnega polja mreže. Namesto da zmanjšujemo prostorsko dimenzijo slike, večamo domet konvolucijskih filtrov. To storimo z umetnim razširjanjem fitrov. Med elemente filtra vstavimo ničle (glej Sliko 2.10) in tako povečamo velikost filtrov, hkrati pa ohranimo število učljivih parametrov.



(a) Navadna konvolucija.

(b) Razširjena konvolucija, vstavljanje enojnega prostora.

Slika 2.10: Prikaz delovanja razširjene konvolucije. Oranžni elementi predstavljajo centre konvolucije.

Razširjena konvolucija torej z enakim številom parametrov povečuje zaznavno polje mreže in ohranja velikost slike. Posledično ne izgubljamo informacij o poziciji. Izgubimo pa performančne prednosti običajnih slojev združevanja, saj operacije vedno izvajamo na originalni velikosti slike.

## 2.2.3 Sloji razširjanja

V številnih aplikacijah potrebujemo sloje, ki izvajajo ravno obratno operacijo od slojev združevanja: reprezentacijo razširijo na večjo prostorsko velikost.

Taki sloji so zelo uporabni pri generativnih modelih [24] in pri semantični segmentaciji [20], kjer želimo iz zgoščenih reprezentacij generirati segmentacijsko masko. Tudi tu lahko izbiramo med različnimi metodami, odvisno od problema, ki ga rešujemo. Prav tako lahko izbiramo med fiksnimi metodami povečevanja, ali pa se znova poslužimo konvolucij.

### Razširjanje z interpolacijo

Razširjanje z interpolacijo (angl. interpolation) je fiksna metoda povečevanja velikosti podatkov, ki razprši elemente orignalne reprezentacije, vmesne elemente pa izračuna s pomočjo interpolacijske funkcije. V nadaljevanju so opisane najpogostejše metode interpolacije v eni dimenziji: interpolacija najbližjih sosedov, linearna interpolacija, kubična interpolacija.

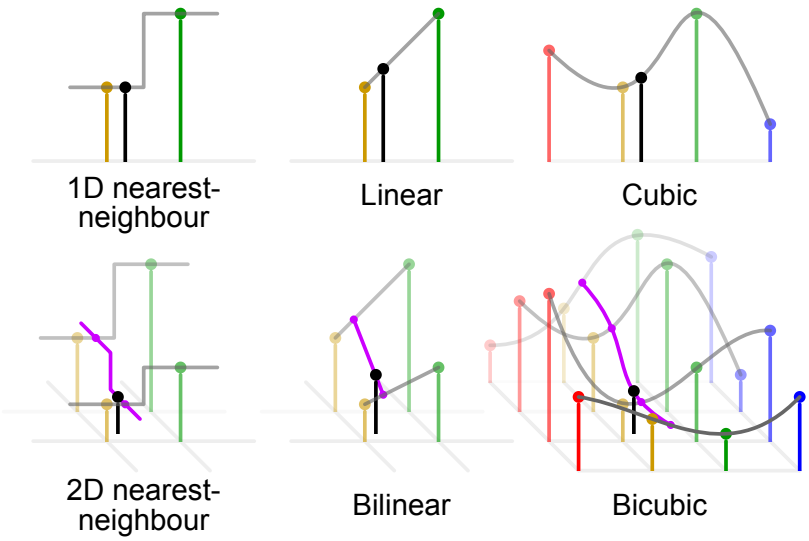
**Interpolacija najbljižjih sosedov** (angl. nearest neighbour interpolation) manjkajoče elemente v povečani reprezentaciji oceni tako, da poišče najbližji obstoječ element v originalni reprezentaciji in ga prepiše (glej Sliko 2.11).

**Linearna interpolacija** (angl. linear interpolation) za izračun vrednosti elementa gleda razdaljo do dveh najbližjih znanih elementov levo in desno od elementa in z uporabo teh razdalj izračuna vrednost po linearni funkciji

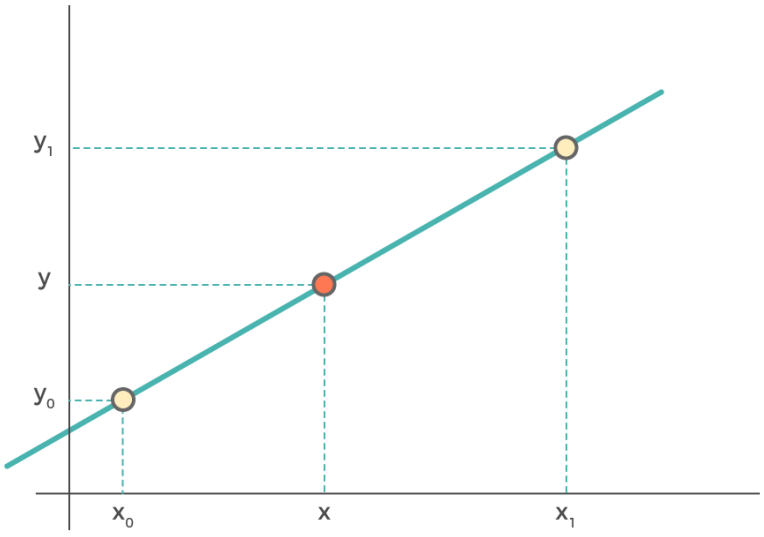
$$y = \frac{y_0(x_1 - x) + y_1(x - x_0)}{x_1 - x_0}, \tag{2.18}$$

kjer sta $T(x_0, y_0)$ in $T(x_1, y_1)$ znani točki, $T(x, y)$ pa je interpolirana točka (glej Sliko 2.12).

**Kubična interpolacija** (angl. cubic interpolation) vrednost izračuna z uporabo kubične funkcije, za kar potrebuje 4 najbližje točke. Rezultat je gladkejša krivulja prehoda med posameznimi točkami, saj upošteva še dodatni točki (glej Sliko 2.11).




Slika 2.11: Primerjava različnih metod interpolacije. S črno točko je označena interpolirana vrednost, z ostalimi barvami pa so označene najbližje točke, na podlagi katerih jo izračunamo. Slika je povzeta po [1].



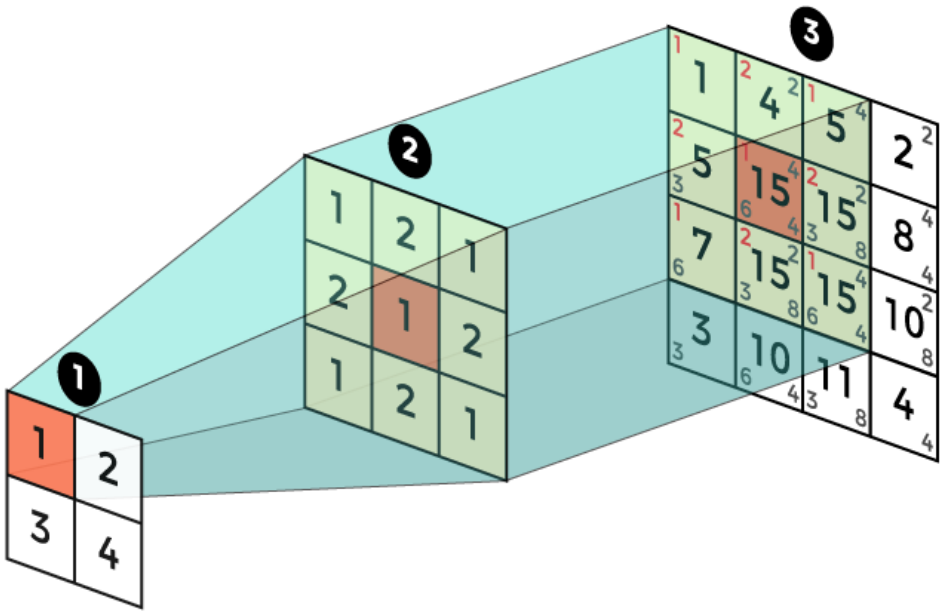

Slika 2.12: Linearna interpolacija. Z rumeno sta označeni znani točki, z oranžno pa interpolirana točka.

**Interpolacija v dveh dimenzijah** Za sloj razširjanja uporabljamo dvodimenzionalne različice omenjenih interpolacij. Te delujejo tako, da najprej zapolnijo manjkajoče vrednosti po eni dimenziji (vodoravno), nato pa ponovijo interpolacijo nad novimi elementi še po drugi dimenziji (vodoravno). Tako dobimo dvodimenzionalno različico interpolacije najbližjih sosedov, bilinearno in bikubučno interpolacijo (glej Sliko 2.11).

### Transponirana konvolucija

Razširjanje z interpolacijo je fiksno, deluje predvidljivo in v reprezentacijo ne dodaja dodatnih informacij, včasih pa je potreben bolj specifičen način razširjanja. Če imamo učljivo metodo, se ta lahko nauči posebnega načina razširjanja, ki upošteva karakteristike problemske domene. Za učljivo razširjanje lahko uporabimo posebno vrsto konvolucije – transponirano konvolucijo (angl. transpose convolution ali deconvolution). Gre za povsem običajen konvolucijski sloj, le da zamenjamo operaciji inference in vzvratnega propagiranja. V nasprotju z navadno konvolucijo, taka transponirana konvolucija razprši informacijo v elementu vhoda med vse elemente izhoda, ki jih filter prekrije (glej Sliko 2.13). Parameter korak še vedno določa za koliko premaknemo okno, le da okno tokrat premikamo na izhodnih podatkih. S povečevanjem koraka povečujemo velikost izhoda. Pri uporabi koraka 2 s pravilnim obdajanjem ničel, na izhodu dobimo natanko dvakrat večjo sliko.

Pri uporabi transponirane konvolucije je potrebno paziti pri izbiri hiperparametrov (korak in velikost filtrov). Problem je ustrezno predstavljen v članku [22]. Zelo hitro lahko pride do neenakomerne porazdelitve na izhodu, saj filter na nekaterih mestih pokrije več elementov vhoda kot na drugih. To na izhodu povzroči značilen vzorec šahovnice. Mreža se lahko nauči kompenzirati za to napako, vendar lahko težavo odpravimo sami. S tem tudi povečamo izkoristek mreže. Če nastavimo velikost filtrov in korak tako, da je velikost filtrov deljiva s korakom, zagotovimo enakomerno zastopanost elementov. Problem je najbolj izrazit pri generiranju slik. Tu imajo namreč že majhne spremembe velik vpliv na končno barvo.




Slika 2.13: Prikaz delovanja transponirane konvolucije na primeru (1 - vhodna matrika, 2 - konvolucijski filter, 3 - izhodna matrika). V izhodni matriki so malimi številkami prikazani prispevki posameznega vhoda. Z rdečo so označeni prispevki trenutnega vhodnega elementa.

### 2.2.4 Polno povezan sloj

Pri polno povezanem sloju je vsak element na izhodu z utežmi povezan z vsakim elementom vhoda. Vsak izhodni nevron ima uteži za vse vhodne povezave. Uteži so učljivi parametri sloja. En element izhoda dobimo kot skalarni produkt vhodov in uteži ter mu dodamo odmik. Polno povezan sloj lahko zapišemo z enačbo

$$\mathbf{Y} = f(\mathbf{XW} + \mathbf{b}), \tag{2.19}$$

kjer je $\mathbf{X}$ vhodna matrika, $\mathbf{W}$ matrika uteži, $\mathbf{b}$ vektor odmikov in $f(\cdot)$ aktivacijska funkcija.

Polno povezani sloji se pri konvolucijskih mrežah običajno pojavijo čisto na koncu, na odločitvenem nivoju. Prvi del konvolucijske nevronske mreže izračuna matriko značilk, ki ima še vedno prostorsko dimenzijo. Tu so značilke še vedno lokalizirane. Nakar reprezentacije sploščimo v eno dimenzionalni vektor. Polno povezan sloj zatem izloči značilke, ki so splošne za celo sliko, potem pa še en polno povezan sloj iz značilk izračuna verjetnosti posameznih razredov.

#### Pretvorba v konvolucijski sloj

Tako konvolucija kot polno povezan sloj elemente na izhodu računata kot skalarni produkt elementov vhoda z utežmi, le da konvolucija nima uteži za vsak element vhoda ampak jih deli med njimi. Če pa povečamo velikost filtrov konvolucije, da se ti ujemajo z velikostjo vhoda, potem na izhodu dobimo strukturo s prostorsko velikostjo $1 \times 1 \times c_Y$, vsak element vhoda pa ima svojo utež, kar je identično polno povezanemu sloju.

Taka implementacija polno povezanega sloja s konvolucijami se izkaže za zelo uporabno v praksi. Denimo, da imamo klasično konvolucijsko nevronsko mrežo s polno povezanimi sloji na koncu, ki smo jo učili na slikah velikosti $256 \times 256$. Če želimo klasificirati neko sliko, ki je drugačna od te velikosti (denimo večja), je ne moremo uporabiti kot vhod mreže, saj bo število nevronov

vhoda v polno povezan sloj drugačno kot pri učenju. Posledično bi potrebovali drugačno število uteži. Preostaneta nam dve opciji. Če želimo sliko zgolj klasificirati, jo lahko interpoliramo na velikost, ki jo mreža sprejema in potem zaženemo inferenco. Če pa želimo detektirati strukture v sliki, moramo izvesti nekakšen algoritem drsečega okna in na vsakem koraku izločiti območje prave velikosti, ter ga poslati skozi mrežo, kar je performančno zelo zahtevno.

Če pa polno povezan sloj implementiramo s konvolucijo, so vhodne slike lahko tudi večje. Ker je polno povezan sloj navadna konvolucija, deluje tudi na večjih vhodih, le da se, podobno kot algoritem drsečega okna, pomika po vhodih. Tako dobimo učinkovito verzijo zgoraj omenjenega algoritma z drsečim oknom, saj se vse konvolucije izračunajo zgolj enkrat. Medtem ko pri zgoraj omenjenem algoritmu večkrat poračunamo konvolucije nad istimi vhodi, saj je delež prekrivanja oken zelo velik.

Pri tem je potrebno poudariti, da algoritma nista povsem identična. Drugi način ni enak algoritmu drsečega okna s korakom 1, saj so vhodi v polno povezan sloj že nekajkrat reducirani v slojih združevanja. Drugi način je torej enak algoritmu drsečega okna s korakom, ki je enak celotnemu koraku mreže. Slednji je enak zmnožku korakov vseh slojev združevanja.

### Število učljivih parametrov

V tem odseku predstavimo število učljivih parametrov polno povezanega sloja. Vsak par vhoda in izhoda ima svojo utež. Če je število elementov na vhodu enako $N_{\mathbf{X}}$ in je število elementov na izhodu enako $N_{\mathbf{Y}}$, potem je število elementov matrike uteži enako

$$N_{\mathbf{W}} = N_{\mathbf{Y}} N_{\mathbf{X}}. \tag{2.20}$$

Vektorju izhoda dodamo še vektor odmika enake dolžine. Število vseh učljivih parametrov polno povezanega sloja je torej

$$N_{\text{par}} = N_{\mathbf{W}} + N_{\mathbf{b}} = N_{\mathbf{Y}} N_{\mathbf{X}} + N_{\mathbf{Y}}. \tag{2.21}$$

Zaradi velikega števila parametrov lahko pri uporabi polno povezanih slojev hitro pride do prekomernega prilagajanja (angl. overfitting). Da bi se temu izognili, se v fazi učenja poleg polno povezanega sloja običajno uporabi še sloj izpuščanja (angl. dropout).

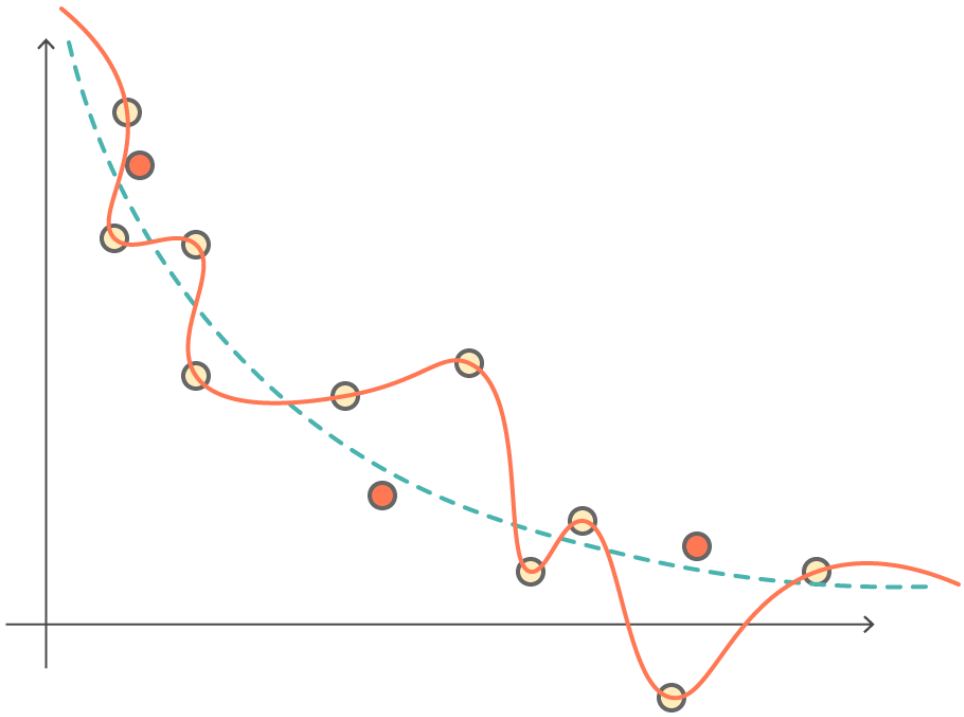
### 2.2.5 Sloj izpuščanja

Ko se model preveč prilagodi na učne podatke in posledično slabše generalizira nove primere, govorimo o prekomernem prilagajanju. Tak model odlično deluje na učni množici, ko pa ga poženemo na primeru, ki ga ni bilo v učni množici, natančnost zelo upade (glej Sliko 2.14). Zaradi ogromnega števila parametrov so globoke nevronske mreže še posebej nagnjene k prekomernemu prilagajanju.

Problem prekomernega prilagajanja lahko zmanjšamo na več različnih načinov. Najbolj zanesljiv način je velika učna množica. Če imamo dovolj zajetno in raznoliko učno množico, ki pokrije veliko večino možnih primerov, do prekomernega prilagajanja pride precej težje. Prav tako to velja za preproste modele. Preprostejši kot je model (po številu učljivih parametrov), bolje se nauči generalizirati.

Za zahtevnejše probleme, pa preprost model ni dovolj. Globoke nevronske mreže imajo veliko moč, zato z njimi lahko predstavimo zelo kompleksne modele. Vendar pa so globoke nevronske mreže, zaradi velikega števila parametrov, sploh v zadnjih polno povezanih slojih, nagnjene k prekomernemu prilagajanju. Mreža se lahko začne preveč zanašati na aktivacije le nekaterih nevronov, druge pa ignorira. Temu se lahko izognemo s slojem izpuščanja (angl. dropout).

Sloj izpuščanja v vsakem koraku učenja naključen del aktivacij vhoda postavi na 0, ostale pa nespremenjene prepiše na izhod. S tem mrežo prisili v to, da informacije enakomerno razporedi po vseh aktivacijah ter efektivno zmanjšuje možnost prekomernega prilagajanja. Delež zavrženih aktivacij (stopnja dropouta) je običajno precej velik (okrog 50%). Po končanem učenju (pri inferenci), sloje dropout izklopimo, da vedno ohranimo vse aktivacije in



Slika 2.14: Prekomerno prilagajanje. Z rumeno barvo so označene točke učne množice, z oranžno pa točke testne množice. Funkcija označena z oranžno barvo se preveč prilega učnim podatkom, zato slabo generalizira. S turkizno barvo je prikazana primerna funkcija, ki bolje generalizira in dobro deluje tudi na testnih podatkih.

ne zavržemo uporabnih informacij.

## 2.3 Učenje mreže

Konvolucijsko nevronsko mrežo učimo z uporabo optimizacije. Definirati moramo cenilno funkcijo, ki nam pove kako veliko napako je mreža naredila pri napovedovanju glede na učno bazo. To funkcijo želimo minimizirati na celotni učni množici. Optimizacijski algoritem na vsakem koraku požene mrežo na delu učne množice, izračuna napako in jo s postopkom vzvratnega propagiranja napake (angl. back propagation) postopoma propagira proti začetku mreže ter ustrezno popravlja uteži. Uteži se popravljajo v nasprotni smeri izračunanih gradientov. S tem poskrbimo, da se napaka postopoma zmanjšuje, cenilna funkcija pa konvergira proti lokalnemu minimumu. V nadaljevanju bomo opisali pomen cenilne funkcije.

### Cenilna funkcija

S cenilno funkcijo ocenjujemo napako napovedi. Mreža se uči z minimiziranjem te funkcije. Zato moramo izbrati primerno cenilno funkcijo, ki se zvezno spreminja z odstopanjem od pravilnih napovedi, ter ne vsebuje prevelikega števila lokalnih minimumov. Z $\hat{y}$ označujemo napovedano vrednost, ki je izhod iz zadnjega sloja, z $y$ pa pravo vrednost. Mrežo lahko predstavimo kot eno veliko funkcijo. Sestavljena je iz kompozituma funkcij posameznih slojev mreže (vhod sloja je izhod predhodnega sloja). Napovedano vrednost lahko torej zapišemo kot funkcijo

$$\hat{y} = f(x) = f_{l_1} \circ f_{l_2} \circ \cdots \circ f_{l_n}(x), \tag{2.22}$$

kjer je $x$ vhod mreže, $f_{l_1}, f_{l_2}, ..., f_{l_n}$ pa funkcije slojev $l_1, l_1, ..., l_n$.

Najpogostejša cenilna funkcija za semantično segmentacijo je križna entropija (angl. cross entropy). Ta je definirana po enačbi

$$c(\hat{y}, y) = -\sum_i y_i \, log(\hat{y}_i), \tag{2.23}$$

po vseh razredih $i$. Z $y_i$ označujemo pravilno vrednost za razred $i$. Prireditev te funkcije na problemsko domeno opišemo v Poglavju 3.2.

# Poglavje 3

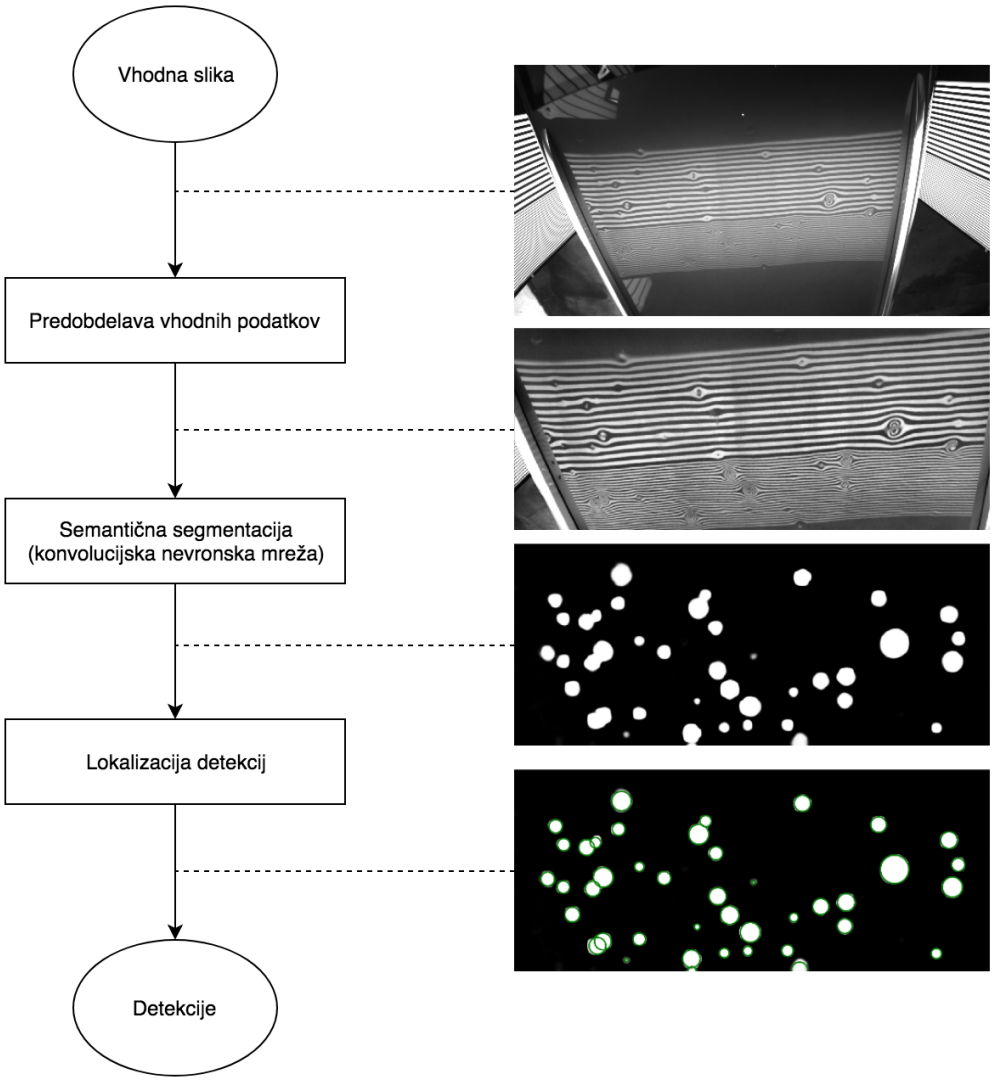
# Metoda za lokalizacijo udrtin

Problem detekcije anomalij definiramo kot tristopenjski postopek. V prvi fazi (faza predobdelave) normaliziramo in pripravimo vhodne slike. V drugi fazi, na podlagi vhodne slike, za vsak slikovni element ocenimo verjetnost, da pripada anomaliji. Rezultat tega postopka je binarna maska, kjer skupki pikslov nakazujejo potencialne udrtine. V tretji fazi preko analize slikovnih elementov maske določimo položaje in velikosti udrtin. Shema treh faz je prikazana na Sliki 3.1.

Na drugo fazo lahko gledamo kot na problem semantične segmentacije. Vsak piksel slike želimo klasificirati v enega izmed dveh razredov: anomalija, ki ga označimo s $P$ ali ne-anomalija, ki ga označimo z $N$. Za vsak razred mreža na izhodu vrne njegovo verjetnost. Ker sta razreda le dva in sta komplementarna, lahko en razred zanemarimo, ter napovedujemo le verjetnost razreda $P$. Verjetnost nasprotnega razreda je tako enaka

$$P(N) = 1 - P(P). \tag{3.1}$$

Izhod mreže je maska, kjer vrednost vsakega elementa predstavlja verjetnost, da se na tem mestu nahaja anomalija. Če je vrednost blizu 0, pomeni, da na tem mestu najverjetneje ni anomalije, če pa je blizu 1, na tem mestu anomalija najverjetneje je. Primer segmentacijske maske je prikazan na Sliki 3.1. Izhod mreže gre nato v metodo lokalizacije udrtin, ki iz segmen-




Slika 3.1: Shema cevovoda metode. Prikazane so tudi naslednje slike od zgoraj navzdol: vhodna slika, slika po predobdelavi vhodnih podatkov, segmentacijska maska in lokalizirane detekcije na segmentacijski maski.

tacijske maske izlušči informacije o lokacijah in velikostih udrtin. V tem poglavju bomo opisali zgradbo in delovanje posameznih delov metode.

## 3.1 Predobdelava vhodnih podatkov

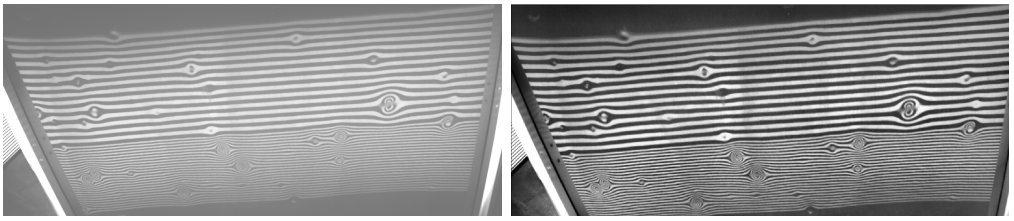
Podatki o širini vzorca, ki jih dobimo iz sistema, omogočajo, da z uporabo normalizacije zmanjšamo zahtevnost problema, ki ga model rešuje. Uporabimo lahko tudi standardne metode izenačevanja svetlosti, ki zmanjšujejo razlike v svetlobnih pogojih med slikami.

### 3.1.1 Prilagoditev skale

V tipični aplikaciji se velikost vzorca spreminja glede na oddaljenost od kamere. Bližje kot je površina objekta kameri, večja je širina vzorca. Model bi se moral naučiti detekcije udrtin pri zelo različnih povečavah. Problem lahko malce poenostavimo z informacijo o oddaljenosti površine, ki jo dobimo iz sistema. Sistem izmeri približno oddaljenost površine. Glede na to informacijo slike povečamo ali pomanjšamo in dosežemo učinek, kot da so posnete na enaki razdalji. S tako normalizacijo poskrbimo, da sta širina vzorca in velikost udrtin na vseh slikah približno enaka.

### 3.1.2 Standardizacija

Nevronske mreže in njene aktivacijske funkcije so načrtovane in najbolje delujejo na vrednostih v bližini intervala $(-1, 1)$. V bazi imamo 8 bitne slike, katerih elementi so celoštevilske vrednosti med 0 in 255. Da bi interval vrednosti prestavili na ustrezen interval, izvedemo postopek standardizacije. Slika je standardizirana, če je njena povprečna vrednost enaka 0, njen standardni odklon pa je enak 1. Sliko standardiziramo tako, da iz slike najprej poračunamo povprečno vrednost ter standardno deviacijo. Zatem od vsake vrednosti v sliki odštejemo povprečno vrednost in dobljeno vrednost delimo s standardno deviacijo.



(a) Originalna slika.

(b) Slika po adaptivnem izenačevanju hitogramov.

Slika 3.2: Adaptivno izenačevanje histogramov.

### 3.1.3 Izenačevanje histogramov

Fotografije so pogosto zajete pod različnimi svetlobnimi pogoji in pri različnih barvah pločevine. Slike se zato precej razlikujejo po kontrastu in distribuciji svetlosti. To razliko želimo kar se da minimizirati, pri čemer uporabimo adaptivno izenačevanje histogramov [23]. Običajno izenačevanje histogramov [6] svetlosti na sliki razporedi na tak način, da je histogram slike čim bolj enakomerno razporejen po celotnem spektru svetlosti. Tako poskrbimo za predvidljiv kontrast na sliki in zmanjšanje razlik med različnimi slikami istih objektov. Adaptivno izenačevanje temelji na istem principu, le da histograme na posameznih delih slike izenačuje ločeno. S tem zagotovimo dober lokalni kontrast in izrazite robove. V našem primeru postopek poskrbi za jasne meje med črtami vzorca (glej Sliko 3.2).

## 3.2 Semantična segmentacija

Semantična segmentacija predstavlja jedro razvite metode. Sestavlja ga konvolucijska nevronska mreža, ki kot vhod sprejme sliko, na izhodu pa vrne segmentacijsko masko anomalij. V tem podpoglavju bomo opisali predlagano arhitekturo in postopek učenja mreže.

### 3.2.1 Arhitektura segmentacijske mreže

Tipična arhitektura mreže za semantično segmentacijo je sestavljena iz dveh delov: iz enkoderja in dekoderja. Enkoder iz slike izlušči značilke, dekoder pa na podlagi slednjih generira segmentacijsko masko. Ena najbolj znanih arhitektur je t.i. arhitektura polno konvolucijske mreže (angl. fully convolutional network, FCN) [16].

Pred FCN so tipične mreže za segmentacijo za enkoder vzele klasično konvolucijsko nevronsko mrežo (npr. AlexNet [13], Inception [28], VGG16 [25]), ki iz slike fiksne velikosti izlušči vektor značilk. Dekoder je ponavadi simetrična preslikava enkoderja, le da namesto slojev združevanja uporabljamo sloje razširjanja. Pri taki arhitekturi lahko na vhod dajemo le slike fiksne velikosti. Ta problem FCN odpravi tako, da odstrani polno povezan sloj iz enkoderja in ga nadomesti s konvolucijskim slojem (glej Poglavje 2.2.4) . Na vhod tako lahko dajemo slike poljubnih velikosti, paziti moramo le, da sta višina in širina deljiva s skupnim korakom mreže. Na izhodu dobimo masko velikosti vhoda.

Druga pomanjkljivost starih arhitektur je, da se izgubi veliko lokalnih informacij. Zaradi zmanjševanja velikosti slike skozi mrežo, se izgubljajo informacije na višji resoluciji, kar se pozna pri končni maski, ki je ponavadi zelo slabo lokalizirana. FCN ta problem rešuje tako, da vpelje tako imenovane preskočne (angl. skip) povezave. Gre za to, da del aktivacij iz zgornjih slojev enkoderja pripeljemo v simetrične sloje v dekoderju. Mreži je tako omogočeno združevanje globalne in lokalne informacije, kar se pozna pri bolj natančni maski.

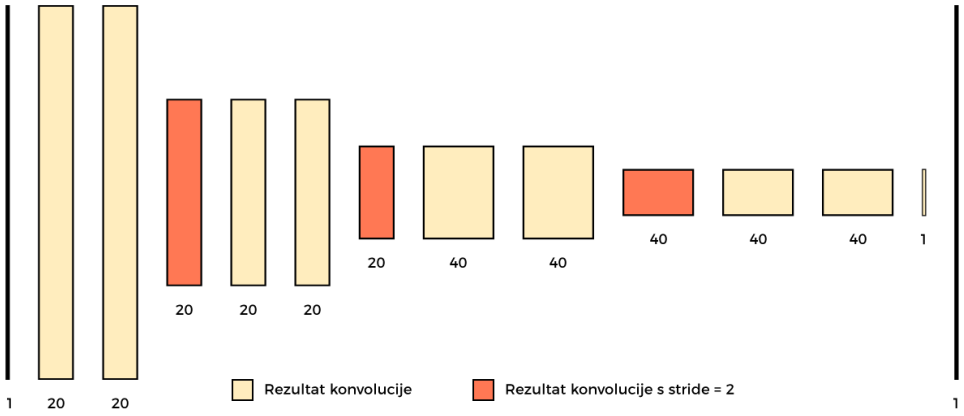
Pri načrtovanju arhitekture smo se zato najprej zgledovali po arhitekturi FCN. Po preliminarni analizi smo prišli do sklepa, da lahko zaradi narave problema arhitekturo bistveno poenostavimo in pospešimo metodo. Model enkoder dekoder je primeren, kadar želimo natančno segmentacijo objektov, ki pokrivajo velik del slike. V našem primeru je segmentacija zgolj korak pri določanju detekcij. Ne zanima nas točna segmentacija udrtine od ozadja, ampak zgolj njena lokacija in radij. Dovolj natančno informacijo lahko do-

bimo z manjšo mrežo. Končna arhitektura (glej Sliko 3.3) vsebuje naslednjih 13 slojev:

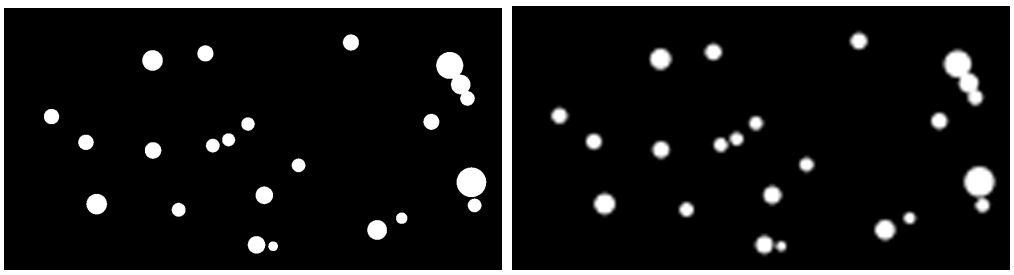
| **vhod** |
|---|
| konvolucija 3x3x20<br>konvolucija 3x3x20<br>konvolucija 3x3x20, korak=2 |
| konvolucija 3x3x20<br>konvolucija 3x3x20<br>konvolucija 3x3x20, korak=2 |
| konvolucija 3x3x40<br>konvolucija 3x3x40<br>konvolucija 3x3x40, korak=2 |
| konvolucija 3x3x40<br>konvolucija 3x3x40<br>konvolucija 3x3x1 |
| bikubična interpolacija 8x |
| **segmentacijska maska** |

Tabela 3.1: Arhitektura mreže.

Slike gredo skozi tri sloje združevanja, vsak izmed njih sliko razpolovi po širini in višini. Skupen korak enkoderja je enak zmnožku vseh korakov, to je $2 \times 2 \times 2 = 8$. Izhod enkoderja je torej slika, ki je osemkrat manjša po širini in višini kot vhod v mrežo. Dekoder smo poenostavili v enostavno bikubično razširitev slike na originalno velikost. Prvič, narava problema ne zahteva velike natančnosti segmentacijske maske. Natančno območje udrtine je težko določiti, zato smo anotacije poenostavili na krogelne elemente, ki približno opišejo območje vdrtine. Ker to območje ni povsem natančno, je nesmiselna uporaba preveč kompleksne mreže, ki bi poskušala jasno določiti meje udrtine. Reducirana maska, ki je izhod enkoderja je že dovolj podrobna, da lahko določimo približno okolico vdrtine, ki je zapisana v anotacijah (glej Sliko 3.4). Bolj podroben opis komponent arhitekture bomo podali v nadaljevanju.




Slika 3.3: Arhitektura naše mreže. Številke pod sloji predstavljajo število kanalov.



(a) Originalna segmentacijska maska. (b) Interpolirana segmentacijska maska.

Slika 3.4: Primerjava originalne segmentacijske maske in segmentacijske maske rekonstruirane z bikubično interpolacijo osemkrat pomanjšane originalne slike. Na obeh maskah so detekcije jasno ločene. Dobro ohranjena je tudi informacija o velikosti.

### 3.2.2 Hiperparametri

Zastavljena nevronska mreža vsebuje veliko hiperparametrov, ki jih je potrebno smiselno nastaviti. Nekatere lahko ustrezno izberemo že po preliminarni analizi, za druge pa je potrebno izvesti primerne eksperimente, s katerimi lahko utemeljimo našo izbiro. Tu bomo definirali hiperparametre mreže ter pojasnili izbiro vrednosti, ki smo jih določili s preliminarno ana-

lizo. Eksperimente, ki smo jih izvedli za določitev ostalih hiperparametrov ter dobljene rezultate, bomo opisali v Poglavju 4.

### Število slojev

Število slojev določa kompleksnost modela. Predsvem je pomembno število konvolucijski slojev. Večje kot je število slojev, bolj kompleksne funkcije je sposoben model predstaviti. Po preliminarni analizi smo se odločili za model, ki vsebuje tri sloje združevanja, kar se nam zdi ustrezen kompromis med natančnostjo in količino globalne informacije. Izhod mreže ima osemkrat nižjo ločljivost od vhoda, kar je še dovolj, da razberemo pozicije in velikosti anomalij (glej Sliko 3.4). Poleg tega taka mreža dobi dovolj okoliške informacije za klasifikacijo anomalije. Število konvolucijskih slojev, ki jih vstavimo med posamezen sloj združevanja in končni predikcijski sloj smo določili z eksperimentom (glej Poglavje 4).

Skupno število slojev je precej manjše od večine sodobnih konvolucijskih mrež, ki se uporabljajo pri semantični segmentaciji ali detekciji objektov. Te mreže so učene na velikem številu razredov, primeri v posameznih razredih pa so bolj raznoliki kot primeri udrtin. Naš problem ne zahteva take kompleksnosti, zato smo se odločili za preprostejšo mrežo, ki je tudi računsko učinkovitejša.

### Število konvolucijskih filtrov

Število konvolucijskih filtrov določamo vsakemu konvolucijskemu sloju posebej. Držali smo se konvencije večine mrež, ki običajno pri vseh konvolucijskih slojih na istem nivoju uporabljajo enako število filtrov.

Število filtrov smo določili z eksperimentalno analizo (glej Poglavje 4), izhajali pa smo iz naslednje razporeditve. Na prvem nivoju (originalna velikost) smo uporabili po 16 filtrov, na drugem 32, na tretjem 64 in na četrtem 128. Z vsakim slojem združevanja se število filtrov podvoji. Tako porazdelitev smo izbrali po naslednji logiki. Zgornji sloji se učijo bolj splošnih značilnosti (na primer robovi), za kar ne potrebujejo toliko filtrov. Glo-

blje kot gremo, bolj sloji združujejo osnovne aktivacije v bolj kompleksne značilnosti. Zadnji konvolucijski sloj vsebuje en filter. Potemtakem na izhodu dobimo sliko z enim kanalom, ki vsebuje predikcije.

Pričakujemo manjše število filtrov, kot jih uporablja večina sodobnih mrež, saj se osredotočamo zgolj na en razred in je problem sorazmerno lahek.

### Velikost konvolucijskih filtrov

Velikost filtrov določa velikost okolice, ki jo konvolucija upošteva. Običajno se uporabljajo simetrični filtri velikosti $2n + 1$, kjer je $n \in \mathbb{N}$. S tem zagotovimo enako pokritost elementov na vse strani od središčnega elementa. Najpogosteje se za filtre uporablja velikost $3 \times 3$, kar je minimalna velikost filtra, da kovolucija upošteva še sosede in ne le elementa samega. Pogosto se uporabljajo tudi filtri velikosti $5 \times 5$, večji od tega pa le v izjemnih primerih, saj povečevanje velikosti filtrov hitro povzroči ogromno število učljivih parametrov. Namesto povečevanja velikosti filtrov lahko podoben učinek dosežemo z dodajanjem slojev, kar nam zagotovi še druge prednosti. Tudi mi smo sledili konvenciji in izbrali filtre velikosti $3 \times 3$.

### Hitrost učenja

Hitrost učenja (angl. learning rate) je parameter, s katerim določamo velikost popravkov uteži, ki jih izvajamo med učenjem. Z njim pomnožimo izračunane gradiente in s tem povečamo ali zmanjšamo magnitudo sprememb uteži. Hitrost učenja vpliva na to, kako hitro bo naš model konvergiral med učenjem. Če hitrost učenja nastavimo na nizko vrednost, se uteži posodabljajo po zelo majhnih korakih, čas učenja do konvergence pa je dolg. Če pa je prevelika, se vrednosti uteži spreminjajo po zelo velikih korakih in lahko se zgodi, da preskočimo lokalni minimum, kar povzroči da učenje ne konvergira ampak divergira. Želimo si neko srednjo pot med obema skrajnostima.

Izbira bolj naprednega optimizacijskega algoritma (npr. Adam [12], RMSprop [9]), kjer se hitrost učenja spreminja skozi postopek učenja ali pa je ločena za posamezen parameter mreže, zmanjša pomen nastavljanja začetne

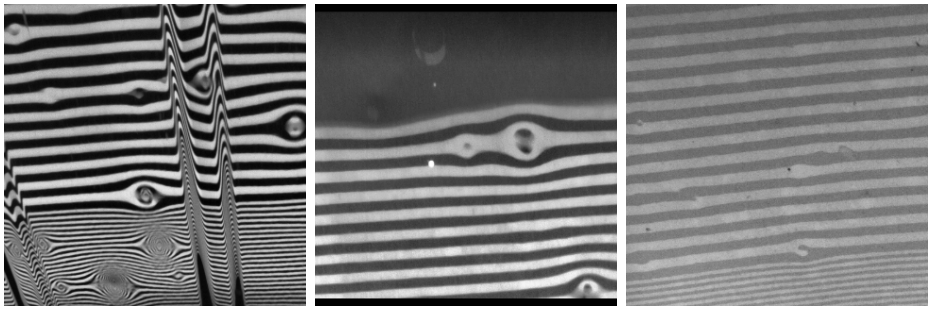
hitrosti učenja, saj lahko algoritem hitro popravi prevelike ali premajhne začetne vrednosti. V takih primerih je dovolj, če hitrost učenja nastavimo približno. Uporabili smo optimizacijsko metodo Adam, vrednost parametra pa smo določili s preprostim poskusom. Hitrost učenja smo spreminjali po različnih velikostnih redih (0.0001, 0.001, ...) ter vsakič pognali učenje. Izbrali smo vrednost, pri kateri je bil videz krivulje učenja najbolj zgleden.

### Velikost izrezanega okna in velikost paketa

Zaradi naše izbire arhitekture je vhod v mrežo lahko poljubne velikosti, le širina in višina morata biti deljiva z 8, saj je celoten korak mreže enak 8. Najmanjša možna slika, ki jo lahko uporabimo kot vhod v mrežo je torej velikosti $8 \times 8$. Navzgor te omejitve v teoriji ni, v praksi pa smo omejeni s pomnilnikom grafične kartice. Proces učenja si namreč na grafični kartici rezervira pomnilnik za vse aktivacije na vseh slojih, ter njihove gradiente, število le teh pa se povečuje z velikostjo slik na vhodu. V bazi podatkov imamo zelo velike slike. Mreže, ki bi jo potrebovali za tako vhodno velikost ne moremo spraviti v pomnilnik grafične kartice. Potrebno je najti primeren kompromis med velikostjo izreza in številom slik v paketu.

Pri majhni velikosti okna mreža vidi le majhen del vhodne slike. Če je ta premajhen, se mreža ne more naučiti kako okolica vpliva na napovedi, četudi arhitektura to podpira, saj teh podatkov preprosto ni v vhodni sliki. Z izbiro večjega okna se lahko izognemo tem težavam, saj imamo na voljo informacijo o večjem delu originalne slike. Izbrati moramo tako velikost, da mreža lahko v celoti izkoristi svoje arhitekturne lastnosti, sicer ustvarimo ozko grlo.

Zaznavno polje nam pove velikost okolice, ki vpliva na rezultate naše mreže. Navadni konvolucijski sloji velikost zaznavnega polja povečajo za $k-1$, kjer je $k$ velikost filtra. Takšno je namreč število sosednjih elementov, ki jih upoštevamo v posamezni dimenziji. Po sloju združevanja se slika razpolovi, zato se povečanje podvoji. Ti dve relaciji lahko zapišemo z enačbama



Slika 3.5: Primer izrezov v učni množici.

$$\begin{aligned} j_{i+1} &= j_i s, \\ r_{i+1} &= r_i + j_i(k-1), \end{aligned} \tag{3.2}$$

kjer je $s$ korak, $k$ pa velikost filtra sloja $i+1$. Parameter $j_i$ predstavlja skok sloja $i$. Pove nam, za koliko se zaznavno polje poveča pri upoštevanju enega sosednjega elementa. Vrednost $r_i$ predstavlja širino zaznavnega polja sloja $i$. Na vhodu sta vrednosti parametrov enaki

$$\begin{aligned} j_0 &= 1, \\ r_0 &= 1. \end{aligned} \tag{3.3}$$

Z enačbama izračunamo, da v naši arhitekturi v zadnjem sloju na vsak piksel maske vplivajo piksli v območju velikosti $91 \times 91$ vhoda. To pomeni, da mreža polno informacijo dobi le za območje, ki je vsaj 45 pikslov oddaljeno od roba slike na vhodu, za preostalo območe pa del informacije predstavljajo ničle zaradi obdajanja. Na robovih je zato učenje slabše. Želimo torej dovolj veliko okno, da bo večina udrtin v notranjem delu in ne na robovih. Izbrali smo velikost okna $320 \times 320$. Tako je notranji del okna, ki vsebuje popolno informacijo velik $230 \times 230$, kar je dovolj, da se veliko udrtin pojavi znotraj tega območja (glej Sliko 3.5).

Ko velikost izreza zmanjšamo, pridobimo še eno prednost. V pomnilniku ostane dovolj prostora, da lahko v enem koraku skozi mrežo v paketu pošljemo večje število izrezanih slik. Za velikost paketa uporabimo največje možno število slik, ki jih še spravimo v pomnilnik grafične kartice. S tem zagotovimo

največjo reprezentacijo učne množice in posledično bolj stabilno učenje.

### 3.2.3 Cenilna funkcija

Za cenilno funkcijo mreže smo uporabili križno entropijo, ki smo jo opisali v (2.23). Ker problem obravnavamo kot binarno klasifikacijo posameznih pikslov, lahko uporabimo binarno križno entropijo, ki upošteva verjetnost razreda ($y$) in nasprotnega razreda ($1 - y$). Definirana je po enačbi

$$\text{cost}_{\text{bin}}(\hat{y}, y) = -y\, log(\hat{y}) - (1 - y)\, log(1 - \hat{y})). \tag{3.4}$$

Križno entropijo izračunamo nad vsakim piklsom, nato pa izračunamo povprečje vrednosti, ki predstavlja končno vrednost cenilne funkcije. To lahko zapišemo z enačbo

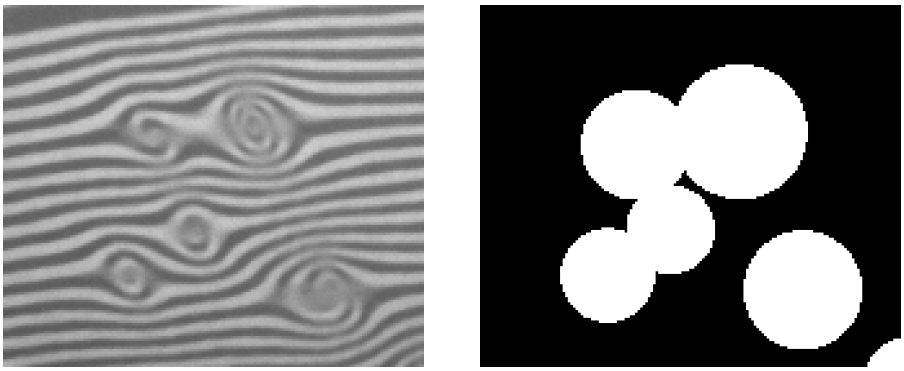
$$\text{cost}_{\text{total}}(\hat{\mathbf{Y}}, \mathbf{Y}) = \frac{\sum_{i=1}^{h_{\mathbf{Y}}} \sum_{j=1}^{w_{\mathbf{Y}}} \text{cost}_{\text{bin}}(\hat{\mathbf{Y}}[i,j], \mathbf{Y}[i,j])}{h_{\mathbf{Y}} w_{\mathbf{Y}}}, \tag{3.5}$$

kjer je $\hat{\mathbf{Y}}$ izhodna segmentacijska maska, $\mathbf{Y}$ pa referenčna segmentacijska maska.

## 3.3 Lokalizacija udrtin

Izhod metode segmentacije je semantična maska, ki za vsak piksel napoveduje verjetnost, da se na tem mestu nahaja udrtina. V učni množici so udrtine predstavljene s krogelnimi oznakami, zato tudi v izhodu mreže pričakujemo bolj ali manj krogelne elemente. Če bi bile udrtine vedno dovolj narazen, bi segmentacijsko masko sestavljali samostojni krogelni elementi, ki bi jih lahko preprosto označili z metodo povezanih komponent (angl. connected components) [26], ki binarno sliko razbije na posamezne povezane skupke (komponente). Vendar se že v učni množici pojavljajo primeri, ko se udrtine nahajajo dovolj skupaj, da se v segmentacijski maski njihovi krogelni odtisi prekrivajo (glej Sliko 3.6). Tak skupek udrtin bi metoda povezanih komponent zaznala kot enotno območje udrtine, kar bi bila solidna rešitev, želimo

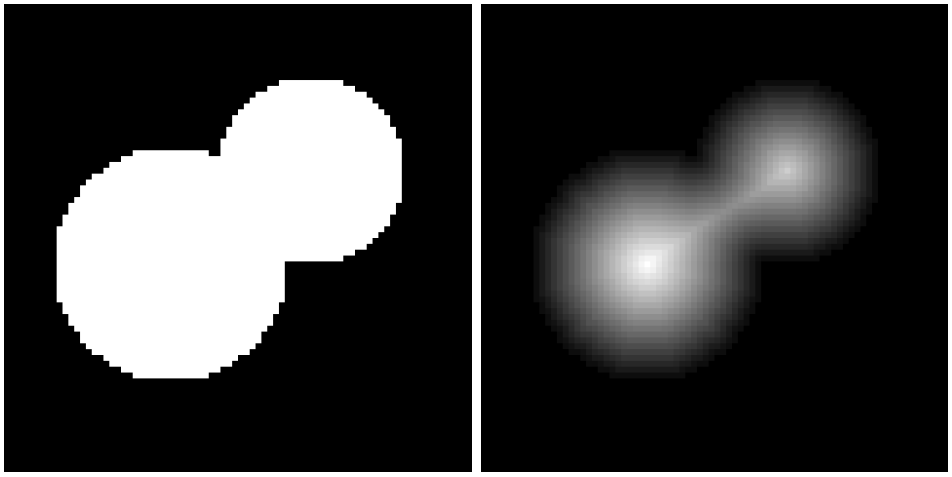
pa si robustnejši sistem. Za razbijanje skupka udrtin na posamezne detekcije smo izkoristili krogelne lastnosti udrtin.



Slika 3.6: Primer skupka v podatkovni bazi in pripadajoča segmentacijska maska.

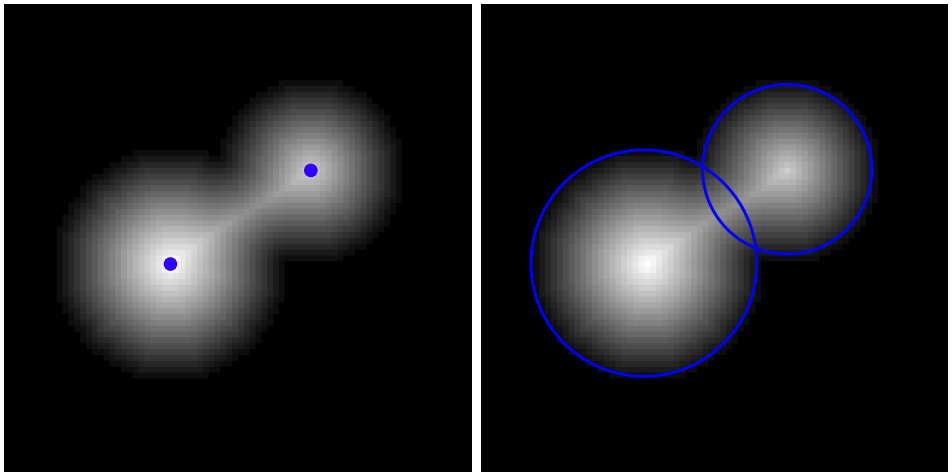
Središče kroga je točka, ki je najbolj oddaljena do najbližjega roba lika. Ko se iz središča premaknemo v katerikoli smeri, se ta razdalja zmanjšuje. Središče kroga je lokalni maksimum razdalje do najbližjega roba. Ko združimo dva ali več krogov, tako da je prekritost posameznega kroga manj kot polovica, se ta lastnost ohrani. Če merimo razdaljo točke do najbližjega roba skupka, dobimo lokalne maksimume v centrih krogov, ki sestavljajo skupek (glej Sliko 3.7). To lastnost smo izkoristili za določanje približnih centrov krogelnih detekcij.

Celoten postopek metode je prikazan na Sliki 3.7. Da lahko računamo razdaljo do roba, moramo masko najprej binarizirati. To storimo z upragovanjem. Mejo upragovanja smo nastavili na 0.5. Točke, ki so nad mejo upragovanja označimo za pozitivne, ostale pa za negativne. Nad binarno sliko izvedemo algoritem računanja oddaljenosti (angl. distance transform) [26], ki za vsako pozitivno točko v sliki poišče razdaljo do najbližje negativne točke. V tej matriki razdalj poiščemo lokalne maksimume, ki predstavljajo približne centre udrtin. Zaradi šuma in nizke ločljivosti se lahko v bližini pravilnega lokalnega maksimuma pojavi več lokalnih maksimumov. Ta šum odpravimo z morfološko dilatacijo [26], ki razširi posamezen lokalni maksimum za par



(a) Binarizirana slika segmentacijske maske.

(b) Matrika razdalj. Svetlost pikslov predstavlja razdaljo.



(c) Označeni lokalni maksimumi.

(d) Lokalizirane detekcije.

Slika 3.7: Prikaz stopenj metode za lokalizacijo detekcij.

pikslov, tako da se šumni vrhovi povežejo v enega. Iz slike izločimo vrhove z metodo povezanih komponent in za vsako komponento izračunamo masni center. Ta predstavlja center naše udrtine. Radij detekcije pa dobimo kot razdaljo centra do najbližjega roba. Preberemo jo iz prej izračunane matrike razdalj.

# Poglavje 4

# Eksperimentalna evalvacija

V tem poglavju opišemo eksperimente, ki smo jih izvedli za določitev ustreznih hiperparametrov mreže. Pokažemo rezultate evalvacije metode lokalizacije udrtin ter predstavimo rezultate končne evalvacije metode in primerjavo z osnovno metodo. V Poglavju 4.1 opišemo uporabljeno strojno in programsko opremo ter druge podrobnosti implementacije. V Poglavju 4.2 predstavimo pogoje in protokol, ki smo ju uporabili pri evalvaciji. Nato sledijo eksperimentalne evalvacije posameznih hiperaparametrov in evalvacija celotne metode.

## 4.1 Podrobnosti implementacije

### 4.1.1 Strojna in programska oprema

Model smo učili in evalvirali na strežniku v laboratoriju Vicos FRI. Specifikacije uporabljene strojne opreme so prikazane v Tabeli 4.1. Segmentacijska konvolucijska nevronska mreža je implementirana v programskem jeziku *python3*, v ogrodju *TensorFlow* [3]. Metoda predobdelave podatkov in metoda lokalizacije vdrtin uporabljata knjižnjici *numpy* in *OpenCV*.

| komponenta | specifikacije |
|---|---|
| CPU | INTEL Xeon E5-1650 v3 3.50GHz (6 jeder) |
| RAM | 64 GB |
| GPU | NVIDIA GeForce GTX 980 |

Tabela 4.1: Specifikacije strežnika.

### 4.1.2 Priprava učne množice

Za hitro in robustno učenje moramo učno množico ustrezno pripraviti. To vključuje odstranjevanje odvečnih informacij, ustrezno združevanje primerov v učne pakete in umetno razširjanje učne množice.

**Obrezovanje** Slike, ki jih zajema sistem, obsegajo široko območje, ki ga avtomobil le delno pokrije. Velik del slike predstavlja ozadje, ki ne prinaša nobene informacije o tem, kje so udrtine. Iz sistema dobimo masko, ki pove na katerem delu slike se pojavlja vzorec. Slike v učni množici obrežemo tako, da vzamemo minimalno površino, ki še vsebuje celotno masko in s tem zmanjšamo količino odvečne informacije.

**Uporaba paketov** Med postopkom učenja bi lahko v mrežo pošiljali po eno sliko na enkrat. Uteži bi se tako v enem koraku popravile zgolj glede na ta primer. Tako učenje je zelo nestabilno in vodi v počasno konvergenco. Zato uvedemo učne pakete. Po več učnih primerov združimo v paket in na njem zaženemo mrežo. Idealno bi bilo, če bi lahko v mrežo poslali paket z vsemi primeri naenkrat. Vendar omejitve pomnilnika in procesorske moči tega ne omogočajo. Druga težava je različna velikost slik v učni množici. Ko podatke združujemo v paket, morajo biti vse slike enako velike. Problem bi lahko rešili z obdajanem, vendar bi taki podatki vsebovali veliko praznine brez informacije. Iz teh razlogov smo se odločili za sistem z izrezovanjem. Na vsakem koraku zgradimo paket naključnih izrezov fiksne velikosti, ki jih izrežemo iz naključih učnih primerov. Naključno izrezovanje zagotavlja enakomerno porazdelitev pokritosti učnih primerov, hkrati pa poskrbimo, da je

paket maksimalno zapolnjen z informacijo. Velikost okna izrezovanja, ter velikost paketov bomo definirali v nadaljevanju, ko bomo opisali hiperparametre mreže.

**Razširjanje učne množice** Razširjanje učne množice (angl. data augmentation) nam omogoča, da s spreminjanjem primerov učne množice umetno generiramo nove prepričljive primere. S tem zmanjšamo problem prekomernega prilagajanja in povečamo robustnost mreže, še posebej pri majhnih učnih množicah.

Pri razširjanju učne množice slik lahko uporabimo številne geometrijske transformacije, barvna prirejanja ter druge metode. Izbrati moramo primerne metode, ki se zdijo smiselne za problemsko domeno. Na naših podatkih je na primer nesmiselna rotacija za 90 stopinj, saj se na nobeni sliki vzorec ne bo pojavil navpično namesto vodoravno. Če pa sliko rotiramo za 180 stopinj, vzorec ostane vodoraven. Dobimo prepričljiv primer udrtine, ki pa je za mrežo drugačen od originala. Konvolucijske nevronske mreže namreč v osnovi niso invariante na rotacije.

Preden sveženj pošljemo v mrežo gre skozi metodo prirejanja, ki naključno izvede naslednje operacije:

- naključna sprememba svetlosti slike (naključna vrednost med -0.1 in 0.1),
- naključna rotacija za 180 stopinj (da ali ne) in
- naključno vodoravno zrcaljenje (da ali ne)

## 4.2 Protokol evalvacije

### 4.2.1 Podatkovna zbirka

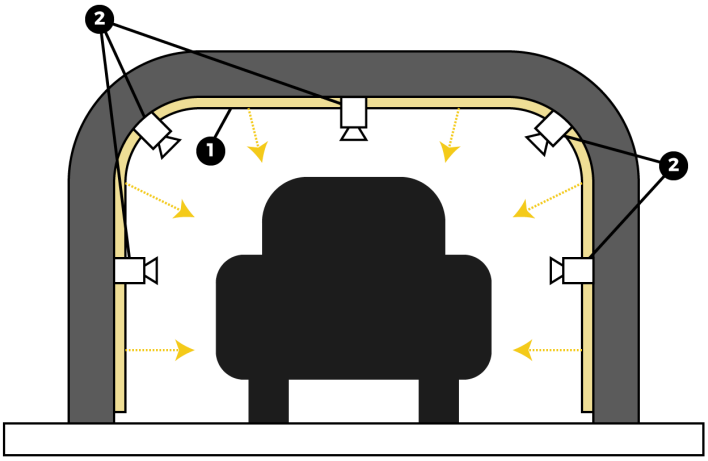
V tem poglavju bomo opisali podatkovno zbirko ki smo jo uporabili za učenje in evalviranje modela. Sprva pa opišemo sistem, ki smo ga uporabili za zajem slik.


Slika 4.1: Primer svetlobnega tunela podjetja AutoScan [2].

### 4.2.2 Zgradba sistema za zajem slik

Metode deflektometrije zahtevajo proces, pri katerem na površino projiciramo vnaprej določen svetlobni vzorec. Na podlagi deformacij, ki se zaradi odbojnih lastnosti pojavijo v vzorcu, lahko zaznamo prisotnost anomalije. Za analizo avtomobilov so razviti posebni svetlobni tuneli. Primer takega sistema je na Sliki 4.1.

Sistem je sestavljen iz dveh komponent: svetila, ki na površino avtomobila projicirajo vzorec in kamere, ki so razporejene po oboku tunela (glej Sliko 4.2). Avtomobil počasi zapeljemo skozi tunel. Svetila, razporejena po površini oboka, nanj projicirajo resast vzorec. Ta je sestavljen iz dveh naborov črt. Prvi vsebuje širše črte, namenjene detekciji večjih udrtin, drugi pa ožje, ki razkrijejo tudi manjše udrtine. Kamere, razporejene po oboku sistema, ves čas zajemajo slike vzorca na vozilu. Število in konfiguracija kamer se razlikuje od sistema do sistema. Sistem, ki smo ga uporabili za zajem slik učne in evaluacijske množice vsebuje 5 kamer, ki snemajo streho, oba stranska in oba poševna pogleda (glej Sliko 4.2). Sistem zajema črnobele slike velikosti $2048 \times 1088$.

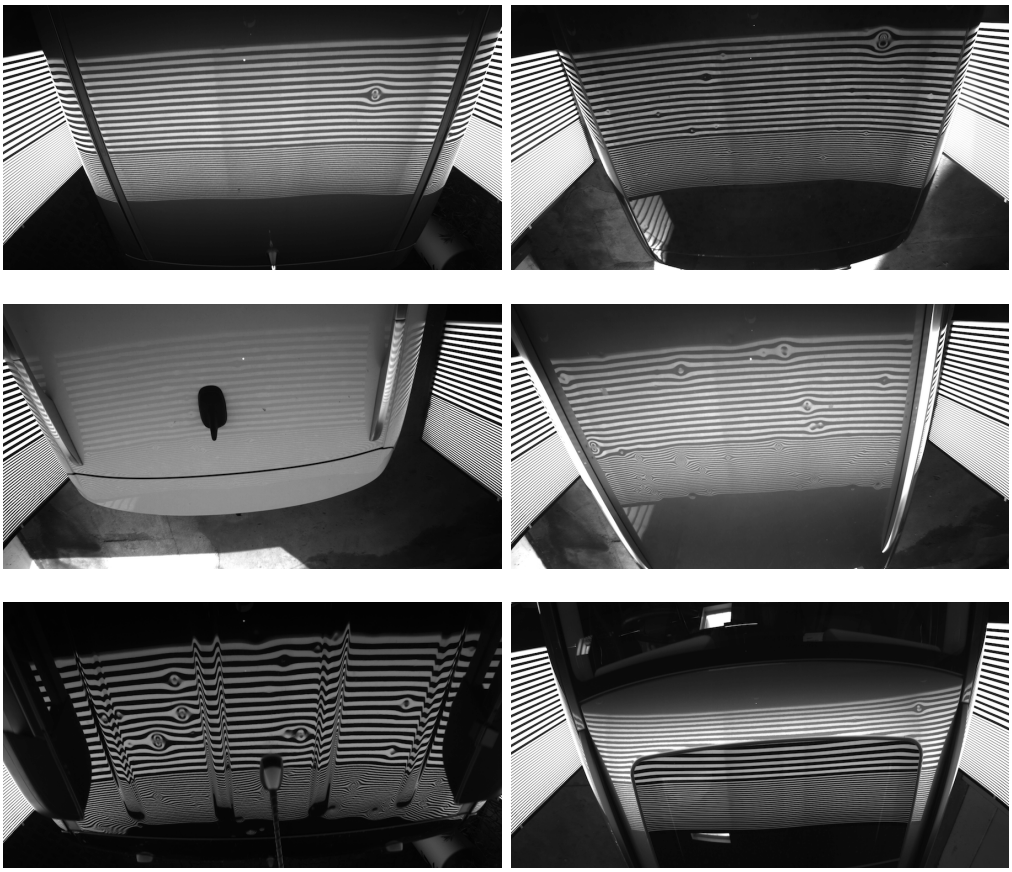



Slika 4.2: Shema sistema (1 - svetlobni obok, 2 - kamere).

### 4.2.3 Podatkovne množice

Za gradnjo učne množice smo zajeli slike na šestih avtomobilih. Za namen naloge smo za učenje in testiranje metode uporabljali zgolj slike zajete na strehi avtomobila. Zaradi velikega prekrivanja smo za vsak avtomobil izbrali 7 zajetih slik ter jih anotirali. Baza torej skupaj vsebuje 42 anotiranih slik. Anotacije posameznega primera so sestavljene iz koordiant centra udrtine ter njenega radija. Primere smo nato naključno razdelili med učno in validacijsko množico v razmerju 2:1. V eksperimentih smo mrežo učili na učni množici, evalvirali pa na validacijski. Na Sliki 4.3 je razvidna raznolikost avtomobilov v učni bazi.

Za evalvacijo končnega modela smo uporabili ločeno testno podatkovno zbirko, sestavljeno iz slik, zajetih na petih novih avtomobilih, ki se niso pojavili v učni in validacijski množici. Testna množica vsebuje 37 slik. Za končno evalvacijo smo model učili na združeni bazi sestavljeni iz učne in validacijske množice.



Slika 4.3: Primeri slik v učni množici.

### 4.2.4 Performančne mere

Prvi aspekt, ki smo ga upoštevali pri merjenju uspešnosti lokalizacije udrtin je lokacija detekcij. Glede na lokacijo lahko ugotovimo, katere detekcije so pravilne, katere napačne, katere pa smo izpustili. Imamo množico detekcij in množico anotacij. Da lahko detekcijo in anotacijo obravnavamo kot ujemajoči se par, mora biti razdalja med njunima centroma pod mejno vrednostjo $d_{\max}$. Pri evalvaciji smo za mejno vrednost uporabili razdaljo $d_{\max} = 20$, kar predstavlja približno dvakratno velikost povprečne udrtine v bazi. Glede na to ali ima posamezna detekcija par v anotacijah jo lahko razvrstimo v eno izmed treh množic. Detekcije, ki imajo par v anotacijah so pravilne detekcije (angl. true positive). Detekcije, ki nimajo para v anotacijah so napačne detekcije (angl. false positive), anotacije, ki ostanejo brez para pa so manjkajoče detekcije (angl. false negative). Vsaka detekcija ima lahko največ en par v anotacijah in obratno. Podvojenih detekcij ne štejemo za pravilne. Glede na število elementov v dobljenih množicah izračunamo metriki natančnost in priklic, ter F-mero, ki združuje obe. V nadaljevanju oznaka $TP$ označuje število pravilnih detekcij, $FP$ število napačnih detekcij, $FN$ pa število manjkajočih detekcij.

### 4.2.5 Natančnost

Natančnost (angl. precision) označuje delež pravilnih detekcij. Dobimo jo kot količnik števila pravilnih detekcij s številom vseh detekcij po formuli

$$\mathrm{Pr} = \frac{TP}{TP + FP}. \tag{4.1}$$

### 4.2.6 Priklic

Priklic (angl. recall) označuje delež detektiranih anotacij. Izračunamo jo kot količnik števila pravilnih detekcij s številom vseh anotacij po enačbi

$$\mathrm{Re} = \frac{TP}{TP + FN}. \tag{4.2}$$

Natančnost in priklic sta zelo povezana, zato moramo gledati oba. Model z visoko natančnostjo in nizkim priklicem sicer detektira malo anotiranih primerov, vendar se tudi redko zmoti. Model z visokim priklicem in nizko natančnostjo detektira večino anotiranih primerov, zraven pa vrine še veliko napačnih detekcij. Želimo si, da bi bili obe vrednosti čim višji.

### 4.2.7 F-mera

F-mera (angl. F-score) združuje metriki natančnosti in priklica. S tem dobimo samostojno vrednost za oceno uspešnosti modela, ki jo lahko uporabimo za lažjo primerjavo modelov. Dobimo jo po enačbi

$$\mathrm{F} = \frac{2}{\frac{1}{\mathrm{Pr}} + \frac{1}{\mathrm{Re}}}. \tag{4.3}$$

### 4.2.8 Napaka velikosti

Drugi aspekt, ki je pomemben pri ocenjevanju uspešnosti metode je velikost detekcij. Napaka velikosti nam pove, kako natančen je model pri napovedovanju velikosti udrtine. Izračunamo jo kot povprečno absolutno napako (angl. mean absolute error) med pari pravilnih detekcij in ujemajočih anotacij, po enačbi

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=0}^{n} \left| r_i - \hat{r}_i \right|, \tag{4.4}$$

kjer je $n$ število parov, $r_i, i = 1, ..., n$ predstavlja velikost anotacije, $\hat{r}_i, i = 1, ..., n$, pa velikost detekcije v paru $i$. Napaka je merjena v isti enoti kot so podane velikosti detekcij, to je v številu pikslov pri normalizirani širini vzorca.

### 4.2.9 Čas procesiranja

V realnih aplikacijah ni pomembna zgolj natančnost modela, ampak tudi hitrost procesiranja. Sistem na avtomobil zajame na stotine slik in metoda jih mora biti sposobna obdelati v primernem času. Pri evalvaciji modela

smo zato merili tudi povprečen čas izvajanja metode na sliko $\bar{t}$. Vsi časi so merjeni v sekundah.

### 4.2.10 Delež prekrivanja

Natančnost in priklic sta metriki, ki ju je smiselno meriti na detekcijah. Za primerjavo natančnosti segmentacije pa se uporabljajo drugačne metrike. V ta namen smo uporabili delež prekrivanja oz. presek z unijo (angl. intersection over union ali IoU), ki primerja napovedano segmentacijsko masko s pravilno segmentacijsko masko. Izračunamo ga kot razmerje med presekom obeh mask in unijo obeh mask. Da lahko računamo presk mask, moramo izhod mreže najprej binarizirati. To storimo z upragovanjem okoli vrednosti 0.5. Pri naši metodi delež prekrivanja ni glavna metrika, saj je segmentacija le vmesni člen do lokalizacije detekcij. Pri evalvaciji parametrov segmentacije smo zato merili tudi ostale metrike.

## 4.3 Primerjava načinov združevanja

V tem eksperimentu smo primerjali delovanje mreže pri uporabi dveh različnih načinov združevanja: združevanja s konvolucijo in združevanja z maksimumom. V ta namen smo trenirali in testirali tri različne mreže:

- **conv-pool-net**: mreža izvaja združevanje z operacijami konvolucije in uporabo koraka
- **max-pool-add-net**: mreža izhaja iz mreže *conv-pool-net*. Konvolucijske sloje združevanja smo pretvorili v navadne sloje (korak smo nastavili na 1) in dodali sloje združevanja z maksimumom.
- **max-pool-replace-net**: mreža izhaja iz mreže *conv-pool-net*. Konvolucijske sloje združevanja smo v celoti nadomestili s sloji združevanja z maksimumom.

| mreža | IoU | Pr | Re | F | MAE[pix] | $\bar{t}$[s] |
|---|---|---|---|---|---|---|
| **conv-pool-net** | **0.25** | 0.88 | **0.79** | 0.833 | **3.22** | **0.0216** |
| **max-pool-add-net** | 0.24 | **0.90** | 0.78 | **0.835** | 3.78 | 0.0328 |
| **max-pool-replace-net** | 0.24 | 0.87 | **0.79** | 0.828 | 3.23 | 0.0242 |

Tabela 4.2: Evalvacija modelov z različnimi sloji združevanja.

Rezultati evalvacije so podani v Tabeli 4.2. Iz tabele je razvidno, da ni bistvenih razlik med različnimi pristopi. Uporaba združevanja s konvolucijo daje podobne rezultate kot združevanje z maksimumom, tako po uspešnosti, kot po času izvajanja. Zaradi dodane prednosti sposobnosti prilagajanja združevanja na domeno, smo se odločili za uporabo združevanja s konvolucijo.

## 4.4 Vpliv števila slojev

V eksperimentu smo testirali vpliv števila slojev na natančnost in delovanje mreže. Arhitektura mreže je sestavljena iz več velikostnih nivojev. Velikostni nivo predstavljajo vsi sloji, ki izvajajo operacije na slikah enake velikosti. Sloji združevanja so torej mejniki velikostnih nivojev. V eksperimentu smo kot konstanto vzeli število slojev združevanja, spreminjali pa smo število slojev, ki se pojavijo na vsakem velikostnem nivoju. Razvili in testirali smo tri konfiguracije mreže:

**three-layer-net**: mreža na vsakem velikostnem nivoju vsebuje po tri konvolucijske sloje (vključno s slojem združevanja),

**two-layer-net**: mreža na vsakem velikostnem nivoju vsebuje po dva konvolucijske sloje (vključno s slojem združevanja),

**one-layer-net**: mrežo na vsakem velikostnem nivoju sestavlja le sloj združevanja s konvolucijo.

Rezultati evalvacije so v Tabeli 4.3. Po pričakovanjih so mreže z večjim številom slojev dajale boljše rezultate, čas izvajanja pa je bil daljši. Največja

| mreža | IoU | Pr | Re | F | MAE[pix] | $\bar{t}$[s] |
|---|---|---|---|---|---|---|
| **three-layer-net** | 0.28 | **0.98** | **0.71** | **0.823** | 4.12 | 0.0215 |
| **two-layer-net** | **0.29** | 0.89 | **0.71** | 0.790 | **3.88** | 0.0157 |
| **one-layer-net** | 0.27 | 0.81 | 0.68 | 0.739 | 5.03 | **0.0118** |

Tabela 4.3: Evalvacija modelov z različnim številom konvolucijskih slojev.

razlika med mrežami je v natančnosti. Tu mreža s po tremi sloji izrazito izstopa. Po preostalih performančnih merah so mreže bolj primerljive. Mreža s po dvema slojema je imela celo najmanjšo napako velikosti, kar lahko pripišemo najboljši segmentaciji (IoU). Kljub počasnejšemu času izvajanja smo se odločili za mrežo s tremi sloji na nivo, saj daje izrazito boljše rezultate od ostalih dveh mrež.

## 4.5 Vpliv števila filtrov

V tem eksperimentu smo spreminjali število filtrov po posameznih velikostnih nivojih. Testirane konfiguracije mrež so opisane v Tabeli 4.4. Izhajali smo iz porazdelitve, kjer se število filtrov na vsakem nivoju podvoji, testirali pa smo tudi bolj ploščato porazdelitev, kjer se število filtrov podvoji na vsakem drugem nivoju.

| mreža | 1. nivo | 2. nivo | 3. nivo | 4. nivo |
|---|---|---|---|---|
| **normal-filter-net** | 16 | 32 | 64 | 128 |
| **double-filter-net** | 32 | 64 | 128 | 256 |
| **half-filter-net** | 8 | 16 | 32 | 64 |
| **skew-filter-net** | 32 | 32 | 64 | 64 |
| **skew-filter-net-small** | 20 | 20 | 40 | 40 |

Tabela 4.4: Število filtrov na posameznem nivoju mreže.

Rezultati evalvacije so v Tabeli 4.5. Mreža z dvojnim številom filtrov (*double-filter-net*) rahlo izboljša natančnost detekcije in segmentacijo. Čas

| mreža | IoU | Pr | Re | F | MAE[pix] | $\bar{t}$[s] |
|---|---|---|---|---|---|---|
| **normal-filter-net** | 0.27 | 0.90 | 0.76 | 0.824 | 4.33 | 0.0216 |
| **double-filter-net** | **0.28** | 0.92 | 0.76 | 0.832 | 4.09 | 0.0376 |
| **half-filter-net** | 0.25 | **0.95** | 0.63 | 0.758 | 5.29 | **0.0144** |
| **skew-filter-net** | 0.27 | 0.91 | **0.81** | **0.857** | **3.40** | 0.0279 |
| **skew-filter-net-small** | 0.27 | **0.95** | 0.74 | 0.832 | 3.94 | 0.0221 |

Tabela 4.5: Evalvacija modelov z različno distribucijo števila konvolucijskih filtrov.

pa se skoraj podvoji. Mreža s polovičnim številom filtrov (*half-filter-net*) prepolovi čas izvajanja, vendar vidimo bistven upad priklica in natančnosti segmentacije. To lahko pripišemo zelo majhnemu številu filtrov na začetnih slojih (8), ki mrežo zelo omejijo. Ta problem rešujejo mreže z bolj ploščato razporeditvijo. Mreža *skew-filter-net* daje najboljše rezultate izmed vseh konfiguracij, vendar je tudi počasnejša. Njena manjša verzija vsebuje manjše število filtrov in ponuja dober kompromis med natančnostjo in hitrostjo. Ker nam je pomembna tudi hitrost, smo izbirali med navadno razporeditvijo in mrežo *skew-filter-net-small*. Obe namreč s praktično identičnim časom izvajanja dajeta primerljive rezultate. Odločili smo se za slednjo, saj ima boljšo F-mero, poleg tega pa se nam zdi porazdelitev filtrov bolj smiselna. Navadna razporeditev namreč na začetku vsebuje zelo majhno število filtrov, na zadnjih slojih pa več kot je potrebno za klasifikacijo enega razreda.

## 4.6 Evalvacija metode za lokalizacijo detekcij

Metodo lokalizacije detekcij smo evalvirali po naslednjem postopku. Vzeli smo anotacije vseh primerov v bazi, ter iz njih generirali segmentacijsko masko. Nad masko smo nato izvedli metodo za lokalizacijo detekcij, ter dobljene detekcije primerjali z originalnimi anotacijami.

Rezultati evalvacije so v Tabeli 4.6. Metoda je iz maske pravilno detektirala skoraj vse anotacije (96 %) ter ni detektirala niti ene napačne detekcije.

| | Pr | Re | F | MAE[pix] | $\bar{t}$[s] |
|---|---|---|---|---|---|
| **lokalizacija detekcij** | 1.00 | 0.96 | 0.980 | 0.50 | 0.0146 |

Tabela 4.6: Evalvacija metode za lokalizacijo detekcij.

Tudi napaka velikosti je zelo majhna (pol piksla). Z rezultati smo zadovoljni, saj bistveno ne omejujejo natančnosti celotne metode. Po pregledu segmentacijskih mask, se zdi višji priklic praktično nemogoč in bi pomenil nestabilno delovanje na šumnih detekcijah.

## 4.7 Evalvacija celotnega sistema

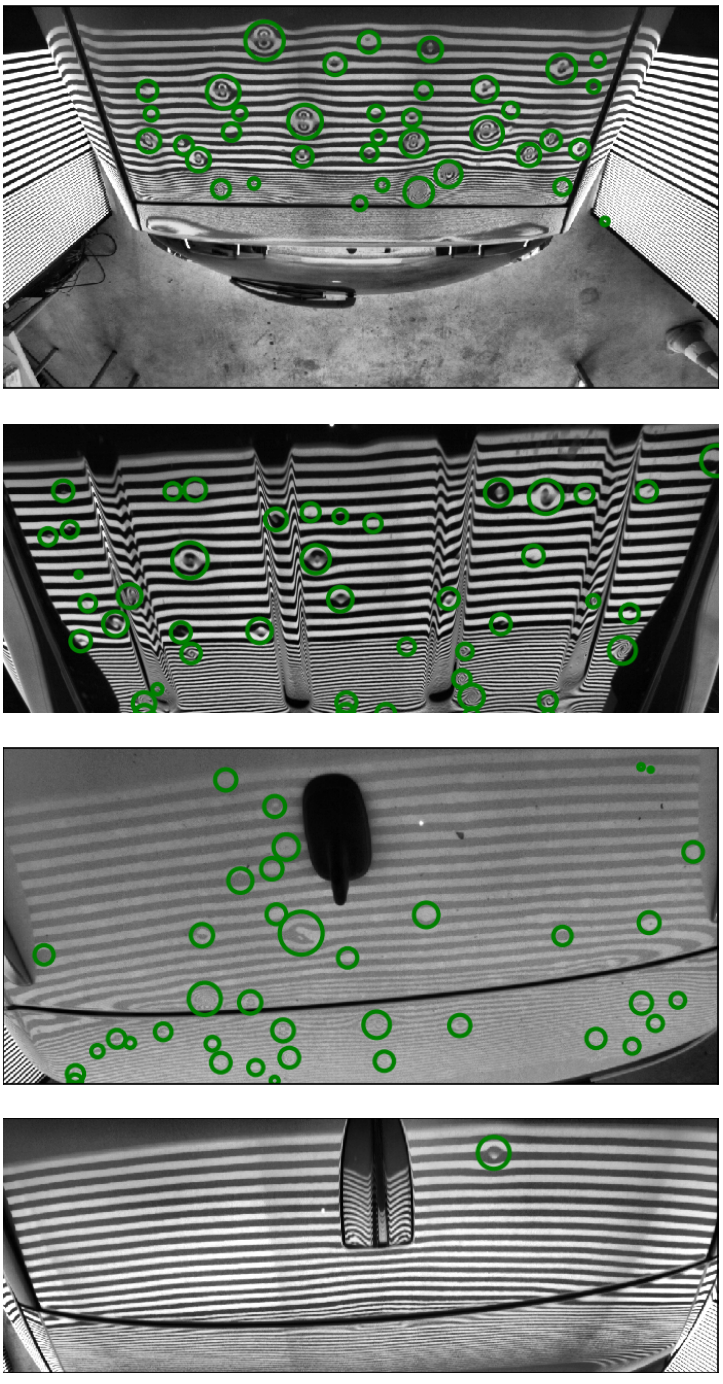
Končni model smo učili na opisani učni množici do konvergence. Za primerjavo smo uporabili komercialno rešitev, razvito v laboratoriju Vicos FRI, ki temelji na metodah deflektometrije. Metoda ni objavljena, deluje pa tako, da na sliki izvede detektor robov, s frekvenčno analizo detektira lokalna odstopanja v širini vzorca in te točke označi kot potencialne anomalije. Točke prostorsko grupira in jih dodatno preveri z razvrščevalnikom, ki temelji na gradientnem podpisu.

| metoda | Pr | Re | F | MAE[pix] | $\bar{t}$[s] |
|---|---|---|---|---|---|
| **naša metoda** | **0.88** | **0.88** | **0.880** | **2.73** | **0.1124** |
| **osnovna metoda** | 0.58 | 0.63 | 0.604 | 5.60 | $\sim 1.9$ |

Tabela 4.7: Evalvacija končne metode in primerjava z osnovno metodo.

Rezultati evalvacije, podani v Tabeli 4.7, kažejo, da sta natančnost in priklic še občutno narasla v primerjavi s preliminarno evalvacijo iz Poglavja 4.5. To lahko deloma pripišemo učenju na združeni bazi učnih primerov. Tudi napaka velikosti je v primerjavi s prejšnjimi evalvacijami bistveno nižja. Te obetavne izboljšave so dober znak, da je možno rezultate še izboljšati z gradnjo večje in natančnejše učne množice.

Na Sliki 4.4 je prikazanih nekaj primerov detekcij na različnih površinah.

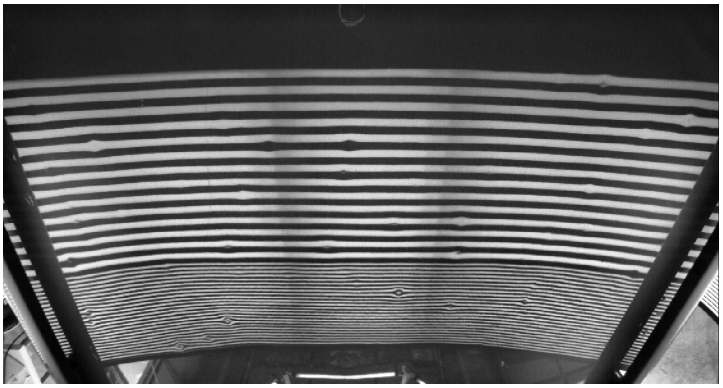


Slika 4.4: Primeri detekcij na različnih površinah.

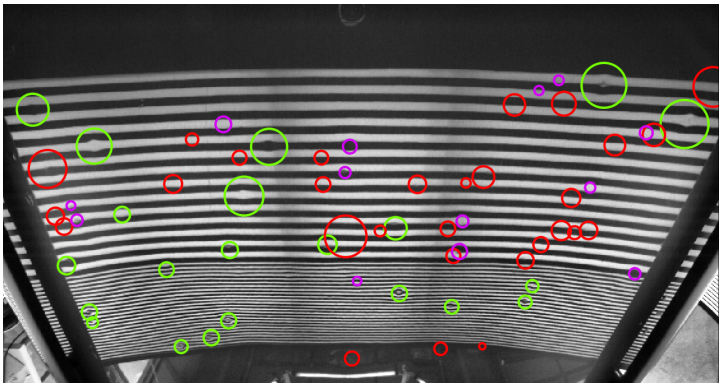
Razvidno je, da se razvita metoda dobro obnese pri različnih barvah odbojnih površin in celo pri zelo kompleksnih površinah, kjer klasične zaradi prevelikih nepravilnosti v vzorcu povsem odpovejo. V dodatnem gradivu[1] prilagamo posnetek, ki prikazuje delovanje metode na celotni sekvenci slik avtomobila, pridobljenih iz svetlobnega tunela.

Ko razvito metodo primerjamo z osnovno metodo vidimo, da je boljša v skoraj vseh aspektih. F-mera je narasla iz 0.604 na 0.880, kar predstavlja kar 45,7 % izboljšavo. Osnovna metoda je še posebej slaba pri lokalizaciji udrtin in njihovih velikosti (glej Sliko 4.5). Detekcije se pogosto pojavljajo precej oddaljene od dejanske udrtine. Rezultati naše metode delujejo veliko bolj smiselni. Poleg tega se pojavi nekaj napačnih detekcij, ki so v resnici pravilne, a niso anotirane v testni množici (glej Sliko 4.6). Tudi čas izvajanja je pri naši metodi bistveno (skoraj 17x) krajši. Pri tem je potrebno poudariti, da je čas izvajanja osnovne metode odvisen od števila udrtin, medtem ko naša metoda nima te omejitve. Razvita metoda slike procesira s povprečno hitrostjo 8.9 slik na sekundo.

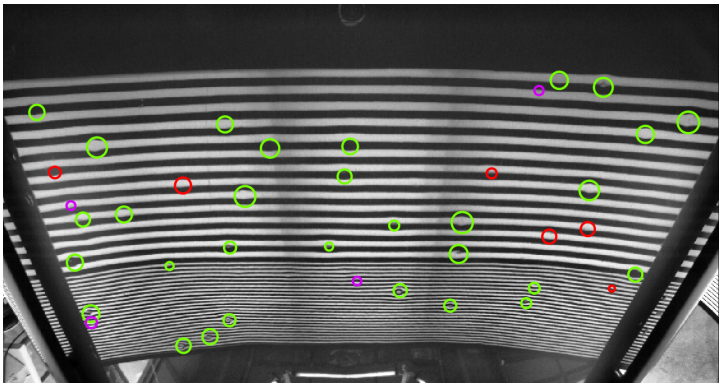
[1]Dodatno gradivo se nahaja na priloženi zgoščenki v direktoriju Dodatno gradivo



(a) Vhodna slika.

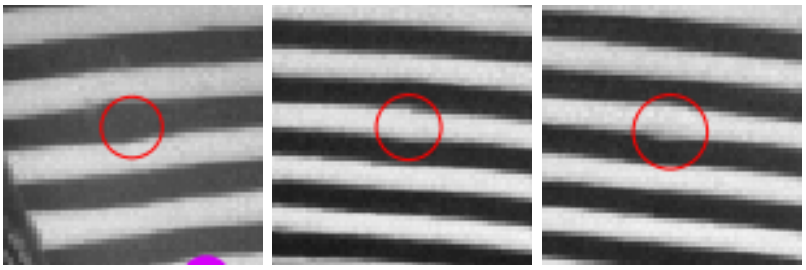


(b) Detekcije osnovne metode.



(c) Detekcije naše metode.

Slika 4.5: Primerjava detekcij osnovne metode z detekcijami naše metode na primeru. Z zeleno barvo so označene pravilne (TP), z rdečo napačne (FP), z vijolično pa manjkajoče (FN) detekcije.



Slika 4.6: Primeri napačnih detekcij, ki so v resnici pravilne.

# Poglavje 5

# Sklep

Predlagamo novo metodo za detekcijo anomalij na reflektivnih površinah s pomočjo deflektometrije. Predlagana metoda se bistveno razlikuje od obstoječih v tem, da je sposobna hitre detekcije z uporabo zgolj ene slike površine in je učljiva, kar pomeni, da ne potrebuje referenčnih modelov objektov, ki jih opazuje. S tem se bistveno poveča njena aplikativna vrednost.

Metoda postavi detekcijo anomalij kot problem semantične segmentacije, ki za vsak slikovni element napove pripadnost anomaliji. Rezultat je segmentacijska maska. Predlagali smo tudi postopek interpretacije segmentacijske maske, ki je sposoben detektirati položaj in velikost okroglih anomalij, četudi se anomalije v veliki meri prekrivajo.

Za postopek semantične segmentacije smo izbrali konvolucijsko nevronsko mrežo. S preliminarno analizo in eksperimenti smo utemeljili izbiro ustrezne arhitekture in hiperparametrov. Mrežo smo učili in evalvirali na problemu detekcije udrtin v strehi avtomobila, kjer smo demonstrirali bistvene izboljšave v primerjavi z osnovno metodo deflektometrije. Razvita metoda dosega natančnost 0.88, priklic 0.88 in F-mero 0.88, kar predstavlja skoraj 50% izboljšavo v primerjavi z osnovno metodo, poleg tega pa je kar 17x hitrejša.

## 5.1 Nadaljnje delo

Tekom razvoja metode so se pojavile ideje za možne izboljšave. Po pregledu rezultatov je postalo očitno, da so anotacije v podatkovni zbirki ponekod nenatančne. Poleg tega je učna množica, ki smo jo uporabljali, dokaj majhna za konvolucijske nevronske mreže. Smiselna bi bila gradnja večje učne množice in natančnejša anotacija primerov. Na nekaterih mestih iz slike ni jasno razvidno ali gre za udrtino ali za umazanijo na avtomobilu. V takih primerih bi bilo smiselno uvesti sistem, ki omogoča anotiranje z ocenjevanjem gotovosti pravilnosti anotacije. S pravilnimi anotacijami bi poskrbeli, da mreže ne kaznujemo za pravilne detekcije.

Druga smiselna izboljšava je upoštevanje večjega števila zaporednih slik pri detekcijah. Sistem zajema slike v gostih intervalih. Glede na to, na kateri del vzorca padejo, so na nekaterih slikah udrtine bolj izrazite kot na drugih. Naša metoda anomalije detektira na eni sami sliki, brez konteksta drugih slik. Uporaba sistema sledenja in potrjevanja detekcij, bi lahko zmanjšala število napačnih detekcij. Obetavna in vredna nadaljnjega raziskovanja pa se zdi tudi ideja spremenjene arhitekture mreže, ki na vhodu sprejme dve zaporedni sliki in se iz te informacije uči pridobivanja detekcij.

# Literatura


[1] Interpolation. `https://en.wikipedia.org/wiki/Interpolation`. Zadnji dostop: 5. 6. 2018.

[2] Spletna stran podjetja autoscan. `http://www.auto-scan.eu/`. Zadnji dostop: 5. 6. 2018.

[3] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[4] B. Denkena, H. Ahlers, F. Berg, Th. Wolf, and H.K. Tönshoff. Fast inspection of larger sized curved surfaces by stripe projection. *CIRP Annals*, 51(1):499 – 502, 2002.

[5] Alberto Garcia-Garcia, Sergio Orts-Escolano, Sergiu Oprea, Victor Villena-Martinez, and José García Rodríguez. A review on deep learning techniques applied to semantic segmentation. *CoRR*, abs/1704.06857, 2017.

[6] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2nd edition, 1992.

[7] Hans J. Tiziani C. Voland Hansjoerg Gaertner, Peter Lehle. Structured light measurement by double-scan technique, 1996.

[8] S. Hershey, S. Chaudhuri, D. P. W. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, M. Slaney, R. J. Weiss, and K. Wilson. Cnn architectures for large-scale audio classification. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 131–135, March 2017.

[9] Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent.

[10] Denis Kolednik in Borut Žalik. Zaznava vdolbin na površjih z deflektometrijo in strojnim učenjem. In *Zbornik triindvajsete mednarodne Elektrotehniške in računalniške konference*, September 2014.

[11] O. Kafri and A. Livnat. Reflective surface analysis using moiré deflectometry. *Appl. Opt.*, 20(18):3098–3100, Sep 1981.

[12] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[13] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[14] T L I Sugata and C K Yang. Leaf app: Leaf recognition with deep convolutional neural networks. 273:012004, 11 2017.

[15] T. Lilienblum, P. Albrecht, R. Calow, and B. Michaelis. Dent detection in car bodies. In *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*, volume 4, pages 775–778 vol.4, 2000.

[16] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[17] Malgorzata Kujawinska Maria Pirga. Adaptive automatic shape measurement system, 1996.

[18] Gerd Hausler Markus C. Knauer, Jurgen Kaminski. Phase measuring deflectometry: a new approach to measure specular free-form surfaces, 2004.

[19] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pages 807–814, USA, 2010. Omnipress.

[20] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. *CoRR*, abs/1505.04366, 2015.

[21] I. Glatt O. Kafri. Moire deflectometry: A ray deflection approach to optical testing. *Optical Engineering*, 24:24 – 24 – 17, 1985.

[22] Augustus Odena, Vincent Dumoulin, and Chris Olah. Deconvolution and checkerboard artifacts. *Distill*, 2016.

[23] Stephen M. Pizer, E. Philip Amburn, John D. Austin, Robert Cromartie, Ari Geselowitz, Trey Greer, Bart ter Haar Romeny, John B. Zimmerman, and Karel Zuiderveld. Adaptive histogram equalization and its variations. *Computer Vision, Graphics, and Image Processing*, 39(3):355 – 368, 1987.

[24] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *CoRR*, abs/1511.06434, 2015.

[25] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[26] Milan Sonka, Vaclav Hlavac, and Roger Boyle. *Image processing, analysis, and machine vision*. Cengage Learning, 2014.

[27] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin A. Riedmiller. Striving for simplicity: The all convolutional net. *CoRR*, abs/1412.6806, 2014.

[28] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015.