Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Peter Mlakar

Siamska nevronska mreža za detekcijo gibanja v video sekvencah

diplomsko delo

univerzitetni študijski program
prve stopnje
računalništvo in informatika

Mentor: izr. prof. dr. Matej Kristan

Ljubljana, 2018

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

V diplomski nalogi obdelajte problem detekcije premikajočih se objektov v videoposnetkih. Osredotočite se na posnetke značilne za video nadzorne sisteme. Izdelajte metodo za detekcijo razlik na nivoju slikovnih elementov med trenutno in referenčno sliko. Kot metodološki okvir izberite konvolucijske nevronske mreže. Metodo analizirajte na prosto dostopnih podatkovnih zbirkah in jo primerjajte s sorodnimi pristopi.

Zahvaljujem se mentorju izr. prof. dr. Mateju Kristanu za usmerjanje ter strokovno vodenje pri izdelavi diplomske naloge. Prav tako se zahvaljujem raziskovalcu Domnu Taberniku, članu laboratorija Vicos FRI, za nasvete in tehnično pomoč. Zahvaljujem se tudi moji družini za podporo in razumevanje, še posebej sestri ter punci.

Kazalo

Povzetek

Abstract

1 Uvod **1**

1.1 Pregled področja 2

1.2 Prispevki 5

1.3 Struktura diplomskega dela 6

2 Umetne nevronske mreže **7**

2.1 Principi umetnih nevronskih mrež 8

2.2 Konvolucijske nevronske mreže 15

3 Detekcija sprememb in gibanja v video sekvencah **21**

3.1 Generiranje ozadja video sekvence 21

3.2 Siamska konvolucijska nevronska mreža 23

4 Eksperimentalna evalvacija **29**

4.1 Programsko in strojno okolje 29

4.2 Postopek učenja 31

4.3 Protokol evalvacije 41

4.4 Analiza rezultatov 45

5 Sklep **61**

5.1 Možnosti izboljšav 62

Povzetek

Naslov: Siamska nevronska mreža za detekcijo gibanja v video sekvencah

Avtor: Peter Mlakar

V diplomski nalogi obravnavamo problem avtomatske detekcije gibanja v video sekvencah posnetih z video nadzornimi sistemi. Trenutno najuspešnejše metode uporabljajo konvolucijske nevronske mreže za reševanje tega problema. Bistvena omejitev teh pristopov je v tem, da potrebujejo ponovno učenje za različne video sekvence, kar zmanjša njihovo aplikativno vrednost. V diplomskem delu predstavimo novo metodo, ki temelji na arhitekturi siamskih konvolucijskih mrež. Mreža s pomočjo siamske arhitekture semantično opiše vhodno sliko sekvence ter model ozadja sekvence. Nadaljnji konvolucijski nivoji detektirajo relevantne razlike ter generirajo verjetnostno masko segmentacije gibanja. Z metodo lahko detekcijo gibanja izvajamo na različnih video sekvencah brez ponovnega učenja. Za izvajanje potrebujemo le referenčno sliko ozadja sekvence, ki jo nato tekom časa samodejno posodablja. Mrežo smo učili na podatkovni zbirki CDNET. Pridobljene rezultate smo primerjali s preostalimi metodami, objavljenimi na spletni strani CDNET. Naša metoda se je po uspešnosti uvrstila na osmo mesto izmed 46 objavljenih algoritmov. Mrežo smo ocenili tudi na evalvacijskih zbirkah Wallflower ter SGM-RGBD, kjer smo jo preizkusili v različnih okoliščinah ter podali kvalitativno analizo njenega delovanje.

Ključne besede: računalniški vid, siamske konvolucijske nevronske mreže, detekcija gibanja, video nadzorni sistemi.

Abstract

Title: Siamese neural network for motion detection in video sequences

Author: Peter Mlakar

We examine the problem of automatic motion detection in video sequences captured by video surveillance systems. The state of the art methods use convolutional neural networks. Their main limitation is that they need to be retrained if they are to be applied on different sequences. In our thesis, we present a novel method which is based on the architecture of siamese convolutional neural networks. Our network semantically describes the input image from the sequence and the model of the background of the sequence. It does this by using the siamese architecture. It then applies convolutional layers to detect relevant differences and generates the final probability segmentation mask. Our approach allows detection on different video sequences without retraining the network on each new sequence. To detect motion only a reference background images is required. The method automatically updates the background image during application. We trained our network on the CDNET data set. We compared our method with the other methods published on the CDNET website. It ranked as the eight best method of the 46 published methods. We also evaluated our method on the Wallflower and SGM-RGBD data sets. There, we tested it in different circumstances and provided qualitative analysis of its performance.

Keywords: computer vision, siamese convolutional neural networks, motion detection, video surveillance systems.

Poglavje 1

Uvod

Detektiranje razlik med ozadjem in ospredjem ter tako prepoznavanje gibanja in sprememb v video sekvencah je problem, ki postaja vse bolj prevalenten v času, ko se vsak dan zberejo velike količine podatkov. Ena izmed domen v kateri se zbirajo velike količine podatkov, so video nadzorni sistemi. Video sekvence avtocestnih območji, nakupovalnih središč ali drugih okolji so raznolike ter pogosto zelo obsežne. V nekaterih je lahko detekcija gibanja ter sprememb ključnega pomena. Ob nadzoru avtocestnega odseka nas zanimajo premikajoča se prevozna sredstva. V nakupovalnih središčih ponoči spremljamo nepričakovano gibanje. Ker so lahko zajete video sekvence dolge, je v mnogih primerih ročna detekcija gibanja dolgotrajen proces. Proces detekcije gibanja v video sekvencah lahko prevedemo na problem detekcije razlike med modelom ozadja sekvence ter posamezne slike iz sekvence. Tako se je tekom časa razvilo mnogo algoritmov, ki samodejno detektirajo te razlike. Ti algoritmi imajo splošno aplikativno vrednost na mnogih področjih. Področje video nadzornih sistemov ni edino, kjer takšni algoritmi pridejo prav. Operacija detektiranja ospredja v dani video sekvenci je pogosto prvi korak za nadaljnje procesiranje. Primer nekaterih drugih aplikacij teh algoritmov so štetje ljudi, detekcija in prepoznavanje dogodkov v video sekvencah ter prepoznavanje objektov.

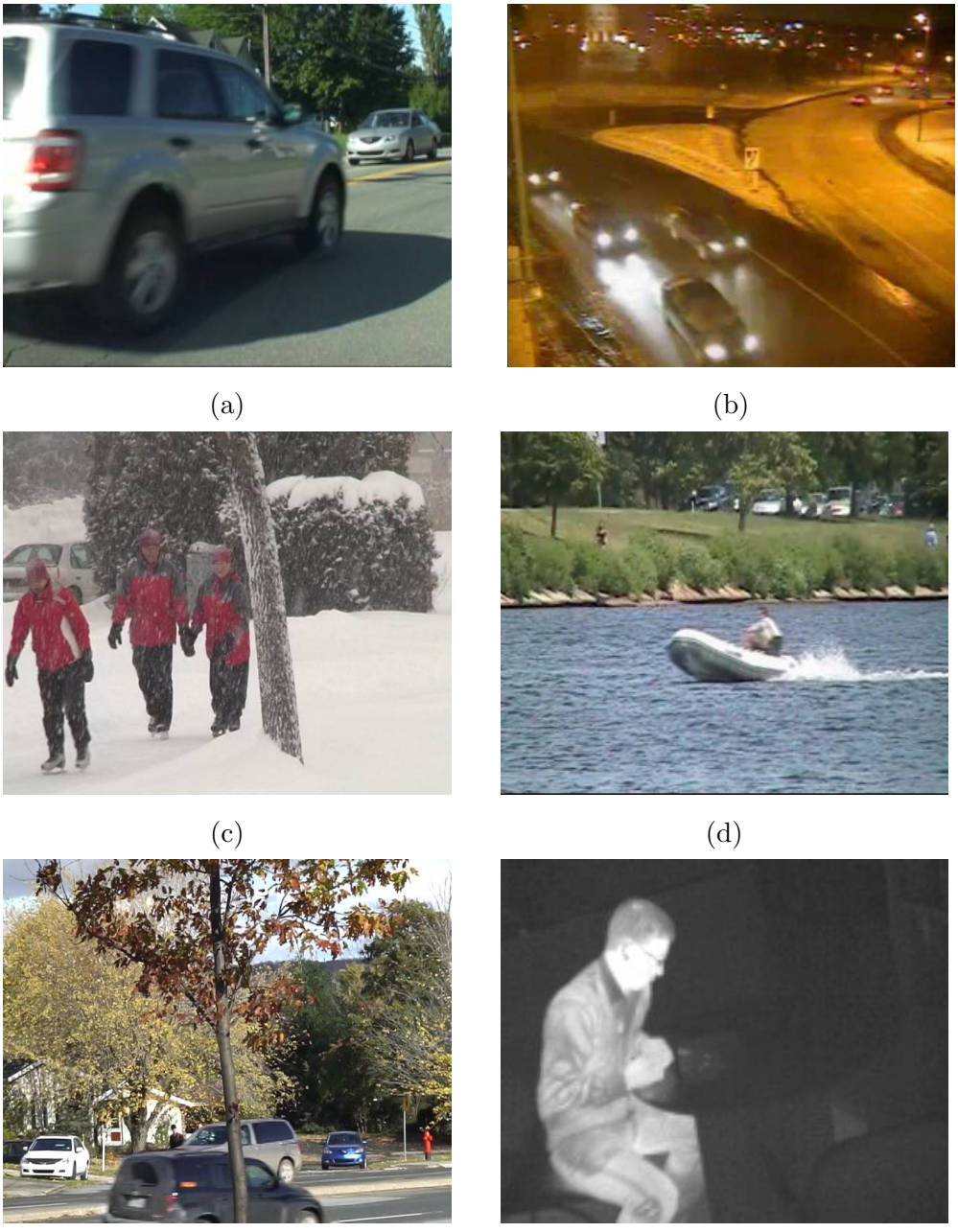
Detekcija gibanja in sprememb predstavlja zahteven problem. Velika va-

riabilnost okolij, v katerih so video sekvence posnete, zahteva od algoritma zmožnost delovanja pod različnimi pogoji. V okolju se lahko pojavljajo dinamični elementi, ki niso nujno del ospredja. Premikajoča se drevesa v vetru, gladina vode, različne padavine, spremembe v osvetlitvi in mnogi drugi pojavi problem detekcije relevantnega gibanja in sprememb dodatno otežijo. Problematični so tudi dogodki, ko je objekt delno zakrit za drugim, ki ni del ospredja. Pogosto težavo povzročajo tudi kamere, ki snemajo okolico. Zaradi namestitve se lahko rahlo premikajo ali tresejo. Slabše video nadzorne kamere v posnetke dodajajo neželeni šum. Na Sliki 1.1 predstavimo nekatera težavna področja detekcije gibanja.

1.1 Pregled področja

Metode za reševanje problema detekcije gibanja ter sprememb v video sekvencah, se pojavljajo že več desetletji [46]. Ta problem lahko rešujemo z generiranjem matematičnega modela ozadja video sekvence. S pomočjo tega lahko na posameznih slikah sekvence razločimo objekte, ki niso del tega ozadja. Metod za modeliranje ozadja je mnogo ter temeljijo na različnih principih modeliranja. Nekateri modeli posamezno slikovno točko v ozadju aproksimirajo z Gaussovo porazdelitvijo [45]. Nadaljni razvoj je poizkušal izboljšati pomankljivosti tega modela, ena od teh je težavnost modeliranja dinamičnega ozadja, zaradi počasnega prilagajanja Gaussove porazdelitve na hitre spremembe [31]. Tako so nastali modeli, ki za posamezne slikovne točke koristijo več Gaussovih distribucij [39] ali modeli, ki izboljšajo učinkovitost posodabljanja parametrov v modelu [49]. Glavna težava teh modelov je predvsem njihova dovzetnost za šum, modeliranje dinamičnega ozadja ter spremembe v osvetlitvi.

Namesto posameznih slikovnih točk lahko modeliramo regije ozadja, kjer izrabimo razmerja med slikovnimi točkami [35]. S tem je izboljšana odpornost algoritma na šum v sliki, a v zameno lahko tak algoritem producira le grobe obrise predmetov v ospredju. Hibridne mešanice obeh pristopov pa



(a) (b)

(c) (d)

(e) (f)

Slika 1.1: Prikaz različnih problematičnih situacij: vsebnost močnih senc (a); posnetek v slabi vidljivosti (b); snežne padavine (c); morska gladina je del dinamičnega ozadja (d); avtomobil je delno zakrit za drevesom (e); posnetki narejeni s termalno kamero (f).

še nadaljno izboljšajo model ozadja, kjer algoritmi uporabijo tako informacije na nivoju slikovnih točk in večjih regij [30]. Njihova slabost je relativno visoka računska kompleksnost [46].

V zadnjih letih se pojavljajo metode, ki uporabljajo konvolucijske nevronske mreže za reševanje teh problemov in močno izboljšujejo rezultate drugih, konvencionalnih metod [44] [14] [9] [26]. ChangeDetection.NET ali CDNET [43], je podatkovna zbirka, ki vsebuje številne video sekvence namenjene testiranju algoritmov, ki v posnetkih detektirajo gibanje in spremembe. Poleg podatkovne zbirke njeni avtorji na njihovi spletni strani prilagajo kriterije za ocenjevanje delovanja posameznih algoritmov ter omogočajo objavo le teh. Prejete algoritme nato ovrednotijo ter njihove rezultate javno objavijo [17]. V nadaljevanju podajamo pregled najuspešnejših objavljenih ter ovrednotenih metod na portalu CDNET, ki v svoji arhitekturi uporabljajo konvolucijske nevronske mreže.

CascadeCNN [44] je metoda, ki uporablja kombinacijo dveh konvolucijskih nevronskih mrež za izdelavo binarne maske ospredja. Izhod prve konvolucijske nevronske mreže je groba maska ospredja. Pridobljeno binarno masko nato združi s sliko, ki je služila kot vhod v prvo mrežo, ter ju pošlje na vhod druge konvolucijske nevronske mreže. Rezultat je maska ospredja, ki je bolj natančna kot tista, ki je bila generirana s prvo mrežo. Namen mreže je, da se uporablja na isti video sekvenci, kot je bila učena. Za uporabo na drugi video sekvenci z drugačnim ozadjem potrebuje ponovno učenje.

Algoritem SematicBGS [14] izboljšuje delovanje konvencionalnih metod odštevanja ozadja z dodajanjem semantike. V sistem vnesejo semantiko z uporabo semantične segmentacijske mreže. Za generiranje končne izhodne maske ospredja uporabljajo verjetnostne modele ozadja in ospredja, ki jih pridobijo s semantično segmentacijo. S predpostavko, da slikovne točke, za katere segmentacija poda nizko verjetnost pripadnosti kateremu od razredov, ki jih mreža segmentira, pripadajo ozadju. Z dodatnimi pravili nato določijo predmete ospredja, kjer rešujejo problem, ko segmentiran objekt pripada ozadju. Konvencionalni algoritem za detekcijo ozadja se uporabi le

v primeru, ko se segmentirani predmet iz ospredja premakne za segmentirani predmet v ozadju. Algoritem je lahko uporabljen na poljubnih video sekvencah, saj razen segmentacijske mreže, ki je učena posebej, ne potrebuje učenja za specifični problem detekcije sprememb.

Mreža *DeepBS* [9] uporablja konvolucijsko nevronsko mrežo za odštevanje ozadja posnete sekvence od posamezne slike v sekvenci. Model ozadja je pripravljen s pomočjo algoritma subSENSE [38], ki generira masko ospredja v sliki na podlagi prostorsko-časovnih binarnih komponent [11] ter barve. Pridobljeno ozadje je nato skupaj s sliko iz video sekvence vstavljeno v konvolucijsko nevronsko mrežo, ki generira končni rezultat. Ta je še dodatno izboljšan s filtrom mediane za popravek manjših nepravilnosti. Prednost tega postopka je, da delovanje ni omejeno samo na sekvenco, na katerih je bil algoritem učen.

Rešitev *FgSegNet* [26] uporablja konvolucijsko nevronsko mrežo, ki se nauči segmentacije ospredja s primerom slike v treh različnih velikostih. Za segmentacijo teh slik uporablja enake parametre. Na vsaki sliki poišče iste vzorce, samo v drugačnih velikostih. Arhitektura konvolucijske mreže, imenovana kodirno-dekodirna struktura, si kodirne nivoje izposodi iz prednaučene mreže *VGG-16* [37]. Podobno kot pri mreži *CascadeCNN* [44] je namenjena uporabi na video sekvenci, na kateri je bila učena.

1.2 Prispevki

Glavni prispevek diplomskega dela, je enostavni algoritem, ki temelji na arhitekturi siamske konvolucijske nevronske mreže. Naloga mreže je uspešna detekcija relevantnega gibanja ter sprememb v video sekvencah. Algoritem je preprost ter je lahko uporabljen na različnih video sekvencah, brez ponovnega učenja nevronske mreže. Metoda, ki jo predstavimo v tej diplomski nalogi, je sposobna dobre detekcije gibanja kljub prisotnosti kompleksnih okoliščin v video sekvencah, ki detekcijo otežujejo.

1.3 Struktura diplomskega dela

Diplomska naloga je sestavljena iz petih poglavji. V Poglavju 2 predstavimo teoretično podlago nevronskih mrež. Opišemo njihovo tipično sestavo ter pregledamo postopek učenja nevronskih mrež. Nato opišemo konvolucijske nevronske mreže, razložimo zakaj se razlikujejo od običajnih umetnih nevronskih mrež ter njihove tipične gradnike. V Poglavju 3 predstavimo naš pristop k reševanju problema detekcije sprememb in gibanja v video sekvencah. Argumentiramo izbiro posameznih komponent v algoritmu ter opišemo klasifikacijsko mrežo VGG-16 [37], ki je bila uporabljena pri razvoju. Začetek Poglavja 4 je namenjen predstavitvi programske opreme MatConvNet [41], s pomočjo katere smo razvili naš algoritem. Opišemo tudi učne zbirke video sekvenc, ki so bile uporabljene za učenje ter evalvacijo nevronske mreže. Opišemo postopek učenja ter prikažemo pridobljene rezultate in jih ovrednotimo. Zadnje, Poglavje 5, je namenjeno povzetku razvite metode ter glavnih ugotovitev. Predstavimo tudi nekatere možnostim nadaljnje izboljšave metode.

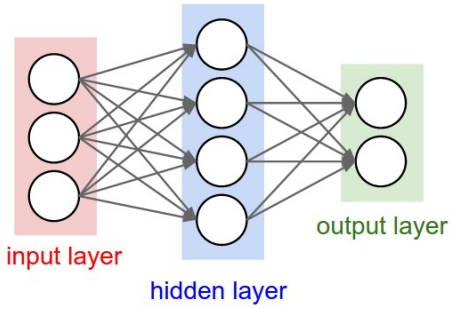
Poglavje 2

Umetne nevronske mreže

Umetne nevronske mreže (angl. *artificial neural networks*) [12] so kompleksni računski modeli, ki svojo uporabnost kažejo na mnogih področjih strojnega učenja. Navdih za umetne nevronske mreže izhaja iz bioloških nevronskih mrež v živalskih možganih, katerih efektivnost se kaže v nas samih. Uporabnost umetnih nevronskih mrež izhaja iz dejstva, da se med postopkom učenja same naučijo značilne opisnike, ki pripomorejo k reševanju problema, brez ekspertnega znanja človeka, ki bi v nasprotnem primeru te opisnike razvil sam. Koncept umetnih nevronskih mrež izvira iz sredine dvajsetega stoletja in je v času svojega obstoja pridobival ter izgubljal popularnost [13]. Glavni omejitvi te metode, ki sta ji v preteklosti preprečevali hitrejši razvoj, sta bili predvsem pomanjkanje močne strojne opreme ter neobstoj obsežnih javnih podatkovnih baz z labeliranimi učnimi primeri. V zadnjem desetletju smo priča velikem porastku dostopnih grafičnih procesorskih enot, ki služijo kot odlični paralelni procesorji, ter hitri rasti interneta, kjer se vsak dan zbere vratolomne količine podatkov [32]. S temi pridobitvami je področje umetnih nevronskih mrež postalo žarišče razvoja in raziskovanja na področju umetne inteligence. Tako so postale nepogrešliv del mnogih komercialnih aplikacij, ki jih uporabljajo Google ter druga podjetja [20]. V našem delu uporabljamo konvolucijsko nevronsko mrežo [22], ki je poseben tip umetne nevronske mreže.

2.1 Principi umetnih nevronskih mrež

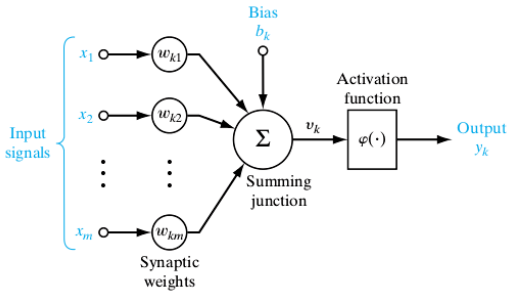
Umetne nevronske mreže so računski modeli, ki za reševanje določenega problema ne potrebujejo ekspertnega znanja o nalogi, saj sami to znanje pridobijo tekom učenja. Med postopkom učenja so izpostavljene atributom x, ki preko preslikave $f(x)$ tvorijo želene rešitve $f(x) = y$. Tako mrežo učimo s pari vhodnih atributov x ter primeri rešitev y glede na te atribute. Naloga umetne nevronske mreže je aproksimacija funkcije $f(x)$ s funkcijo $g(x, \alpha)$. Tekom učenja se mreža poskuša naučiti vrednosti parametrov α, s katerimi bi najbolje opisala želeno, a mreži neznano, funkcijo $f(x)$. Umetno nevronsko mrežo si lahko predstavljamo kot acikličen računski graf, kjer podatki tečejo po povezavah med vozlišči. Tem vozliščem pravimo *nevroni*. Nevroni so urejeni v nivoje. Nivojem, ki se nahajajo med vhodnim in izhodnim nivojem, pravimo *skriti nivoji*.



Slika 2.1: Primer enostavne nevronske mreže, povzeto po [1].

Primer enostavne nevronske mreže vidimo na Sliki 2.1. Podatki v nevronski mreži tečejo preko povezav med nevroni od vhodov do izhodov. Vsaka vhodna povezava v nevron je utežena z utežjo w_n, ki označuje moč povezave. Večja kot je moč, bolj pomembna je povezava in obratno. Nevron nad prejetimi vhodi izračuna obteženo vsoto glede na moči povezav, po katerih so podatki prispeli

$$y = \sum_{n=0}^{N} w_n x_n + b. \tag{2.1}$$



Slika 2.2: Primer nevrona v nevronski mreži, povzeto po [2].

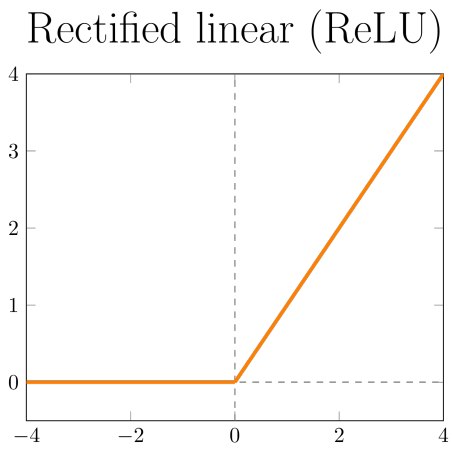
Vrednost n predstavlja indeks ene izmed N povezav, ki tečejo v nevron. V enačbi 2.1 vrednost w_n označuje moč povezave n in x_n izhodno vrednost nevrona, ki po tej povezavi pošilja podatke. Vsoti $\sum_{n=0}^{N} w_n x_n$ se nato prišteje še dodatna utež b nevrona samega. Ker je utežena vsota linearna operacija potrebujemo še nelinearnost v mreži. Tako nevronska mreža pridobi veliko boljšo sposobnost aproksimacije funkcij. To dosežemo tako, da nevronu dodamo *aktivacijsko funkcijo*. To je nelinearna odvedljiva funkcija, ki uteženo vsoto y preslika na določen interval. Primer takega intervala je $[0, y]$. Nelinearnih aktivacijskih funkcij je veliko, ena izmed glavnih lastnosti, ki jih morajo imeti vse, je odvedljivost. Ta lastnost je pomembna pri učenju umetne nevronske mreže. Najbolj pogosto uporabljena aktivacijska funkcija je *rektificirana linearna enota* ali *ReLU*. ReLU je funkcija, ki sledi predpisu

$$f(x) = max(0, x). \tag{2.2}$$

Na Sliki 2.3 opazimo, da je ReLU nelinearna funkcija, sestavljena iz dveh linearnih delov, ki se stikata v točki $x = 0$. Omenili smo, da je pomembna lastnost aktivacijskih funkcij njihova odvedljivost. Tako za ReLU velja, da je odvod določen po predpisu

$$\frac{df}{dx} = \begin{cases} 1, & \text{for } x > 0 \\ 0, & \text{for } x < 0 \end{cases}. \tag{2.3}$$

Odvod ni definiran v točki $x = 0$, zato je vrednost tega odvisna od posamezne implementacije. ReLU razvijalcu umetne nevronske mreže nudi hiter



Slika 2.3: Graf funkcije ReLU, povzeto po [3].

izračun vrednosti funkcije kot njenega odvoda. Tekom časa se je izkazala kot zelo uporabna aktivacijska funkcija na področju nevronskih mrež, kjer je tudi najbol uporabljena aktivacijska funkcija [33]. Ena izmed negativnih lastnosti funkcije ReLU je pojav imenovan *umiranje enot*. V nekaterih primerih lahko med učenjem nevron preide v stanje, kjer za večino vhodov postane neaktiven, saj so ti negativni ob vstopu v aktivacijsko funkcijo. To onemogoča nadaljnji pretok gradientov med učenjem in tako nevron umre. Da bi se temu problemu izognili, lahko uporabljamo druge vrste ReLU aktivacijskih funkcij, kot so *prepustni ReLU*, ki omogoča prehod gradienta tudi, ko enota ni aktivna [19].

2.1.1 Učenje umetne nevronske mreže

Za izboljšanje delovanja umetne nevronske mreže potrebujemo kriterij, s katerim ocenjujemo točnost rezultatov. Temu kriteriju pravimo *funkcija napake*, ki za napoved nevronske mreže t glede na podatke x in točni rezultat y poda napako te napovedi. Cilj učnega algoritma je minimizacija te napake, saj se s tem izboljša delovanje nevronske mreže. Primer pogosto uporabljene funkcije napake je *logaritmična napaka*. Njen matematični predpis je defini-

ran z enačbo

$$E(x) = -\sum_{n=0}^{M} y_n log(p_n). \tag{2.4}$$

Vrednost M označuje število vseh razredov, ki jih klasificiramo. y_n je binarni indikator, ki zavzame vrednost $y_n = 1$, če je razred n prisoten v podanih podatkih x. Če razred n v podatkih ni prisoten velja, da je $y_n = 0$. Vrednost p_n je verjetnost, s katero umetna nevronska mreža napove prisotnost razreda n v podatkih x. Pri optimizaciji umetne nevronske mreže nas predvsem zanima kako se vrednost napake spremeni, če spremenimo parametre mreže. Potrebujemo rešitev enačbe odvoda funkcije napake za vhodne podatke x glede na uteži mreže w

$$\frac{\partial E_x(w)}{\partial w}. \tag{2.5}$$

Ker funkcija napake ni konveksna in običajno vsebuje mnogo lokalnih minimumov, globalnega minimuma ne moremo poiskati eksplicitno. Problem nastane že pri sami evalvaciji vseh delnih odvodov funkcije napake glede na parametre mreže w. Tu se poslužujemo algoritma imenovanega *vzvratni prehod*. Ta algoritem omogoča učinkovit izračun odvodov, potrebnih za minimizacijo napake. Deluje po principu verižnega pravila odvajanja, ki definira izračun odvodov kompozitov funkcij. Za vsak nevron v mreži je potrebno izračunati njegovo *napako* δ. Za poljubni nevron a z indeksom i je ta napaka definirana z odvodom

$$\delta_i \equiv \frac{\partial E_x}{\partial a_i}. \tag{2.6}$$

S pomočjo verižnega pravila lahko izraz razširimo v končno obliko.

$$\delta_i \equiv \frac{\partial E_x}{\partial a_i} = \sum_k \frac{\partial E_x}{\partial a_k}\frac{\partial a_k}{\partial a_i} = h'(a_i)\sum_k \delta_k w_{ki}. \tag{2.7}$$

$$\frac{\partial E_x}{\partial w_{ji}} = \delta_j h(a_i). \tag{2.8}$$

Vsota po indeksu k poteka po nevronih, katerim nevron a_i pošilja podatke po povezavi z utežjo w_{ki}. Člen $h(a)$ predstavlja aktivacijsko funkcijo nevrona a. Tako je utemeljena potreba po odvedljivosti aktivacijskih funkcij s členom $h'(a_i)$. Odvod funkcije napake glede na poljubno utež v mreži nato

izračunamo z zmnožkom vrednosti napake nevrona na izhodnem delu povezave ter izhodom nevrona na vhodnem delu povezave. Opazimo, da so za izračun napake poljubnega nevrona δ_i, potrebne napake nevronov iz višjega nivoja δ_k. Na izhodnem nivoju te napake trivialno izračunamo ter njihove vrednosti propagiramo po mreži nazaj ter na poti pridobimo napake vseh ostalih nevronov. Od tod izhaja ime vzvratni prehod, saj informacije o napakah nevronov tečejo po mreži v obratni smeri, iz izhodov proti vhodom.

Zadnje dejanje v postopku učenja nevronske mreže je posodobitev vrednosti uteži tako, da se v naslednji iteraciji učenja napaka mreže zmanjša. Zaradi naštetih omejitev moramo za optimizacijo uteži uporabiti numerične metode. Uporabimo *gradientni spust*, kjer informacije o gradientu uteži izkoristimo za njihovo posodobitev. Vrednost uteži w_{ji} se tako tekom učenja spreminja po predpisu

$$w_{ji} = w_{ji} - \gamma \frac{\partial E_x}{\partial w_{ji}}. \tag{2.9}$$

Vrednost γ označuje hitrost učenja. Izberemo jo lahko sami in se tekom učenja ohrani kot konstanta. Lahko se tudi spreminja glede na poljubno določene kriterije, kot so na primer lastnosti gradienta samega.
Vidimo lahko, da postopek učenja umetnih nevronskih mrež poteka v dveh delih. V prvem delu umetna nevronska mreža izračuna rezultate glede na podane podatke x. Izračunajo se tudi aktivacije vseh nevronov v skritih nivojih nevronske mreže. Vrednosti aktivacij se shranijo, saj so potrebne v nadaljevanju. V drugem delu moramo posodobiti parametre nevronske mreže tako, da bo ta ob naslednji iteraciji izračunala točnejše rezultate.

Regularizacija učenja

Tekom učenja lahko naletimo na dodatno oviro, imenovano *pretirano prilagajanje*. Umetne nevronske mreže lahko aproksimirajo širok spekter funkcij ter iz vhodnih podatkov izluščijo podrobnosti, ki pomagajo pri klasifikaciji učnih primerov, a niso nujno reprezentativne za splošno reševanje problema. Take nevronske mreže zelo dobro delujejo na učni množici podatkov, a slabo ocenijo primere, do katerih tekom učenja niso imele dostopa. Da bi mini-

mizirali pretirano prilagajanje, uporabimo postopke *regularizacije*. Nekatere regularizacijske metode lahko vgradimo v sam učni algoritem. Ena izmed teh metod je *L2 normalizacija uteži* (angl. *L2 normalization*). Funkciji napake prištejemo dodatni regularizacijski člen, ki napako mreže poveča glede na velikosti uteži. Večje kot so uteži, bolj vplivajo na vrednost napake. Tako velja, da

$$L_2(w) = \frac{\lambda}{2} \sum_w w^2. \tag{2.10}$$

Regularizacijski termin prištejemo funkciji napake. Tako dobimo novo funkcijo za merjenje učinkovitosti mreže

$$E_r(w) = E(w) + L_2(w). \tag{2.11}$$

S parametrom λ nadziramo vpliv regularizacije na celotno funkcijo napake. Bližje kot je 0, manjši je vpliv regularizacije. Z L_2 regularizacijo mrežo silimo k učenju uteži, ki se počasi približujejo vrednosti 0, če niso uporabne pri klasifikaciji. Zato tej metodi pravimo tudi *propadanje uteži*. Druge metode regularizacije se usmerjajo na spreminjanje vhodnih podatkov. Če vemo, da določene transformacije nad vhodnimi podatki ne spremenijo izhodne vrednosti, lahko te transformacije dodamo podatkom, ki vstopajo v mrežo. Te transformacije so lahko dodan šum, v primeru 2-dimenzionalnih slik tudi afine transformacije, spreminjanje kontrasta, megljenje in mnoge druge. S tem izboljšamo verjetnost, da se tekom učenja mreža iz vhodnih podatkov nauči reprezentativne lastnosti.

Za učenje umetnih nevronskih mrež potrebujemo veliko količino podatkov. Mrežo lahko učimo na več k vhodnih podatkih x naenkrat. Tako lahko uteži mreže prilagodimo glede na njihove gradiente, ki so pridobljeni na vsakem primeru vhodnih podatkov x. Številu vhodnih podatkov k, na katerih se mreža naenkrat uči ter katere uporabi za izboljšanje delovanja, pravimo *velikost vhodne skupine* (angl. *batch size*). Zaželeno je, da je mreža tekom učenja naenkrat izpostavljena vsem vhodnim podatkom. Tako bi najbolje ocenila gradiente uteži ter se natančneje približevala minimumu. Ker zaradi velikanskih količin podatkov pogosto to ni mogoče, v praksi raje uporabljamo

k, za katerega velja $k << M$, kjer je M celotna količina podatkov. Manjša vrednost k omogoča hitrejše učenje, saj se vrednosti uteži posodobijo šele, ko je predelana celotna vhodna skupina. Prav tako je vrednost gradientov uteži, zaradi manjšega vzorca vhodnih podatkov, manj stabilna. Posledično je velikost k kompromis med hitrostjo in natančnostjo učenja.

Dodaten postopek, ki umetni nevronski mreži omogoča hitrejše učenje ter doseganje boljših rezultatov, je *normalizacija vhodne skupine* (angl. *batch normalization*) [23]. Tekom učenja se podatkom, ki tečejo preko mreže, spreminja distribucija vrednosti. To je posledica računanja operacij, ki jih definirajo nevroni. Ker se lahko distribucije hitro spreminjajo, se morajo predvsem nevroni v skritih nivojih prilagajati na te spremembe. Ta dodatna naloga upočasnjuje učenje ter ga naredi bolj odvisnega od začetnih vrednosti uteži ter hitrosti učenja. Tej vrsti spremembe distribucije pravimo tudi *notranji kovariantni zamik* (angl. *internal covariate shift*). Normalizacija vhodne skupine poskuša ta pojav odpraviti z normalizacijo vrednosti izhoda linearnega dela nevrona pred vstopom v nelinearno aktivacijsko funkcijo. Normaliziramo jih glede na povprečje μ_m ter varianco σ_m^2 celotne vhodne skupine. Izračun normalizacije $N(S_i) = y_i$ za skupino $S_m = x_0, ..., x_m$ pridobimo z naslednjimi enačbami

$$\mu_m = \frac{1}{m}\sum_{i=0}^{m} x_i, \tag{2.12}$$

$$\sigma_m^2 = \frac{1}{m}\sum_{i=0}^{m}(x_i - \mu_m)^2, \tag{2.13}$$

$$y_i = \lambda\frac{x_i - \mu_m}{\sqrt{\sigma_m^2 + \epsilon}} + \beta. \tag{2.14}$$

Vrednosti λ in β se nivo normalizacije nauči tekom učenja. Njuna naloga je skaliranje in zamik normaliziranih vrednosti. Tako ohranimo reprezentativnost posameznih nivojev.

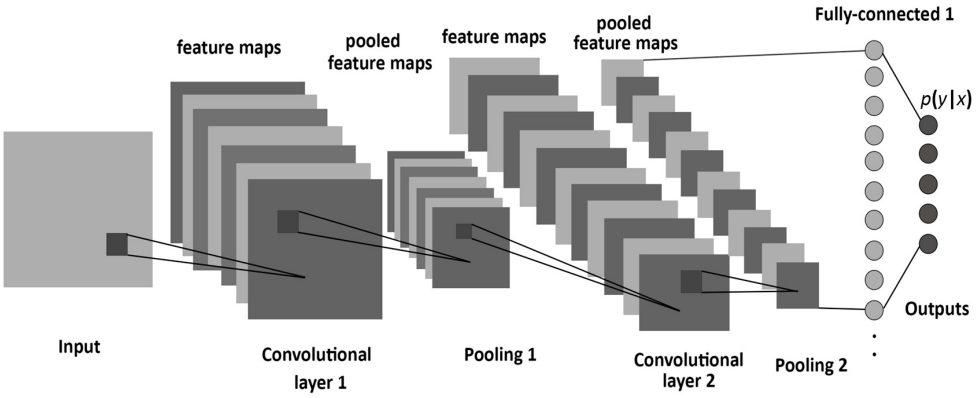
2.2 Konvolucijske nevronske mreže

Konvolucijske nevronske mreže [22] so vrsta umetnih nevronskih mrež, katerih glavni gradnik je linearna matematična operacija, imenovana konvolucija. Te specializirane mreže se najpogosteje uporabljajo za procesiranje vizualnih informacij, kot so digitalne fotografije. Prav tako so zaradi uporabe konvolucije invariantne na določene transformacije, ki so pogosto prisotne v slikah. Zaradi teh lastnosti so konvolucijske nevronske mreže močna orodja, ki se uspešno uporabljajo na področju računalniškega vida. Tam rešujejo probleme klasifikacije slik ter semantične segmentacije.

2.2.1 Principi konvolucijskih nevronskih mrež

Vhodni podatki konvolucijskih nevronskih mrež so najpogosteje vizualne informacije. Predstavljene so v formatu $W \times H \times C$, kjer je W širina slike, H njena višina in C število barvnih kanalov. Tej več dimenzionalni matriki pravimo *tenzor*. Operacije, ki jih mreža izvaja nad vhodnimi podatki, so urejene v *nivoje*. Umetne nevronske mreže v podatkih iščejo vzorce. Pri konvolucijskih nevronskih mrežah opazimo podobno obnašanje. Konvolucijski nivoji bližje vhodnemu nivoju v sliki razpoznavajo najbolj preproste vizualne vzorce. Primer takšnih vzorcev so enostavni robovi. Nadaljnji nivoji preproste značilke, kot so robovi, združujejo v kompleksnejše entitete. V zadnjih nivojih so te lahko že celotni objekti. Konvolucijske mreže s hierarhijo operacij iz enostavnejših značilk zgradijo bolj kompleksne.
Izhod nivojev konvolucijske nevronske mreže so večdimenzionalni tenzorji. Pri procesiranju vizualnih informacij pogosto zajemajo štiri dimenzije. Prvi dve dimenziji predstavljata prostorsko velikost informacij. Za posamezno značilko definirata število nevronov, ki se v vhodnih podatkih odzovejo na prisotnost te značilke. Tretji dimenziji tenzorja pravimo *zbirka značilk* tega nivoja. Vsak element te dimenzije predstavlja določeno značilko. Nahajanje te značilke, kot je na primer enostaven rob, v vhodnih podatkih najdemo v prvih dveh dimenzijah tenzorja. Številčnost zbirke značilk na posameznih nivo-

jih z globino konvolucijske nevronske mreže praviloma narašča. Arhitekturo konvolucijske nevronske mreže razvijalec določi sam, tako kot pri umetnih nevronskih mrežah. Ta se razlikuje glede na problemska področja. Na vsa-



Slika 2.4: Primer arhitekture nivojev konvolucijske nevronske mreže. Povzeto po [7].

kem nivoju konvolucijske nevronske mreže izbiramo med mnogimi različnimi operacijami, ne samo konvolucijami. Nekatere smo omenili že pri umetnih nevronskih mrežah. Primer take funkcije je ReLU ali pa funkcija napake. Druge, še neomenjene, opišemo v nadaljevanju.

Konvolucija

Konvolucija (angl. *convolution*) je linearna matematična operacija med dvema funkcijama. V primeru vizualnih informacij sta ti dve funkciji vhodna slika in *konvolucijski filter*, ki je po širini in višini manjši ali enak vhodni sliki. Rezultat konvolucije je spremenjena vhodna slika glede na podani filter. Za izračun spremenjene slike konvolucijski filter premikamo po vhodni sliki ter za vsako slikovno točko, na kateri filter v celoti leži znotraj slike, izračunamo predpis

$$g(i,j) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} I(i-m, j-n) f(m,n). \tag{2.15}$$

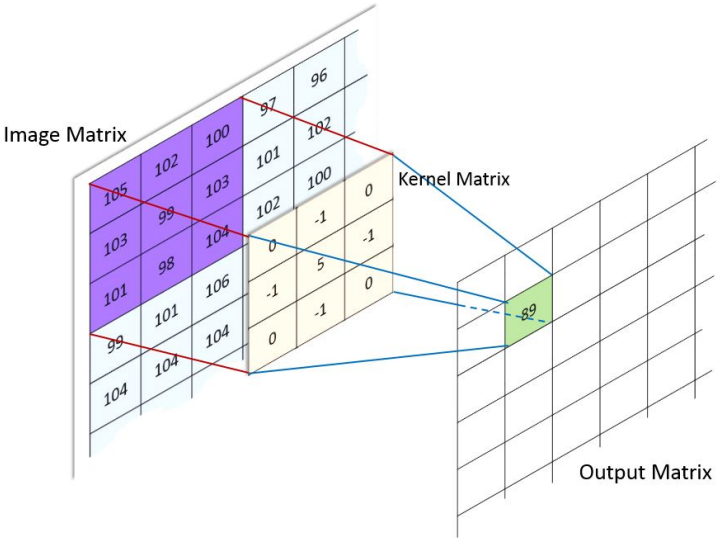
Izračun konvolucije slike I s filtrom f na točki slike (i,j) je tako definiran kot utežena vsota intenzitet slike I na lokaciji (i,j) ter okoliških slikovnih

točkah. Uteži te vsote in število slikovnih točk okoli (i, j), uporabljenih v vsoti, definirata filter f ter njegova velikost (M, N). Operacijo konvolucije ponavadi zapišemo z operatorjem $*$. Zapis enačbe konvolucije je

$$g = I * f. \tag{2.16}$$

Ker mora konvolucijski filter v celoti ležati znotraj slike, je rezultat konvolucije manjši od vhodne slike. Vhodni sliki lahko ob robove dodamo slikovne točke z intenziteto 0. S tem dosežemo, da se velikost slike po konvoluciji ohrani.

Rezultat konvolucije slike s filtrom si lahko interpretiramo kot odziv podanega filtra na sliko. Slika 2.6 predstavlja primer filtra, ki v vhodni sliki poišče robove.

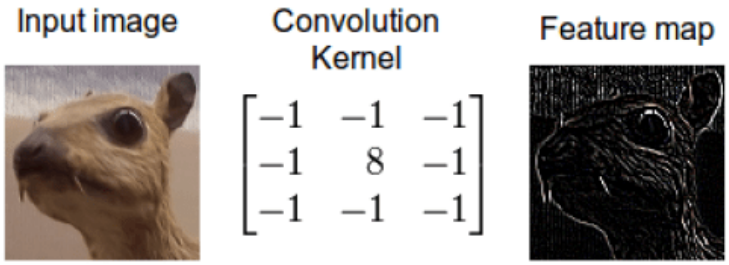


Slika 2.5: Primer konvolucije slike z jedrom, povzeto po [4].

S pomočjo konvolucije nevronske mreže v svojo arhitekturo vgradijo operacije, ki so prilagojene naravi vizualnih informacij, kot so slike. Predpostavimo, da na vrednost določene slikovne točke bolj vplivajo blizu ležeče slikovne točke kot bolj oddaljene. To nam omogoča, da močno zmanjšamo število učljivih parametrov ter pohitrimo samo računanje rezultata, saj je prostorska velikost filtra mnogo manjša od same slike. Prav tako je to

način regularizacije nevronske mreže, saj se z manjšim številom povezav ta težje pretirano prilagodi podatkom. Tej lastnosti konvolucije pravimo *redka povezanost*. Konvolucijo implementiramo kot drseči filter, katerega odziv računamo na vsaki slikovni točki vhodne slike. Tako za vsako točko uporabimo isto skupino filtrov, da izračunamo njihov odziv na sliki. Tej lastnosti pravimo *deljenje uteži*. Ta omogoča, da nadaljnjo prihranimo na številu učljivih parametrov nevronske mreže. Pomembna lastnost, ki jo nevronske mreže pridobijo z implementacijo konvolucije, je *invarianca glede na premik vhoda*. Če definiramo funkcijo f, ki vhodne podatke x translira za nek premik p, potem za konvolucijo g velja

$$g(f(x,p)) = f(g(x),p). \tag{2.17}$$



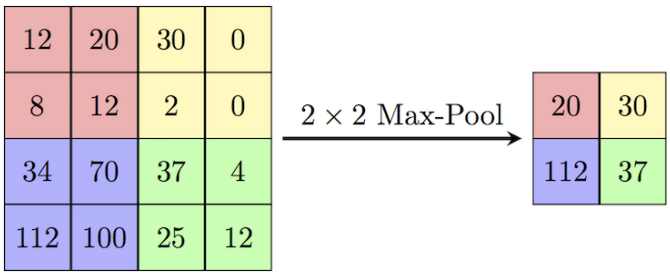
Slika 2.6: Primer konvolucijskega filtra, ki v vhodni sliki izpostavi robove, povzeto po [5].

Ko v nevronsko mrežo vstavljamo konvolucijski nivo, potrebujemo določiti nekatere parametre tega nivoja. Sam rezultat konvolucije je pogojen z izbiro teh parametrov. Parametri konvolucijskega nivoja so *velikost filtrov*, *korak konvolucije* ter *način zapolnitve robov*. Velikost filtrov konvolucije nam omogoča, da določimo kolikšen del vhodnih podatkov ali slike filter prekriva. Prav tako s tem parametrom določimo koliko filtrov bomo uporabili na tem nivoju. Več kot je filtrov, več je značilk, ki jih nevronska mreža išče v vhodnih podatkih. Če generiranih filtrov ne želimo računati na vsaki slikovni točki, lahko povečamo korak konvolucije, da ta izpusti n slikovnih točk med konvolucijo. Da odziv izračunamo za vsako slikovno točko, korak nastavimo

na 1. Če je korak večji od 1, je izhod prostorsko manjši od vhoda v odvisnosti od velikosti izbranega koraka. Omenili smo, da konvolucija zmanjša velikost vhodne slike, tudi če jo izvajamo s korakom velikosti 1. Vhodni sliki smo zato dodali robove in tako ohranili njeno velikost po konvoluciji. Kako nastavimo intenzitetne vrednosti robov, določimo z izbiro načina zapolnitve robov. Postopkov za zapolnitev robov je veliko. Najenostavnejši postopek robove zapolni s slikovnimi točkami z intenziteto enako 0.

Združevanje

Združevanje (angl. *max pooling*) je operacija, ki določeno točko v sliki nadomesti z rezultatom funkcije te točke in točk v njeni okolici. Za razliko od konvolucije, nivo združevanja ponavadi nima učljivih parametrov in je preprost za implementacijo. Pogosto uporabljene funkcije so povprečje, L_2 norma ali pa uteženo povprečje. Najbolj popularna je funkcija $max(I)$, ki poišče maksimalno vrednost v določeni regiji. Vse te funkcije dodatno omogočajo lokalno invariantnost na premik.



Slika 2.7: Primer max združevanja z velikostjo koraka 2 ter velikostjo regije združevanja 2×2, povzeto po [6].

Operacija združevanja se najpogosteje ne računa na vsaki slikovni točki v vhodni sliki. Ker ta funkcija vrne nekakšno posplošitev določene regije v sliki, ne potrebujemo njenega rezultata na vsaki slikovni točki. V praksi se velikokrat izpusti vsako drugo točko v posamezni vrstici in stolpcu. Rezultat združevanja je tako približno prostorsko štirikrat manjša slika. Številu slikovnih točk za katerega se premaknemo rečemo *korak*. To je parameter

katerega vrednost določimo ob deklaraciji združevalnega nivoja. Prav tako ob inicializaciji izberemo velikost regije združevanja ter funkcijo združevanja. Arhitekture nevronskih mrež se močno razlikujejo v odvisnosti od problema, ki ga rešujejo. Najpogosteje opazimo sosledje konvolucijskega ter aktivacijskega nivoja. Temu lahko sledi nivo združevanja. Med nivojema konvolucije ter aktivacijske funkcije običajno vstavimo normalizacijo vhodne skupine. Kombinacije teh nivojev ponavljamo poljubno krat. Na koncu mreže dodamo funkcijo napake, ki ocenjuje točnost pridobljenih rezultatov.

Poglavje 3

Detekcija sprememb in gibanja v video sekvencah

Problem detekcije sprememb in gibanja v video sekvencah je kompleksen. Naš pristop temelji na postopku iskanja razlike med sliko ozadja ter posameznimi slikami sekvence. Naša rešitev je sestavljena iz dveh delov. V prvem delu generiramo približek ozadja za vsako sekvenco v podatkovni zbirki. To ozadje v nekaterih primerih ročno določimo. Kjer to ni mogoče, ozadje generiramo s filtrom mediane. V drugem delu razlike med slikami sekvence ter sliko ozadja poiščemo s pomočjo siamske konvolucijske nevronske mreže [15].

3.1 Generiranje ozadja video sekvence

Za iskanje razlik med ozadjem video sekvence ter posameznimi slikami iz sekvence potrebujemo način ocenjevanja ozadja. Metod za ocenjevanje ozadja je veliko [46] [45] [31] [39] [49] [35]. Razlikujejo se po kompleksnosti ter kvaliteti generiranih modelov ozadja [46]. Glavna naloga vseh omenjenih algoritmov je ta, da predmete, ki niso del ozadja, ne vključijo v svoj model. Pri tem ohranjajo ter posodabljajo slikovne točke dejanskega ozadja. Generiranje modela ozadja je zahteven proces. Mnogo motečih dejavnikov v video sekvencah onemogoča ocenjevanje povsem točnih slik ozadja na podlagi vi-

zualnih ter časovnih informacij. Objekti v ozadju in ospredju so si lahko po barvi podobni ter tako težko razločljivi. Hitre in dinamične spremembe v osvetlitvi lahko pokvarijo model ozadja ter povzročijo neželene aktivacije detekcijskega algoritma. Nekateri objekti v ospredju nenadno končajo gibanje za dlje časa ter ga kasneje ponovno nadaljujejo [43]. Načini spopadanja s takimi težavami so odvisni od posameznih implementacij algoritmov, ki ozadja generirajo. Zaradi teh in drugih elementov video sekvenc so modeli ozadja pogosto le približki dejanskega ozadja. Zaželeno je, da so algoritmi detekcije razlik robustni na določene anomalije, ki se lahko pojavijo v generiranem ozadju. V našem delu za oceno slike ozadja, kjer te ne moremo določiti ročno, uporabimo *filter mediane*. Predpostavljamo, da se objekti v ospredju pogosto gibljejo in se tako v sekvenci pojavljajo le v delčku vseh slik. Ker je video sekvenca zbirka več slik posnetih ob različnih časih, si premikajoče objekte pred ozadjem interpretiramo kot šum v posnetku, ki moti sliko ozadja. Filter mediane se lahko v različnih oblikah uporablja za odpravljanje šuma v slikah [18]. Ta deluje tako, da vsako točko v sliki nadomesti z mediano poljubno velike regije slikovnih točk v njeni okolici.

(a) (b)

Slika 3.1: V sliki (b) je večina šuma iz slike (a) odstranjenega s pomočjo filtra mediane.

V našem primeru ne odpravljamo šuma v prostorski komponenti video posnetka. Šum odpravljamo v časovni komponenti. Vsaka slikovna točka tako pridobljenega ozadja je mediana isto-ležečih točk slik sekvence, posnetih ob

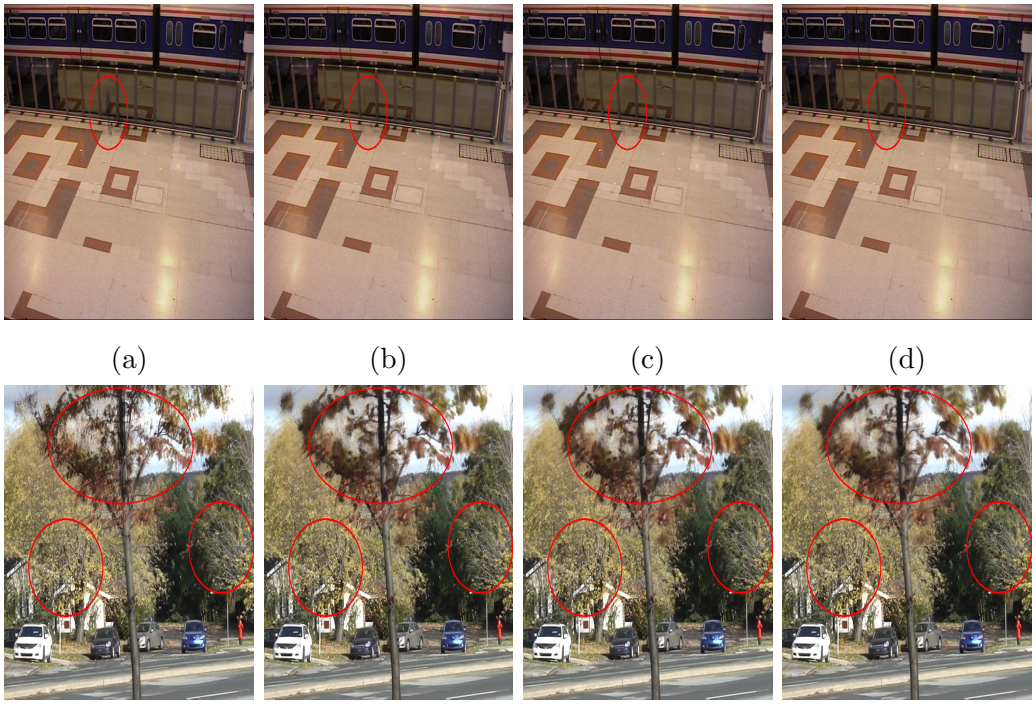
različnih časih. Koliko slik iz sekvence obravnavamo pri izračunu ozadja je odvisno od več dejavnikov. Ob večji prisotnosti objektov ospredja v video sekvenci potrebujemo večje število slik za boljšo oceno ozadja. Dinamično je lahko tudi samo ozadje sekvence. Če za izračun modela ozadja vzamemo preveč slik, lahko to gibanje pokvari reprezentacijo dejanskega ozadja. Tako vidimo, da je število slik, ki jih obravnavamo pri izračunu mediane, kompromis med natančnostjo videza ozadja ter številom anomalij, ki jih povzročajo predmeti v ospredju.

Za demonstracijo tega kompromisa smo generirali 8 ozadji 2 video sekvenc, kjer smo za vsako ozadje uporabili N naključno izbranih različnih slik. Za N smo določili 10, 50, 100 ter 200 slik. Na Sliki 3.2 v prvi vrstici opazimo vpliv anomalij objektov iz ospredja na kvaliteto ozadja, v drugi natančnost reprezentacij objektov v ozadju zaradi njihovega gibanja. Vidimo, da z naraščanjem števila slik zmanjšujemo število anomalij, ki jih povzročajo objekti v ospredju. Hkrati tudi izgubljamo natančnost reprezentacij objektov v ozadju, ki se tekom sekvence premikajo.

Izračun mediane je lahko za velike količine slik preveč zamuden postopek. Da ga pohitrimo, lahko izrabimo redundantnost vizualnih informacij. Sliki ob času t ter $t + 1$ sta si v veliko primerih zelo podobni, saj se vsebina v zaporednih slikah malo spreminja. Količina sprememb iz slike ob času t ter $t + 1$ je odvisna od hitrosti spreminjanja objektov v okolju, ki ga snemamo ter števila slik na sekundo, ki jih lahko posname kamera. Za izračun smo uporabili vsako $\frac{N}{s}$ sliko. Vrednost s določa koliko zaporednih slik izpustimo.

3.2 Siamska konvolucijska nevronska mreža

Ko imamo za video sekvenco generiran ozadje, moramo poiskati relevantne razlike med posameznimi slikami iz sekvence ter sliko ozadja. Preprosto odštevanje ter nato vpragovanje vrednosti nudi slabe rezultate zaradi velike količine šuma, ki je lahko prisoten v slikah. Za reševanje tega problema smo razvili siamsko konvolucijsko nevronsko mrežo. Naloga mreže je, da ob



(a) (b) (c) (d)

(e) (f) (g) (h)

Slika 3.2: Za generiranje slik (a) in (e) smo uporabili 10 naključno izbranih slik iz posamezne video sekvence, za (b) in (f) 50, za (c) in (g) 100 ter za (d) in (h) 200 slik. V prvi vrstici lahko opazimo, da z večanjem števila slik, ki jih obravnavamo, odpravljamo anomalije, ki jih povzročajo elementi ospredja v ozadju. V drugi vrstici lahko opazimo, da povečevanje števila slik zmanjšuje kvaliteto ozadja, ki se tekom sekvence giblje.

podani sliki ozadja video sekvence ter posamezni sliki iz sekvence določi binarno masko relevantnega gibanja, ki se nahaja v sliki. S pomočjo nivojev konvolucijske nevronske mreže VGG-16 [37] semantično opišemo sliko iz video sekvence ter generirano sliko njenega ozadja. Tako posamezne slikovne točke pridobljenih zbirk značilk ne predstavljajo barv v sliki, temveč kompleksnejše vizualne elemente. Ti so na nižjih nivojih robovi, črte, točke, na višjih tudi sestavljeni objekti. Tako razlike iščemo med kompleksnejšimi vizualnimi objekti.

3.2.1 Konvolucijska mreža VGG-16

Konvolucijska nevronska mreža VGG-16 [37] je mreža, katere naloga je klasifikacija objektov, ki so prisotni v slikah. Učena in evalvirana je bila na podatkovni zbirki ILSVRC [34], ki vsebuje slike 1000 različnih objektov. Zbirka vsebuje slike prevoznih sredstev, živali, ljudi ter drugih objektov. Cilj te podatkovne zbirke je testiranje algoritmov, katerih naloga je detekcija vsebine slik.

Mreža je sestavljena iz 16 nivojev, ki vsebujejo učljive parametre. Uporablja sosledje 13 konvolucijskih nivojev ter 3 polno povezane nivoje. Vsi konvolucijski nivoji imajo enake, 3×3 velike konvolucijske filtre. Mreža vsebuje 5 nivojev združevanja, ki sledijo nekaterim konvolucijskim nivojem. Naloga teh je zmanjševanje prostorskih dimenzij izhodov konvolucijskih nivojev. Vsaki konvoluciji sledi nelinearna funkcija ReLU. Maksimalna širina zbirk značilk konvolucijskih nivojev je 512 značilk. Polno-povezani nivoji, ki sledijo konvolucijskim, število značilk povečajo na 4096. Njihova prostorska velikost je 1, torej predstavljajo vektor 4096 različnih skalarnih vrednosti. V zadnjem polno povezanem nivoju se izhod zmanjša na 1000 značilk. Zadnji nivo mreže je nivo softmax, ki vsako izmed 1000 značilk preslika v klasifikacijsko verjetnost za posamezni razred. Mreža VGG-16 na podatkovni zbirki ILSVRC 2012 dosega točnost 92.8 procentov za 5 najverjetnejših napovedi (angl. *top 5 accuracy*) ter 75.6 procentov za 1 najverjetnejšo napoved (angl. *top 1 accuracy*) [28]. V članku [37] avtorji za doseganje boljše točnosti združijo rezultate dveh VGG arhitektur, VGG-16 ter VGG-19. VGG-19 je verzija mreže VGG-16, ki vsebuje dodatne konvolucijske nivoje.

3.2.2 Siamska konvolucijska nevronska mreža *SubNet*

Naša konvolucijska mreža, v nadaljevanju mreža *SubNet*, je mreža, ki smo jo razvili za opravljanje naloge detekcije razlik med sliko ozadja ter sliko, v kateri sta prisotna tako ozadje kot ospredje. Za opravljanje te naloge je mreža sestavljena iz dveh delov. Prvi del mreže *SubNet* vhodni sliki se-

mantično opiše glede na vizualno vsebino slik. To stori s pomočjo 7 najnižjih konvolucijskih nivojev mreže VGG-16. Ker je mreža VGG-16 naučena klasificirati 1000 različnih objektov, njeni konvolucijski filtri zajemajo široko področje vizualnih značilk. Te raznolike filtre uporabimo za opisovanje vhodnih slik mreže *SubNet*. Konvolucijski nivoji se istočasno izvedejo nad obema vhodnima slikama s pomočjo siamske arhitekture. Prvi del mreže *SubNet* si lahko predstavljamo kot dve ločeni konvolucijski mreži, ki si med seboj delita parametre. Temu delu mreže pravimo tudi *enkoder*.

Drugi del mreže *SubNet* je zadolžen za iskanjem relevantnih razlik med vhodnima slikama ter za generiranje končne verjetnostne segmentacijske maske. Vhodni podatki tega dela mreže so rezultati sedmega konvolucijskega nivoja prvega dela mreže. To so pari zbirk značilk istih konvolucijskih filtrov nad slikama ozadja ter vhodno sliko sekvence. Pare zbirk značilk nato odštejemo med seboj ter izračunamo absolutne vrednosti razlik. Pridobljene vrednosti predstavljajo razlike med enakimi vizualnimi elementi v sliki ozadja ter sliki iz video sekvence. Ker obdržimo razlike iz vhodne slike ter slike ozadja, lahko detektiramo tudi objekte, ki jih mreža ne more semantično opisati s svojimi filtri. Da bi detektirali take objekte, se morajo ti nahajati na površini, ki jo mreža s svojimi konvolucijskimi filtri lahko opiše. Objekte v ospredju lahko opišemo tudi z vsemi tistimi regijami ozadja, ki jih objekt ospredja zakriva ter jih mreža lahko semantično opiše.

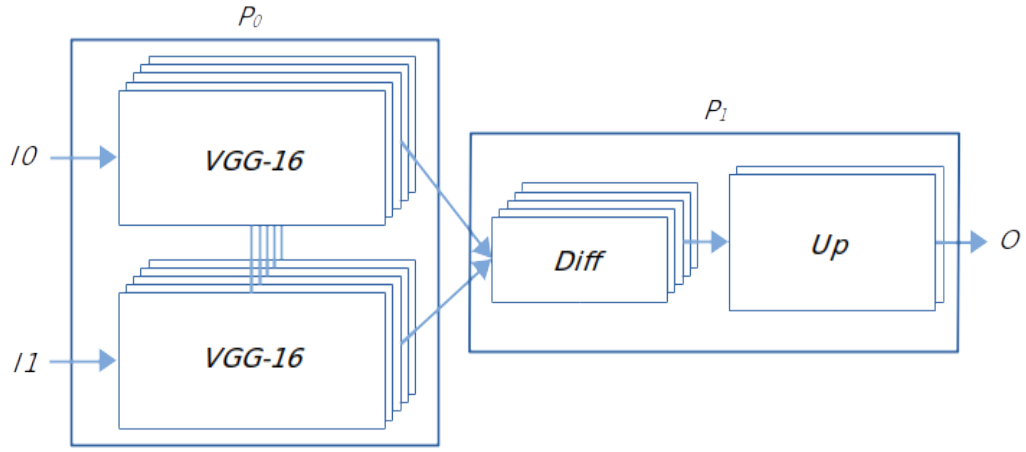
Gibanje je lahko prisotno tudi v objektih ozadja. To lahko privede do potencialno neželenih detekcij. Da bi se temu izognili, smo nivoju absolutne razlike dodali štiri konvolucijske nivoje. Tako lahko nekatere oblike gibanja mreža tekom učenja zavrže ter ohrani druge. Izhod teh konvolucijskih nivojev je po velikosti 16 krat manjši kot velikost vhodne slike ter vsebuje 256 različnih značilk. Rezultate moramo zato povečati do prvotne velikosti vhoda ter zmanjšati število značilk. To storimo z dvema nivojema transponirane konvolucije s korakom 2. Ta nivo ima učljive konvolucijske filtre. Ker je naloga transponirane konvolucije povečanje vhodnih podatkov, njene filtre inicializiramo tako, da tej izvajajo bi-kubično interpolacijo.

Vsak nivo transponirane konvolucije velikost slike poveča 4 krat. Po vsakem nivoju transponirane konvolucije uporabimo nivo navadne konvolucije, ki postopoma zmanjšuje število značilk, dokler to ni enako 2.

V mreži po vsakem konvolucijskem nivoju ter nivoju transponirane konvolucije uporabimo normalizacijo skupine, kateri sledi aktivacijska funkcija ReLU. Izjema je le zadnji nivo konvolucije, kateremu sledi nivo softmax. Ta vrednosti izhoda zadnjega nivoja mreže normalizira, da je vsota teh po tretji dimenziji enaka 1. Tako sta izhod mreže dve verjetnostni maski vhodne slike. V prvi vsaka točka predstavlja verjetnost, da istoležna slikovna točka v vhodni sliki pripada statičnemu ozadju. V drugi vsaka točka predstavlja verjetnost, da istoležna slikovna točka v vhodni sliki pripada ospredju, ki ga v večini primerov sestavljajo gibajoči se objekti. Pri učenju nevronske mreže nivoju softmax dodamo še nivo logaritmične napake, ki je določena z enačbo 3.1

$$E(x) = -\sum_{n=0}^{M} y_n log(p_n), \tag{3.1}$$

kjer je x posamezna točka v izhodu nevronske mreže. Vrednost M teče čez celotno število razredov, ki jih segmentiramo. V našem primeru je $M = 1$, saj imamo dva razreda, kjer je prvi označen z vrednostjo 0. Vrednost y_n predstavlja oznako, ali posamezna slikovna točka pripada razredu n. Verjetnost p_n predstavlja napoved mreže, da slikovna točka pripada razredu n. Tako moramo oceniti vsako točko v izhodu nevronske mreže, da pridobimo napako segmentacije. To tekom učenja poskušamo zmanjšati. Arhitektura mreže $SubNet$ je orisana v Sliki 3.3.



(a)

Slika 3.3: Oris arhitekture mreže *SubNet*. Oznaki *I0* ter *I1* predstavljata vhodne podatke v nevronsko mrežo. Oznaka *O* predstavlja izhod nevronske mreže. P_0 ter P_1 predstavljata prvi del mreže ter drugi del mreže. Modre črte brez puščic, ki povezujejo nivoje označene kot *VGG-16*, nakazujejo deljenje učljivih parametrov teh nivojev. To je siamski del mreže *SubNet*. Izhodi dela P_0 vstopijo v drugi del mreže. Tu se značilke v nivojih razlike, ki so označeni z *Diff*, med seboj odštejejo ter nadaljnjo procesirajo. Nivoji v delu mreže, ki je označen z oznako *Up*, povečajo vhodne vrednosti na velikost vhodov mreže ter zmanjšajo število značilk.

Poglavje 4

Eksperimentalna evalvacija

V tem poglavju opišemo postopek učenja ter evalvacije siamske konvolucijske nevronske mreže. V Poglavju 4.1 je predstavljeno programsko ter strojno okolje, v katerem smo učenje in evalvacijo izvajali. Nato v Poglavju 4.2 predstavimo sam postopek učenja. Opišemo podatkovne zbirke, ki so bile uporabljene tako za učenje kot evalvacijo. Nato opišemo kako smo učne slike izbrali ter jih pripravili za postopek učenja. Sledi opis vrednosti učnih parametrov, ki smo jih uporabili pri učenju. Nato v Poglavju 4.3 predstavimo kriterije po katerih smo ocenjevali delovanje nevronske mreže. Na koncu v Poglavju 4.4 predstavimo rezultate, ki jih dosega nevronska mreža.

4.1 Programsko in strojno okolje

4.1.1 MatConvNet

Za razvoj siamske konvolucijske nevronske mreže, ki detektira spremembe in gibanje, smo uporabili programsko orodje MatConvNet [41]. Integrirano v razvojno okolje MATLAB [29], MatConvNet nudi hitro prototipiranje ter eksperimentiranje z različnimi vrstami konvolucijskih nevronskih mrež. Zaradi podpore izvajanja učenja na grafičnih procesorskih enotah, lahko s tem orodjem učimo kompleksne nevronske mreže na veliki količini podatkov. MatConvNet posamezne nivoje konvolucijske mreže prikaže kot enostavne gra-

dnike, ki jih dodajamo celotni strukturi mreže ter jih povezujemo med seboj. Ti nivoji predstavljajo optimizirane standardne rutine, kot so konvolucija, ReLu in druge operacije, pogosto prisotne v arhitekturi konvolucijskih nevronskih mrež. Različne nivoje definiramo in združimo v dveh oblikah konvolucijskih nevronskih mrež. Ti dve sta *simplenn* ter *dagnn*. Simplenn implementira enostavnejše nevronske mreže, ki vsebujejo linearno topologijo nivojev. Dagnn definicija nevronskih mrež nam omogoča sestavo bolj kompleksnih topologij, ki niso nujno linearne. V našem delu uporabljamo definicijo dagnn za implementacijo nevronske mreže.

Dagnn nam omogoča objektno orientirano pisanje nevronske mreže, kjer razred dagnn implementira posamezne nivoje nevronske mreže ter metode, ki omogočajo kreiranje teh nivojev. Ko ustvarimo prazno mrežo dagnn, vanjo dodamo nov operacijski nivo s klicem metode *addLayers*. Argumenti te metode so ime nivoja, ki ga definiramo, vrsta operacije, ki jo želimo dodati, ime vhoda in izhoda tega nivoja ter imena učljivih parametrov mreže, da lahko do njih enostavneje dostopamo kasneje. Različne operacije, ki jih podamo metodi addLayers, so definirane v razredu dagnn. Primer teh je *dagnn.Conv*, ki omogoča dodajanje konvolucijskega nivoja. Implementiranih je še mnogo drugih funkcij, ki so pogosti gradniki konvolucijskih nevronskih mrež. Če želimo v nevronski mreži med seboj povezati dva nivoja, za vhod enega podamo ime izhoda drugega nivoja. Paziti moramo, da so dimenzije podatkov, ki se pretakajo med njima, kompatibilne z velikostmi filtrov ali drugih učljivih parametrov, ki jih nivoja vsebujeta. Ustvarimo lahko tudi lastni nivo s poljubno funkcionalnostjo. Da to storimo, je potrebno definirati njegovo delovanje v času prehoda podatkov po mreži naprej in nazaj. Podati moramo izračunane odvode operacij nivoja glede na prejete vhodne podatke.

MatConvNet nam s svojo zbirko implementiranih primerov uporabe okolja omogoča, da hitro spoznamo enostavnejše vidike ustvarjanja ter učenja nevronske mreže. Prek njihove uradne spletne strani [28] lahko dostopamo do mnogih prednaučenih modelov popularnih arhitektur, kot so VGG ali AlexNet [25]. Te lahko prenesemo ter uporabimo v lastnih projektih.

4.1.2 Strojno okolje

Učenje in evalvacija sta potekala na strežniku v laboratoriju Vicos FRI. Ta strežnik vsebuje procesor tipa *INTEL Xeon E5-1650 v3 3.50GHz*. Temu je na voljo *64 GB* sistemskega pomnilnika. Učenje in evalvacijo smo izvajali s pomočjo diskretne grafične kartice *NVIDIA GeForce GTX 980*, ki vsebuje *4 GB* grafičnega pomnilnika.

4.2 Postopek učenja

Med postopkom razvoja konvolucijske nevronske mreže smo uporabili več različnih podatkovnih zbirk. Učenje mreže smo izvajali na podatkovni zbirki CDNET [43], dostopni na strani [17]. Za testiranje njenega delovanja smo poleg zbirke CDNET uporabili še zbirki Wallflower [40] in SGM-RGBD [16]. Podatkovna zbirka Wallflower je dostopna na spletni strani [42], zbirka SGM-RGBD na [36]. Razlog, da je zbirka CDNET uporabljena tako pri učenju kot evalvaciji, je v tem, da takšen protokol evalvacije uporabljajo na strani CDNET [17] za ocenjevanje objavljenih algoritmov.
Za učenje na podatkovni zbirki CDNET je na voljo le podmnožica vseh slik posameznih video sekvenc. Učenje lahko izvajamo na vseh slikah, ki so na voljo, ali le na podmnožici teh. Nato algoritem objavimo na spletni strani CDNET, kjer se evalvacija našega algoritma izvede tudi na preostalih slikah sekvence, ki niso na voljo med učenjem. Tako želijo avtorji preprečiti pretirano prilagajanje na video sekvence.
Razvite mreže na podatkovnih zbirkah Wallflower ter SGM-RGBD nismo učili ter ju uporabljamo izključno za evalvacijo delovanja.

CDNET

Podatkovna zbirka CDNET [43] vsebuje 11 različnih kategorij, ki so poimenovane *Bad Weather*, *Low Framerate*, *Baseline*, *Dynamic Background*, *Night Videos*, *Camera Jitter*, *PTZ*, *Shadow*, *Turbulence*, *Intermittent Object Motion* ter *Thermal*. Vsaka izmed teh kategorij predstavlja problematične okoliščine,

ki otežujejo detekcijo gibanja. Posamezni primeri izbranih kategorij so prikazani na Sliki 4.1 ter Sliki 4.2. V nadaljevanju predstavimo izbrane kategorije iz podatkovne zbirke CDNET, na katerih smo učili našo konvolucijsko nevronsko mrežo.

Kategorija *Baseline* vsebuje 4 video sekvence. Zaradi preprostosti primerov, ki vsebujejo le manjše število zahtevnejših elementov, je detekcija gibanja lažja kot v drugih kategorijah. *Dynamic background* vsebuje 6 video sekvenc. V teh sekvencah je prisotno močno gibanje v ozadju. Primer tega so drevesa, ki se zibljejo v vetru ter valovanje vodne gladine. *Shadow* prav tako vsebuje 6 video sekvenc. Pri teh se v ospredju pojavljajo premikajoče se sence. Sence lahko prihajajo iz ozadja. Prav tako jih povzročajo premikajoči se objekti, ki so del ospredja. *Thermal* vsebuje 5 video sekvenc. Video sekvence v tej kategoriji so posnete z infrardečo kamero. *Bad Weather* vsebuje 4 video sekvence, posnete v slabih vremenskih razmerah in slabi vidljivosti. V vseh primerih prikazujejo okolje v snežnem metežu. *Low Frame-Rate* vsebuje 4 video sekvence. Vse sekvence so posnete z internetnimi kamerami (ang. IP camera), kjer je omejena količina prenosa podatkov. Sekvence so posnete z majhnim številom sličic na sekundo. Zajeto gibanje je sporatično in nenadno. Night vsebuje 6 posnetkov prometnih okolij. Video sekvence so posnete ponoči, kjer prevladuje predvsem slaba vidljivost. Žarometi luči ter njihovi odboji dodatno otežujejo zaznavanje relevantnega gibanja. Zadnja kategorija *Air Turbulence* vsebuje 4 video sekvence. Video je posnet z infrardečo kamero na vroč poletni dan. Za snemanje je uporabljena telephoto leča. Zaradi teh dejavnikov se v posnetku pojavlja turbulenca, ki močno moti detekcijo gibanja. Objekti v ospredju se po velikosti močno spreminjajo.
Camera jitter vsebuje 4 posnetke raznolikih okolij. Video sekvence so posnete s kamerami, ki se tekom snemanja tresejo. Tako nenadno gibanje predstavlja težaven problem pri segmentaciji relevantnega gibanja.

Kategorije *Baseline*, *Dynamic background*, *Shadow*, *Thermal*, *Bad Weather*, *Low Frame-Rate*, Night, *Air Turbulence* ter *Camera Jitter* smo uporabili za učenje nevronske mreže. Za učenje so nam na voljo vse kategorije v

podatkovni zbirki CDNET. Razlog, da med učenjem nismo uporabili kategorije *PTZ* je v tem, da se tekom sekvenc kategorije kamera preveč premika. Tako lahko ta popolnoma zamenja okolje, ki ga opazujemo. Ker učenje izvajamo z enim samim ozadjem na sekvenco, bi takšni učni primeri predstavljali povsem ne-reprezentativne razlike med ozadjem in posameznimi slikami sekvence. Med učenjem prav tako ne uporabljamo sekvenc kategorije *Intermittent Object Motion*. Ne uporabljamo jih zato, ker so v nekaterih sekvencah prisotni objekti, ki so sprva del ozadja ter se nato nenadoma začenjajo premikati. Naša metoda, ki za učenje uporablja statično ozadje, bi v teh primerih zaznala spremembe tako v ozadju kot ospredju. Ker med njima ne more diskriminirati, bi takšne spremembe mrežo silile k učenju irelevantnih opisnikov razlik. Učenja ne izvajamo na problemskih situacijah v kategorijah *PTZ* ter *Intermittent Object Motion*, uporabimo ju le za evalvacijo.

Podatkovna zbirka CDNET le za podmnožico slik video sekvenc vsebuje maske pravilnih segmentacij gibanja. Tako avtorji zbirke želijo preprečiti pretirano prilagajanje podatkom. Segmentacijske maske vsebujejo 5 številskih label, ki denotirajo posamezne regije v sliki. Te labele so označene z vrednostmi 0, 50, 85, 170 in 255, ter so v maski reprezentirane z različnimi sivinskimi nivoji. Labela 0 označuje ozadje, 50 močno senco, 85 regijo, katere segmentacija nas ne zanima, 170 območje nedefiniranega gibanja, ki pogosto obroblja premikajoče se objekte ter 255 relevantno gibanje. Vsaki video sekvenci v podatkovni zbirki je priložena tudi binarna maska regije v video sekvenci, za katero nas zanima segmentacija gibanja. Tej regiji pravimo *regija interesa* (angl. *region of interest*), saj detekcije zunaj te regije pri evalvaciji zavržemo ter jih tudi ne upoštevamo tekom učenja.

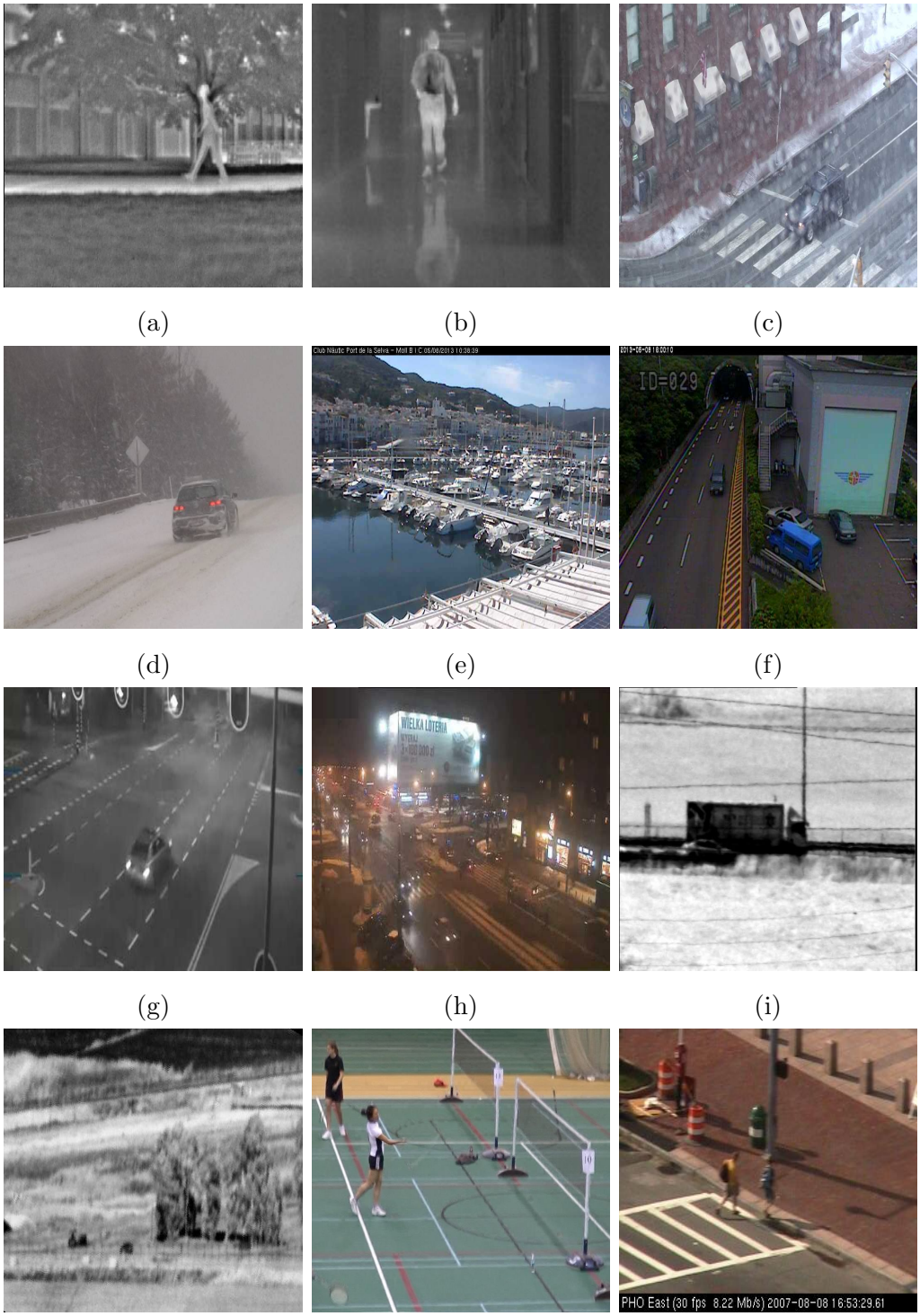
Wallflower

Podatkovna zbirka Wallflower [40] je namenjena testiranju algoritmov, ki iščejo razlike med ozadjem ter ospredjem video sekvenc. Vsebuje 7 kategorij, ki pokrivajo različna problematična področja. Ta področja zajemajo počasne in hitre spremembe v osvetlitvi okolja, gibanje objektov v ozadju ter objekte,

(a) (b) (c)

(d) (e) (f)

Slika 4.1: Primera kategorije *Baseline* (a), (b); primera kategorije *Dynamic background* (c), (d); primera kategorije *Shadow* (e), (f).

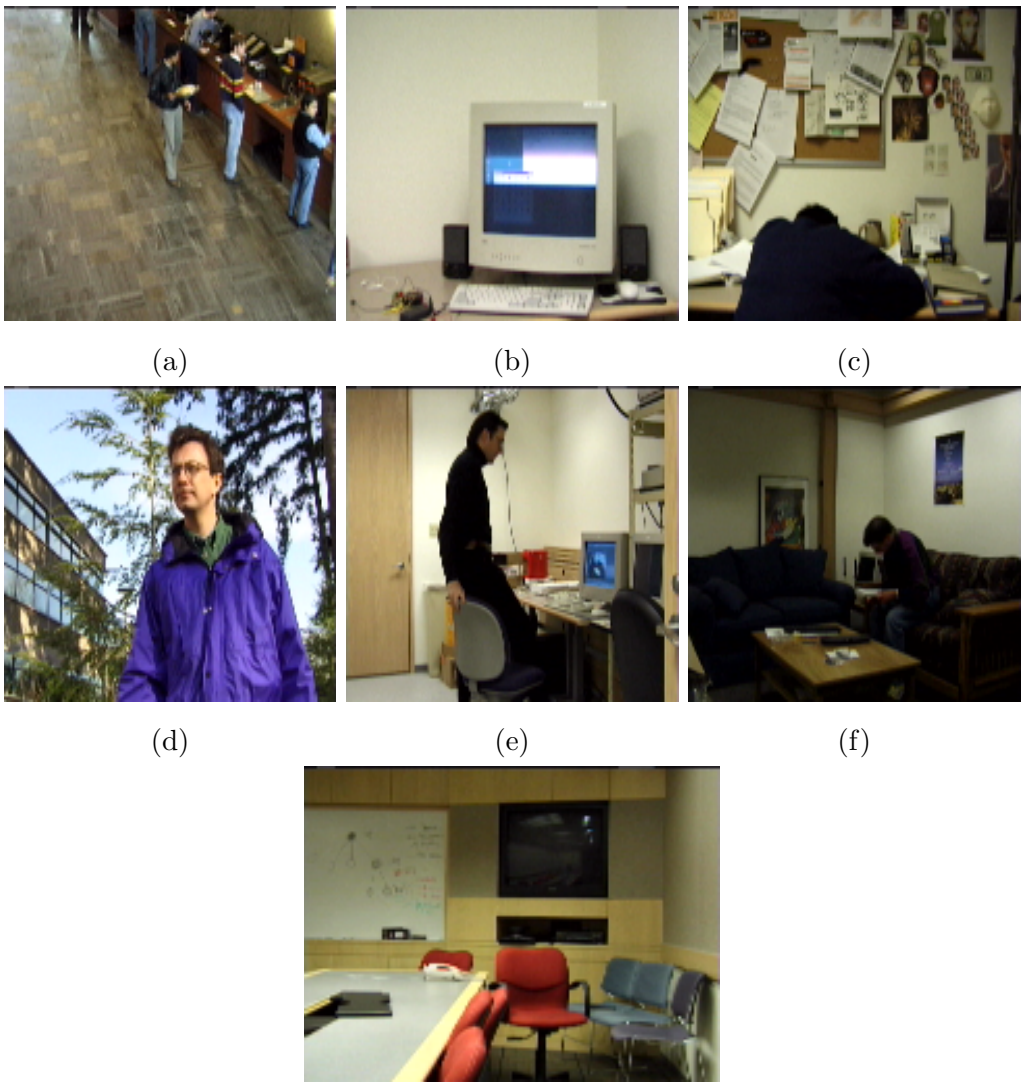


Slika 4.2: Primera kategorije *Thermal* (a), (b); primera kategorije *Bad Weather* (c), (d); primera kategorije *Low Frame-Rate* (e), (f); primera kategorije *Night Videos* (g), (h); primera kategorije *Turbulence* (i), (j); primera kategorije *Camera Jitter* (k), (l).

ki se nenadoma začnejo gibati po daljšem času mirovanja. Kategorije so poimenovane *Bootstrap*, *Camouflage*, *Foreground Aperture*, *Waving Trees*, *Light Switch*, *Time of Day* ter *Moved Object*. Binarna maska ospredja je na voljo za eno sliko iz posamezne video sekvence. V binarni maski je gibanje označeno z oznako, ki ima vrednost 255, ozadje z labelo 0. Primeri kategorij podatkovne baze so vidni na Sliki 4.3. Na tej podatkovni zbirki konvolucijske nevronske mreže ne učimo. Uporabljamo jo zgolj za evalvacijo. Pri evalvaciji uporabljamo vse kategorije zbirke Wallflower. Izjema je kategorija *Moved Object*, ki v binarni maski ospredja ne vsebuje nobenih slikovnih točk, ki bi pripadale ospredju. Tako ne moremo določiti nekaterih metrik uspešnosti delovanja algoritma, ki jih uporabljamo za evalvacijo.

SGM-RGBD

Podatkovna zbirka SGM-RGBD [16] [36] je prav tako namenjena testiranju in evalvaciji algoritmov detekcije relevantnih razlik med ozadjem sekvence ter posameznimi slikami sekvence. Vsebuje 33 video sekvenc, razdeljenih v 7 kategorij. Tako kot v ostalih podatkovnih bazah se kategorije razlikujejo glede na različne težavne okoliščine, ki so prisotne v sekvencah. Poleg običajnih RGB slik so priložene tudi globinske slike okolja, označene s črko D. Teh naš algoritem ne uporablja pri delovanju. Nekatere zajete video sekvence v podatkovni bazi vsebujejo primere sprememb osvetlitve okolja. V drugih sekvencah so objekti ospredja lahko po barvi ali po oddaljenosti od kamere podobni ozadju. V sekvencah so lahko prisotne močne sence, ki jih mečejo gibajoči objekti. Vsi našteti dejavniki otežujejo detekcijo relevantnega gibanja. Vsebovane kategorije se imenujejo *Illumination Changes*, *Color Camouflage*, *Depth Camouflage*, *Intermittent Motion*, *Out of Sensor Range*, *Shadows* ter *Bootstrapping*. Segmentacijske maske ospredja vsebujejo podobne oznake kot maske podatkovne zbirke CDNET. Tu so prisotne vse oznake 0, 85, 170 ter 255, z izjemo oznake 50. Oznaka 0 predstavlja ozadje, 85 regijo, za katero nas segmentacija ne zanima, 170 nedefinirano gibanje ter 255 relevantno gibanje. Maske so na voljo le za podmnožico slik v video sekvenci, saj s tem avtorji



(a) (b) (c)

(d) (e) (f)

(g)

Slika 4.3: Primer kategorije *Bootstrap* (a); primer kategorije *Camouflage* (b); primer kategorije *Foreground Aperture* (c); primer kategorije *Waving Trees* (d); primer kategorije *Light Switch* (e); primer kategorije *Time of Day* (f); primer kategorije *Moved Object* (g).

preprečujejo pretirano prilagajanje na učne primere. Primeri kategorij podatkovne baze so vidni na Sliki 4.4. Tudi to podatkovno bazo uporabljamo le za evalvacijo delovanja nevronske mreže in je ne uporabljamo za učenje. Nevronsko mrežo evalviramo na vseh kategorijah podatkovne zbirke. Izjema je kategorija *Illumination Changes* zaradi enakih razlogov kot kategorija *Moved Object* v podatkovni zbirki Wallflower.

4.2.1 Izbor in priprava učnih podatkov

Izbor učnih kategorij

Pri učenju konvolucijske nevronske mreže smo uporabili vse video sekvence v izbranih 9 kategorijah podatkovne zbirke CDNET. Kategorije, ki smo jih izbrali za učenje so *Baseline*, *Dynamic Background*, *Thermal*, *Turbulence*, *Low Framerate*, *Shadow*, *Camera Jitter*, *Bad Weather* in *Night Videos*. S takim izborom učnih primerov lahko mrežo naučimo detekcije relevantnih razlik. V učnih sekvencah se ozadje vizualno ne spreminja preveč, razen v primerih dinamičnega gibanja, kar nam omogoča, da lahko nevronsko mrežo učimo na tak način. Ker za učenje uporabljamo le eno sliko ozadja za celotno sekvenco, kategoriji *PTZ* ter *Intermittent Object Motion* ne uporabimo pri učenju, ker se ozadje tekom sekvence preveč spreminja. Uporabimo ju le pri evalvaciji.

Izbor učnih slik

Učne slike smo izbirali naključno. Za vsako sekvenco smo izbrali N naključnih slik, v katerih je poleg ozadja prisotno tudi ospredje. To smo storili zato, ker je število slikovnih točk, ki predstavljajo ozadje, veliko večje od števila točk, ki predstavljajo ospredje. Ker so vizualne informacije v video sekvencah pogosto redundantne v zaporednih slikah, je število izbranih slik mnogo manjše od celotnega števila slik v video sekvenci. V povprečju vsebuje posamezna video sekvenca 3000 slik, najmanj 600 ter največ 7400. Pri učenju smo uporabili N, ki je enak 300.

(a) (b) (c)

(d) (e) (f)

(g)

Slika 4.4: Primer kategorije *Illumination Changes* (*a*); primer kategorije *Color Camouflage* (*b*); primer kategorije *Depth Camouflage* (*c*); primer kategorije *Intermittent Motion* (*d*); primer kategorije *Out of Sensor Range* (*e*); primer kategorije *Shadows* (*f*); primer kategorije *Bootstrapping* (*g*).

Določanje slike ozadja

Za vsako učno video sekvenco smo generirali eno sliko ozadja. V večini primerov smo v množici slik poiskali referenčno sliko, ki ne vsebuje objektov ospredja. Ta slika je nato predstavljala model ozadja za celotno video sekvenco. V nekaterih video sekvencah referenčne slike brez elementov ospredja ni bilo mogoče najti. V takih primerih smo ozadje generirali z lokalno mediano 100 slik, kjer vsako deseto sliko izpustimo. Slike, ki smo jih uporabili za določanje mediane, se nahajajo v tistem delu sekvence, ki vsebuje najmanj gibanja. Taka mediana se je izkazala za dober približek ozadja.

Pred-procesiranje ter umetno razširjanje učnih podatkov

Vhodni podatki v nevronsko mrežo *SubNet* so pari naključno izbranih slik iz video sekvence ter slika ozadja te sekvence. Ker so prvi nivoji mreže *SubNet* enaki nivojem mreže VGG-16, podatke centriramo glede na povprečne vrednosti kanalov RGB slik, s katerimi je bila učena mreža VGG-16. Te vrednosti so dostopne v meta podatkih modela mreže VGG-16, ki je dostopen na strani Matconvnet [28]. Ker so video sekvence v različnih velikostnih formatih, v vsaki vhodni sliki izberemo regijo veliko 100×100 slikovnih točk ter jo uporabimo za učenje. Enako regijo izberemo tudi v ozadju. Tako ohranimo velikosti ter razmerja vizualnih informacij, ki se nahajajo v sekvencah ter ne izgubljamo informacij z zmanjševanjem ali rezanjem slik na manjše velikosti. Središče izbrane regije je vedno v območju interesa video sekvence. Podatke tekom učenja umetno razširjamo tako, da jih naključno preslikamo preko vertikalne osi.

Izenačevanje razredov

Ker je slikovnih točk, ki pripadajo ospredju v slikah veliko manj kot točk, ki pripadajo ozadju, smo funkcijo napake utežili glede na razmerje med številom teh točk. Napačna segmentacija ospredja je tako povzročila večjo vrednost napake kot napačna klasifikacija ozadja. S tem smo dosegli, da je mreža

tekom učenja hitreje konvergirala h kvalitetnejši rešitvi problema.

Parametri učenja

Za učenje smo uporabili napredni optimizacijski algoritem Adam [24]. Prednost uporabe tega algoritma je v tem, da samodejno tekom učenja adaptira velikost učne hitrosti. Tako nam olajša določanje začetne učne vrednosti, saj je zmožen hitro popraviti učne vrednosti, ki so preveč optimistične ali preveč konzervativne. Začetno učno hitrost smo nastavili na 0.001. Vrednost *L2* regularizacije uteži smo nastavili na 0.00005. Skupina slik (angl. *batch size*), s katero smo naenkrat učili nevronsko mrežo ter jo uporabili za optimizacijo uteži, je vsebovala 30 naključno izbranih slik iz celotne učne množice podatkov. Vrednosti teh parametrov smo določili eksperimentalno s preizkušanjem različnih kombinacij. Izbrali smo tisto kombinacijo parametrov, s katerimi je mreža dosegala najboljše rezultate na manjši učni množici. Nevronsko mrežo smo učili 400 epoh. V vsaki epohi mreža enkrat preide čez celotno množico učnih podatkov.

4.3 Protokol evalvacije

4.3.1 Dinamično prilagajanje ozadja

Delovanje naše konvolucijske nevronske mreže je pogojeno z določanjem referenčne slike ozadja, ki služi kot model ozadja tekom evalvacije. Celotno sekvenco bi lahko evalvirali z enim samim ozadjem, ki ga določimo na enak način kot pri učenju. Problem lahko nastane v okoljih, ki se tekom časa spreminjajo. Spremembe v ozadju, kot so počasna sprememba osvetlitve ali akumulacija manjših vizualnih sprememb, lahko vodijo v neželene aktivacije. Da bi v določenih primerih take neželene posledice sprememb zamejili, mreži omogočimo, da tekom izvajanja prilagaja model ozadja. To storimo tako, da med evalvacijo video sekvence, na vsakih $N + 1$ zaporednih slik generiramo novo sliko ozadja. Nova slika ozadja je kombinacija generiranih slik ozadja

za vsako izmed N zaporednih slik. Posamezno ozadje pridobimo po predpisu

$$M_n = dilate(M_n, D), \tag{4.1}$$

$$O_n = RM_n + I_n(1 - M_n), \tag{4.2}$$

$$O_n = w_n O_n + (1 - w_n)R, \tag{4.3}$$

$$w_n(n, N) = \frac{N - n + 1}{N}, N >= 1, 1 <= n <= N, \tag{4.4}$$

kjer je O_n ozadje posamezne slike I_n ter R predstavlja referenčno sliko. Masko gibanja M_n, ki jo za sliko I_n ter referenčno sliko R generira mreža, razširimo z operacijo *dilate* s krožnim elementom D, katerega polmer je velik 8 slikovnih točk. Ta operacija odebeli binarne elemente v vhodni maski, ter jih vizualno naredi večje [21]. To storimo zato, da zajamemo tudi nekatere anomalije okoli objektov ospredja, ki nastajajo zaradi gibanja. Ozadje O_n je tako kombinacija dveh slik. Iz referenčne slike vzamemo slikovne točke, ki so bile v sliki I_n označene kot ospredje v razširjeni maski gibanja M_n. Iz slike I_n vzamemo vse ostale slikovne točke, ki niso del gibanja $(1 - M_n)$. Nato posamezno sliko ozadja O_n obtežimo z utežjo w_n ter prištejemo referenčno sliko obteženo z vrednostjo $(1 - w_n)$. Vrednosti w_n zavzemajo interval $[\frac{1}{N}, 1]$. To storimo zato, ker detekcije v dinamičnih okoljih tekom časa, z naraščanjem vrednosti n proti N, postajajo nestabilne. S tem časovno obtežimo detekcije ter izboljšamo delovanje evalvacijskega algoritma v dinamičnih okoljih . Ko dosežemo zaporedno sliko $N + 1$, zamenjamo referenčno sliko R z mediano N generiranih ozadji. Nato postopek ponovimo. Novo referenčno sliko R uporabljamo za evalvacijo dokler je ponovno ne zamenjamo.

Za vsako sliko, ki je na voljo v podatkovnih zbirkah, generiramo segmentacijsko masko gibanja. Te maske nato primerjamo s pravilnimi segmentacijami gibanja. V primeru podatkovne zbirke CDNET nimamo dostopa do vseh pravilnih mask segmentacij gibanja za vse slike v video sekvencah. Zato rezultate naše metode naložimo na strežnik strani CDNET [17]. Ta evalvacijo opravi na vseh slikah ter metodo ovrednoti glede na dosežene rezultate.

4.3.2 Performančne mere

Rezultat naše konvolucijske nevronske mreže je tridimenzionalni tenzor velikosti $W \times H \times C$. Prvi dve dimenziji sta enaki širini ter višini vhodne slike mreže. Tretja dimenzija je po velikosti enaka $C = 2$. Lahko si ga predstavljamo kot dvokanalno sliko, v kateri vrednosti slikovnih točk prvega kanala podajajo verjetnost, da istoležne slikovne točke v vhodni sliki pripadajo ozadju. V drugem kanalu se nahajajo verjetnosti, da istoležne slikovne točke v vhodni sliki pripadajo osredju.

Da pridobimo binarno masko ospredja potrebujemo le drugi kanal rezultata. To je 2D tenzor P_f. Nato vrednosti v P_f binariziramo glede na fiksno mejo M. Vse verjetnosti, ki so v $P_f >= M$ postanejo enake 1, ostale, manjše od M, 0. Za določanje binarne maske smo pri evalvaciji uporabili $M = 0.9$. Če bi izbrali manjšo vrednost M, bi lahko gibanje v ozadjih bolj dinamičnih sekvenc povzročalo neželene detekcije. Mreža lahko takšno gibanje detektira dalj časa. V takšnem primeru tvegamo, da se določen del modela ozadja ne posodobi. To bi lahko povzročalo še nadaljnje lažne detekcije, zato želimo poročati le tisto gibanje, za katerega je mreža zelo gotova, da pripada dejanskemu gibanju objektov ospredja. Pridobljeno masko napovedi mreže primerjamo s pravilno masko napovedi za isto sliko. Število pravilno napovedanih slikovnih točk ospredja označimo kot *TP* (angl. *true positives*). Z oznako *TN* (anlg. *true negatives*) označimo število slikovnih točk, ki jih mreža pravilno kategorizira kot ozadje. Številu slikovnih točk, ki jih mreža napačno napove kot ozadje pravimo *FN* (angl. *false negatives*) ter obratno, številu napačnih napovedi gibanja *FP* (angl. false positives). Te štiri vrednosti, *TP*, *TN*, *FN* ter *FP*, služijo kot podlaga za izračun nadaljnjih kriterijev za ocenjevanje delovanja mreže.

Priklic

Priklic (angl. *recall*) je mera, ki nam sporoči kolikšen delež vseh slikovnih točk ospredja mreža pravilno označi kot ospredje. Definirana je z enačbo

$$Re = \frac{TP}{TP + FN}. \tag{4.5}$$

Natančnost

Natančnost (angl. *precision*) nam sporoči kolikšen delež slikovnih točk, klasificiranih kot ospredje, je pravilen. Definirana je z enačbo

$$Pr = \frac{TP}{TP + FP}. \tag{4.6}$$

Specifičnost

Specifičnost (angl. *specificity*) je mera, s katero določimo delež pravilnih klasifikacij slikovnih točk ozadja. Zapišemo jo z enačbo

$$Sp = \frac{TN}{TN + FP}. \tag{4.7}$$

FPR

Z mero *FPR* (angl. *false positive rate*) določimo kolikšen delež ozadja slike mreža napačno klasificira kot ospredje. Ta delež zapišemo z enačbo

$$FPR = \frac{FP}{FP + TN}. \tag{4.8}$$

FNR

Mera *FNR* (angl. *false negative rate*) določa kolikšen delež ospredja slike mreža napačno klasificira kot ozadje. Ta delež določimo z enačbo

$$FNR = \frac{FN}{FN + TP}. \tag{4.9}$$

PWC

Mera PWC (angl. *percentage of wrong classifications*) določa kolikšen procent vseh napovedi, tako ozadja kot ospredja, je napačen. To izračunamo z enačbo

$$PWC = 100\frac{FN + FP}{TP + FN + FP + TN}. \tag{4.10}$$

F-mera

F-mera predstavlja harmonično sredino mer natančnosti ter priklica. Kjer natančnost in preciznost opisujeta različni lastnosti klasifikacije, F-mera združi ta opisa ter lahko služi kot samostojna enota za ocenjevanje delovanja klasifikatorja. Definirana je z enačbo

$$F = 2\frac{PrRe}{Pr + Re}. \tag{4.11}$$

4.4 Analiza rezultatov

Mrežo *SubNet* smo primerjali z metodami, ki so objavljene na spletni strani podatkovne zbirke CDNET [17]. Upoštevali smo mere uspešnosti, po katerih so algoritmi ocenjeni na spletni strani. Metodo DeepBS [9], ki smo jo predstavili v Poglavju 1.1, so avtorji evalvirali tudi na podatkovni zbirki Wallflower. To nam je omogočilo, da smo našo mrežo primerjali z metodo DeebBS na tej množici. Primerjavo na tej množici smo izvedli le za *F mero*, saj s to mero v članku [9] vrednotijo uspešnost svoje metode. Rezultate naše metode na podatkovni zbirki *SGM-RGBD* smo predstavili samostojno, brez primerjave z drugimi metodami. Dodali smo tudi celotne evalvacijske rezultate mreže *SubNet* na podatkovnih zbirkah CDNET, SGM-RGBD ter Wallflower za vsako kategorijo, ki jo podatkovna zbirka vsebuje.

Po kvantitativni analizi rezultatov smo analizirali delovanje mreže na specifičnih kategorijah, ki predstavljajo različna problemska področja detekcije

gibanja. Predstavili smo nekaj domen, na katerih mreža deluje dobro ter domene, na katerih deluje slabo.

4.4.1 Kvantitativna analiza

Mreža *SubNet* na celotni podatkovni zbirki CDNET dosega rezultate predstavljene v Tabeli 4.1. Mere uspešnosti so predstavljene za vsako kategorijo podatkovne zbirke.

	SubNet						
CDNET	Re	Sp	FPR	FNR	PWC	Pr	F-m
Bad Weather	0.88	0.99	0.0005	0.11	0.2	0.96	0.92
Low Framerate	0.86	0.99	0.0003	0.13	0.14	0.77	0.78
Night Videos	0.86	0.99	0.002	0.13	0.69	0.93	0.89
PTZ	0.92	0.57	0.42	0.07	42.09	0.01	0.03
Turbulence	0.85	0.98	0.001	0.14	0.13	0.78	0.78
Baseline	0.96	0.99	0.0001	0.03	0.09	0.99	0.98
Dynamic Background	0.95	0.99	0.0004	0.04	0.08	0.88	0.91
Camera Jitter	0.96	0.99	0.0009	0.03	0.21	0.98	0.97
Intermittent Object Motion	0.79	0.87	0.12	0.20	11.64	0.68	0.65
Shadow	0.97	0.99	0.0008	0.02	0.16	0.97	0.97
Thermal	0.94	0.99	0.0007	0.05	0.4	0.98	0.96

Tabela 4.1: Rezultati mreže *SubNet* na podatkovni zbirki CDNET. V skrajno levem stolpcu se nahajajo imena posameznih kategorij, ki jim sledijo vrednosti mer uspešnosti.

Mreža *SubNet* je dosegla rang 11.18. Glede na dosežene rezultate se je uvrstila na 8. mesto izmed 46 objavljenih algoritmov. Range enajstih najboljših metod na podatkovni zbirki CDNET smo predstavili v Tabeli 4.2. Tako lahko uspešnost naše metode bolje umestimo med druge evalvirane metode. Prav tako smo izbrali le 11 najboljših metod, izmed celotnih 46, zaradi lažje preglednosti.

CDNET	Rang
FgSegNet_v2 [8]	1.36
FgSegNet_S [27]	1.91
FgSegNet [26]	2.73
BSPVGAN [48]	4.00
BSGAN [47]	5.27
Cascade CNN [44]	6.91
IUTIS-5 [10]	10.27
SubNet	11.18
SemanticBGS [14]	11.55
IUTIS-3 [10]	13.91
DeepBS [9]	14.09

Tabela 4.2: Rangi enajstih najboljših metod na podatkovni zbirki CDNET. Rangi so določeni s povprečjem mer uspešnosti algoritmov preko vseh podatkovnih zbirk.

V Tabeli 4.3 ter Tabeli 4.4 lahko vidimo range enajstih najboljših metod za posamezno kategorijo podatkovne zbirke CDNET. Opazimo lahko, da naša metoda v veliki večini kategorij deluje dobro. V kategorijah *PTZ* ter *Intermitten Object Motion* deluje izrazito slabše. Razlog za tem je predvsem ta, da metode nismo razvijali za delovanje v takšnih situacijah.

CDNET	BW	LF	NV	PTZ	T	B	DB
FgSegNet_v2 [8]	1.71	1.29	1.29	1.29	1.57	1.43	2.71
FgSegNet_S [27]	2.14	2.29	2.14	2.14	3.29	1.57	1.57
FgSegNet [26]	3.43	3.43	2.57	2.57	2.29	3.29	1.71
BSPVGAN [48]	4.43	4.71	4.71	4.00	5.29	4.43	4.00
BSGAN [47]	7.14	6.14	5.71	5.00	8.14	6.00	5.00
Cascade CNN [44]	8.71	7.43	5.71	6.00	9.43	7.00	6.00
IUTIS-5 [10]	15.57	12.57	21.29	17.86	14.43	10.14	11.71
SubNet	6.86	6.29	6.00	34.14	14.57	5.71	8.71
SemanticBGS [14]	14.86	10.71	16.86	13.29	27.57	15.43	8.00
IUTIS-3 [10]	21.14	15.29	25.71	19.71	14.57	12.29	13.14
DeepBS [9]	11.43	24.43	16.86	26.57	10.00	11.00	17.14

Tabela 4.3: Rangi enajstih najboljših metod na podatkovni zbirki CDNET za kategorije *Bad Weather*, *Low Framerate*, *Night Videos*, *Turbulence*, *Baseline* ter *Dynamic Background*.

CDNET	CJ	IOM	S	Th
FgSegNet_v2 [8]	1.00	1.29	1.29	1.57
FgSegNet_S [27]	2.43	2.29	2.43	2.00
FgSegNet [26]	2.57	2.43	2.29	3.29
BSPVGAN [48]	4.29	4.86	4.29	5.14
BSGAN [47]	5.14	4.86	5.71	15.43
Cascade CNN [44]	6.43	14.29	7.00	17.86
IUTIS-5 [10]	15.57	14.43	12.14	16.29
SubNet	6.14	28.71	5.29	5.57
SemanticBGS [14]	12.57	10.86	8.57	20.57
IUTIS-3 [10]	18.71	18.29	17.57	18.43
DeepBS [9]	8.86	21.57	9.86	23.86

Tabela 4.4: Rangi enajstih najboljših metod na podatkovni zbirki CDNET za kategorije *Camera Jitter*, *Intermittent Object Motion*, *Shadow* ter *Thermal*.

Pri podatkovni zbirki Wallflower lahko v tabeli Tabela 4.5 vidimo, da naša

metoda prekaša metodo DeepBS v vseh kategorijah. Izjema je le kategorija *Camouflage*.

	SubNet	*DeepBS*
Wallflower	F-m	F-m
Bootstrap	**0.8**	0.75
Camouflage	0.97	**0.99**
Foreground Aperture	**0.95**	0.66
Light Switch	**0.83**	0.61
Time Of Day	**0.63**	0.55
Waving Trees	**0.97**	0.96

Tabela 4.5: Rezultati metod *SubNet* ter DeepBS na podatkovni zbirki Wallflower. V skrajno levem stolpcu se nahajajo imena posameznih kategorij, ki jim sledi F-mera za posamezno kategorijo ter posamezno metodo.

V Tabeli 4.6 ter Tabeli 4.7 so prikazani rezultati naše mreže za podatkovni zbirki *SGM-RGBD* ter *Wallflower* za vsako posamezno kategorijo, ki jo vsebujeta.

	SubNet						
SGM-RGBD	Re	Sp	FPR	FNR	PWC	Pr	F-m
Bootstrapping	0.76	0.99	0.009	0.23	3.78	0.9	0.82
Color Camouflage	0.42	0.99	0.002	0.57	15.24	0.96	0.50
Depth Camouflage	0.76	0.99	0.001	0.23	2.13	0.96	0.80
Intermittent Motion	0.71	0.96	0.03	0.28	3.84	0.69	0.67
Out Of Range	0.92	0.99	0.0006	0.07	0.38	0.98	0.95
Shadows	0.56	0.99	0.003	0.43	5.33	0.94	0.67

Tabela 4.6: Rezultati mreže *SubNet* na podatkovni zbirki SGM-RGBD. V skrajno levem stolpcu se nahajajo imena posameznih kategorij, ki jim sledijo vrednosti mer uspešnosti.

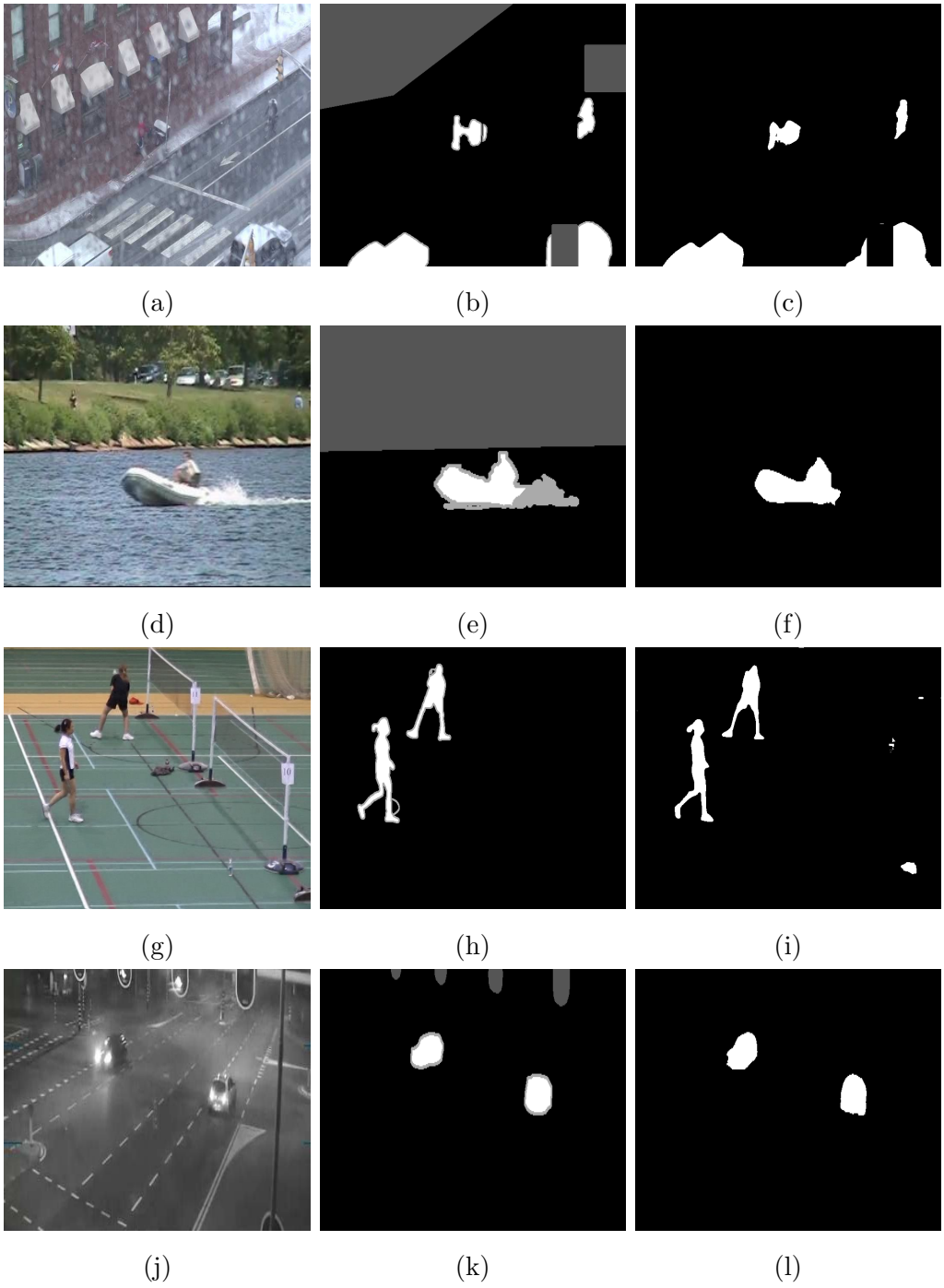
	SubNet						
Wallflower	Re	Sp	FPR	FNR	PWC	Pr	F-m
Bootstrap	0.69	0.99	0.005	0.3	4.38	0.94	0.80
Camouflage	0.96	0.99	0.008	0.03	2.16	0.99	0.97
Foreground Aperture	0.93	0.99	0.006	0.006	2.22	0.98	0.95
Light Switch	0.90	0.95	0.04	0.09	3.56	0.77	0.83
Time Of Day	0.48	0.99	0.002	0.51	3.66	0.94	0.63
Waving Trees	0.94	0.99	0.001	0.05	1.16	0.99	0.97

Tabela 4.7: Rezultati mreže *SubNet* na podatkovni zbirki Wallflower. V skrajno levem stolpcu se nahajajo imena posameznih kategorij, ki jim sledijo vrednosti mer uspešnosti.

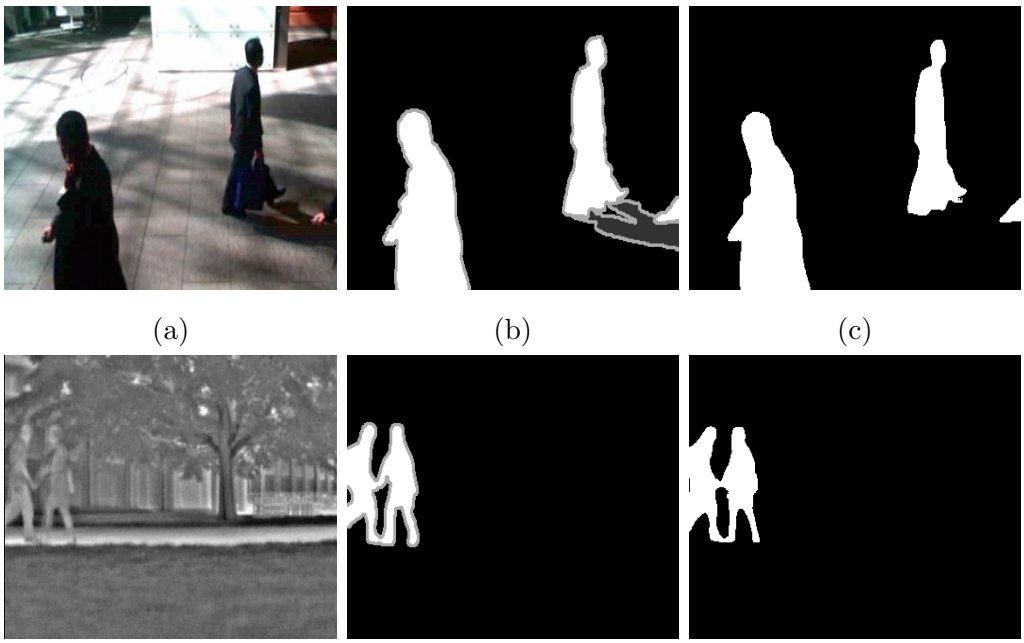
4.4.2 Kvalitativna analiza

Pridobljeni rezultati prikazujejo tako močne kot šibke točke naše metode. Navkljub različnim težavnim primerom kot so dinamično ozadje, nočni posnetki, posnetki s termalno kamero, tresenjem kamere, slabim vremenom ali močnimi sencami, naša mreža uspešno zaznava ter detektira relevantne razlike med ozadjem ter posamezno sliko sekvence. To je razvidno v Tabeli 4.1. Na Sliki 4.5 ter Sliki 4.6 so prikazani primeri dobrih segmentacij relevantnega gibanja v sekvencah podatkovne zbirke CDNET. Za podatkovni zbirki Wallflower ter SGM-RGBD so primeri zadovoljivih segmentacij prikazani na Sliki 4.7. Zanimiv je tudi površinski vpogled v delovanje nevronske mreže. Na Sliki 4.8 so prikazani izhodi nivoja razlike ter štirih konvolucijskih nivojev, ki mu sledijo. Ker so izhodi večdimenzionalni tenzorji, smo jih za lažji prikaz sešteli po tretji dimenziji ter vrednosti normalizirali na interval $[0, 1]$. Rezultat je ena črno-bela slika za posamezni nivo, kot je razvidno na Sliki 4.8. Tako lahko opazujemo na katerih delih vhodne slike mreža tvori najmočnejše aktivacije. S pomočjo takega vpogleda lahko ugotovimo, zakaj na nekaterih kategorijah naša mreža ne deluje dobro.

Dinamično spreminjanje modela ozadja omogoča naši mreži, da na določenih kategorijah dosega boljše rezultate kot bi jih sicer. Kategorija *Time Of Day*



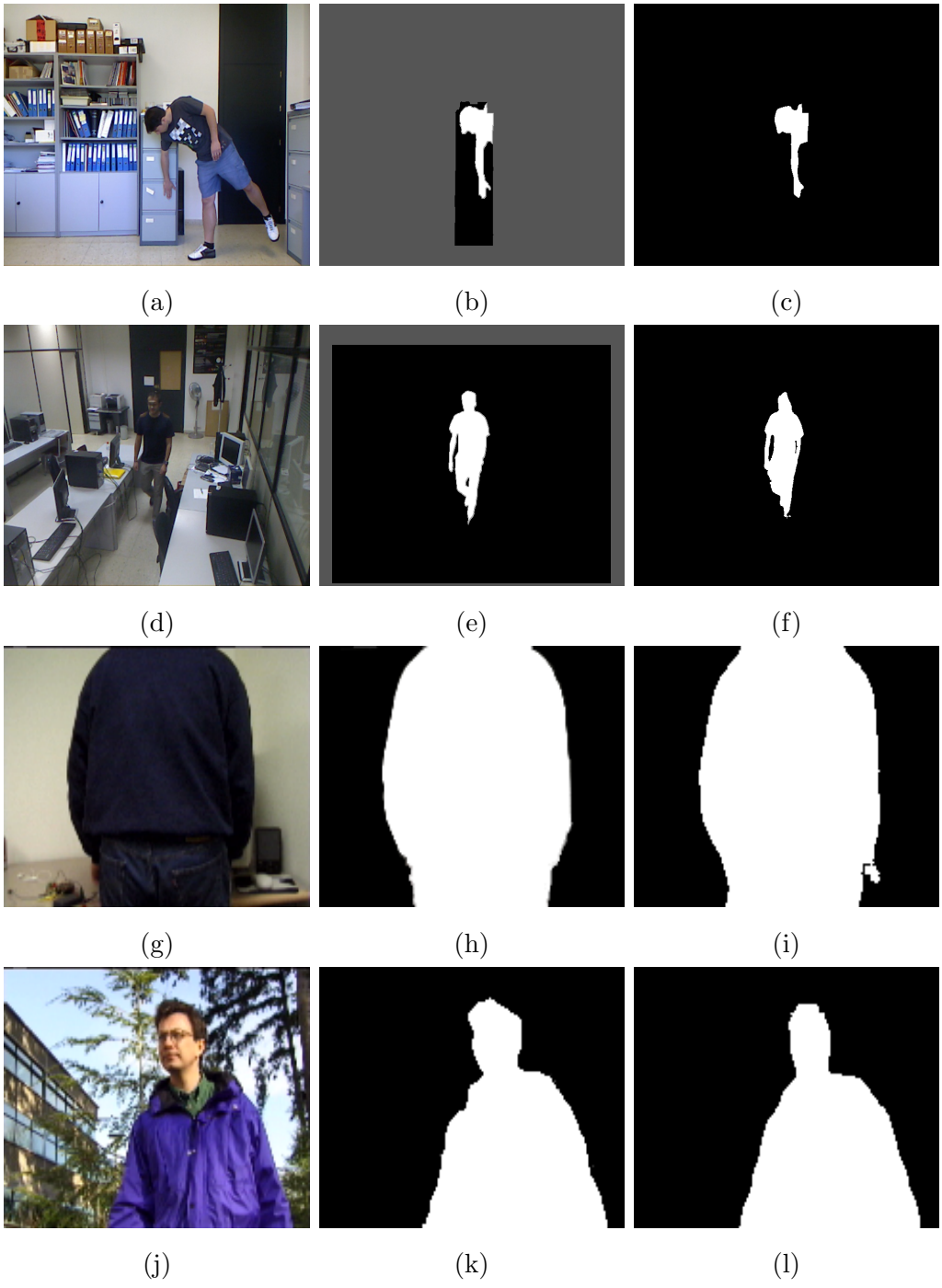
Slika 4.5: Prikaz vhodnih slik, pravilnih segmentacij ter segmentacij naše mreže za primere kategorij *Bad Weather*, *Dynamic Background*, *Camera Jitter* ter *Night Videos*. V prvem stolpcu so prikazane vhodne slike, v drugem pravilne segmentacije ter v tretjem izhodi naše nevronske mreže. Vidimo, da so v teh primerih segmentacijske maske zadovoljivo natančne ter dajejo primerne rezultate kljub prisotnosti dinamičnih elementov, kot so vremenski pojavi ali tresenje kamere.



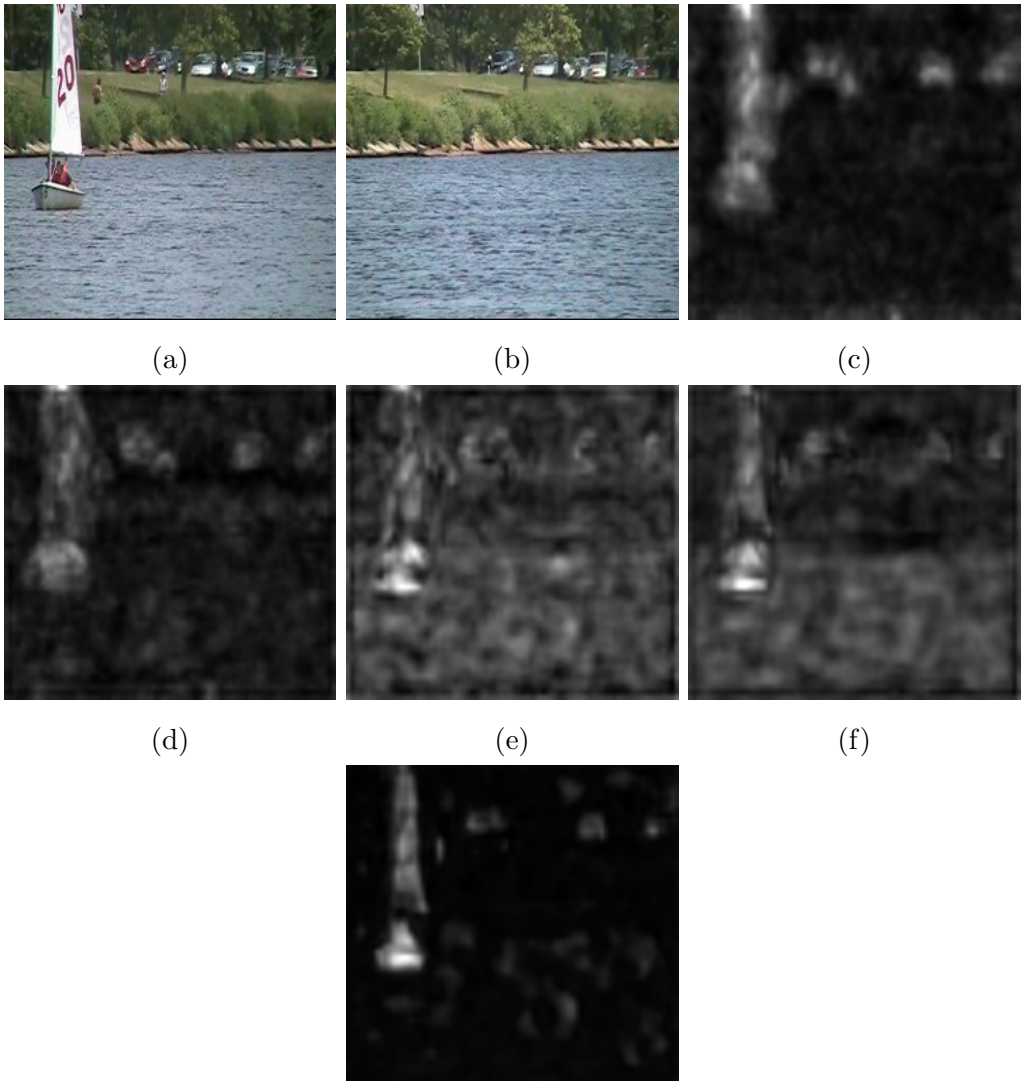
(a) (b) (c)

(d) (e) (f)

Slika 4.6: Prikaz vhodnih slik, pravilnih segmentacij ter segmentacij naše mreže za primera kategorij *Shadow* ter *Thermal*. V prvem stolpcu so prikazane vhodne slike, v drugem pravilne segmentacije ter v tretjem izhodi naše nevronske mreže. Vidimo, da so v teh primerih segmentacijske maske zadovoljivo natančne ter dajejo primerne rezultate kljub prisotnosti premikajočih se senc ter so neodvisne od barvnega spektra, v katerem so objekti predstavljeni.



Slika 4.7: Prikaz vhodnih slik, pravilnih segmentacij ter segmentacij naše mreže za primera kategorij *Depth Camouflage* ter *Out Of Range* podatkovne zbirke SGM-RGBD ter kategoriji *Camouflage* in *Waving Trees* podatkovne zbirke Wallflower. V prvem stolpcu so prikazane vhodne slike, v drugem pravilne segmentacije ter v tretjem izhodi naše nevronske mreže.



(a) (b) (c)

(d) (e) (f)

(g)

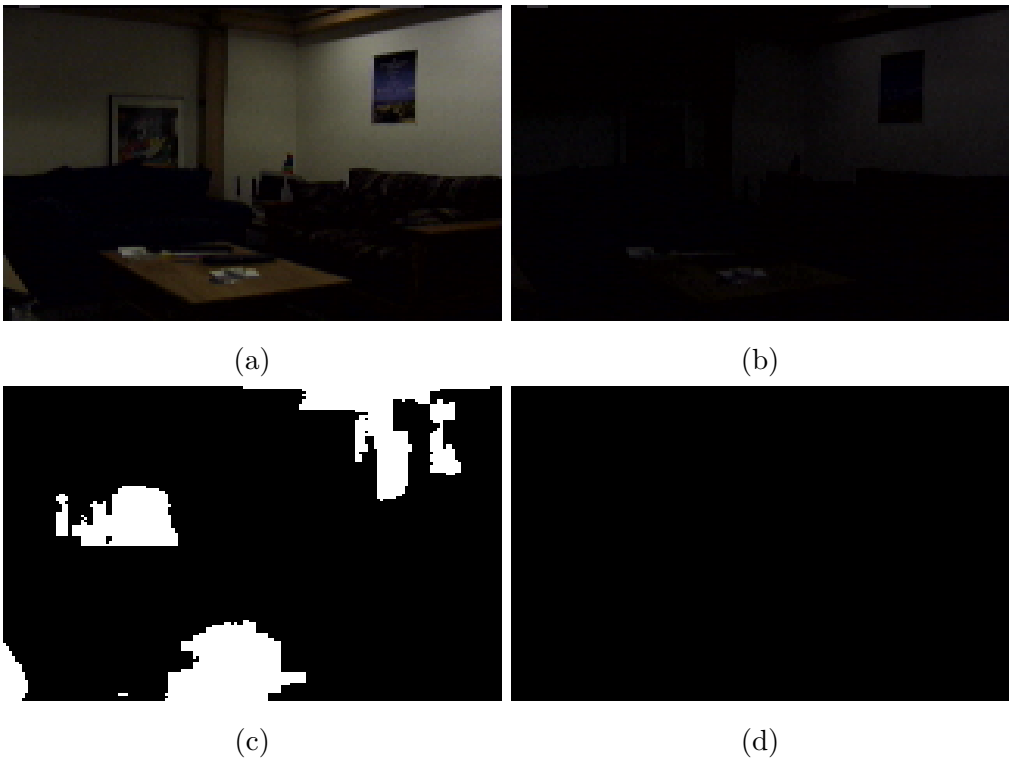
Slika 4.8: Sliki (a) ter (b) prikazujeta primer vhodne slike ter slike ozadja, ki vstopata v prvi del mreže *SubNet*. Slika (c) prikazuje rezultat nivoja absolutne razlike, ki med seboj odšteje zbirke značilk zadnjega siamskega konvolucijskega nivoja prvega dela mreže. Sledijo slike (d), (e), (f) ter (g), ki vsebujejo prikaz aktivacij štirih konvolucijskih nivojev, ki sledijo nivoju razlike. Iz prikazanega lahko sklepamo, da mreža postopoma vizualnim informacijam dodaja definicijo ter ločuje relevantne vizualne segmente od šuma. To so le površinska sklepanja, kaj natančno nevronska mreža počne, je težko intuitivno ugotoviti.

podatkovne zbirke Wallflower je primer take kategorije. Vsebuje video sekvenco, v kateri se počasi ter postopoma spreminja nivo osvetlitve opazovanega okolja. Če bi evalvacijo izvajali s statičnim modelom ozadja, bi sprememba svetlobe povzročala mnogo lažnih detekcij. Ker ozadje mreža popravlja glede na lastne detekcije tekom izvajanja, lahko postopne spremembe v osvetlitvi uspešno integrira v novo sliko ozadja. Na Sliki 4.9 lahko vidimo primerjavo med rezultatom evalvacije s statičnim modelom ozadja ter ozadjem, ki ga mreža dinamično spreminja.

Problematična področja delovanja

V podatkovni zbirki CDNET je kategorija, na kateri mreža *SubNet* najslabše deluje, kategorija *PTZ*. Hitro spreminjajoča se ozadja zaradi premikanja kamere so glavni razlog, zaradi katerega mreža na tej kategoriji ne deluje dobro. Kljub dinamičnemu prilagajanju ozadja, tako hitrih sprememb mreža ni sposobna modelirati. Posledica tega so deformirane slike ozadja, ki tvorijo mnogo nezaželenih detekcij. Primer teh lahko vidimo na Sliki 4.10. Kljub temu, da lahko ločimo spodnji obris avtomobila je poleg tega segmentirana še velika količina regij, ki jih je mreža zaradi nagle menjave ozadja zaznala kot gibanje. Število napačnih detekcij lahko zmanjšamo z nastavitvijo parametra N pri dinamičnem modeliranju ozadja na čim manjšo vrednost. Kljub temu se večine lažnih detekcij ne moremo znebiti. Ob razvijanju konvolucijske nevronske mreže se nismo posvetili reševanju problematike te kategorije, a smo mrežo vseeno evalvirali na njej.

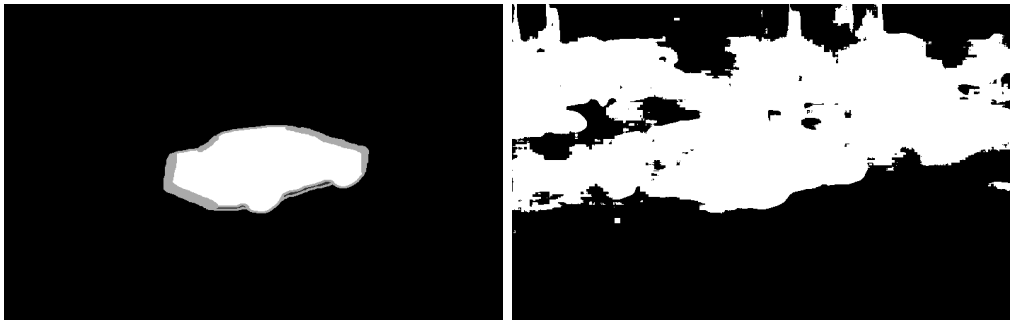
Kategorija v podatkovni zbirki SGM-RGBD, ki mreži povzroča največ težav, je kategorija *Color Camouflage*. Opazimo lahko, da mreža dosega visoko natančnost a zelo majhen priklic na primerih te kategorije. Tu je glavna diskriminativna lastnost gibanja razlika v globini med objektom ospredja ter ozadjem. Naša mreža deluje le na podlagi vizualnih informacij ter tako na teh primerih deluje slabo. Na Sliki 4.11 lahko vidimo primer, na katerem nevronska mreža dosega slabe rezultate. Opazimo lahko, da zaradi vizualne podobnosti bele škatle pred belim ozadjem mreža v tem delu slike ne more



(a) (b)

(c) (d)

Slika 4.9: Slika (*a*) vsebuje primer vhodne slike za sekvenco *Time Of Day*. Slika (*b*) vsebuje začetno referenčno sliko, ki ni osvetljena. Sprememba v osvetlitvi iz referenčne slike na vhodno je znatna, a se zgodi postopoma. Na sliki (*c*) vidimo rezultat segmentacije brez dinamičnega modeliranja ozadja, na sliki (*d)* je prikazan rezultat mreže, kjer ozadje dinamično modeliramo. Vidimo lahko, da v primeru, kjer ozadja ne popravljamo tekom evalvacije, zaradi spremembe v svetlobi, mreža te spremembe detektira kot gibanje. S popravljanjem ozadja lahko te počasne, irelevantne spremembe vključimo v sliko ozadja.



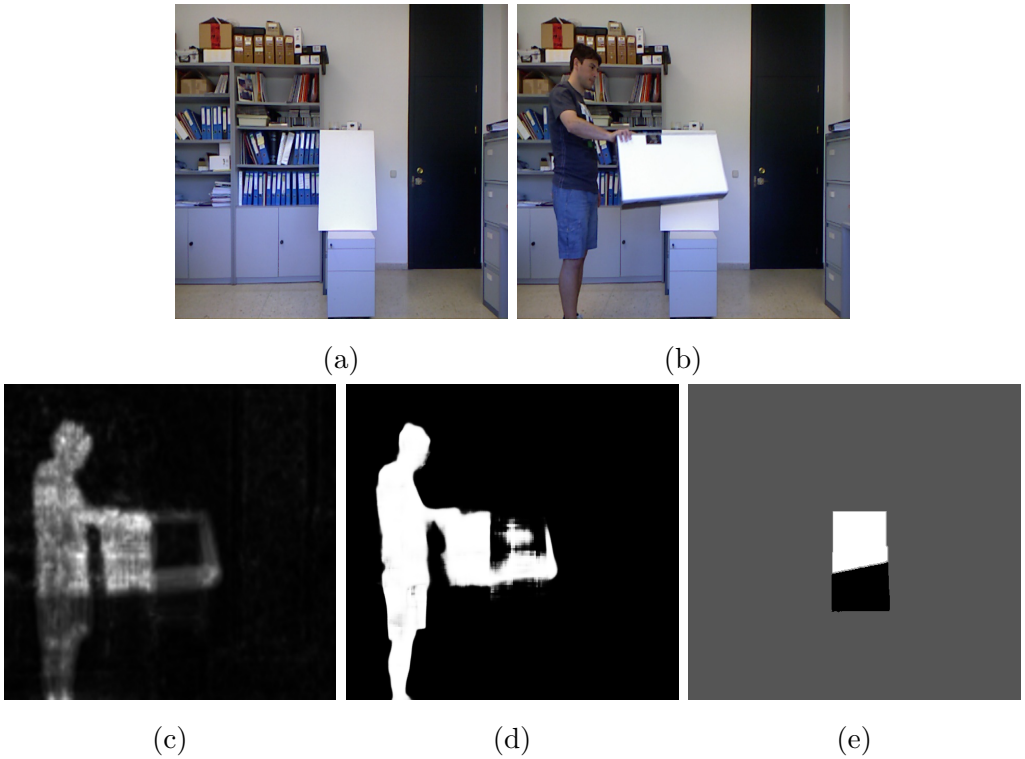
(a) (b)

Slika 4.10: Slika (a) predstavlja želen primer segmentacije vhodnih podatkov. Na sliki (b) vidimo generiran rezultat naše nevronske mreže.

detektirati relevantnih razlik. Ker sta obe površini opisani z enakimi opisniki, ki se nahajajo na enakem mestu, se vrednosti teh med seboj odštejejo. Tako na tem mestu slabo detektiramo razlike.

V podatkovni zbirki SGM-RGBD lahko opazimo tudi, da mreža v večini kategorij dosega visoko natančnost. Ta je enaka ali višja kot 0.9. Izjema je kategorija *Intermittent Object Motion*, kjer mreža dosega nizko natančnost v vrednosti 0.67. Tu na delovanje mreže zelo vpliva določanje začetne slike ozadja. V primeru na Sliki 4.12 lahko vidimo, da naš algoritem detektira spremembo tudi na mestu, kjer je bila pred premikom pozicionirana nepremična škatla. Ta je bila del referenčne slike, ki opisuje ozadje. Ker naša mreža zaradi nivoja absolutne razlike enakovredno obravnava spremembe tako v vhodni sliki kot sliki ozadja, ne more diskriminirati med gibanjem v ospredju ter gibanjem v ozadju. Tekom razvoja nevronske mreže se na reševanje takšnih problemskih področji nismo osredotočali.

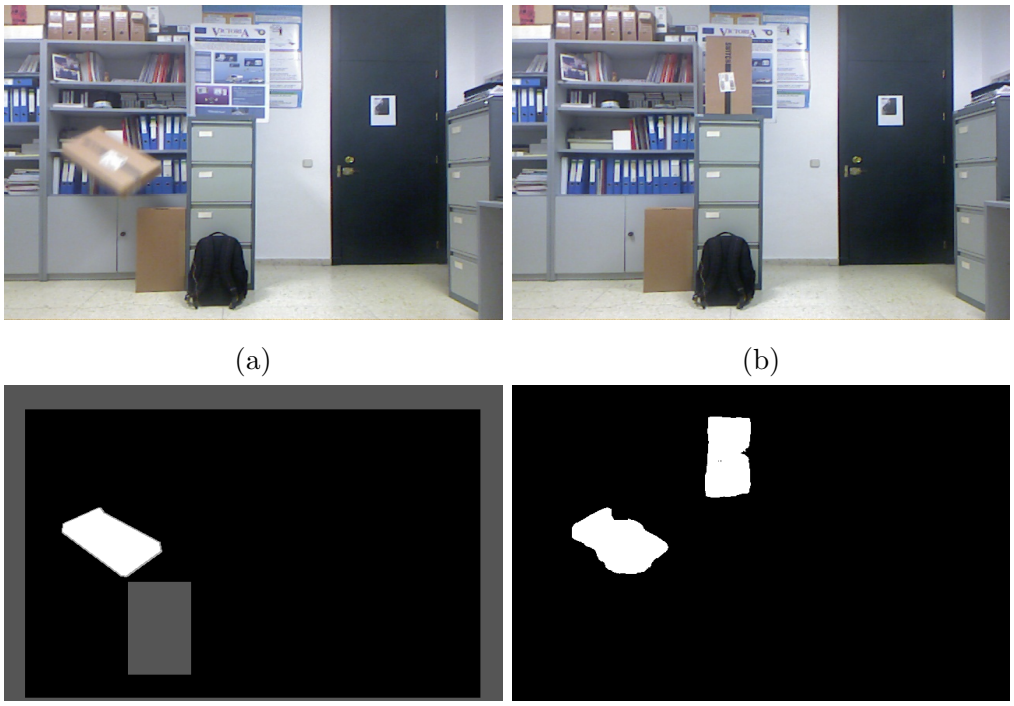
Priklic je problematičen v mnogih kategorijah podatkovne zbirke SGM-RGBD, ki je le v eni od kategorij višji kot 0.9. V nekaterih video sekvencah, zaradi njihove kratkosti, nismo mogli določiti kvalitetne referenčne slike ozadja. Zaradi prisotnosti objektov ospredja v ozadju se je zmanjšal priklic in v nekaterih primerih tudi natančnost. To lahko opazimo v Tabeli 4.6 pod kategorijo *Bootstrapping*.



(a) (b)

(c) (d) (e)

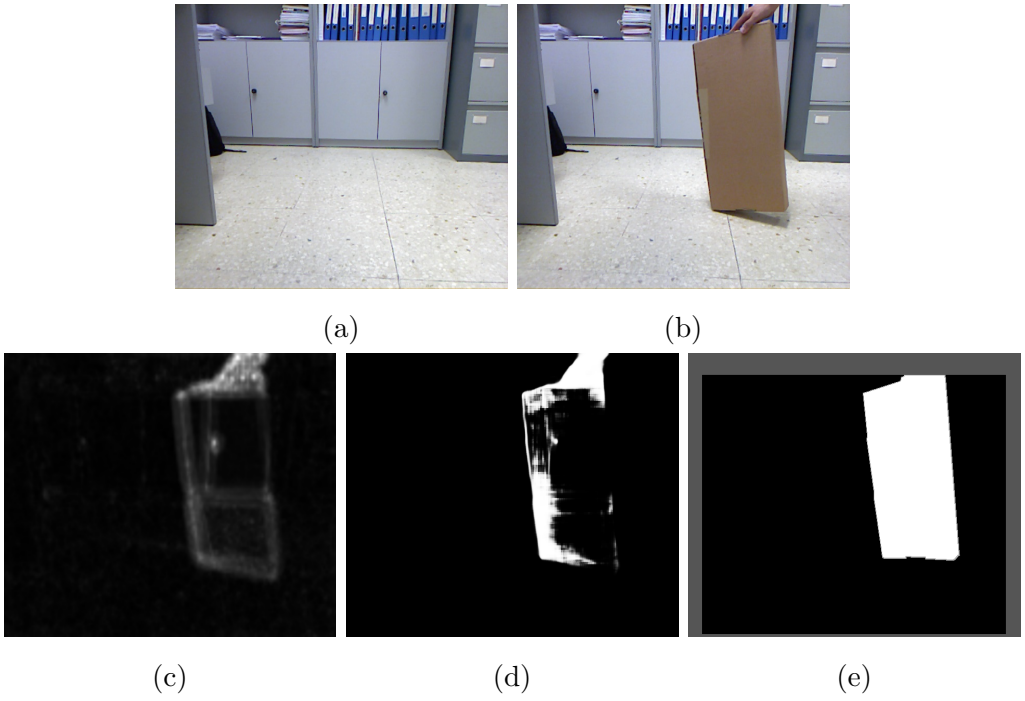
Slika 4.11: Sliki (a) ter (b) prikazujeta primer vhodne slike iz sekvence ter ozadja sekvence. Sledi slika (c), ki vsebuje rezultat nivoja razlike. Slika (d) prikazuje verjetnostno masko gibanja v sekvenci. Na sliki (e) vidimo, da nas pri segmentaciji zanima le manjši okvir, kjer prihaja do prekrivanja škatle ter belega ozadja. Ker je segmentacija v tej regiji slaba, je slabo tudi delovanje mreže na tem primeru.



(a) (b)

(c) (d)

Slika 4.12: Slika (a) predstavlja vhodno sliko iz sekvence, kjer se del ozadja, ki je viden v sliki (b) začne premikati. Slika (c) vsebuje pravilno segmentacijsko masko gibanja, slika (d) vsebuje segmentacijsko masko, ki jo naredi naša mreža. Zaradi prisotnosti škatle v referenčni sliki ozadja ta ob premiku povzroča lažne detekcije, ki zmanjšujejo natančnost segmentacije mreže.



Slika 4.13: Sliki (a) ter (b) prikazujeta primer vhodne slike iz sekvence ter ozadja sekvence. Sledi slika (c), ki vsebuje rezultat nivoja razlike. Slika (d) prikazuje verjetnostno masko gibanja v sekvenci. Na sliki (e) vidimo želeno masko segmentacije.

Kategorija *Shadow* podatkovne zbirke SGM-RGBD v nekaterih sekvencah vsebuje objekte, ki jih nevronska mreža slabo vizualno opiše. Tako je natančnost mreže v tej kategoriji visoka, a priklic majhen. Primer tega lahko vidimo na Sliki 4.13. Klub temu, da je barvna razlika med objektom ospredja ter ozadjem velika, ju prvi del mreže $SubNet$ opiše s podobnimi opisniki. Te se v nivoju razlike med seboj odštejejo. Tako drugi del mreže $SubNet$ ne more dobro detektirati razlik. To je poglavitni razlog za nizek priklic mreže na tej podatkovni zbirki. Opazimo lahko, da je roka v isti sliki dobro segmentirana, saj je verjetno opisana z drugačnimi opisniki kot ozadje. Z drugačnim izborom mreže, katere nivoje si sposodimo za prvi del mreže $SubNet$, ki bi bolje semantično ter diskriminativno opisala takšne objekte, bi priklic lahko povečali.

Poglavje 5

Sklep

V diplomskem delu smo predlagali novo metodo za detektiranje gibanja ter sprememb v video sekvencah. Metoda temelji na siamski arhitekturi konvolucijske nevronske mreže, ki je sposobna detekcije ter segmentacije gibanja v mnogih različnih težavnih okoliščinah. Za delovanje potrebuje referenčno sliko ozadja sekvence, ki jo evalvira. Ta je lahko določena ročno, v nekaterih primerih avtomatično s filtrom mediane. Ker mreža ne potrebuje ponovnega učenja za uporabo na različnih video sekvencah, to izboljša njeno aplikativno vrednost. Postopek učenja je računsko zelo zahtevna operacija.

Detekcijo gibanja ter sprememb v video sekvencah smo si zastavili kot problem iskanja razlik med ozadjem video sekvence ter posameznimi slikami iz te sekvence. Mrežo smo naučili iskati relevantne razlike ter ji med evalvacijo omogočili, da z detekcijami gibanja popravlja ter posodablja referenčno sliko ozadja, ki jo uporablja za detektiranje razlik. Mreža se je na javni podatkovni zbirki CDNET [43] uvrstila na 8. mesto izmed 46 objavljenih metod. Na tej podatkovni zbirki dosega povprečni priklic *0.9*, povprečno natančnost *0.81* ter povprečno F-mero *0.8*.

Našo metodo smo evalvirali tudi na podatkovnih zbirkah Wallflower [40] ter SGM-RGBD [16]. Rezultate smo opisali ter predstavili problemska področja, na katerih naša metoda deluje dobro ter področja, na katerih deluje slabo.

5.1 Možnosti izboljšav

Med razvojem metode smo razmišljali o možnih izboljšavah. Ker naš algoritem ni bil razvit za reševanje določenih problematičnih kategorij detekcije gibanja ter sprememb, je ena od smeri izboljšav naslavljanje teh kategorij. Siamsko konvolucijsko mrežo smo učili z eno samo sliko ozadja za celotno učno video sekvenco. Takšen način učenja lahko privede do situacij, kjer drastične spremembe v ozadju povzročajo karakteristično vizualno gibanje, ki ga mora mreža zavreči. To lahko v sekvencah z dinamičnim ozadjem izničuje korelacijo med karakterističnim gibanjem ter detektiranim gibanjem, saj želimo, da mreža karakteristično gibanje označi kot gibanje. Prav tako zaradi postopka učenja mreže nismo mogli učiti na video sekvencah, ki vsebujejo premikajoče se kamere. Zaradi tega, mreža zelo slabo deluje na takšnih primerih.

Da bi izboljšali delovanje naše metode, bi lahko mrežo učili sekvenčno, kjer bi segmentacijo $n-1$ iteracije mreže uporabili za določanje ozadja iteracije n. Tako bi omogočili mreži, da informacije o spremembah prenaša preko celotne video sekvence ter se nauči bolje razlikovati med relevantnim ter irelevantnim gibanjem. Tega lahko povzročajo gibajoče se kamere ali počasne ter hitre spremembe v okolici. Vsi zgoraj opisani problemi in smernice bodo predmet nadaljnjih raziskav.

Literatura

[1] Dosegljivo: http://cs231n.github.io/neural-networks-1/. [Dostopano dne 28.8.2018].

[2] Dosegljivo: https://themenwhostareatcodes.files.wordpress.com/2014/03/neuron-diagram-01.png. [Dostopano dne 28.8.2018].

[3] Dosegljivo: https://michelleful.github.io/PyCon2017/images/latex_generated_images/relu.png. [Dostopano dne 28.8.2018].

[4] Dosegljivo: http://machinelearninguru.com/_images/topics/computer_vision/basics/convolution/1.JPG. [Dostopano dne 28.8.2018].

[5] Dosegljivo: https://i1.wp.com/timdettmers.com/wp-content/uploads/2015/03/convolution.png. [Dostopano dne 28.8.2018].

[6] Dosegljivo: https://computersciencewiki.org/images/8/8a/MaxpoolSample2.png. [Dostopano dne 28.8.2018].

[7] A Framework for Designing the Architectures of Deep Convolutional Neural Networks. Dosegljivo: https://res.mdpi.com/entropy/entropy-19-00242/article_deploy/html/images/entropy-19-00242-g001.png. [Dostopano dne 28.8.2018].

[8] L. Ang Lim and H. Yalim Keles. Learning Multi-scale Features for Foreground Segmentation. *ArXiv e-prints*, August 2018.

[9] Mohammadreza Babaee, Duc Tung Dinh, and Gerhard Rigoll. A deep convolutional neural network for background subtraction. *CoRR*, abs/1702.01731, 2017.

[10] Simone Bianco, Gianluigi Ciocca, and Raimondo Schettini. Combination of video change detection algorithms by genetic programming. PP:1–1, 04 2017.

[11] G. Bilodeau, J. Jodoin, and N. Saunier. Change detection in feature space using local binary similarity patterns. In *2013 International Conference on Computer and Robot Vision*, pages 106–112, May 2013.

[12] Christopher M. Bishop. Pattern recognition and machine learning (information science and statistics). pages 225–281, Berlin, Heidelberg, 2006. Springer-Verlag.

[13] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[14] M. Braham, S. Piérard, and M. Van Droogenbroeck. Semantic background subtraction. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 4552–4556, Sept 2017.

[15] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a šiamese"time delay neural network. In *Proceedings of the 6th International Conference on Neural Information Processing Systems*, NIPS'93, pages 737–744, San Francisco, CA, USA, 1993. Morgan Kaufmann Publishers Inc.

[16] Massimo Camplani, Lucia Maddalena, Gabriel Moyá Alcover, Alfredo Petrosino, and Luis Salgado. A benchmarking framework for background subtraction in rgbd videos. In Sebastiano Battiato, Giovanni Maria Farinella, Marco Leo, and Giovanni Gallo, editors, *New Trends in Image*

Analysis and Processing – ICIAP 2017, pages 219–229, Cham, 2017. Springer International Publishing.

[17] Cdnet spletna stran. Dosegljivo: `http://changedetection.net/`. [Dostopano: 18. 7. 2018].

[18] J. C. Church, Yixin Chen, and S. V. Rice. A spatial median filter for noise removal in digital images. In *IEEE SoutheastCon 2008*, pages 618–623, April 2008.

[19] Convolutional neural networks for visual recognition. Dosegljivo: `http://cs231n.github.io/neural-networks-1/#actfun`. [Dostopano: 19. 7. 2018].

[20] Deep mind google. Dosegljivo: `https://deepmind.com/applied/deepmind-google/`. [Dostopano: 19. 7. 2018].

[21] Pojasnilo operacije dilate v dokumentaciji opencv. Dosegljivo: `https://docs.opencv.org/3.1.0/db/df6/tutorial_erosion_dilatation.html`. [Dostopano: 28. 8. 2018].

[22] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep learning. pages 330–372. MIT Press, 2016. `http://www.deeplearningbook.org`.

[23] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

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

[25] 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.

[26] Long Ang Lim and Hacer Yalim Keles. Foreground segmentation using a triplet convolutional neural network for multiscale feature encoding. *CoRR*, abs/1801.02225, 2018.

[27] Long Ang Lim and Hacer Yalim Keles. Foreground segmentation using convolutional neural networks for multiscale feature encoding. *Pattern Recognition Letters*, 112:256 – 262, 2018.

[28] Spletna stran matconvnet. Dosegljivo: `http://www.vlfeat.org/matconvnet/`. [Dostopano: 21. 7. 2018].

[29] MATLAB. *version 7.10.0 (R2012a)*. The MathWorks Inc., Natick, Massachusetts, 2012.

[30] Y. Nonaka, A. Shimada, H. Nagahara, and R. Taniguchi. Evaluation report of integrated background modeling based on spatio-temporal features. In *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 9–14, June 2012.

[31] M. Piccardi. Background subtraction techniques: a review. In *2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No.04CH37583)*, volume 4, pages 3099–3104 vol.4, Oct 2004.

[32] Zbiranje podatkov na spletu. Dosegljivo: `https://blog.microfocus.com/how-much-data-is-created-on-the-internet-each-day/`. [Dostopano: 19. 7. 2018].

[33] Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for activation functions. *CoRR*, abs/1710.05941, 2017.

[34] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[35] D. Russell and Shaogang Gong. A highly efficient block-based dynamic background model. In *IEEE Conference on Advanced Video and Signal Based Surveillance, 2005.*, pages 417–422, Sept 2005.

[36] Spletna stran sbm-rgbd podatkovne zbirke. Dosegljivo: `http://rgbd2017.na.icar.cnr.it/SBM-RGBDdataset.html`. [Dostopano: 2. 8. 2018].

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

[38] P. L. St-Charles, G. A. Bilodeau, and R. Bergevin. Subsense: A universal change detection method with local adaptive sensitivity. *IEEE Transactions on Image Processing*, 24(1):359–373, Jan 2015.

[39] C. Stauffer and W. E. L. Grimson. Adaptive background mixture models for real-time tracking. In *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, volume 2, page 252 Vol. 2, 1999.

[40] Kentaro Toyama, John Krumm, Barry Brumitt, and Brian Meyers. Wallflower: Principles and practice of background maintenance. pages 255–261. IEEE Computer Society Press, September 1999.

[41] Andrea Vedaldi and Karel Lenc. Matconvnet: Convolutional neural networks for matlab. In *Proceedings of the 23rd ACM International Conference on Multimedia*, MM '15, pages 689–692, New York, NY, USA, 2015. ACM.

[42] Spletna stran wallflower podatkovne zbirke. Dosegljivo: `https://www.microsoft.com/en-us/research/project/test-images-for-wallflower-paper/`. [Dostopano: 2. 8. 2018].

[43] Y. Wang, P. M. Jodoin, F. Porikli, J. Konrad, Y. Benezeth, and P. Ishwar. Cdnet 2014: An expanded change detection benchmark dataset.

In *2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 393–400, June 2014.

[44] Yi Wang, Zhiming Luo, and Pierre-Marc Jodoin. Interactive deep learning method for segmenting moving objects. *Pattern Recognition Letters*, 96:66 – 75, 2017. Scene Background Modeling and Initialization.

[45] C. R. Wren, A. Azarbayejani, T. Darrell, and A. P. Pentland. Pfinder: real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):780–785, Jul 1997.

[46] Yong Xu, Jixiang Dong, Bob Zhang, and Daoyun Xu. Background modeling methods in video analysis: A review and comparative evaluation. *CAAI Transactions on Intelligence Technology*, 1(1):43 – 60, 2016.

[47] Wenbo Zheng, Kunfeng Wang, and Fei-Yue Wang. Background subtraction algorithm with bayesian generative adversarial networks. 44, 05 2018.

[48] Wenbo Zheng, Kunfeng Wang, and Fei-Yue Wang. A novel background subtraction algorithm based on parallel vision and bayesian gans. 05 2018.

[49] Z. Zivkovic. Improved adaptive gaussian mixture model for background subtraction. volume 2, pages 28–31, 2004. cited By 1208.