Technická univerzita v Košiciach
Fakulta elektrotechniky a informatiky

Metódy hlbokého učenia pre segmentáciu obrazu

Diplomová práca

2019 **Bc. Maroš Plšík**

Technická univerzita v Košiciach
Fakulta elektrotechniky a informatiky

Metódy hlbokého učenia pre segmentáciu obrazu

Diplomová práca

Študijný program:	Hospodárska informatika
Študijný odbor:	9.2.10 Hospodárska informatika
Školiace pracovisko:	Katedra kybernetiky a umelej inteligencie (KKUI)
Školiteľ:	Ing. Peter Bednár PhD.
Konzultant:	

Košice 2019 **Bc. Maroš Plšík**

Názov práce:	Metódy hlbokého učenia pre segmentáciu obrazu
Pracovisko:	Katedra kybernetiky a umelej inteligencie, Technická univerzita v Košiciach
Autor:	Bc. Maroš Plšík
Školiteľ:	Ing. Peter Bednár PhD.
Konzultant:	
Dátum:	3. 5. 2019
Kľúčové slová:	Hlboké neurónové siete, Hlboké učenie, Segmentácia videa, Spracovanie obrazu, Online trénovanie, PyTorch

Abstrakt: Cieľom tejto diplomovej práce je implementácia riešenia úlohy segmentácie objektov vo videu s čiastočným dohľadom, ktoré je založené na metódach hlbokých neurónových sietí. Toto riešenie, vytvorené podľa existujúceho prístupu, v diplomovej práci rozšírime pridaním nových trénovacích dát a implementáciou online trénovania. Model v pôvodnej aj v modifikovanej verzii testujeme na dátovej množine využívanej aj inými výskumníkmi, vďaka čomu môžeme následne vykonať porovnanie medzi naším riešením a prístupmi iných ľudí. Takisto vieme vyhodnotiť prínos nami implementovaných úprav a analyzovať silné a slabé stránky architektúry, na ktorej je naše riešenie postavené.

Thesis title:	Deep learning methods for image segmentation
Department:	Department of Cybernetics and Artificial Intelligence, Technical University of Košice
Author:	Bc. Maroš Plšík
Supervisor:	Ing. Peter Bednár PhD.
Tutor:	
Date:	3. 5. 2019
Keywords:	Deep neural networks, Deep learning, Video segmentation, Image processing, Online training, PyTorch

Abstract: The aim of this thesis is to implement a solution for semi-supervised video object segmentation task using methods of deep neural networks. We implement our solution based on another preexisting approach and try to further improve its accuracy. To achieve better results we train the network using new extensive dataset and implement our version of online training. We test the base model as well as the improved one on a generally used testing dataset, what allows us to compare our solution with other approaches as well as analyze the accuracy improvement we achieve by using our modifications. At the same time, we examine strengths and weaknesses of the architecture based on which our solution is built.

TECHNICKÁ UNIVERZITA V KOŠICIACH

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Katedra kybernetiky a umelej inteligencie

ZADANIE DIPLOMOVEJ PRÁCE

Študijný odbor: **Hospodárska informatika**

Študijný program: **Hospodárska informatika**

Názov práce:

Metódy hlbokého učenia pre segmentáciu obrazu

Deep learning methods for segmentation of video sequences

Študent: **Bc. Maroš Plšík**

Školiteľ: **Ing. Peter Bednár, PhD.**

Školiace pracovisko: **Katedra kybernetiky a umelej inteligencie**

Konzultant práce:

Pracovisko konzultanta:

Pokyny na vypracovanie diplomovej práce:

1. Podať teoretický prehľad problematiky segmentovania obrazu a hlbokých neurónových sietí
2. Navrhnúť a implementovať vhodnú architektúru pre segmentovanie obrazu založenú na metódach hlbokého učenia
3. Otestovať a vyhodnotiť implementovanú architektúru na zvolenej množine testovacích dát
4. Vypracovať dokumentáciu podľa pokynov katedry a vedúceho práce (napr. hlavná časť 50-70 strán, tlačená forma v nerozoberateľnej väzbe, používateľská a systémová príručka, DVD obsahujúce príslušné dokumenty a softvérový výstup práce)

Jazyk, v ktorom sa práca vypracuje: slovenský

Termín pre odovzdanie práce: 26.04.2019

Dátum zadania diplomovej práce: 31.10.2018

.....

prof. Ing. Peter Sinčák, CSc.

vedúci garantujúceho pracoviska

TECHNICKÁ UNIVERZITA V KOŠICIACH FAKULTA ELEKTROTECHNIKY A INFORMATIKY -12-

.....

prof. Ing. Liberios Vokorokos, PhD.

dekan fakulty

Čestné vyhlásenie

Vyhlasujem, že som záverečnú prácu vypracoval(a) samostatne s použitím uvedenej odbornej literatúry.

Košice, 3.5.2019 .....

Vlastnoručný podpis

Poďakovanie

Rád by som touto cestou poďakoval vedúcemu mojej diplomovej práce, pánovi Ing. Petrovi Bednárovi, PhD. za zdieľanie jeho bohatých vedomostí a za jeho rady a pripomienky k tejto práci.

Takisto sa chcem poďakovať mojej rodine, priateľom a najmä priateľke Káji, ktorá ma stále, aj keď nie vždy úspešne, motivovala a motivuje k zodpovednému prístupu k škole a k štúdiu.

Obsah

Úvod **1**

1 Formulácia úlohy a cieľ práce **2**

2 Teoretický rozbor **4**
2.1 Reprezentácia obrazu a jeho spracovanie 4
2.2 Segmentácia obrazu 5
2.2.1 Prahovanie 7
2.2.2 Detekcia hrán 8
2.2.3 Zhlukovanie 9
2.2.4 Segmentácia použitím konvolučných neurónových sietí . . . 10
2.3 Úvod do neurónových sietí 12
2.3.1 Neurónová sieť a hlboké učenie 15
2.3.2 Konvolučné neurónové siete - CNN 17
2.3.3 Rekurentné neurónové siete - RNN 20
2.3.4 Reziduálne neurónové siete 24

3 Analýza súčasného stavu **27**
3.1 One Shot Video Object Segmentation - OSVOS 27
3.1.1 Architektúra 27
3.1.2 Trénovanie 28
3.1.3 Použitie a vlastnosti 29
3.1.4 Príklady implementácie 31
3.2 MaskTrack 32
3.2.1 Architektúrau 32
3.2.2 Trénovanie 34
3.2.3 Použitie a vlastnosti 35

3.2.4 Príklady implementácie 36

4 Návrh a implementácia riešenia 37
4.1 Existujúce riešenie 37
4.1.1 Architektúra 37
4.1.2 Trénovanie 39
4.1.3 Testovanie 40
4.2 Návrh modifikácii modelu 41
4.2.1 Lucid data dreaming 41
4.2.2 Konvolučné orientované kontúry - COB 43
4.3 Metodológia testovania 45
4.3.1 Jaccardov index 46
4.3.2 F miera ohraničenia 47
4.4 Použité dáta 47
4.4.1 DAVIS 47
4.4.2 YouTube - VOS 49

5 Dosiahnuté výsledky 51
5.1 Pôvodný model RGMPorigin 51
5.1.1 Popis 51
5.1.2 Výsledky 52
5.2 Nový model RGMPnew 54
5.2.1 Popis 54
5.2.2 Výsledky 55
5.3 Model RGMPnew + online trénovanie 57
5.3.1 Popis 57
5.3.2 Výsledky 57
5.4 Porovnanie a vyhodnotenie 60

6 Ďalšia práca 62

Záver 63

Literatúra 64

Zoznam skratiek 70

Zoznam príloh **71**

1 Požadovaný hardvér **1**

2 Príprava **2**
2.1 Príprava súborovej štruktúry 2
2.2 Inštalácia požadovaného softvéru 2

3 Testovanie modelu **3**

4 Trénovanie modelu **5**

1 RGMP model a utility **1**
1.1 model.py 1
1.2 utils.py 1

2 Trénovanie a testovanie **5**
2.1 test.py 5
2.2 train.py 6

3 Lucid data dreaming **7**
3.1 PatchMatch.py 7
3.2 patchPaint.py 7
3.3 lucidDream.py 8
3.4 lucid_generate.py 8

Zoznam obrázkov

1.1 Ukážka segmentácie videa. 2

2.1 Využitie segmentácie prahovaním v súčinnosti s ďalšími technikami na spočítanie fliaš v debničke. 8
2.2 Segmentácia obrazu použitím algoritmu K-means. 9
2.3 Rozdiel medzi sémantickou segmentáciou a segmentáciou inštancií. 12
2.4 Naučené reprezentácie modelu na rozpoznávanie písaných čísel. . 16
2.5 Architektúra CNN. 18
2.6 Fungovanie konvolučnej vrstvy. 19
2.7 $MAX()$ pooling vyberá najvyššiu hodnotu z každého okna s rozmermi 2×2. 20
2.8 Rozdiel medzi neurónmi doprednej a rekurentnej siete. 21
2.9 Alternatívy mapovania vstupných vektorov na výstupné. 23
2.10 RNN sieť rozbalená v čase t. 23
2.11 Reziduálny blok. 25
2.12 Porovnanie reziduálnej (hore) a tradičnej siete (dole). 26

3.1 Architektúra OSVOS zložená z dvoch modulov. 28
3.2 Priebeh trénovania a testovania v architektúre OSVOS. 29
3.3 OSVOS - rozdiel v kvalite segmentácie vzhľadom na trvanie dotrénovania na prvej segmentovanej snímke - 10 sekúnd / 1 minúta. . . 30
3.4 Porovnanie výsledku bežnej konvolúcie a atrous konvolúcie. 32
3.5 Proces segmentácie statického obrázku v modeli DeepLabv2. 33
3.6 Model je natrénovaný tak, aby dokázal z kombinácie snímky v čase t a segmentačnej masky z predchádzajúcej snímky v čase $t-1$ vygenerovať segmentačnú masku pre danú snímku v čase t. 34

3.7 Výsledky segmentácie pri použití segmentačnej masky z predchádzajúcej snímky a pri použití ohraničujúceho rámiku. 35

4.1 Architektúra RGMP s detailne popísanými modulmi. 38
4.2 Umelé generovanie trénovacích vzoriek zo statických obrázkov použitím dvoch rôznych postupov. 39
4.3 Rekurentné trénovanie modelu. 40
4.4 Generovanie dát na základe prvej anotovanej snímky. 42
4.5 Proces spracovania obrazu v architektúre COB. 44
4.6 Prvý rad ilustruje výstupy skupín konvolučných vrstiev pre štyri rôzne orientácie. V druhom rade sú postupne zobrazené: 1 - pôvodný obrázok, 2 - sila detegovaných kontúr, 3 - rozpoznaná orientácia kontúr a 4 - detegované kontúry na všetkých úrovniach obrazu. . . 45
4.7 Ukážka anotovaných snímok z dátovej množiny DAVIS. 48
4.8 Ukážka anotovaných snímok z dátovej množiny YouTube-VOS. . . . 50

5.1 Ukážka výsledkov segmentácie modelu RGMPorigin. 53
5.2 Porovnanie výsledkov segmentácie modelov RGMPorigin a RGMPnew v čase t. 55
5.3 Porovnanie výsledkov segmentácie modelov RGMPnew a RGMPnew+Lucid. 58

2.1 Štruktúra adresárov po rozbalení source.zip. 2

Zoznam tabuliek

5.1 Porovnanie vplyvu rôznych konfigurácií online trénovania na presnosť modelu. 59
5.2 Kvantitatívne vyhodnotenie a porovnanie metód segmentácie. . . . 61

Úvod

Miera využívania strojového učenia v každodennom živote sa celosvetovo neustále zvyšuje. Zatiaľčo ešte pred niekoľkými rokmi boli ľudia ohromení schopnosťou digitálneho fotoaparátu automaticky zaostriť na tvár človeka, dnes dokážu počítače šoférovať naše autá, komunikovať, písať básne, rozpoznávať obraz, zvuk či text, pričom niektoré z týchto činností dokážu robiť spoľahlivejšie, rýchlejšie a s menšou námahou ako človek.

Jednou z oblastí informatiky, ktorú výrazne ovplyvnil rozvoj strojového učenia je aj počítačové videnie. S algoritmami a technikami, ktoré patria do tejto oblasti, sa dnes môžeme stretnúť a stretávame na každom kroku - od automatických úprav fotografií zhotovených našimi inteligentnými telefónmi, cez RTG diagnostiku pacientov v nemocniciach až po virtuálnu realitu. Dôležitou súčasťou týchto troch oblastí, ale aj mnohých ďalších, je proces segmentácie obrazu.

Segmentácia obrazu, či už statického v podobe fotiek alebo dynamického v podobe videa, je úloha počítačového videnia, ktorej cieľom je identifikovať objekt záujmu v obraze a tento objekt čo najpresnejšie označiť a ohraničiť. Vďaka vývoju riešení tejto úlohy vie telefón upraviť tú časť fotografie, na ktorej sa nachádza odfotografovaná osoba, počítač vie na RTG snímke ohraničiť nádor a autonómny automobil vie, na ktorej časti cesty sa nachádza on a kde presne sa nachádzajú iné autá a prekážky.

Cieľom tejto práce je využiť techniky strojového učenia, konkrétne umelé neurónové siete, na implementáciu riešenia úlohy segmentácie videa, ako aj podať teoretický výklad o týchto oblastiach informatiky a vytvoriť prehľad o súčasných riešeniach, ich fungovaní a ich možnostiach.

1 Formulácia úlohy a cieľ práce

Hlavným cieľom tejto diplomovej práce je implementácia existujúceho riešenia poloautomatickej segmentácie videa a následné vylepšenie tohto riešenia použitím metód strojového učenia s primárnym zameraním na metódy neurónových sietí a hlbokého učenia.

Model, ktorý bude popri dokumentácii hlavným výstupom tejto práce, bude fungovať nasledovným spôsobom. Na vstupe prijme video obsahujúce jeden alebo niekoľko sledovaných objektov. K prvej snímke videa bude dodaná aj segmentačná maska, ktorá bude presne vymedzovať tieto objekty od pozadia a od seba navzájom. Model, založený na metódach hlbokého učenia, túto sekvenciu spracuje a na výstupe vygeneruje segmentačné masky pre všetky ostatné snímky videa, pričom tieto masky budú čo najpresnejšie označovať sledované objekty v jednotlivých snímkach, vrátane pozadia. Proces segmentácie je ilustrovaný na Obr.1.1.

Sledované objekty v tomto prípade predstavujú jednotlivé inštancie konkrétnych objektov. Tieto objekty vo videách budú meniť svoju polohu, tvar, farby aj nasvietenie a pred týmito objektmi sa môžu na niektorých snímkach objaviť iné objekty v podobe prekážok. Sledované objekty môžu takisto presahovať cez seba navzájom a v niektorých prípadoch nemusia byť zaznamenané v svojej celkovej podobe,

Obr. 1.1: Ukážka segmentácie videa. Model sa z prvej anotovanej snímky naučí reprezentáciu sledovaného objektu a v ďalších snímkach samostatne segmentuje daný objekt. Zdroj:[26]

či už v časti alebo v celom videu. Zahŕňa to aj situácie kedy sa objekt na krátky čas úplne stratí z dohľadu. Vo videách sa môžu objavovať náhle a nečakané pohyby, obraz nemusí byť dostatočne zaostrený alebo môže byť roztrasený. So všetkými týmito nedokonalosťami sa musí vedieť model vysporiadať.

Ďalšou podmienkou je schopnosť modelu spracovať video v ľubovoľnom rozlíšení. Ak túto podmienku nebude možné splniť, bude na vstupe implementované predspracovanie dátovej množiny, ktoré zabezpečí konštantné rozlíšenie dát vstupujúcich do modelu. Model bude schopný spracovať video vo farebnom spektre RGB a s ľubovoľnou dĺžkou trvania.

Hlavný cieľ práce je rozdelený na niekoľko menších podcieľov:

1. **Štúdium problematiky.** Štúdium zahŕňa témy ako segmentácia obrazu, metódy hlbokého učenia a využitie týchto metód v praxi.
2. **Získanie dát.** Je potrebné nájsť dátové množiny, ktoré využívajú aj iní výskumníci pri trénovaní a testovaní modelov segmentácie obrazu.
3. **Štúdium existujúcich riešení a výber riešenia vhodného na ďalšie vylepšovanie.** Zvolené riešenie musí dosahovať výsledky blízke úrovni súčasných najlepších modelov (SOTA), byť relatívne jednoduché na implementáciu a poskytovať dostatočný priestor na pridanie ďalších vylepšení a optimalizáciu celkovej funkčnosti.
4. **Implementácia vybraného riešenia.** Riešenie budeme implementovať tak, aby dosahovalo výsledky podobné alebo veľmi zhodné s výsledkami, ktoré dosiahli autori riešenia v ich referenčnej publikácii.
5. **Implementácia vylepšení.** Tieto by mali byť založené na metódach neurónových sietí, no niektoré môžu využívať aj iné prístupy strojového učenia. Ich implementáciou by nemalo dôjsť k funkčným obmedzeniam modelu a mali by zvyšovať jeho celkovú presnosť a výkonnosť.
6. **Vyhodnotenie výsledkov.** Vylepšený model bude porovnaný s inými existujúcimi riešeniami a takisto s pôvodným modelom bez vylepšení. Porovnanie prebehne na základe dvoch metrík - *Jaccardovho indexu* a *F miery ohraničenia*. Cieľom je zvýšiť presnosť pri zachovaní rýchlosti a stability riešenia a v ideálnom prípade sa priblížiť alebo prekonať aktuálne SOTA riešenia.

2 Teoretický rozbor

Obsahom kapitoly je podrobný popis teoretických poznatkov, ktoré boli využité pri spracovaní tejto záverečnej práce. V prvej polovici kapitoly budú rozobraté témy ako digitálny obraz, metódy jeho spracovania, čo znamená segmentácia obrazu a niekoľko praktických prístupov k segmentácii. Podkapitoly v druhej polovici budú obsahovať teoretický vstup do témy neurónových sietí, bude popísané ich praktické využite v rôznych oblastiach a nakoniec budú podrobne vysvetlené typy neurónových sietí využité pri implementácii praktickej časti tejto práce.

2.1 Reprezentácia obrazu a jeho spracovanie

Každý digitálny obraz je nositeľom veľkého množstva užitočných informácií. Zdravý ľudský mozog dokáže veľkú časť týchto informácií extrahovať z obrazu rýchlo a bez veľkej námahy, čo je výsledkom evolúcie trvajúcej milióny rokov. Extrakcia informácií z obrazu však tvorí zásadný problém pre človekom umelo vytvorené počítačové systémy. Postupom času sa ľuďom darí vytvárať algoritmické postupy, vďaka ktorým dokáže počítač spracovať čím ďalej tým zložitejšie obrazy a extrahovať z nich potrebné informácie. Disciplína z oblasti počítačových vied, ktorá sa venuje tejto úlohe, sa nazýva Spracovanie digitálneho obrazu.

Táto disciplína je v dnešnej dobe bežnou súčasťou nášho každodenného života, od automatických úprav fotografií, ktoré vytvoríme, cez rozpoznávanie tvárí a objektov vo fotografiách a videách až po optické rozpoznávanie znakov vďaka ktorému vieme prevádzať text z obrazovej do bitovej formy. Spracovanie obrazu je oblasť so širokým záberom, ktorá sa dá rozdeliť na niekoľko menších častí podľa aplikácie [46]:

1. **Vylepšenie obrazu.** Tento proces predstavuje zvýšenie kvality obrazu takým

spôsobom, aby pozorovateľovi obraz po úprave poskytoval viac kvalitnejších informácií ako pred úpravou. Medzi operácie vylepšenia obrazu patrí zmena úrovne kontrastu, zvýraznenie hrán, transformácie jasu a saturácie a niekoľko ďalších.

2. **Obnovenie obrazu.** Proces veľmi podobný vylepšeniu obrazu s tým rozdielom, že obnovenie obrazu prebieha objektíve a využíva matematické a pravdepodobnostné metódy. Patrí sem odstránenie šumu, optimalizácia kontrastu, zaostrenie rozmazaného obrazu a ďalšie.

3. **Kompresia obrazu.** Cieľom kompresie je zníženie redundancie obrazu a jeho uloženie v efektívnejšej podobe pri zachovaní väčšiny podstatných informácií.

4. **Morfologické spracovanie.** „Morfologické spracovanie obrazov zahŕňa množinu nelineárnych operácií súvisiacich s tvarom alebo morfológiou čŕt v obraze.“ [39]

5. **Segmentácia obrazu.** Zahŕňa rozdelenie obrazu na niekoľko častí podľa podobnosti pixelov alebo podľa sémantiky segmentovaných objektov.

6. **Detekcia a rozpoznávanie.** Tieto operácie sa sústredia na rozpoznanie konkrétnych objektov v obraze a ich pomenovanie alebo bližší popis.

Spracovanie videa je do istej miery veľmi podobným procesom ako spracovanie statického obrazu. Procesy ako vylepšenie obrazu a obnovenie obrazu prebiehajú rovnako v prípade videí ako aj v prípade obrázkov, na druhú stranu procesy kompresie alebo segmentácie obrazu môžu v prípade videí využiť časo-priestorové informácie, teda informácie o zmenách medzi jednotlivými snímkami, čo im umožňuje dosahovať lepšie výsledky.

2.2 Segmentácia obrazu

Rastrový digitálny obraz v matematickom ponímaní predstavuje funkciu dvoch reálnych premenných $f(x, y)$, v ktorej premenné x a y označujú polohu jednotlivých pixelov v obraze a výsledné hodnoty funkcie f predstavujú úroveň jasu pixelov na určených pozíciách. Každý takýto obraz je zložený z menších častí, ktoré je možné nazvať oblasťami záujmu (regions of interest), alebo jednoducho oblasťami. To

znamená, že obraz vo väčšine prípadov obsahuje kolekciu objektov, ktoré tvoria základ týchto regiónov. Práve tento princíp je základným konceptom, ktorý sa využíva v úlohách segmentácie obrazu.

„Pri segmentácii rozdeľujeme digitálny obraz na viaceré oblasti (segmenty) za účelom zjednodušenia a/alebo zmeny reprezentácie obrazu na takú, ktorá je zmysluplnejšia a ktorú je jednoduchšie analyzovať. Segmentácia obrazu obvykle lokalizuje objekty a ich hranice.“[39]

Cieľom segmentácie obrazu je oddeliť od seba tie oblasti, ktoré reprezentujú zmysluplné celky ako napríklad zalesnené alebo zastavané oblasti na satelitných obrázkoch, jednotlivé osoby na fotkách, orgány na medicínskych snímkach alebo objekty na vozovke okolo auta schopného autonómneho riadenia. Na to, aby boli segmentované oblasti užitočné pre analýzu, mali by tieto oblasti výrazne súvisieť s objektmi, ktoré sú obsahom týchto oblastí. Zmysluplná segmentácia je prvým a kritickým krokom v procese analýzy obsahu obrazu. Úspech tejto analýzy závisí v prvom rade na spoľahlivosti a presnosti segmentačného algoritmu, modelu.

Proces segmentácie obrazu má široké využitie a uplatňuje sa v rozličných oblastiach každodenného života. Dôležitú úlohu zohráva najmä v týchto aplikáciách:

- strojové videnie:
  - automatizácia výrobných a kontrolných systémov,
  - inšpekcia kvality a odolnosti materiálov,
  - navádzanie robotických systémov,
- detekcia objektov:
  - ľudí alebo tvárí,
  - prírodných alebo umelých štruktúr na satelitných snímkach,
- rozpoznávanie objektov:
  - konkrétnych osôb,
  - textu - značky áut, poznámky, knihy, atď.,
  - identifikácia živočíšnych a rastlinných druhov,

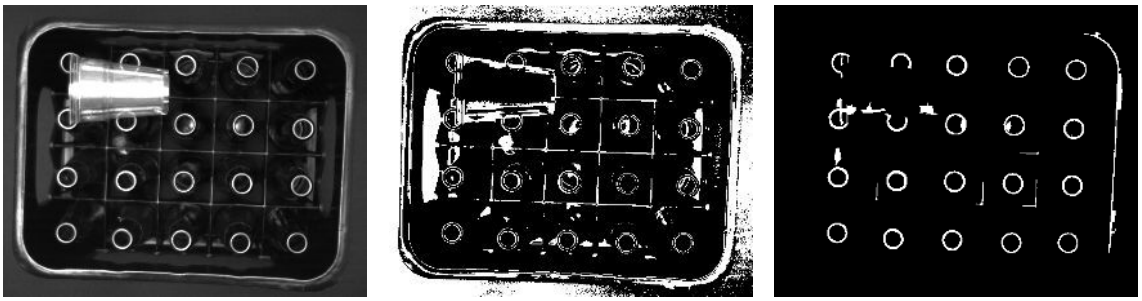
  - identifikácia vesmírnych objektov,
- medicína:
  - detekcia nádorov,
  - identifikácia vírusu/baktérie,
  - asistencia pri chirurgických zákrokoch,
- rozšírená a virtuálna realita.

V súčasnosti existuje niekoľko hlavných prístupov k automatickej alebo poloautomatickej segmentácii obrazu [30], z ktorých niektoré budú bližšie popísané v nasledujúcej časti.

2.2.1 Prahovanie

Prahovanie patrí medzi techniky segmentácie založené na vizuálnej podobnosti oblastí. Je to najjednoduchší prístup segmentácie obrazu, ktorý stále nachádza svoje uplatnenie v špecifických oblastiach, kde sa neočakáva vysoká variabilita úrovní svetelnosti segmentovaných objektov v obraze. Vhodným príkladom je detekcia hraníc jednotlivých prírodných útvarov alebo pásem na satelitných snímkach krajiny prípadne automatická kontrola kvantity a kvality výrobkov uložených na výrobnom páse.

Základný algoritmus prahovania segmentuje jednotlivé oblasti v obraze podľa ich úrovne svetelnosti - pixely so svetelnosťou nižšou ako je zadefinovaná hodnota budú priradené do jednej kategórie a ostatné pixely budú priradené do druhej kategórie. V bežnej implementácii to znamená, že jednej skupine pixelov bude priradená minimálna svetelnosť - čierna farba a druhej skupine naopak maximálna svetelnosť - biela farba. Na prvý pohľad jednoduchý algoritmus je možné výrazne funkcionálne obohatiť a zvýšiť tak jeho použiteľnosť v praxi. Rozšíriteľnosť spočíva v automatizácii nastavovania prahu a v umožnení použitia rozdielnych hodnôt prahu pre rôzne oblasti obrazu. Automatické nastavovanie prahových úrovní je možné dosiahnuť viacerými technikami, medziiným aj využitím histogramu cieľového obrazu, analýzou zhlukov podobných pixelov v obraze, štatistickými metódami, minimalizáciou variancie, atď. [25]



Obr. 2.1: Využitie segmentácie prahovaním v súčinnosti s ďalšími technikami na spočítanie fliaš v debničke.

2.2.2 Detekcia hrán

Detekcia hrán je technika spracovania obrazu, ktorá slúži predovšetkým ako dodatočná súčasť procesu segmentovania obrazu. Táto technika je založená na princípe hľadania výrazných lokálnych zmien intenzity. Pre nájdenie týchto zmien v obraze definovanom ako funkcia $f(x, y)$ sa najčastejšie využíva výpočet úrovne (G) a smeru (θ) gradientu na pozícii (x, y), ktorý je definovaný vektorom

$$\nabla f \equiv grad(f) \equiv \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \tag{2.1}$$

a to podľa vzorcov:

$$G[f(x, y)] = \sqrt{G_x^2 + G_y^2} \tag{2.2}$$

$$\theta(x, y) = \tan^{-1}(G_y/G_x) \tag{2.3}$$

kde θ predstavuje uhol gradientu vzhľadom na os x.

Proces detekcie hrán sa skladá z troch základných krokov [30]:

1. **Filtrovanie a vylepšenie obrazu.** Pre dosiahnutie zmysluplných výsledkov detekcie hrán je potrebné odstrániť maximálne možné množstvo šumu z obrazu bez straty informácií o zmenách gradientu hrán.

2. **Detekcia bodov hrán.** Algoritmus musí rozhodnúť o tom, ktoré body sú súčasťou hrán a ktoré sú iba neodstráneným šumom.

3. **Lokalizácia hrán.** Nie všetky nájdené hrany sú relevantné pre konkrétnu aplikáciu, preto je nutné identifikovať iba tie hrany, ktoré používateľovi prinesú úžitkovú hodnotu.

Obr. 2.2: Segmentácia obrazu použitím algoritmu K-means s definovaným počtom zhlukov $K = 3$. Zdroj:[30]

2.2.3 Zhlukovanie

Algoritmy segmentácie na princípe zhlukovania umiestňujú pixely do spoločného príznakového priestoru, kde tieto pixely usporadúvajú do zhlukov na základe podobnosti príznakov a následne podľa týchto zhlukov kategorizujú jednotlivé pixely do konkrétnych segmentov.

Medzi najčastejšie využívané zhlukovacie algoritmy patrí K-means. Kritériom pri vytváraní zhlukov je v tomto prípade vzájomná vzdialenosť pixelov v príznakovom priestore, takže čím bližšie sú pixely pri sebe, tým väčšia je pravdepodobnosť, že patria do rovnakej nezávislej oblasti, do jedného zhluku [44]. Výhodou algoritmu K-means je jednoduchosť jeho implementácie, rýchlosť spracovania vstupných dát a dobrá škálovateľnosť. Nevýhodou je nutnosť manuálne zvoliť počet cieľových zhlukov, pričom táto informácia je v niektorých situáciách ťažko odhadnuteľná aj samotným expertom. Proces fungovania algoritmu K-means je nasledovný [37]:

1. Používateľ zvolí K - počet hľadaných zhlukov.
2. Náhodná inicializácia K centier zhlukov.
3. Výpočet vzdialenosti všetkých bodov v priestore od všetkých centier zhlukov

a následné priradenie bodov k najbližším centrám.

4. Každý zhluk prepočíta súradnice svojho centra spriemerovaním súradníc všetkých bodov priradených danému zhluku.

5. Opakovanie krokov 3 a 4 do momentu, kedy sa centrá zhlukov nemenia alebo keď bude dosiahnutý zadaný počet iterácií.

Ukážka výsledkov zhlukovania algoritmom K-means je zobrazená na Obr.2.2.

2.2.4 Segmentácia použitím konvolučných neurónových sietí

Vývoj konvolučných neurónových sietí (CNN), ktoré budú popísané v kapitole 2.3.2, spôsobil výrazné zmeny v prístupe k riešeniu úloh ako je detekcia objektov v obraze, klasifikácia obrazu a samozrejme aj segmentácia obrazu. Prednosťou CNN pri riešení segmentácie obrazu je najmä schopnosť extrahovať sémantické informácie o objektoch v obraze a na základe týchto informácií objekty segmentovať od pozadia alebo medzi sebou navzájom s relatívne vysokou presnosťou. V nasledujúcich odsekoch popíšeme niekoľko prístupov k segmentácii s využitím CNN, ktoré sú špecificky určené na segmentáciu objektov vo videách.[29]

Lokálne šírenie informácií

Tieto metódy vytvárajú grafy, ktoré spájajú superpixely[1] a časti objektov medzi dvomi po sebe idúcimi snímkami. Týmto spôsobom dokážu sledovať zmeny v pozícii a polohe objektu medzi dvomi snímkami a tak postupne segmentovať celé video. Nevýhodou je postupné nabaľovanie chyby spôsobené tým, že sa informácie vždy šíria iba z predchádzajúcej snímky a takýto model teda nie je dobre použiteľný v prípade dlhších videí. Vstupom do takéhoto modelu je video vrátane ground truth (pravdivej) segmentačnej masky, ktorá označuje jeden alebo niekoľko sledovaných objektov. Výstupom sú segmentačné masky označujúce rovnaké objekty ako prvá ground truth maska.

Globálne šírenie informácií

V tomto prípade sa informácie šíria medzi snímkami z väčšieho časového rozpätia, čím metóda zabezpečuje lepšiu schopnosť udržania informácie o vzhľade objektu

[1]Skupina pixelov, ktoré sú si veľmi podobné farbou a úrovňou jasu. Viac informácií:[41]

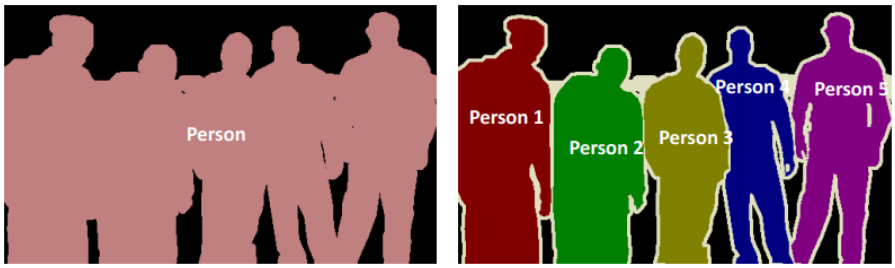
naprieč celým videom. Prepojenia na väčšiu vzdialenosť však spôsobujú aj výrazne vyššie výpočtové nároky, čo je možné obísť redšími prepojeniami - namiesto prepájania pixelov resp. malých oblastí sú grafovou metódou prepojené väčšie oblasti reprezentujúce samostatné objekty alebo ich väčšie segmenty. Tento postup má nevýhodu v znížení schopnosti segmentácie na úrovni detailov. Vstupy a výstupy týchto modelov sú rovnaké ako v prípade modelov založených na lokálnom šírení informácií.

Segmentácia inštancií

Modely z tejto kategórie generujú segmentačné masky pre každú snímku osobitne bez využitia informácií z predchádzajúcich snímok ani z ich segmentačných masiek. Výstupom modelu v jednom kroku je maska, ktorá predstavuje vždy iba jednu vysegmentovanú inštanciu objektu. Kombináciou niekoľkých krokov tak tento model vygeneruje segmentačnú masku všetkých inštancií objektov, ktoré bol model schopný detegovať. Tieto riešenia nevyužívajú informácie z predchádzajúcich snímok ani z ich segmentačných masiek, takže aj keď netrpia na postupné nabaľovanie chyby z predchádzajúcich predikcií, ich výkonnosť je výrazne obmedzená absenciou informácií o objektoch z predchádzajúcich snímok. Konečným výstupom modelu sú segmentačné masky označujúce väčšinu alebo všetky inštancie objektov nachádzajúcich sa v jednotlivých snímkach videa.

Interaktívna segmentácia

Existujú situácie, kedy je potrebná vyššia presnosť segmentácie akú poskytujú súčasné SOTA riešenia, ktoré fungujú bez dohľadu alebo len s čiastočným dohľadom experta. Preto vzniklo niekoľko riešení, ktoré poskytujú expertovi možnosť kontinuálne zlepšovať anotáciu označovaním nesprávne klasifikovaných pixelov alebo celých oblastí. Model tak v priebehu spracovania sekvencie vylepšuje naučenú reprezentáciu sledovaného objektu na základe vstupov od experta v podobe jednoduchých náčrtov a označení. Výhodou je veľmi vysoká presnosť, nevýhodou samotná nutnosť experta kontrolovať celý proces segmentácie. Výstupom modelov sú segmentačné masky označujúce tie objekty, ktoré počas práce modelu manuálne označil expert. Tieto kategórie nie sú navzájom exkluzívne a v praxi je možné stretnúť sa s kombináciami dvoch alebo výnimočne aj viacerých spomenutých konceptov. Je takisto dôležité spomenúť, že väčšina týchto riešení funguje na princípe



Obr. 2.3: Rozdiel medzi sémantickou segmentáciou (vľavo) a segmentáciou inštancií (vpravo). Zdroj[2]

sémantickej segmentácie, čo znamená, že model označuje aj niekoľko objektov z rovnakej kategórie ako jeden objekt. Výnimkou sú modely fungujúce na báze segmentácie inštancií, kde model označuje každú inštanciu objektu samostatne. Rozdiel medzi sémantickou segmentáciou a segmentáciou inštancií je zobrazený na Obr.2.3.

Podľa potrebného zásahu experta do procesu segmentácie sa modely často rozdeľujú do troch skupín:

1. **Segmentácia bez dohľadu.** Tieto modely nevyžadujú na vstupe nič okrem testovaného videa a masky, ktoré sú ich výstupom, segmentujú všetky alebo väčšinu objektov vo videu.

2. **Segmentácia s čiastočným dohľadom.** Na vstupe týchto modelov je okrem testovaného videa aj ground truth segmentačná maska pre prvú snímku, ktorú musel manuálne vytvoriť expert a ktorá slúži na bližšiu špecifikáciu objektu, ktorý experta zaujíma.

3. **Interaktívna segmentácia.** Tento typ segmentácie je bližšie popísaný na predchádzajúcej strane.

2.3 Úvod do neurónových sietí

Pre pochopenie konceptu neurónových sietí je nevyhnutné v skratke popísať proces strojového učenia. Pojem „strojové učenie“ vznikol z otázky: mohol by jedného

[2]https://towardsdatascience.com/understanding-semantic-segmentation-with-unet-6be4f42d4b47

dňa počítač prekročiť hranicu, kedy vykonáva úlohy, ktoré mu zadefinoval človek, a vyriešiť úlohu tak, že by k nej sám našiel riešenie? Táto a niekoľko ďalších otázok priniesli nový pohľad na proces programovania. V klasickom programovaní používateľ zadá dáta a pravidlá, podľa ktorých má počítač tieto dáta spracovať a po spracovaní dát vrátiť na výstup riešenie problému. V procese strojového učenia však používateľ zadá dáta vrátane očakávaných odpovedí a program sa sám naučí pravidlá spracovania a na výstupe vráti tieto pravidlá, ktoré je možné využiť neskôr pri riešení podobného problému s inou dátovou množinou. [15]

Systém strojového učenia teda nemá striktne naprogramované pravidlá, ktorými sa bude riadiť pri riešení problému, naopak, je schopný samostatne sa naučiť reprezentácie z dát, ktoré mu umožnia automaticky riešiť rovnaký problém na rôznych vstupných dátach. Strojové učenie vyžaduje pre svoju funkčnosť tri prvky:

1. Trénovacie dáta - texty, obrázky, zvukové stopy, číselné množiny alebo dáta v ďalších formátoch.

2. Očakávané výsledky - napríklad v prípade klasifikácie objektov na fotkách budú očakávané výsledky predstavovať názvy objektov nachádzajúcich sa na týchto fotkách.

3. Spôsob ohodnotenia funkčnosti algoritmu - presnosť výsledkov algoritmov strojového učenia je potrebné overiť, na čo slúžia rôzne matematické aparáty. Ich použitím získava algoritmus informáciu o tom, do akej miery sú ním vygenerované výsledky podobné reálnym hodnotám. Práve tento prvok predstavuje samotné učenie sa algoritmu.

Algoritmy strojového učenia sa v súčasnosti využívajú v takmer všetkých oblastiach nášho každodenného života na riešenie mnohých problémov, ktoré môžeme rozdeliť do troch skupín:

1. Problémy dolovania z dát, kde veľké databázy obsahujú potenciálne užitočné súvislosti, ktoré je možné využiť pri riešení konkrétnych problémov. Príkladom môže byť zisťovanie preferencií zákazníkov elektronických obchodov alebo určenie všeobecných pravidiel pri zisťovaní kredibility záujemcov o pôžičky.

2. Problémy, pre ktorých riešenie človek nie je schopný vytvoriť dostatočne presné algoritmy. Ako príklady uvedieme detekciu tvárí na fotkách alebo transformáciu hovorenej reči do textu v reálnom čase.

3. Problémy s neustále meniacimi sa podmienkami, napríklad predikcia vývoja cien na burze alebo logistické plánovanie.

Existuje niekoľko spôsobov kategorizácie algoritmov strojového učenia, no najčastejšie sa rozdeľujú podľa ich určenia do štyroch, nie nevyhnutne exkluzívnych kategórií:

1. **Učenie s učiteľom.** Vstupmi algoritmu sú trénovacie dáta X a k nim zodpovedajúce pravdivé hodnoty y. Algoritmus sa naučí reprezentáciu dát a závislosti medzi dátami tak, aby bol schopný čo najpresnejšie predikovať hodnoty y pre zadané dáta X. Často využívané algoritmy: naivný Bayesov klasifikátor, rozhodovacie stromy, lineárna regresia, SVM, rôzne typy neurónových sietí a ďalšie.

2. **Učenie bez učiteľa.** Algoritmus sa na neoznačených dátach učí závislosti medzi dátami, ktoré by expert nebol schopný definovať. Tento prístup je vhodný napríklad v situáciách, kedy nie sú k dispozícii žiadne označené trénovacie dáta prípadne ak je potrebné zvýšiť presnosť iných klasifikačných alebo regresných algoritmov. Algoritmy z tejto kategórie sú vo väčšine prípadov založené na jednom zo štyroch princípov: zhlukovanie, detekcia anomálií, hľadanie asociácií a autoenkódery.[5]

3. **Kombinácia učenia s učiteľom a bez učiteľa.** V tomto prípade je k dispozícii malé množstvo označených a veľké množstvo neoznačených dát. Algoritmus sa z označených dát naučí aké rôzne predikcie môže generovať a na neoznačených dátach sa naučí hranice pre jednotlivé predikcie.[1] Dobrým príkladom algoritmu z tejto kategórie je GAN neurónová sieť - generatívna konkurenčná sieť.[7]

4. **Učenie založené na spätnej väzbe.** Algoritmus hľadá spôsob ako vyriešiť problém na neoznačených dátach, resp. v prostredí bez dopredu určených pravidiel, a to iba na základe spätnej väzby, ktorú dostáva za každý vykonaný krok. Týmto spôsobom je napríklad počítač schopný naučiť sa hrať rôzne hry

bez toho, aby mu človek akýmkoľvek spôsobom špecifikoval pravidlá hry. Príkladmi algoritmov sú Q-learning, Monte Carlo a rôzne implementácie hlbokých neurónových sietí v súčinnosti s inými algoritmami. V dobe písania tejto práce populárne algoritmy spoločnosti DeepMind, ktoré sú schopné hrať väčšinu PC hier lepšie ako človek, sú takisto založené na tomto type učenia.[6]

Z predchádzajúceho zoznamu je jasné, že neurónové siete nepredstavujú jeden konkrétny typ algoritmu, ale je možné implementovať ich mnohými spôsobmi, pričom sú schopné riešiť takmer všetky problémy z oblasti strojového učenia. Neurónovým sieťam sú venované nasledujúce podkapitoly.

2.3.1 Neurónová sieť a hlboké učenie

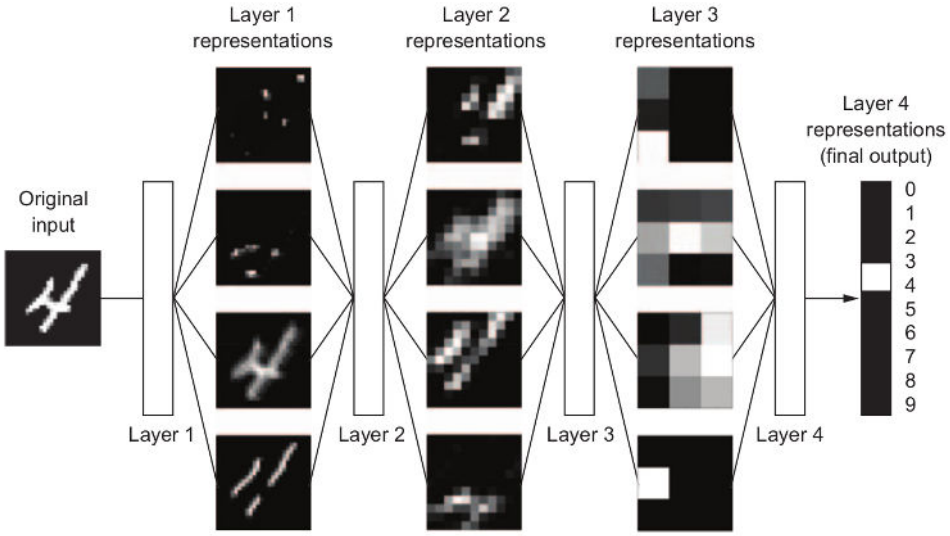
Kevin Gurney vo svojej knihe „An introduction to neural networks“[49] popisuje neurónovú sieť ako navzájom prepojenú skupinu jednoduchých procesných jednotiek, alebo aj uzlov, ktorých funkcionalita je inšpirovaná fungovaním neurónov v mozgu živočíchov. Procesná schopnosť tejto siete je uložená v prepojeniach medzi jednotlivými uzlami, ktoré sa odborne nazývajú váhy. Sieť nadobúda hodnoty svojich váh procesom postupného prispôsobovania sa, alebo aj učenia sa, na množine trénovacích vzorov.

Mozog a umelá neurónová sieť navzájom zdieľajú dve podobné vlastnosti:

1. Vedomosti o svojom okolí získavajú procesom učenia sa
2. Na uloženie získaných vedomostí slúžia váhy/prepojenia medzi jednotlivými (umelými) neurónmi

Okrem týchto podobností sú ale mozog a neurónová sieť výrazne odlišné. Hlavný rozdiel spočíva v tom, že veľká časť mozgu pracuje kontinuálne, zatiaľčo veľká časť neurónovej siete je iba pasívnym úložiskom dát. Mozog tak vykonáva množstvo procesov paralelne a je teda schopný využívať väčšinu svojej kapacity v každom momente. Druhou dôležitou výhodou mozgu je jeho schopnosť meniť svoju štruktúru počas svojho života, čo mu umožňuje efektívnejšie sa učiť a takisto kompenzovať chyby z minulosti.

Pri vytváraní neurónovej siete je nevyhnutné hneď na začiatku určiť jej veľkosť a



Obr. 2.4: Naučené reprezentácie modelu na rozpoznávanie písaných čísel. Zdroj:[15]

architektúru. Veľkosť siete je špecifikovaná počtom vrstiev v sieti, počtom uzlov v jednotlivých vrstvách a takisto v počte prepojení medzi jednotlivými uzlami v celej sieti. Vzhľadom na počet vrstiev rozlišujeme plytké a hlboké neurónové siete. Plytká sieť je zložená zo vstupnej a výstupnej vrstvy a z veľmi malého počtu skrytých vrstiev, zvyčajne iba z jednej. Naopak hlboké siete obsahujú väčšie množstvo skrytých vrstiev, čo im umožňuje naučiť sa reprezentácie, ktoré zodpovedajú niekoľkým úrovniam abstrakcie.

Na obrázku 2.4 je zobrazená hlboká sieť zložená zo štyroch skrytých vrstiev, ktorá transformuje obrázok na reprezentácie, ktoré sú postupne menej podobné pôvodnému obrázku a zároveň obsahujú viac informácií o očakávanom výsledku.

Existuje niekoľko typov uzlov - procesných jednotiek - ktoré je možné použiť na vytvorenie vrstiev, ktoré sú ďalej použité na vytvorenie samotnej siete. Architektúra siete je charakterizovaná práve typom použitých uzlov a vrstiev, počtom vrstiev a ich usporiadaním v sieti. Existuje tak niekoľko hlavných architektúr, z ktorých každá má svoje špecifické využitie, výhody a nevýhody. V nasledujúcich podkapitolách bude popísaných niekoľko architektúr, ktoré sú relevantné pre praktickú časť tejto diplomovej práce.

2.3.2 Konvolučné neurónové siete - CNN

Výskumníci D.H. Hubel a T.N. Wiesel uskutočnili v 50. a 60. rokoch 20. storočia výskum [53], ktorého výsledkom bol nový model vizuálneho vnímania u cicavcov. Hubel a Wiesel objavili vo vizuálnej časti mozgovej kôry mačiek a opíc neuróny, ktoré reagujú na malé oblasti zrakového poľa. Konkrétne objavili dva typy neurónov:

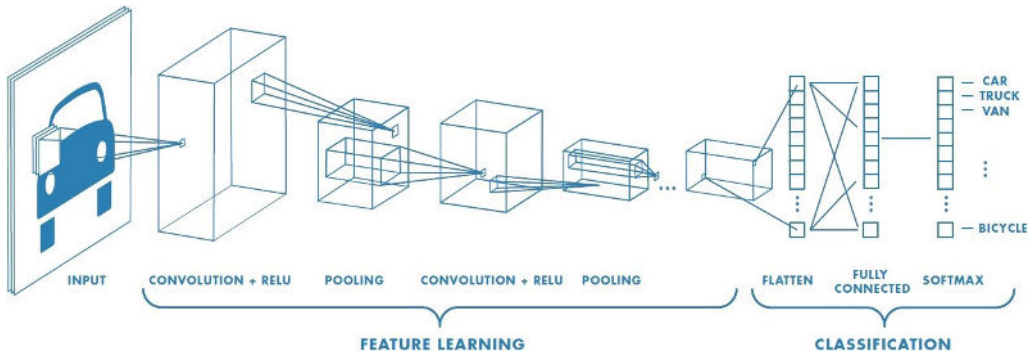
- Jednoduché neuróny (S cells) - aktivujú sa ak identifikujú jednoduché tvary, ideálne rovné čiary, vo veľmi malej pevne danej oblasti vnímania a v špecifickom uhle.
- Komplexné neuróny (C cells) - tieto neuróny majú väčšie pole vnímania a ich aktivácia nie je až tak závislá od polohy vnímaného tvaru v danom poli. Slovo *komplexné* v tomto prípade predstavuje viac flexibilné, keďže tieto neuróny reagujú na tvary nachádzajúce sa v rôznych častiach zrakového poľa.

Kunihiko Fukushima, inšpirovaný spomenutým výskumom, predstavil v roku 1980 tzv. neokognitron, čo je hierarchická viacvrstvová neurónová sieť, ktorú je možné využiť napríklad na úlohy rozpoznávania rukou písaného textu. Dve dôležité súčasti tejto siete - konvolučné vrstvy a downsampling vrstvy - tvoria základ v súčasnosti využívaných CNN. [52]

Na základe práce od K. Fukushima vyvinul Yann LeCun v 90.rokoch prvú konvolučnú neurónovú sieť, ktorá sa dokázala sama naučiť koeficienty konvolučných masiek a to využitím princípu spätnej propagácie. Táto sieť bola schopná rozpoznávať rukou písané čísla.

CNN je viacvrstvová neurónová sieť špecificky navrhnutá pre rozpoznávanie útvarov v dvojdimenzionálnom obraze s vysokou mierou robustnosti voči premenlivosti obrazu - zmena mierky, natočenia, priblíženia obrazu a podobne. Využíva sa na riešenie konkrétnych úloh ako je klasifikácia a segmentácia obrazu, detekcia objektov, rozpoznávanie znakov a niekoľko ďalších. Mimo spracovania obrazu sa tieto siete začali využívať aj na spracovanie jazyka a ďalšie menej konvenčné úlohy.

Použitie tradičnej neurónovej siete s plne prepojenými vrstvami je v prípade obrazových vstupných dát výrazne neefektívne - aj obrázok s nízkym rozlíšením obsahuje veľké množstvo informácií v podobe pixelov, čoho následkom v plne



Obr. 2.5: Architektúra CNN. Zdroj:[4]

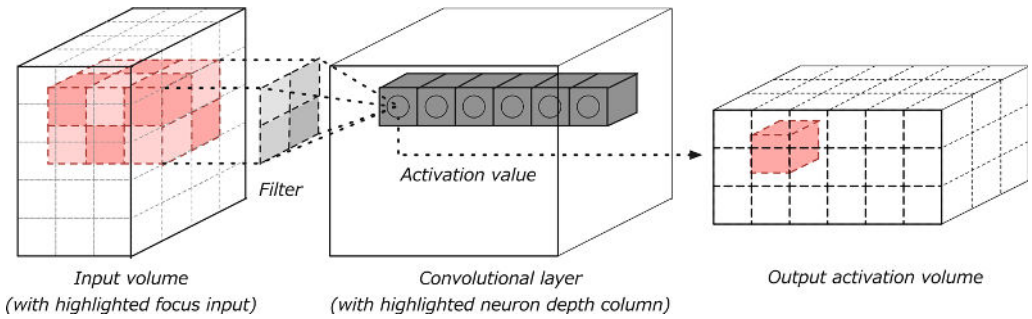
prepojenej sieti je neúmerné množstvo trénovateľných váh medzi neurónmi. To bol jeden z hlavných dôvodov vzniku CNN. Skryté vrstvy v CNN sú prispôsobené obrazovým dátam tým, že ich neuróny sú usporiadané v troch dimenziách: výška, šírka a hĺbka (hĺbka v tomto prípade reprezentuje farebný kanál). Ďalšou vlastnosťou CNN je, že neuróny v jednej vrstve nie sú prepojené so všetkými neurónmi v predchádzajúcej vrstve ale iba s neurónmi z malej oblasti tejto vrstvy. Špecifický je aj výstup CNN, ktorý je zredukovaný na vektor hodnôt s dĺžkou n, ktoré predstavujú pravdepodobnosti klasifikácie vstupných dát do jednej z n tried.[2]

Architektúra CNN určenej na úlohu klasifikácie obrazu, ktorá je zobrazená aj na Obr.2.5, je rozdelená na dve hlavné časti (nerátajúc vstupnú vrstvu):

1. **Extrakcia príznakov.** V tejto časti prebieha extrakcia príznakov z vstupných dát. Celá časť siete je zložená z niekoľkých menších blokov, ktoré sú zložené z vrstiev dvoch typov - konvolučné a pooling (združovacie). Vstupom do tejto časti sú obrazové dáta a výstupom sú informácie o príznakoch nájdených v dátach vrátane lokalizácie týchto príznakov.

2. **Klasifikácia.** Táto časť je zložená z niekoľkých úplne prepojených vrstiev, ktoré fungujú ako klasifikátor na základe príznakov extrahovaných v predchádzajúcich vrstvách. Výstupom sú hodnoty pravdepodobnosti výskytu konkrétnych objektov na obrázku.

Našou úlohou v tejto práci nie je klasifikácia obrazu ale jeho segmentácia, preto zvyšok tejto podkapitoly neurónovým vrstvám, ktoré slúžia na extrakciu príznakov, keďže tie tvoria základ väčšiny riešení segmentácie obrazu.



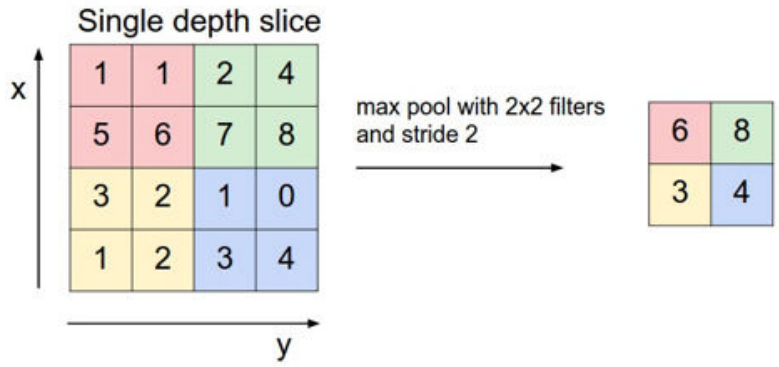
Obr. 2.6: Fungovanie konvolučnej vrstvy. Zdroj:[13]

Konvolučná vrstva

Konvolučné vrstvy slúžia na extrakciu príznakov z dát a zohrávajú kľúčovú úlohu v CNN. Základným stavebným kameňom konvolučnej vrstvy je matematická operácia zvaná konvolúcia. V prípade využitia CNN na obrazových dátach sa konvolúcia vykonáva využitím konvolučnej masky (filtra) s ľubovoľným rozlíšením (napr. 3×3), ktorá sa posúva postupne ponad každý pixel vstupného obrazu. V každom kroku sa matica masky a matica prináležiacich pixelov obrázku pod touto maskou vynásobia maticovým násobením. Výsledok každého násobenia sa nakoniec zapíše do mapy príznakov, ktorá je výstupom pre danú konvolučnú masku. Takýchto masiek sa bežne v jednej konvolučnej vrstve kombinuje niekoľko, napr. 10. Rozmery jednej takto poskladanej vrstvy by tak boli $3 \times 3 \times 10$. Fungovanie konvolučnej siete je zobrazené aj na Obr.2.6.

Použitie niekoľkých skrytých konvolučných vrstiev umožňuje CNN naučiť sa príznaky zo vstupných dát na niekoľkých úrovniach abstrakcie. To znamená, že na prvých vrstvách sa sieť učí rozpoznávať jednoduché tvary ako napr. hrany a jednoduché geometrické útvary. V neskorších vrstvách spája vstupné informácie a učí sa identifikovať jednotlivé časti objektov ako koleso na aute, oko človeka a podobne. V posledných vrstvách (za podmienky, že sieť je zostavená z vhodného počtu vrstiev) sa CNN učí rozpoznávať objekty v celku - ľudí, autá, psy, atď.

Vďaka schopnosti siete extrahovať príznaky z niekoľkých úrovní abstrakcie môže používateľ natrénovať sieť na veľkej dátovej množine, zmraziť všetky skryté vrstvy okrem malého počtu vrstiev na konci, ktoré sú určené na identifikáciu a lokalizáciu už konkrétnych objektov, a tieto vrstvy dotrénovať pre konkrétnu doménu objektov. Týmto spôsobom je možné vytvárať modely určené na riešenie veľmi špecifických



Obr. 2.7: $MAX()$ pooling vyberá najvyššiu hodnotu z každého okna s rozmermi 2×2. Zdroj:[2]

úloh, napríklad identifikáciu nádorov na RTG snímkach bez nutnosti trénovať celú sieť od začiatku.[2]

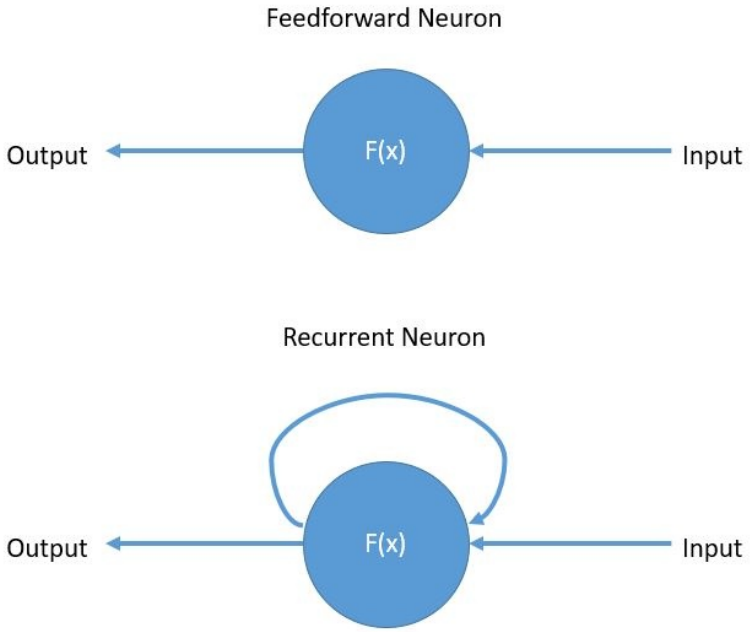
Pooling vrstva

S postupným pridávaním konvolučných vrstiev exponenciálne narastá počet parametrov celej siete, ktoré je potrebné trénovať. Preto sa za konvolučné vrstvy vkladajú pooling vrstvy, ktorých úlohou je kontinuálne redukovať dimenzionalitu siete znižovaním počtu parametrov. Následkom je kratší čas trénovania modelu a lepšia prevencia pred preučením.

Operácia pooling-u, podobne ako konvolúcia, sa vykonáva nad celým vstupným obrazom postupným prechádzaním ponad jednotlivé pixely. Namiesto konvolučnej masky sa využíva jednoduché „okno“, ktorého veľkosť, rovnako ako rozmery konvolučnej masky, vopred zadefinuje používateľ. Toto okno, s rozmermi napr. 2×2, prechádza ponad jednotlivé skupiny pixelov a podľa typu pooling-u vykonáva matematickú operáciu s hodnotami pixelov nachádzajúcich sa pod týmto oknom. V praxi sa používajú dve operácie - $MAX()$, ktorá vyberie zo skupiny hodnôt tú najvyššiu a $AVG()$, ktorá hodnoty spriemeruje. Pre ukážku $MAX()$ pooling-u viď Obr.2.7.[17]

2.3.3 Rekurentné neurónové siete - RNN

Predikcia na sekvenčných dátach predstavuje v dátovej analytike špecifický typ problému. Sekvenčné dáta je možné popísať ako zoradenú dátovú množinu, v



Obr. 2.8: Rozdiel medzi neurónmi doprednej a rekurentnej siete. Zdroj[3]

ktorej navzájom súvisiace prvky nasledujú za sebou. Príkladom môžu byť DNA sekvencie, časové sekvencie, v ktorých sú jednotlivé prvky zoradené podľa časového údaju alebo pre našu prácu relevantné videosekvencie.

V doprednej neurónovej sieti putujú informácie vždy iba v jednom smere - zo vstupnej vrstvy, cez skryté vrstvy až do výstupnej vrstvy. Každá informácia prechádza priamo cez sieť a nikdy sa nevracia k uzlu, cez ktorý už prešla. Takáto sieť si nedokáže zapamätať informácie extrahované z minulých vstupov, pracuje vždy iba s aktuálnym záznamom a nedisponuje informáciou o čase plynúcom v dátach. Tieto vlastnosti predurčujú dopredné siete na riešenie úloh ako rozpoznávanie tvárí na statických obrázkoch alebo klasifikácia na základe statických dát.

Naproti tomu vstupom do rekurentnej siete nie je len aktuálna vzorka z dátovej množiny ale aj informácie o predchádzajúcich vzorkách, ktoré touto sieťou prešli. Informácie v rekurentnej sieti putujú cyklicky a táto sieť tak nadobúda schopnosť pamätať si informácie a súvislosti z minulosti. Rozdiel medzi fungovaním neurónu

[3]https://wiki.tum.de/display/lfdv/Recurrent+Neural+Networks+-+Combination+of+RNN+and+CNN

doprednej a rekurentnej siete je zobrazený na obr. 2.8

Ak je vstupom do siete sekvencia $(x_1, ..., x_T)$ tak štandardná RNN vypočíta výstupnú sekvenciu $(y_1, ..., y_T)$ iteráciou nasledujúcich vzťahov:

$$h_t = sigm(W_{xh}x_t + W_{hh}h_{t-1}) \tag{2.4}$$

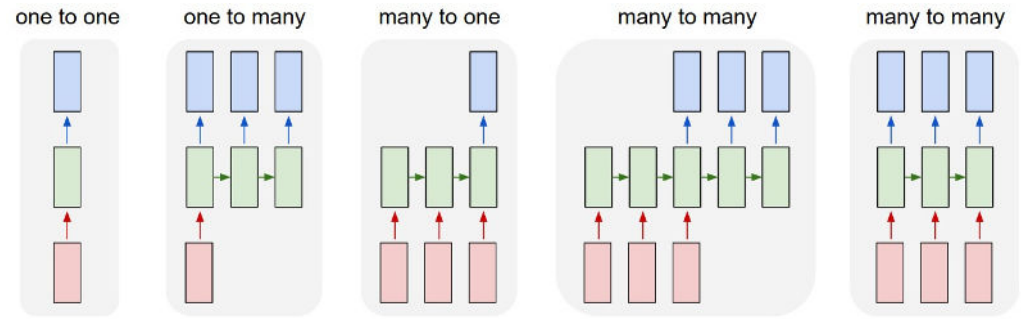
$$y_t = W_{yh}h_t \tag{2.5}$$

kde:

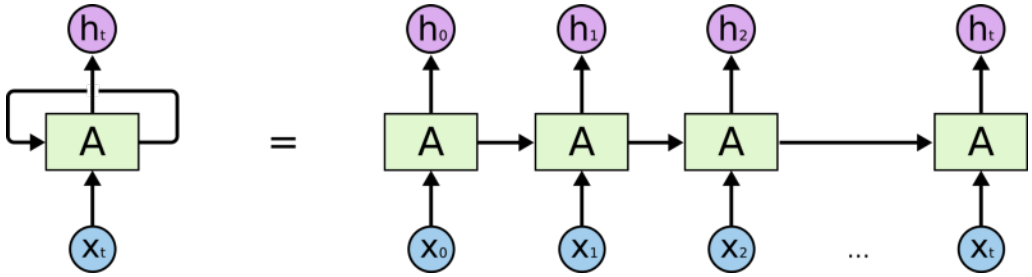
- h_t - stav skrytej vrstvy v čase t,
- y_t - prvok z výstupnej sekvencie y v čase t,
- $sigm$ - sigmoidná funkcia skrytej vrstvy,
- W_{xh} - váhy medzi vstupnou a skrytou vrstvou,
- W_{hh} - váhy medzi skrytými vrstvami,
- W_{hy} - váhy medzi skrytou a výstupnou vrstvou. [38]

RNN akceptuje na vstupe vektor x, na základe ktorého generuje výstupný vektor y. Vektor y však nie je ovplyvnený iba vstupom x, ale z veľkej časti aj celou históriou vstupov, ktoré sieť do tohto momentu spracovala. Súčasťou RNN je skrytá vrstva, ktorá sa aktualizuje každým novým vstupom, a ktorá v sebe uchováva informácie zo všetky spracovaných vstupov. V najjednoduchšom prípade predstavuje túto skrytú vrstvu vektor h. Na vstupe RNN je vždy sekvencia $x_1, x_2, ..., x_t$. Na výstupe je potom sekvencia $y_1, y_2, ..., y_t$. Prvkami sekvencie sú vždy jedno alebo viacrozmerné číselné vektory. Z uvedeného príkladu vyplýva, že do RNN vstupujú vždy dve informácie - aktuálna vzorka v podobe číselného vektoru x_t a informácia o predchádzajúcich vzorkách spracovaných sieťou v podobe vektoru h_t. Toto je kritickou vlastnosťou RNN, pretože jej to umožňuje získavať zo sekvencií informácie o minulosti a tým pádom presnejšie predikovať budúce udalosti. Druhou výhodou je, že narozdiel od doprednej siete, ktorá dokáže mapovať iba jeden vstup na jeden výstup (angl. one to one), RNN dokáže mapovať (Obr.2.9):

- jeden vstup na viac výstupov (angl. one to many) - viacslovný popis prostredia/objektov na jednom obrázku



Obr. 2.9: Alternatívy mapovania vstupných vektorov na výstupné. Zdroj[4]



Obr. 2.10: RNN sieť rozbalená v čase t. Zdroj[5]

- viac vstupov na jeden výstup (angl. many to one) - analýza sentimentu, kde viacslovný text klasifikujeme ako pozitívny/negatívny
- viac vstupov na viac výstupov (angl. many to many) - preklad textu
- synchronizovaný vstup a výstup (angl. many to many) - klasifikácia videa, kde označujeme každý snímok osobitne.[3]

Učenie neurónových sietí prebieha procesom zvaným *spätná propagácia chyby*. V bežnej doprednej sieti sa chyba šíri od poslednej vrstvy až k prvej, čo následne zabezpečuje učenie jednotlivých vrstiev na základe vypočítaných gradientov týchto chýb. RNN je však cyklická sieť, to znamená, že na prvý pohľad nie je možné šíriť chybu spätne po vrstvách. Sieť je však možné rozbaliť v čase, čím získame sieť zloženú z n blokov, pričom n predstavuje počet časových krokov, ktoré sa v sieti nazbierali. Všetky bloky v tejto sieti zdieľajú rovnaké váhy. Obrázok 2.10 naznačuje ako tento proces vyzerá v praxi.

Gradient, ktorý sieť počíta v každom uzle siete zo spätne šírenej hodnoty chyby,

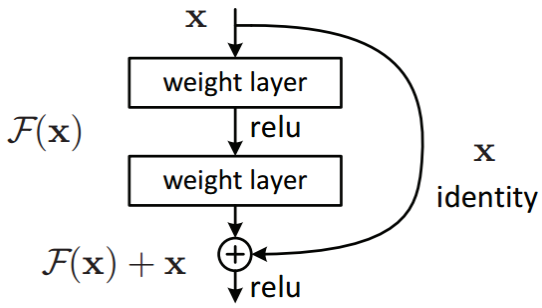
[4] http://karpathy.github.io/2015/05/21/rnn-effectiveness/
[5] http://colah.github.io/posts/2015-08-Understanding-LSTMs/

sa s rastúcim počtom vrstiev v sieti exponenciálne zmenšuje. V doprednej sieti s desiatkami vrstiev tak vzniká problém s tréningom vrstiev na začiatku siete, keďže gradient v týchto vrstvách je už tak nízky, že nedokáže spôsobiť relevantné zmeny váh. Tento problém je bežne označovaný ako vymiznutie gradientu. Následkom tohto problému je výrazné spomalenie a v najhoršom prípade úplné zastavenie učenia sa modelu. Rovnaký problém existuje aj v RNN a vzniká v prípade, keď je snaha naučiť model príliš vzdialené závislosti medzi vstupom a výstupom.[3]

Vymiznutie gradientu v rekurentných sieťach bolo vyriešené vynájdením tzv. LSTM (angl. Long-Short Term Memory) sietí. LSTM predstavuje rozšírenie klasickej RNN, ktorého cieľom je predĺžiť pamäť RNN, čím umožňuje sieti naučiť sa závislosti, ktoré sú od seba na časovej osi príliš vzdialené. LSTM ukladá informácie do pamäte, ktorá je konceptuálne podobná bežnej pamäti v počítači, t.j. sieť dokáže informácie čítať, zapisovať aj mazať. O tom, ktoré informácie uložiť a ktoré vymazať rozhoduje sieť, ktorá sa postupne učí ktoré informácie sú dôležité a ktoré môžu byť vymazané. V tejto pamäti je tak možné uchovať dôležité informácie aj z ďalekej minulosti sekvencie.[50]

2.3.4 Reziduálne neurónové siete

Hlboké dopredné neurónové siete viedli v posledných rokoch k veľkým úspechom v riešení úloh ako je klasifikácia obrazu, segmentácia obrazu a veľa ďalších. Hlavným dôvodom úspechu je schopnosť týchto sietí extrahovať zo vstupných dát informácie na rôznych úrovniach abstrakcie - od jednoduchých príznakov ako hrany na obrázku po celé objekty, od foném v hovorenom texte po celé slová. S prehlbovaním siete vkladaním ďalších skrytých vrstiev však vznikajú dva problémy. Prvým je problém miznutia gradientu [51], ktorý sa prejavuje rovnako ako v RNN - so zvyšujúcim sa počtom vrstiev v sieti spôsobuje zmenšovanie gradientu, a to až na takú úroveň, kedy sieť stráca schopnosť učiť sa. Tento problém bol adresovaný a vo veľkej miere vyriešený použitím ReLU aktivačnej funkcie vrátane inicializácie normalizovaných váh [43] a implementáciou normalizačných vrstiev - *batch normalization* - do siete [35]. Po vyriešení miznúceho gradientu sa objavil ďalší problém - degradácia presnosti [36], ktorý sa prejavuje paradoxom klesajúcej presnosti pri trénovaní so zväčšujúcou sa hĺbkou siete. Výskumníci tento problém adresovali vytvorením reziduálnych neurónových sietí.[34]



Obr. 2.11: Reziduálny blok. Zdroj:[34]

Základnou stavebnou jednotkou reziduálnych sietí, skrátene ResNet, je reziduálny blok. V tradičnej neurónovej sieti putujú informácie z jednej vrstvy priamo do nasledujúcej vrstvy. V reziduálnej sieti putujú informácie rovnako z jednej do nasledujúcej vrstvy a zároveň do vrstvy vzdialenej 2-3 prepojenia dopredu. Zoskupenie vrstiev s týmto špecifickým typom prepojení sa nazýva reziduálny blok. Tento blok predstavuje základnú stavebnú jednotku reziduálnych neurónových sietí, skrátene ResNet. Ilustrácia reziduálneho bloku je zobrazená na Obr.2.11

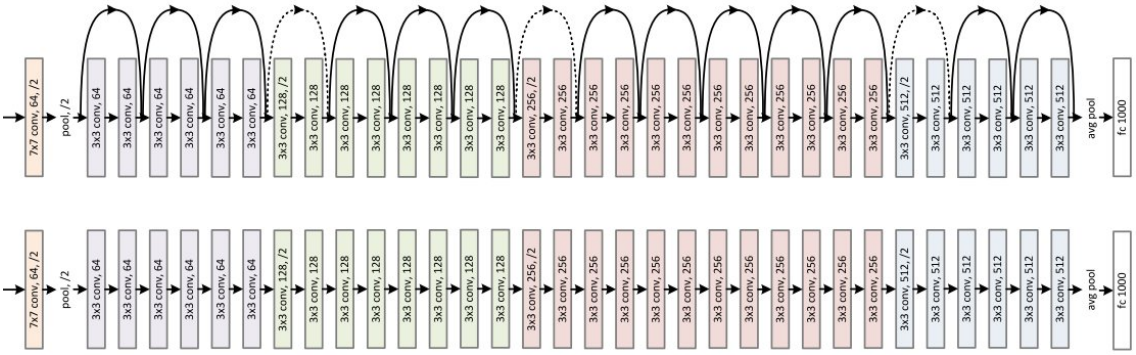
Experimenty ukazujú, že s rastúcou hĺbkou neurónovej siete najprv stúpa jej presnosť, až dosiahne svoje maximum a pridávaním ďalších vrstiev sa jej presnosť znižuje. Takto hlboké siete už dokonca nie sú schopné naučiť sa ani tzv. identickú funkciu, čo je vzhľadom na určenie neurónových sietí jednoznačne nežiadúci efekt. Porovnanie medzi tradičnou a reziduálnou sieťou je zobrazené na Obr.2.12.

Pre ilustráciu fungovania reziduálneho bloku poslúži nasledujúci príklad. Máme vytvorenú neurónovú sieť, ktorej vstupom je identická funkcia x a úlohou tejto siete je naučiť sa rozloženie $H(x)$. Tradičná sieť by sa snažila všetky svoje vrstvy namapovať na funkciu x, čo by pri väčšom množstve vrstiev mohlo zlyhať. Reziduálna sieť sa takisto snaží naučiť reálnu distribúciu $H(x)$, ale využíva pri tom tzv. skip prepojenie (prepojenie medzi vrstvami, ktoré v sieti nie sú uložené priamo za sebou), cez ktoré prenáša identickú funkciu x a samotné vrstvy v reziduálnom bloku sa mapujú už iba na tzv. rezíduum $F(x)$, ktoré definujeme nasledovne:

$$F(x) = H(x) - x \tag{2.6}$$

a otočením dostaneme rovnicu:

$$H(x) = F(x) + x \tag{2.7}$$



Obr. 2.12: Porovnanie reziduálnej (hore) a tradičnej siete (dole). Zdroj:[34]

Výskumníci, ktorí vynašli reziduálne siete pozorovaním zistili, že je pre sieť jednoduchšie namapovať sa na rezíduum $F(x)$ ako na pôvodnú distribúciu $H(x)$. V prípade potreby namapovať reziduálny blok iba na identickú funkciu x je to teoreticky možné jednoducho definovaním rezídua na nulu.[34]

Použitie skip prepojenia umožňuje sieti šíriť do prvých vrstiev vyššie hodnoty gradientov, čím sa tieto vrstvy učia rovnako rýchlo ako vrstvy na konci siete. Použitie reziduálnych blokov má takisto za následok obohatenie siete o „dynamický charakter“ - používateľ takmer nikdy nevie s istotou určiť ideálny počet vrstiev v svojom modeli, vložením skip prepojenia medzi vrstvy však umožní svojej sieti preskočiť proces trénovania pre tie vrstvy v sieti, ktoré nie sú nijak prospešné a nezvyšujú jej celkovú presnosť. Sieť tak vie lepšie reagovať na rôznu zložitosť vstupných dát a optimalizovať svoje fungovanie.

Siete postavené na architektúre ResNet alebo na konceptoch z tejto architektúry sa využívajú najmä na riešenie úloh, ktoré si vyžadujú veľmi hlboké neurónové siete. Do tejto oblasti patria jednoznačne úlohy zamerané na prácu s obrazom, keďže práve obraz obsahuje veľké množstvo informácií na rôznych úrovniach abstrakcie. Od vzniku prvej verzie ResNet siete ubehlo niekoľko rokov a za tento čas vzniklo mnoho alternatívnych riešení, ako napríklad ResNeXt. ResNeXt zavádza nový hyperparameter zvaný *kardinalita*, ktorý reprezentuje počet nezávislých ciest, inak povedané počet nezávislých zoskupení vrstiev v jednom reziduálnom bloku s jedným skip prepojením. Vznikajú aj ďalšie alternatívy a samotný koncept rezíduí a skip prepojení je obľúbený medzi výskumníkmi, ktorí hľadajú nové prístupy a riešenia v oblasti hlbokého učenia.

3 Analýza súčasného stavu

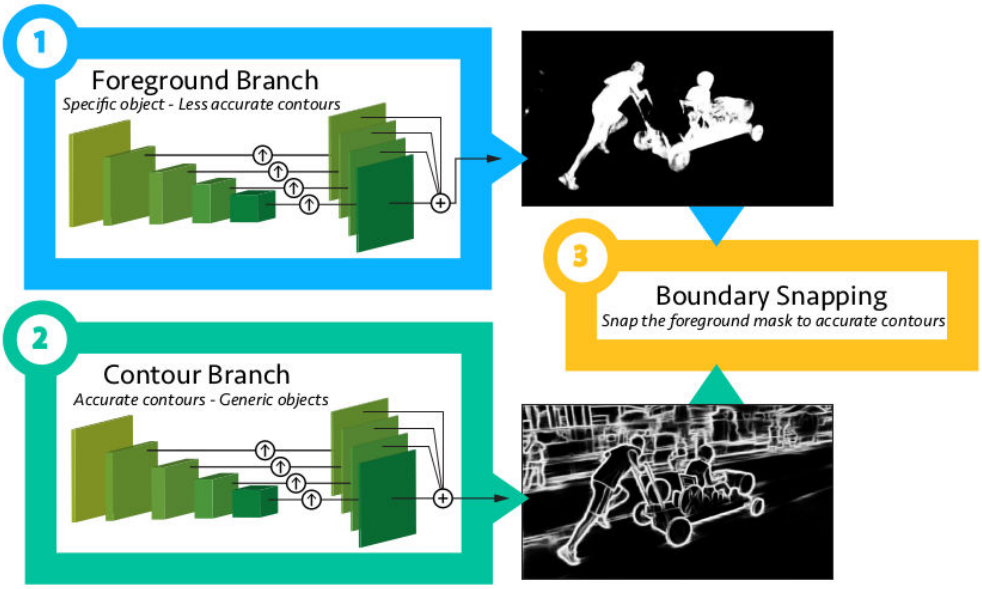
Problém segmentácie videa je v súčasnosti možné riešiť viacerými technikami od prahovania až po hlboké neurónové siete. Táto práca je zameraná na metódy hlbokého učenia, preto budú v nasledujúcich podkapitolách bližšie popísané dve hlavné architektúry hlbokého učenia, ktorých variácie sa využívajú vo väčšine ďalších riešení segmentácie videa a ktoré aj v svojej základnej podobe dosahujú veľmi dobré výsledky.

3.1 One Shot Video Object Segmentation - OSVOS

3.1.1 Architektúra

Základom architektúry OSVOS [26] je konvolučná sieť, ktorá bola pôvodne vytvorená za účelom detekcie a klasifikácie objektov v obraze. V prípade prvej implementácie architektúry OSVOS bola použitá sieť VGG. Implementácia tejto alebo podobnej siete zabezpečuje schopnosť identifikácie príznakov objektov v obraze, čím vytvára veľmi dobrú inicializáciu pre následnú úlohu segmentácie.

Sieť VGG je zložená z niekoľkých blokov konvolučných + ReLU vrstiev, pričom po každej skupine nasleduje pooling vrstva zmenšujúca mapu príznakov. Na konci siete je niekoľko úplne prepojených vrstiev, ktoré slúžia na úlohu klasifikácie. Tieto vrstvy boli pri implementácii do architektúry OSVOS odstránené. Ďalej bola sieť VGG upravená tak, že na výstupe každého bloku Conv+ReLU je vytvorený preskok, jeho výstup je zväčšený na jednotný rozmer a tento výstup s výstupmi z ostatných blokov je na konci spojený do jednej mapy príznakov z rôznych úrovní abstrakcie vstupného obrazu. Rozmery tejto mapy sú rovnaké ako rozmery vstupného obrazu. Na koniec siete je pridaná chybová funkcia určená na binárnu klasifikáciu, kde jedna trieda predstavuje pozadie a druhá sledovaný objekt.



Obr. 3.1: Architektúra OSVOS zložená z dvoch modulov. Zdroj:[26]

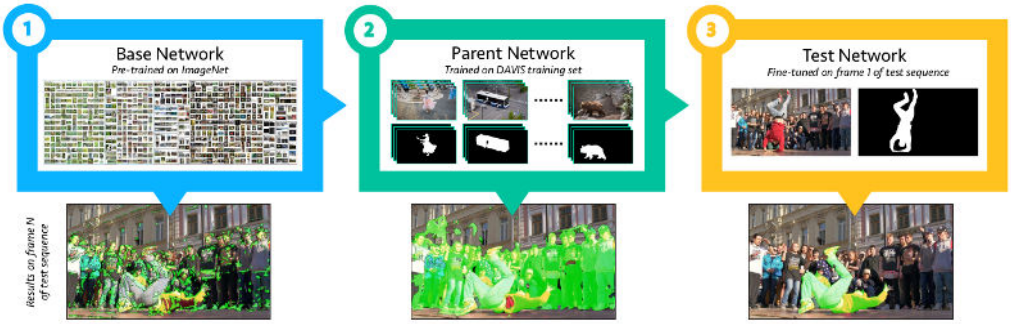
Architektúra OSVOS je vytvorená tak, aby dokázala segmentovať objekty na samostatných snímkach a výsledok segmentácie by mal byť rovnaký nezávisle od toho, kde v obraze sa cieľový objekt nachádza. Tento prístup je výhodný pri segmentácii samostatných fotografií, v úlohe segmentácie videa však necháva priestor pre využitie informácií o pozícii objektu v priebehu sekvencie. Táto príležitosť je využitá implementáciou ďalšieho modulu, ktorý slúži na detekciu všetkých kontúr v obraze. Tento modul je postavený na rovnakej architektúre ako prvý, klasifikačný modul, využíva však inú chybovú funkciu. Využitie tohto modelu znižuje rýchlosť ale zvyšuje presnosť celej architektúry. Obidva moduly sú zobrazené na Obr.3.1.

3.1.2 Trénovanie

Trénovanie modelu prebieha v dvoch fázach:

1. **Offline trénovanie.** Konvolučná sieť, ktorá tvorí základ tejto architektúry je predtrénovaná na dátovej množine ImageNet[1] zloženej z viac ako 1 milióna anotovaných obrázkov. Sieť v tejto fáze bez ďalšieho trénovania nie je schopná segmentácie, preto musí byť jednorazovo natrénovaná na množine anotovaných videí, čím sa naučí všeobecný koncept segmentácie objektov od pozadia.

[1] `http://www.image-net.org/`



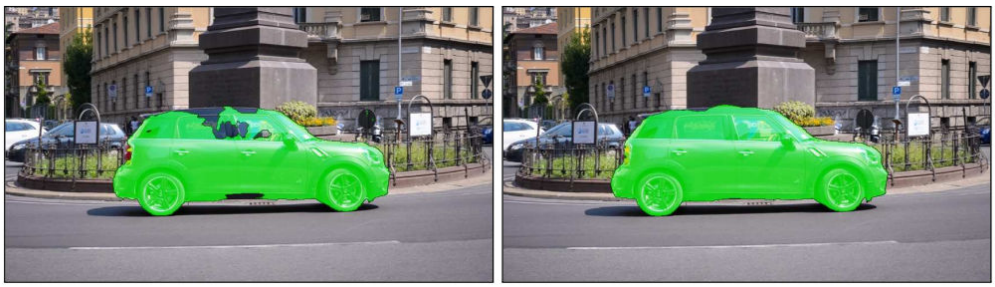
Obr. 3.2: Priebeh trénovania a testovania v architektúre OSVOS. Zdroj:[26]

Pre rozšírenie dátovej množiny sú pred trénovaním tieto dáta rozmnožené použitím zrkadlenia a približovania obrazu. Sieť v tejto fáze - po natrénovaní - už dokáže samostatne segmentovať objekty od pozadia.

2. **Online trénovanie a testovanie.** V tejto fáze sa model dotrénuje na dvojici prvej snímky testovacieho videa a jej segmentačnej masky. Týmto dotrénovaním sa model naučí vzhľad objektu, ktorý chceme sledovať a segmentovať od pozadia a túto informáciu potom využíva pri segmentovaní nasledujúcich snímok. Z časového hľadiska je tak potrebné v tejto fáze brať do úvahy dva časy - čas trénovania na prvej dvojici snímka+segmentačná maska a čas, ktorý model potrebuje na predikovanie segmentačných masiek pre nasledujúce snímky. Tento "predikčný"čas je závislý iba od HW, na ktorom testovanie prebieha, prvý čas je však možné definovať manuálne a práve od neho závisí, ako dobre sa model naučí reprezentáciu cieľového objektu a ako presne dokáže daný objekt segmentovať vo zvyšku videa. Rozdiel v kvalite segmentácie vzhľadom na trvanie dotrénovania je zobrazený na Obr.3.3.

3.1.3 Použitie a vlastnosti

Dôležitý prínos architektúry OSVOS je schopnosť naučiť konvolučnú sieť rozpoznávať konkrétnu inštanciu objektu iba pomocou prvej anotovanej snímky. Architektúra je postavená na konvolučnej sieti určenej na klasifikáciu obrazu, táto sieť je natrénovaná na segmentovaných videách a dotrénovaná na dvojici obraz+maska, čím sa naučí segmentovať konkrétnu inštanciu objektu. Výhodou tohto prístupu je práca siete na niekoľkých úrovniach abstrakcie - od všeobecných sémantických informácií o rôznych kategóriách objektov k detailom inštancie sledovaného objektu.



Obr. 3.3: OSVOS - rozdiel v kvalite segmentácie vzhľadom na trvanie dotrénovania na prvej segmentovanej snímke - 10 sekúnd / 1 minúta. Zdroj:[26]

Architektúra OSVOS sa odlišuje od väčšiny iných prístupov aj v tom, že spracúva každú snímku videa samostatne a nevyužíva pri tom informácie zo zvyšku sekvencie. Výrazne sa tak zvyšuje presnosť modelu v situáciách, kedy sa vo videu objavujú pred objektom prekážky, objekt dočasne zmizne z dohľadu alebo sa objekt nepredvídateľne pohybuje. Väčšina iných prístupov sa spolieha na malú variáciu v zobrazení objektu medzi jednotlivými snímkami, čím sa výrazne znižuje ich presnosť na rýchlych videách a takmer úplne zlyhávajú pri segmentácii nekompletných videí. Nevýhodou týchto riešení je aj postupné nabaľovanie chyby segmentácie.

Ďalšou vlastnosťou je možnosť využiť viac ako jednu anotovanú snímku na dotrénovanie siete. Navýšením počtu použitých anotovaných snímok je možné výrazne zvýšiť presnosť modelu na úkor mierneho predĺženia trvania dotrénovania. Model je tak možné prispôsobiť situácii podľa požiadaviek používateľa na presnosť a takisto podľa jeho možností z pohľadu dodania viac ako jednej anotovanej snímky.

Poslednou dôležitou vlastnosťou architektúry OSVOS je jej pomer rýchlosť/presnosť a najmä možnosť manuálne tento pomer ovplyvňovať. Ako už bolo spomenuté, poslednou fázou pred testovaním modelu je jeho dotrénovanie na anotovanej prvej snímke. Používateľ má možnosť definovať trvanie tohto trénovania, čím výrazne ovplyvní aj konečnú presnosť modelu. Predĺženie dotrénovania zo 100ms na niekoľko sekúnd môže spôsobiť zvýšenie presnosti o viac ako 8 percent [26]. Takto môže používateľ prispôsobiť fungovanie modelu jeho potrebám a možnostiam.

To, že sa model postavený na architektúre OSVOS učí vzhľad cieľového objektu iba z prvej snímky, je aj jeho nevýhodou. Aj keď vďaka tomu dokáže segmentovať snímky osobitne, z prvej snímky sa naučí iba značne obmedzenú reprezentáciu

objektu, čo spôsobí zníženie presnosti v prípade výraznej transformácie vzhľadu objektu v priebehu videa. Inými slovami, model nevyužíva temporálne informácie zo zvyšku videa a nevie sa naučiť presnejšiu reprezentáciu sledovaného objektu.

3.1.4 Príklady implementácie

Medzi najzaujímavejšie implementácie patrí riešenie [23], ktoré je zložené z dvoch hlavných častí. Prvá časť využíva OSVOS na generovanie návrhov segmentácie na základe prvej anotovanej snímky a algoritmus FCIS [31] na generovanie návrhov na základe videných sémantických tried. Druhú časť predstavuje algoritmus multisegmentového sledovania - SPT [40], ktorý priradí návrh segmentácie objektu z algoritmu FCIS k návrhu z OSVOS časti, čím sa prepája proces lokalizácie sledovaného objektu s určením jeho hraníc deliacich ho od pozadia. Toto riešenie navyše využíva *Lucid data dreaming* techniku adaptovanú z riešenia [16], ktorá slúži na umelé rozšírenie dátovej množiny. Na spresnenie konečného výsledku segmentácie je použitá sieť Dense CRF [42]

Inou zaujímavou alternatívou využitia OSVOS architektúry je riešenie [11], ktorého zaujímavosťou je, že nevyužíva žiaden algoritmus zameraný na sémantické informácie obsiahnuté v obraze, takže je schopné podať lepšie výsledky aj pri segmentovaní doposiaľ nevidených kategórií objektov. Proces segmentácie tu prebieha v štyroch krokoch:

1. Na základe informácií o pohybe objektu je vygenerovaný návrh v podobe ohraničujúceho rámiku.

2. Kombináciou transformovanej masky z predchádzajúcej snímky a predikcie masky z OSVOS modelu je vytvorený hrubý návrh segmentačnej masky pre aktuálnu snímku.

3. Návrh je orezaný podľa rámiku z prvého bodu a táto časť obrazu je odoslaná na vstup do konvolučnej siete určenej na zdokonalenie výslednej masky.

4. Na celú sekvenciu segmentačných masiek sú použité Markovove náhodné polia, ktoré využívajú priestorové a časové informácie o sledovaných objektoch vo videu a na základe nich ešte viac zdokonalia predikované masky.

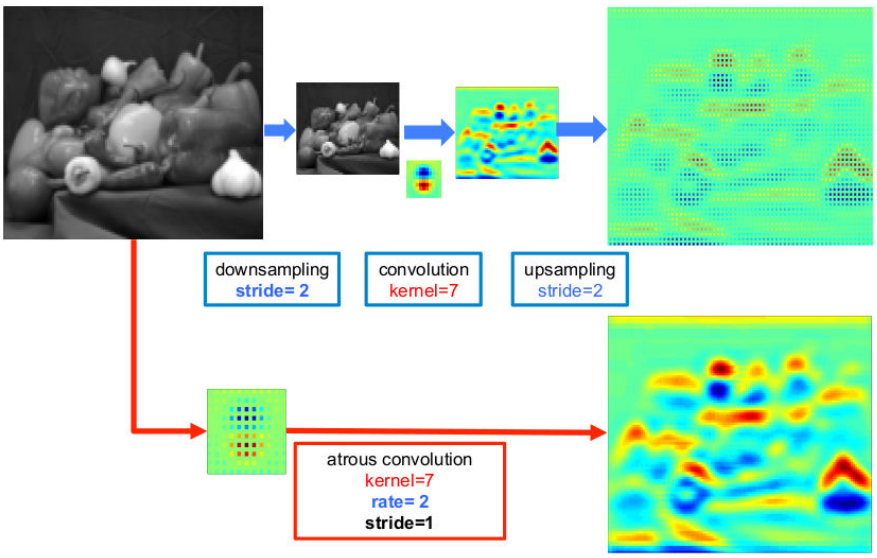
Ďalšie riešenia postavené na architektúre OSVOS sú napr. [24], [14], [8] a iné.

3.2 MaskTrack

MaskTrack [29] je ďalšia architektúra postavená na metódach hlbokého učenia, ktorá je určená na poloautomatickú segmentáciu videí. Narozdiel od architektúry OSVOS, MaskTrack využíva pri segmentovaní informácie z predchádzajúcich snímok a k nim vytvorených segmentačných masiek. Modely postavené na tejto architektúre dosahujú výsledky na úrovni SOTA riešení.

3.2.1 Architektúrau

Základom architektúry MaskTrack je model DeepLabv2 [27], ktorý slúži na sémantickú segmentáciu statických obrázkov. DeepLabv2 je, podobne ako OSVOS, postavený na konvolučnej sieti - VGG/Resnet natrénovanej na úlohu klasifikácie obrazu. Pre prispôsobenie siete na úlohu segmentácie sú posledné úplne prepojené vrstvy nahradené špecifickým typom konvolučných vrstiev. Použitie bežných konvolučných vrstiev vrátane pooling-u spôsobuje prudkú redukciu rozlíšenia mapy príznakov generovanej v jednotlivých vrstvách. Tento problém je adresovaný použitím konvolučných vrstiev (orig. atrous konvolúcia)[27], ktorých konvolučný filter má umelo navýšené rozmery, a to vložením nulových hodnôt medzi pôvodné hodnoty filtra. Fungovanie atrous konvolúcie je zobrazené na Obr.3.4.



Obr. 3.4: Porovnanie výsledku bežnej konvolúcie (hore) a atrous konvolúcie (dole). Zdroj:[27]

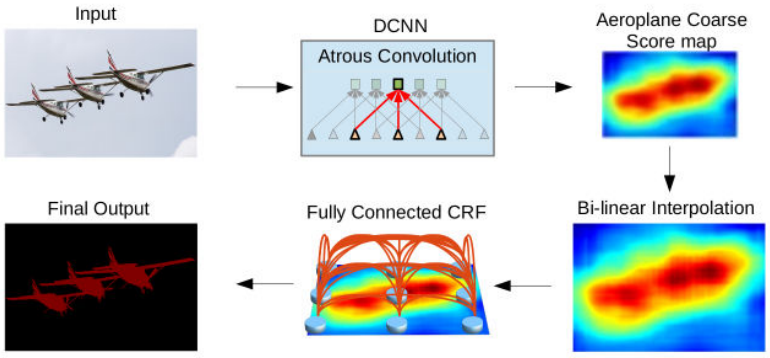
Dopad týchto úprav na fungovanie siete je nasledovný:

1. Je možné dynamicky meniť rozmery výslednej mapy príznakov zmenou rozmerov konvolučných filtrov v jednotlivých vrstvách.

2. Aj keď majú tieto filtre vyššie rozlíšenie, počet trénovateľných parametrov siete sa nemení, pretože relevantné sú iba nenulové hodnoty filtrov.

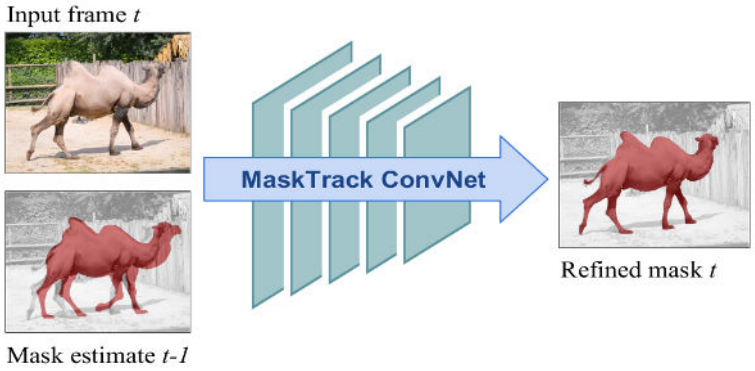
Ďalšia metóda implementovaná v sieti DeepLabv2 rieši problém výskytu objektov v obraze v rôznych veľkostiach. Namiesto toho, aby bol na vstupe použitý jeden obraz v niekoľkých veľkostiach, využíva sa iba pôvodná verzia obrazu. Na koniec siete je pridaných niekoľko paralelne uložených vrstiev atrous konvolúcie, každá s inou veľkosťou konvolučného filtra, z ktorých výstupné mapy príznakov sa ďalej spracúvajú v osobitných vetvách, tie sa na konci zlúčia a vznikne finálny výstup.

Hlboká konvolučná sieť s väčším množstvom konvolučných a pooling vrstiev generuje presné výsledky z pohľadu klasifikácie, no hrany segmentovaných objektov sú veľmi vyhladené. Pre spresnenie hrán využíva DeepLabv2 postprocessing metódu CRF - conditional random field. Za konvolučnou časťou siete je integrovaná úplne prepojená CRF z [42], ktorá v spojení s informáciami získanými z výstupu konvolučnej časti dokáže lokalizovať hrany objektov čím efektívne zvýši segmentačnú presnosť celého modelu. Fungovanie modelu DeepLabv2 je zobrazené na Obr.3.5.



Obr. 3.5: Proces segmentácie statického obrázku v modeli DeepLabv2. Zdroj:[27]

DeepLabv2, ktorý je súčasťou MaskTrack architektúry, vykonáva sémantickú segmentáciu obrazu, t.j. segmentuje celý obraz na všetky klasifikované objekty a pozadie. Hlavným cieľom v úlohe segmentácie videa je ale oddelenie jedného alebo niekoľkých špecifikovaných objektov od pozadia. Na prispôsobenie sa tejto úlohe využíva MaskTrack dve oddelené fázy trénovania.

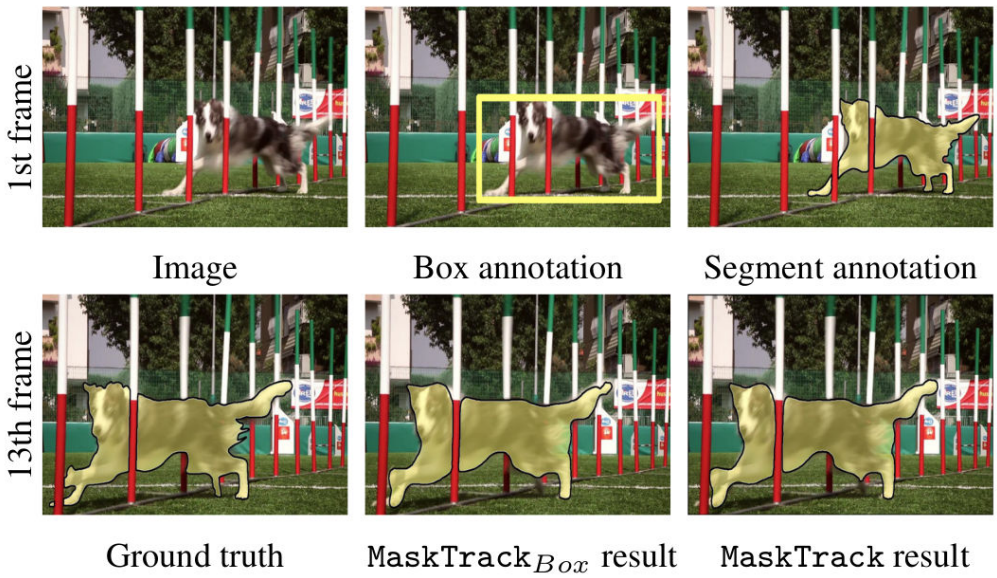


Obr. 3.6: Model je natrénovaný tak, aby dokázal z kombinácie snímky v čase t a segmentačnej masky z predchádzajúcej snímky v čase $t-1$ vygenerovať segmentačnú masku pre danú snímku v čase t. Zdroj:[29]

3.2.2 Trénovanie

Prvá fáza, offline trénovanie, zabezpečuje navedenie siete na segmentáciu špecifickej inštancie objektu použitím predchádzajúcej segmentačnej masky. Konvolučná sieť, ktorá je základom MaskTrack-u, je rozšírená z troch na štyri kanály: RGB + kanál pre masku. Kanály RGB predstavujú snímku, ktorá bude segmentovaná a štvrtý kanál predstavuje hrubý náčrt sledovaného objektu. V praxi sa cez tento kanál šíri maska z predchádzajúcej snímky, dostatočne dobré výsledky je však možné dosiahnuť aj použitím jednoduchých ohraničujúcich rámikov, ktoré iba naznačujú polohu objektu v obraze. Dáta prechádzajúce štvrtým kanálom tak nehovoria veľa o vzhľade objektu ale iba o jeho umiestnení v obraze. Základný princíp fungovania architektúry MaskTrack je zobrazený na Obr.3.6

Okrem offline trénovania má svoje miesto v tejto architektúre aj online trénovanie, ktoré prebieha v testovacej fáze modelu. Pôvodne využívané najmä v úlohách lokalizácie objektov v obraze, online trénovanie nachádza využitie aj v úlohe segmentácie obrazu, čo je možné vidieť aj v iných prístupoch (napr. OSVOS). MaskTrack však volí rozdielnu stratégiu ako OSVOS - na prvú anotovanú snímku a jej masku aplikuje afínne transformácie a operácie ako rotácia a priblíženie obrazu, čím sa vygeneruje $\sim 10^3$ dvojíc snímka + maska. Vygenerovaná množina je následne použitá na online dotrénovanie modelu, ktorý sa tak naučí presnejšiu reprezentáciu sledovaného objektu.



Obr. 3.7: Výsledky segmentácie pri použití segmentačnej masky z predchádzajúcej snímky (v strede) a pri použití ohraničujúceho rámiku (vpravo). Zdroj:[29]

3.2.3 Použitie a vlastnosti

Ako už bolo spomenuté, MaskTrack očakáva v štvrtom kanáli vstupných dát segmentačnú masku predchádzajúcej snímky, no za cenu miernej degradácie výsledkov sa uspokojí aj s ohraničujúcim rámikom, ktorý určuje polohu sledovaného objektu v obraze. Model je tak využiteľný aj v situáciách, kedy nie je presne anotovaná ani prvá snímka. Porovnanie výsledkov segmentovania pri použití segmentačnej masky a ohraničujúceho rámiku v štvrtom kanáli je zobrazené na obr. 3.7

Ďalšou výhodou je možnosť využiť v testovacej fáze viac ako jednu anotovanú snímku. S rastúcim množstvom anotovaných snímok sa zvyšuje výsledná presnosť modelu, čo môže byť prospešné napríklad v úlohách video-editácie.

Model postavený na architektúre MaskTrack dosahuje veľmi dobré výsledky segmentácie pre väčšinu videí, ktoré sa vyznačujú problémami ako rýchle pohyby sledovaného objektu, prekážky pred objektom, dočasné zmiznutie objektu z obrazu atď. Výnimkou sú sekvencie s výraznými pravidelnými pohybmi obrazu spôsobenými roztrasenou kamerou, v ktorých model podáva podpriemerné výsledky.

Poslednou dôležitou charakteristikou je možnosť trénovať celú sieť na statických anotovaných obrázkoch, napríklad na množinách ako PASCAL-S, MSRA10K,

COCO, atď. Týmto odpadá nutnosť trénovať model výhradne na anotovaných videách a rozširujú sa možnosti využitia modelu aj v doménach, v ktorých existujú anotované iba statické obrázky.

3.2.4 Príklady implementácie

MaskTrack má mierne vyššie zastúpenie v modeloch určených na segmentáciu videa a tieto modely dosahujú vo väčšine prípadov mierne lepšie výsledky ako modely postavené na architektúre OSVOS.

Ukážkovým príkladom implementácie architektúry MaskTrack je práca [19], s ktorou jej tvorcovia zvíťazili v súťaži zameranej na segmentáciu videí v roku 2017 - DAVIS challenge[2]. Ako väčšina iných úspešných riešení, aj tu autori využili techniku rozširovania dátovej množiny pomocou procesu Lucid data dreaming. Z funkcionálneho hľadiska je implementácia modelu rozdelená na dva moduly:

1. **Modul šírenia segmentačnej masky.** Tento modul je vytvorený ako adaptácia architektúry MaskTrack, narozdiel od pôvodnej architektúry je však zložený z dvoch vetiev. Prvou vetvou sa šíri iba maska z predchádzajúcej snímky. Z rozdielov medzi aktuálnou a predchádzajúcou snímkou je modelom FlowNet2.0 [28] extrahovaná informácia o optickom toku, ktorá putuje do druhej vetvy. Výsledný návrh na segmentačnú masku je vygenerovaný kombináciou aktuálnej snímky, masky z predchádzajúcej snímky a optického toku medzi aktuálnou a predošlou snímkou.

2. **Re-identifikačný modul**, ktorý adresuje problém prekážok pred cieľovým objektom a náhlych zmien vzhľadu objektu. Vo chvíli kedy model prestane segmentovať jednu inštanciu objektu vo videu, re-identifikačný modul navrhne nových kandidátov na túto inštanciu a týchto kandidátov porovná s informáciami z predchádzajúcich segmentačných masiek. Ak sa mu podarí re-identifikovať potrebnú inštanciu, odošle informáciu o nej obojsmerne do najbližších snímok, čím zabezpečí plynulé pokračovanie v segmentácii.

Z ďalších implementácií architektúry MaskTrack môžeme spomenúť ešte riešenie [18], ktoré takisto využíva variáciu re-identifikačného modulu alebo riešenie [16], ktoré prinieslo techniku Lucid data dreaming, popísanú v kapitole 4.2.1.

[2] `https://davischallenge.org/`

4 Návrh a implementácia riešenia

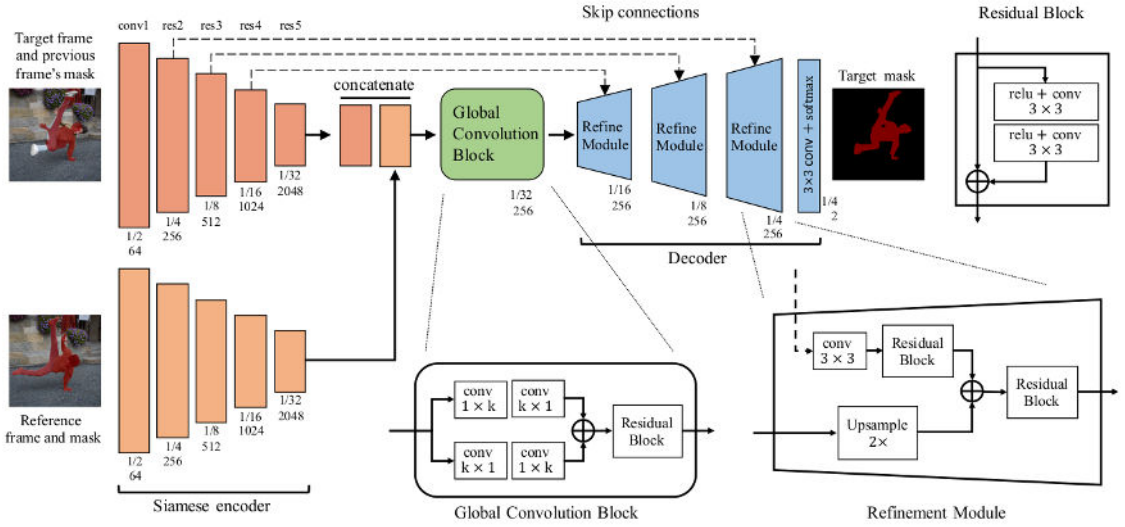
Hlavným cieľom tejto práce z pohľadu softvéru je implementovať existujúce riešenie určené na segmentáciu videí a toto riešenie následne doplniť implementáciou metód strojového učenia pre dosiahnutie lepších výsledkov segmentácie. V tejto kapitole bude detailne popísané pôvodné riešenie, z ktorého sme vychádzali a ktoré sme modifikovali ako aj samotné modifikácie, ktoré sme implementovali.

4.1 Existujúce riešenie

RGMP - Reference-Guided Mask Propagation - je architektúra hlbokého učenia, ktorá bola vytvorená špecificky pre riešenie úlohy segmentácie objektov vo videách. Model implementovaný na základe tejto architektúry sa vyznačuje presnosťou blížiacou sa aktuálnym SOTA riešeniam a zároveň niekoľkonásobne vyššou rýchlosťou spracovania testovacích dát. Rovnako ako Mask-Track, aj táto architektúra využíva proces šírenia segmentačnej masky z predchádzajúcej snímky, no výrazne sa odlišuje v svojej celkovej stavbe, ktorej kostru tvorí štruktúra enkóder-dekóder.

4.1.1 Architektúra

Stavba architektúry, zobrazenej na Obr.4.1, sa začína enkóderom, ktorý je v tomto prípade rozdvojený na dve zhodné časti s navzájom zdieľanými váhami. Vstupom do prvej časti je cieľový tok dát obsahujúci aktuálne segmentovanú snímku vrátane masky z predchádzajúcej snímky. Vstupom do druhej časti je referenčný tok dát, ktorý obsahuje prvú snímku videa v RGB a k nej manuálne anotovanú masku. Obe časti enkódera tak pracujú so 4 obrazovými kanálmi a sú implementované tak, aby dokázali spracovať videá v ľubovoľnom rozlíšení. Z pohľadu štruktúry neurónových vrstiev je enkóder postavený na sieti ResNet50, ktorej váhy sú inicializované z modelu predtrénovaného na dátovej množine ImageNet.



Obr. 4.1: Architektúra RGMP s detailne popísanými modulmi. Zdroj:[10]

Pokračovaním za enkóderom je globálny konvolučný blok, ktorého úlohou je lokalizácia sledovaného objektu v cieľovom obraze, a to tak, že hľadá spoločné znaky medzi referenčným a cieľovým obrazom. Prax ukazuje, že konvolučné masky s vyšším rozlíšením umožňujú konvolučnej sieti presnejšie lokalizovať konkrétne prvky v obraze avšak na úkor výrazného navýšenia výpočtovej náročnosti. Preto je v tejto architektúre implementovaná globálna konvolúcia [21], ktorá využíva masky s rozmermi $1 \times k + k \times 1$ a paralelne $k \times 1 + 1 \times k$. Takto rozložené masky dosahujú podobné výsledky ako maska s rozmermi $k \times k$ no s výrazne nižšou výpočtovou náročnosťou. Výstup z globálnej konvolúcie je ešte spracovaný reziduálnym blokom, ktorý je zložený z dvoch konvolučných vrstiev.

Do poslednej časti RGMP architektúry - dekódera - vstupujú dva osobitné dátové toky. Prvým je výstup z globálneho konvolučného bloku a druhým sú informácie z enkódera, konkrétne z časti s cieľovým tokom dát, a to prostredníctvom skip prepojení medzi enkóderom a dekóderom. Dekóder je zložený z niekoľkých blokov, tzv. refinement modulov, ktorých úlohou je postupne navyšovať rozlíšenie dát z globálneho konvolučného bloku a to za pomoci dát z jednotlivých blokov enkóderu. Refinement modul je inšpirovaný prácou [33], no pôvodne použité konvolučné vrstvy sú nahradené reziduálnymi. Za refinement modulmi je umiestnená posledná konvolučná vrstva a model je zakončený vrstvou aktivačnej funkcie softmax, ktorej výstupom je konečná segmentačná maska objektu. Táto maska má

Obr. 4.2: Umelé generovanie trénovacích vzoriek zo statických obrázkov použitím dvoch rôznych postupov. Zdroj:[10]

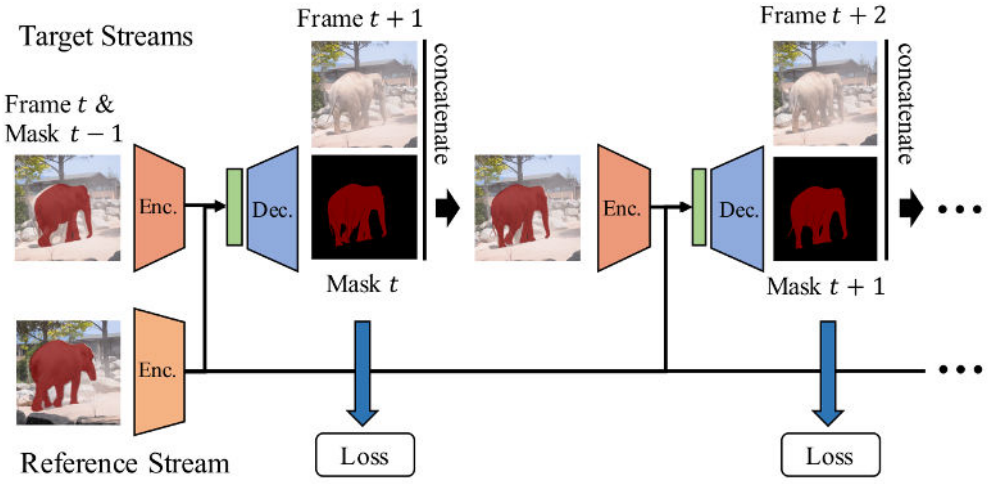
štvrtinové rozlíšenie vstupného obrazu.

4.1.2 Trénovanie

Aj napriek použitiu predtrénovaných váh v enkóderi je potrebné zvyšok siete natrénovať špecificky pre úlohu segmentácie. Toto trénovanie si vyžaduje veľké množstvo dát v podobe dvojíc referenčná snímka + cieľová snímka. V roku 2018, v čase vzniku tejto architektúry, bola najlepšia dostupná dátová množina DAVIS 2017, ktorá však sama o sebe nestačila na dostatočné natrénovanie daného modelu. To bolo dôvodom implementácie dvojfázového procesu trénovania:

1. V prvej fáze prebieha predtrénovanie celého modelu na dátovej množine vytvorenej transformáciou množín statických segmentovaných obrázkov, ako napr. VOC alebo MSRA10K. Generovanie tejto množiny prebiehalo aplikáciou náhodných transformácií obrazu (rotácia, priblíženie, rôzne afínne transformácie) na statické obrázky (Obr.4.2).

2. Druhá fáza je zameraná na dotrénovanie modelu na dátach z anotovaných videí, napríklad na množine DAVIS 4.4.1. Dvojice referenčnej a cieľovej snímky vrátane ich masiek sú generované náhodným výberom snímok z videa. Takto



Obr. 4.3: Rekurentné trénovanie modelu. Trénovacia chyba sa počíta v každom časovom kroku a model sa aktualizuje spätnou propagáciou v čase. Zdroj:[10]

generované trénovacie dáta ale nereflektujú realitu, pretože v nich nie je zachytená kumulácia chyby, ktorá sa prejavuje v testovacej fáze. Tento problém je vyriešený tak, že namiesto exaktnej masky k predchádzajúcej snímke sa využíva modelom predikovaná maska. V praxi je model napojený rekurentne sám na seba a do cieľového toku dát posiela masku, ktorú vygeneroval pre predchádzajúcu snímku. Proces rekurentného trénovania je zobrazený na Obr.4.3.

Týmto spôsobom natrénovali výskumníci pôvodný model (*RGMPorigin*), ktorého výkonnosť takisto zahrnieme vo vyhodnotení. Váhy modelu *RGMPorigin* sú verejne dostupné na stiahnutie, no sú uložené v podobe, ktorá nedovoľuje tento model dodatočne trénovať a využiť ho tak pri našich experimentoch. Preto sme v našej práci natrénovali model úplne nanovo (*RGMPnew*) a to použitím kombinácie dátových množín YouTube-VOS4.4.2 a DAVIS4.4.1. Práve na model *RGMPnew* sme aplikovali naše modifikácie a využili sme ho vo všetkých našich experimentoch.

4.1.3 Testovanie

Testovacia fáza prebieha podobne ako pri architektúrach MaskTrack alebo OSVOS. Prvá anotovaná snímka je vybraná ako referenčná, ostatné snímky sú sekvenčne anotované modelom. Model v testovacej fáze nevyužíva online trénovanie na naučenie sa vzhľadu sledovaného objektu, čo skracuje proces testovania a je jednou z

výhod RGMP architektúry.

Referenčná snímka sa počas testovacej fázy nemení, preto je výstup druhej časti enkódera s referenčnou snímkou vypočítaný iba raz počas celého procesu testovania. Tento postup poskytuje ďalšie výrazné zrýchlenie testovania modelu. Pre lepšie zachytenie objektov rôznej veľkosti sú vstupné dáta spracované v troch rôznych mierkach - 0.5, 0.75 a 1.0 a výstupy sú na konci testovania spriemerované.

Model postavený na RGMP dokáže segmentovať videá s jedným alebo s viacerými sledovanými objektmi. Narozdiel od architektúr OSVOS, MaskTrack a väčšiny ďalších, ktoré segmentujú objekty vždy len po jednom, RGMP implementuje novú techniku, tzv. softmax agregáciu. Táto technika priradzuje pravdepodobnosti jednotlivým inštanciám, pričom tieto pravdepodobnosti musia byť kladné čísla a ich suma musí byť rovná 1. Tento algoritmus funguje na základe rovnice:

$$p_{i,m} = \sigma(logit(\hat{p}_{i,m})) = \frac{\hat{p}_{i,m}/(1-\hat{p}_{i,m})}{\sum_{j=0}^{M} \hat{p}_{i,j}/(1-\hat{p}_{i,j})} \tag{4.1}$$

kde:

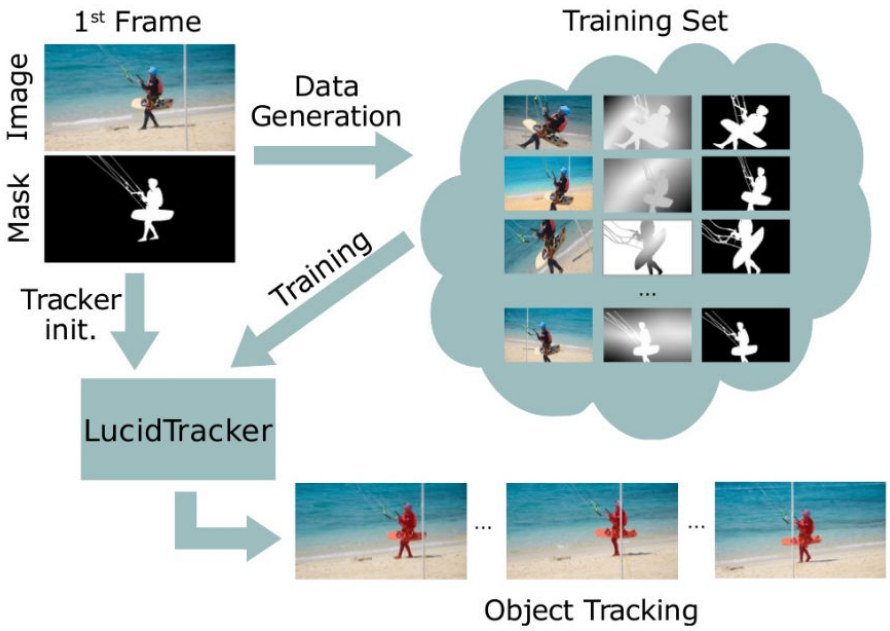
- σ - softmax funkcia,
- $logit$ - logit funkcia,
- $\hat{p}_{i,m}$ - modelom vygenerovaná pravdepodobnosť, že pixel na pozícii i reprezentuje objekt m,
- M - počet inštancií (objektov) vo videu.

Model aplikuje tento postup na každú snímku, pričom vypočítané pravdepodobnosti $p_{i,m}$ pre jednu snímku sú referenčným vstupom pre segmentovanie ďalšej snímky.

4.2 Návrh modifikácii modelu

4.2.1 Lucid data dreaming

Experimenty ukazujú, že model určený na segmentáciu videa je schopný dosiahnuť výrazne vyššiu presnosť pri segmentovaní videa, ak je dodatočne natrénovaný



Obr. 4.4: Generovanie dát na základe prvej anotovanej snímky. Zdroj:[16]

na dátach podobných danému testovanému videu. Inými slovami, čím viac snímok testovaného videa dokážeme manuálne anotovať a použiť na trénovanie modelu, tým lepšie výsledky dosiahne segmentáciou zvyšku daného videa. Manuálne anotovať snímky je však časovo náročný proces, preto za účelom generovania trénovacích dát z prvej anotovanej snímky vznikla technika Lucid data dreaming [16], ktorej výstupom môže byť až niekoľko tisíc navzájom rôznych párov snímka + maska. Cieľom je vygenerovať dáta, ktoré budú predstavovať simuláciu videa a to len použitím prvej anotovanej snímky. Dodatočné trénovanie modelu na týchto dátach sa nazýva aj online trénovanie.

Tradičný prístup generovania snímok, v ktorom sa vykonávajú iba jednoduché transformácie obrazu, nie je schopný dostatočne pokryť očakávané vizuálne zmeny v priebehu videa - zmeny osvetlenia alebo uhlu pohľadu, prekážky pred sledovaným objektom, zmena farieb objektu, atď. Lucid data dreaming však využíva niekoľko pokročilých metód, čím adresuje väčšinu zo spomenutých problémov. Postup algoritmu, zobrazený aj na obr. 4.4 je nasledovný:

1. Zmeny svetelnosti. Obrázok je upravený zmenou hodnôt saturácie - zložka S - a zložky V vo farebnom priestore HSV použitím náhodných hodnôt z vymedzeného rozsahu.

2. Vyrezanie objektu. Cieľový objekt je podľa segmentačnej masky vyrezaný zo snímky a prázdne miesto v snímke sa vyplní textúrou vygenerovanou pomocou algoritmu PatchMatch, ktorý dokáže hodnoverne zakryť chýbajúce miesta v obrázkoch podľa ich obsahu [45]. Vzniknú tak dva obrázky - pozadie a cieľový objekt.

3. Pohyb objektu. Aplikáciou afínnych transformácií na obrázok s cieľovým objektom sa dosiahne simulácia pohybu a zmeny tvaru tohto objektu. Na prvej snímke z dvojice je objekt vložený na pozadie na náhodné miesto a na druhej snímke s odstupom maximálne 10% veľkosti daného objektu. V oboch snímkach je aplikovaná náhodná rotácia o $\pm 30°$, zmena mierky o $\pm 15\%$ a deformácia roviny v rozsahu $\pm 10\%$ veľkosti objektu.

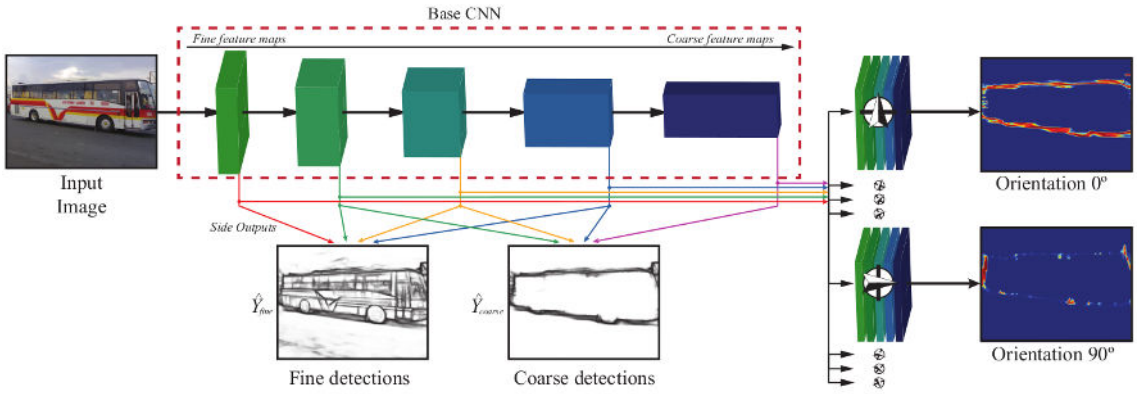
4. Pohyb kamery. Na pozadie sa aplikujú rovnaké operácie ako na obrázok cieľového objektu v predchádzajúcom bode, čím sa dosiahne simulácia zmeny pohľadu kamery.

5. Spojenie pozadia a cieľového objektu. Obrázky pozadia a cieľového objektu sa spoja použitím techniky Poissonovo zmatňovanie [48]. Vďaka tomu, že všetky transformácie vykonané na snímkach sú známe, k daným snímkam tento algoritmus vygeneruje aj ich segmentačné masky (M_{t-1}, M_t).

Lucid data dreaming sme do modelu implementovali tak, aby bolo možné zvoliť si konkrétne transformácie, ktoré sa s obrázkami vykonajú. Postupnou zmenou týchto transformácií a ich parametrov ich budeme vedieť navzájom porovnať a zistiť tak, aké trénovacie dáta je najlepšie generovať pre dosiahnutie čo najvyššej presnosti.

4.2.2 Konvolučné orientované kontúry - COB

COB je architektúra založená na CNN, ktorá generuje viacúrovňové informácie o orientovaných kontúrach objektov v obraze, pričom vychádza z bežnej klasifikácie pomocou CNN. Model postavený na tejto architektúre generuje informácie nielen o orientácii kontúr ale aj o ich „sile“ na jednotlivých úrovniach v obraze. COB je využiteľná pri implementácii samostatného modelu na ohraničovanie objektov v obraze ale aj ako modulárna súčasť zložitejších modelov určených na riešenie úloh ako je návrh objektov v obraze, sémantická segmentácia, detekcia objektov a ďalšie.



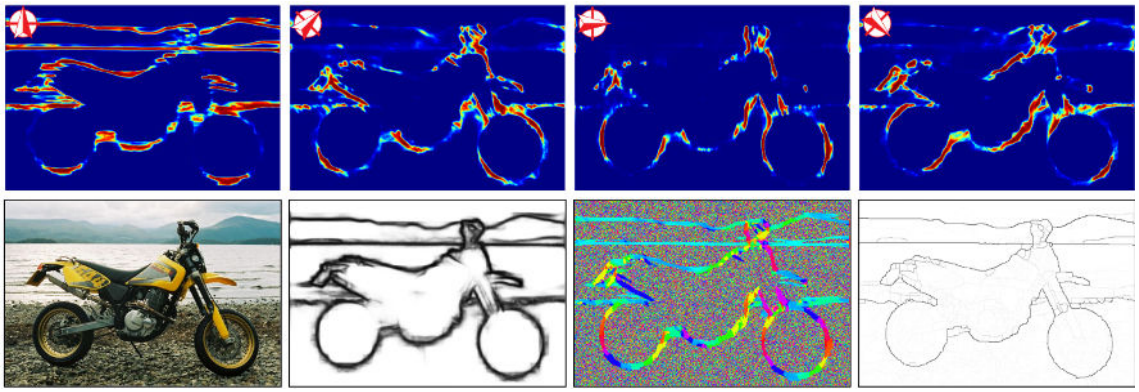
Obr. 4.5: Proces spracovania obrazu v architektúre COB. Zdroj:[20]

COB dosahuje v čase písania tejto práce SOTA výsledky v úlohe detekcie kontúr.

Jednou zo základných vlastností konvolučných sietí je to, že extrahujú príznaky na niekoľkých úrovniach v obraze - od prvej vrstvy, ktorá rozlišuje hrany a základné tvary po poslednú vrstvu, ktorá dokáže rozpoznať konkrétne objekty. Tento princíp je využitý aj v modeli pre detekciu kontúr, ktorý v prvých vrstvách vníma aj veľmi jemné kontúry na lokálnej úrovni no s posunom k hlbším vrstvám sa model začína viac sústrediť na hrubšie kontúry konkrétnych objektov. CNN je preto ideálnym základným modelom pre vytvorenie architektúry na rozpoznávania kontúr.

Základom COB je CNN model postavený na architektúre VGG, ResNet alebo im podobnej, ktorý je natrénovaný na dátovej množine určenej na klasifikáciu obrazu, ako napríklad ImageNet. Z tohto modelu sú odstránené posledné úplne prepojené vrstvy, ktoré slúžia na klasifikáciu objektov, rovnako ako aj normalizačné vrstvy, keďže konečný model spracúva v jednej iterácii iba jeden obrázok. Zvyšok siete je rozdelený do M blokov, kde každý blok deteguje kontúry na inej úrovni v obraze. Na Obr.4.5 je zobrazený následný proces - informácie z prvých štyroch úrovní sa trénovateľnou lineárnou kombináciou zlúčia do tzv. jemného (fine) výstupu, ktorý reprezentuje jemnejšie kontúry a informácie z posledných štyroch úrovní sa rovnakou operáciou zlúčia do tzv. hrubého (coarse) výstupu, ktorý reprezentuje hrubšie kontúry.

Základný konvolučný model v COB je rozšírený o modul, ktorého úlohou je identifikácia orientácie detegovaných kontúr. Modul je rozdelený na K častí, z ktorých každá je tvorená M konvolučnými vrstvami. K reprezentuje zvolený počet identi-



Obr. 4.6: Prvý rad ilustruje výstupy skupín konvolučných vrstiev pre štyri rôzne orientácie. V druhom rade sú postupne zobrazené: 1 - pôvodný obrázok, 2 - sila detegovaných kontúr, 3 - rozpoznaná orientácia kontúr a 4 - detegované kontúry na všetkých úrovniach obrazu. Zdroj:[20]

fikovateľných orientácií a M reprezentuje počet prehľadávaných úrovní v obraze - rovnaké M ako v základnej CNN. Každá konvolučná vrstva tak spracúva kontúry z špecifickej úrovne obrazu a snaží sa identifikovať orientáciu týchto kontúr. Výstupy všetkých vrstiev sa spoja do jedného výsledného obrazu, ktorý obsahuje informácie o orientácií kontúry v každom bode obrazu. Výsledok celého procesu je zobrazený na Obr.4.6. Informácie o orientácii kontúr nemajú pre úlohu segmentácie obrazu žiaden prínos, preto tento modul v našej práci neimplementujeme.[20]

4.3 Metodológia testovania

Ohodnotenie modelu, ktorý je výstupom tejto práce, je rozdelené na dve časti - kvantitatívne testovanie a kvalitatívne testovanie. Kvalitatívne testovanie predstavuje subjektívny popis výsledkov segmentácie, v ktorom sa zameriavame na schopnosť modelu spracovať rôzne dátové množiny z ľubovoľnej domény a takisto správanie sa modelu v prípadoch, kedy sa vo vstupných dátach vyskytnú niektoré z týchto špeciálnych prípadov: deformácia sledovaného objektu, prítomnosť prekážky pred sledovaného objektom, nízke rozlíšenie obrazu, roztrasený obraz a niekoľko ďalších, ktoré sú spomenuté v kapitole 4.4.1.

Kvalitatívne ohodnotenie je spracované formou krátkeho textu, v ktorom sú zdôraznené prednosti modelu ale takisto aj jeho slabé stránky a situácie, v ktorých

model nedosahuje dostatočne vysokú presnosť.

Kvantitatívne testovanie je zamerané na ohodnotenie modelu z pohľadu numerickej presnosti segmentácie a jeho primárnym účelom je možnosť vzájomného porovnania sa s inými modelmi. V tejto práci využívame dve metriky - *Jaccardov index* a *F miera ohraničenia*. Obe metriky boli použité na ohodnotenie modelov vo všetkých troch ročníkoch súťaže *DAVIS challenge*[1] zameranej na úlohu segmentácie videí, ktorej sa doteraz zúčastnilo viac ako 30 tímov s ich riešeniami. Vďaka tomu vieme efektívne porovnať kvalitu segmentácie nášho modelu s inými riešeniami.

Pri tých modeloch, v ktorých popise ich tvorcovia špecifikovali aj čas spracovania testovacích snímok, bude táto informácia uvedená. Aj napriek rôznorodosti použitých hardvérových prostriedkov si tak čitateľ urobí približný prehľad o tom, ktoré modely sú schopné spracovať testovacie dáta v reálnom čase a ktoré nie.

4.3.1 Jaccardov index

Jaccardov index J, niekedy nazývaný aj *Koeficient podobnosti oblastí*, meria podobnosť medzi ground truth segmentáciou a oblasťou, ktorá je výstupom použitého modelu. Hodnota predstavuje percentuálny podiel správne klasifikovaných vzoriek k všetkým vzorkám z dátovej množiny. V úlohe segmentácie obrazu poskytuje vhodnú metriku na intuitívne meranie podielu správne klasifikovaných pixelov ku všetkým pixelom. Výpočet J podľa vzorca [32]:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} \tag{4.2}$$

kde v kontexte obrazových dát:

- A - ground truth segmentačná maska,
- B - segmentačná maska vygenerovaná modelom.

Pre účely vyhodnotenia vypočítame priemernú hodnotu pre všetky snímky $JMean$ hodnotu návratnosti $JRecall$, ktorá predstavuje pomer medzi počtom snímok s $J > 0.5$ k počtu všetkých snímok.

[1] `https://davischallenge.org/index.html`

4.3.2 F miera ohraničenia

Bežne označovaná ako *F1* alebo *F miera*, táto metrika slúži na meranie výkonnosti klasifikačného modelu, pričom vo výpočte berie do úvahy dve metriky - presnosť p a návratnosť r. Pre úlohu segmentácie obrazu vznikla alternatíva s názvom *F miera ohraničenia*, ktorá je zameraná na ohodnotenie segmentácie z pohľadu ohraničenia (kontúr) segmentovaných objektov. Medzi okrajovými bodmi ground truth segmentačnej masky a masky vygenerovanej modelom sa skonštruuje párny (bipartitný) graf, ktorého dĺžka hrán sa následne zapíše a porovná s definovanou hraničnou hodnotou. Ak je vzdialenosť medzi dvojicou okrajových bodov vyššia ako hraničná hodnota, daný bod sa označí ako príliš vzdialený od ground truth masky - predikovaná kontúra je príliš vzdialená od kontúry ground truth masky. [47] Z vyhodnotenia klasifikácie jednotlivých bodov sa vypočíta presnosť ohraničenia p_c a návratnosť ohraničenia r_c, ktoré sa nakoniec dosadia do vzorca [32] pre F mieru ohraničenia:

$$F(A, B) = \frac{2p_c r_c}{p_c + r_c} \tag{4.3}$$

Tak ako pri *Jaccardovom indexe*, aj tu použijeme vo vyhodnotení dve vypočítané hodnoty - priemer $FMean$ a hodnotu návratnosti $FRecall$, ktorá je vypočítaná rovnako ako $JRecall$.

4.4 Použité dáta

V tejto práci využívame dve dátové množiny - DAVIS a Youtube-VOS. Obe množiny boli vytvorené a spracované špecificky pre úlohu segmentácie objektov vo videách a obe predstavujú veľmi kvalitný základ pre trénovanie a testovanie modelov. Pre vyhodnotenie nášho modelu uprednostníme množinu DAVIS, na ktorej bolo doteraz natrénovaných a otestovaných niekoľko desiatok riešení, čo nám umožní kvantitatívne porovnať náš model so staršími ako aj s aktuálnymi SOTA riešeniami.

4.4.1 DAVIS

Dátová množina DAVIS [32][22] - Densely Annotated Video Segmentation - vznikla pred vyše troma rokmi ako referenčná množina pre vtedy začínajúcu rovnomennú sútaž DAVIS challenge, ktorá je zameraná výhradne na problém segmentácie objektov vo videách. V tejto práci budeme používať najnovšiu verziu tejto množiny z

Obr. 4.7: Ukážka anotovaných snímok z dátovej množiny DAVIS. Zdroj:[22]

roku 2017.

Množina sa skladá z 90 anotovaných videí s rôznorodým obsahom, ktoré sú zložené z celkovo 6208 snímok v rozlíšení minimálne 480p. Väčšina videí je k dispozícii aj vo vyšších rozlíšeniach ako 1080p alebo 4k, no v súťaži DAVIS sú v testovacej fáze využívané videá v jednotnom rozlíšení 480p. Videá boli nasnímané vo frekvencii 24 snímok za sekundu, pričom trvanie jedného videa sa pohybuje v rozmedzí dvoch až štyroch sekúnd. Niekoľko ukážok z množiny je zobrazených na Obr.4.7.

Každé video obsahuje jeden alebo niekoľko sledovaných objektov, pričom ku každému objektu je priradená osobitná anotácia v podobe segmentačnej masky. Pre dosiahnutie diverzity videí a pre obmedzenie preučenia modelov na týchto dátach boli do tejto množiny vybrané také videá, ktoré obsahujú jeden alebo kombináciu niekoľkých atribútov sťažujúcich úlohu segmentácie:

- podobnosť sledovaného objektu s pozadím alebo s inými objektmi,
- výrazná deformácia sledovaného objektu v priebehu videa,
- rozmazaný obraz spôsobený rýchlym pohybom kamery alebo sledovaného objektu,
- nízke rozlíšenie videa,
- prítomnosť prekážky pred sledovaným objektom,
- dočasné zmiznutie sledovaného objektu mimo záber kamery,

- zmena vzhľadu sledovaného objektu z dôvodu zmeny uhla pozorovania alebo osvetlenia objektu,
- roztrasený obraz,
- interakcia objektov s pozadím alebo medzi sebou navzájom,
- vysoká zložitosť hrán objektu, napríklad príliš úzke objekty alebo diery v objektoch,
- dynamické pozadie, ktoré sa výrazne mení v priebehu videa.

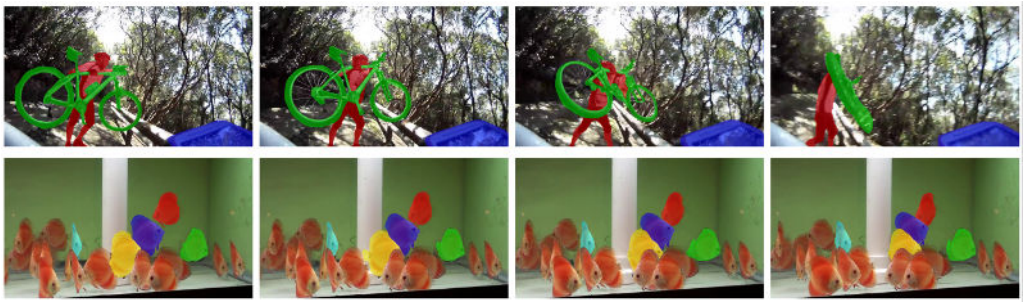
Súčasťou každého videa je súbor anotácií v podobe segmentačných masiek, ktoré jednoznačne identifikujú všetky pixely reprezentujúce sledovaný objekt. Ak sa vo videu vyskytuje iba jeden sledovaný objekt, priložené segmentačné masky sú v binárnom formáte. V prípade niekoľkých objektov sú tieto masky viacfarebné, pričom jedna farba zodpovedá jednému osobitnému objektu. Pre čo najväčšiu univerzálnosť dát boli zvolené sledované objekty z najrôznejších domén - ľudia, iné živočíchy, dopravné prostriedky, objekty každodennej potreby, atď.

Množina je rozdelená na dve podmnožiny - trénovacia a validačná. Testovacia časť, ktorá je takisto k dispozícii, obsahuje iba prvú anotovanú snímku a teda na nej nie je možné otestovať kvantitatívnu presnosť modelu. Vo validačnej fáze sme tak použili videá z nasledujúcej množiny - YouTube-VOS - a validačnú časť množiny DAVIS sme použili pre otestovanie nášho modelu. Podrobnejšie informácie o použití dát budú popísané v kapitole Dosiahnuté výsledky5.

4.4.2 YouTube - VOS

Dátová množina YouTube-VOS [12] vznikla koncom roka 2018 ako odpoveď na zvyšujúci sa záujem o riešenie úloh segmentácie videí a zároveň nedostatočné množstvo dát pre daný problém. Ako už názov napovedá, množina je tvorená selekciou súborov zo známej webovej služby YouTube.

Obsahom tejto množiny je 4453 videí s dĺžkou jednej sekvencie 3-6 sekúnd a s celkovou dĺžkou 334 minút. 7822 rôznych objektov je rozdelených do 94 kategórií v podobe zvierat, dopravných prostriedkov, ľudí vykonávajúcich aktivity, objektov



Obr. 4.8: Ukážka anotovaných snímok z dvoch sekvencií z dátovej množiny YouTube-VOS. Zdroj:[12]

každodennej potreby atď. V celej množine je manuálne anotovaných 191378 snímok. Anotovaná je len každá piata snímka v celom videu, čo je zdôvodnené tým, že korelácia medzi piatimi po sebe idúcimi snímkami je dostatočne silná na to, aby bolo možné kvalitne natrénovať model a zároveň znížiť nároky na manuálne anotovanie. Výsledkom je tak video s frekvenciou 30 snímok za sekundu a anotácie v podobe segmentačných masiek s frekvenciou 6 snímok za sekundu. Ukážka anotácií je zobrazená na Obr.4.8.

Medzi hlavné výhody použitia YouTube videí patrí rôznorodosť objektov a ich pohybov, ktoré sú obsahom sekvencií. V náhodných videách natočených v každodenných situáciách sa prirodzene vyskytuje väčšina prípadov, ktoré zvyšujú náročnosť segmentácie. Videá uložené na YouTube sú vytvorené ako amatérmi tak aj profesionálmi, čo pridáva na diverzite ich kvality. Roztrasený alebo neostrý obraz, náhle a nečakané pohyby sledovaného objektu, prekážky pred sledovaným objektom ale aj ďalšie situácie sú bežnou súčasťou väčšiny videí na YouTube. Práve tieto vlastnosti zabezpečia, že model natrénovaný na týchto dátach bude schopný poskytnúť dostatočne kvalitné výsledky aj pri nasadení do každodennej prevádzky.

Vďaka YouTube tak máme k dispozícii novú dátovú množinu určenú špecificky na úlohu segmentácie objektov vo videách, ktorá obsahuje 15-krát viac anotovaných snímok a celkovo 50x viac sekvencií ako množina DAVIS. YouTube-VOS množina však v čase písania práce existuje ešte iba niekoľko mesiacov a zatiaľ nie je k dispozícii veľa publikácií, v ktorých by výskumníci trénovali a testovali model na tejto množine. Z toho dôvodu použijeme túto množinu primárne vo fáze trénovania modelu pre zvýšenie schopnosti generalizácie a celkové zlepšenie výsledkov segmentácie.

5 Dosiahnuté výsledky

Obsahom tejto kapitoly je popis a vyhodnotenie experimentov, ktoré sme vykonali s tromi rôznymi variantmi nášho modelu - RGMPorigin, RGMPnew a RGMPnew+Lucid. Súčasťou kapitoly budú ukážky výsledkov segmentácie, kvalitatívne zhodnotenie a identifikácia silných a slabých stránok daných modelov. V nasledujúcom texte tak popíšeme výsledky dosiahnuté experimentmi, zhodnotíme vplyv pridaných trénovacích dát a naše riešenia porovnáme s inými prácami.

Všetky tri verzie modelov sme otestovali na validačnej časti dátovej množiny DAVIS4.4.1 zloženej z 30 videí, keďže na týchto sekvenciách boli otestované a vyhodnotené takmer všetky modely vytvorené za posledný rok. Aj napriek označeniu *validačná množina* sme namiesto týchto videí použili vo fáze validácie 100 videí z množiny YouTube-VOS4.4.2. Ukážky segmentácie zobrazíme v každej podkapitole na snímkach z niekoľkých videí, ktoré najlepšie reprezentujú rôznorodosť množiny a špecifické hraničné prípady, ktoré model zvládol alebo nezvládol podľa očakávania.

Všetky výsledky segmentovania modelov RGMPnew a RGMPnew+Lucid z testovacej fázy je možné prezrieť si v zložke **rgmp/results** v priloženom .zip súbore.

5.1 Pôvodný model RGMPorigin

5.1.1 Popis

Tento model bol jeho tvorcami natrénovaný najprv na umelo vygenerovaných dátach zo statických obrázkov (hlavné trénovanie) a následne na trénovacej časti množiny DAVIS zloženej zo 60 videí (dodatočné trénovanie). Hlavné trénovanie prebiehalo na výsekoch zo snímok s rozmermi 256×256 a následné dotrénovanie

na výsekoch s rozmermi $256 \times 512(vka \times rka)$, pričom z videí z množiny DAVIS bola použitá iba každá piata snímka pre simuláciu rýchleho pohybu. Rýchlosť učenia bola v priebehu trénovania na hodnote $1e-5$ a použitá bola optimalizačná metóda *Adam*[1]. Trénovanie modelu trvalo celkovo približne 5 dní [10].

5.1.2 Výsledky

Model postavený na architektúre RGMP využíva pre segmentovanie informácie z prvej anotovanej snímky a z predchádzajúcej predikovanej masky. Práve použitie poslednej predikovanej masky v každom kroku je najslabším článkom architektúry RGMP, o čom je možné presvedčiť sa preskúmaním výstupných segmentačných masiek, ktorých ukážka je zobrazená na Obr.5.1. Už na prvej sekvencii *camel* si môžeme všimnúť, že zatiaľčo približne do polovice videa prebiehala segmentácia v poriadku, v momente príchodu druhej inštancie do záberu začal model segmentovať aj ju, pričom si tento trend zachoval až do skončenia videa. Tento a podobné artefakty sú spôsobené tzv. postupnou akumuláciou chyby, ktorá je výsledkom používania poslednej predikovanej masky pri segmentovaní súčasnej snímky.

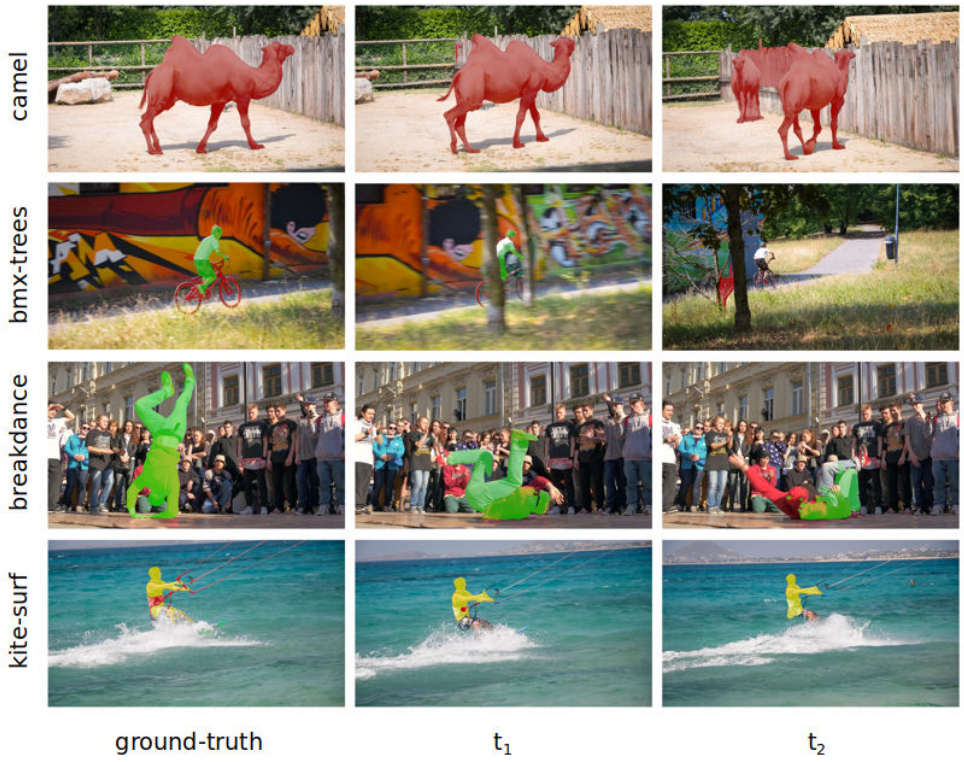
Akumuláciu chyby môžeme ešte lepšie pozorovať vo videu *bmx-trees*. Model postupne od začiatku videa prestáva segmentovať jednu menšiu inštanciu (bicykel), po prvej prekážke si ešte celkom dobre drží segmentovaného jazdca no po ďalšej prekážke stráca už aj jeho a až do konca videa ho nie je schopný znovu identifikovať. Akumulácia chyby sa prejavuje dvoma spôsobmi:

1. Naberanie informácií - v priebehu videa model vysegmentuje na jednej snímke skupinu nesprávnych pixelov a na ďalších snímkach sa táto skupina zväčšuje. Do konca videa tak môže byť segmentovaný objekt, napr. auto, ktorého súčasťou je aj veľká, nesprávne identifikovaná časť pozadia.

2. Strata informácií - objekt alebo časť objektu sa na chvíľu vytratí z dohľadu, no po znovuobjavení ho už model nie je schopný identifikovať. Pri dlhšom videu sa môže stať, že v jeho neskorších fázach nebude sledovaný objekt reprezentovaný ani jedným pixelom.

Ďalší nedostatok modelu je viditeľný vo videu *breakdance*, ktorého súčasťou je jedna inštancia objektu - človek, ktorý vykonáva náhle pohyby a na viacerých snímkach

[1] `https://arxiv.org/abs/1412.6980`



Obr. 5.1: Ukážka výsledkov segmentácie modelu RGMPorigin. Časy t_1 a t_2 označujú iba relatívnu časovú postupnosť.

je zobrazený v nezvyklých polohách. Približne do polovice videa model veľmi dobre segmentuje človeka aj v menej prirodzených polohách, no po niekoľkých náhlych pohyboch začne model strácať informácie o sledovanom objekte, až to dôjde do situácie, ktorá je zobrazená na Obr.5.1 v čase t_2. Model tu už očividne nie je schopný identifikovať celého človeka a segmentuje mu už iba časť nôh. Do konca videa sa modelu nepodarí znovu identifikovať zvyšok tela.

Poslednou ukážkou je video *kite-surf*, kde sa okrem akumulácie chyby prejavuje aj ďalší nedostatok architektúry RGMP - neschopnosť identifikovať príliš malé objekty. V tomto prípade stráca model už po niekoľkých snímkach takmer všetky informácie o vybavení surfera - o surfe a o výstroji, ktorú má surfer na sebe. V takmer celom videu je segmentovaný iba samotný surfer a ďalšie dva objekty nie sú znova identifikované.

V tabuľke 5.2 sú vypísané výsledky metrík, ktoré tento model dosiahol v testovacej fáze ako aj rýchlosť modelu, s akou spracoval vstupné dáta (pre možnosť porovna-

nia s inými modelmi ide rýchlosť segmentácie jednej inštancie). Ako môžeme vidieť, v porovnaní s inými riešeniami vrátane súčasneho SOTA dosahuje RGMP najvyššiu rýchlosť v spracovaní testovacích dát pričom si zachováva presnosť segmentácie porovnateľnú so SOTA riešením.

5.2 Nový model RGMPnew

5.2.1 Popis

Naša verzia modelu sa z pohľadu architektúry nijak výrazne neodlišuje od pôvodného modelu RGMPorigin. Zmenou si však prešli procesy načítavania a predprípravy dát a mierne sme upravili aj testovaciu časť. Najvýraznejšou zmenou je proces trénovania, ktorý sme implementovali od nuly iba na základe niekoľko málo informácií z pôvodnej publikácie[10]. Trénovanie prebiehalo takisto na trénovacej časti množiny DAVIS (60 sekvencií) ale k týmto dátam sme pridali aj 800 videí z množiny YouTube-VOS. Týmto postupom sme sa vyhli nutnosti použitia umelo vygenerovaných snímok zo statických obrázkov, pričom sa nám podarilo zabezpečiť rovnakú, ak nie ešte vyššiu variabilitu trénovacích dát.

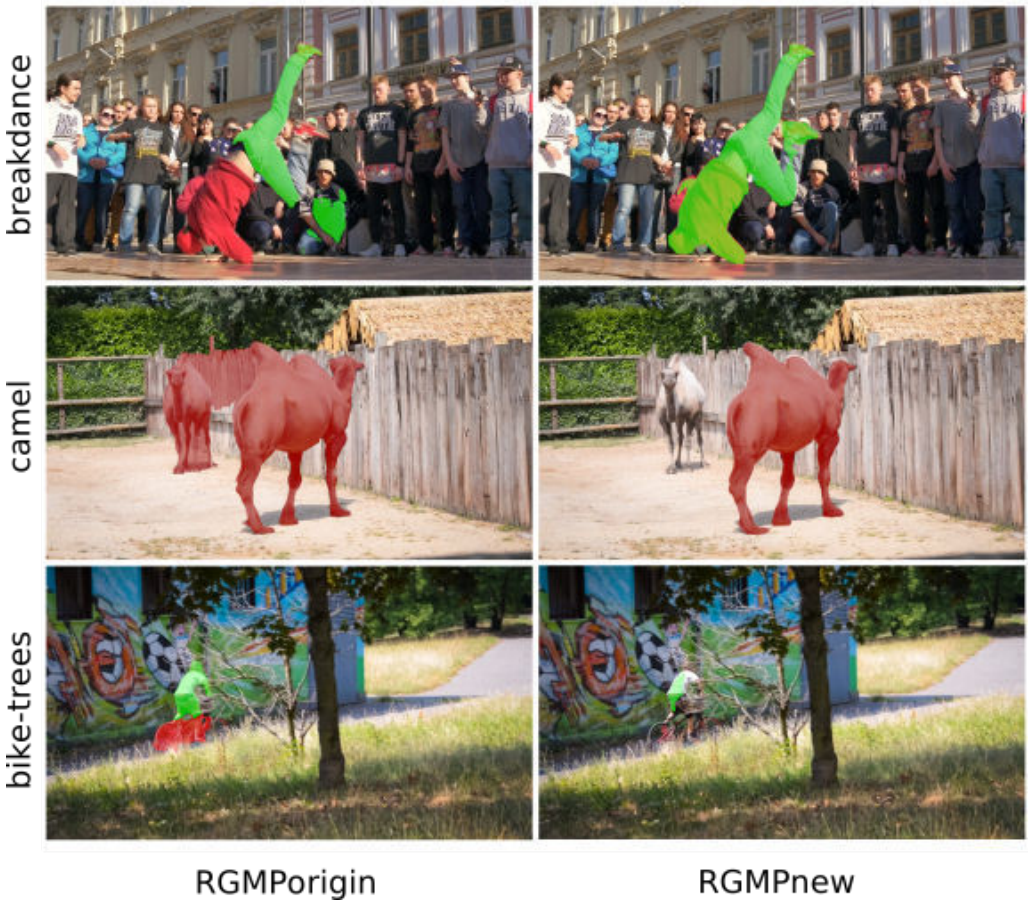
Druhou zmenou vo fáze trénovania je použitie snímok s rozmermi $480 \times 854 (v \times rka)$, čo sú rozmery takmer všetkých videí v množine DAVIS. Množina YouTube-VOS obsahuje videá s vyšším rozlíšením, tie sú preto pred použitím orezané. Vyššie rozlíšenie sme nemohli využiť najmä z dôvodov pamäťového obmedzenia použitej GPU a naopak použitie menších výrezov zo snímok spôsobilo nezanedbateľný prepad presnosti natrénovaného modelu.

Pre pamäťové obmedzenia použitej GPU sme model trénovali na dávkach obsahujúcich iba štyri rôzne sekvencie, z ktorých každá bola zložená zo štyroch náhodne vybraných po sebe idúcich snímok. Architektúra RGMP je postavená tak, že model v testovacej fáze segmentuje snímky na základe predchádzajúcich výstupov, čím sa postupne môže nazbierať chyba segmentovania. Preto je dôležité trénovať podobné modely na dlhších častiach videí aby sa model naučil pracovať s rastúcou chybou a aby vedel lepšie využívať časové a priestorové závislosti vo videách.

Ostatné parametre trénovania ostali rovnaké ako pri pôvodnom modeli - rýchlosť učenia bola nastavená na $1e-5$ a použitá bola optimalizačná metóda *Adam*. Model

sme z dôvodov hardvérových obmedzení trénovali v niekoľkých fázach, celkové trvanie trénovania odhadujeme na 4-5 dni.

5.2.2 Výsledky



Obr. 5.2: Porovnanie výsledkov segmentácie modelov RGMPorigin a RGMPnew v čase t.

Model sme natrénovali niekoľkokrát, vždy v mierne inej konfigurácii, až kým sme nedosiahli presnosť takmer zhodnú s presnosťou pôvodného modelu. Zaujímavé sú rozdiely v jednotlivých metrikách medzi RGMPorigin a RGMPnew. Zatiaľčo hodnoty $JMean$ a $FMean$ (priemerné hodnoty metrík) sú medzi dvomi verziami modelu takmer zhodné, hodnoty návratnosti $JRecall$ a $FRecall$ sú v prípade RGMPorigin približne o 10% vyššie (relatívny prírastok).

Takáto kombinácia hodnôt nám naznačuje dôležitý rozdiel medzi naším a pôvodným modelom. Vyššia hodnota návratnosti, či už ide o F mieru ohraničenia

alebo Jaccardov index značí, že RGMPorigin je schopný v porovnaní s RGMPnew segmentovať viac snímok s hodnotami J a F vyššími ako 50%. Na druhú stranu však vidíme, že samotné priemerné hodnoty $JMean$ a $FMean$ majú oba modely takmer zhodné. Toto správanie môžeme vysvetliť tak, že model RGMPorigin segmentuje každú snímku s trochu nižšou presnosťou akú dosahuje RGMPnew, ale zato sú jeho výsledky konštantné aj pri dlhšej sekvencii a nie je až taký náchylný na akumuláciu chyby z predchádzajúcich predikcií. RGMPnew podáva v bežných situáciách lepšie výsledky, no je náchylnejší na akumuláciu chyby a ak v priebehu videa stratí priveľa informácií o sledovanom objekte, vo zvyšku videa už tento objekt nedokáže znova identifikovať. Toto správanie je zobrazené aj na niekoľkých ukážkových snímkach na Obr.5.2 - na videu *bike-trees* môžeme vidieť, že zatiaľčo RGMPorigin segmentuje objekt aj v prípade jeho čiastočného zakrytia, RGMPnew počas niekoľkých snímok stráca informácie o objekte a do konca videa sa mu už nepodarí tento objekt znova identifikovať.

Na druhú stranu, napr. v prípade videí *breakdance* alebo *camel* zobrazených na Obr.5.2 pozorujeme lepšie výsledky modelu RGMPnew, a to najmä v podobe schopnosti lepšie identifikovať a segmentovať celý objekt. Tieto dve videá sú relatívne stále z pohľadu zmeny pozadia a sledované objekty v nich nie sú nijak „prerušované“ prekážkami alebo odchádzaním mimo záber.

Aby sme zistili dôvod vzniknutých rozdielov medzi fungovaním oboch modelov, uskutočnili sme malý experiment, v ktorom sme natrénovaný model RGMPnew dodatočne dotrénovali na 100 videách z množiny YouTube-VOS, pričom sme ale do procesu trénovania zahrnuli aj rekurentné rozšírenie (4.1.2). Takto trénovaný model vykazoval po niekoľkých desiatkach iterácií zvyšujúce sa hodnoty $JaFRecall$, bohužiaľ sme však pozorovali aj znižovanie hodnôt $JaFMean$. Dôvodom znižovania hodnôt je pravdepodobne hardvérové obmedzenie, ktoré nám nedovolilo trénovať na dostatočne veľkých dávkach, ktoré by zároveň obsahovali dostatočne dlhé sekvencie. Z experimentu tak dedukujeme, že použitie rozsiahlejšej trénovacej množiny pomáha modelu pri segmentácii detailov a zvyšuje jeho schopnosť presnejšie detegovať okraje segmentovaných objektov. Použitie dlhších videí, aj keď v menších dávkach, zlepšuje schopnosť modelu segmentovať objekty za prekážkami prípadne objekty, ktoré počas videa výrazne menia svoj vzhľad. Použitím výkonnejšieho hardvéru by sme boli schopní model natrénovať na dátach vo väčších

dávkach a s väčšou dĺžkou sekvencií, čo by podľa našich predpokladov mohlo ešte významne zvýšiť jeho presnosť.

Vo výsledku sme tak natrénovali model, ktorý podáva voči pôvodnému modelu lepšie výsledky vo väčšine videí okrem malého počtu tých, v ktorých sú objekty veľmi nekonzistentné a príliš často sa pred nimi objavujú prekážky.

5.3 Model RGMPnew + online trénovanie

5.3.1 Popis

Ako už bolo spomenuté, Lucid data dreaming modul (4.2.1) sme implementovali tak, aby bol využiteľný pre dodatočné online trénovanie existujúceho modelu počas testovacej fázy. Používateľ si môže pred začiatkom testovacej fázy online trénovanie nakonfigurovať a to zmenou počtu epoch, počtu umelo vygenerovaných anotovaných snímok a výberom konkrétnych transformácií, prostredníctvom ktorých sa budú z prvej anotovanej snímky generovať ďalšie.

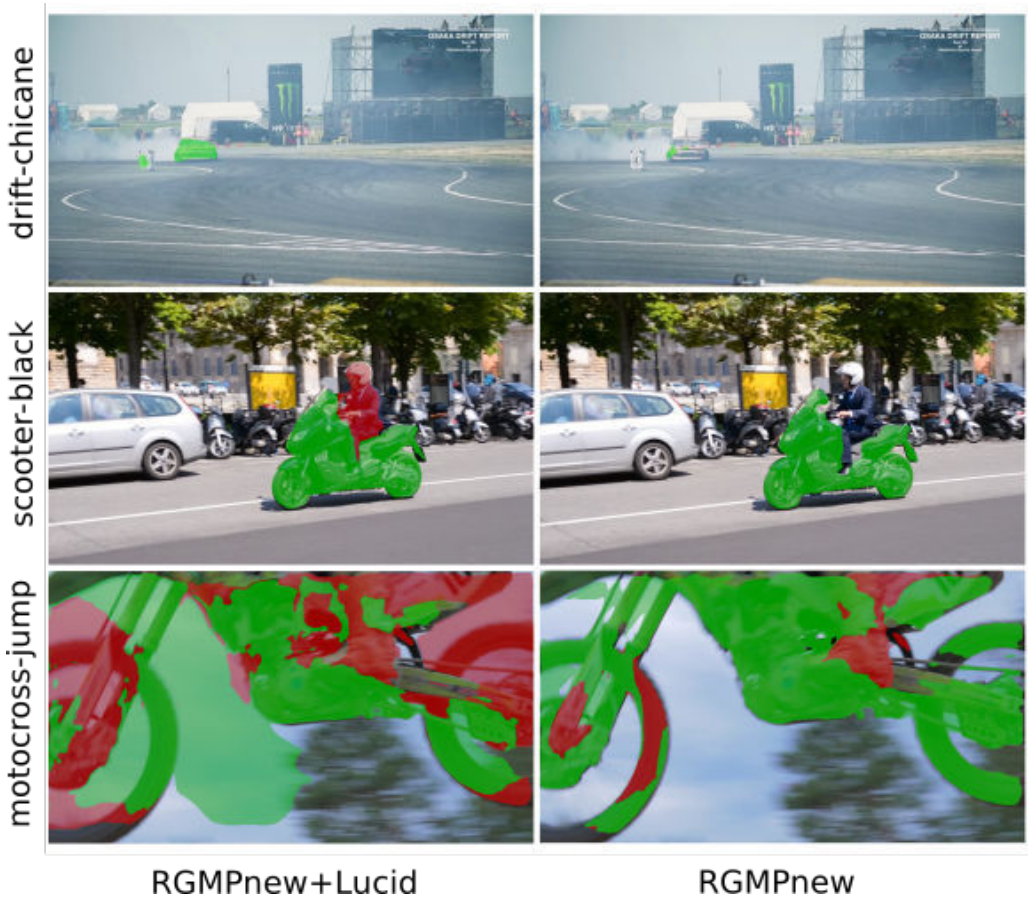
Pre použitie a otestovanie online trénovania sme si zvolili náš model RGMPnew, ktorý približne zodpovedá pôvodnému modelu RGMPorigin. Model sme nijak dodatočne nemodifikovali a pri porovnávaní novozískaných výsledkov sme používali hodnoty metrík, ktoré sú súčasťou tabuľky 5.2.

V experimentálnej fáze našej práce sme najviac času venovali vplyvu online učenia na výsledky segmentácie. Náš model RGMPnew sme online učením dotrénovali v približne tridsiatich osobitných iteráciách, pričom každé dotrénovanie prebiehalo v inej kombinácii počtu epoch, počtu vygenerovaných snímok a s inými transformáciami obrazu.

5.3.2 Výsledky

V procese hľadania najlepšej konfigurácie online trénovania sme vykonali približne 30 experimentov, ktorými sme došli k niekoľkým záverom:

1. Dodatočné trénovanie na umelo vygenerovaných snímkach dokáže pomôcť pri spresnení detailov segmentácie a pri naučení lepšej reprezentácie objektu. Na druhú stranu, ak má samotný model pred dotrénovaním problémy so



Obr. 5.3: Porovnanie výsledkov segmentácie modelov RGMPnew a RGMPnew+Lucid.

segmentáciou dlhých videí kvôli akumulácii chyby, online trénovanie nijak nepomôže k riešeniu tohto problému, dokonca môže v niektorých situáciách uškodiť pri segmentovaní takýchto videí.

2. Prínos každej transformácie obrazu použitej pri generovaní snímok je individuálny a závisí výhradne od aktuálneho videa. Vo všeobecnosti nemá výber použitých transformácií veľký význam pri online trénovaní, a to najmä ak ide o väčšiu množinu rôznych videí.

3. Pôsobenie online trénovania musí byť striktne obmedzené vhodnou kombináciou nízkeho počtu epoch trénovania a nízkej rýchlosti učenia. Vyššia rýchlosť a/alebo počet epoch spôsobia preučenie modelu na týchto snímkach a aj keď prispejú k vyššej presnosti na prvých snímkach, výrazne znížia schopnosť identifikácie objektov v neskorších fázach videa, kedy sú už objekt aj pozadie

Počet epoch	Počet snímok	Learning rate	Rýchlosť spracovania	$JMean$	$FMean$
10	5	1e-6	4s	65.9	69.5
10	30	1e-6	10s	60.5	64.7
30	10	1e-6	9s	61.1	64.8
30	10	1e-7	9s	63.3	67.1

Tabuľka 5.1: Porovnanie vplyvu rôznych konfigurácií online trénovania na presnosť modelu. Zvýraznená je konfigurácia s najlepšími výsledkami.

vo väčšine prípadov výrazne zmenené.

4. Zvyšovanie množstva vygenerovaných snímok je prospešné iba za podmienky, že sa adekvátne k tomu mierne navýši aj „množstvo učenia“ - zvýši sa počet epoch alebo rýchlosť učenia.

5. Ak si zvolíme použitie iba malého množstva transformácií, vygenerované obrázky si budú navzájom veľmi podobné a model sa na nich rýchlejšie preučí.

Ako bolo spomenuté - výber použitých transformácií nemá významný vplyv na výslednú presnosť trénovaného modelu, preto sme naše ďalšie experimenty sústredili na optimalizáciu samotného procesu trénovania. V tabuľke 5.1 sú zobrazené výsledné hodnoty metrík po online trénovaní modelu RGMPnew v štyroch rôznych konfiguráciách. Vybrané boli také konfigurácie, ktoré spôsobili výrazné zmeny v presnosti modelu. Počas týchto experimentov boli povolené všetky dostupné transformácie obrazu: zmena jasu, pretočenie obrazu, rotácia obrazu, zmena mierky, orezanie, náhodné rozmiestnenie sledovaných objektov na pozadí, transformácia spline-u a afinné transformácie. Všetky tieto operácie sa vykonávali zvlášť na sledovanom objekte a zvlášť na pozadí. Prvý záznam v tabuľke 5.1 predstavuje konfiguráciu, ktorá dosiahla najlepší výsledok v testovacej fáze. Položka „počet snímok“ označuje počet umelo vygenerovaných snímok pre online trénovanie. Položka „rýchlosť spracovania“ označuje celkové priemerné spracovanie jednej snímky, pričom tento údaj zahŕňa proporcionálny čas generovania snímok, online trénovania aj testovania.

Z pohľadu kvality segmentovaných snímok pozorujeme výrazné zlepšenie najmä pri videách, pri ktorých náš model RGMPnew príliš skoro stratil informáciu o sledovanom objekte a v priebehu videa nevedel tento objekt znovu identifikovať. Dobrým príkladom sú videá *drift-chicane* alebo *scooter-black*, z ktorých ukážky sú zobrazené aj na Obr.5.3. Obidve videá boli základným modelom RGMPnew dobre segmentované iba počas prvých pár snímok, no už po chvíli stratili sledovaný objekt a znovu ho začali segmentovať až oveľa neskôr vo videu. Online dotrénovanie pomohlo modelu upevniť svoje „vedomosti“ o vzhľade sledovaného objektu a zamedziť tak prípadnej zlej identifikácii tohto objektu.

Priemerné hodnoty metrík pre celú množinu sa nám podarilo použitím online trénovania zvýšiť iba o 2-3%, no preskúmaním výsledkov segmentácie a porovnaním týchto výsledkov medzi niekoľkými verziami modelu sme pozorovali, že online trénovanie dopomohlo k viac stabilným výsledkom naprieč celou množinou. Výsledky segmentovania niektorých videí boli mierne horšie ako s modelom bez online trénovania, no pri asi jednej pätine množiny sme zaznamenali výrazné zlepšenie. V praxi tak online trénovanie má zmysel najmä ak chceme dosiahnuť vyššiu konzistenciu výsledkov na veľkej množine aj za cenu jemnej degradácie výkonnosti na niektorých videách.

5.4 Porovnanie a vyhodnotenie

Experimentmi sme ukázali, že model postavený na architektúre RGMP má potenciál dosiahnuť lepšie výsledky ako jeho pôvodná implementácia, a to v prvom rade natrénovaním modelu na rozsiahlejšej dátovej množine, ako je napr. YouTube-VOS, a v druhom rade implementáciou metódy online trénovania modelu do testovacej fázy, ktorá napomáha zvýšiť konzistentnosť výsledkov naprieč celou množinou.

Výsledky našich modifikácií nie sú tak výrazné, aby zásadne zmenili fungovanie modelu a vo veľkej miere zlepšili jeho výkonnosť, no experimentmi sme ukázali, že architektúra RGMP má priestor na zlepšovanie. Z dôvodu vysokých hardvérových a časových nárokov na trénovanie modelu sme v tejto fáze využili iba približne pätinu dostupných videí z množiny YouTube-VOS. Domnievame sa, že predĺžením doby trénovania a použitím všetkých dostupných anotovaných videí by sme dosiahli ďalšie zvýšenie presnosti modelu, a to najmä v podobe lepšej segmentácie

Metóda	J Mean	J Recall	F Mean	F Recall	Rýchlosť
RGMPorigin	64.8	74.1	68.6	77.7	0.18s
RGMPnew	63.6	69.4	67.6	73.2	0.18s
RGMPnew+Lucid	65.9	68.2	69.5	71.8	6s
PReMVOS	73.9	83.1	81.8	88.9	>20s
OSVOS-S	64.7	74.2	71.3	80.7	4.5s
OnAVOS	61.6	67.4	69.1	75.4	13s
OSVOS	56.6	63.8	63.9	73.8	9s
CINM	67.2	74.5	74.0	81.6	-

Tabuľka 5.2: Kvantitatívne vyhodnotenie a porovnanie metód segmentácie. Zvýraznená zelenou je najlepšia alternatíva modelu na architektúre RGMP, modrou aktuálne SOTA riešenie.

dlhších videí a videí, ktoré zahŕňajú rôzne výzvy z pohľadu segmentácie.

Model RGMP podáva veľmi dobré výsledky a pracuje rýchlo, čo ho predurčuje na použitie na dátach v reálnom čase. Veľkým nedostatkom, ktorý ho z takéhoto použitia čiastočne diskvalifikuje, je jeho náchylnosť na akumuláciu chyby v dlhších videách, čo by v reálnom čase mohlo už po niekoľkých sekundách spôsobiť neschopnosť podávať relevantné výsledky. Tento problém je spôsobený veľkou závislosťou modelu od jeho predošlých predikcií, čo v konzistentných videách dokáže výrazne pomôcť so sledovaním objektu, no naopak pri videách s náhlymi zmenami vzhľadu objektu a s prekážkami má táto vlastnosť opačný efekt a potláča schopnosť modelu identifikovať stratené alebo náhle zmenené objekty.

Iné modely, ako napr. PReMVOS [19], využívajú tzv. re-identifikáciu objektov, ktorá pomáha práve v situáciách, keď model začína strácať sledovaný objekt. Tieto pokročilejšie riešenia však pracujú oveľa pomalšie a teda ani tie nie sú vhodné na použitie na dátach v reálnom čase.

Modul COB sme v našej práci neimplementovali z dôvodu výraznej časovej náročnosti experimentálnej fázy už len pri trénovaní základného modelu, ktoré sme museli opakovať niekoľkokrát pre dosiahnutie požadovaných výsledkov a ďalej pri rozsiahlom experimentovaní s online trénovaním.

6 Ďalšia práca

Ďalšia práca na vylepšovaní modelu spočíva v prvom rade v implementácii COB modulu, ktorý bol pôvodne súčasťou aj tejto práce a ktorý by mohol dopomôcť k zvýšeniu presnosti výsledkov najmä z pohľadu lepšej segmentácie okrajov objektov. Ako sme spomenuli vo vyhodnotení, architektúra RGMP je od začiatku postavená tak, že sa vo veľkej miere spolieha na predchádzajúce predikované masky, čo jej v hraničných prípadoch výrazne znižuje schopnosť identifikácie a následnej segmentácie objektov vo videu. Tento nedostatok je možné vyriešiť implementáciou modulu re-identikácie objektov, akú používa napr. riešenie [19].

V súčasnosti existujú desiatky rôznych modelov a variácií základných architektúr neurónových sietí, ktoré dokážu riešiť segmentáciu videí na dostatočne vysokej úrovni. Takmer každé z týchto riešení využíva jeden alebo niekoľko unikátnych prístupov k zvýšeniu presnosti modelu. Štúdiom týchto riešení by bolo možné vybrať niekoľko ďalších modifikácií, ktoré by mohli mať potenciálny prínos pre architektúru RGMP, implementovať a otestovať ich. Ide napríklad o využitie optického toku [16] medzi jednotlivými snímkami, hĺbkové mapy [9], úprava výsledkov použitím DenseCRF [42], spomínaný re-identifikačný modul [19] a ďalšie vylepšenia.

Záver

Štúdiom vedeckých publikácií zameraných na oblasti počítačového videnia, strojového učenia a neurónových sietí sme si utvorili teoretický prehľad o týchto témach, získané poznatky sme následne spracovali a zahrnuli v diplomovej práci. Z ďalších publikácií sme získali informácie o rôznorodých riešeniach problému segmentácie videa. Dve architektúry, ktoré využíva väčšina týchto modelov, podrobne popísali a vzájomne porovnali z pohľadu ich stavby, procesu trénovania a vlastností.

Popri štúdiu modelov určených na segmentáciu sme si urobili prehľad aj o dátových množinách, ktoré sa v tejto oblasti využívajú pre účely trénovania a testovania modelov. Dve najvýznamnejšie dátové množiny sme popísali v teoretickej časti práce.

Z existujúcich riešení segmentácie videa sme vybrali jedno s dobrým pomerom rýchlosti spracovania dát a výslednej presnosti, ktoré sme na základe publikácie a existujúceho čiastkového kódu implementovali a pomocou ktorého sme natrénovali model schopný segmentácie videa s čiastočným dohľadom. Nami natrénovaný model sa vyznačuje podobnou rýchlosťou a presnosťou segmentácie ako pôvodný model od autorov danej publikácie.

V experimentálnej časti práce sme sa zamerali na vplyv zmeny trénovacích dát a takisto vplyv online trénovania modelu v testovacej fáze. Ukázali sme, že dátové množiny, ktoré sú v čase práce dostupné, sú s ohľadom na kapacitu existujúcich modelov dostatočne rozsiahle a nepredstavujú obmedzenie pri trénovaní, ako tomu bolo iba niekoľko rokov dozadu. Takisto sme ukázali, že použitie dodatočného online trénovania v priebehu testovacej fázy má pre model postavený na architektúre RGMP zmysel, a to ako z pohľadu zvýšenia celkovej presnosti segmentácie, tak aj z pohľadu zvýšenia konzistentnosti výsledkov naprieč dátovou množinou.

Literatúra

1. CASTLE, Nikki. *What is Semi-Supervised Learning?* Dostupné tiež z: https://www.datascience.com/blog/what-is-semi-supervised-learning.
2. KARPATHY, Andrej. *CS231n: Convolutional Neural Networks for Visual Recognition*. Dostupné tiež z: http://cs231n.github.io/convolutional-networks/.
3. KARPATHY, Andrej. *The Unreasonable Effectiveness of Recurrent Neural Networks*. Dostupné tiež z: http://karpathy.github.io/2015/05/21/rnn-effectiveness/.
4. PATEL, Shyamal; PINGEL, Johanna. *Introduction to Deep Learning: What Are Convolutional Neural Networks?* Dostupné tiež z: https://www.mathworks.com/videos/introduction-to-deep-learning-what-are-convolutional-neural-networks--1489512765771.html.
5. SALIAN, Isha. *What's the Difference Between Supervised, Unsupervised, Semi-Supervised and Reinforcement Learning?* Dostupné tiež z: https://blogs.nvidia.com/blog/2018/08/02/supervised-unsupervised-learning/.
6. SILVER, David. *Deep Reinforcement Learning*. Dostupné tiež z: https://deepmind.com/blog/deep-reinforcement-learning/.
7. CRESWELL, A.; WHITE, T.; DUMOULIN, V.; ARULKUMARAN, K.; SENGUPTA, B.; BHARATH, A. A. Generative Adversarial Networks: An Overview. *IEEE Signal Processing Magazine*. 2018, roč. 35, č. 1, s. 53–65. Dostupné tiež z: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8253599&isnumber=8253412.
8. GUO, P.; ZHANG, L.; ZHANG, H.; LIU, X.; REN, H.; ZHANG, Y. Adaptive Video Object Segmentation with Online Data Generation. *The 2018 DAVIS Challenge on Video Object Segmentation - CVPR Workshops*. 2018.

9. HE, L.; WANG, G.; HU, Z. Learning Depth From Single Images With Deep Neural Network Embedding Focal Length. *IEEE Transactions on Image Processing*. 2018, roč. 27, č. 9, s. 4676–4689.

10. OH, Seoung Wug; LEE, Joon-Young; SUNKAVALLI, Kalyan; KIM, Seon Joo. Fast Video Object Segmentation by Reference-Guided Mask Propagation. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

11. S. XU L. Bao, P. Zhou. Class-Agnostic Video Object Segmentation without Semantic Re-Identification. *The 2018 DAVIS Challenge on Video Object Segmentation - CVPR Workshops*. 2018.

12. XU, Ning; YANG, Linjie; FAN, Yuchen; YUE, Dingcheng; LIANG, Yuchen; YANG, Jianchao; HUANG, Thomas S. YouTube-VOS: A Large-Scale Video Object Segmentation Benchmark. *CoRR*. 2018, roč. abs/1809.03327. Dostupné tiež z: `http://arxiv.org/abs/1809.03327`.

13. GIBSON, Adam; PATTERSON, Josh. *Deep Learning*. 2017. ISBN 9781491924570. Dostupné tiež z: `https://www.oreilly.com/library/view/deep-learning/9781491924570/ch04.html`.

14. CHENG, J.; LIU, S.; TSAI, Y.-H.; HUNG, W.-C.; GUPTA, S.; GU, J.; KAUTZ, J.; WANG, S.; YANG, M.-H. Learning to Segment Instances in Videos with Spatial Propagation Network. *The 2017 DAVIS Challenge on Video Object Segmentation - CVPR Workshops*. 2017.

15. CHOLLET, Francois. *Deep Learning with Python*. 1st. Greenwich, CT, USA: Manning Publications Co., 2017. ISBN 9781617294433.

16. KHOREVA, A.; BENENSON, R.; ILG, E.; BROX, T.; SCHIELE, B. Lucid Data Dreaming for Object Segmentation. *The 2017 DAVIS Challenge on Video Object Segmentation - CVPR Workshops*. 2017.

17. KRIZHEVSKY, Alex; SUTSKEVER, Ilya; HINTON, Geoffrey E. ImageNet Classification with Deep Convolutional Neural Networks. *Commun. ACM*. 2017, roč. 60, č. 6. ISSN 0001-0782.

18. LE, T.-N. et al. Instance Re-Identification Flow for Video Object Segmentation. *The 2017 DAVIS Challenge on Video Object Segmentation - CVPR Workshops*. 2017.

19. LI, X.; QI, Y.; WANG, Z.; CHEN, K.; LIU, Z.; SHI, J.; LUO, P.; LOY, C. Change; TANG, X. Video Object Segmentation with Re-identification. *The 2017 DAVIS Challenge on Video Object Segmentation - CVPR Workshops*. 2017.

20. MANINIS, Kevis-Kokitsi; PONT-TUSET, Jordi; ARBELÁEZ, Pablo Andrés; GOOL, Luc Van. Convolutional Oriented Boundaries: From Image Segmentation to High-Level Tasks. *CoRR*. 2017, roč. abs/1701.04658. Dostupné tiež z: `http://arxiv.org/abs/1701.04658`.

21. PENG, Chao; ZHANG, Xiangyu; YU, Gang; LUO, Guiming; SUN, Jian. Large Kernel Matters - Improve Semantic Segmentation by Global Convolutional Network. *CoRR*. 2017, roč. abs/1703.02719. Dostupné tiež z: `http://arxiv.org/abs/1703.02719`.

22. PONT-TUSET, Jordi; PERAZZI, Federico; CAELLES, Sergi; ARBELÁEZ, Pablo; SORKINE-HORNUNG, Alexander; VAN GOOL, Luc. The 2017 DAVIS Challenge on Video Object Segmentation. *arXiv:1704.00675*. 2017. Dostupné tiež z: `https://arxiv.org/abs/1704.00675`.

23. SHABAN, A. et al. Multiple-Instance Video Segmentation with Sequence-Specific Object Proposals. *The 2017 DAVIS Challenge on Video Object Segmentation - CVPR Workshops*. 2017.

24. VOIGTLAENDER, P.; LEIBE, B. Online Adaptation of Convolutional Neural Networks for the 2017 DAVIS Challenge on Video Object Segmentation. *The 2017 DAVIS Challenge on Video Object Segmentation - CVPR Workshops*. 2017.

25. YUHENG, Song; HAO, Yan. Image Segmentation Algorithms Overview. *CoRR*. 2017, roč. abs/1707.02051. Dostupné tiež z: `http://arxiv.org/abs/1707.02051`.

26. CAELLES, Sergi; MANINIS, Kevis-Kokitsi; PONT-TUSET, Jordi; LEAL-TAIXÉ, Laura; CREMERS, Daniel; GOOL, Luc Van. One-Shot Video Object Segmentation. *CoRR*. 2016, roč. abs/1611.05198. Dostupné tiež z: `http://arxiv.org/abs/1611.05198`.

27. CHEN, Liang-Chieh; PAPANDREOU, George; KOKKINOS, Iasonas; MURPHY, Kevin; YUILLE, Alan L. DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. *CoRR*.

2016, roč. abs/1606.00915. Dostupné tiež z: http://arxiv.org/abs/1606.00915.

28. ILG, Eddy; MAYER, Nikolaus; SAIKIA, Tonmoy; KEUPER, Margret; DOSOVITSKIY, Alexey; BROX, Thomas. FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks. *CoRR*. 2016, roč. abs/1612.01925. Dostupné tiež z: http://arxiv.org/abs/1612.01925.

29. KHOREVA, Anna; PERAZZI, Federico; BENENSON, Rodrigo; SCHIELE, Bernt; SORKINE-HORNUNG, Alexander. Learning Video Object Segmentation from Static Images. *CoRR*. 2016, roč. abs/1612.02646. Dostupné tiež z: http://arxiv.org/abs/1612.02646.

30. KURUVILLA, J.; SUKUMARAN, D.; SANKAR, A.; JOY, S. P. A review on image processing and image segmentation. In: *2016 International Conference on Data Mining and Advanced Computing (SAPIENCE)*. 2016, s. 198–203. Dostupné tiež z: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7684170&isnumber=7684105.

31. LI, Yi; QI, Haozhi; DAI, Jifeng; JI, Xiangyang; WEI, Yichen. Fully Convolutional Instance-aware Semantic Segmentation. *CoRR*. 2016, roč. abs/1611.07709. Dostupné tiež z: http://arxiv.org/abs/1611.07709.

32. PERAZZI, F.; PONT-TUSET, J.; MCWILLIAMS, B.; VAN GOOL, L.; GROSS, M.; SORKINE-HORNUNG, A. A Benchmark Dataset and Evaluation Methodology for Video Object Segmentation. In: *Computer Vision and Pattern Recognition*. 2016.

33. PINHEIRO, Pedro H. O.; LIN, Tsung-Yi; COLLOBERT, Ronan; DOLLÁR, Piotr. Learning to Refine Object Segments. *CoRR*. 2016, roč. abs/1603.08695. Dostupné tiež z: http://arxiv.org/abs/1603.08695.

34. HE, Kaiming; ZHANG, Xiangyu; REN, Shaoqing; SUN, Jian. Deep Residual Learning for Image Recognition. *CoRR*. 2015, roč. abs/1512.03385. Dostupné tiež z: http://arxiv.org/abs/1512.03385.

35. IOFFE, Sergey; SZEGEDY, Christian. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *CoRR*. 2015, roč. abs/1502.03167. Dostupné tiež z: http://arxiv.org/abs/1502.03167.

36. HE, Kaiming; SUN, Jian. Convolutional Neural Networks at Constrained Time Cost. *CoRR*. 2014, roč. abs/1412.1710. Dostupné tiež z: `http://arxiv.org/abs/1412.1710`.

37. SHALEV-SHWARTZ, Shai; BEN-DAVID, Shai. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014. ISBN 9781107057135.

38. SUTSKEVER, Ilya; VINYALS, Oriol; LE, Quoc V. Sequence to Sequence Learning with Neural Networks. *CoRR*. 2014, roč. abs/1409.3215. Dostupné tiež z: `http://arxiv.org/abs/1409.3215`.

39. BUNDZEL, Marek; ZOLOTOVÁ, Iveta. *Počítačové videnie v praxi*. 2013.

40. LI, Fuxin; KIM, Taeyoung; HUMAYUN, Ahmad; TSAI, David; REHG, James M. Video Segmentation by Tracking Many Figure-Ground Segments. 2013. Dostupné tiež z: `https://web.engr.oregonstate.edu/~lif/SegTrack2/segtrack2_cameraready.pdf`.

41. ACHANTA, R.; SHAJI, A.; SMITH, K.; LUCCHI, A.; FUA, P.; SÜSSTRUNK, S. SLIC Superpixels Compared to State-of-the-Art Superpixel Methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2012, roč. 34, č. 11, s. 2274–2282. Dostupné tiež z: `http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6205760&isnumber=6304885`.

42. KRÄHENBÜHL, Philipp; KOLTUN, Vladlen. Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials. *CoRR*. 2012, roč. abs/1210.5644. Dostupné tiež z: `http://arxiv.org/abs/1210.5644`.

43. GLOROT, Xavier; BORDES, Antoine; BENGIO, Yoshua. Deep Sparse Rectifier Neural Networks. In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. 2011, zv. 15, s. 315–323. Proceedings of Machine Learning Research. Dostupné tiež z: `http://proceedings.mlr.press/v15/glorot11a.html`.

44. SULAIMAN, S. N.; MAT ISA, N. A. Adaptive fuzzy-K-means clustering algorithm for image segmentation. *IEEE Transactions on Consumer Electronics*. 2010, roč. 56, č. 4, s. 2661–2668. ISSN 0098-3063.

45. BARNES, Connelly; SHECHTMAN, Eli; FINKELSTEIN, Adam; GOLDMAN, Dan B. PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing. *ACM Transactions on Graphics* (*Proc. SIGGRAPH*). 2009, roč. 28, č. 3.

46. GONZALEZ, Rafael C.; WOODS, Richard E. *Digital Image Processing* (*3rd Edition*). Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2006. ISBN 013168728X.

47. MARTIN, D. R.; FOWLKES, C. C.; MALIK, J. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2004, roč. 26, č. 5, s. 530–549. ISSN 0162-8828.

48. SUN, Jian; JIA, Jiaya; TANG, Chi-Keung; SHUM, Heung-Yeung. Poisson Matting. *ACM Trans. Graph.* 2004.

49. GURNEY, Kevin. *An Introduction to Neural Networks*. Bristol, PA, USA: Taylor & Francis, Inc., 1997. ISBN 1857286731.

50. HOCHREITER, Sepp; SCHMIDHUBER, Jürgen. Long Short-Term Memory. *Neural Comput.* 1997, roč. 9, č. 8. ISSN 0899-7667.

51. BENGIO, Y.; SIMARD, P.; FRASCONI, P. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*. 1994, roč. 5, č. 2, s. 157–166. Dostupné tiež z: `http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=279181&isnumber=6922`.

52. FUKUSHIMA, Kunihiko. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*. 1980, roč. 36, č. 4.

53. HUBEL, D.; WIESEL, T. Receptive fields, binocular interaction, and functional architecture in the cat's visual cortex. *Journal of Physiology*. 1962, roč. 160.

Zoznam skratiek

CNN Konvolučná neurónová sieť.

COB Konvolučné orientované kontúry.

RNN Rekurentná neurónová sieť.

SOTA State-of-the-art.

Zoznam príloh

Príloha A Zdrojový kód

Príloha B CD médium – záverečná práca v elektronickej podobe, zdrojový kód, natrénovaný model

Príloha C Používateľská príručka

Príloha D Systémová príručka

Technická univerzita v Košiciach
Fakulta elektrotechniky a informatiky

Metódy hlbokého učenia pre segmentáciu obrazu

Príloha C: Používateľská príručka

2019 **Bc. Maroš Plšík**

1 Požadovaný hardvér

Výstup tejto práce v podobe spustiteľných skriptov bol vytvorený a úspešne otestovaný na počítačovej zostave, podľa ktorej sme zadefinovali nasledujúce minimálne požiadavky:

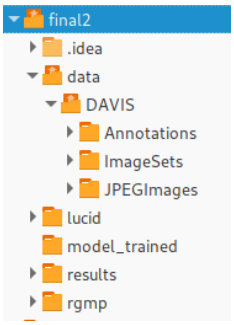
- 2-jadrový CPU Intel Xeon @ 2.2GHz
- 12GB RAM
- GPU s pamäťou aspoň 14GB a s podporou CUDA (napr. Nvidia Tesla K80)
- testovanie: 3GB HDD / trénovanie: 18GB HDD
- pripojenie na internet
- OS Windows, Linux, Mac

Tieto požiadavky sme stanovili na základe procesu trénovania videa, ktorý je náročnejší na hardvérové prostriedky. Fáza testovania by mala byť spustiteľná aj na menej výkonnej stanici.

2 Príprava

2.1 Príprava súborovej štruktúry

Z priloženého CD rozbaľte súbor **source.zip** do vlastného adresára. Z URL `https://data.vision.ee.ethz.ch/csergi/share/davis/DAVIS-2017-trainval-480p.zip` stiahnite súbor a jeho obsah rozbaľte do podadresára **rgmp/data**. Štruktúra adresárov by mala vyzerať tak, ako na Obr.2.1.



Obr. 2.1: Štruktúra adresárov po rozbalení **source.zip**.

2.2 Inštalácia požadovaného softvéru

Na spustenie všetkých skriptov je potrebná inštalácia Python verzie 3.6 alebo vyššej. Pre inštaláciu doplňujúcich knižníc je možné použiť nástroje ako *pip* alebo *Conda*. Všetky požadované knižnice sú vypísané v priloženom súbore **requirements.txt**. V prípade potreby si vytvorte nové virtuálne prostredie prostredníctvom *virtualenv* alebo *Conda*. Všetky potrebné knižnice si môžete nainštalovať príkazom (pip):

```
pip install -r requirements.txt
```

alebo (Conda)

```
conda install --file requirements.txt
```

3 Testovanie modelu

V adresári **rgmp/data** máte uloženú dátovú množinu DAVIS, ktorá pozostáva z 90 videí. Otestovať funkcionalitu modelu môžete na validačnej časti množiny, ktorá sa skladá z 30 videí. Model môžete otestovať naraz na všetkých 30 videách alebo si môžete zvoliť názov videa, o ktoré máte záujem. Názvy videí nájdete v súboroch **rgmp/data/DAVIS/ImageSets/2017/train.txt** a **.../val.txt**. Postup pre spustenie testovania je nasledovný:

1. Rozbaľte **rgmp.zip** do vlastného adresára.
2. Stiahnite súbor z webovej adresy `https://data.vision.ee.ethz.ch/csergi/share/davis/DAVIS-2017-trainval-480p.zip` a rozbaľte ho do podadresára **rgmp/data**.
3. V priloženom súbore **config.py** upravte parameter **config['online']** - podľa toho, či máte záujem o online trénovanie počas testovania. Ak túto možnosť zapnete, môžete si nastaviť dodatočné parametre online trénovania, ktoré sa takisto nachádzajú v danom súbore.
4. Ak chcete trénovať vlastný natrénovaný model, je potrebné skopírovať ho do adresára **rgmp/model_trained** a jeho názov zadať do konfiguračného súboru do položky **config['trained_model_name']**
5. Ak nechcete testovať model na všetkých videách, upravte obsah súboru **rgmp/data/DAVIS/ImageSets/2017/val.txt** tak, že v ňom necháte iba tie názvy videí, ktoré chcete použiť pri testovaní.
6. Spustite skript **test.py**.
7. Po každom spracovanom videu budú vypísané dosiahnuté hodnoty metrík J

index a *F miera ohraničenia*. Na konci celého procesu budú vypísané sprieme-rované hodnoty metrík pre celý otestovaný súbor.

8. Všetky predikované masky sa uložia do zložky **rgmp/results/myresults/**

Alternatívne dáta, ktoré je možné použiť v testovacej fáze, sú testovacie dáta množiny DAVIS, ktoré však obsahujú anotovanú iba prvú snímku videa. URL adresa na stiahnutie dát: `https://data.vision.ee.ethz.ch/csergi/share/davis/DAVIS-2017-test-dev-480p.zip`. Postup testovania týchto dát je zhodný s testovaním na validačnej množine, okrem toho, že na týchto dátach nie je možné vykonať kvantitatívnu evaluáciu.

Model je možné testovať aj na vlastnom videu, vyžaduje si to však zdĺhavú prípravu v podobe konverzie videa na obrázky, úprava rozlíšenia a vytvorenie prvej segmentačnej masky. Z toho dôvodu tento prípad nie je zahrnutý v používateľskej príručke.

4 Trénovanie modelu

1. Pre natrénovanie modelu je potrebné použiť dáta z dátovej množiny YouTube-VOS. Dáta sú dostupné k stiahnutiu na adrese `https://drive.google.com/drive/folders/1bI5J1H3mxsIGo7Kp-pPZU8i6rnykOw7f`. Stiahnite si súbor **train.zip** a rozbaľte ho do dočasného adresára.

2. Z rozbalených dát skopírujte obsah adresára **Annotations** do adresára **rgmp/data/DAVIS/Annotations** a rovnako skopírujte dáta z **JPEGImages** do **rgmp/data/DAVIS/JPEGImages**.

3. V prípade potreby zmeny parametrov trénovania je potrebné zmeniť ich priamo v zdrojovom kóde v súbore **train.py**.

4. Spustite skript **train.py**.

5. Model sa po každých 50 epochách uloží do adresára **rgmp/model_trained**. Model je možné kedykoľvek znova načítať a pokračovať v jeho trénovaní.

Trénovanie modelu do stavu, kedy bude dosahovať podobné výsledky ako náš natrénovaný model RGMPnew môže aj na veľmi výkonnej výpočtovej stanici trvať niekoľko dní. S vyšším množstvom pamäte GPU a operačnej pamäte odporúčame experimentovať aj s parametrami trénovania a snažiť sa maximalizovať ako veľkosť dávok, tak aj dĺžku jednotlivých sekvencií v týchto dávkach. Zvýšenie týchto parametrov môže viesť k lepším výsledkom, než aké sme prezentovali v našej práci. Rovnako môže dôjsť k zlepšeniu aj v prípade využitia celej trénovacej množiny YouTube-VOS, no trénovanie na tejto množine môže trvať až príliš veľké množstvo času.

Technická univerzita v Košiciach
Fakulta elektrotechniky a informatiky

Metódy hlbokého učenia pre segmentáciu obrazu

Príloha D: Systémová príručka

2019 **Bc. Maroš Plšík**

1 RGMP model a utility

Model RGMP a všetky pomocné skripty sú uložené v adresári **rgmp/rgmp** v dvoch súboroch: **model.py** a **utils.py**. Model bol vytvorený vo frameworku PyTorch a jeho fungovanie je prispôsobené na výpočtové stanice s GPU podporujúcou CUDA. Základ modelu tvorí zdrojový kód z `https://github.com/seoungwugoh/RGMP`, ktorý je súčasťou publikácie o architektúre RGMP [10]. My sme tento kód obohatili o proces trénovania, upravili na novšiu verziu Frameworku a modifikovali aby lepšie vyhovoval nášmu použitiu.

1.1 model.py

Tento súbor obsahuje definíciu modelu v podobe neurónových vrstiev a ich konfigurácií. Každý blok vrstiev je definovaný triedou, ktorá je zložená z inicializačnej metódy a z metódy, ktorá určuje spôsob šírenia informácií v sieti:

```
class Encoder(nn.Module):
        def __init__(self):
                ...
        def forward(self, in_f, in_p):
                ...
```

Všetky triedy vyzerajú podobne, pričom výstup jedného bloku je vždy vstupom to do toho ďalšieho.

1.2 utils.py

Obsahom tohto súboru sú pomocné funkcie, ktoré slúžia na načítanie dát, ich spracovanie a jednotlivé metriky, ktoré sa počítajú v testovacej fáze.

```
def to_cuda_variable(xs):
Prenesie vstupné dáta do pamäte grafickej karty.
args:
  xs - vstupné dáta
def upsample(x, size):
def downsample(xs, scale):
Zväčší/zmenší rozmery vstupných obrázkov.
args:
  x/xs - vstupné dáta v podobe obrázkov
out:
  upravené obrázky

def iou(pred, gt):
Vypočíta hodnotu Jaccardovho indexu medzi predikovanou maskou
a ground truth maskou.
args:
  pred - predikovaná maska
  gt - ground truth maska
out:
  hodnota J-indexu

def lucid_loader(config, seq_name):
Načíta dáta vygenerované metódou Lucid data dreaming.
args:
  config - konfigurácia obsahujúca cesty k súborom
  seq_name - názov sekvencie, ktorá sa ma načítať
out:
  masky, snímky a informácie o sekvencii

class DAVIS(Dataset):
Podtrieda Datasetu. Uchováva a umožňuje prácu s obrazovými dátami.
Zahŕňa aj evaluačné funkcie.
        def __init__(self, config, phase, train_frames=0):
        Inicializačná metóda.
```

```
args:
  config - konfigurácia s cestami k súborom, atď.
  phase - fáza train/val/test
  train_frames - koľko snímok má byť na výstupe metódy

def __len__(self):
Metóda na výpočet počtu načítaných sekvencií

def __getitem__(self, index):
Metóda na načítanie dát zo súborov, ich zjednotenie
na fixné rozlíšenie a prípravu na spracovanie neurónovou sieťou.
args:
  index - index práve načítavanej sekvencie
out:
  načítané masky, snímky a informácie o sekvencii

def jaccard_index(self, target='all'):
Vypočíta Jaccardov index pre celú dátovú množinu.
args:
  target - určuje, na ktorých sekvenciách má túto mierku
  vypočítať
out:
  celkový priemer a návratnosť Jaccardovho indexu pre všetky
  zvolené sekvencie

def _seg2bmap(self, seg, width=None, height=None):
Vytvorí binárnu masku, ktorá obsahuje iba okraje sledovaného
objektu.
args:
  seg - segmenty v snímke
  width, height - požadované rozmery výstupnej masky
out:
  bmap - výstupná maska okrajov objektu
```

```
def db_eval_boundary(self, target='all', bound_th=0.008):
Vyhodnotí presnosť predikovaných okrajov ku ground-truth
okrajom objektu, nazvanú aj F-miera ohraničenia.
args:
  target - určuje, na ktorých sekvenciách má danú metriku
  vypočítať
out:
  celkový priemer a návratnosť F-miery ohraničenia pre všetky
  zvolené sekvencie
```

2 Trénovanie a testovanie

2.1 test.py

Slúži na testovanie modelu. Načíta dáta, pripraví ich na spracovanie, v prípade potreby z nich vygeneruje umelé dáta použitím Lucid data dreaming a dodatočne na týchto dátach natrénuje model. Tento model následne použije na predikovanie segmentácie na testovacích videách.

```
def encode_MS(val_F1, val_P1, scales, model):
Použije natrénovaný enkóder na zakódovanie referenčného prúdu dát.
args:
  val_F1 - prvá snímka
  val_P1 - ground-truth maska prvej snímky
  scales - mierky, v ktorých sa spracujú dáta
  model - natrénovaný model
out:
  referenčný prúd dát

def propagate_MS(ref, val_F2, val_P2, scales, model):
Slúži na posúvanie sekvencie cez sieť.
args:
  ref - referenčný prúd dát
  val_F2 - aktuálna snímka
  val_P2 - posledná predikovaná maska
out:
  val_E2 - aktuálna predikovaná maska
```

```
def infer_SO(all_F, all_M, num_frames, scales, model):
def infer_MO(all_F, all_M, num_frames, scales, model):
Inferencia snímok s jednou alebo viacerými inštanciami objektov.
args:
  all_F - všetky snímky z daného videa
  all_M - všetky masky daného videa
  num_frames - počet snímok
  scales - mierky, v ktorých sa spracujú dáta
  model - natrénovaný model
out:
  všetky predikované masky

def online_training(online_data, epochs, lr, model):
Vykoná online trénovanie modelu na zadanej sekvencii umelo
vygenerovaných dát.
args:
  online_data - množina vygenerovaných dát
  epochs - počet epoch online trénovania
  lr - rýchlosť učenia modelu
  model - predtrénovaný model
```

2.2 train.py

Vykonáva trénovanie a validáciu modelu na zadaných dátach. Výstupom je natrénovaný model architektúry RGMP. Súčasťou súboru je aj jedna pomocná funkcia.

```
def propagate_MS_training(ref, F2, P2, model):
Pomocná funkcia, ktorá, podobne ako propagate_MS, slúži
na posúvanie trénovacej sekvencie cez sieť.
  ref - referenčný prúd dát
  val_F2 - aktuálna snímka
  val_P2 - posledná predikovaná maska
out:
  val_E2 - aktuálna predikovaná maska v softmax forme (ešte pred
  binarizáciou)
```

3 Lucid data dreaming

Tento samostatný modul slúži na generovanie množiny anotovaných snímok na základe jednej manuálne anotovanej snímky, pričom tieto vygenerované dáta simulujú pohyb sledovaného objektu vo videu. Proces generovania začína oddelením objektu od pozadia a zaplnením prázdneho miesta tak, aby sa zachovala kontinuita obrazu. Následne sú na pozadie a objekt aplikované transformácie a nakoniec je objekt opäť spojený s pozadím. Kód bol prevzatý z `https://github.com/yelantingfeng/pyLucid` a mierne upravený pre naše potreby, preto v tejto časti nepopíšeme všetky funkcie dopodrobna.

3.1 PatchMatch.py

Tento súbor je k dispozícii v dvoch verziách - **PatchMatchOrig.py**, ktorý je určený pre počítače bez GPU s podporou CUDA a **PatchMatchCuda.py**, ktorý je naopak prispôsobený na prácu s rozhraním CUDA. Obsahom súboru je algoritmus, ktorý dokáže inteligentne zaplniť prázdne miesto v obrázku na základe okolia.

Tento algoritmus je volaný v každej iterácii, kedy je potrebné vygenerovať dáta na základe vstupnej antovanej snímky. Vstupom je pôvodný obrázok a obrázok s vyrezaným sledovaným objektov. Výstupom je snímka so zaplneným prázdnym miestom.

3.2 patchPaint.py

Funkcia **paint**, ktorá je jedinou funkciou v tomto súbore, funguje ako rozhranie medzi algoritmami Lucid dreaming a PatchMatch. Iteratívne volá algoritmus **PatchMatch.py** a generuje snímky so zaplneným pozadím v niekoľkých mierkach.

3.3 lucidDream.py

Hlavný súbor funkcií v celom algoritme Lucid data dreaming. Zahŕňa všetky transformácie obrazu, ktoré sa aplikujú na vstupné dáta - zmenu jasu objektu, rotáciu, modifikácia pozadia alebo sledovaného objektu pomocou splajnových operácií, orezávanie obrazu a ďalšie. Popíšeme aspoň hlavnú funkciu, ktorá je volaná iteratívne pri generovaní každej novej anotovanej snímky:

```
def dreamData(img, gt, bgimg):
Náhodne aplikuje rôzne transformácie obrazu na vstupné dáta.
args:
  img - pôvodná snímka
  gt - anotácia snímky img
  bgimg - snímka pozadia s vyrezaným objektom a zaplneným
  prázdnym miestom
out:
  im_1 - snímka vygenerovaná transformáciou pôvodnej snímky
  gt_1 - maska zodpovedajúca snímke im_1
  bb_1 - maska, ktorá simuluje predchádzajúcu predikovanú masku
```

3.4 lucid_generate.py

Tento súbor slúži ako rozhranie medzi procesom online trénovania modelu a procesom generovania snímok pomocou Lucid dreaming algoritmu. Obsahuje jedinú funkciu:

```
def lucid_generate(qty, seq_name):
Podľa požadovaného počtu snímok zavolá iteratívne funkciu dreamData
a vygenerované snímky s maskami uloží do adresára.
args:
  qty - požadovaný počet anotovaných snímok
  seq_name - názov videa, z ktorého sa má vybrať prvá snímka,
  na základe ktorej sa vygenerujú nové snímky
out:
  snímky, k nim vytvorené masky a simulované predchádzajúce
  masky sú uložené do priradeného adresára
```