VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

HLUBOKÉ NEURONOVÉ SÍTĚ PRO KLASIFIKACI OBJEKTŮ V OBRAZE

DEEP NEURAL NETWORKS FOR CLASSIFYING OBJECTS IN AN IMAGE

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. TOMÁŠ MLYNARIČ**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. MICHAL HRADIŠ, Ph.D.**
SUPERVISOR

BRNO 2018

Zadání diplomové práce/19405/2017/xmlyna06

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2017/2018

Zadání diplomové práce

Řešitel: **Mlynarič Tomáš, Bc.**
Obor: Počítačová grafika a multimédia
Téma: **Hluboké neuronové sítě pro klasifikaci objektů v obraze**
Deep Neural Networks for Classifying Objects in an Image
Kategorie: Zpracování obrazu

Pokyny:

1. Prostudujte možnosti použití hlubokých neuronových sítí pro klasifikaci různých tříd v obraze monokulární kamery z běžného silničního provozu.
2. Navrhněte vhodný přístup s využitím hluboké neuronové sítě pro detekci objektů v obraze (např. vozovka/chodník, vozidlo, chodec/cyklista, dopravní značení, vegetace, budova, obloha).
3. Připravte vhodnou anotovanou datovou sadu.
4. Navržený postup implementujte s využitím vhodných knihoven.
5. Ověřte funkčnost a stanovte úspěšnost klasifikace na reálných datech ze silničního provozu.
6. Vytvořte plakát a krátké video prezentující klíčové výsledky vašeho řešení.

Literatura:

- Bradski G. R., Kaehler A. *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly Media, Inc. 2008.
- M. Sonka, V. Hlaváč, R. Boyle. *Image Processing, Analysis, and Machine Vision*, CL-Engineering, ISBN-13: 978-0495082521, 2007.
- Dále dle pokynu vedoucího.

Při obhajobě semestrální části projektu je požadováno:

- Body 1, 2, 3 a částečně bod 4.

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci dřívějších projektů (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Hradiš Michal, Ing., Ph.D.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2017
Datum odevzdání: 23. května 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 66 Brno, Božetěchova 2
L.S.

doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Abstrakt

Tato práce se zabývá klasifikací objektů v obraze s použitím hlubokých neuronových sítí. Jako algoritmus pro klasifikaci byla zvolena segmentace celé scény pracující na video sekvencích a využívající informace mezi dvěma snímkami videa. Pro tuto úlohu bylo použito extrahování informací pomocí optického toku, na základě kterého byly dále warpovány aktivační mapy vrstev neuronových sítí. Dvě architektury neuronových sítí byly upraveny pro práci s videem, na kterých byly následně provedeny experimenty. Výsledky experimentů ukazují, že použití videa umožňuje zlepšit přesnost (IoU) vůči stejné architektuře pracující s obrázky.

Abstract

This paper deals with classifying objects using deep neural networks. Whole scene segmentation was used as main algorithm for the classification purpose which works with video sequences and obtains information between two video frames. Optical flow was used for getting information from the video frames, based on which features maps of a neural network are warped. Two neural network architectures were adjusted to work with videos and experimented with. Results of the experiments show, that using videos for image segmentation improves accuracy (IoU) compared to the same architecture working with images.

Klíčová slova

segmentace obrazu, monokulární kamera, hluboké neuronové sítě, video, optický tok, warping, Cityscapes, Keras, Tensorflow

Keywords

image segmentation, monocular camera, deep neural networks, video, optical flow, warping, Cityscapes, Keras, Tensorflow

Citace

MLYNARIČ, Tomáš. *Hluboké neuronové sítě pro klasifikaci objektů v obraze*. Brno, 2018. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Michal Hradiš, Ph.D.

Hluboké neuronové sítě pro klasifikaci objektů v obraze

Prohlášení

Prohlašuji, že jsem tuto práci vypracoval samostatně pod vedením pana Ing. Michala Hradiše, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Tomáš Mlynarič
21. května 2018

Poděkování

Tato práce vznikla za podpory projektů CERIT Scientific Cloud (LM2015085) a CESNET (LM2015042) financovaných z programu MŠMT Projekty velkých infrastruktur pro VaVaI. Chtěl bych také poděkovat všem, kteří mne při práci na tomto projektu podporovali, především mému vedoucímu, rodině a přítelkyni. Velké díky patří také bratrovi za poskytnutí hardwaru pro trénování sítí.

Obsah

Kapitola 1

Úvod

Vidění je jedním z biologických smyslů živočišné říše. Ať už se jedná o primitivní stvoření či inteligentní bytosti, vždy jsou vybaveny jistým smyslem pro vnímání svého okolí. Co se týče počítačů, ty dostaly tuto schopnost již v 60. létech minulého století, kdy začaly první pokusy s algoritmy pro získávání informací z obrazu. Za tuto dobu prošlo počítačové vidění velkými změnami, jež umožnilo využití těchto algoritmů napříč mnoha odvětvími informačních technologií. V dnešní době je počítačové vidění aplikováno v různých oborech, ať už se jedná o rozpoznávání tváří na sociálních sítích, kontrolu kvality výrobků v industriálním prostředí či nově vznikající obor samořídítelných vozidel. Myšlenka autonomních vozidel se zrodila již dávno, nicméně realizace je stále na začátku, a to i přes to, že se tento obor v posledních letech začal značně rozvíjet a mnoho automobilových firem dnes investuje do výzkumu týkajícího se možnosti řídit vozidlo bez řidiče. Slovní spojení autonomní vozidlo (*autonomous car*) se začalo razantně používat a tento trend se za posledních deset let zvýšil v podstatě o 100%[1]. Co ale způsobuje, že je dnes taková poptávka po této technologii a díky čemu začíná být technologie proveditelná? Jako impulz můžeme brát vývoj hardwaru, přesněji grafických karet a současně vývoj strojového učení, konkrétně hlubokých neuronových sítí. Synergie těchto dvou odvětví umožnila použití umělé inteligence pro specifické úlohy, jakými je například klasifikace objektů v obraze, které do té doby nedávaly buďto dobré výsledky nebo byly příliš výpočetně náročné.

Princip samořídítelných aut je postaven na získávání informací okolo vozidla v reálném čase, na základě kterých může vozidlo bezpečně zvládnout řízení. V dnešní době většina samořídítelných aut využívá senzor, tzv. *LIDAR* (*Light Detection And Ranging*), který umožňuje na základě rotujících laserových paprsků „naskenovat“ 3D okolí vozidla a poté tyto informace analyzovat pro orientaci ve scéně. Tyto senzory jsou však velmi drahé, co (zatím) omezuje možnosti masového rozšíření těchto vozidel. Druhým faktem je, že je tento senzor obvykle umístěný na střechu vozidla, co kazí dojem estetičnosti a rotující součástky uvnitř něj nejsou velmi robustní řešení. Lidé však pro zvládnutí řízení nepotřebují znát kompletní scénu okolo vozidla a spoléhají na vizuální vjemy scény, z velké části, před sebou. V takovém případě dostatečně přesný a rychlý systém pro rozpoznání scény z kamery vozidla (případně několika kamer) by mohl fungovat podobně a více zpřístupnit autonomní auta.

Cílem této práce je představit systém pro klasifikaci objektů v obraze pouliční scény s použitím hlubokého učení. Výstupem systému bude rozpoznání celé scény díky použití segmentace obrazu. Jako vstup tohoto systému má sloužit klasický výstup z monokulární

[1] Informace z webu *Google Trends*, dostupné na https://trends.google.com/trends/explore?date=2008-05-16%202018-05-16&q=autonomous%20cars

kamery, tzn. nepoužívat specializované kamery pro získání podrobných informací o scéně (např. stereo kamery získávající hloubková data). Většina modelů neuronových sítí pro segmentaci obrazu dnes pracuje na jednotlivých obrázcích. Jelikož je zadáním této práce využití kamery, je možné využívat skutečnosti, že ve video záznamech na sebe jednotlivé snímky videa navazují, co umožňuje tyto informace využívat ve prospěch lepší segmentace pouliční scény.

V jednotlivých kapitolách jsou postupně představeny informace projednávající tuto problematiku. Kapitola 2 představuje základní úrovně klasifikace objektů v obraze a následně projednává o existujících metodách detekce objektů a segmentace scény. Část kapitoly se také věnuje využívání informací z video sekvencí a existujícím pracím používající tyto principy. V kapitole 3 jsou prezentovány datové sady umožňující použití pro segmentační úlohy pouliční scény a také, která datová sada byla v případě této práce použita. Specifická problematika týkající se této práce je v kapitole 4, kde jsou popsány použité architektury neuronových sítí a implementace využívání informací z videa a princip inference neuronové sítě na videu. Experimenty nad navrženými modely a jejich analýza jsou pak prezentována v kapitole 5.

Kapitola 2

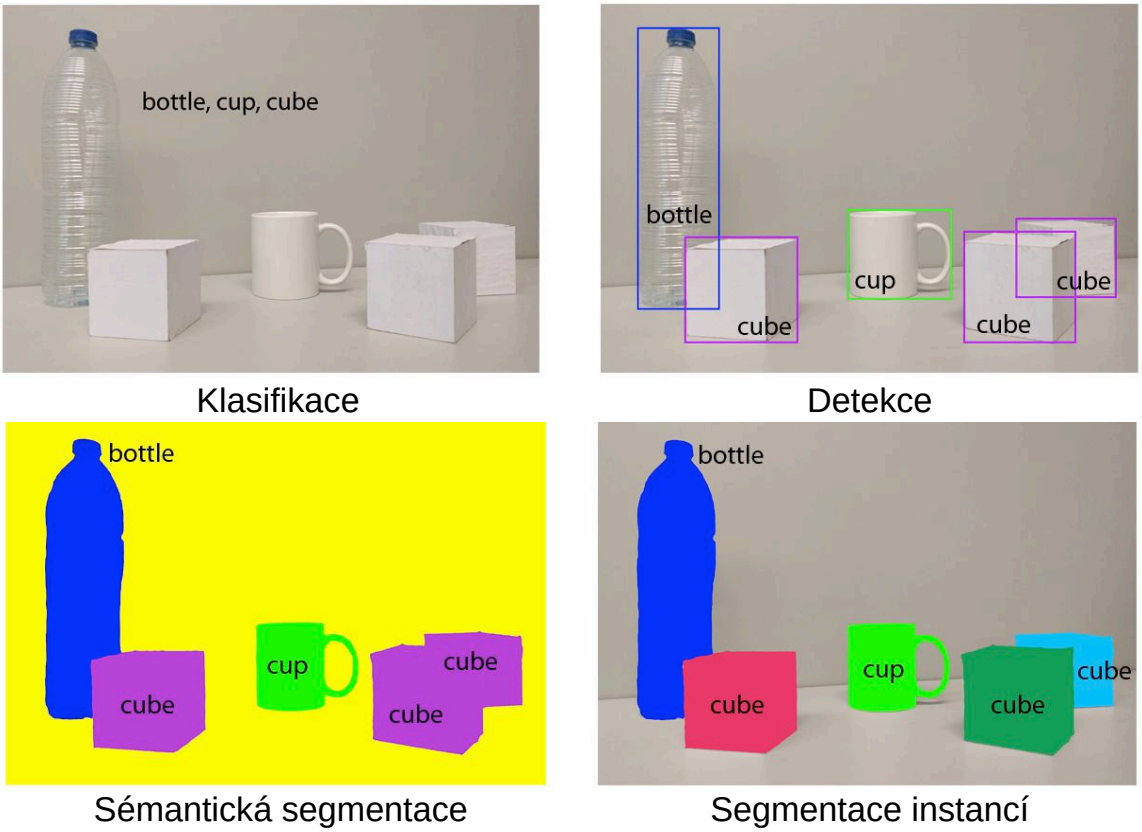
Klasifikace objektů v obraze

Tato kapitola se zabývá teorii klasifikace objektů v obraze a existujícími možnostmi této problematiky. Nejdříve jsou představeny různé úrovně klasifikace objektů, dále pak v sekci 2.1, resp. 2.2 jsou představeny specifičtější úlohy – detekce objektů, resp. segmentace. Následující sekce 2.3 ukazuje metody pracující s video sekvencemi a nakonec v sekci 2.4 je představeno porovnání metod zmiňovaných v této práci. Cílem této kapitoly je tedy představit možnosti, jak lze v dnešní době řešit problematiku klasifikace objektů v obraze a ukazuje některé významné práce, které se těchto problémů týkají.

Problematika klasifikace objektů v obraze je jednou z několika typických úloh počítačového vidění. V této oblasti je problematika aplikována již dlouhou dobu a existuje spousta algoritmů, které tento problém řeší klasickými metodami (bez použití neuronových sítí). Mezi algoritmy pracující s klasifikačními problémy je možné řadit například *Histogram of Oriented Gradients* (`HOG`), *Scale-Invariant Feature Transform* (`SIFT`) či *Support Vector Machines* (`SVM`). Přestože jsou algoritmy do jisté míry funkční, často jsou výpočetně náročné. V roce 2012 však přišla změna díky představení hluboké neuronové sítě *AlexNet*, která, v soutěži *Large Scale Visual Recognition Challenge* (`ILSVRC`), pokořila ostatní týmy v přesnosti (metrika *accuracy*) o 10.8%, takže práce byla o 41% lepší než jiné práce té doby. Tato událost způsobila novodobou revoluci v oblasti počítačového vidění a ukázala, že princip hlubokého učení dokáže podstatně lépe řešit úlohy pracující s počítačovým viděním. [9]

Existuje několik úrovní přístupů, jak je možné objekty v obraze klasifikovat (představeny na obrázku 2.1). Tyto úrovně se liší jak množstvím zpracovaných informací, tak, především, různými typy výstupů a mírou informace o objektech v daných výstupech. [22][25]

Klasifikace Základním algoritmem je klasifikace celého vstupního obrazu. Vstupní obraz je zpracováván jako celek a je získána pouze pravděpodobnost, do jaké třídy daný objekt v obrazu patří. Obvykle je výstupem vektor pravděpodobností všech známých (trénovaných) tříd, kde právě jedna by měla být správně. V případě více objektů ve vstupním obrazu, klasifikátor může nastavit pravděpodobnost obou objektu jako vysokou, nicméně to záleží na způsobu klasifikace jednotlivých algoritmů. Datové sady pro tyto algoritmy (např. *ImageNet*) obvykle obsahují velký počet obrázků, které mají malé rozměry (např. zmiňovaný *ImageNet* obsahuje obrázky velikosti 224×224) a je na nich výřez daného objektu pro klasifikaci.

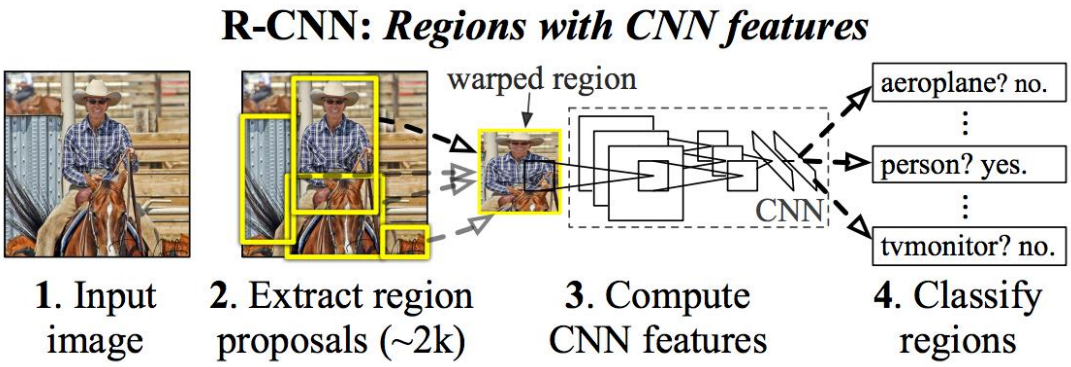


Obrázek 2.1: Porovnání výsledků různých úloh klasifikace objektů v obraze (zdroj: [8])

Lokalizace Druhým přístupem je současně klasifikace a lokalizace objektu v obrazu. Algoritmy tohoto typu rozšiřují předchozí myšlenku klasifikace o získání dalších informací ke kategorii predikovaného vstupu, a to pozice bodů ohraničujícího obdélníku (*bounding box*) buď jako čtyři rohové body (x_0, y_0, x_1, y_1), nebo dva body a rozměry obdélníku (x_0, y_0, w, h). V případě lokalizace více objektů zároveň jsou tyto algoritmy nedostačující, protože fungují pouze pro získání informací jednoho objektu ve vstupním obrazu. Více objektů nelze analyzovat, protože by bylo potřeba pracovat s předem neznámým počtem objektů.

Detekce Algoritmy detekce objektů rozšiřují koncept lokalizace tím, že umožňují vyhledat ve vstupním obraze neurčitý počet předem známých tříd objektů a k nim přiřadit ohraničující obdélníky. Naivní způsob získání detekce pro různý počet objektů by mohl využívat algoritmus klouzavého okna (*sliding window*) a pro všechny získané regiony aplikovat klasifikaci z předchozích příkladů, nicméně toto řešení nebývá již využíváno, protože je neúprosně výpočetně náročné a obsahuje redundantní výpočty nad vstupním obrazem (překrývající oblasti klouzavého okna jsou zpracovávány vícekrát). Podobné principy však první hluboké neuronové sítě využívaly (popsané v sekci 2.1).

Segmentace Poslední typ získávání informací o objektech v obraze je segmentace, kde je pro každý pixel vstupního obrazu prováděna klasifikace, tedy pro každý pixel je přiřazena třída, do které daný objekt patří. Tímto způsobem jsou získávány masky určující jednotlivé objekty v obrazu. Segmentaci lze provádět dvěma způsoby a získat tak různou míru informace. První způsob je sémantická segmentace, kde pro různé objekty stejného typu je predikována stejná třída (výstupní maska pak může být matice velikosti vstupního obrazu a počtu známých tříd). Druhým způsobem je segmentace instancí, která rozšiřuje sémantickou segmentaci o informace o jednotlivých objektech. U tohoto druhu segmentace budou dva různé objekty patřící do stejné kategorie odlišeny jako dvě různé instance.



Obrázek 2.2: Princip funkčnosti sítě *R-CNN* (zdroj: [11])

2.1 Detekce objektů

Tato sekce popisuje přístupy k detekci předem neznámého počtu objektů na vstupu. Jsou ukázány jak detektory založené na vyhledaných regionech ve vstupním obraze, tak algoritmy fungující nad celým vstupem najednou.

Sítě založené na regionech

Jedny z prvních hlubokých neuronových sítí pro detekci objektů byly konvoluční sítě založeny na vyhledávání regionů zájmů (*region-based convolutional networks*). Tyto regiony byly nejprve získávány jinými algoritmy než samotnou neuronovou sítí a pro každý region následně byla prováděná klasifikační úloha. Až ve vylepšených verzích neuronových sítí byla přidáno získávání regionů jako součást sítě.

R-CNN a odvozené Tyto sítě se skládají z několika modulů, kde každý modul se specifikuje na jednu úlohu (viz obrázek 2.2). Prvním krokem této sítě je získání regionů zájmu (*Region of Interest*), kde autoři uvádějí, že je získáváno cca 2 000 regionů pro každý vstupní obraz. Druhým krokem je inference konvoluční neuronové sítě pro každý takto získaný region a nakonec nad těmito výsledky je provedena klasifikace (pomocí *Support Vector Machine*) a lineární regrese pro získání ohraničujících obdélníků. Jak již bylo zmiňováno v předchozí sekci, pokud se nalezené regiony překrývaly, tato síť provádí redundantní výpočty kvůli inferenci neuronové sítě pro každý tento region. Síť *R-CNN* dosahuje průměrné přesnosti (*mean average precision – mAP*) 53.3% na datové sadě *PASCAL VOC 2012*. Tato síť však

procházela vývojem a tak vznikly dvě další verze – *Fast R-CNN* a *Faster R-CNN*, kterých cílem bylo minimalizovat redundantní výpočty a umožnit tak rychlejší inferenci i trénování.

První úpravou sítě vznikla síť *Fast R-CNN* , která byla zrychlena tím, že je jejím vstupem celý obrázek spolu s navrženými regiony zájmu. Obrázek je nejprve zpracován několika vrstvami sítě pro získání mapy příznaků a současně pro všechny navržené regiony zájmu se projde vrstvou `RoIPooling`, která extrahuje příznaky ze zmíněné mapy pro daný obrázek. Dále je průchod sítí podobny, jako v klasické sítí *R-CNN*, s tím rozdílem, že místo *SVM* je použita funkce *softmax*, která udává pravděpodobnost jednotlivých tříd přes všechny známé třídy. Tato síť na stejné datové sadě dosáhla *mAP* 65.7% a byla zrychlena cca 213×.

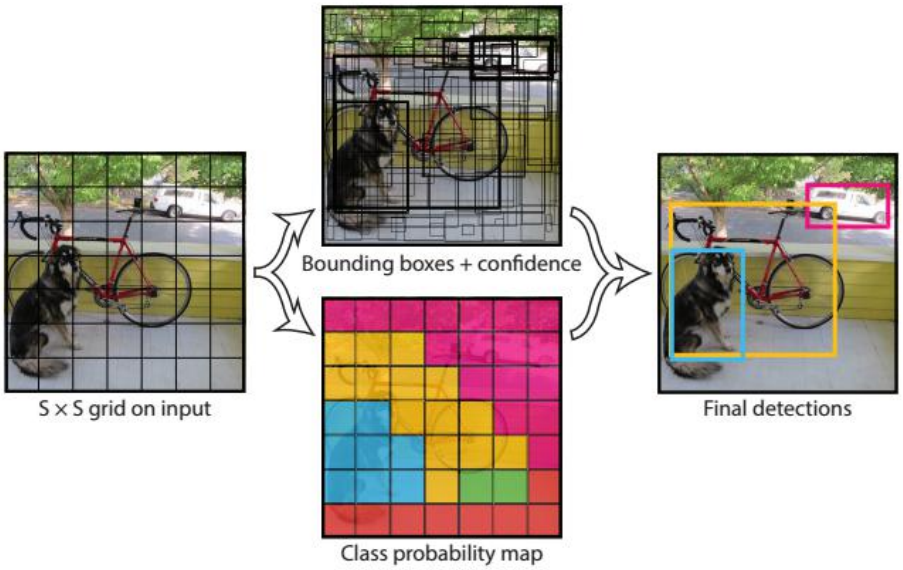
Nedostatkem sítě byl algoritmus navrhující regiony (*selective search*), který je v neuronové síti *Faster R-CNN* nahrazen samostatnou neuronovou sítí *region proposal network (RPN)*. Tato vnitřní síť generuje několik možných regionů pro k různých velikostí ohraničujících obdélníku (*anchor boxes*) a skóre, jestli se na dané lokaci nachází objekt (*objectness*). Výstup této sítě je následně napojen na, v podstatě, obdobu sítě *Fast R-CNN*. Na stejné datové sadě (pouze *PASCAL VOC 2012* pro srovnání) dosáhla síť *mAP* 67.0% [11][10]

Jednoprůchodové detektory

Tato kategorie hlubokých neuronových sítí přede vším upravuje předchozí myšlenku získávání regionů zájmů. Jednoprůchodové detektory (*Single Shot Detectors*) vylepšují tento koncept tak, že úlohy získání regionu zájmu a provedení klasifikace nad těmito regiony jsou spojeny v jedné síti. Obecně tedy tento typ sítí používá pouze jeden průchod neuronovou sítí pro celý obrázek a klasifikace probíhá až nad aktivační mapou (např. použití 1×1 konvoluční vrstvy, která provede klasifikaci nad celým obrázkem). Získání ohraničujícího obdélníku je následně možné druhým algoritmem nad aktivační mapou. Tento algoritmus ale často získá příliš mnoho obdélníků pro stejný objekt, takže výsledek je samostatně spíše nepoužitelný. Pro tento případ je potřeba použít nějakou post-processing metodu, která tento počet obdélníků zredukuje. Jednou z často používaných metod je potlačení nemaximálních hodnot (*non-maxima supression*), která redukuje počet obdélníků pro nějaký objekt tak, že zůstane jen ten nejvhodnější.

YOLO Princip fungování sítě *You Only Look Once (YOLO)* spočívá v rozdělení celého vstupního obrazu na mřížku velikosti $S \times S$ (viz obrázek 2.3), kde pro každou buňku je predikováno B ohraničujících obdélníků a „jistota“ (*confidence*), že se v buňce nějaký objekt nachází (bez ohledu na třídu). Pro každý box je také predikované klasifikační skóre. Podle míry „jistoty“ je možné nastavit, kolik objektů se bude detekovat a kolik zahazovat (výchozí nastavení je detekování 25% objektů). Na výsledek (predikované ohraničující obdélníky) je pak aplikován algoritmus *non-maxima suppression* pro správné potlačení šumu. Problém této sítě však je, že dokáže predikovat pouze jednu třídu v dané buňce a tedy špatně lokalizuje malé objekty.

Později byla tato síť vylepšena a vzniklo *YOLO9000*. Tato síť je plně odvozená od předchozí a obsahuje několik zásadních změn. Důležitou změnou bylo zrušení plně propojených vrstev a zavedení konvolučních vrstev na jejich místě, což umožňuje trénování pro různé velikosti vstupních obrázků (*multi-scale training*). Místo rozdělení obrázků na mřížku byly použity *anchor boxy* podobně jako v *Faster R-CNN*. Autoři také využili několik datových sad (*COCO*, *ImageNet* a *WordTree*), které byly zkombinovány pro účely této práce, což zaručilo lepší generalizaci sítě. [19][20]



Obrázek 2.3: Princip sítě *YOLO* (zdroj: [19])

SSD Podobně jako předchozí síť funguje také *Single Shot Detector (SSD)*. Jako základní síť je použita modifikovaná verze *VGG16* pro získání mapy příznaků. Lokalizace objektů probíhá pomocí *anchor boxů* obdobně jako u *Faster R-CNN*, nicméně ty jsou navrhovány v různých částech sítě (s různým rozlišením), čímž síť dokáže dobře detekovat různě velké objekty. K potlačování více detekcí jednoho objektu je také použitý algoritmus *non-maximum supression*. [15]

2.2 Segmentace obrázků

Předchozí sekce popisovala, jakým způsobem lze detekovat objekty v obrázku a získat jejich lokaci uvnitř obrazu pomocí ohraničujících obdélníků. Pokud by ale bylo potřeba detekovat spojité části v obrázku, jako je například, chodník či čáry na vozovce, obdélníky by byly v této situaci nevhodné, protože tyto objekty můžou zahrnovat třeba i polovinu scény a mít detekovaný obdélník přes polovinu obrazu není vhodné. Pro tento typ úloh je vhodnější zjistit přímo celou oblast, kde se daný objekt nachází a získat tak jeho binární masku. Výstupem takového algoritmu jsou binární masky všech pixelů pro všechny predikované třídy (viz obrázek 2.1). Myšlenka segmentačních neuronových sítích je rozvinutí konceptu klasifikačních a detekčních sítí. V oblasti segmentace existuje také úloha podrobnější – získat binární masky pro všechny třídy a zároveň rozlišit jednotlivé instance stejné třídy. Tento princip je znám jako segmentace instancí (*instance segmentation*), nicméně z pohledu samořídítelných aut není až tak klíčový, jelikož pro správné rozhodování stačí objekt nalézt.

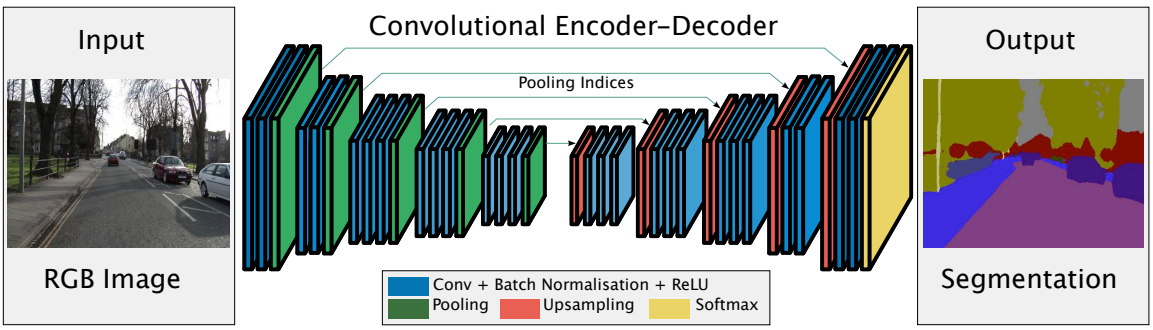
Sémantická segmentace čelí základnímu problému mezi obsahem a lokací: globální informace představuje *co*, zatímco lokální informace představuje *kde*. Tato sekce se zabývá základními architekturami segmentačních sítí, později také nejmodernějšími.

Mask R-CNN Jak bylo řečeno, segmentace instancí není klíčová pro orientaci samořídítelných aut, nicméně je potřeba zmínit neuronovou síť *Mask R-CNN*. Tato síť kombinuje výše zmíněnou úlohu detekce a úlohu segmentace. Je založená na základě sítě *R-CNN* (viz sekce 2.1) a k detekci ohraničujících obdélníků přidává také segmentaci pro každý detekovaný objekt tak, že k síti je přidána další větev určující binární masku, která pro dané pixely určuje, zda je součástí objektu. Jelikož ale výsledné segmentace objektů byly vždy nezarovnané o pár pixelů, místo vrstvy `RoIPooling` (viz sekce 2.1) byla zavedena vrstva `RoIAlign`, která používá místo poolingu, bilineární interpolaci. [12]

Kodér – Dekodér sítě

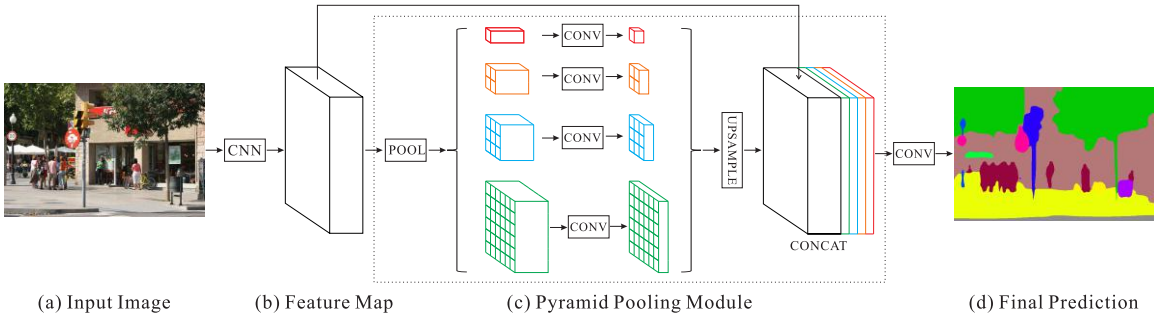
Základní myšlenkou neuronových sítí je, že vstupní obraz jistým způsobem zakódovávají, takže vstupní obraz je zmenšován, ale zvětšuje se počet kanálů. Na takových datech dobře funguje vyhledávání příznaků a jsou vhodné ke klasifikaci či detekci. Pro úlohy segmentací je ale potřeba provézt klasifikaci pro každý pixel vstupního obrazu a pak tuto informaci dostat do původní velikosti vstupního obrazu. Zakódovaný obrázek je tedy nutné znovu dekódovat a získat tak jeho původní velikost s příznaky získanými ve vnitřních částech sítě. První z neuronových sítí pro segmentaci fungují jako kodér – dekodér systémy. Kódovací část bývá často natrénovaná klasifikační síť (typu *VGG*, *ResNet*, apod.) a rozdíly jsou následně zejména v dekódovací části, případně ve způsobu propojení těchto dvou částí. Tato propojení, tzv. *skip connections*, umožňují lepší kombinování globálních a lokálních informací, co napomáhá přesnosti segmentace. [23]

Fully Convolutional Network Jednou z prvních segmentačních sítí je plně konvoluční síť (*Fully Convolutional Network – FCN*). Tato síť je postavena na natrénované klasifikační síti *VGG16*. Segmentace je získána za použití *skip* spojení z různých částí *VGG* sítě (z `conv4` a `conv3` bloků), které poskytují různou míru informace v různých částech sítě. Nadvzorkování (*upsampling*) do původní velikosti obrázku je zaručeno díky transponované konvoluční vrstvě (*transponed convolution*), která představuje trénovatelný bilineární interpolační filtr, do které jsou přičteny hodnoty ze zmíněných *skip* spojení. Jak již název napovídá, síť nepoužívá plně propojené vrstvy na konci kódovací částí. [16]



Obrázek 2.4: Architektura sítě *SegNet* (zdroj: [16])

SegNet Podobně jako *FCN* funguje také síť *SegNet*. Také je postavena na základní síti *VGG16* a autoři uvádějí, že byla tato síť pro extrakci příznaků použita předtrénovaná. Architekturu této sítě je možné vidět na obrázku 2.4. Síť se skládá z několika bloků, které jsou složeny z konvolučních vrstev, batch normalizace a vrstvy max-pooling. Každý takový



Obrázek 2.5: Architektura sítě *PSPNet* (zdroj: [27])

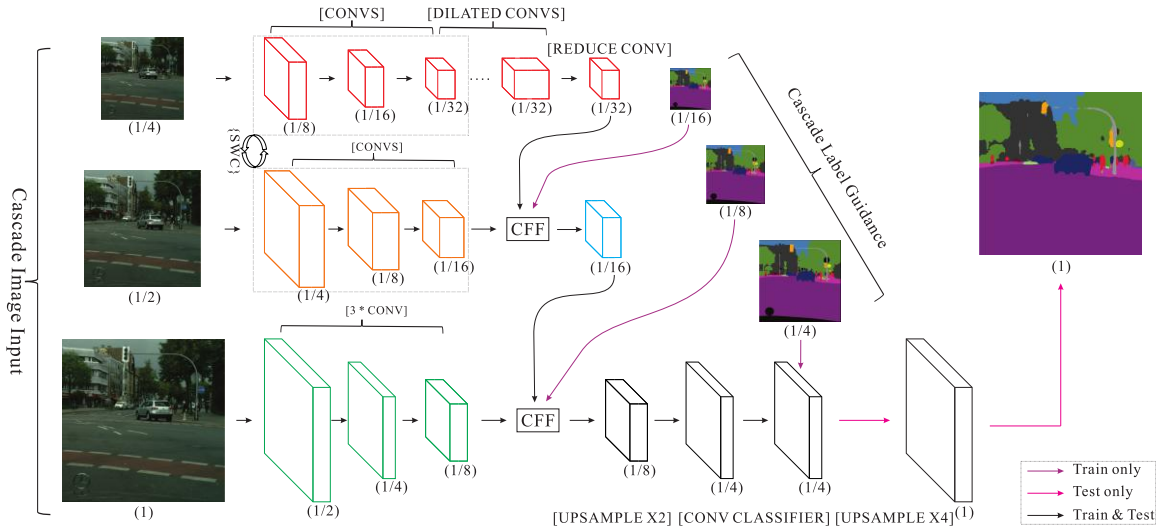
blok je následně připojen (*skip connection*) k obdobnému bloku v dekódovací části sítě. Autoři sítě uvádějí také jiný způsob propojení kódovací části s dekódovací a to tak, že v kódovacích pooling vrstvách jsou zapamatovány indexy, které byly vybrány pro max-pooling a ty jsou pak předány dekódovací části pro nadvzorkování. To je prováděno několika způsoby, mezi kterými je např. transponovaná konvoluce či bilineární interpolace. Tento způsob umožňuje šetřit paměťové nároky sítě.

V práci je také uvedena síť *SegNet-Basic*, která se skládá pouze ze čtyř zmiňovaných kódovacích, resp. dekódovacích bloků. V každém bloku je pouze jedna vrstva konvoluce následována batch normalizací, ReLU nelinearitou a max-pooling vrstvou. Tato menší síť je implementována bez *skip* spojení. [1]

Pyramid Pooling

Druhým způsobem, jak získat vhodné příznaky vstupního obrazu je *Pyramid Pooling*. Tato vrstva pracuje s aktivační mapu konvoluční vrstvy ve velké hloubce neuronové sítě (často až jedna z posledních vrstev kódovací části sítě), kde je provedeno paralelně několik operací s různou mírou informace (velikost konvolučních vrstev či pooling vrstev), které jsou posléze spojeny (pomocí operace konkatenace či sečtení). První zmínkou tohoto přístupu u neuronových sítí je v práci *Spatial Pyramid Pooling* autorů He a spol [13]. Ve zmiňované práci je však přístup použit pro klasifikační úlohu a je vložen do neuronové sítě těsně před plně propojené vrstvy, co autorům umožňovalo použití různých velikostí vstupních obrázků. Tento modul je však dále rozšířen, již pro účely segmentace, v pracích *PSPNet* či *ICNet*.

Pyramid Scene Parsing (PSPNet) Je jedna z nejmodernějších architektur neuronových sítí pro sémantickou segmentaci. Jako základní architekturu sítě pro získávání příznaků je použita modifikovaná verze sítě *ResNet* s použitím rozšířených konvolucí (*dilated convolutions*). Místo předem zmiňovaného přístupu kódování – dekódování rozšiřuje princip vrstvy *Spatial Pyramid Pooling*, kde na konci kódovací části sítě je přidán modul *Pyramid Pooling Module* (viz obrázek 2.5), který paralelně aplikuje vrstvy *global average pooling*, konvoluce, batch normalizace a aktivace ReLU, které jsou následně interpolovány na vstupní velikost modulu a konkatenovány. Na konci sítě je pak aplikováno několik vrstev, které mají za cíl zvětšit získané příznaky do původní velikosti vstupního obrazu s požadovaným počtem klasifikačních tříd. Pro lepší trénování autoři také uvádí pomocnou chybovou funkci *auxiliary loss*, která umožňuje rychlejší trénování sítě. [27]



Obrázek 2.6: Architektura sítě *ICNet* (zdroj: [26])

Image Cascade Network (ICNet) Tato síť vychází ze zmiňované sítě *PSPNet*. Rozšiřuje ideu této sítě ve prospěch časové i paměťové náročnosti. Na obrázku 2.6 je možné vidět základní architekturu sítě, která se skládá ze tří větví, kde každá zpracovává vstupní obraz v jiné velikosti (rozměrech). V původní velikosti vstupního obrazu je aplikováno menší množství konvolučních vrstev tak, aby byly příznaky zpracovány s dostatečnou přesností, ale výpočty nebyly příliš náročné. Druhá větev zmenšuje vstup na $1/2$ původní velikosti a aplikuje přesnější získávání příznaků pomocí více vrstev a výsledek je pak přičten k příznakům z první větve. Poslední větev vstup zmenšuje na $1/4$ a aplikuje podobnou síť jako *PSPNet*, na jejímž konci je obdobný modul *Pyramid Pooling* a výsledek je přičten k předchozí vrstvě. Místo operace konkatenace u tohoto modulu je použita operace sčítání. Architektura se také liší tím, že pro trénovací část není výsledná segmentace v původní velikosti vstupního obrazu, ale její rozměry jsou pouze $1/4$ velikosti vstupu. V takovém případě během testovací fáze je přidána navíc vrstva bilineární interpolace, která zvětší zmiňovanou výstupní segmentační masku do původní velikosti vstup. Podobně jako u sítě *PSPNet* i u této práce autoři uvádějí pomocné chybové funkce – pro každou zmiňovanou větev jednu. Výsledná chybová funkce je pak váženě sečtena dle rovnice

$$L = \lambda_1 L_1 + \lambda_2 L_2 + \lambda_3 L_3, \tag{2.1}$$

kde λ_1 představuje váhu první větve (v plném rozlišení) a λ_2 a λ_3 váhy následujících větví a L představuje *loss* funkci dané větve, kde L bez indexu je finální funkce. Hodnoty těchto váh jsou 1.0, 0.4 a 0.16. [26]

Obrázek 2.7: Vstupní obraz a rozdíl dvou barevných, po sobě jdoucích snímků videa z datové sady *Cityscapes*. Vstupní snímky jsou přeloženy přes sebe s poloviční průhledností.

2.3 Segmentace videa

V předchozí sekci byly popsány základní i jedny z nejmodernějších neuronových sítí pro sémantickou segmentaci obrazu. Vývoj tohoto typu sítí v poslední době dosahuje velmi dobrých výsledků, nicméně s větší přesností sítě roste také časová i paměťová náročnost. Poslední trend však ukazuje, že je možné dosáhnout optimalizovanějšího výkonu s menší penalizaci přesností (viz architektura *ICNet* v sekci 2.2). Většina těchto sítí však funguje na jednotlivých obrázcích a získává pouze segmentaci pro každý tento obrázek samostatně. Sítě nepracují s video sekvencemi a nevyužívají tak žádných temporálních informací, které se mezi snímky videa mohou vyskytovat. Známy jsou pouze tři práce využívající video pro sémantickou segmentaci: architektury upravené moduly *NetWarp*, síť *PEARL*[1] a framework pro práci s časem v neuronových sítích *Clockwork*. Architektury těchto sítí jsou popsané v sekci 2.3. V této sekci jsou však nejprve představeny, jaké jsou možnosti pro využívání informací z video sekvencí pro segmentaci obrazu.

Využití informací z videa

Pro využívání informací z video sekvencí je vhodné pracovat přímo s jejich jednotlivými snímky a aplikovat na tyto snímky nějaké algoritmy, které vytáhnou potřebné informace. Naivním řešením by bylo použít absolutní rozdíl mezi oběma snímky (ukázka na obrázku 2.7) a tuto informaci propagovat dále neuronovou sítí. Toto řešení ale není dostačující, jelikož získává informace pouze o tom, ve kterých oblastech obrazu se snímky změnily a umožňuje získání pseudo hran (spíše hranových oblastí). Změny ale neříkají, jak se objekty změnily či dokonce jakým směrem. Informace o posuvech a změnách objektů mezi po sobě jdoucími obrázky umožňuje lépe vystihnout optický tok.

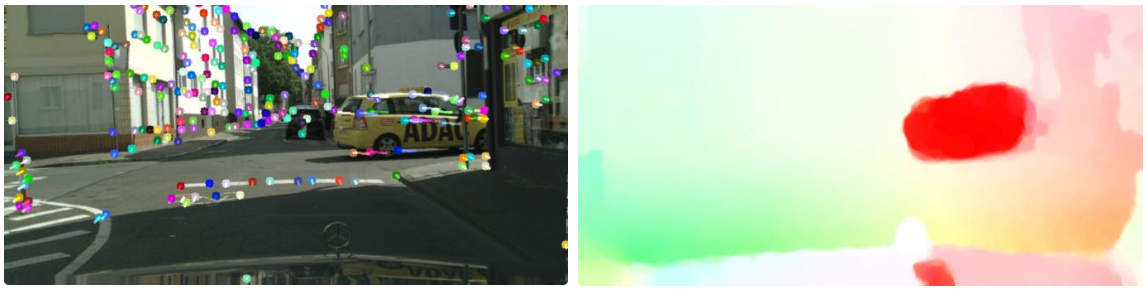
Optický tok Představuje informaci o pohybu objektů mezi dvěma po sobě jdoucími snímky obrazu. Vyobrazuje 2D matici vektorů, kde každý vektor je vektor posunutí bodu objektu z prvního snímku do druhého. Vektory jsou definovány jako úhel posunu (jakým směrem se bod mezi snímky posunul) a intenzita posunu. Základními předpoklady pro fungování optického toku jsou:

1. intenzity pixelů jednoho objektu se nezmění mezi jednotlivými snímky
2. sousedící pixely mají podobný pohyb

[1] Článek k této práci nebyl veřejně nalezen, proto síť není v práci dále zmiňovaná

Existují dva způsoby získávání optického toku. První způsob – řídký optický tok (*sparse optical flow*) je získáván pouze pro významné body vstupního obrazu (viz obrázek 2.8a). Tyto body je možné získat různými algoritmy, jako je například detekce rohů a je potřeba, aby byly určené před získáváním optického toku. Optický tok je následně počítán pouze v těchto bodech. Výstupem tohoto algoritmu je seznam vektorů optického toku pro významné body

Druhou možností je získání hloubkového optického toku (*dense optical flow)*, který umožňuje sledovat změny objektů ve všech pixelech vstupního obrazu (viz obrázek 2.8b). V tomto případě je výstupem dvou kanálová matice stejné velikosti jako vstupní obraz, kde kanály jsou x a y složky optického toku. [18]

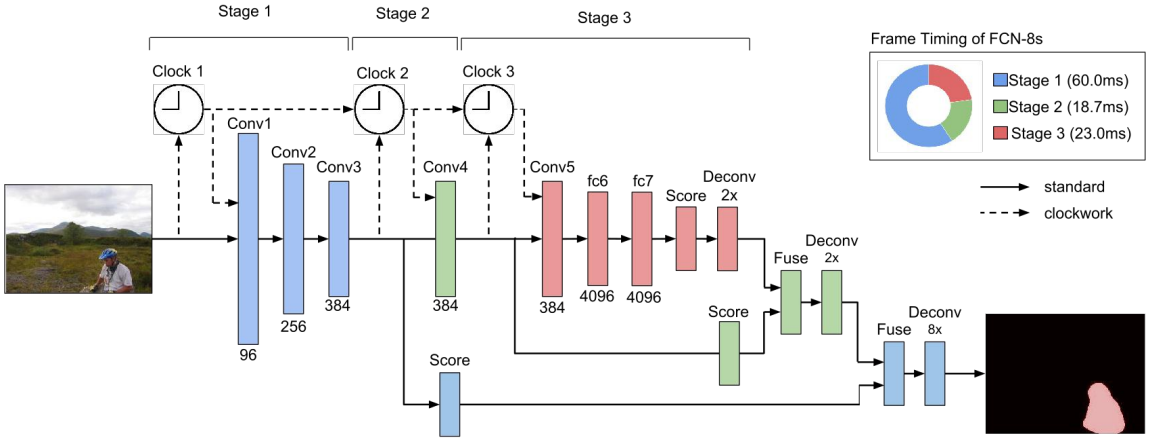


(a) Řídký optický tok (Lucas-Kanade) (b) Hloubkový optický tok (Gunner Farneback)

Obrázek 2.8: Ukázky základních typů optického toku na datové sadě *Cityscapes*. Je možné si všimnout, že hloubkový optický tok je v tomto případě získán i z odlesku na kapotě vozidla

Warpování obrázků Je druh nelineární transformace obrazu, díky čemuž je možné vstupní obraz upravovat pomocí libovolné mapovací funkce. Tuto transformaci je možné využívat k různým obrazovým efektům (např. rybí oko). Algoritmus se skládá ze dvou kroků: mapování a interpolace, kde mapování určuje, jakým způsobem se pixely vstupního obrazu transformují na výstupní obraz (je možné provádět dopředné nebo zpětné mapování) a interpolace umožňuje správné získávání barev pro výsledný obraz. Interpolaci je potřeba provádět z toho důvodu, že výsledky mapování obrazu nemusí být celočíselné a tudíž je potřeba rozhodnout, která barva vstupního obrazu bude použita (nejčastěji používané algoritmy jsou buď nejbližší soused, nebo bilineární interpolace). Warpování existuje ve dvou variantách – úsečkový warping (*feature based warping*) a síťový warping (*mesh warping*). První ze zmiňovaných používá úsečky pro deformaci vstupního obrazu do výstupního. Umožňuje vhodnou úpravu lokálních částí obrazu. Jelikož je ale warping v této práci použit pro úpravu obrazu na základě výstupu optického toku, tento druh warpování není potřeba a dále není zmiňován. U síťového warpingu je pro vstupní obraz přiřazena pomocná síť (matice), kde pomocí posuvu uzlových bodů dané sítě je možné transformovat vstupní obraz. Objekt v jednotlivých buňkách sítě není propagován do jiných buněk – pouze je transformován v rámci této buňky. [28]

Warpování je možné provádět na základě výsledků optického toku z předchozí sekce (2.3). V případě použití hloubkového optického toku je získána 2D matice posuvů bodů mezi po sobě jdoucími snímkami videa. Tato matice může sloužit jako zmiňovaná pomocná matice síťového warpingu a v takovém případě je možné snímek videa v čase $t-1$ zarovnat na snímek v čase t.



Obrázek 2.9: Framework *Clockwork* na architektuře *FCN* (zdroj: [24])

Existující metody video segmentace

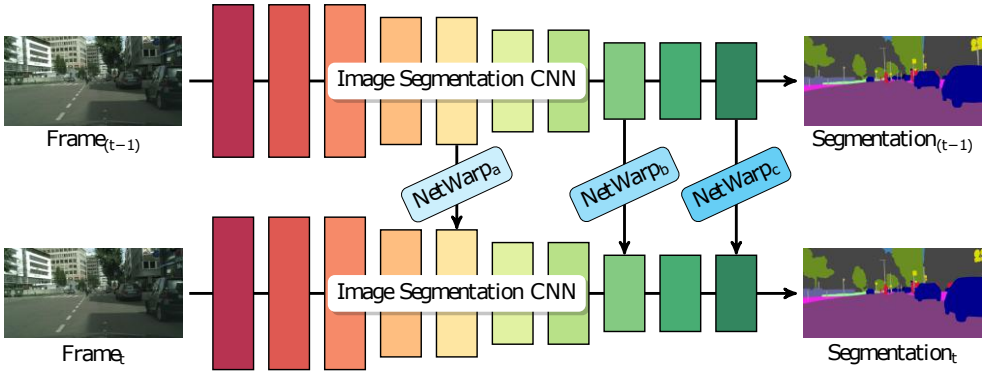
V této sekci jsou zmíněny nejnovější sítě pro práci s videem. Jelikož je toto odvětví segmentačních neuronových sítí teprve na vzestupu, pouze dva přístupy k video segmentací jsou popsány, a to *Clockwork* a *NetWarp*. Oba tyto přístupy pracují s existujícími architekturami pro segmentaci obrázků a pouze přidávají způsob, jak tyto sítě upravit pro použití na videích.

Clockwork

První prací, která využívá využívá závislostí mezi snímkami videa pro účely sémantické segmentace je *Clockwork*. Tato práce se nezabývá vytvořením architektury sítě, která by nějakým způsobem pracovala s videem, nýbrž autoři představují framework, který spouští inferenci neuronové sítě ve vhodné časové okamžiky. Zjištěním autorů je, že v různě hlubokých vrstvách segmentační sítě se příznaky mění odlišným tempem – v hlubších vrstvách jsou více stabilní a neměnné, naopak v mělkých vrstvách jsou změny nestálé a více lokální. Jako základní architektura pro segmentaci obrázků byla použita síť *FCN* (popsána v sekci 2.2), která slouží k segmentaci jednotlivých snímků videa a byla rozdělena na několik modulů (na obrázku 2.9 je znázorněna architektura, kde barevně jsou označeny jednotlivé moduly). Tyto moduly slouží ke vkládání předem zapamatovaných výstupů předchozí části sítě. V architektuře je definováno několik časovačů, které rozhodují, zda je potřeba, aby se segmentace přepočítaly v celé síti, či pouze v některém z jejího modulu. Přenášení informací mezi snímky videa funguje online způsobem, tzn. síť může využívat pouze aktuální a předchozí snímky. Byly představeny dva přístupy k aktivování jednotlivých modulů v síti – fixní, které používají konstantní počet snímku pro přepočítání příznaků v různých fázích sítě, a adaptivní – aktualizující se na základě změn ve vstupních datech. Pro adaptivní způsob se pro první snímek přepočítá celá síť, následně další snímek je vložen na začátek sítě, ale inference skončí na konci prvního modulu (vrstva `score_pool4`). Pokud se spočítané skóre této vrstvy liší od předchozího snímku o předem danou hranici, inference s tímto snímkem proběhne i pro další moduly sítě (a tím i hlubší vrstvy sítě). Tímto způsobem lze dosáhnout rychlejší segmentace na videích, jelikož není potřeba provádět inferenci na celé síti pro získaní segmentace aktuálního snímku. [24]

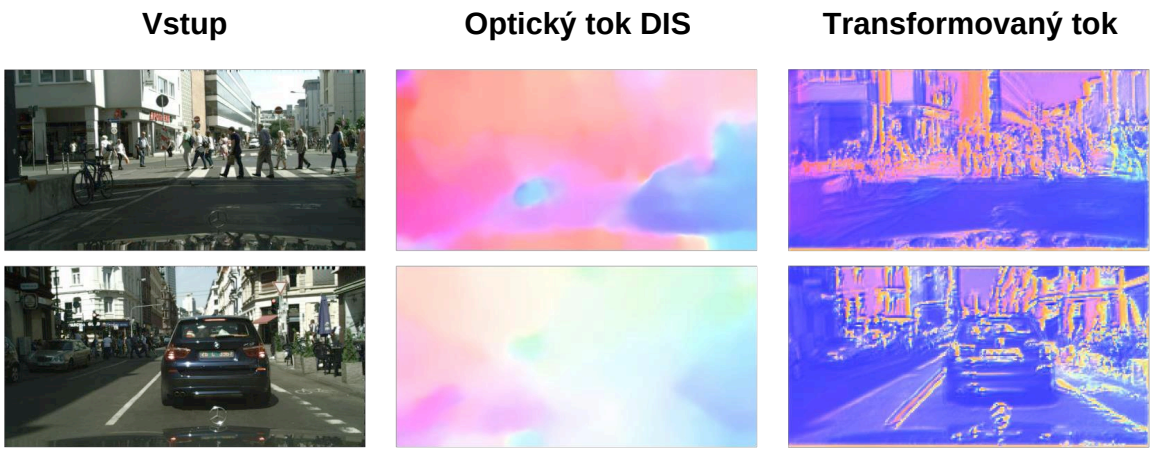
NetWarp

Neuronová síť *NetWarp* (viz obrázek 2.10) je jedna z nejmodernějších architektur pro sémantickou segmentaci, která upravuje obrázkové segmentační sítě v sítě pracující s videem. Motivací autorů jsou mimo jiné informace zjištěné na základě práce frameworku *Clockwork* (viz sekce 2.3). K získávání informací mezi jednotlivými snímkami videa je použit optický tok, pomocí něhož jsou transformovány příznakové mapy různých vrstev pro předchozí snímek videa (autoři poukazují na to, že síť by mohla být použita pro více než jeden předchozí snímek, nicméně kvůli paměťové náročnosti modelu s touto možností není pracováno). [7]

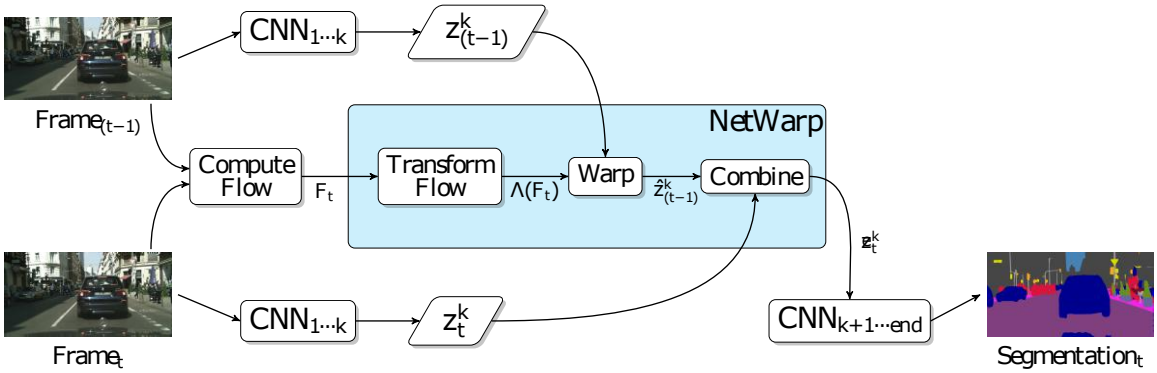


Obrázek 2.10: Architektura sítě *NetWarp* (zdroj: [7])

Transformace optického toku Zjištěním autorů práce je, že pouhý optický tok není dostačující ke zpřesnění segmentace, a proto byla vytvořena malá neuronová síť k transformaci optického toku. Vstupem této sítě jsou předchozí a aktuální snímek videa, optický tok mezi těmito snímky a rozdíly těchto dvou snímků, které jsou konkatenovány v 11 kanálový tensor. Tato síť se dále skládá ze tří konvolučních vrstev s počtem filtrů 16, 32 a 2 s velikostí jádra 3×3. Výstup poslední zmiňované vrstvy je pak konkatenován s původním optickým tokem a nakonec je přidána konvoluční vrstva s dvěma filtry (stejně jako optický tok) a 3×3 jádrem. Výstup této sítě je možné vidět na obrázku 2.11.



Obrázek 2.11: Transformace optického toku pomocí modulu *FlowCNN* (zdroj: [7])



Obrázek 2.12: *NetWarp* modul pro dva snímky videa (zdroj: [7])

NetWarp modul Hlavním bodem práce je vytvoření tzv. *NetWarp* modulu (viz obrázek 2.12), který umožňuje transformovat vybrané příznakové mapy neuronové sítě. Tento modul není závislý na typu neuronové sítě a je možné ho vložit do různých hloubek. Vstupem tohoto modulu jsou příznakové mapy předchozího a aktuálního snímku videa, a optický tok mezi odpovídajícími vstupními obrázky. Warpování je prováděno pomocí bilineární interpolace a warpuje se tak, aby předchozí příznaková mapa byla zarovnána na aktuální. Výstupem je pak warpovaná příznaková mapa předchozího vstupu, která je zkombinovaná s příznakovou mapou aktuálního vstupu. Spojení výstupů je na základě lineární kombinace dle rovnice

$$z_t^k = w_1 \circ z^k{}_t + w_2 \circ \hat{z}_{(t-1)}^k, \tag{2.2}$$

kde t a k představují časový index, resp. index vrstvy neuronové sítě, z je aktivační mapa dané vrstvy a $\hat{z}$ je warpovaná aktivační mapa. Parametry w_1 a w_2 jsou vektory se stejným počtem kanálů jako vrstva z^k a představují trénovatelné váhy. Operace $\circ$ představuje násobení jednotlivých kanálů matice z skalárem vektoru w. Výchozí nastavení parametrů w_1 a w_2 pro trénování sítě je 1.0 resp. 0.0, takže síť na začátku trénování sítě pracuje pouze s aktuálním snímkem videa.

Experimenty a výsledky Experimenty jsou prováděné na dvou známých datových sadách pro segmentací – *Cityscapes* a *CamVid* (viz sekce 3.1) s různými architekturami segmentačních sítí. Finální experimenty byly provedeny na datové sadě *Cityscapes* a, v době vzniku práce, nejlepší sítí *PSPNet* (popsána v sekci 2.2), na které práce s videem zlepšila mIoU o 0.7% a dosáhla *IoU* 81.5% mIoU. Díky těmto výsledkům je v současné době nejlepší sítí pracující s video sekvencemi na datové sadě *Cityscapes* (dle [4]). V práci je také ukázáno, že přidání tohoto modulu do existující neuronové sítě zvyšuje výpočetní náročnost minimálně (několik milisekund) a záleží na zvolení algoritmu optického toku.

2.4 Porovnání segmentačních metod

V této sekci jsou porovnány zmiňované segmentační neuronové sítě. Výsledky měření na těchto sítích a datové sadě *Cityscapes* (popsána v sekci 3.1) je možné vidět v tabulce 2.1. Ta obsahuje informace o přesnosti (*mIoU*) jednotlivých sítí, časové náročnosti inference a kdy byla publikována. Je možné vidět, že postupem času se sítě buďto razantně zlepšují v přesnosti, nebo jsou podstatně rychlejší než předchůdci za cenu menší přesnosti. Nejlepší sítí, ze zde popsaných, je síť *NetWarp*, která je postavena na základě architektury *PSPNet*. Pro oba případy těchto sítí autoři uvádějí výsledky *multi scale* varianty (zkratka *MSc* v tabulce), která predikuje na vstupním obrazu v různých velikostech a výsledek je poté sloučen. Časová náročnost těchto dvou sítí není nejlepší, co odpovídá zmiňovanému kompromisu mezi výkonem a rychlostí, nicméně síť *NetWarp* se od *PSPNet* liší minimálně a z hlediska časové náročnosti záleží více na výběru algoritmu optického toku než na síti samotné. Co se týče frameworku *Clockwork* s použitím sítě *FCN*, autoři neuvádějí přímo čas potřebný k inferenci, nicméně dané IoU je naměřené pro inferenci co druhého snímku videa, takže by rychlost mohla dosahovat ~ 0.25 sekund (na základě rychlosti klasické *FCN* architektury). Z druhé strany porovnávání je nejrychlejší sítí architektura *ICNet*, která umožňuje pracovat v reálném čase se snímkami videa (cca 33 snímků za sekundu) i přesto, že IoU této sítě dosahuje velmi dobrých výsledků.

Metoda	mIoU [%]	Časová náročnost [s]	Publikováno
PSPNet (MSc) + NetWarp	**81.50**	0.51	**10.08.2017**
PSPNet (MSc)	80.80	0.50	04.12.2016
ICNet	67.70	**0.03**	27.04.2017
FCN 8s	65.30	0.50	20.05.2016
Clockwork FCN	64.40	*50% snímků*	11.08.2016
SegNet	57.00	0.06	02.11.2015

Tabulka 2.1: Porovnání segmentačních sítí na datové sadě *Cityscapes*. U sítě *Clockwork FCN* je uvedeno 50% snímků, co znamená, že síť při dané přesnosti používá co druhý snímek pro plnou segmentaci. (zdroje: [4], [26], [27], [7])

Kapitola 3

Datové sady

Tato kapitola se věnuje veřejně dostupným datovým sadám pro účely segmentace. Nejprve je obecně vysvětleno, jaké informace pro segmentační úlohy jsou potřeba a v jakém formátu se často vyskytují, dále v sekci 3.1 jsou popsány existující datové sady a nakonec v sekci 3.2 je porovnání těchto sad, výběr hlavní sady a příprava pro tuto práci.

Segmentační datové sady se musí skládat z dvojic (I, M), kde I představuje vstup ke trénování/validaci a M představuje anotovanou masku daného obrázku. Vstupní obrázky se v různých datových sadách liší jak rozměry, tak kvalitou – v závislosti na použité technice získávání těchto dat (často jsou data anotovaná buď ručně, nebo semi-automatizovaně). Rozdílná je také hustota anotací, kdy existují řídké anotace, u kterých nejsou pokryty všechny pixely obrázků, a husté anotace, kde pro každý pixel vstupního obrázku je určena třída, do které daný pixel patří. Anotační maska je také v různých datových sadách přikládaná jiným způsobem. Nejčastěji se vyskytují tyto reprezentace anotačních masek:

- Tříkanálový obrázek (RGB) stejné velikosti jako vstupní obraz I obsahující barevné regiony, kde každá barva představuje jednu klasifikační třídu. Každá klasifikační třída pak musí mít specifikovanou barvu, kterou je reprezentována v anotované masce a tyto barvy by měly být unikátní, jinak může docházet k přiřazení dvou tříd k jednomu pixelu vstupu.
- Jednokanálový obrázek stejné velikosti jako I, kde na daných regionech jsou přímo zapsány identifikátory známých klasifikačních tříd (často jako index do seznamu klasifikačních tříd).
- Textový soubor ($JSON$), který obsahuje souřadnice regionů jednotlivých klasifikačních tříd. V tomto případě je potřeba tento soubor zpracovat do podobného formátu jako v předchozích bodech, případně nějakým způsobem načítat tyto polygony během trénování.

Pro trénování neuronových sítí platí, že čím více dat, tím lepších výsledků je možné dosáhnout. Existující největší datové sady (např. datová sada $COCO$[1]) jsou však obecného charakteru a nejsou zaměřené na objekty pouliční scény. Pro specifické účely, jako je tato práce, je potřeba zvolit z menších dostupných datových sad. Tyto datové sady by měly obsahovat typické objekty, které se nachází v pouliční scéně a pohled na tuto scénu by měl být z palubní desky vozidla či oblasti podobné pohledu řidiče. Výčet takovýchto tříd je možné vidět v tabulce 3.1. Datové sady tyto objekty často řadí jak do jednotlivých tříd,

[1] Common Object in Context – dostupné na http://cocodataset.org/

tak do kategorií, do kterých spadají, co může v některých typech úloh být vhodné, jelikož nemusí záležet na tom, zda algoritmus detekuje auto či dodávku, ale je důležité vědět, že je to vozidlo. Pro účely této práce je také potřeba zmínit, že datové sady je potřeba rozlišit na základě toho, zda obsahují náhodné situace, či video sekvence. V prvním ze zmiňovaných – obrázkové datové sady – jsou data rozmístěna do různých situací, různých měst, časových úseků apod. V druhém typu datových sad – video sekvence – jsou data pořízena pomocí videa a všechny snímky (nebo alespoň část z nich) jsou dostupné a seřazeny chronologicky.

Kategorie	plocha	člověk	vozidlo	konstrukce	pouliční objekt	příroda	ostatní
Třída	cesta chodník parking	chodec cyklista motocyklista	auto motorka dodávka	budova plot zeď	semafor sloup značení	terén vegetace sníh	obloha voda

Tabulka 3.1: Přehled typických klasifikačních tříd pro segmentační datové sad (zdroj: [5])

3.1 Dostupné datové sady

Pro algoritmy hlubokého učení je potřeba zvolit vhodnou datovou sadu (případně více), na které bude daná problematika natrénována. Tato sekce popisuje existující datové sady, které obsahují anotace určené pro sémantickou segmentaci. Popsáno je prostředí, ve kterém se datové sady pohybují, velikost datové sady i to, zda obsahují video sekvence a s jakou frekvencí jsou jednotlivé snímky sekvencí.

Obrázek 3.1: Vstupní a anotovaný obrázek z datové sady *CamVid*. Datová sada obsahuje anotované značení cest (zdroje: [2][3])

Cambridge Labeled Objects in Video (CamVid) Je jedna z prvních datových sad určená pro sémantickou segmentaci pouliční scény pořízena ve městě Cambridge. Vstupní obrázky jsou z kamery umístěné na palubní desce auta (příklad viz obrázek 3.1). Obsahuje 701 obrázků městských scén spolu s anotacemi pro 32 klasifikačních tříd. Vstupní data tvoří sekvence videa, nicméně ty jsou pouze čtyři a obsahují 101, 305, 171, resp. 124 navazujících snímků. Datová sada neobsahuje žádné informace, jak ji rozdělit na trénovací, validační a testovací část. Rozlišení jednotlivých snímků je 960×720 pixelů. Výhodou této datové sady je, že obsahuje anotované značení na vozovce (jízdní pruhy, značky, apod.), takže trénované

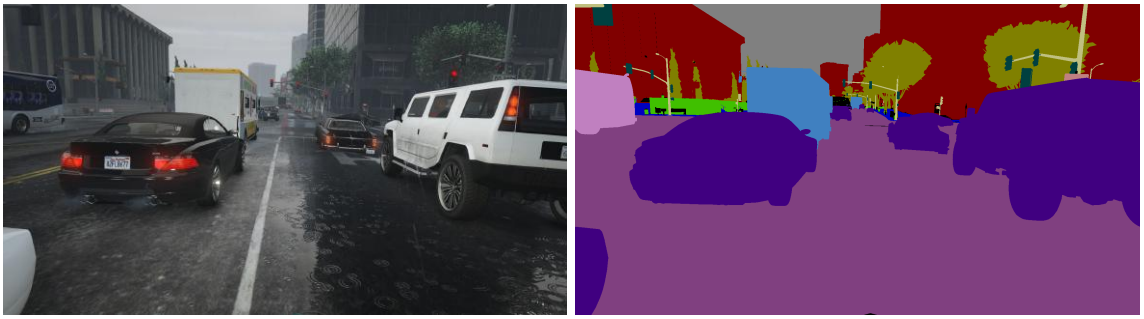
algoritmy mohou pracovat jak s orientací ve scéně, tak s orientací na vozovce. Nevýhodou však je, že obsahuje velmi malé množství anotovaných snímků a situace v jednotlivých snímkách jsou od sebe vzdáleny několik metrů a tak nelze dobře analyzovat prostorové závislosti mezi nimi (není to nemožné, nicméně warpování obrázků pro některé snímky bývá nesmyslné). [2][3]

Mapillary Vistas Jednou z největších datových sad pro úlohy segmentace pouliční scény je datová sada *Mapillary Vistas*. Obsahuje 25 000 hustě anotovaných záběrů, konkrétně 18 000 trénovacích, 2 000 validačních a 5 000 testovacích obrázků. Je možné predikovat 100 různých klasifikačních tříd, které se řadí do kategorií podobných těm v tabulce 3.1. Data obsahují obrázky z různých částí světa (napříč kontinenty) za různorodých povětrnostních podmínek (sníh, déšť, mlha, ...), příklad viz obrázek 3.2, a jsou pořízeny kamerami s odlišnými parametry (ohnisková vzdálenost, poměr stran, atd.), což napomáhá generalizování neuronových sítí, které využívají tuto datovou sadu. Rozlišení jednotlivých obrázků je ve `4K` ($3\,264 \times 2\,448$), což umožňuje detailní informace o objektech (nicméně v takovémto rozlišení je možné trénovat pravděpodobně pouze na nejlepších grafických kartách současnosti). Tuto datovou sadu je však možno použít pouze pro algoritmy pracující na jednotlivých obrázcích a ne na video sekvencích, jelikož vstupní data neobsahují žádné předchozí informace (nejsou ani žádným způsobem seřazeny). Výhodou naopak je, že obsahuje anotace pro značení na vozovce a situace různého počasí (např. sníh). [17]

Obrázek 3.2: Vstupní obrázek a anotační maska z datové sady *Mapillary Vistas*. Tato datová sada se odlišuje především různými povětrnostními podmínkami. Na obrázku je možné vidět zasněženou krajinu a odpovídající klasifikační třídy sněhu. (zdroj: [17])

Playing for Data – GTA 5 Je syntetická datová sada obrázků vyextrahovaných ze hry *Grand Theft Auto V*. Obsahuje 24 966 obrázků scén ze hry spolu se stejným počtem anotací (příklad vstupu viz obrázek 3.3). Anotace byly prováděné semi-automaticky a jsou dostupné pro celý obrázek (hustě anotované). Datová sada není nikterak rozdělena na trénovací, validační a testovací část, nicméně je přiložena matice `split.mat`, která obsahuje rozdělení na dané části. Tato datová sada umožňuje predikci 27 tříd, které jsou kompatibilní s ostatními zmiňovanými datovými sadami. Vstupní obrázky i anotace jsou v rozlišení $1\,914 \times 1\,052$ získané v různých denních obdobích, za různého počasí. Sada poskytuje pouze anotované obrázky, takže je potřeba načíst data na základě barev jednotlivých tříd. Jednotlivé obrázky

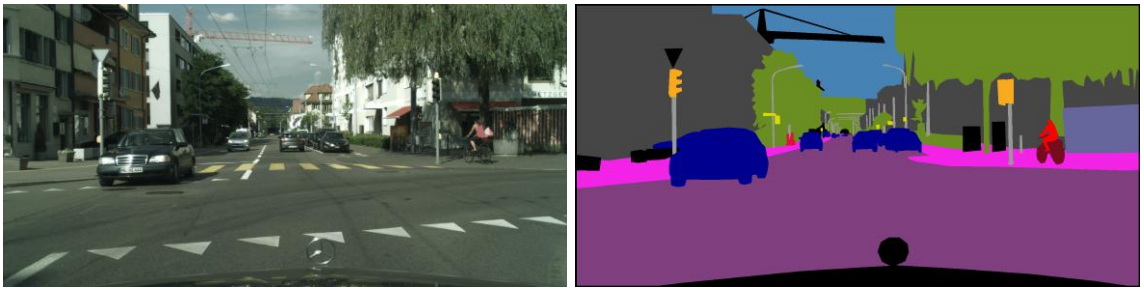
datové sady na sebe navazují a tvoří tak sekvence videa. Problémem využití těchto sekvencí však je, že jednotlivé snímky jsou od sebe vzdáleny několik sekund a tudíž nelze pořádně využít závislosti mezi snímky, jelikož často objekt, který je na jednom snímku, na druhém již buď není, nebo je příliš vzdálen. [21]



Obrázek 3.3: Vstupní obrázek a anotační maska z datové sady *Playing for Data – GTA 5*. Je možné vidět, že i přes to, že datová sada obsahuje syntetické obrázky, kvalita a různorodost je veliká, co podporuje použití v reálném prostředí (zdroj: [21])

Cityscapes Poslední z představených datových sad je *Cityscapes*. Tato sada je velmi populární v oboru hlubokého učení, zvláště pro úlohy sémantické segmentace. Vstupní obrázky (příklad viz 3.4) jsou pořízeny v ulicích Německa (např. Frankfurt, Berlín, Mnichov, atd.) během jara až podzimu, takže datová sada neobsahuje žádné snímky, na kterých by se např. nacházel sníh. Počasí, během kterého byly pořizovány snímky do datové sady také není příliš rozličné, jelikož obsahuje pouze situace s mírným počasím (maximálně zamračeno, bez deště). I přes tyto nedostatky se stala datová sada celosvětovým standardem. V této sadě je 5 000 hustě anotovaných obrázků, které jsou rozděleny na 2 975 trénovacích, 500 validačních a 1 525 testovacích.[2]. K dispozici je také 20 000 řídce anotovaných obrázku. Vstupní rozlišení obrázků datové sady je $2\,048 \times 1\,024$ pixelů. Obrázky pocházejí se sekvencí videa a jednotlivé anotace jsou vždy 20. snímkem při vzorkovací frekvenci 30 snímků za sekundu, což odpovídá $1,8$ sekundovým intervalům. Zbylé pořízené snímky jsou dostupné na požádání autorů práce a je tak možné využívat video k trénování a predikci segmentace. Datová sada také obsahuje jiné informace získané během pořizování dat, jakými jsou například `GPS` koordináty, hloubková data ze stereo kamery, venkovní teplota a jiné. Velkou výhodou této datové sady (a pravděpodobně důvod, proč je sada tolik populární) je fakt, že autoři poskytují veřejný benchmark, ve kterém je možné porovnávat přesnost různých sítí. Nevýhodou však je, že nejsou anotovány jízdní pruhy a značení na vozovce. Další informace o této datové sadě jsou popsány v sekci 3.2. [5]

[2] Anotace pro testovací obrázky nejsou dostupné a slouží pro účely benchmarku.



Obrázek 3.4: Vstupní obrázek a anotační maska datové sady *Cityscapes* (zdroj: [5])

	Video sekvence	Vzdálenost mezi snímky	Množství obrázků	Značení vozovky	Reálná data	Rozlišení obrázků
Mapillary Vistas	✗	✗	25000	✓	✓	$3\,264 \times 2\,448$
Playing for Data	✓	$\sim 2s$	24966	✗	✗	$1914 \times 1\,052$
CamVid	✓	$\sim 0.5s$	701	✓	✓	960×720
Cityscapes	✓	$\sim 0.03s$	5000	✗	✓	$2\,048 \times 1\,024$

Tabulka 3.2: Porovnání kritérií datových sad pro účely segmentace na videu (zdroje viz 3.1)

3.2 Výběr a příprava datové sady

Pro účely této práce, a obecně neuronových sítí založených na segmentaci videa, je potřeba, aby datová sada splňovala několik základních kritérií, dle kterých je pak rozhodováno, jakou datovou sadu zvolit:

1. Obsahuje video sekvence (jednotlivé obrázky na sebe musí navazovat)
2. Vzdálenost mezi jednotlivými snímky
3. Množství obrázků pro trénování/validaci
4. Obsahuje značení vozovky
5. Data nejsou syntetická (pochází z reálných obrázků)

Tyto informace jsou popsány u jednotlivých datových sad v sekci 3.1. Souhrn a porovnání je také v tabulce 3.2, kde je možné vidět, že pokud by cílem práce nebylo využívat informace propagované mezi snímkami videa, byla by datová sada *Mapillary Vistas* nejlepší volbou, jelikož obsahuje nejvíce dat s nejlepším rozlišením, data jsou velmi různorodá a obsahuje dokonce i značení vozovky. Druhou možností, co se týče množství dat je použití syntetické sady *Playing for Data*, která obsahuje sekvence snímků, nicméně tyto snímky jsou od sebe příliš daleko a tudíž nejsou vhodné. V případě použití této datové sady, je možné, že by bylo potřeba pro reálné situace dotrénovat síť na jiné datové sadě, která obsahuje obrázky z reality. Použitelná vzdálenost mezi snímky již je v datové sadě *CamVid*, nicméně tato sada obsahuje velmi malé množství obrázků a vzdálenosti mezi snímky někdy působí problémy při počítání optického toku a následném warpování obrázků. Poslední možností ze zmiňovaných datových sad je *Cityscapes*. Tato datová sada se jeví jako vhodná volba, jelikož splňuje všechna výše zmiňovaná kritéria, kromě toho, že anotace neobsahují značení vozovky. Nevýhodou je, že obsahuje pouze snímky s dobrým počasím, což v praxi

znesnadňuje situaci v horším počasí. Tato datová sada je tedy použita ve experimentech a v následujících sekcích je popsána příprava této sady k další práci.

Struktura datové sady Pro získání datové sady je potřeba se přihlásit na oficiálních stránkách, kde je možné následně stáhnout zmiňované anotační masky a vstupní obrázky. Data pro všechny snímky video sekvencí nejsou automaticky dostupné a je potřeba zažádat autory datové sady o jejich souhlas k přístupu. Po odsouhlasení je soubor `leftImg8bit_sequence_trainvaltest.zip` k dispozici ke stažení. Struktura datové sady se liší od jiných tím, že je silně rozdělená, dle toho, kde byly data (video nahrávky) pořízeny, ze které sekvence pochází apod. Strukturu složek je možné vidět níže:

```
{root}/{type}{video}/{split}/{city}/{city}_{seq:0>6}_{frame:0>6}_{type}{ext}
```

Toto rozdělení je složitější na manipulaci, jelikož je potřeba procházet specifické podsložky a spojovat tak data, jelikož často nejsou informace o místech pořízení potřebná (alespoň ne v případě této práce).

Zmenšení na disku Datová sada s kompletními video sekvencemi má před dekomprimací 320 `GB`. Jelikož ale na trénovacím počítači není dostatek místa, bylo potřeba připravit menší verzi sady, a to konkrétně vzít několik snímků před zmiňovaným 20. snímkem, pro který existuje anotační maska. Byl tedy vytvořen skript, který prochází všechny složky datové sady a překopíruje zadané množství předchozích snímků do cílové složky (pro účely této práce byly vybrány vždy tři předchozí snímky). Většina dat obsahuje několik krátkých sekvencí, kde jednoduše jde z názvu zjistit 20. snímek a vzít několik předchozích. Bohužel v některých městech však bylo natočeno méně, podstatně delších, sekvencí a tento algoritmus nefungoval správně. Finální algoritmus tedy nalézal anotované obrázky a k nim se podle jména souboru snažil najít předchozí vstupní snímky. Takto upravená datová sada byla zmenšena na 43.4 `GB`, co následně mohlo být použito na všech použitých počítačích (použitý hardware viz sekce 5.1).

Kapitola 4

Návrh řešení

V předchozích kapitolách bylo popsáno, jaké možnosti existují v oblasti strojového učení pro klasifikaci objektů v obraze. V kapitole 2 jsou popsány algoritmy jak pro detekci objektů (a tím získání ohraničujících obdélníků pro dané objekty), tak algoritmy segmentace celého vstupního obrazu. Oba tyto způsoby se řadí do problematiky klasifikace objektů v obraze a je možné problém řešit oběma způsoby. Pro koncové aplikace využívající klasifikaci obrazu (např. pro samořiditelná auta) by bylo dokonce možné sloučit obě zmíněné metody. Důvodem tohoto sloučení je fakt, že detekce vyniká v odhadování objektů specifických rozměrů (typu vozidlo, člověk, apod.), kdežto segmentace umožňuje získávat spojité části obrazu, pro které by ohraničující obdélník byl nesmyslný (např. cesta, pruhy, vegetace, atd.). Druhým důvodem pro studium obou těchto možností je ten, že se vývoj hlubokých neuronových sítí vyvíjel obdobným způsobem – od klasifikace, přes detekci, segmentaci až k práci s videem. Tato práce se však dále zaměřuje pouze na aplikaci sémantické segmentace na videu a jejím cílem je experimentovat se segmentačními neuronovými sítěmi.

V této kapitole jsou popsány způsoby, jak bylo k problematice přistupováno. V sekci 4.1 jsou popsány způsoby získávání informací mezi snímkami videa a následná aplikace těchto informací v hlubokých neuronových sítích. Dále sekce 4.2 popisuje s jakými modely neuronových sítí je pracováno a jakým způsobem jsou upraveny pro využití na videu. Nakonec pak sekce 4.3 zmiňuje, jakým způsobem jsou generována data pro trénování těchto neuronových sítí.

4.1 Využití závislostí ve videu

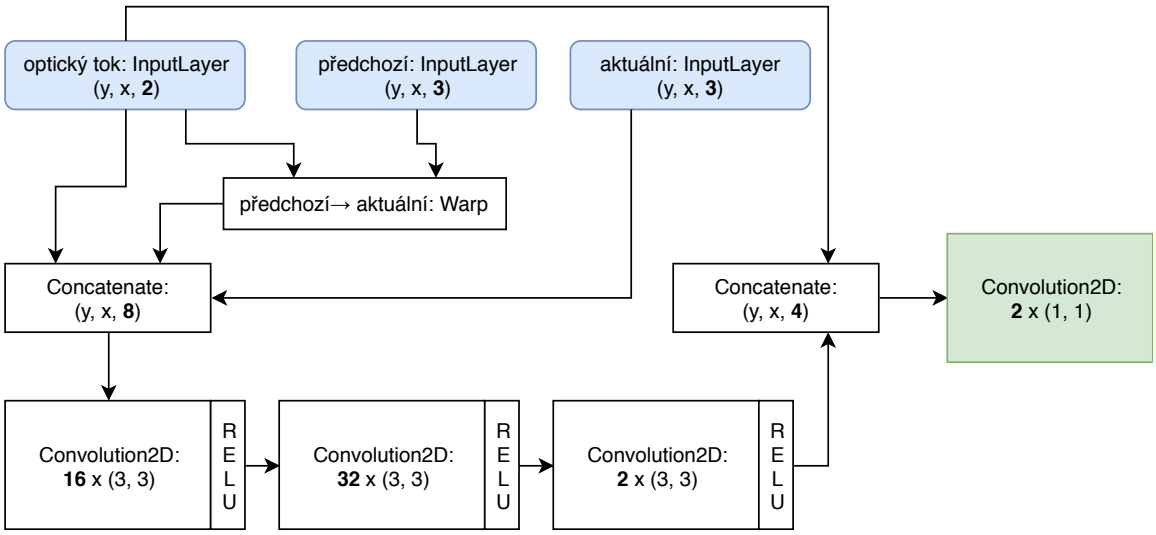
Pro získávání informací mezi jednotlivými snímkami videa je použit algoritmus hloubkového optického toku (viz sekce 2.3). Informace získané z optického toku, tj. jakým směrem se objekty mezi snímkami posunuly, je vhodné dále upravovat tak, aby co nejvíce odpovídaly aktuálnímu snímku videa. Tento optický tok je počítán mezi snímkami videa mimo samotnou neuronovou síť a je do ní vkládán spolu s předchozím a aktuálním snímkem videa. Počítán je reverzní optický tok, tzn. posuv oblastí z aktuálního snímku na předchozí. Optický tok je počítán online způsobem, tzn. pro první snímek videa je tok nedostupný a je možné ho použít až od 2. snímku videa. Pro trénování neuronové sítě je zpracování optického toku součástí generátoru dat a je určován snímky, které následně plní trénovací frontu (viz sekce 4.3). V této práci byl použit základní algoritmus z knihovny *OpenCV* – hloubkových optický tok Gunner Farneback (na základě práce [18]). Bylo také experimentováno s jinými algoritmy optických toků (mj. *DIS flow*, který byl také použit v práci *NetWarp* popsané

v sekci 2.3), nicméně v pozdější fázi experimentů již jiné algoritmy nebyly aplikovány, jelikož většina experimentů probíhala na výpočetním centru *Metacentrum* (viz sekce 5.1), kde se jiné algoritmy optických toků nepodařilo zprovoznit. *Metacentrum* umožňuje sice zkompilovat vlastní programy, nicméně po dlouhém zkoušení se toto nepodařilo. Jiné druhy optických toků jsou nově také součástí rozšíření knihovny *OpenCV – Contrib*, nicméně to se také nepovedlo na *Metacentru* zprovoznit kvůli konfliktům s originální verzí knihovny a nemožnosti „lokální“ odinstalace.

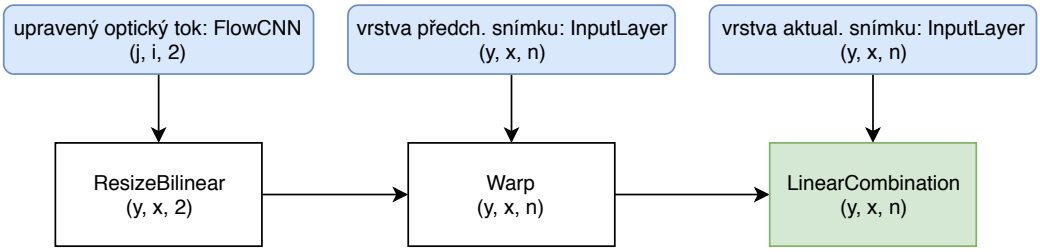
Pro další úpravu snímků a zarovnání předchozích snímků na aktuální je použit druh transformace obrazu – mesh warping. Tato transformace se využívá ve vybraných vrstvách neuronové sítě a její výsledek je dále zpracováván neuronovou sítí.

Upravení FlowCNN V sekci 2.3 byla popsána malá neuronová síť, která upravuje optický tok a přidává více informací o objektech v obrazu, díky propojení optického toku s původním obrazem. Jelikož zmiňovaná síť nedávala stejné vizuální výsledky, jaké autoři zmiňovali ve své práci, byla tato síť upravena tak, aby se výsledky více blížily a dávaly tak větší smysl pro warping. Náčrt upraveného modulu je možné vidět na obrázku 4.1, kde jsou jednotlivé vrstvy představeny obdélníky spolu z názvem a rozměry výstupů vrstvy (případně počtem filtrů a jádrem u konvolučních vrstev). Cílem je umožnit warpování různých aktivačních map předchozího snímku videa na aktuální. Na základě tohoto faktu byl místo rozdílů vstupních snímků použit warpovaný předchozí snímek videa s použitím běžného optického toku. Dále bylo upraveno konkatenování vstupů (na obrázku označeny modrou barvou) tak, že tvoří tensor o 8 kanálech místo zmiňovaných 11 kanálů v dané práci. Tato skutečnost dává větší smysl také proto, že se často u konvolučních sítí počet kanálů zdvojuje v každé další vrstvě, což odpovídá 16 kanálům v konvoluční vrstvě po dané konkatenaci. Další konvoluční vrstvy jsou aplikovány stejným způsobem jako ve zmiňované práci, tj. 32, resp. dva filtry a výstup je konkatenován s původním optickým tokem. Poslední změna je také ve výstupní konvoluční vrstvě modulu (označeno zelenou barvou), kde je místo původního 3×3 jádra konvoluce, použito jádro 1×1. Tato změna také dává větší smysl, jelikož v poslední konkatenované vrstvě již jsou všechny informace o upraveném optickém toku zahrnuty a je přidán původní optický tok. Konvoluce 1×1 tedy v podstatě slouží pouze pro redukci dimenze výstupní vrstvy a tím simulování dvoukanálového optického toku.

Warpování aktivačních map Klíčovým algoritmem, jak využívat informace mezi snímkami videa je transformace obrazu – warping. Tuto transformaci by bylo možné používat několika způsoby a teoreticky dosáhnout různého zlepšení v segmentaci. Naivním a zároveň nejjednodušším způsobem transformace může být získání segmentační masky pro aktuální a předchozí snímek videa a segmentační masku předchozího snímku warpovat a přičíst k aktuální masce. Je možné, že tento způsob by dosahoval dobrých výsledků, nicméně nevyužívá žádné informace uvnitř neuronové sítě, nýbrž pouze její výsledek. Pro optimálnější využití je vhodné pracovat s warpováním uvnitř neuronové sítě a tedy pracovat s aktivačními mapami různých vrstev. Tuto operaci využívají také autoři v práci 2.3 a je tedy i předmětem této práce. Warpování aktivačních map vyžaduje několik přípravných operací jež jsou zapouzdřeny v modulu *NetWarp*, který je možné vložit do neuronové sítě na různá místa. Na obrázku 4.2 je představena struktura tohoto modulu. Modrou barvou jsou zvýrazněny vstupy modulu, jimiž jsou upravený optický tok z modulu *FlowCNN* (popsaný v sekci 4.1), vybraná vrstva neuronové sítě pro předchozí snímek videa a obdobná vrstva pro aktuální snímek. Umístění *NetWarp* modulu v neuronové síti není zcela předvídatelné, je možné kombinovat více modulů v jedné sítí. Tato poloha vytváří další hyperparametr



Obrázek 4.1: Upravený modul *FlowCNN*



Obrázek 4.2: Upravený NetWarp modul

pro trénování dané neuronové sítě. Podobně jako u transformace optického toku je také v tomto modulu rozdíl oproti originální práci. Jelikož je možné modul vkládat do různých hloubek neuronové sítě, je potřeba změnit rozměry upraveného optického toku tak, aby byly stejné jako velikosti aktivačních map jednotlivých vrstev. Pro tuto operaci autoři originální práce používají *pooling* vrstvu, která jej zmenší na požadovanou velikost, nicméně v případě této práce bylo místo *poolingu* použita vrstva `ResizeBilinear`, jež využívá funkci `tf.image.resize_bilinear` knihovny *Tensorflow* a zmenšuje tak rozměry pomocí bilineární interpolace.

Z důvodu, že operace warpování je součástí neuronové sítě, je nutné, aby tato operace byla diferencovatelná. Pro tento případ byl použit algoritmus inspirovaný částí sítě *Spatial Transformer Network*, konkrétně vytváření parametrizovatelné sítě a diferencovatelného samplování pomocí této sítě [14]. V práci ukazují, že je možné samplovat buď pomocí algoritmu nejbližšího souseda nebo bilineární interpolace. V této práci tedy byla použita bilineární interpolace pro převzorkování pixelů vstupní aktivační mapy na výstupní.

Výše zmiňované operace modulu *NetWarp* zaručují získání transformované aktivační mapy, která je zarovnána na aktivační mapu pro aktuální snímek videa. Posledním krokem, který je potřeba aplikovat je sjednocení těchto informací s aktivační mapou aktuálního kroku. V tomto případě je možné použít operaci sčítání matic a dosáhnout tak požadovaného výsledku, nebo sčítání provézt podrobněji a tedy lineárně zkombinovat tyto dvě vrstvy. Pro účely této práce byla vytvořena vrstva `LinearCombination` (na obrázku 4.2 zvýraz-

něno zelenou barvou), která sčítá jednotlivé kanály spojovaných vrstev parametricky, kde parametry jsou trénovatelné váhy (způsob spojování ukázán na rovnici (2.2)). Rozdíl oproti původní práci je v nastavení výchozích váh, a to v obou případe na 1.0, jelikož bylo v průběhu implementace zjištěno, že se dané váhy příliš nemění a nejlepší výsledky dává vrstva při nastavení obou snímků videa.

4.2 Architektury sítí

V předchozích sekcích zmíněné způsoby aplikace video informací je potřeba aplikovat do nějaké segmentační neuronové sítě. Cílem této práce není vymyslet novou architekturu neuronové sítě, nýbrž aplikovat zpracování videa do sítí pracujících s jednotlivými obrázky. Vhodné segmentační sítě jsou představeny v sekci 2.2, kde jsou popsány jak základní, tak nejmodernější sítě pro tento typ úlohy.

Prvotní pokusy se segmentačními sítěmi byly prováděny na síti *FCN* (viz sekce 2.2). Díky použití základní architektury *VGG16* je tato síť jednoduše pochopitelná a zdá se být jako vhodný kandidát pro základní experiment. Nevýhodou této sítě však je to, že síť *VGG16* je velmi paměťově náročná a nevešla se do paměti dostupných grafických karet (použitý hardware viz sekce 5.1) a proto byla tato síť vyloučena z dalších experimentů. Po tomto zjištění padlo zaměření na ještě jednodušší neuronové sítě a také na sítě optimalizované pro mobilní zařízení (jelikož ty ze své podstaty nemůžou být paměťově náročné kvůli hardwarovému omezení mobilních zařízení).

Další pokusy byly se sítí *SegNet*, která je popsána v sekci 2.2. Tato síť je však také postavena na architektuře *VGG-16*, nicméně šetří paměťové nároky díky propojením pouze indexů částí kódovacích vrstev. I v tomto případě však byl problém s pamětí grafických karet, nicméně v práci autoři uvádějí také podstatně menší síť nazvanou *SegNet-Basic*, která se skládá jen z několika modulů a umožňuje získávat segmentační masku. Tuto síť se již podařilo zprovoznit na dostupném hardwaru a byla tedy dále využívána pro experimenty. Na této síti byly nejprve trénovány algoritmy pro obrázkovou segmentaci a následně experimentováno se získáváním informací z videa a implementace modulů *FlowCNN* a *NetWarp*. Architektura sítě *SegNet-Basic* byla zjednodušena do podoby pouze kodéru-dekodéru, tzn. bez skip spojení, ani spojení indexů. V práci jsou dále označovány experimenty se sítí *SegNet* při použití architektury *SegNet-Basic*. Originální, velká síť, nebyla použita pro experimenty. Druhou sítí pro experimenty již je originální, neupravená síť – *ICNet* (viz sekce 2.2), která je (v současné době) jednou z nejlepších sítí pro mobilní zařízení a proto nebyl problém s hardwarovým omezením.

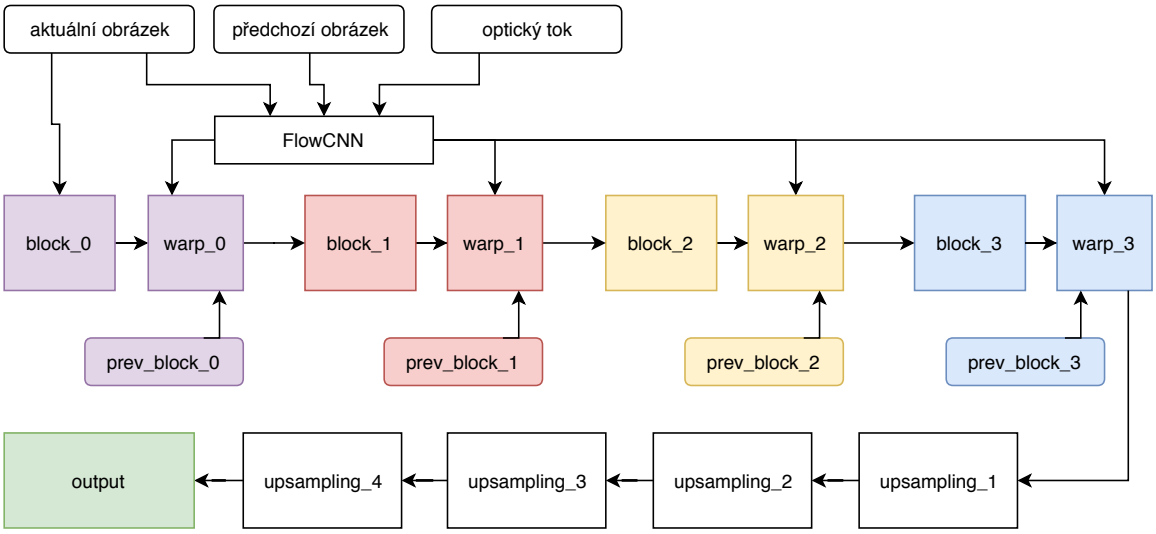
Obě zmiňované sítě využívají pouze jeden modul *FlowCNN*, který je mezi všemi použitými *NetWarp* moduly sdílen, co simuluje použití jednoho výsledku optického toku a *NetWarp* modul si musí velikost tohoto optického toku zmenšit podle dané vrstvy, na které je aplikován.

Na začátku experimentovaní s videem u daných segmentačních sítí byla větev pro předchozí snímek videa duplikována, co znamenalo, že síť trénovala vrstvy obou větví pouze daným snímkem. Při testování sítě pak pro nultý snímek videa byl použit stejný snímek (jelikož předchozí snímek není ještě dostupný), pro každý další snímek videa již byl využit předchozí. V pozdější fázi práce se sítěmi jsem si však uvědomil, že síť obsahuje redundantní vrstvy a tím se zvětšuje jak paměťová, tak časová náročnost dané sítě. Po následujícím prozkoumání možností byly sítě upraveny tak, že obsahovaly pouze jednu větev pro segmentaci a musely být upraveny zvlášť pro trénovací a testovací fázi. Při tréninku sítě byly jednotlivé snímky videa propagovány těmi samými vrstvami, tzn., že váhy pro aktuální a předchozí

snímek videa jsou sdíleny. Tento princip ale umožňuje, v testovací fázi, sítí propagovat pouze jeden snímek videa. V tomto případě jsou vstupy neuronové sítě rozšířeny o vrstvy, na které je každý *NetWarp* modul napojen. Pro nultý snímek videa se pak segmentace provádí bez využití warpování a vrstvy pro předchozí snímek jsou nastaveny na nulové hodnoty. V dalších snímcích je potřeba si uchovat výsledek inference sítě pro konkrétní vrstvy, které jsou vstupem *NetWarp* modulů a ty pak použít jako vstup sítě pro každý další krok.

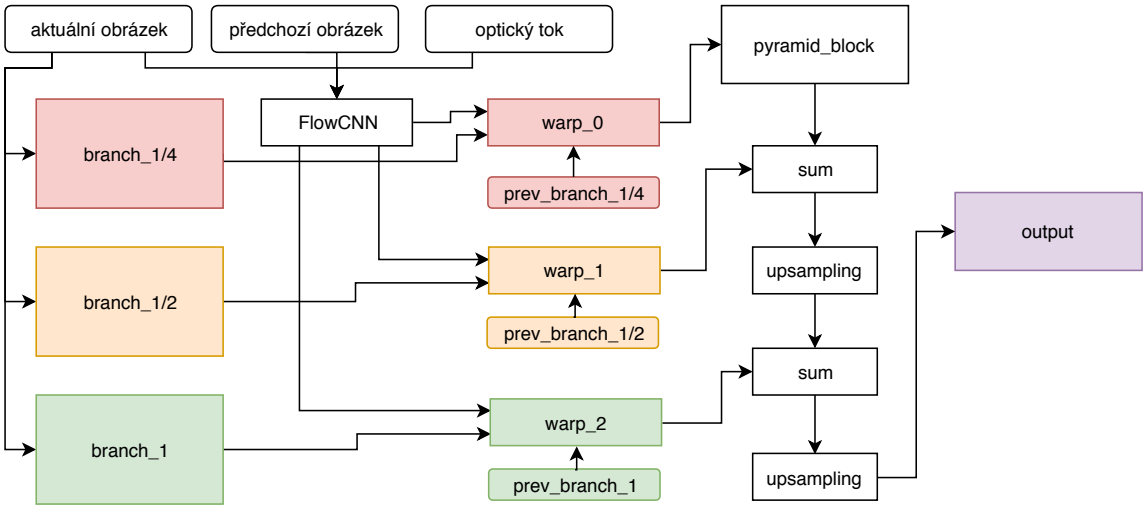
Následuje specifický popis obou použitých architektur.

SegNetWarp Je upravenou sítí základní sítě *SegNet-Basic*. Ta se skládá z několika bloků, které kombinují konvoluční vrstvu, batch normalizaci, aktivaci (ReLU), a max pooling. Tato síť neobsahuje žádná spojení (*skip connections*) mezi kódovací a dekódovací části. Architektura sítě byla upravena zmiňovanými *NetWarp* bloky tak, že byly vloženy vždy mezi kódovací blok. Tato architektura (pro testovací fázi) se všemi zkoušenými pozicemi *NetWarp* bloků je představena na obrázku 4.3. Zelenou barvou je označený výstup sítě, který je ve stejných rozměrech, jako vstupní data a počtem kanálů odpovídajících počtu klasifikovaných tříd. Ostatními barvami jsou vždy vyznačeny části sítě, které je potřeba spojovat a docílit tak správného warpování aktivačních map. Zaoblený obdélník `prev_block_*` znázorňuje vstup sítě, který je výstupem daného bloku pro předchozí snímek videa.



Obrázek 4.3: Architektura sítě *SegNetWarp* se všemi navrženými polohami modulu *NetWarp*. Tato architektura zobrazuje síť pro testovací fázi, tzn. obsahuje bloky výsledků předchozího kroku.

ICNetWarp Síť obsahuje stejnou architekturu jako originální síť (popsána v sekci 2.2), nicméně na obrázku 4.4 je možné vidět zjednodušenou verzi této architektury (pro přehlednost). Tato síť dělí vstupní obraz na tři větve, kde každá větev jej zpracovává v jiném rozlišení a získává příznakové mapy s různou úrovní detailů. Pro tuto síť byly navrženy tři místa vložení *NetWarp* modulu – na „konci“ každé z větví. Na obrázku jsou seskupeny vrstvy dle barev (barevné označení odpovídá původnímu obrázku 2.6 sítě *ICNet*). Fialovou barvou je označen výstup sítě. V blocích *upsampling* jsou zahrnuty vrstvy nadvzorkování (v této práci pomocí bilineární interpolace), konvoluční vrstva a batch normalizace.



Obrázek 4.4: Architektura sítě *ICNetWarp* se všemi navrženými polohami modulu *NetWarp* pro testovací fázi

4.3 Generování trénovacích dat

Jelikož je segmentace prováděna nad celým vstupním obrazem najednou a je získávána pro všechny pixely daného vstupu, je vhodné mít vstupní obrázky ve velkém rozlišení. Dalo by se říci, že čím větší rozlišení, tím kvalitnější může být segmentace. Kvůli této vlastnosti se ale žádná větší datová sada pro segmentaci nevejde celá do paměti `RAM` počítače a je potřeba zajistit automatické plnění fronty vstupních dat během trénování sítě. Pro tuto skutečnost byl navržen a implementován generátor trénovacích a validačních dat. Jelikož je v práci experimentováno s více datovými sadami, byl navržen obecný generátor `BaseDataGenerator` plnící trénovací frontu pouze obrázky a segmentační maskou, a `BaseFlowGenerator`, který přidává manipulaci s optickým tokem. Použité datové sady pak využívají dědičnost pro získání informací o vstupních obrázcích, případně přidávají zvláštní funkcionalitu pro práci s datovou sadou. V práci byly implementovány tři generátory pro datové sady *Playing for Data – GTA 5*, *CamVid*, a *Cityscapes* (datové sady popsané v sekci 3.1). Poslední ze zmíněných byl také upraven pro práci se sítí *ICNet*, které generátor musí místo jedné segmentační masky vkládat 3, zmenšené vhodně podle výstupů sítě.

Generátory pro klasické neuronové sítě, fungující na jednotlivých obrázcích, musí plnit frontu dvojicemi (I, M), kde I je vstupní obrázek a M je anotační maska. Pro trénování na video sekvencích je však potřeba tento generátor upravit a rozšířit o další informace, které bude síť využívat. Tato skutečnost je popsána v rovnici (4.1), kde je vstupní parametr I upraven tak, že jsou přidány snímky videa z předchozích časových úseků (reprezentovány posloupností $I_{-n} \dots I_0$). Tato práce nevyužívá snímky videa z budoucnosti, což je v rovnici naznačeno indexem 0 u parametru I. 2D matice optického toku je představena parametrem O a je spočítána mezi snímky určenými operací $\leftarrow$, která zároveň určuje směr optického toku (tedy reverzní optický tok).

$$I = (I_{-n} \dots I_0, O_{-j \leftarrow 0}) \tag{4.1}$$

Obrázky jsou během trénování po každé epoše náhodně zamíchány, aby nedocházelo k naučení neexistujících závislostí mezi jednotlivými epochami, a na jednotlivé obrázky jsou aplikovány tyto úpravy (příklad viz obrázek 4.5):

1. Zmenšení velikosti obrázků – pro účely této práce jsou obrázky zmenšeny na 1/4 původní velikosti a to z důvodu paměťových nároků segmentačních sítí (ačkoliv síť *ICNet* by šlo trénovat na původní velikosti obrázku, pro srovnávací účely se sítí *SegNet* byla velikost zachována). Toto omezení dále implikuje, že výsledky nelze porovnávat s existujícími metodami přímo, protože ty jsou vždy trénovány na původním rozlišení obrázků datové sady.
2. Náhodná změna jasu – aplikované pouze pro trénovací data
3. Normalizování do rozsahu 0 – 1
4. Vycentrování okolo 0 – odečtení střední hodnoty pixelů datové sady. Datová sada *Cityscapes* má tuto hodnotu předpočítanou.
5. Normalizování podle standardní odchylky – také tato hodnota je v datové sadě *Cityscapes* předpočítána

Obrázek 4.5: Augmentace vstupních obrázků datové sady *Cityscapes*

4.4 Inference na videu

Použité architektury v této práci jsou vždy upraveny tak, aby byly použitelné na video sekvencích. Inference neuronových sítí využívajících předchozí snímky videa vyžaduje specifický algoritmus, aby bylo možné využívat warpování informací z předchozích snímků. Princip je nastíněn v algoritmu 1. Jako první je potřeba nainicializovat model s použitím natrénovaných váh a dále připravit video pro zpracování. Je potřeba mít předem připravené, které vrstvy z předchozího snímku budou používány pro segmentaci aktuálního snímku. Tyto vrstvy jsou zahrnuty do výstupů modelu. Následně se zpracovává každý snímek samostatně, dokud video neskončí. Situace se dále liší pro první snímek videa a zbytek. V případě prvního snímku síť nemá žádné předchozí zpracované snímky a tedy je potřeba vytvořit umělé vstupy do neuronové sítě tak, aby segmentace proběhla pouze na aktuálním snímku. Ve všech dalších snímcích videa již informace z předchozího snímku dostupné jsou a tedy použijí se uložené výstupy požadovaných vrstev spolu s předchozím i aktuálním snímkem videa, mezi kterými se spočítá optický tok. V dalším kroku je již možné provézt inferenci neuronové sítě s warpováním. Výstupy sítě po inferenci je potřeba uložit do dočasné proměnné pro budoucí zpracování. Mezi výstupy již také budou predikované masky segmentovaných tříd, které je dále možné zpracovávat dle požadavků aplikace.

Algorithm 1: Algoritmus zpracování videa s inferencí neuronové sítě pro získání segmentace

```
while true do
    načti snímek videa;
    if první snímek then
        připrav vstupy se syntetickými hodnotami předchozího snímku;
    else
        spočítej optický tok;
        připrav vstupy s použitím předchozího snímku;
    end
    proveď inferenci neuronové sítě;
    ulož výstupy potřebných vrstev pro warpování;
    nastav aktuální snímek jako předchozí;
end
```

Pro simulaci vstupních informací pro první snímek videa byly vyzkoušeny dva způsoby. První z nich (a více logický) bylo použití nulové matice stejných rozměrů jako vrstva určená k warpování. V takovém případě by se větve předchozího snímku v podstatě ignorovaly, protože by se hodnoty po vynásobení naučené váhy vrstvy `LinearCombination` ignorovaly. Nicméně v tomto případě docházelo k artefaktům v segmentačních maskách u prvního snímku a výsledek byl podstatně horší než u sítě nepracující na videu. Druhou možností tedy bylo místo nulové matice použít pro všechny předchozí vrstvy matici jednotkovou, co umožnilo získat kýžený výsledek segmentace pouze aktuálního snímku videa. V této práci byl pak navržen algoritmus pro evaluaci různých sítí na stejných videích (`video_inference.py`), který umožňuje načíst více modelů sítí a provézt inferenci na stejném videu pro každou síť tak, aby všechny sítě byly aplikovány na stejných sekvencích videa stejným způsobem.

Kapitola 5

Experimenty

Tato kapitola prezentuje experimenty, které byly prováděné s navrženými neuronovými sítěmi. V sekci 5.1 jsou představeny nástroje, které byly pro práci se sítěmi použity včetně hardwaru, na kterých bylo trénováno. Následně sekce 5.3 se zaobírá metrikami pro měření výsledků segmentace během trénování i validace. Poslední sekce ukazuje, jakým způsobem je potřeba pracovat s modely sítí během inference na videu.

Všechny zde zmiňované experimenty využívaly datovou sadu *Cityscapes*, která je popsaná v sekci 3.1. Z důvodů velikosti paměti grafických karet, na kterých probíhaly experimenty (viz sekce 5.1) byly rozměry vstupních obrázků zmenšeny na 1/4 původní velikosti, konkrétně na velikost 512×256 `px`. U všech experimentů byl také použit optimalizátor chybové funkce *Adam* se specifickým nastavením parametrů v různých experimentech.

5.1 Použité nástroje

Prototypování a finální experimenty byly implementovány pomocí frameworku *Keras*[1] s backendem implementovaným v *Tensorflow*[2]. Jelikož se neuronové sítě chovají z velké části jako černé skříňky, je potřeba monitorovat stav trénování a porovnávat výsledky jednotlivých experimentů. Pro tento případ byl použit nejprve framework *TensorBoard*, který má spoustu funkcionality, jak prozkoumávat dané neuronové sítě a zjistit tak, kde by bylo možné sítě vylepšit, nicméně problém nastává v použití tohoto nástroje s frameworkem *Keras* a trénování pomocí generátoru dat (co v tomto případě bylo aplikováno). V tomto nastavení jsou možnosti *TensorBoardu* omezené na zobrazování průběhu trénování a validace (grafy výsledků) a zobrazení grafů sítí. V pozdější fázi jsem také objevil monitorovací nástroj *LossWise*[3], který umožňuje podobnou funkcionalitu jako *TensorBoard*, s tím rozdílem, že jsou výsledné hodnoty posílány na server daného frameworku místo ukládání na lokální disk, jako tomu je v případě *TensorBoard*. Tato funkčnost, ačkoliv se zdá triviální, je velmi užitečná v případě, že trénování probíhá na více počítačích a je potřeba výsledky nějakým způsobem agregovat, aby je bylo možné jednoduše porovnávat. Problémem frameworku *LossWise* jsou však omezené možnosti manipulace s grafy, takže v případě, že je sítí více či obsahují velké množství hodnot (framework ve výchozím nastavení získává každou iteraci při trénování neuronové sítě) se framework seká a je obtížné ho použít.

[1] Dostupné na https://keras.io/
[2] Dostupné na https://www.tensorflow.org/
[3] Dostupný na https://losswise.com/

Název GPU	Velikost paměti	Taktovací frekvence
NVIDIA Tesla K20m	4743 MiB	706 MHz
NVIDIA Tesla K20Xm	5700 MiB	732 MHz
NVIDIA GeForce GTX 1060	6072 MiB	1506 MHz

Tabulka 5.1: Grafické karty použité k trénování neuronových sítí

Trénování probíhalo na několika dostupných počítačích, jak na gridové službě *Metacentrum*, tak na vlastním počítači. *Metacentrum* poskytuje dva clustery, *Doom* a *Zubat*, s možností použití knihovny *cuDNN* pro optimalizované výpočty na grafických kartách. V tabulce 5.1 je možné vidět přehled grafických karet, na kterých byly sítě trénovány. Jelikož grafické karty nejsou nejmodernějšího charakteru (zvláště neobsahují dostatečnou velikost paměti), bylo trénování obtížné a časově velmi náročně i přes to, že vstupní data byly podstatně zmenšeny (viz např tabulka 5.2). Dobu trénování se podařilo ke konci experimentů zkrátit díky triku s kopírováním dat na lokální disky výpočetních uzlů, bohužel většina experimentů nebyla ovlivněna.

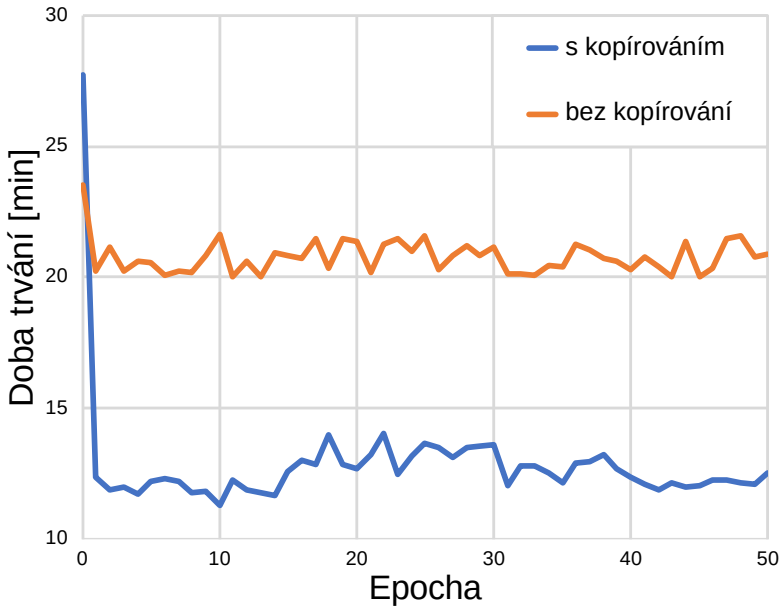
5.2 Trénování sítí

Trénování segmentačních neuronových sítí je časově náročná záležitost, jelikož je potřeba získávat výstup stejné velikosti jako vstupní data a pro rozumně kvalitní segmentaci je potřeba použít velmi hluboké neuronové sítě. Pro debugování takovýchto sítí, zvláště při implementaci nových modulů pracujících s daty, pro které síť nebyla navržena je potřeba zmenšit tuto dobu trénování. Pro tento přístup byly sítě nejprve trénovány s velmi malým množstvím dat a toto množství bylo v případě potvrzení implementované hypotézy zvyšováno. Při těchto scénářích je potřeba omezit míru náhody, jež se může vyskytovat při trénování neuronových sítí. Pro tuto minimalizaci byla nejprve nastavena globální hodnota *seed* pro pseudo náhodné generátory knihovny *numpy* tak, aby ve všech experimentech byla stejná. Knihovnu *numpy* využívají použité frameworky pro práci s neuronovými sítěmi (*Tensorflow* i *Keras*). V dalším kroku musela být vypnuta augmentace vstupních dat a regularizace, jelikož toto se také provádí náhodným způsobem. Jelikož se pracuje s velmi malým množstvím dat, je potřeba dále zajistit, aby byly použity stejné vstupní obrázky napříč experimenty – dosaženo vypnutím náhodného promíchávání datové sady. V tomto případě ale reprezentovaná data nerespektovaly správné rozložení klasifikačních tříd, nicméně i přes to bylo možné rozpoznat zda nějaká implementace funguje lépe či hůře. Takto upravené generování dat a trénování při použití 20 snímků datové sady po dobu 150 epoch trvalo v průměru hodinu.

Kopírování dat na lokální disk Jak bylo zmíněno v předchozí sekci, většina experimentů probíhala na gridové službě *Metacentrum*. Výpočty na těchto serverech jsou přiřazovány na předem neurčené uzly (dle parametrů a dostupnosti uzlů), které jsou propojeny síťovými disky. Návod k práci s daty na *Metacentru*[4] zmiňuje, že pro větší datové sady by mělo být přistupováno přímo k síťovým diskům, kde se data nachází. Většinu času na tomto projektu tak bylo prováděno, jelikož velikost datové sady je cca 44GB, nicméně později bylo zjištěno, že výpočty na této gridové službě trvají velkou dobu z důvodu nevyužití poten-

[4]Dostupný na https://wiki.metacentrum.cz/wiki/Working_with_data/Working_with_data_in_a_job

ciálu grafické karty, jelikož načítání dat ze síťového disku a následná manipulace s těmito daty trvalo delší dobu nežli trénování daného snímku na grafické kartě. Po zjištění tohoto faktu byl prozkoumáno, jakým způsobem by bylo možné trénování zrychlit. Na *Metacentru* je možné specifikovat tzv. *scratch* složku, která je vytvořena pouze pro účely dané úlohy a nachází se vždy na stejném uzlu jako výpočet. Pro urychlení plnění trénovací fronty byla zavedena funkce v generátoru dat, která kontroluje, zda požadovaný soubor existuje v dané *scratch* složce a pokud ne, překopíruje tento soubor do této složky. Výsledkem této manipulace bylo sice zpomalení první epochy trénování kvůli čekání na zkopírovaný soubor, nicméně v průběhu dalšího trénování již k tomuto čekání nedocházelo. Na obrázku 5.1 je možné vidět porovnání průběhů trénování stejné sítě *ICNetWarp0* pro algoritmus využívající kopírování dat a bez něj. Doba zkompletování první epochy v případě kopírovacího algoritmu trvá cca 28 minut, kdežto bez kopírování 24 minut. Pro další epochy však kopírovací algoritmus urychlil trénování 1.6× z 21 minut na 13 minut v průměru. Tento způsob urychlení trénování však byl aplikován až ke konci práce s experimenty a tedy ve většině času nebyl uplatněn.



Obrázek 5.1: Průměrná doba trénování sítě *ICNetWarp1* na clusteru *Doom* výpočetního centra *Metacentrum*. Porovnány jsou algoritmus využívající kopírování dat na lokální disk uzlu a bez něj

5.3 Měření výsledků

Měření výsledků je pro hluboké učení klíčové, jelikož trénování trvá dlouhou dobu a není jednoduše zjistitelné, zda daný experiment funguje lépe či hůře. V případě segmentačních úloh je výsledkem mapa predikovaných tříd, které je možné vizualizovat a tím výsledky porovnávat kvalitativně, nicméně výsledky se často liší minimálně a proto jsou kvalitativní výsledky porovnávány pouze pro určitě částí obrazu (například hrany objektů, tenké objekty, či obecně problematické části). V takovém případě je potřeba provádět kvantitativní měření a tím mít přesně specifikované, jak dobře si daná metoda vede oproti jiným.

Categorical crossentropy Pro účely trénování a zpětné propagace chyby a minimalizování chybové funkce byla použita funkce *categorical crossentropy*, která umožňuje pracovat s více predikovanými třídami a je definovaná rovnicí

$$-\sum_{c=1}^{M} y_{o,c} \log(p_{o,c}), \tag{5.1}$$

kde M je počet predikovaných tříd, c je predikována třída, o je sledovaná hodnota, y je binární indikátor (0 nebo 1), zda je daná třída c správně klasifikována pro o a p je pravděpodobnost, zda je o dané třídy c. Tato chybová funkce více penalizuje špatné predikce s velkou pravděpodobností. [6]

Mean intersection over union V předchozím odstavci byla popsána chybová funkce, která definuje, jak se má neuronová síť učit a jakým směrem se mají gradienty propagovat. Pro získání informací o vhodných výsledcích je možné použít algoritmus, který nemá přímo vliv na proces učení. Nejčastějším algoritmem pro tyto účely je *intersection over union* (někdy také označován jako *Jaccardův index*). Tato metrika určuje poměr mezi průnikem a sjednocením mezi anotovanou maskou a predikovanou maskou. Průnik je v tomto případě specifikován počtem *true positive*, sjednocení pak jako součet *true positive*, *false negative* a *false positive*. Tato hodnota je spočítána pro každou predikovanou třídu samostatně a následně je zprůměrována pro získání jednoho čísla pro všechny třídy, což je pak možné jednoduše vizualizovat a porovnávat vůči jiným experimentům. Tuto metriku je možné reprezentovat rovnicí

$$mIoU = \frac{1}{k+1} \sum_{i=0}^{k} \frac{p_{ii}}{\sum_{j=0}^{k} p_{ij} + \sum_{j=0}^{k} p_{ji} - p_{ii}}, \tag{5.2}$$

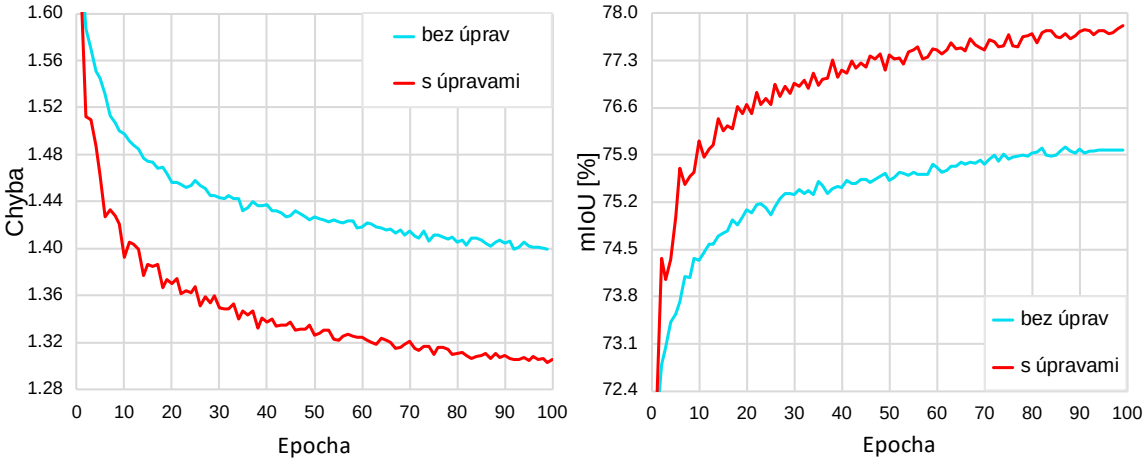
kde $k+1$ je označen počet predikovaných tříd, p_{ij} je počet pixelů třídy i predikovaných do třídy j, tzn. p_{ii} představuje počet *true positive*, p_{ij} *false positive* a p_{ji} *false negative*. [8]

5.4 Realizované experimenty

Tato sekce obsahuje analýzu experimentů a objevů získaných v průběhu této práce. Nejprve je ukázán experiment jež vedl ke změnám v původní architektuře modulu *FlowCNN*, dále je ukázán experiment porovnávající kvantitativně výsledky na obou zmiňovaných architekturách. Nakonec experiment s jiným nastavením chybové funkce a tím i dosažení kvalitativně lepších výsledků za cenu nepřesného trénování sítí.

Úprava FlowCNN

V předchozí sekci 4.1 bylo ukázáno, že modul upravující optický tok byl upraven vůči originální práci. Bylo použito warpování obrázků, upraveny vstupy modulu a parametry poslední konvoluční vrstvy. Použití změn vedlo ke zvýšení efektivity sítě, co je možné vidět na obrázku 5.2. Tento experiment byl proveden na architektuře sítě *ICNetWarp2* s parametry trénování $batch = 8$, $lr = 0.001$ a $lr\_decay = 0.049$. Modul vytvořený přesně podle originální práce dosahoval po 100 epochách trénování mIoU 76.10% a upravený modul 77.77%. Na základě těchto poznatků byly dále všechny experimenty prováděny na sítích s upraveným modulem *FlowCNN*.



(a) Průběh chybové funkce

(b) Průběh mIoU

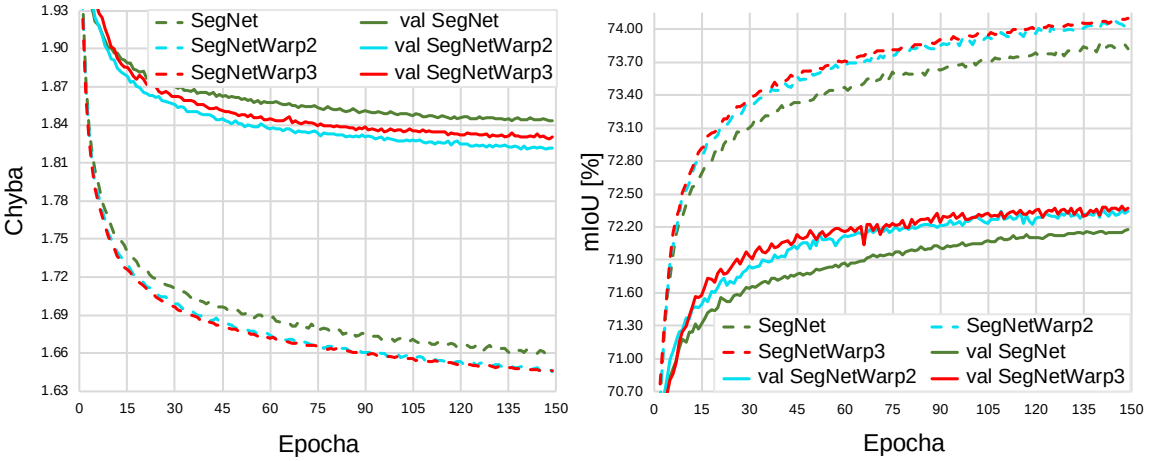
Obrázek 5.2: Průběh hodnot pro validační sadu na síti *ICNetWarp2* při použití originálního a upraveného modulu *FlowCNN*

Kvantitativní experiment

Cílem tohoto experimentu bylo kvantitativně porovnat trénování a výsledky obou použitých neuronových sítí k určení, v jakých vrstvách a kombinacích mohou sítě využívající video lépe fungovat. Pro porovnání těchto experimentů byl zvolen předem počet epoch –150, které obě sítě v tomto experimentu měly během trénování dosáhnout. Parametry trénování byly nastaveny $lr = 0.001$ a $lr\_decay = 0.0051$, které byly empiricky získané tak, aby sítě vhodně konvergovaly a nepřetrénovaly se. Obě implementované architektury jsou porovnávány zvlášť, jelikož cílem je zjistit úspěšnost warpovacích modulů vůči základní architektuře dané sítě. Z výsledků sítí je možné potvrdit zmiňovaný fakt, že není možné porovnávat sítě trénované na menším rozlišení vůči originálním sítím, jelikož např. síť *ICNet* dosahuje v kvalitativním experimentu mIoU 68.05%, nicméně v původní práci autoři uvádějí dosažené mIoU pouze 67.7% (popsané v sekci 2.4). Pro zajímavost (a případné budoucí řešitelé) jsou v tabulkách u jednotlivých výsledků experimentů uvedeny přibližné časy trénování dané sítě[5].

SegNetWarp Tato síť upravuje architekturu *SegNet-Basic*, do které byly navrženy tři lokace *NetWarp* modulů – vždy za jeden kódovací blok sítě (popsané v sekci 4.2). Nastavení *batch* pro trénování bylo nastaveno na čtyři n-tice trénovacích dat, což je největší možná velikost, kterou se podařilo zprovoznit na zmiňovaných grafických kartách. Trénování po dobu 150 epoch u této sítě znamenalo 111 450 iterací neuronovou sítí. Průběh hodnot trénování je možné vidět na obrázku 5.3 a finální výsledky pro toto nastavení jsou v tabulce 5.2. Tato tabulka je rozdělena do dvou částí: výsledky chybové funkce a mIoU a představeny jsou architektury *SegNetWarp0* – *SegNetWarp3*. Z výsledků je vidět, že základní verze sítě *SegNet* (bez warpovacích úprav) dosáhla na validační sadě 72.17% mIoU a nejlépe fungovala architektura s použitím warpovacího modulu na konci 3. kódovacího bloku (*SegNetWarp3*), která dosáhla mIoU 72.37% na validační sadě a je o 0.2% lepší než originální síť. Druhou nejlépe fungující sítí, dle výsledků, je síť *SegNetWarp2*. Zajímavým faktem je, že síť dosahuje lepší validační chyby než síť *SegNetWarp3*, přesto mIoU je trochu horší.

[5] Údaje získané z nástroje *TensorBoard*



(a) Průběh chybové funkce

(b) Průběh mIoU

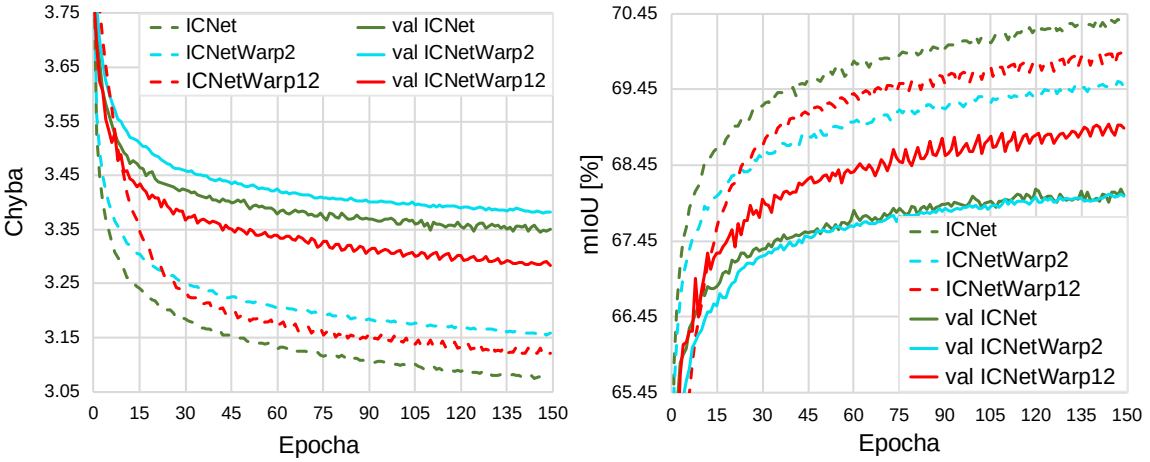
Obrázek 5.3: Průběh chybové funkce a mIoU na testovací a validační fázi pro sítě *SegNet*, *SegNetWarp2* a *SegNetWarp3*

	chyba	val chyba	mIoU [%]	val mIoU [%]	trénováno
SegNet	1.661	1.843	73.82	72.17	1 d 19 h
SegNetWarp0	1.657	1.836	73.85	72.23	2 d 08 h
SegNetWarp1	1.650	1.832	74.00	72.26	2 d 07 h
SegNetWarp2	1.647	1.821	74.07	72.34	2 d 10 h
SegNetWarp3	1.646	1.830	74.09	72.37	2 d 11 h

Tabulka 5.2: Výsledky kvalitativního experimentu trénované sítě *SegNet* a upravených sítí modulem *NetWarp*

ICNetWarp Druhá síť, na které byly prováděny experimenty byla síť *ICNet*. Ta je v základu optimalizovaná pro rychlost a výpočetní výkon (což je vhodné při daném hardwaru, na kterém byly sítě trénovány). Pro experimenty bylo nastaven $batch = 8$, co znamená (55 650 iterací) při použití 150 epoch. Průběh trénování a validace je možné vidět na obrázku 5.4 a výsledky těchto experimentů v tabulce 5.3. Trénování této sítě nebylo pro daný počet epoch příliš úspěšné, jelikož ve většině případů základní síť *ICNet* funguje lépe než upravené sítě. Základní síť dosáhla mIoU na validační sadě 68.05%, kdežto pouze jediná varianta – *ICNetWarp12* – fungovala lépe v mIoU rovným 68.94%. Důvodem tohoto neúspěchu může být fakt, že síť obsahuje operace pro zmenšení velikosti vstupního obrazu na $1/2$, $1/4$ a $1/8$ původní velikosti. Pokud tyto operace aplikujeme na rozměry vstupních dat (512×256), získáme pro nejmenší větev velikost 64×32 pixelů. Na tyto snímky pak jsou dále aplikovány konvoluční a *pooling* vrstvy sítě, tím pádem získání informací o menších objektech (nebo tenkých objektech) je téměř nemožné.

Analýza výsledků Po rozboru výsledků trénování sítí byl zjištěn fakt, že rozdíly mezi upravenými architekturami a základními u obou trénovaných sítí jsou velmi malé. Za tuto skutečnost může více faktorů. Prvním z nich je, že všechny sítě jsou trénovány na dané pa-



(a) Průběh chybové funkce

(b) Průběh mIoU

Obrázek 5.4: Průběh chybové funkce a mIoU na testovací a validační fázi pro sítě *ICNet*, *ICNetWarp2* a *ICNetWarp12*

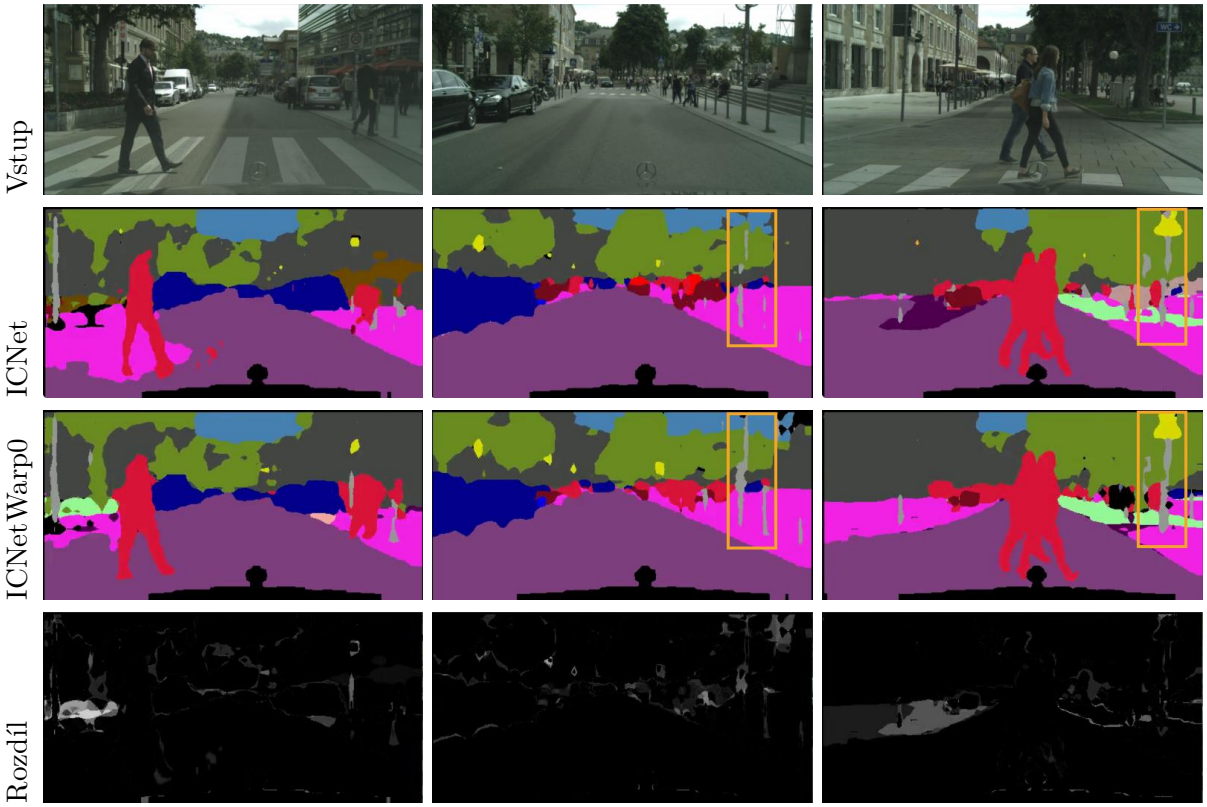
	chyba	val chyba	mIoU [%]	val mIoU [%]	trénováno
ICNet	3.08	3.35	70.37	68.05	0 d 16 h
ICNetWarp0	3.19	3.43	69.32	67.38	1 d 05 h
ICNetWarp1	3.17	3.42	69.35	67.82	1 d 06 h
ICNetWarp2	3.16	3.38	69.48	68.04	1 d 06 h
ICNetWarp12	3.13	3.28	69.87	68.94	1 d 05 h
ICNetWarp012	3.16	3.41	69.65	67.91	1 d 05 h

Tabulka 5.3: Výsledky kvantitativního experimentu trénované sítě *ICNet* a upravených sítí modulem *NetWarp*

rametry příliš krátkou dobu a tím pádem jsou všechny nedotrénované (*underfitted*). Během trénování sítě vhodně konvergovaly (co je možné vidět na grafech u daných sítí), nicméně omezení 150 epoch neumožnilo zkonvergovat do nejlepšího minima chybové funkce. Druhým problémem těchto experimentů je, že všechny sítě mají odlišnou architekturu a volit jedno nastavení parametrů nemusí znamenat, že sítě zkonvergují stejným tempem. V obou případech mělo být trénování ponecháno delší dobu pro více epoch, nicméně z důvodů, že trénování jednoho modelu trvalo déle než jeden den (u některých warpovacích modelů déle než dva dny), zkoušení různých hyperparametrů bylo velmi obtížné a je možné, že nebyly zvoleny nejlepší parametry pro trénování.

Kvalitativní experiment

Po prozkoumání výsledků předchozího experimentu bylo zjištěno, že síť *ICNet* dosahuje horšího mIoU (68.05%) než síť *SegNet* (72.17%), což dle originálních prací těchto architektur mělo být opačně (viz měření v sekci 2.4). Z tohoto důvodu byl proveden druhý experiment, který si kladl za cíl natrénovat vybranou síť lepším způsobem a porovnat ji vůči sítě bez využívání videa. Pro tento případ byla pro experimenty použita síť *ICNet*. Pro trénování byl



Obrázek 5.5: Kvalitativní výsledky na demo video sekvenci datové sady *Cityscapes*. Shora představeny výstupy sítě *ICNet*, *ICNetWarp0* a nakonec rozdíl těchto výstupů.

použit optimalizátor *Adam* s parametry bylo $lr = 0.001$ bez umělého snižování *learning rate*. V tabulce 5.4 je možné vidět dosažené výsledky a na obrázku 5.5 výstupy sítě. Lze vidět, že sítě dosáhly podstatně lepší mIoU (nad 80%). Toto mIoU je vůči originálním pracím vyšší z důvodu menšího vstupu a tedy ztráty detailů pixelů, které by jinak na velkém rozlišení chyběly. Nejlépe ze vyzkoušených sítí dopadla architektura *ICNetWarp0*. Rozdíl mezi sítí využívající video a základní je možné vidět například u tenkých objektů jako jsou sloupy či dopravní značky. Také jsou v některých případech zcela eliminovány chybně segmentované části.

	chyba	mIoU [%]	val chyba	val mIoU [%]
ICNet	0.572	89.31	0.968	84.41
ICNetWarp0	0.499	90.77	0.887	85.88
ICNetWarp2	0.826	90.84	0.912	85.37
ICNetWarp012	0.674	87.76	0.878	85.76

Tabulka 5.4: Výsledky kvalitativního experimentu na síti *ICNet* testované na datové sadě *Cityscapes*

Kapitola 6

Závěr

Cílem této práce bylo, za pomocí hlubokých neuronových sítí, klasifikovat objekty v obraze pouliční scény. Úlohu je možné řešit dvěma způsoby, a to detekovat objekty ve scéně a získat tak ohraničující obdélníky daných objektů, nebo provádět segmentaci celého vstupního obrazu. Tyto způsoby získávání různé míry informace z klasifikačních algoritmů byly rozebrány spolu s existujícími přístupy k dané problematice. Pro účely této práce byl zvolen druhý ze zmiňovaných algoritmů, sémantická segmentace obrazu, díky její větší informační přínosnosti. Jedním z požadavků bylo použití klasického obrazu z monokulární kamery bez použití specializovaných zařízení. Většina dnešních řešení segmentačních neuronových sítí pracuje s jednotlivými obrázky a trend používání videa je teprve na vzestupu. Na základě toho byl cíl práce rozšířen o pracování s video sekvencemi místo obrázků a využití potenciálních informací, které se mezi snímkami videa mohou vyskytovat. Díky této skutečnosti jsou v kapitole 2 popsána jak existující řešení pracující s obrázky, tak pro práci s videem.

Využití videa staví na práci Gadde a spol. [7], kde byl použit princip warpování aktivačních map pomocí optického toku mezi dvěma po sobě jdoucími snímky videa. Na základě těchto informací byly implementovány dva moduly, jeden pro upravení optického toku a druhý pro následné warpování aktivačních map v různých vrstvách neuronové sítě. Tyto moduly pak upravují dvě architektury neuronových sítí – *SegNet-Basic* a *ICNet*. Na těchto sítích následně byly prováděny experimenty pro ověření funkčnosti práce s videem a zároveň pro stanovení relativních rozdílů úspěšnosti mezi základními architekturami a těmi používající video. Hypotéza použití optického toku pro zlepšení segmentace na videích byla potvrzena a sítě upraveny těmito moduly dosahují lepších výsledků.

Během této práce jsem narazil na několik problémů, které by mohly být přínosné příštím řešitelům. Největším problémem bylo hardwarové omezení ve formě starších grafických karet na dostupných vývojových prostředích. V této práci bylo použito několik počítačů k vývoji, nejvíce však výpočetní grid *Metacentrum*, na kterých jsou starší grafické karty. Toto omezení dále implikovalo znesnadnění prací se segmentačními sítěmi, jelikož nebylo možné použít originální architektury nejnovějších sítí a bylo třeba se zaměřit na více paměťově efektivní architektury. I přes zúžení výběru sítí byl problém s pamětí grafických karet, co vyústilo v použití zmenšených rozměrů vstupních videí a tím nemožnost přímého porovnání výsledků modelů s jejími originálními pracemi.

Do budoucna by bylo vhodné natrénovat zmiňované sítě delší dobu, co by umožnilo přesnější náhled do výsledků video segmentace. Vhodným experimentem by bylo trénovat sítě na lepším hardwaru a v plném rozlišení a porovnat tak, jak video informace ovlivňují klasické architektury pracující s obrázky. Z hlediska rozšíření funkcionality zmiňovaných sítí,

bylo by možné otestovat použití více snímků do minulosti, případně na základě minulosti predikovat budoucí snímek, který by mohl být použitý pro aktuální krok videa.

Co se týká segmentačních neuronových sítí pro simulaci lidského vidění pro použití v samořídítelných vozidlech, situace z dnešního hlediska není dle mého názoru velmi použitelná. I přes to, že nejmodernější sítě dosahují velmi dobrých výsledků a vznikají velké datové sady, které umožňují správnou generalizaci pro všechny možné situace na silnicích, v některých případech je výsledek segmentace velmi chaotický, co by mohlo systém vozidla mást. Jako vhodný příklad užití však vidím v blízké budoucnosti použití těchto systému jako doprovodné systémy pro řidiče vozidel, kterými můžou být například systém varování při sjíždění mimo vozovku či pomocné informace o objektech v nepřehledných situacích během řízení.

Literatura

[1] Badrinarayanan, V.; Kendall, A.; Cipolla, R.: SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *CoRR*, ročník abs/1511.00561, 2015.

[2] Brostow, G. J.; Fauqueur, J.; Cipolla, R.: Semantic Object Classes in Video: A High-Definition Ground Truth Database. *Pattern Recognition Letters*, ročník 30, č. 2, 2009: s. 88–97.

[3] Brostow, G. J.; Shotton, J.; Fauqueur, J.; aj.: Segmentation and Recognition Using Structure from Motion Point Clouds. In *ECCV (1)*, 2008, s. 44–57.

[4] Cordts, M.; Omran, M.; Ramos, S.; aj.: Benchmark Suite - Cityscapes Dataset. https://www.cityscapes-dataset.com/benchmarks/#pixel-level-results, [Online; navštíveno 13.1.2018].

[5] Cordts, M.; Omran, M.; Ramos, S.; aj.: The Cityscapes Dataset for Semantic Urban Scene Understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[6] Fortuner, B.: Loss Functions — Machine Learning Cheatsheet. http://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html#cross-entropy, [Online; navštíveno 10.5.2018].

[7] Gadde, R.; Jampani, V.; Gehler, P. V.: Semantic Video CNNs through Representation Warping. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[8] Garcia-Garcia, A.; Orts-Escolano, S.; Oprea, S.; aj.: A Review on Deep Learning Techniques Applied to Semantic Segmentation. *CoRR*, ročník abs/1704.06857, 2017.

[9] Gershgorn, D.: ImageNet: the data that spawned the current AI boom — Quartz. https://qz.com/1034972/the-data-that-changed-the-direction-of-ai-research-and-possibly-the-world/, Červenec 2017, (Online; navštíveno 14.1.2018).

[10] Girshick, R.: Fast R-CNN. In *International Conference on Computer Vision (ICCV)*, 2015.

[11] Girshick, R.; Donahue, J.; Darrell, T.; aj.: Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition*, 2014.

[12] He, K.; Gkioxari, G.; Dollár, P.; aj.: Mask R-CNN. *CoRR*, ročník abs/1703.06870, 2017.

[13] He, K.; Zhang, X.; Ren, S.; aj.: Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. *CoRR*, ročník abs/1406.4729, 2014.

[14] Jaderberg, M.; Simonyan, K.; Zisserman, A.; aj.: Spatial Transformer Networks. *CoRR*, ročník abs/1506.02025, 2015.

[15] Liu, W.; Anguelov, D.; Erhan, D.; aj.: SSD: Single Shot MultiBox Detector. *CoRR*, ročník abs/1512.02325, 2015.

[16] Long, J.; Shelhamer, E.; Darrell, T.: Fully Convolutional Networks for Semantic Segmentation. *CoRR*, ročník abs/1411.4038, 2014.

[17] Neuhold, G.; Ollmann, T.; Bulò, S. R.; aj.: The mapillary vistas dataset for semantic understanding of street scenes. In *Proceedings of the International Conference on Computer Vision (ICCV), Venice, Italy*, 2017, s. 22–29.

[18] OpenCV: Optical Flow. https://docs.opencv.org/3.3.1/d7/d8b/tutorial_py_lucas_kanade.html, [Online; navštíveno 24.4.2018].

[19] Redmon, J.; Divvala, S. K.; Girshick, R. B.; aj.: You Only Look Once: Unified, Real-Time Object Detection. *CoRR*, ročník abs/1506.02640, 2015.

[20] Redmon, J.; Farhadi, A.: YOLO9000: Better, Faster, Stronger. *CoRR*, ročník abs/1612.08242, 2016.

[21] Richter, S. R.; Vineet, V.; Roth, S.; aj.: Playing for Data: Ground Truth from Computer Games. In *European Conference on Computer Vision (ECCV)*, *LNCS*, ročník 9906, editace B. Leibe; J. Matas; N. Sebe; M. Welling, Springer International Publishing, 2016, s. 102–118.

[22] dos Santos, L. A.: Object Localization and Detection · Artificial Inteligence. https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/object_localization_and_detection.html, [Online; navštíveno 2.1.2018].

[23] Shah, M.: Semantic Segmentation using Fully Convolutional Networks over the years. https://meetshah1995.github.io/semantic-segmentation/deep-learning/pytorch/visdom/2017/06/01/semantic-segmentation-over-the-years.html, Červen 2017, [Online; navštíveno 12.1.2018].

[24] Shelhamer, E.; Rakelly, K.; Hoffman, J.; aj.: Clockwork convnets for video semantic segmentation. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, ročník 9915 LNCS, 2016: s. 852–868, ISSN 16113349.

[25] Tryolabs - Javier Rey: Object detection: an overview in the age of Deep Learning - Tryolabs Blog. https://tryolabs.com/blog/2017/08/30/object-detection-an-overview-in-the-age-of-deep-learning/, Srpen 2017, [Online; navštíveno 5.1.2018].

[26] Zhao, H.; Qi, X.; Shen, X.; aj.: ICNet for Real-Time Semantic Segmentation on High-Resolution Images. *arXiv preprint arXiv:1704.08545*, 2017.

[27] Zhao, H.; Shi, J.; Qi, X.; aj.: Pyramid Scene Parsing Network. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[28] Žára, J.; Beneš, B.; Sochor, J.; aj.: *Moderní počítačová grafika*. Computer Press, 2005, ISBN 80-251-0454-0.

Příloha A

Obsah přiloženého DVD

doc	zdrojové kódy technické práce a samotná práce v PDF
results	ukázky testovacích videí
src	zdrojové kódy použité k trénování a testování neuronových sítí
weights	váhy natrénovaných sít
video.mp4	video prezentace této práce