VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

SEGMENTACE OBRAZOVÝCH DAT POMOCÍ HLUBOKÝCH NEURONOVÝCH SÍT

IMAGE SEGMENTATION WITH DEEP NEURAL NETWORK

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **Bc. RADEK PAZDERKA**
AUTHOR

VEDOUCÍ PRÁCE **Ing. JAROSLAV ROZMAN, Ph.D.**
SUPERVISOR

BRNO 2019

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav inteligentních systémů (UITS) Akademický rok 2018/2019

Zadání diplomové práce

Student: **Pazderka Radek, Bc.**
Program: Informační technologie Obor: Inteligentní systémy
Název: **Segmentace obrazových dat pomocí hlubokých neuronových sítí**
Image Segmentation with Deep Neural Network
Kategorie: Zpracování obrazu

Zadání:

1. Prostudujte problematiku segmentace obrazových dat pomocí hlubokých neuronových sítí (DNN).
2. Vytvořte datovou sadu pro trénování/testování úspěšnosti segmentačních metod. Zaměřte se na scénáře z městského prostředí a silniční sítě (chodci, vozidla, silnice, autobusy apod.). Rozsah datové sady zvolte takový, aby dostatečně pokrývala všechny objekty, které se mají segmentovat.
3. Navrhněte vlastní algoritmus s využitím DNN pro segmentaci obrazových dat. Tento algoritmus implementujte, analyzujte a testujte na vytvořené datové sadě.
4. Zhodnoťte dosažené výsledky a navrhněte možnosti dalšího pokračování práce. Vytvořte plakátek a krátké video prezentující výsledky vaší práce.

Literatura:

- B. Shuai, Z. Zuo, B. Wang and G. Wang, "Scene Segmentation with DAG-Recurrent Neural Networks," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 6, pp. 1480-1493, 1 June 2018. doi: 10.1109/TPAMI.2017.2712691
- V. Badrinarayanan, A. Kendall and R. Cipolla, "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, pp. 2481-2495, 1 Dec. 2017. doi: 10.1109/TPAMI.2016.2644615

Při obhajobě semestrální části projektu je požadováno:

- první dva body zadání

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce: **Rozman Jaroslav, Ing., Ph.D.**
Vedoucí ústavu: Hanáček Petr, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 22. května 2019
Datum schválení: 1. listopadu 2018

Abstrakt

Tato diplomová práce se zaměřuje na segmentaci scény z dopravního prostředí. Řešením tohoto problému jsou segmentační neuronové sítě, které umožňují klasifikovat každý pixel scény. V rámci této diplomové práce byla vytvořena vlastní segmentační neuronová síť, která dosáhla lepších výsledků než dosavadní state-of-the-art architektury. Práce se také zaměřuje na segmentaci ptačích pohledů na vozovku, ze kterých neexistují volně dostupné anotované datové sady. za tímto účelem byl vytvořen automatický nástroj pro generování syntetických datových sad z PC hry Grand Theft Auto V. Práce srovnává sítě trénované pouze na syntetických datech a sítě trénované na společně reálných a syntetických datech. Experimenty dokazují, že syntetická data lze využít na segmentaci dat z reálného prostředí. Také byl implementován systém, který umožňuje veškerou práci se segmentačními neuronovými sítěmi.

Abstract

This masterś thesis is focused on segmentation of the scene from traffic environment. The solution to this problem is segmentation neural networks, which enables classification of every pixel in the image. In this thesis is created segmentation neural network, that has reached better results than present state-of-the-art architectures. This work is also focused on the segmentation of the top view of the road, as there are no freely available annotated datasets. For this purpose, there was created automatic tool for generation of synthetic datasets by using PC game Grand Theft Auto V. The work compares the networks, that have been trained solely on synthetic data and the networks that have been trained on both real and synthetic data. Experiments prove, that the synthetic data can be used for segmentation of the data from the real environment. There has been implemented a system, that enables work with segmentation neural networks.

Klíčová slova

konvoluční neuronová síť, segmentace obrazu, analýza dopravy, generování umělé datové sady, vlastní neuronová síť

Keywords

convolutional neural network, image segmentation, traffic analysis, generating synthetic dataset, custom neural network

Citace

PAZDERKA, Radek. *Segmentace obrazových dat pomocí hlubokých neuronových sít.* Brno, 2019. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Jaroslav Rozman, Ph.D.

Segmentace obrazových dat pomocí hlubokých neuronových sít

Prohlášení

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

.....
Radek Pazderka
22. května 2019

Poděkování

Nejprve bych rád poděkoval firmě Artin s.r.o, která mi zapůjčila zdrojové kódy umožňující generování anotovaných textur ze hry GTAV. Dále bych rád poděkoval svému vedoucímu diplomové práce Ing. Jaroslavu Rozmanovi, Ph.D. za jeho věcné připomínky na konzultacích.

Obsah

Úvod

V dnešní době informací je stále větší požadavek na zpracovávání nasbíraných obrazových dat. Jako je např. klasifikace, detekce, sledování objektů nebo segmentace celé scény. K takovému množství dat již nestačí lidská síla a musí se začít využívat počítače. K řešení těchto problémů se v dnešní době používají neuronové sítě. Neuronové sítě se inspirovaly lidským mozkem. Mozkové buňky neboli neurony jsou v lidském mozku mezi sebou propojené pomocí synapsí. Každé propojení neuronů má určitou sílu. Když do neuronu ze vstupních neuronů projde silný vzruch, tak se daný neuron aktivuje. Stejný koncept je i v počítačových neuronových sítí. Každý neuron v počítačovém světě je definovaný jako suma vstupních neuronů vynásobena silou propojení neboli váhou. Když tento součet dosáhne určité hodnoty je aktivační funkcí aktivován.

V této diplomové práci se nebudeme zabývat úplnými základy neuronových sítí. Zaměříme se na konvoluční neuronové sítě, které provádí segmentaci celého obrazu. Jinými slovy bude se jednat o neuronovou síť, která bude klasifikovat každý pixel vstupního obrazu. Výstupem této neuronové sítě bude maska vstupního obrazu, která bude uvádět přesné pozice a tvary všech definovaných tříd.

V teoretické části práce budeme detailněji rozebírat důležité části konvolučních neuronových sítí, které jsou velice důležité pro praktické použití. Při popisu budu používat zažité anglické pojmy, u kterých české překlady nejsou standardně používané. Všechny tyto a další důležité pojmy budou sepsány a vysvětleny v příloze.

Výsledky této práce se použijí v praxi k segmentaci obrazu z parkovišť. Bude se jednat o součást systému, který monitoruje a zjišťuje zaplněnost daného parkoviště. Neuronová síť, která bude výstupem této diplomové práce bude pomáhat systému v lepším pochopení a popsání situací vzniklých na parkovišti. Mohou to být např. stojící auta v zakázaných zónách, nebo např. pro spočítání podélně parkujících aut a případně určení volných míst u podélného parkování. Dále se výsledky této práce využijí jako součást záložního navigačního systému pro drony. Tento záložní systém se využije v případě, když dronu vypadne naváděcí systém. V tento moment začne segmentovat scénu po sebou, aby nalezl volné místo, kde by mohl bezpečně přistát a nikoho nezranil nebo sebe nezničil.

Tato diplomová práce je rozdělena do deseti kapitol. V první kapitole se seznámíme s konvolučními neuronovými sítěmi. Uvedeme si základní vrstvy a bloky. Ve druhé kapitole se zaměříme na segmentační sítě, seznámíme se se specifickými vrstvami, které jsou pro tyto sítě typické. Dále si uvedeme jednotlivé moduly a modely, které se v současné době používají k segmentaci obrazu. Ve třetí kapitole si popíšeme generátor jedinečné syntetické datové sady ze známé hry GTAV, která bude použita k trénování segmentační sítě pro segmentaci scény z ptačího pohledu. Ve čtvrté kapitole sjednotíme volně dostupné datové sady společně s vlastní vygenerovanou a provedeme nad ní analýzu. Následně tato datová sada bude využita k trénování a validaci zvolených sítí. V páté kapitole se zaměříme na implementovaný systém, který umožňuje jednoduchou práci se segmentačními sítěmi. Konkrétně

se jedná o trénování, validaci, testování, optimalizaci, analýzu a vytváření segmentačních neuronových sítí. V následující šesté kapitole se seznámíme s architekturou vlastní navržené segmentační neuronovou sítě SeparNet. V sedmé kapitole se zaměříme na trénování vlastní sítě společně se state-of-the-art architekturami. V osmé kapitole provedeme experimenty a zhodnotíme kvalitu jednotlivých sítí. V deváté kapitole si uvedeme závěrečnou optimalizaci vítězné sítě experimentů. V závěrečné kapitole zhodnotíme dosažené výsledky.

Kapitola 1

Konvoluční neuronové sítě

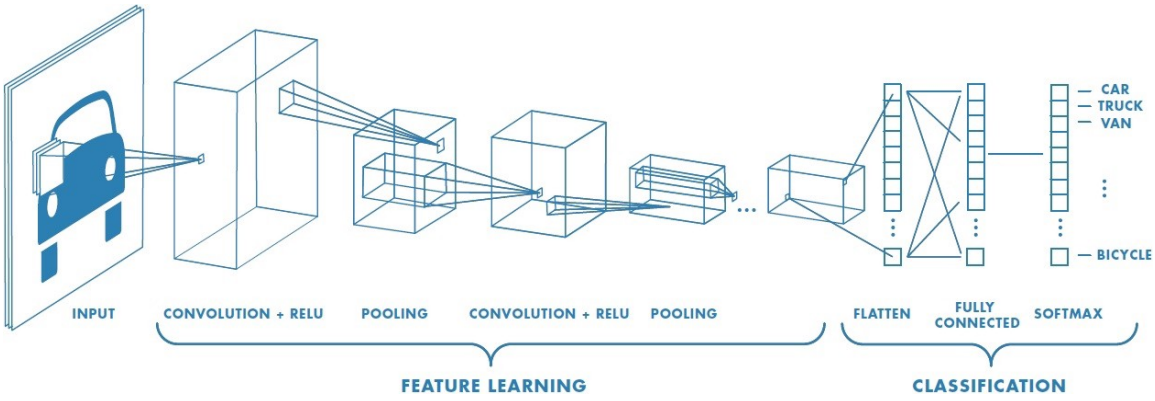
V této kapitole si představíme konvoluční neuronové sítě (Convolutional neural network, CNN). Budeme se věnovat základním vrstvám, které jsou nedílnou součástí CNN sítí. Dále si uvedeme bloky, které se skládají z uvedených vrstev. Pomocí těchto bloků můžeme efektivně zpracovávat data. Všechny informace v této kapitole budou důležitou prerekvizitou pro segmentační neuronové sítě, kterým se bude věnovat celá další kapitola.

1.1 Architektura

Na obrázku 1.1 je klasifikační CNN, která se rozděluje na dvě základní části:

První část CNN je extraktor vlastností (feature extractor, FE), který obsahuje konvoluční, ReLU a pooling (sdružující) vrstvy. Slouží k tomu, aby z obrázku získal ty nejvýraznější vlastnosti, které ideálně popisují daný obrázek. Tyto vlastnosti se nazývají mapa aktivací (feature map). S FE blízce souvisí pojem *podvzorkování*, který udává kolikrát menší je výsledná mapa aktivací oproti vstupnímu obrázku[1].

Druhá část je klasifikační. Skládá se z plně propojených vrstev pro klasifikaci výstupu z první části CNN. Je zakončená vrstvou Softmax, která nám procentuálně ohodnotí výstup z CNN.



Obrázek 1.1: Ukázka architektury konvoluční neuronové sítě [27].

[1]Pro CNN, která má na vstupu obrázek 224x224 a hodnotu podvzorkování 32, vznikne mapa aktivací 7x7.

1.2 Vrstvy

V této sekci si uvedeme nejdůležitější vrstvy CNN, které jsou primárně využívány k sestavení struktury výsledné sítě.

1.2.1 Konvoluční vrstva

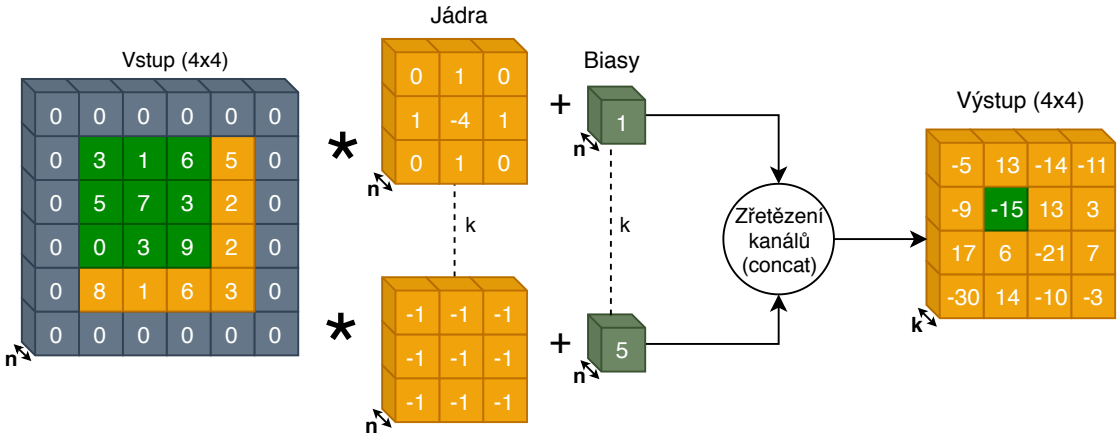
Jedná se o základní vrstvu CNN. Konvoluční vrstva využívá operace konvoluce, která je definovaná následovně:

$$g(x,y) = (\omega * f)(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} \omega(s,t) f(x-s, y-t) \tag{1.1}$$

kde:

- $g(x,y)$ je výstup po filtraci.
- ω je jádro konvoluce.

Na obrázku 1.2 je znázorněna ukázka celého výpočtu dopředného průchodu konvoluční vrstvou. Na vstupu se nachází N-kanálový obrázek daného rozměru. Konvoluční vrstva má definovaný počet a rozměr konvolučních jader. Počet kanálů v konvolučním jádru musí odpovídat počtu kanálů ve vstupním obrázku. Počet konvolučních jader odpovídá počtu kanálů výstupního obrázku. Po vykonání operace konvoluce se k výsledku připočítává i hodnota bias, která pomáhá při učení neuronové sítě tím, že posouvá aktivační funkci doprava nebo doleva. Nejen že konvoluční vrstva zvýrazňuje hrany v různých směrech, ale také dokáže měnit počet kanálů výstupu při použití konvolučního jádra rozměru 1x1, což je velmi užitečnou vlastností při optimalizaci sítě.



Obrázek 1.2: Ukázka konvoluční vrstvy.

Při trénování konvoluční vrstvy se aktualizují hodnoty jednotlivých jader. Cílem učení je nastavit konvoluční jádra tak, aby byly co nejvíce citlivá na hrany daných objektů, se kterými má síť umět pracovat.

V praxi se nahrazuje operace konvoluce maticovým násobení. Můžeme tedy psát, že konvoluční vrstva je definovaná takto:

$$Output = Input \cdot Kernel + Bias \tag{1.2}$$

Ve finále se ke konvoluční vrstvě přidává nelinearita. Je to z toho důvodu, abychom zamezili „spojení“ konvolučních operací neboli maticového násobení do jedné operace. Musíme mít tyto operace separované. Nejčastější funkce pro nelinearitu se používá ReLU (kapitola 1.2.4).

Parametry konvoluční vrstvy:

- **Stride** (krok): jedná se o počet pixelů, o který se má konvoluční jádro posouvat při výpočtu.
- **Padding** (zarovnání): udává velikost přidaného nulového okraje ke vstupu.
- **Kernel size**: udává velikost konvolučního jádra.

Při vytváření architektury konvoluční sítě je nutné vědět, jak se vypočítá velikost výstupní aktivační mapy po průchodu konvolucí. Následující výpočet 1.3 je navržený pro výpočet výstupní velikosti mapy aktivací.

$$out_x \times out_y = [floor(\frac{in_x + 2 * pad - kernel_x}{stride} + 1)] \times [floor(\frac{in_y + 2 * pad - kernel_y}{stride} + 1)] \tag{1.3}$$

Optimalizace konvoluční vrstvy:

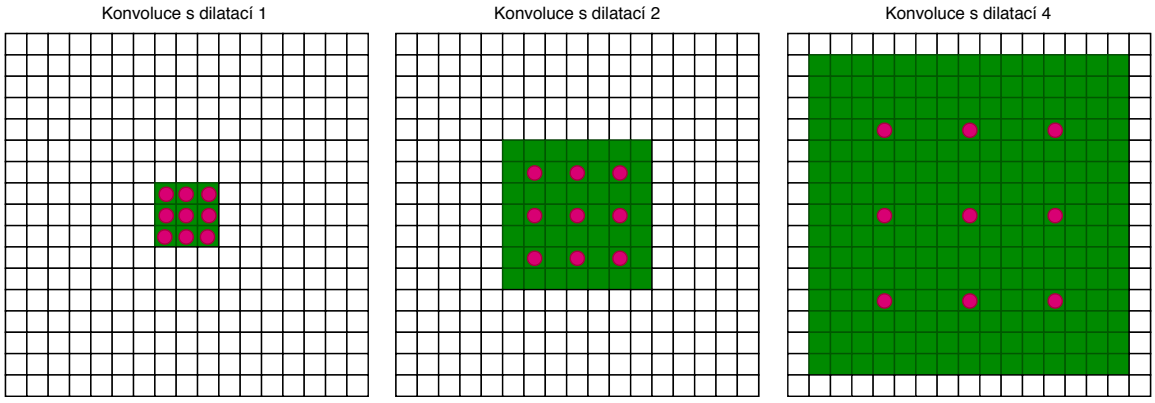
- **Používat místo konvolučních vrstev 5x5 vrstvy 3x3**. Když si spočítáme receptivní pole (kapitola 1.3) při použití dvou konvolučních vrstev 3x3, vyjde nám oblast 25x25. Stejná hodnota nám vyjde i při použití jedné konvoluční vrstvy 5x5. Rozdíl ale spočívá v počtu použitých vah v jednotlivých jádrech. Při variantě se dvěma vrstvami 3x3 použijeme 18 vah a ve druhé variantě 25 vah. Z toho vyplývá, že je vhodnější používat dvě 3x3 vrstvy místo jedné 5x5.
- **Používat 1x1 konvoluci k redukci nebo ke zvýšení počtu kanálů v mapě aktivací.** Příklad: Budeme mít residuální blok, do kterého vstupuje 256-ti kanálová mapa aktivací. Nejprve s pomocí konvoluce 1x1 s 64 jádry snížíme počet kanálu na 64, poté provedeme operaci konvoluce 3x3 a následně provedeme konvoluci 1x1 s 256 jádry, abychom získali zpět 256 kanálů. Konvoluce 3x3 s 64 kanály je výrazně rychlejší než konvoluce s 256 kanály. Touto optimalizací se nesnižuje úspěšnost, naopak se může zvýšit.
- **Použití konvoluce Depthwise Separable**. Jedná se o optimalizaci výpočtu konvoluce, která nám sníží počet násobení při jejím výpočtu. Násobení je pro CPU/GPU náročná operace, proto bychom se měli snažit počet těchto operací snížit.

1.2.2 Dilatační konvoluce

Dilatační konvoluce se používají pro rozšíření kontextu sítě. Tato konvoluce klade důraz na vzdálenější pixely v receptivním poli (kapitola 1.3).

Bylo prokázáno, že dilatační konvoluce snižují rozmazání v sémantických mapách segmentace a pracují s informacemi o vzdálenějších pixelech bez potřeby použití pooling vrstev [14].

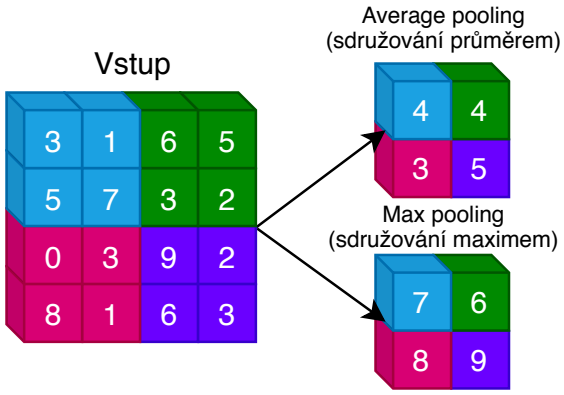
Účinnost dilatačních konvolucí závisí na dobré volbě parametru dilatační vzdálenosti (dilation rate), který udává roztažení jádra. Proto je důležité znát koncept podvzorkování. Není možné nastavit velkou dilatační vzdálenost na malou aktivační mapu. To by způsobilo degradaci konvoluce až na rozměr 1x1. Na obrázku 1.3 je ukázka konvoluce s různými dilatačními vzdálenostmi.



Obrázek 1.3: Dilatační konvoluce s dilatacemi 1, 2 a 4.

1.2.3 Pooling vrstva

Pooling vrstva slouží k redukci rozlišení mapy aktivací. Rozděluje se na dva základní typy. První typ se nazývá Max pooling a druhý Average pooling. Jejich název vyplývá z toho, zda vybírají z předem definovaného okolí maximální nebo průměrnou hodnotu.



Obrázek 1.4: Ukázka pooling vrstvy.

Na obrázku 1.4 jsou znázorněny oba základní typy pooling vrstev, které redukují rozlišení aktivační mapy o 75 %. Pooling vrstva nemá žádné váhy, které by mohla učit, má

pouze statické parametry.

Parametry pooling vrstvy:

- **Kernel size:** Specifikuje velikost pooling filtru.
- **Pool:** Výběr pooling metody (metoda vybírající maximální nebo průměrné hodnoty).
- **Pad (padding):** Udává kolik pixelů se má přidat z každé strany vstupu.
- **Stride:** Počet pixelů, o který se posouvá pooling jádro při výpočtu.

1.2.4 ReLU

ReLU (Rectified Linear Unit) - jedná se o nejpoužívanějších aktivační nelineární vrstvu, která provádí aktivaci neuronů. Aktivaci provádí tak, že všechny hodnoty menší než nula nastaví na nulu (deaktivuje je) a hodnotám větším než nula ponechá stejnou hodnotu. Tato vrstva se používá po konvoluční vrstvě, aby se mezi konvoluce zavedla nelinearita. Výpočet ReLU je znázorněn v rovnici 1.4.

$$f(x) = x^+ = max(0, x) \tag{1.4}$$

1.2.5 Plně propojená vrstva

Slouží ke klasifikaci mapy aktivací. V této vrstvě je každý neuron propojený s každým neuronem následující vrstvy. Obecně je plně propojená vrstva následována nelineární aktivační funkcí[2]. Poslední plně propojená vrstva obsahuje tolik neuronů kolik máme výstupních tříd. Následně se pomocí vrstvy Softmax získají pravděpodobnosti jednotlivých tříd.

Plně propojenou vrstvu lze nahradit konvoluční vrstvou s jádrem velikosti 1x1. Po nahrazení se neuronová síť začne nazývat plně konvoluční neuronová síť a tím přestane být závislá na rozlišení vstupního obrázku[3].

1.3 Receptivní pole

Receptivní pole (Receptive field) udává oblast vstupního prostoru, která ovlivňuje určitou jednotku v síti. Nejčastěji vyjadřuje kolik pixelů vstupního prostoru ovlivní 1 bod v mapě aktivací. Tato oblast by měla být dostatečně velká – zejména pro segmentační sítě. Je velmi pravděpodobné, že se na obrázku objeví větší plochy[4], které musí síť správně klasifikovat. S malým receptivním polem by kontext o okolních pixelech byl příliš malý a pravděpodobně by síť nefungovala správně.

Pro zvětšení této oblasti se používá několik metod: složení sítě z více konvolučních vrstev, podvzorkování (pooling, striding), použití dilatačních konvolucí, atd. V teorii platí, že když přidáme více konvolučnách, případně pooling vrstev, tak lineárně zvětšíme receptivní pole.

[2] Příkladem může být funkce *sigmoid*.
[3] Klasifikátory s plně propojenými vrstvami jsou závislé na rozlišení vstupního obrázku.
[4] Například silnice, obloha, budovy, apod.

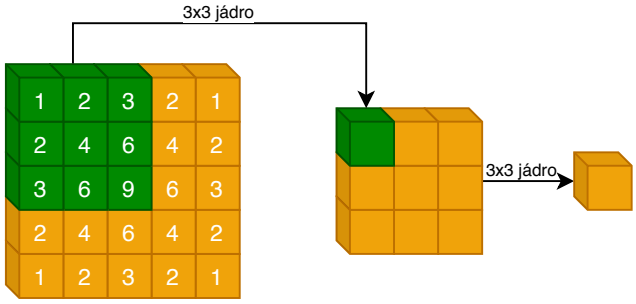
Teoretická velikost receptivního pole se určuje podle vzorečku:

$$r_{size} = n_{filters} * (k \ - 1) + 1 \tag{1.5}$$

kde:

- r_{size} je výsledná délka strany receptivního pole.
- $n_{filters}$ je počet filtrů (počet konvolučních, pooling vrstev).
- k je velikost filtrovacího jádra.

V praxi ovšem často nastává problém [25], že ne všechny pixely v receptivním poli přispívají stejně k výslednému bodu v aktivační mapě. V dopředném průchodu sítí mohou být centrální pixely receptivního pole propagovány různými cestami, tudíž jednotlivé pixely různě ovlivňují receptivní pole.



Obrázek 1.5: Použití pixelů v receptivním poli [25].

Na obrázku 1.5 jsou uvedeny dvě vrstvy využívající filtry 3x3. Číselné hodnoty v zeleném poli odpovídají počtu použití jednotlivých pixelů do výsledných hodnot. Je patrné, že pixely uprostřed byly použity několikrát častěji než pixely na okrajích, což může připomínat Gaussovo pravděpodobnostní rozložení. Je možné, že se receptivní pole bude ještě dynamicky měnit při trénování sítě [25]. Příkladem může být neuronová síť složená z pěti konvolučních vrstev s jádry 3x3, kde receptivní pole by mělo velikost 11x11 ($r_{size} = 5 * (3 - 1) + 1 = 11$).

1.4 Bloky

Bloky představují efektivní spojení elementárních vrstev do větších celků. Z nich je poté možné navrhnout vlastní strukturu neuronové sítě.

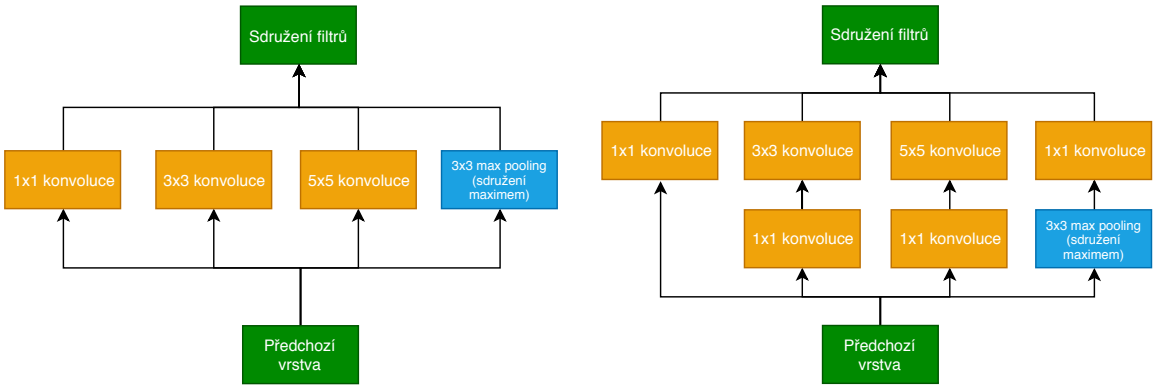
1.4.1 Inception blok

Inception bloky byly navrženy proto, aby neuronové sítě nerostly příliš do hloubky, ale spíše do šířky. Přidává paralelní větve, které obsahují různé typy filtrů. Příliš hluboké sítě trpí problémem přetrénování, případně neschopností se vůbec natrénovat[5]. Dále inception bloky řeší problém velice rozmanitých datových sad. Příkladem může být datová sada zvířat. Každé zvíře může být vyfoceno zblízka, zdálky, zboku, s různým pozadím atd. Pro CNN bez inception bloků může být klasifikace této datové sady obtížná, protože definujeme jednu třídu úplně odlišnými obrázky.

[5]Neplatí pro sítě s reziduálními bloky. U nich je hloubka teoreticky neomezená.

Konvoluční vrstvy s menšími jádry budou na obraz nahlížet spíše v lokálním měřítku zatímco konvoluční vrstvy s většími jádry[6] budou na obraz nahlížet spíše na globální úrovni.

Variant inception bloků je mnoho. Na obrázku 1.6 je uvedena základní struktura takového bloku. Je složený z paralelních konvolučních vrstev velikostí 1x1, 3x3 a 5x5 a jedné vrstvy max pooling velikosti 3x3. Všechny tyto vrstvy přijímají data z předchozí vrstvy a následně se všechny jejich výstupy spojí pomocí konkatenace.



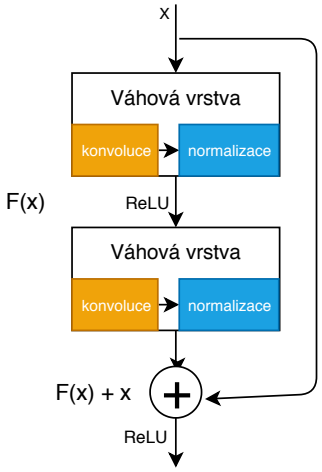
Obrázek 1.6: Základní inception bloky ze sítě inceptionv1 [42].

Tento typ bloku se využívá u sítí Inception v1-v4 [42][44] a sítě Inception-ResNet, která kombinuje inception a residuální bloky [41].

1.4.2 Residuální blok

Residuální bloky se používají při výstavbě hlubokých sítí. Klasické hluboké sítě stále snižují rozměr své mapy aktivací, čímž jsou limitovány. Residuální bloky tomuto jevu předcházejí pomocí takzvaného identického propojení. Jak je ilustrováno na obrázku 1.7, vstupní data X vstoupí do filtračních vrstev $F(x)$ a jejich výstup je konkatenován s daty před filtrací. Tím se neztrácí informace o výsledku předchozí vrstvy. Podobný koncept využívají bloky LSTM (Long Short-Term Memory) [38], které slouží v rekurentních sítích k uchování informace.

[6] Popř. více konvolučních vrstev v jedné větvi za sebou, což je více efektivní a má stejné vlastnosti.



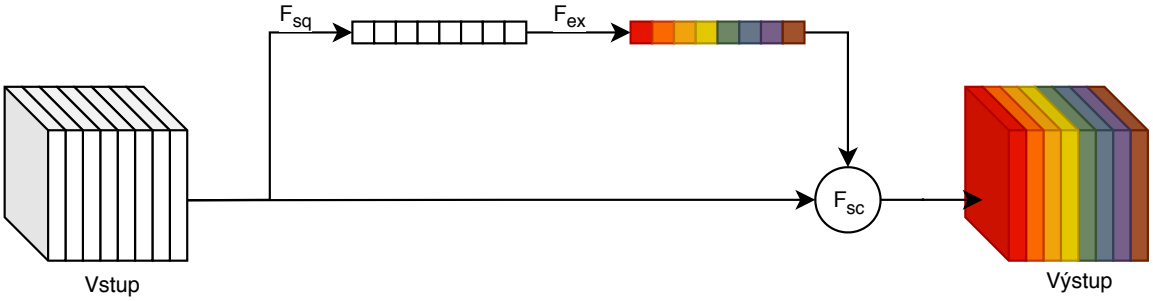
Obrázek 1.7: Ukázka reziduálního bloku.

Reziduální bloky se především využívají v sítích Resnet-X[7], kde X udává hloubku dané sítě [15].

1.4.3 Squeeze-and-Excitation blok

Squeeze-and-Excitation blok (SE blok) slouží k tomu, abychom mohli adaptivně odlišit kanály, které nesou důležité informace od těch, které nemají významnou informační hodnotu.

Do tohoto bloku vstupuje mapa aktivací a z každého kanálu je získaná průměrná hodnota pomocí vrstvy *GlobalAveragePooling* (F_{sq} v ukázce 1.8). Poté následují plně propojené vrstvy (dense layers), které jsou zakončené funkcí sigmoid. Tato funkce má obor hodnot definovaný jako interval $H = (0, 1)$ a v tomto kontextu určuje významnost jednotlivých kanálů. Funkce F_{sc} následně výstup z funkce sigmoid vynásobí s původní mapou aktivací [16].



Obrázek 1.8: Ukázka struktury SE bloku.

[7] Často používané reziduální sítě jsou: Resnet-34, Resnet-50, Resnet-101 nebo Resnet-152

Kapitola 2

Segmentační neuronové sítě

Segmentace scény založené na sémantické segmentaci je základní téma v počítačovém vidění. Cílem je přiřadit každý pixel scény do definované třídy. Sémantická segmentace poskytuje kompletní pochopení scény tak, že predikuje číslo třídy, lokaci a také tvar každého objektu[1] nebo každého typu pozadí[2] ve scéně. Výstupem je maska, ze které lze všechny tyto informace vyčíst. Sémantická segmentace se používá převážně pro automatické řízení aut, robotické snímání, popř. segmentace jader z buněk apod.

V této kapitole se seznámíme s konvolučními neuronovými sítěmi, které provádí sémantickou segmentaci celé scény (dále jen *SS-CNN*). V této kapitole si uvedeme vrstvy, které jsou pro sémantickou segmentaci typické. Dále se seznámíme s existujícími SS-CNN, které jsou specializované na segmentaci různých typů dat. Tyto informace poté poskytnou teoretický základ k tomu, abychom v dalších kapitolách byli schopni navrhnout vlastní SS-CNN, která bude segmentovat obrázky z ptačího pohledu s vynikající úspěšností.

2.1 Up-sampling vrstvy

Jedná se o vrstvy specifické pro SS-CNN. Vrstvy up-sampling se snaží zvětšit rozlišení mapy aktivací. Používá se v SS-CNN v části pro dekódování, abychom postupně zvětšovali mapu aktivací až na původní velikost vstupu.

Máme několik možností, jak zvětšit rozlišení mapy aktivací:

2.1.1 Metoda nejbližšího souseda

Tato metoda pouze rozkopíruje data do více buněk. Tuto metodu využijeme při modifikaci velikosti masek v datové sadě. Při modifikaci velikosti je důležité, aby se nedopočítávaly další barvy a zachovaly se původní.

V tabulkách 2.1 je ukázka aplikace metody nejbližšího souseda na vstup 2x2. Požadavkem je dvojnásobné zvětšení rozměru aktivační mapy.

[1] Např. auta, lidé, nákladní auta, motorky, apod.
[2] Např. obloha, infrastruktura, budovy, apod.

1	2
3	4

1	1	2	2
1	1	2	2
3	3	4	4
3	3	4	4

Tabulka 2.1: Příklad metody nejbližšího souseda.

2.1.2 Metoda bilineární interpolace

Jedná se o metodu, která počítá každý pixel jako váhový průměr z okolních pixelů v závislosti na vzdálenosti. Jedná se výpočetně náročnější metodu[3]. Pomocí této metody dokážeme lépe aproximovat data. V následujících tabulkách je uveden příklad 2.2.

1	2
3	4

1	1,25	1,75	**2**
1,5	1,75	2,25	2,5
2,5	2,75	3,25	3,5
3	3,25	3,75	**4**

Tabulka 2.2: Příklad metody bilineární interpolace.

2.1.3 Metoda transponované konvoluce

Pokud chceme síť naučit, jak zvětšovat rozlišení u aktivačních map adaptivně, můžeme použít transponovanou konvoluci. Tato transponovaná konvoluce má váhy, které se učí, stejně jako běžná konvoluce popsaná v kapitole 1.2.1.

Transponovaná konvoluce pracuje se stejnými parametry jako běžná konvoluce, jen s tím rozdílem, že mají opačný význam.

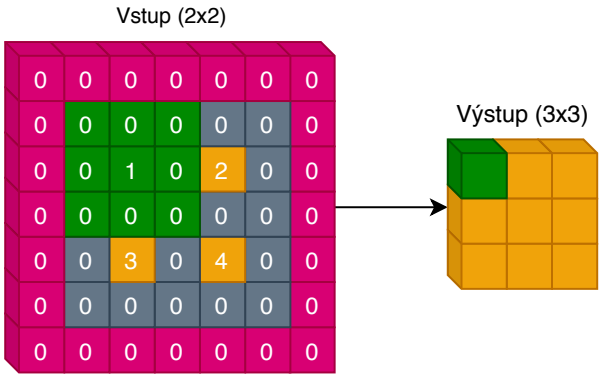
Parametry transponované konvoluce

- **Stride** definuje vzdálenost mezi čísly původní vstupní matice. Pokud nastavíme stride na hodnotu větší než 1, nově vzniklé pozice budou vyplněny nulami. Udává násobné zvětšení výsledného obrazu.
- **Padding** (zarovnání) udává velikost odebraného okraje od vstupu. Udává zmenšení výsledného obrazu.
- **Kernel size** udává velikost konvolučního jádra.

Běžná konvoluce je lineární operace a můžeme ji reprezentovat maticovým násobením. K dosažení transponované konvoluce musíme pouze transponovat jádro.

Na obrázku 2.1 je uvedena ukázka transponované konvoluce (dekonvoluce) s parametry Padding=1 (červená barva), Stride=1 (šedá barva), Kernel size=3x3 (zelená barva).

[3] V porovnání s metodou nejbližšího souseda



Obrázek 2.1: Ukázka transponované konvoluční vrstvy („dekonvoluce“).

Podle vzorečku 2.1 můžeme vypočítat výstupní velikost[4], která je užitečná při návrhu struktur sítí.

$$out_x \times out_y = [s * (in_x - 1) - 2 * pad + k_x] \times [s * (in_y - 1) - 2 * pad + k_y] \tag{2.1}$$

kde:

- s: hodnota parametru stride
- pad: hodnota parametru padding(zarovnání)
- in_x, in_y: vstupní rozlišení obrazu
- out_x, out_y: výstupní rozlišení obrazu

2.2 Moduly

V této podkapitole si uvedeme různé moduly, které se používají v SS-CNN pro kvalitnější segmentaci.

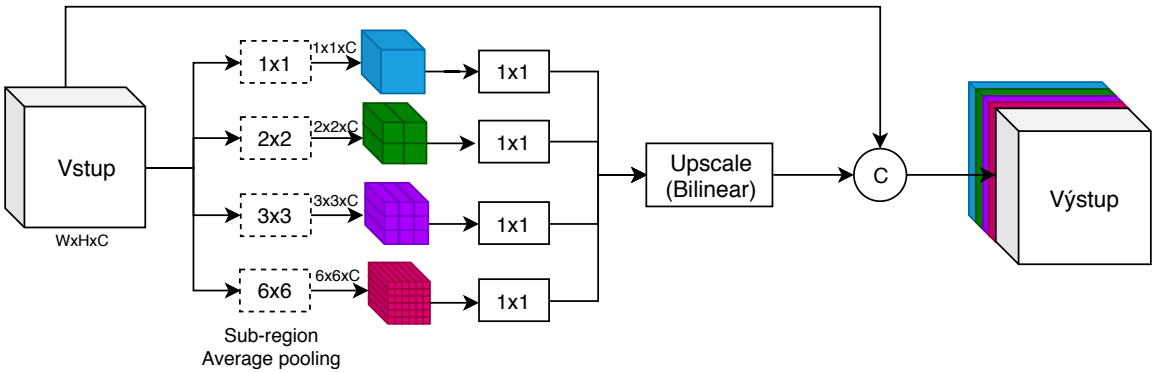
2.2.1 Pyramid Pooling modul

Pyramid pooling modul (PPM) se používá v případě, že potřebujeme k mapě aktivací přidat i globální informaci. Tato informace se získává pomocí globálních pooling vrstev.

Do PPM vstupuje mapa aktivací, která se rozkopíruje do čtyř větví. V každé větvi se celá mapa aktivací rozdělí na uvedený počet částí[5]. V každé části se vypočítá average pooling. Následně každá část vstupuje do 1x1 konvoluce, která pouze sníží počet kanálů mapy aktivací na čtvrtinu. Následně se pomocí bilineární interpolace každá dílčí mapa aktivací zvětší na původní rozměr vstupu a všechny dílčí aktivační mapy se sjednotí společně se vstupem. Celý postup je ilustrován na obrázku 2.2.

[4] [1*(2-1) - 1 * 1 + 3]×[1*(2-1) - 1 * 1 + 3] = 3×3

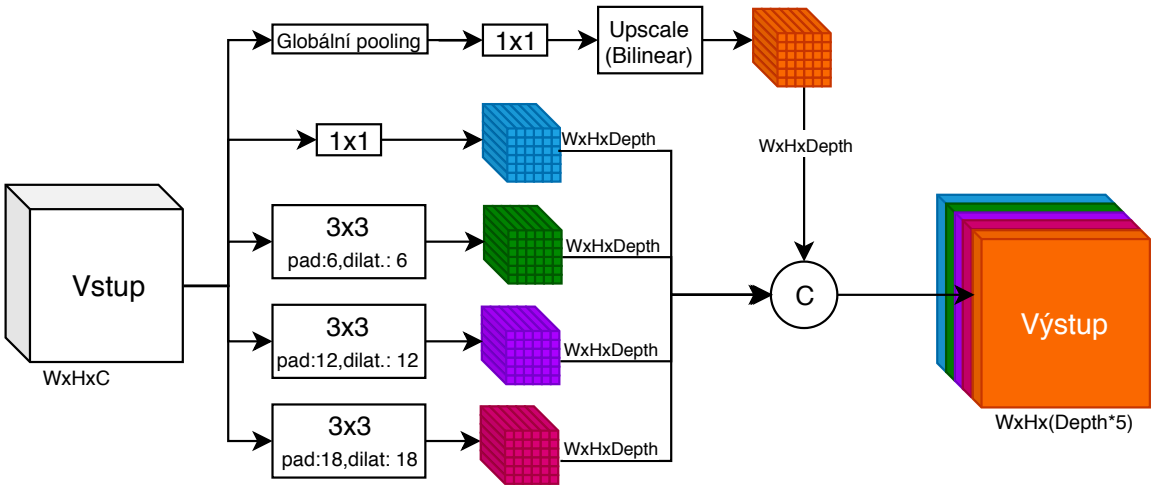
[5] Na 1x1, 2x2, 3x3 nebo na 6x6 částí.



Obrázek 2.2: Pyramid Pooling module.

2.2.2 Atrous Spatial Pyramid Pooling modul

Atrous Spatial Pyramid Pooling modul (ASPP) je součástí sítí DeepLab. Je velice podobný PPM modulu, jen s rozdílem, že využívá Atrous konvoluce, také známé jako dilatační konvoluce. Tyto konvoluce umožňují zachytit větší kontext. Tyto konvoluce jsou poté aplikovány s rozdílnými dilatačními faktory. Společně s dílčí mapou aktivací interpolovanou z globálního poolingu a vstupu se všechny mapy aktivací sjednotí do jednoho výstupu. Celý modul je zobrazen na obrázku 2.3.



Obrázek 2.3: Atrous Spatial Pyramid Pooling module.

2.2.3 Multi-scale context Aggregation (MSC)

Jedná se o kontextový modul založen na dilatačních konvolucích. Skládá se z sedmi sekvenčních konvolucí s jádry 3x3 s rostoucími dilatacemi. Poslední 1x1 konvoluce zajišťuje redukci kanálů na vstupní hodnotu C.

Základní kontextové moduly mají pouze C kanálů po celou dobu. Zatímco velký kontextový modul má zvětšující se počet kanálů až na 32 násobek vstupních kanálů. Tyto kanály se musí následně zredukovat poslední vrstvou. Celý kontextový model je znázorněn v tabulce 2.3. Tento modul se využívá v síti AdapNet, která byla navržena pro segmentaci venkovního prostředí [45].

Multi-scale context Aggregation

Vrstva	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Konvoluce	3x3	3x3	3x3	3x3	3x3	3x3	3x3	1x1
Dilatace	1	1	2	4	8	16	1	1
Receptivní pole	3x3	5x5	9x9	17x17	33x33	65x65	67x67	67x67
Typ: Základní	C	C	C	C	C	C	C	C
Typ: Velký	2C	2C	4C	8C	16C	32C	32C	C

Tabulka 2.3: Ukázka Multi-scale context Aggregation.

2.3 Encoder-Decoder model

Encoder-Decoder (E-D) je základní model SS-CNN sítí. Jeho základní myšlenka spočívá v extrakci mapy aktivací z obrázku a následné dekódování, čímž se získá maska vstupu. Tento model se skládá ze tří částí:

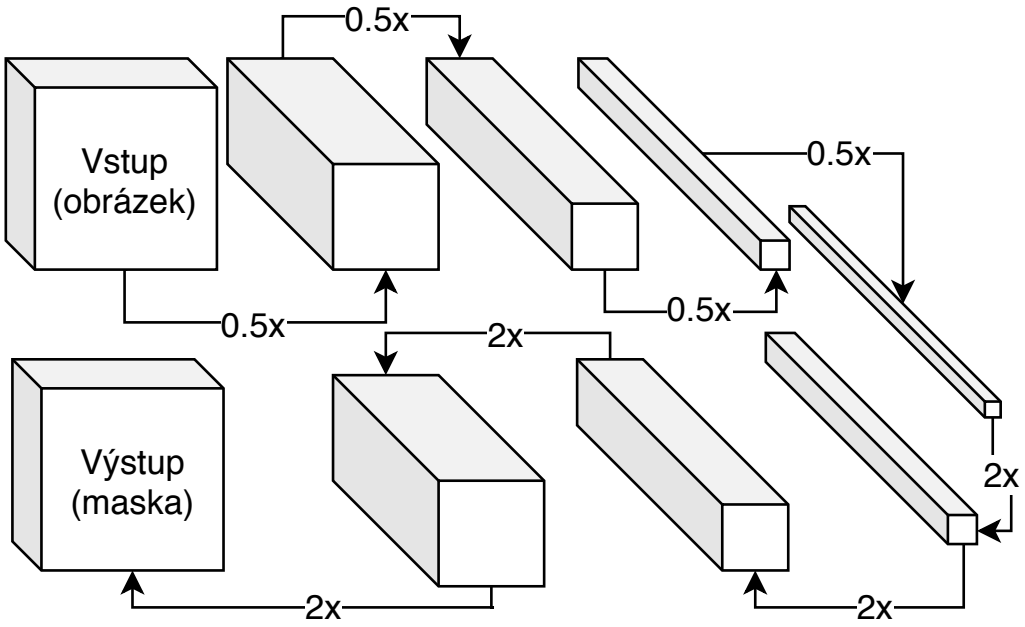
1. **Kódovací část** (encoder): Tato část má za úkol ze vstupního obrazu získat mapu aktivací, která ho reprezentuje. Neboli snižovat rozlišení vstupního obrazu, ale také zvyšovat počet kanálů. Lze využít známé klasifikační sítě (VGG, ResNet, Inception, NasNet [36]). Je možné také vytvořit vlastní síť, která se těmito sítěmi inspiruje a využije z nich jen část[6].

2. **Dekódovací část** (decoder): Tato část se snaží z mapy aktivací získat původní velikost obrázku. Využívá k tomu různé metody up-sampling popsané v 2.1 doprovázené konvolučními vrstvami.

3. **Klasifikační část**: Tato část se většinou skládá pouze z jedné vrstvy a to je vrstva Softmax, která z výstupu dekódovací části oklasifikuje každý pixel. Po klasifikaci získáme požadovanou masku vstupního obrazu.

Na obrázku 2.4 je znázorněna struktura E-D modelu.

[6]Takováto síť už nebude mít předtrénované váhy a bude potřeba jí trénovat od začátku (from Scratch).



Obrázek 2.4: Encoder-decoder model.

Prvním představitelem tohoto typu sítí byla síť U-Net, který se používá především pro biomedicínské segmentace obrazu [34]. Mezi další sítě typu Encoder-Decoder patří Segnet [2], TernausNet [18] apod.

Mezi modely vycházející z myšlenky E-D patří SPP (Spatial Pyramid Pooling) a EDAC (Encoder-Decoder with Atrous Conv). SPP se využívá v síti PSPNet [46]. Tento model obohatil kódovací část o PPM (Pyramid Pooling modul, 2.2.1) a dekódovací část omezil pouze na jednu bilineární interpolaci. Obodobné myšlenky se držel model EDAC, který obohatil E-D model o ASPP modul s dilatačními konvolucemi (kapitola 2.2.2). Tento model je využíván u sítí DeepLab [7].

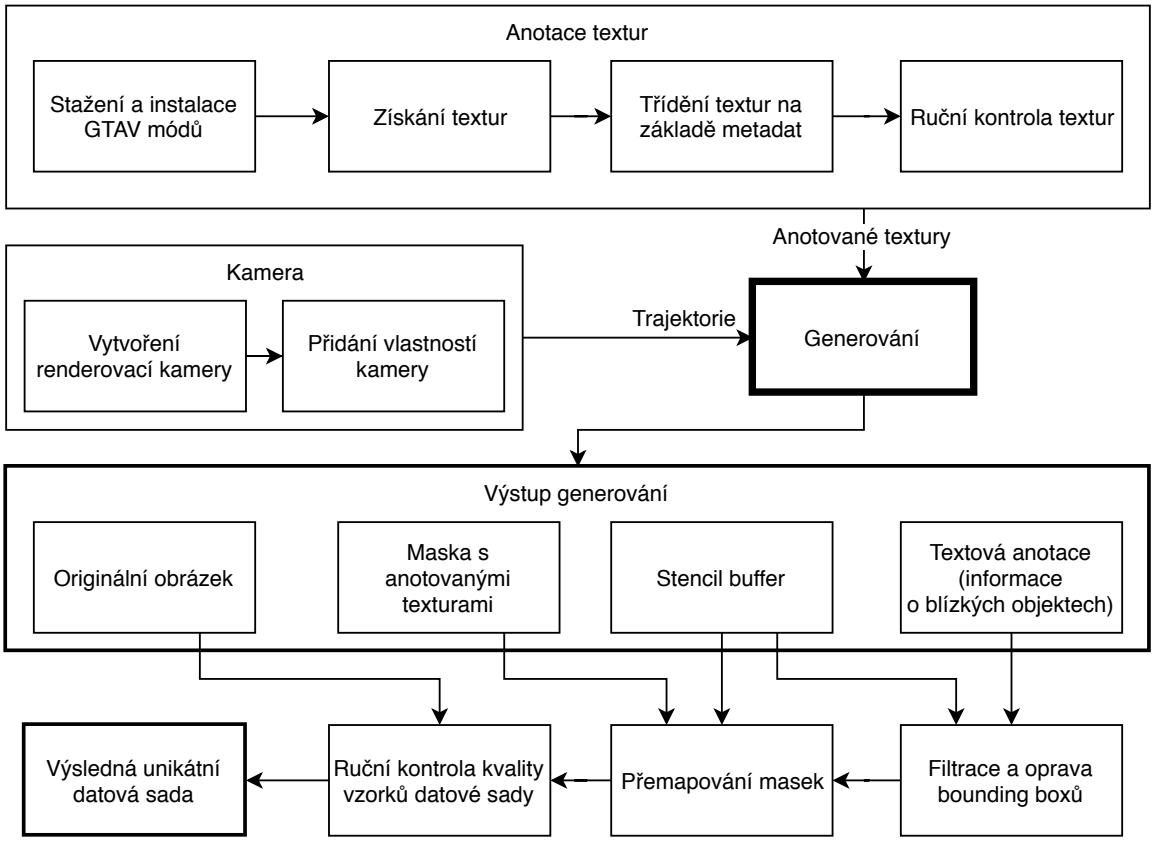
Kapitola 3

Generátor datové sady

Tato kapitola bude věnována generátoru umělé datové sady ze známé počítačové hry Grand Theft Auto V (GTAV). V rámci této diplomové práce byl vytvořen automatický nástroj pro vytváření umělé datové sady ze silničního prostředí. Tento nástroj uspokojuje potřebu získání specifické datové sady skládající se z topview[1] ptačích pohledů na dopravní situaci. Takto definovaná datová sada není volně dostupná a ruční anotování masek scén je velmi časově i finančně náročné. Vlastní generátor syntetických dat je ideálním řešením.

Výsledná datová sada je použitelná pro trénování a validaci segmentačních, detekčních nebo i klasifikačních neuronových sítí. Celý pracovní postup (workflow) je zobrazen na obrázku 3.1. Skládá se ze čtyř základních částí. První část se věnuje úpravě kvality a získání důležitých textur ze hry pro následnou anotaci. Druhá část má na starost logiku snímání herního světa, tzn. práci s vykreslovací kamerou. Ve třetí části se využijí první dvě části a začíná se vytvářet datová sada. V poslední čtvrté části se výstup z generování upravuje do výsledné datové sady. V této kapitole si tento uvedený proces podrobněji popíšeme a uvedeme si nástroje, které byly nedílnou součástí vývoje.

[1] Pohled z výšky svírající s vozovkou 90 stupňů.



Obrázek 3.1: Pracovní postup generování datové sady z hry GTAV.

3.1 Důležité nástroje

- **CodeWalker** [10] je nástroj pro získávání textur z GTAV. Umožňuje také načtení celého herního světa do 3D modelu a poskytuje možnost najít a získat jen specifické textury.
- **Script Hook V .NET** [4]. Jedná se o plugin pro hru GTAV, který umožňuje spouštění skriptů napsaných v .NET jazyce. Je to nástavba nad modifikací ScriptHookV.
- **OpenIV** [1] je nástroj umožňující instalaci rozšíření.

3.2 Anotace textur

V této kapitole si popíšeme postup anotace textur. Textury jsem anotoval proto, abych byl schopný získávat anotovanou masku pozadí scény. Jednalo se o anotaci infrastruktury (silnice, chodníky, cesty, koleje, apod.) a budov(střechy, zdi, okna, apod.). Postup anotace textur lze rozdělit do následujících bodů:

1. Stažení a nainstalování grafických rozšíření pro vyšší kvalitu obrazu. Jednalo se o rozšíření *LA_ROADS* a *NaturalVision Remastered*. Tyto rozšíření zvýšily grafickou kvalitu silnic a vozidel. Další alternativní rozšíření se jmenuje *REDUX*, které zvyšuje kvalitu textur u vozidel.

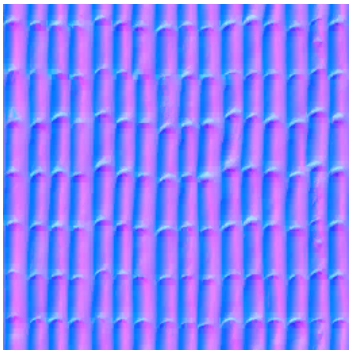
2. Získání všech herních textur pomocí nástroje CodeWalker, který jsem upravil, abych získával více informací o jednotlivých texturách. Jednalo se o informaci, která uváděla, ze kterého balíčku byla textura získaná. Počet všech získaných textur se pohyboval okolo 420 000.

3. Vytvoření nástroje, který roztřídí textury na základě klíčových slov v jejich názvech a v názvech balíčků, do kterých patří. Mezi základní klíčová slova se řadily: „build“, „buildings“, „road“, „ground“, „roof“, atd. Tím jsem byl schopný vyfiltrovat 90 % textur, které nebyly důležité. Ze zbývajících přibližně 40 000 textur jsem následně musel získat odrazové textury (specular textures)[2] a normálové textury (normal textures)[3]. Ukázka všech tří uvedených typů textur je na obrázcích 3.5.

4. Vyfiltrování duplicit ze získaných textur s pomocí dedikovaného nástroje. Duplicity jsou nalezeny pomocí výpočtu hash hodnoty souboru.

Textury zpravidla měly více klíčových slov, musel jsem tedy všechny kombinace manuálně projít a roztřídit do základních tříd (budovy, silnice). Nastávaly i situace, kdy jedna textura byla použitá jak pro silnici, tak i pro střechu budovy apod.

Obrázek 3.2: Textura střechy.



Obrázek 3.3: Normálová textura.

Obrázek 3.4: Odrazová textura.

Obrázek 3.5: Ukázka textury střechy, její normálové a odrazové mapy.

3.3 Práce s kamerou

Pomocí vlastní vykreslovací kamery je možné programově nastavovat přesné pozice a směry pohledů, které mají být vyfoceny a anotovány. Renderovací kameře jsem vytvořil tři režimy:

- Statická kamera snímající jedno místo z jednoho místa. Umožňuje natáčení dopravní situace bez pohybu kamery.

- Dynamická kamera pohybující se po kružnici v dané výšce okolo statického bodu. Tato kamera by se využila, kdyby bylo zapotřebí anotovat významné oblasti a jejich dění ze všech úhlů. Může se jednat např. o vytížené křižovatky simulující pohledy z více kamer.

[2] Odrazové textury označují barevnou změnu pixelů po nasvícení.
[3] Normálové textury označují zakřivení povrchu textury pomocí normálových vektorů.

- Dynamická kamera pohybující se po kružnici v dané výšce okolo dynamických bodů. Tento režim jsem využil v této práci. Kamera vždy nalezne blízké vozidlo, které fotí z různých úhlů a výšek a takto pokračuje k dalšímu vozidlu. Všechny vyfocené vozidla si zaznamenává do databáze, aby se zamezilo duplicitám v datové sadě.

3.4 Generování

Pro samotné generování je zapotřebí mít anotované textury a vytvořenou logiku kamery, která se pohybuje po herním světě. Generování se provádělo v rozlišení 4k, pro zachycení většího detailu. Implementaci tohoto modulu pro získání anotovaných textur a stencil bufferu mi poskytla firma Artin s.r.o.

Výstupem generování jsou čtyři soubory. Jedná se o:

- **Originální obrázek** zobrazuje scénu hry bez jakékoli úpravy.
- **Maska s anotovanými texturami** zobrazuje scénu, kde všechny anotované textury mají svou definovanou barvu.
- **Stencil buffer** obsahuje masku scény, ve které se nachází informace o všech dopravních prostředcích[4], lidi, oblohy a vegetace[5]
- **Textová anotace** obsahuje všechny důležité informace o blízkých objektech kamery. Jedná se o 2D a 3D bounding box objektu, v případě vozidla jeho přesný typ a model, jeho barvu, apod.

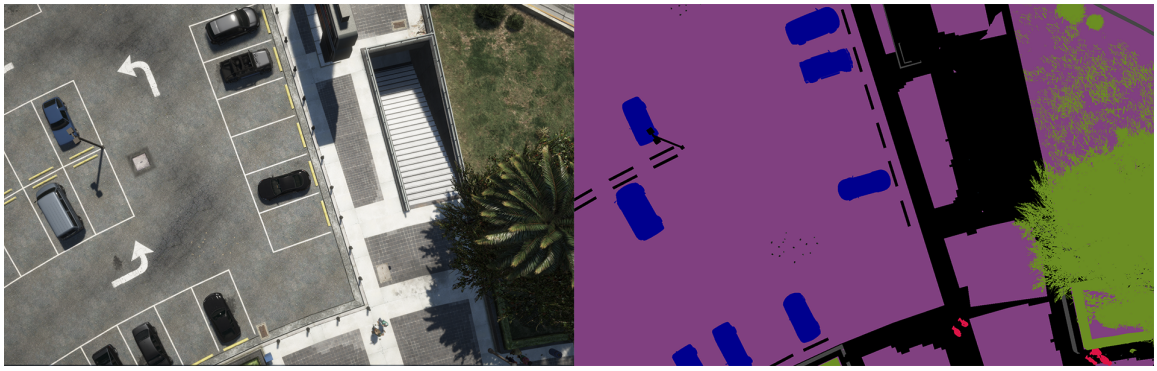
3.5 Zpracování výstupu generátoru

Nejprve se nad výstupem z generátoru musí spustit skript, který vyfiltruje objekty z textové anotace, které nejsou na scéně, např. jsou za kamerou, případně jsou za nějakou překážkou. Tato filtrace se provádí na základě stencil bufferu. Následně se musí vytvořit finální maska scény, která kombinuje informace ze stencil bufferu a z anotovaného snímku. Pro zaručení vysoké kvality datové sady je vhodné výsledné originální snímky a jeho masky ručně zkontrolovat a případně některé smazat. Takto jsem vytvořil datovou sadu, která čítá 4 100 snímků v rozlišení 4K[6].

3.6 Výstup generátoru

Výstupem celého procesu generování je originální obrázek a k němu korespondující maska. Na obrázku 4.1 je uvedena ukázka vlastní GTAV datové sady. Skládá se převážně z pohledů na zaparkovaná auta na parkovištích a na jedoucí auta po městě.

[4] auta, autobusy, kola, motorky,...
[5] stromy a keře
[6] Rozlišení 3840 x 2160.



Obrázek 3.6: Ukázka datové sady ze hry GTAV.

Kapitola 4

Datová sada

V této kapitole se seznámíme s datovou sadou, která bude v této práci využita k trénování, validaci a k testování segmentačních neuronových sítí. V úvodu této kapitoly si uvedeme obecné požadavky na datové sady a poté se podrobněji seznámíme s vlastní datovou sadou.

Datová sada je nejdůležitější částí při trénování neuronových sítí. Proto se v této práci velice dbá na to, aby byla datová sada co nejlépe připravena. Datová sada by měla obsahovat snímky scény z dopravního prostředí, k něm korespondující masky a celá datová sada by měla obsahovat soubor s popisky jednotlivých barev v masce. Datová sada by neměla obsahovat příliš mimotřídních chyb. Jedná se o nejednoznačnost určení tříd objektů v datové sadě.[1].

Na obrázku 4.1 je zobrazena ukázka datové sady, na které se nachází originální obrázek, maska a popisky jednotlivých barev v masce.

Obrázek 4.1: Ukázka datové sady.

4.1 Pravidla při vytváření datové sady

Při vytváření datové sady pro trénování neuronových sítí bychom měli mít na paměti těchto šest základních pravidel:

1. První pravidlo praví, že bychom měli získat co nejvíce volně dostupných nám relevantních datových sad. Následně bychom tyto datové sady měli sjednotit do jednotné struktury, se kterou se bude jednoduše pracovat.

[1] Na jednom snímku tvrdíme, že daný objekt je auto a na dalším tvrdíme, že ten stejný objekt je nákladní auto.

2. Druhé pravidlo praví, že bychom měli všechny datové sady zkontrolovat a odstranit případné chyby.

3. Třetí pravidlo doporučuje vytvořit analýzu výsledné datové sady. Může se jednat např. o statistiky ohledně zastoupení jednotlivých tříd v datové sadě. Obecně bychom se měli snažit, aby třídní zastoupení bylo vyvážené, neboli abychom měli vyváženou datovou sadu. Pokud tedy nechceme, aby nějaká třída měla větší prioritu než ostatní.

4. Čtvrté pravidlo doporučuje využít různé augmentace pro rozšíření datové sady. Díky augmentaci datové sady neuronová síť získá odolnost vůči nepříznivým podmínkám (přímé slunce, déšť, sníh, stín, apod.) nebo do určité míry vůči poškození obrazu (rozostření kamery, zamlžení čočky).

5. Páté pravidlo mluví o syntetických datových sadách. Toto pravidlo doporučuje doplnit si vlastní datovou sadu o uměle vygenerovaná data. Případně tyto data použít pro předtrénování sítí. Může se jednat o datovou sadu Synthia [35], případně datovou sadu vygenerovanou ze hry GTAV [32].

6. Šesté pravidlo uvádí minimální limit datové sady na 2 000 vzorků na jednu třídu, kde každý vzorek by měl být unikátní.

4.2 Vlastní datová sada

Vlastní datová sada se skládá ze šesti datových sad. Jedná se o datové sady *CamVid* [6], *CityScapes* [9], *KITTI* [11], *MAPILLARY-VISTAS* [29] a *GTA5-street* [33]. Tyto sady budou sloužit k předtrénování sítí. Jedná se o datové sady z pohledu jedoucího auta na silnici. My však budeme potřebovat datové sady, které budou získané z výšky 10 až 20 metrů a budou směrované na dopravní situaci. K řešení tohoto problému využijeme generátor datové sady ze hry GTAV (kapitola 3).

4.2.1 Úprava datové sady

V datové sadě Cityscapes a GTA5-street jsou často snímky foceny s kapotou auta ve spodní části snímku. Tato kapota patří vždy do třídy *void*. Obecně není vhodné, aby se na každém snímku datové sady objevoval stejný objekt. Síť by se na tento objekt přetrénovala a neuměla by správně klasifikovat ostatní objekty. Proto je vhodné tyto kapoty z datové sady odstranit.

Datové sady KITTI a CamVid jsem zvětšil na dvojnásobné rozlišení, aby je bylo možné zařadit do výsledné datové sady. Při trénování segmentačních sítí se vyžaduje, aby minimální velikost rozlišení datové sady byla 512x512.

4.2.2 Sjednocení datových sad

Před sjednocením datových sad je nutné vytvořit si seznam vlastních tříd, které budeme chtít segmentovat ze scény. V našem případě nás nejvíce budou zajímat všechny dopravní prostředky, lidé, budovy, silnice, atd. Každá datová sada má vlastní počet tříd, které segmentuje. Je nutné vytvořit mapování všech původních tříd na naše nově vytvořené. Poté vytvořit algoritmus, který všechny masky podle mapování překreslí. Důležité je už na začátku myslet na to, aby se zachovávaly původní data. Je to z toho důvodu, kdybychom se v průběhu experimentů rozhodli, že chceme změnit mapování tříd, tak abychom pouze změnili konfiguraci algoritmu a znova si datové sady vygenerovali.

V Tabulce 4.1 je uvedený souhrn všech dílčích datových sad, ze kterých se skládá výsledná datová sada použitá v této práci.

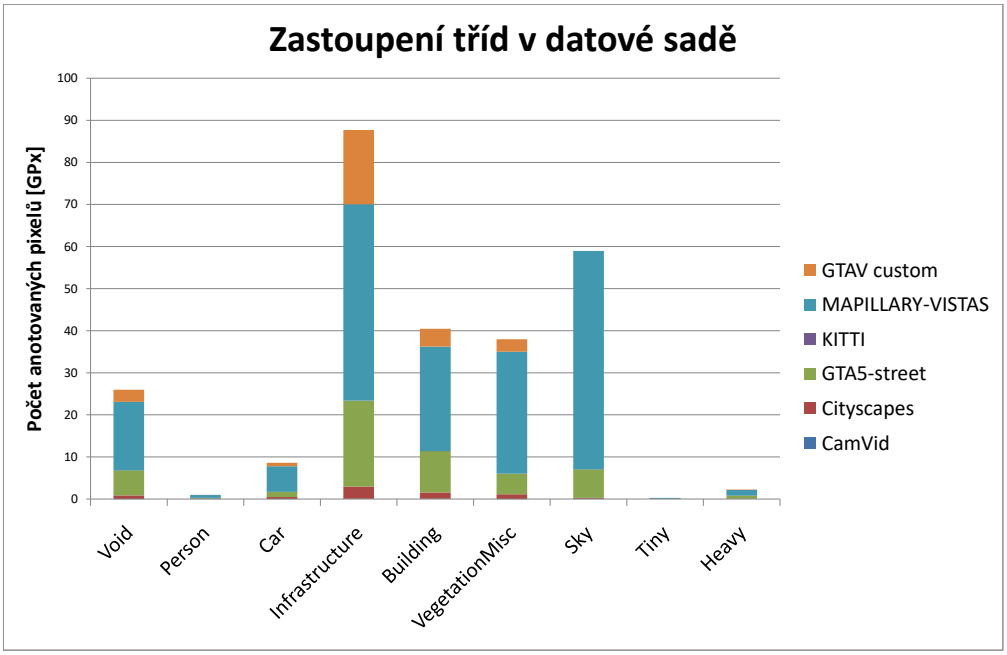
Název datové sady	Počet snímků	Průměrné rozlišení	Počet anotovaných pixelů [10^9 px]	Použití	Původní počet tříd
KITTI	200	2484x750	0,368	Předtrénování	29
CamVid	701	960x720	0,464	Předtrénování	12
CityScapes	3475	2048x1024	6,616	Předtrénování	30
GTAV-street	**24966**	1914x1052	45,158	Předtrénování	30
Mapillary-Vistas	20000	3264x2448	**168,425**	Předtrénování	**66**
GTAV-custom	4100	**3840x2160**	27,205	Dotrénování	9

Tabulka 4.1: Přehled dílčích datových sad.

4.2.3 Analýza datové sady

Pro analýzu datové sady byl vytvořen graf zastoupení jednotlivých tříd 4.2. Z grafu lze vyčíst, že převážná část anotovaných pixelů je pozadí scény, jako je např. infrastruktura, budovy, vegetace nebo obloha. Mezi nejlépe zastoupenou třídou z dopravních prostředků jsou auta. Z grafu také vyčteme, že $26,5 * 10^9$ pixelů spadá do třídy *void*, která obsahuje všechny ostatní objekty, které nepatří do žádné námi definované třídy. V ideálním případě by tato třída měla být co nejmenší.

Dále pomocí tohoto grafu můžeme zjistit, že datová sada není vyvážená v rámci popředí. Třída aut je několikanásobně větší než třída lidí, případně velkých (Heavy) nebo malých (Tiny) dopravních prostředků. S touto informací dále musíme pracovat při použití této datové sady pro trénování SS-CNN. Musíme tedy využít algoritmus, který nám tuto datovou sadu vybalancuje. Více o přípravě datové sady před vstupem do sítě je obsaženo v následující kapitole 5.



Obrázek 4.2: Graf zastoupení jednotlivých tříd ve vlastní datové sadě.

Mezi další důležitou statistickou informaci patří výpočet průměrné (mean) hodnoty celé datové sady určené pro trénování. Je důležitá proto, abychom byli schopni normalizovat vstupní data do intervalu $(-0.5, 0.5)$, které budou vstupovat do sítě. Tabulka 4.2 zobrazuje výsledné hodnoty.

Datová sada	**R složka**	**G složka**	**B složka**	**Počet pixelů [GPx]**
CamVid	98.683	102.061	104.130	0.484
Cityspaces	73.453	83.212	72.780	7.288
GTAV	112.892	111.799	108.391	50.211
KITTI	96.673	101.607	97.831	0.372
Mapillary-Vistas	106.919	116.970	119.888	176.828
GTAV custom	98.235	102.654	106.356	34.007
Celkem	**104,124**	**110,765**	**113,906**	**269,198**

Tabulka 4.2: Průměrné (mean) hodnoty barevných složek jednotlivých datových sad.

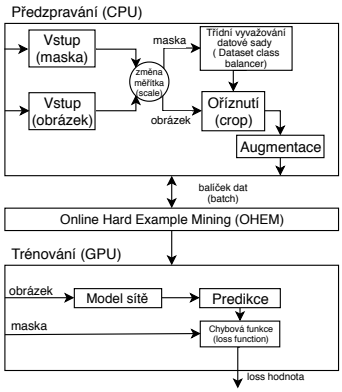
Kapitola 5

Systém pro trénování segmentačních neuronových sítí

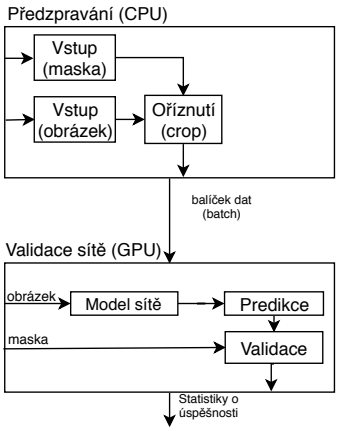
V této kapitole se seznámíme s implementovaným systémem pro optimální práci se segmentačními neuronovými sítěmi. Bude se jednat o systém podporující sestavení vlastních, případně přidání již existujících modelů sítí. Celý systém je napsaný ve skriptovacím jazyce Python s využitím frameworku Tensorflow a knihovny TensorRT.

5.1 Struktura systému

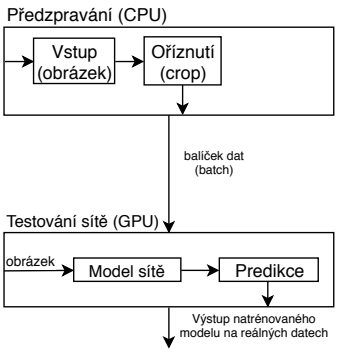
V této sekci si uvedeme základní strukturu systému. Bude se jednat o trénování, validaci a následné testování natrénovaného modelu. Celá architektura je ilustrována na obrázku 5.4.



Obrázek 5.1: Pracovní postup pro trénování sítě.



Obrázek 5.2: Pracovní postup pro validaci sítě.



Obrázek 5.3: Pracovní postup pro testování výsledného modelu sítě.

Obrázek 5.4: Struktury systému pro trénování, validaci a testování segmentačních neuronových sítí.

Pracovní postup pro **trénování modelu** (obrázek 5.1) začíná na CPU vstupem vzorku z datové sady, skládající se z obrázku scény a jeho masky, do bloku, který této dvojici upraví měřítko, neboli zmenší/zvětší vstupní obrázek a jeho masku (více v sekci 5.4). Následně maska vstupuje do bloku vyvažující třídy datové sady (class balancer, 5.5), který vybere

ideální souřadnice výřezu ze vstupního obrázku na základě masky. Na základě informace o ideálních souřadnicích výřezu se maska i obrázek vyříznou. Tyto výřezy vstupují do augmentačního bloku, ve kterém je provedena mírná deformace dat (kapitola 5.3). Výsledný obrázek a jeho maska se zabalí do balíčku (batch[1]). Tento batch vstupuje do bloku OHEM, který rozhodne, zda daný batch obsahuje dostatek nových informací pro síť, ze kterých by se mohla něco naučit (více v sekci 5.6). V případě, že data obsahují nové informace, vstupují do modelu sítě. Model sítě přijímá obrázek scény, ze kterého se pokusí získat nejpřesnější masku. Tato predikce vstupuje společně s originální maskou (ground truth) do chybové funkce (loss function). Chybová funkce vypočítá chybu predikce. Tuto chybu optimalizační algoritmus propaguje zpětným průchodem sítí a aktualizuje váhy sítě.

Pomocí **validace modelu** jsme schopní získat jeho metriky. Jedná se např. o pixelovou přesnost predikce(Accuracy, Precision), IoU, Recall apod. Do tohoto pracovního postupu musí vstupovat obrázek společně s maskou, stejně jako to bylo u trénování modelu. Rozdíl spočívá v tom, že se neprovádí žádné algoritmy vybírající ideální výřez nebo augmentace. Pouze se ze vstupních dat získá náhodný výřez obrazu i masky a tato dvojice vstupuje do sítě. Pomocí predikce ze sítě a masky jsme následně schopní vypočítat dané metriky sítě. Datová sada pro validaci musí obsahovat dostatek vzorků pro jednoznačné prokázání daných metrik.

Poslední pracovní postup pro **testování modelu** spočívá pouze ve vyhodnocení natrénované sítě nad reálnými daty, na které bude síť použita. Z toho vyplývá, že do tohoto pracovního postupu nevstupuje maska dané scény.

5.1.1 Zpracování většího rozlišení vstupu

V případě, že potřebujeme segmentovat větší snímky, než nám dovolí paměť na GPU, je nutné tyto snímky rozsekat na menší části. Snímky rozdělíme podle mřížky daného rozměru. Musíme snímky dělit s překrytím z důvodu toho, že obecně neuronové sítě zpracovávající obraz nedávají kvalitní výstupy z okrajů obrazu. Tyto kraje musíme při závěrečném skládání výsledné masky obrazu vymazat.

5.2 Subsystém pro přípravu sítí

Každá neuronová síť se skládá ze základních čtyř částí. Jedná se o strukturu (graf) sítě, loss (chybovou) funkci, optimalizační algoritmus a datovou sadu. První tři části systém umožňuje jednoduše přidávat a měnit. Pro všechny tři části byl využit návrhový vzor Builder.

Struktura sítě: Pokud se struktura sítě pro kódování skládá z feature extractoru (Resnet, MobileNet,...), tak je nutné přidat koncové body (endpoints) přes které bude napojený na zbytek modelu. Koncové body se zpravidla přidávají na místo za pooling vrstvu, která snižuje rozlišení mapy aktivací. Ukázka tohoto napojení je na obrázku návrhu vlastní architektury sítě 6.1.

Optimalizační algoritmus: Systém umožňuje nastavení dvou optimalizačních algoritmů. Jedná se o RMSProb a SGD (Stochastic Gradient Descent).

Loss funkce: Systém umožňuje přidání libovolné loss funkce. Nejčastější loss funkce pro segmentační je Cross Entropy loss [26], případně Focal loss [23].

[1] 4-dimenzionální pole (Batch_size, Height, Width, Channels)

5.3 Augmentace

Augmentace datové sady slouží k jejímu rozšíření o nové vzorky. Augmentace pomáhá k tomu, aby neuronové sítě byly invariantní vůči modifikacím objektů, které klasifikují, detekují, či segmentují ze scény. V této sekci je uvedeno několik augmentací, které je možné použít k rozšíření datové sady.

Existují i specifické augmentace, které se využívají pouze pro datové sady určené k trénování klasifikátorů nebo detektorů. Tyto augmentace vynecháme a budeme se soustředit pouze na augmentace vhodné pro datové sady určené k trénování neuronových sítí sloužící k segmentaci obrazu.

- **JPEG komprese:** Jedná se o ztrátovou kompresi, která dokáže změnit obrázek na úrovni dat, ale pro člověka se zdá vizuálně velice podobný.
- **Modifikace velikosti:** Obraz je možné augmentovat pomocí zmenšení a následného zvětšení obrazu do původní velikosti[2]. Při modifikacích velikostí je nutné zachovat poměr stran a využít Nearest Neighbor kompresi. Jedná se o kompresi, která se snaží zachovat největší vizuální podobnost. Hodnoty pixelů se však změní a pro neuronovou síť jsou to nová data.
- **Zaostření / Rozostření:** Jedná se o augmentaci, která zvýrazňuje nebo rozmazává hrany všech objektů v obraze.
- **Zrcadlové otočení obrazu:** Jedná se o nejjednodušší, ale i o efektivní augmentaci, pomocí které je možné jednoduše zdvojnásobit velikost datové sady.
- **Změna počasí:** Do obrazu můžeme přidat mlhu, déšť, padající sněhové vločky.
- **Rybí oko.** Touto augmentací se budeme snažit co nejvíce napodobit výstupy z reálných kamer sledující dopravní situaci.
- **Gamma filter**. Pomocí gamma filtru můžeme měnit světlost a intenzitu obrazu.
- **Změna teploty barvy** vyjádřená v jednotce Kelvin, je jednotkou pro barvu světla, která osvětluje náš objekt. Čím vyšší je teplota barev, tím je obraz modřejší a čím je teplota barev nižší, tím je obraz červenější.
- **Přidání šumu.** Přidáním náhodného šumu do obrazu učiníme neuronovou síť odolnou vůči zrnění obrazu, či nedokonalosti kamery.
- **Stereografická projekce**. Jedná se o mapovací funkci, která promítá povrch koule na rovnou plochu. Tato projekce deformuje hrany objektů. Tuto projekci musíme aplikovat na obraz i na jeho masku.
- **Změna jasu obrazu.** Jedná se o pouhé přičtení konstantní hodnoty v rozmezí (-255,255) ke každému pixelu obrazu.
- **Změna odstínu (HUE)**. Pomocí této augmentace změníme odstín barev v obraze.
- **Generative Adversarial Networks (GAN):** V případě, že máme velice omezenou datovou sadu a potřebujeme ji rozšířit je vhodné použít GAN. Jedná se o neuronovu síť, která dokáže generovat obrázky z originálního snímku, které lze použít pro trénování neuronové sítě [5].

[2] Využívá se bilineární nebo bikubická interpolace.

5.4 Předzpracování dat

Předzpracování dat zajišťuje kvalitnější výsledky. Je nutné každý obrázek, který bude určen k trénování, testování nebo k validaci neuronové sítě, předzpracovat stejně. V případě trénování provedeme nejprve nad obrázkem augmentaci.

- V případě, že bude nutné předzpracovat obraz, který je minimálně dvojnásobně větší než požadované vstupní rozlišení sítě, je nutné ho rozdělit do několika menších celků s překryvem a jednotlivé části vložit do sítě. Po získání jednotlivých masek je nutné masky složit do sebe a získat výslednou masku velikosti jako původní obraz.
- V případě, že obraz určený k segmentaci má maximálně dvojnásobnou velikost, je možné pouze vstup zmenšit na požadovanou velikost a výslednou masku zvětšit na původní velikost. Při zmenšování a zvětšování obrazu se musí zachovat poměr stran. Takto je možné se vyhnout velké deformaci objektů, na které by neuronová síť nemusela být připravena.
- V případě, že vstupní obraz má menší rozlišení než požadované vstupní rozlišení sítě, je nutné ho zvětšit na požadovanou velikost.

Výběr typu předzpracování je možné si zvolit podle toho, jak kvalitní obraz bude neuronová síť dostávat, a zda je možné, aby si ho zmenšila nebo je nutné obraz rozdělovat na podvýřezy. V případě podvýřezů je obraz získán z velké vzdálenosti s vysokou kvalitou obrazu. Při zmenšení bychom ztratili malé objekty, např. auta, která bychom chtěli segmentovat. Musíme se obecně snažit, aby byl neuronové síti poskytnut obraz ze stejné vzdálenosti s podobnou velikostí objektů. V případě malého a nekvalitního obrazu je nutné jej zvětšit.

Online předzpracování - CPU online generuje vzorky, které předává síti na učení - augmentace, scale, výběr vzorků atd. Při rychlém CPU se neomezí rychlost trénování. Obrovskou výhodou je, že pouze v kódu měním postup předzpracování a nemusím manipulovat s velkými soubory jako je u offline předzpracování.

Offline předzpracování - Před začátkem trénování je připraven LMDB(Lightning Memory-Mapped Database) nebo tfrecord, ve kterých jsou zapsaná surová data, ze kterých se síť bude učit. Výhodou je vyšší rychlost trénování, protože CPU není vytíženo předzpracováním. CPU pouze předává data z databáze. Nevýhoda vzniká při trénování z rozsáhlých datových sad. Je vytvořena velká databáze datové sady a je obtížné manipulovat či modifikovat tuto databázi v průběhu času.

5.5 Dataset class balancer

Dataset class balancer („vyvažovač“ tříd datové sady) pomocí masky vybere oblast, která má pro nás největší informační hodnotu. Jedná se o oblast, ve které se vyskytuje velké množství tříd objektů, které síť dlouho nevstoupily do sítě. Není vhodné síti předávat stále vzorky pozadí[3], protože by se málo trénovala na popředí[4]. Po získání souřadnic ideální oblasti, se tato oblast vyřízne z masky i ze vstupního obrazu.

[3] budovy, obloha, silnice, vegetace
[4] auta, nákladní auta, motorky, lidé

5.6 Online Hard Example Mining

Online Hard Example Mining (OHEM) je technika učení neuronových sítí, která zabraňuje trénování modelu sítě na vzorcích, které v daný moment učení pro síť nemají žádnou novou informační hodnotu. OHEM funguje tak, že nejprve nechám data dopředným průchodem sítí (forward propagation), tím získám loss hodnotu. Pokud tato loss hodnota je vysoká, tak to znamená, že je na daném snímku hodně nových informací pro síť, tím pádem nechám aktualizovat váhy zpětným průchodem sítí. V případě, že loss hodnota je příliš nízká, což znamená to, že síť tyto data již umí správně klasifikovat, tak se tyto data zahazují a zpětný průchod (back propagation)[5] sítí přeskakují. Tato technika velice přispívá k tomu, aby se síť nepřetrénovala.

V případě, že se v datové sadě nachází malé procento vzorků s vyšší informační hodnotou pro trénovanou síť, je možné trénování předčasně zastavit. Tento fenomén se nazývá early stopping. Obecně máme nastavený pevný počet epoch trénování a díky této technice můžeme skončit trénování dynamicky podle vstupních dat.

Mezi omezení OHEM algoritmu patří fakt, že se tento algoritmus musí spouštět až po prvních pár epochách, protože při počátečních iteracích loss hodnota velice kolísá a OHEM by vybíral špatná data k trénování.

[5] Back propagation aktualizuje váhy trénované sítě pomocí optimalizačního algoritmu.

Kapitola 6

Vlastní segmentační neuronová síť

V této kapitole si popíšeme vlastní segmentační neuronovou síť, kterou jsem v rámci této diplomové práce navrhl. Proto, aby bylo možné vlastní neuronovou síť vytvořit, je nutné navrhnout strukturu sítě, zvolit loss funkci a optimalizační algoritmus. Obecně je nutné ještě vytvořit datovou sadu, ale tuto máme již připravenou.

6.1 Návrh sítě

Navrženou síť jsem pojmenoval SeparNet(Squeeze-and-Excitation, Ppm, Aspp, Refine block) podle základních bloků, ze kterých je složena. Tato síť je typu Encoder-Decoder (2.4).

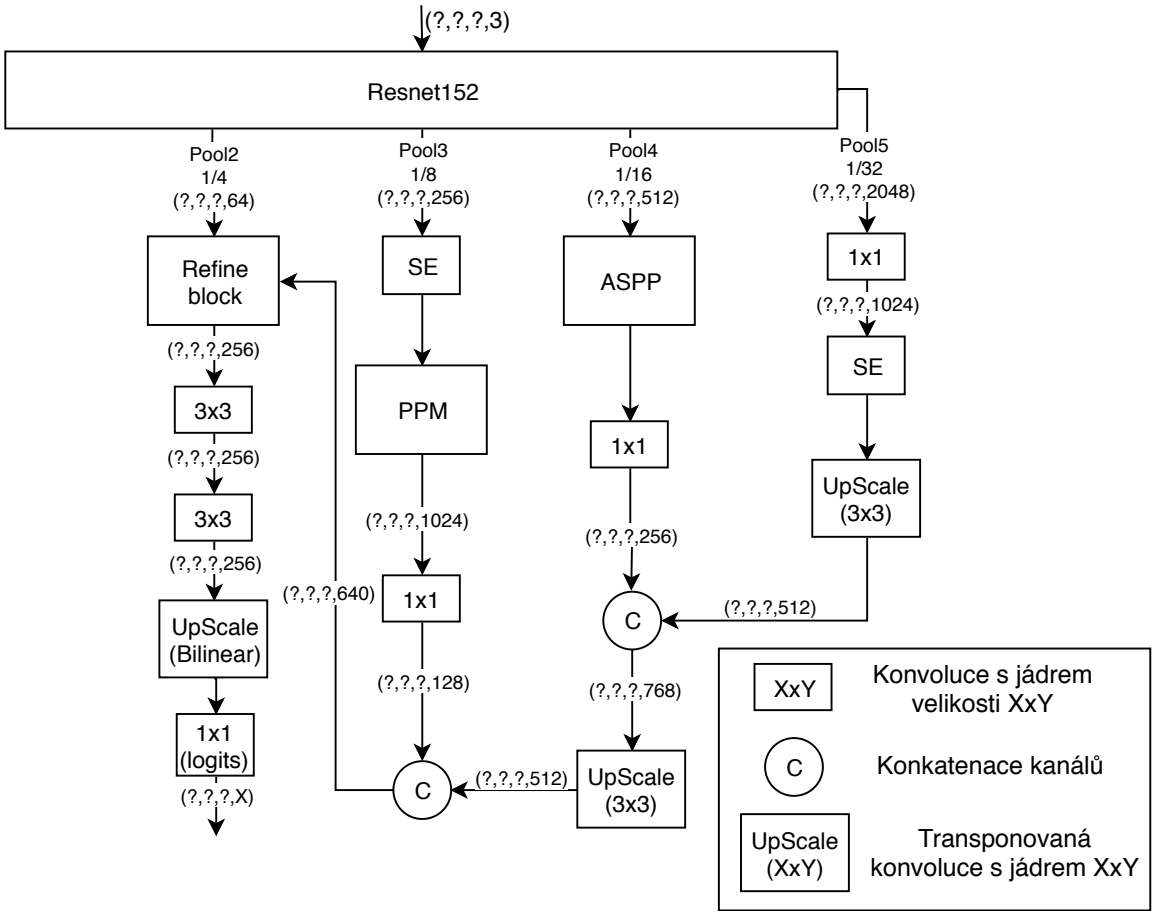
Při návrhu sítě jsem využil silné části dosavadních state-of-the-art architektur, sjednotil je do jedné sítě a odstranil nedokonalosti, kterých se jednotlivé architektury dopouštěly. Jedná se o architektury RefineNet [22] s využitím Refine bloku, který sjednocuje dva proudy s rozdílným rozlišením. Dále se jedná o PSPNet [46] s využitím PPM(2.2.1). Následně DeepLabV3+ [8] s využitím modulu ASPP (2.2.2) využívající dilatační konvoluce pro získání většího kontextu. A poslední významný blok využitý v návrhu je Squeeze and Excitation block (SE), který klasifikuje jednotlivé kanály mapy aktivací (1.4.3).

Při návrhu sítě je důležité hledět na počty kanálů jednotlivých map aktivací. Pro redukci, případně expanzi kanálů v mapě aktivací se používá konvoluce s jádrem velikosti 1x1. Kdyby se tyto počty nehlídaly, tak by se mohlo stát, že se při trénování síť nedokáže celá načíst do grafické paměti a způsobilo by to buď vysoké zpomalení trénování nebo pád celé aplikace.

Na rozdíl od některých stávajících sítí (PSPNet, DeepLabV3(+), DenseASPP, FRRN-A, apod.) nepoužívají mapu aktivací podvzorkovanou 32x. Tato mapa aktivací udržuje informace o velkých objektech minimálně 32x32 v původním obraze. S těmito informacemi je vhodné v naší síti pracovat.

Rodina sítí DeepLab využívá bilineární interpolaci pro zvětšování rozlišení mapy aktivací 4x. V navržené síti využívám pro adaptivní zvětšování rozlišení transponovanou konvoluci s jádrem 3x3, která zvětší mapu aktivací vždy na dvojnásobnou velikost (2.1).

Navržená síť je plně konvoluční, což nám umožňuje síti předávat libovolně velké obrázky scén, které má segmentovat. Jediné omezení je paměť grafické karty. Celý návrh sítě je uvedený na obrázku 6.1.



Obrázek 6.1: Struktura vlastní sítě SeparNet.

6.2 Loss funkce

Loss funkci pro uvedenou síť jsem zvolil pixel-wise cross entropy loss, běžně používanou pro segmentační neuronové sítě. Tato funkce počítá každý pixel zvlášť. Porovnává každý pixel predikce s ground truth. Aktivuje se na každý pixel zvlášť a následně se loss hodnota ze všech pixelů zprůměruje.

$$pixel\_loss = -\sum_{classes} y_{true} * log(y_{pred}) \tag{6.1}$$

Výsledná loss hodnota je definovaná takto:

$$cross\_entropy\_loss = \frac{1}{w * h} \sum_{x}^{w} \sum_{y}^{h} pixel\_loss_{x,y} \tag{6.2}$$

6.3 Optimalizační algoritmus

Optimalizační algoritmus jsem pro předtrénování zvolil RMSProb, který je velice rychlý v nalezení globálního minima funkce. Pro dotrénování jsem použil SGD, který se řadí mezi pomalejší optimalizační, ale přesnější algoritmy.

Kapitola 7

Trénování

V této kapitole se zaměříme na trénování sítí. Pomocí prvotního experimentu si vybereme state-of-the-art architektury, které budeme trénovat. Zvolené sítě nejprve natrénujeme na sjednocené datové sadě z pohledů řidiče auta. Na základě výsledků experimentů vybereme sítě, které dotrénujeme na vlastní datové sadě vygenerované ze hry GTAV. K těmto architekturám přidáme i vlastní navrženou síť SeparNet.

Celý čas budeme sítě trénovat metodou supervised learning (trénování s učitelem), ve kterém síti předáváme dvojice obraz-maska[1]. V průběhu trénování se budou ukládat checkpointy (kontrolní body) sítí. Tyto checkpointy budeme analyzovat a zjišťovat kvalitu těchto záznamů. Výstupem trénování jsou nejlepší checkpointy od každého zvoleného modelu sítě.

7.1 Výběr sítě

Aby bylo možné moudře se rozhodnout, která state-of-the-art architektura sítě bude ideální, je vhodné tyto sítě podrobit testům. Tabulka 7.1 zobrazuje 46 implementovaných architektur sítí, jejich počty učících parametrů a průměrný čas vyhodnocení 1000 snímků velikosti 512x512x3 na grafické kartě GeForce RTX 2080 Ti. Některé architektury vyžadují extrakci mapy aktivací, kterou má na starost feature extractor. V systému je implementovaný Resnet s 50, 101 a 152 vrstvami[15], MobileNetV2 [37], SEResNeXt s 50 a 101 vrstvami [17].

State-of-the-art architektury jsem vybral následující:

- **FC-DenseNet56** z hlediska nízkého počtu parametrů sítě, z experimentálního hlediska, zda takto malá síť bude schopná pojmout velké množství informací,
- **FC-DenseNet103** z důvodu vysoce kvalitních výsledků uváděných v článku [19],
- **RefineNet s ResNet152** [28] z hlediska vysokého počtu parametrů,
- **DeepLabV3+ s ResNet152** z hlediska rychlosti a specializaci na segmentaci dopravní scény [8],
- **FRRN-A a FRRN-B** na základě kvalitních výsledků uvedených v článku [31].

[1] Existuje také unsupervised learning [12] (trénování bez učitele) a reinforcement learning [21], ve kterém je znám pouze výsledek, ale není známo, jak se k němu dopracovat (časté použití v herním průmyslu)

Název modelu	Feature extractor	Počet parametrů (M)	Průměrný čas (ms)
FC-DenseNet56 [19]	-	1,4	70,2
FC-DenseNet67 [19]	-	3,4	105,5
FC-DenseNet103 [19]	-	9,2	107,9
AdapNet [45]	-	21,1	20,5
FRRN-A [31]	-	17,7	79,4
FRRN-B [31]	-	24,8	81,7
Encoder-Decoder [3]	-	35,0	43,1
Encoder-Decoder-Skip [3]	-	35,0	42,9
MobileUNet [39]	-	8,8	42,7
MobileUNet-Skip [39]	-	8,9	43,4
RefineNet [28]	ResNet50/101/152/ MobileNetV2/ SEResNeXt50/101	66,7/85,6/101,3/ 44,2/ 67,5/87,6	36,1/42,8/50,4/ 30,4/ 44,9/58,5
PSPNet [46]	ResNet50/101/152/ MobileNetV2/ SEResNeXt50/101	37,0/56,0/71,7/ 14,2/ 39,3/59,4	23,5/23,6/23,7/ 22,2/ 30,1/29,8
GCN [30]	ResNet50/101/152/ MobileNetV2/ SEResNeXt50/101	24,0/43,0/58,7/ 2,4/ 24,7/44,8	15,1/21,3/28,4/ 9,7/ 24,2/37,4
DeepLabV3 [8]	ResNet50/101/152/ MobileNetV2/ SEResNeXt50/101	27,6/46,6/62,3/ 3,0/ 32,0/52,2	11,6/11,7/12,9/ 8,1/ 22,2/35,4
DeepLabV3_plus [8]	ResNet50/101/152/ MobileNetV2/ SEResNeXt50/101	28,9/48,0/63,6/ 4,3/ 33,3/53,5	14,2/14,3/15,6/ 11,0/ 25,5/38,7
DenseASPP [19]	ResNet50/101/152/ MobileNetV2/ SEResNeXt50/101	24,8/43,8/59,4/ 3,2/ 25,6/45,8	6,9/7,2/7,8/ 6,7/ 14,2/16,8
SeparNet (vlastní)	Resnet152	80,2	48,2

Tabulka 7.1: Tabulka zobrazující přehled testovaných sítí. Je uvedený počet trénovacích parametrů a rychlost, za kterou je síť schopná vyhodnotit vstup o rozměru 512x512x3 na grafické kartě GeForce RTX 2080 Ti.

7.2 Průběh trénování

V průběhu trénování se nejprve zaměříme na časovou a paměťovou náročnost při trénování sítí. Následně budeme sledovat vývoj hodnot loss napříč epochami trénování jednotlivých sítí.

V tabulce 7.2 jsou zobrazeny všechny architektury sítí, které trénujeme. Jsou u nich uvedené počty parametrů, čas potřebný k natrénování jedné epochy, velikost vstupu sítě a paměťovou náročnost grafické paměti. Z tabulky lze vyčíst, že síť DeepLabV3+ se s velkým rozdílem trénuje nejrychleji a síť SeparNet využívá při trénování nejméně grafické paměti.

Název sítě	Feature extractor	Parametry (M)	Čas trénování 1 epochy	Velikost vstupu	Paměť (GB)
FC-DenseNet56	ne	1.62	1h 50min	640	9.27
FC-DenseNet103	ne	9.27	2h 42min	512	10.98
FRRN-A	ne	17.74	4h 12min	704	9.76
FRRN-B	ne	24.75	4h 35min	704	9.76
RefineNet	Resnet152	101.34	2h 30min	704	6.85
DeepLabV3+	Resnet152	63.62	**44min**	704	7.54
SeparNet(vlastní)	Resnet152	80.33	2h 12min	704	**6.72**

Tabulka 7.2: Trénování state-of-the-art architektur. Batch size má velikost 1.

Náš navržený systém(kapitola 5) zaznamenává po každé natrénované epoše průměrnou loss hodnotu. Při trénování se musí tyto hodnoty hlídat a kontrolovat, zda konvergují k hodnotě z intervalu $(0, 0.5)$. V opačném případě to nejčastěji indikuje chybu trénování. Může se jednat o nekvalitní datovou sadu, chybu v předzpracování dat, špatně zvolené hyperparametry sítě, nejčastěji learning rate, apod.

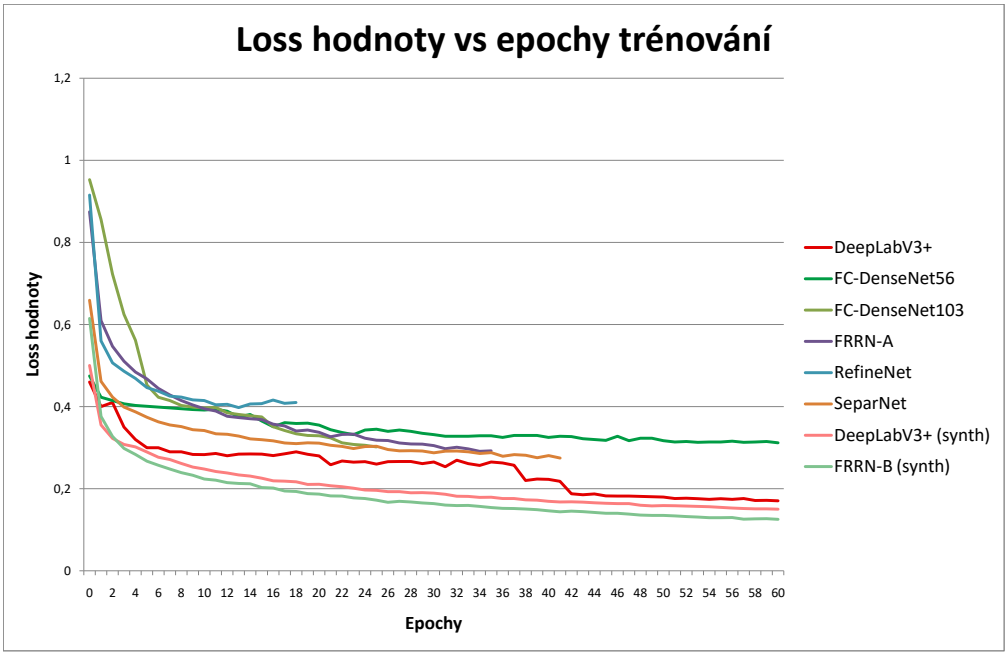
7.2.1 Předtrénování sítí

Na grafu 7.1 jsou uvedeny pro srovnání všechny sítě, které jsem trénoval na datových sadách z pohledu řidiče. Sítě DeepLabV3+(synth) a FRRN-B(synth) jsou trénované pouze na syntetických datech z pohledu řidiče (datová sada GTAV-street). Ostatní sítě jsou trénované na smíšené datové sadě.

Z grafu lze vyčíst to, že u všech sítí mají loss hodnoty správný trend a to znamená, že se správně učí. Graf 7.1 zobrazuje prvních 60 epoch trénování.

Hodnoty loss u sítích trénovaných pouze na syntetických datech se dostaly na nižší hodnoty než v případě trénování na smíšené datové sadě. Je to z důvodu toho, že síť, která se trénovala pouze na syntetických datech měla jednodušší úkol pro trénování (omezená datová sada).

Sítě DeepLabV3+ a FRRN-A jsem trénoval na 400 epoch v rámci experimentu, zda se sítě přetrénují a následně ztratí sílu generalizace na jiné dopravní situace než ty, které měly v trénovací datové sadě. Experiment ukázal, že se sítě nepřetrénovaly a měly stejnou kvalitu jako ty, které se trénovaly pouze do 60. epochy.



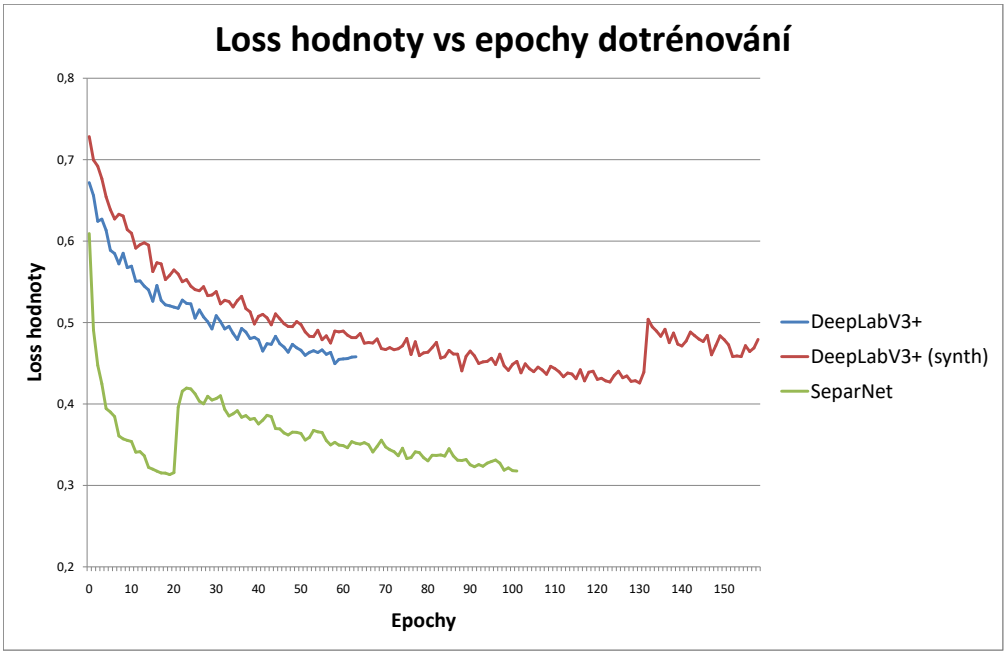
Obrázek 7.1: Průběh loss hodnot napříč epochami. Trénováno na datové sadě reálných i syntetických dat dohromady.

7.2.2 Dotrénování sítí

Na základě experimentů 8.3, které byly provedeny nad sítěmi, které segmentují scény z pohledu řidiče, jsem vybral nejlepší dvě sítě – DeepLabV3+ a SeparNet. Ty byly trénované na smíšené datové sadě. Dále jsem zvolil DeepLabV3+(synth), která byla trénovaná na syntetické datové sadě. Všechny tyto tři sítě se budou dotrénovávat na vlastní datové sadě (GTAV_custom), která síti umožní segmentovat scény z ptačích pohledů.

Graf 7.2 zobrazuje průběh loss hodnoty zvolených sítích. Je zde patrné, že u sítě SeparNet na 22. epoše a u sítě DeepLabV3+(synth) 132. epoše byl spuštěn modul OHEM, který síti začal předávat jen vzorky z datové sady, které síť neuměla správně segmentovat.

Výstup tohoto trénování byl podroben experimentům(kapitola 8).



Obrázek 7.2: Průběh loss hodnot napříč epochami. Dotrénování na syntetické datové sadě z ptačího pohledu.

7.3 Možné další vylepšení trénování

V této sekci jsou uvedeny budoucí možnosti vylepšení systému pro trénování a je zde uvedeno rozšíření v podobě restartů hodnot learning rate.

7.3.1 Restarty learning rate

Restarty learning rate [13] umožňují dostat se z lokálního minima při trénování a tím zvýšit kvalitu výsledného natrénovaného modelu. Toto vylepšení by se využilo v případě, kdyby hrozilo uváznutí v lokálním minimu při optimalizaci neuronové sítě. Využilo by se nejčastěji v případech, kdybychom neměli dostatečně robustní datovou sadu. V případě dostatečné datové sady se všechny lokální minima rovnají globálnímu. Této myšlenky využívá optimalizační algoritmus Stochastic Gradient Descent with Restarts (SGDR) [24].

Podle následujících kroků lze implementovat algoritmus restartů learning rate:

1. Nalezení minimální a maximální hodnoty learning ratu pomocí postupného zvyšování hodnoty learning rate a sledování hodnoty loss. V případě, že se loss hodnota nemění, znamená to, že je LR velice nízký. V případě, že loss hodnota nekonverguje k nule ale diverguje, znamená to, že je LR příliš vysoký. Optimální hodnoty LR jsou v případě, že loss hodnota konverguje k nule [40].

2. Po nalezení rozmezí LR hodnot $(\eta_{min}, \eta^i_{max})$ použije se rovnice 7.1 pro výpočet aktuální hodnoty LR.

$$\eta_{curr} = \eta_{min} + \frac{1}{2}(\eta_{max} - \eta_{min})(1 + cos(\frac{T_{current}\ mod\ T}{T}\pi)) \tag{7.1}$$

kde:

- η_{curr} je aktuální LR.
- η_{min}, η_{max} definují rozsah LR.
- $T_{current}$ reprezentuje aktuální číslo iterace.
- T reprezentuje počet iterací v jednom cyklu.

V případě, že platí $(T_{current}\ mod\ T_i) + 1 == T_i$, tak se může začít skokově snižovat hodnota η_{max}.

7.3.2 Deep supervision

Deep supervision je technika, která přidává loss funkce do hlubokých sítí. Tato technika zajišťuje postupné trénování od prvních k posledním vrstvám sítě[2]. Kdybychom neměli tyto přidané loss funkce v paralelních větvích, tak by se nám u hlubokých sítí trénovaly pouze poslední vrstvy. Gradient by se nedokázal propagovat celou sítí. Tuto techniku Deep supervision využila společnost Google ve své klasifikační síti GoogleNet [43] nebo segmentační síť UNet++ [47].

7.3.3 Změna optimalizačního algoritmu za běhu

Jedná se o modul umožňující dynamickou změnu optimalizačního algoritmu v případě, kdy systém pomocí OHEM zjistí, že model je již zoptimalizovaný daným algoritmem. Změna na více precizní, ale časově náročnější optimalizační algoritmus zajistí finální doladění sítě do nejvyšší možné kvality[3] [20].

[2] Myšlena převážně síť GoogleNet a Inception
[3] Jedná se o změnu z optimalizačního algoritmu ADAM / RMSProb na SGD.

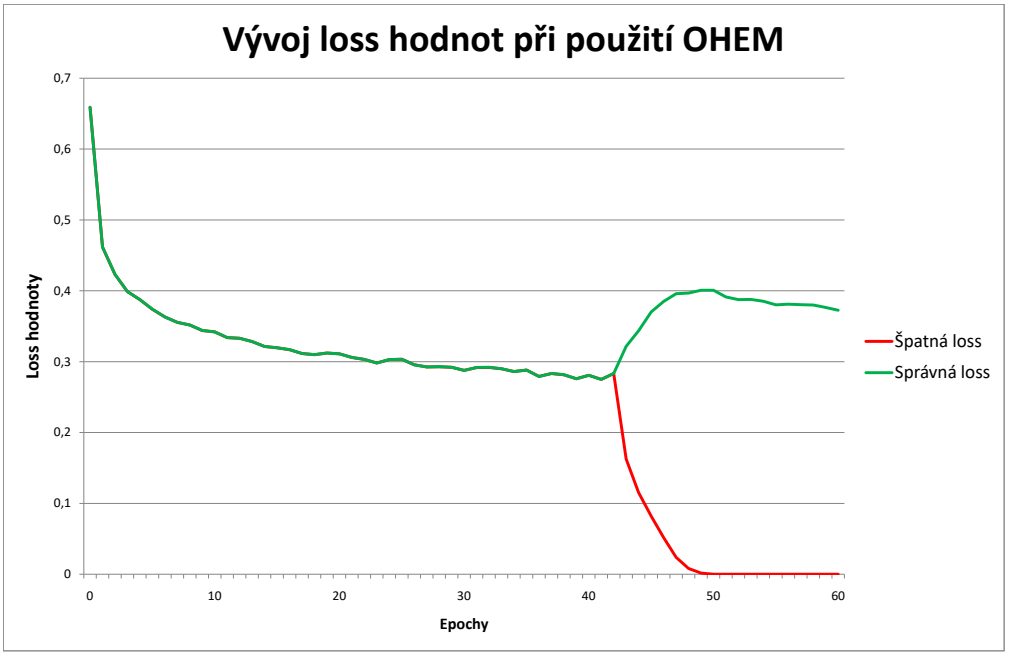
Kapitola 8

Experimenty

V této kapitole budeme experimentovat s natrénovanými modely sítí. Nejprve provedeme experiment s OHEM, ze kterého získáme jeho důležité vlastnosti. Následně budeme získávat základní metriky jednotlivých modelů. Bude se jednat o metriky Accuracy, Precision, Recall, F1 score a IoU (Intersection of Union). Pomocí těchto experimentů vybereme ty nejlepší modely a zhodnotíme výsledky. V závěrečné části kapitoly budeme testovat nejlepší modely na schopnost segmentovat scény z reálného provozu.

8.1 OHEM experiment

Nejprve budeme experimentovat s nastavením OHEM. Lze nastavit, aby vybíral pro síť těžké vzorky, ze kterých se bude učit nové vlastnosti (správná volba) nebo může být chybně nastaven a vybírat jen jednoduché vzorky, což způsobí znatelnou degradaci výsledného modelu. Ověření, zda je OHEM správně nastaven můžeme kontrolovat z vývoje loss hodnoty při trénování, která je zobrazena na grafu 8.1. Loss hodnota by se správně při použití OHEM měla znatelně zvýšit a následně začít klesat. Navýšení loss hodnoty způsobí to, že síti začneme předávat pouze těžké vzorky, ve kterých dělá velké množství chyb. Časem se i tyto těžké vzorky naučí správně segmentovat a loss hodnota začne klesat.



Obrázek 8.1: Průběh loss hodnot napříč epochami s využitím metody OHEM.

Tabulka 8.1 zobrazuje výsledné metriky sítí před a po použití OHEM. Tyto hodnoty byly získány z předtrénování sítě SeparNet a z dotrénování sítí SeparNet a DeepLabV3+(synth). Z tabulky můžeme vyčíst, že OHEM obecně zvyšuje přesnost segmentace u méně zastoupených tříd[1] a vysoce zastoupeným třídám spíše kvalitu sníží, protože síť při trénování dostává menší procento této třídy.

Name	AP	Prec	Recall	F1	IoU	Void	Build	Prsn	Sky	Hvy	Inf	Tiny	Car	Veg
SeparNet	0.9309	0.9464	0.9309	0.9312	**0.7109**	**0.5643**	0.8952	0.8126	**0.9396**	0.9166	0.9536	0.9494	0.9044	**0.8649**
OHEM	**0.9317**	**0.9508**	**0.9317**	**0.9338**	0.7057	0.5373	**0.9184**	**0.8203**	0.9380	**0.9177**	**0.9551**	**0.9504**	**0.9161**	0.8592

SeparNet	0.9249	0.9346	0.9249	0.9244	0.5847	**0.7168**	0.6336	0.7332	-	0.9884	**0.9600**	-	0.9384	**0.7811**
OHEM	**0.9330**	**0.9436**	**0.9330**	**0.9341**	**0.5981**	0.7025	**0.6766**	**0.7444**	-	**0.9901**	0.9590	-	**0.9418**	0.7680
DeepLabV3+(synth)	0.8311	**0.9584**	0.8311	0.8403	0.4025	0.4550	0.4032	**0.6853**	-	0.9643	**0.9855**	-	0.8950	**0.7495**
OHEM	**0.8454**	0.8964	**0.8454**	**0.8525**	**0.4163**	**0.4652**	**0.4440**	**0.6853**	-	**0.9661**	0.9427	-	**0.9271**	0.7340

Tabulka 8.1: Zobrazení metrik sítí při použití OHEM.

[1] Car, Heavy, Tiny

8.2 Validace

V této sekci si nejprve uvedeme základní metriky, podle kterých budeme určovat kvalitu modelu. Následně budeme validovat sítě trénované na pohled řidiče auta a následně sítě trénované na ptačí pohledy. V závěru podkapitoly zhodnotíme všechny výsledky.

8.2.1 Metriky

V této sekci budou uvedeny testované metriky. Nejprve si musíme zavést základní pojmy:

- **True Positive (TP)**: GT (Ground Truth) maska má korespondující predikci všech definovaných tříd.
- **False Positive (FP)**: predikce nemá korespondující GT masku. Segmentujeme objekt, který ve scéně neexistuje.
- **True Negative (TN)**: GT má korespondující predikci ve třídě void[2].
- **False Negative (FN)**: GT nemá korespondující predikci. Nejsme schopní identifikovat objekt.

Testované metriky jsou následující:

- **Accuracy** udává kolik procent pixelů z predikce koresponduje s maskou. Vzorec následuje níže.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{8.1}$$

  Podle tohoto vzorečku se počítá také **třídní přesnost** (class-specific accuracy), která udává přesnost jednotlivých klasifikovaných tříd.

- **Precision** popisuje „čistotu“ našich pozitivních predikcí vzhledem k GT. Jedná se o poměr mezi počtem pixelů segmentovaných tříd a GT.

$$Precision = \frac{TP}{TP + FP} \tag{8.2}$$

- **Recall** popisuje „úplnost“ našich pozitivních predikcí vůči všem anotovaným objektům. Neboli kolik procent objektů jsme segmentovali vůči všem objektům.

$$Recall = \frac{TP}{TP + FN} \tag{8.3}$$

- **F1 score** udává harmonický průměr mezi precision a recall metrikami.

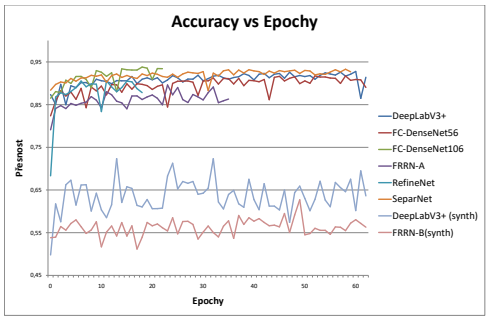
$$F1score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{8.4}$$

- **IoU (Intersection of Union)** udává podíl mezi průnikem (intersection) a sjednocením (union) pixelů predikce (prediction) a masky (target).

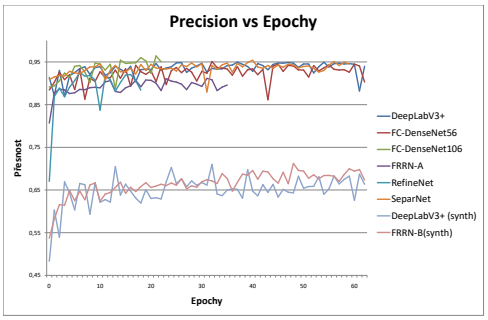
$$IoU = \frac{target \bigcap prediction}{target \bigcup prediction} \tag{8.5}$$

[2]Třída, která obsahuje všechny nedefinované třídy zájmu.

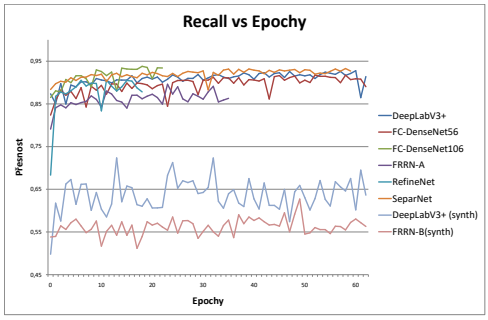
8.2.2 Validace sítí pro pohled z pozice řidiče



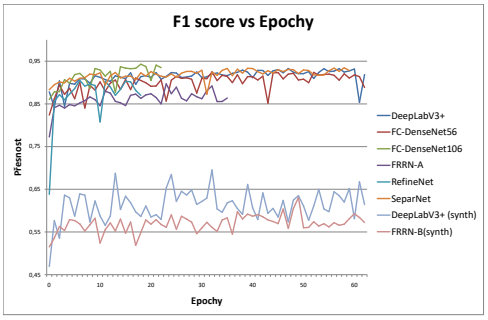
Obrázek 8.2: Metrika Accuracy v průběhu epoch trénování.



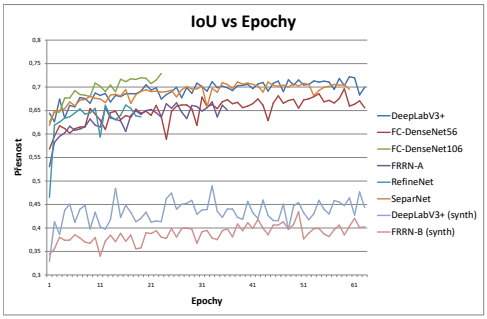
Obrázek 8.3: Metrika Precision v průběhu epoch trénování.



Obrázek 8.4: Metrika Recall v průběhu epoch trénování.



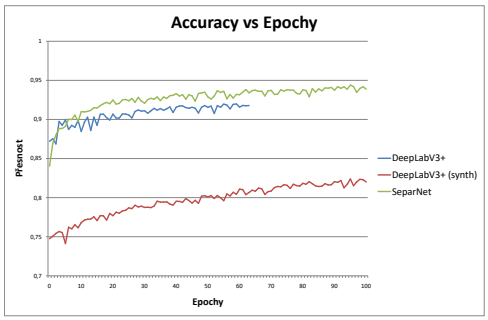
Obrázek 8.5: Metrika F1 score v průběhu epoch trénování.



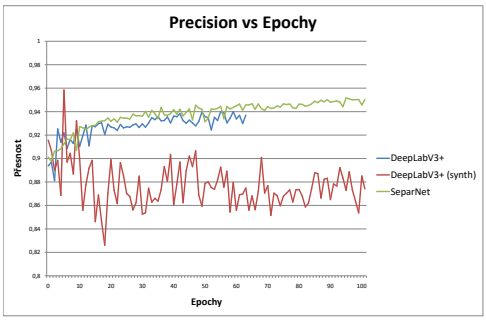
Obrázek 8.6: Metrika IoU v průběhu epoch trénování.

Validační datová sada se skládá z 1 000 anotovaných reálných i syntetických snímků, které nebyly použité v trénovací datové sadě. Z výsledků validace vychází s nejlepším skóre síť FC-DenseNet106 a hned za ní síť SeparNet. Sítě, které se trénovaly pouze na syntetických datech podle očekávání dopadly hůře.

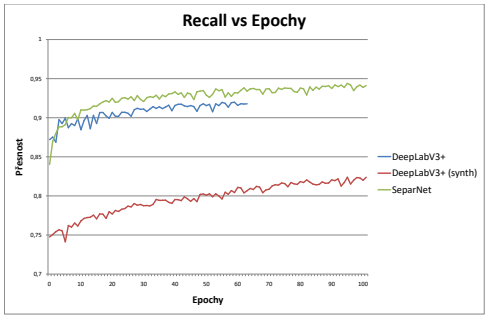
8.2.3 Validace sítí pro ptačí pohled



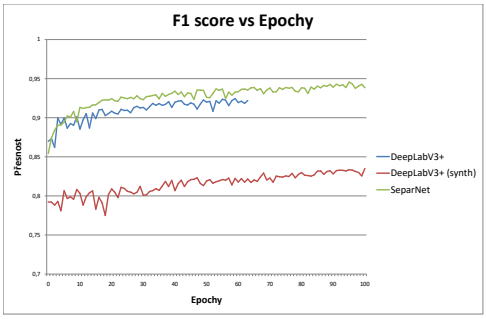
Obrázek 8.7: Metrika Accuracy v průběhu epoch trénování.



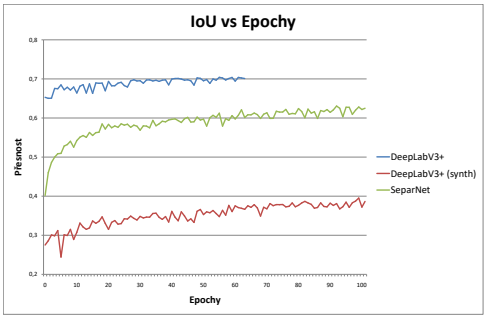
Obrázek 8.8: Metrika Precision v průběhu epoch trénování.



Obrázek 8.9: Metrika Recall v průběhu epoch trénování.



Obrázek 8.10: Metrika F1 score v průběhu epoch trénování.



Obrázek 8.11: Metrika IoU v průběhu epoch trénování.

Validační datová sada pro ptačí pohledy se skládá z 500 anotovaných syntetických snímků. Z výsledků validace nejlépe vychází SeparNet, v metrice IoU DeepLabV3+.

8.2.4 Zhodnocení výsledků

Bylo trénováno osm sítí na pohledu řidiče auta. Z toho šest sítí na datové sadě obsahující jak reálná, tak i syntetická data. Zbývající dvě sítě byly trénovány pouze na syntetických datech. Tabulka 8.2 shrnuje všechny metriky sítí, které byly v rámci této diplomové práce zjištěny. Nejlépe se umístila síť FC-DenseNet103, která ve většině metrik zvítězila. Druhých nejlepších výsledků dosáhla síť SeparNet.

Obecně však nelze říci, která síť byla nejlepší. Vždy budeme mít různé požadavky na síť. Pokud budeme potřebovat nejkvalitnější segmentaci a nebude nás zajímat čas zpracování, ani paměťová náročnost, tak by byla vhodná síť FC-DenseNet103. V případě, že bychom potřebovali kvalitní výsledky v reálném čase, tak by se nabízely sítě SeparNet nebo DeepLabV3+, atd.

	AP	Prec	Recall	F1	IoU	Prm (M)	Rychlost (ms)	Paměť (GB)	Čas trénování (1 epocha)
DeepLabV3+	0.9276	0.9495	0.9276	0.9329	**0.7221**	63.6	**15.6**	**5.48**	**44min**
FC-DenseNet56	0.9174	0.9508	0.9174	0.9250	0.6965	3.4	70.2	7.42	**1h 50min**
FC-DenseNet103	**0.9380**	**0.9639**	**0.9380**	**0.9429**	**0.7290**	9.2	107.9	10.98	2h 42min
FRRN-A	0.8960	0.9114	0.8960	0.8968	0.6685	17.7	79.4	7.09	4h 12min
RefineNet	0.9053	0.9254	0.9053	0.9053	0.6618	**101.3**	50.4	7.09	2h 30min
SeparNet	**0.9329**	**0.9538**	**0.9329**	**0.9346**	0.7109	**80.2**	**48.2**	**4.88**	2h 12min
DeepLabV3+ (synth)	0.7329	0.7203	0.7329	0.7073	0.5097	63.6	**15.6**	**5.48**	**44min**
FRRN-B (synth)	0.7329	0.7198	0.7329	0.7073	0.5097	24.8	81.7	7.09	4h 35min

	Void	Building	Person	Sky	Heavy	Inf	Tiny	Car	Veg
DeepLabV3+	0.6240	0.9067	**0.8817**	0.9547	0.9429	0.9427	**0.9963**	**0.9855**	**0.9520**
FC-DenseNet56	0.5641	0.8957	0.8227	0.9467	0.9325	0.9409	0.9527	**0.9625**	**0.9231**
FC-DenseNet103	0.6728	**0.9336**	**0.8972**	**0.9594**	**0.9593**	0.9481	**0.9802**	0.9472	0.8625
FRRN-A	0.6420	0.8801	0.8338	**0.9575**	0.9271	0.9703	0.9557	0.9204	0.8785
RefineNet	**0.7847**	0.9118	0.8001	0.9544	0.9101	**0.9888**	0.9606	0.9225	0.8763
SeparNet	**0.7012**	**0.9252**	0.8466	0.9540	0.9302	**0.9800**	0.9576	0.9269	0.9084
DeepLabV3+ (synth)	0.6927	0.8831	0.8569	0.8029	0.9375	0.9660	0.9452	0.8925	0.8714
FRRN-B (synth)	0.6905	0.9201	0.8501	0.8029	**0.9484**	0.9514	0.9452	0.9096	0.8247

Tabulka 8.2: Zhodnocení výsledků sítí segmentujících scény z pohledu řidiče.

Na základě experimentů z reálného provozu (kapitola 8.3) byly zvoleny tři sítě pro dotrénování na ptačích pohledech. Tabulka 8.3 zobrazuje všechny výsledné metriky sítí specializované na ptačí pohled. Třídy Sky a Tiny v tabulce nejsou zahrnuty, protože validační sada tyto třídy neobsahovala.

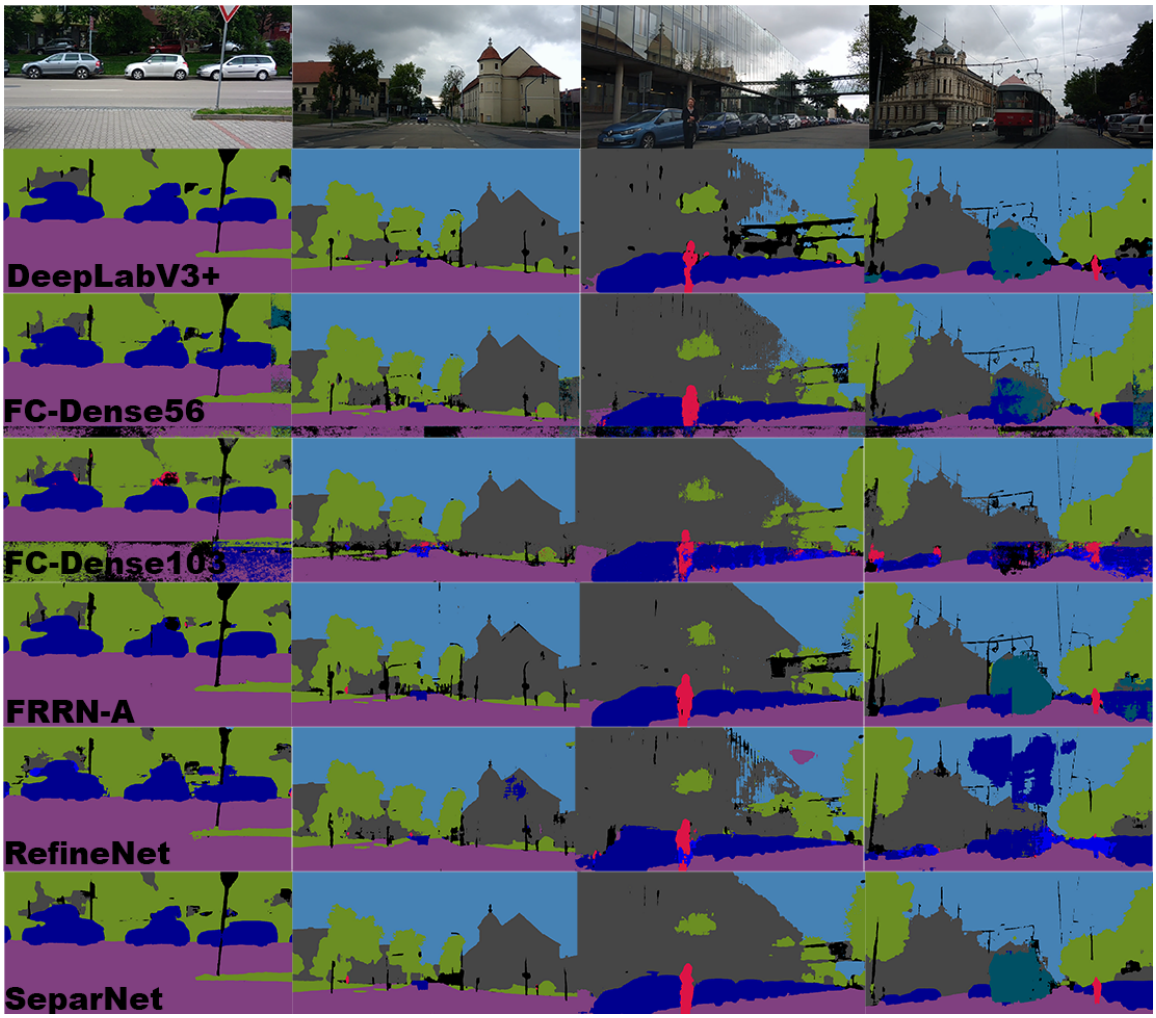
	AP	Prec	Recall	F1	IoU	Prm (M)	Rychlost (ms)	Paměť (GB)	Čas trénování (1 epocha)
DeepLabV3+	0.920	0.940	0.920	0.925	**0.704**	63.6	**15.6**	5.48	**44min**
DeepLabV3+ (synth)	0.811	0.958	0.811	0.823	0.376	63.6	**15.6**	5.48	**44min**
SeparNet	**0.944**	**0.952**	**0.944**	**0.946**	0.631	**80.2**	48.2	**4.88**	2h 12min

	Void	Building	Person	Sky	Heavy	Inf	Tiny	Car	Veg
DeepLabV3+	0.718	**0.805**	**0.918**	-	0.972	0.983	-	0.953	**0.908**
DeepLabV3+ (synth)	0.365	0.392	0.685	-	0.961	**0.985**	-	0.870	0.726
SeparNet	**0.754**	0.729	0.758	-	**0.992**	0.972	-	**0.964**	0.812

Tabulka 8.3: Zhodnocení výsledků sítí segmentujících scény z ptačího pohledu.

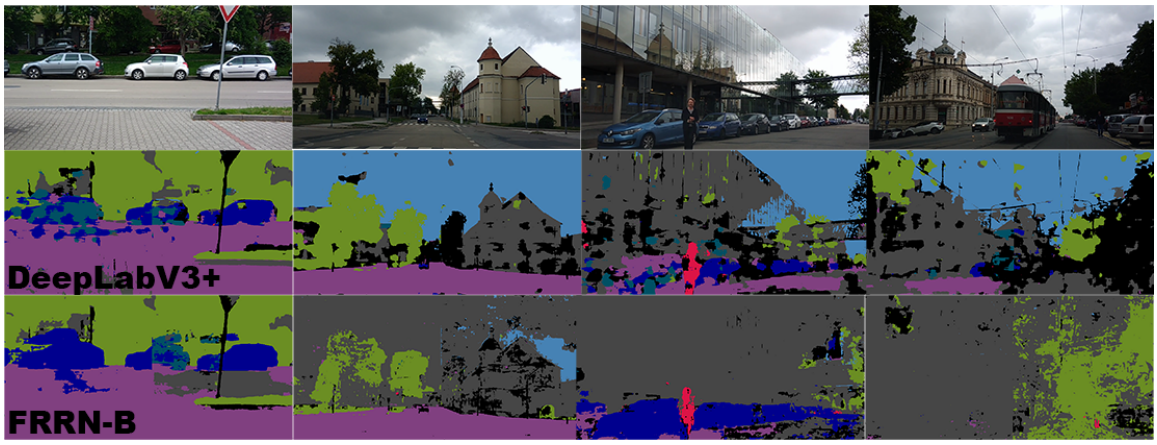
8.3 Testování na reálných datech

Experimenty na datech z reálného provozu shrnují schopnost všech sítí adaptovat se reálné scéně. Obrázek 8.12 zobrazuje výstupy sítí trénovaných na smíšené datové sadě. Podle subjektivního hlediska se sítě SeparNet a DeepLabV3+ dokázaly nejlépe přizpůsobit reálným datům. Síť FC-DenseNet103, která měla nejlepší validační výsledky, reálná data příliš segmentovat nezvládla.



Obrázek 8.12: Sítě trénované na reálné i syntetické datové sadě testovány na reálných datech.

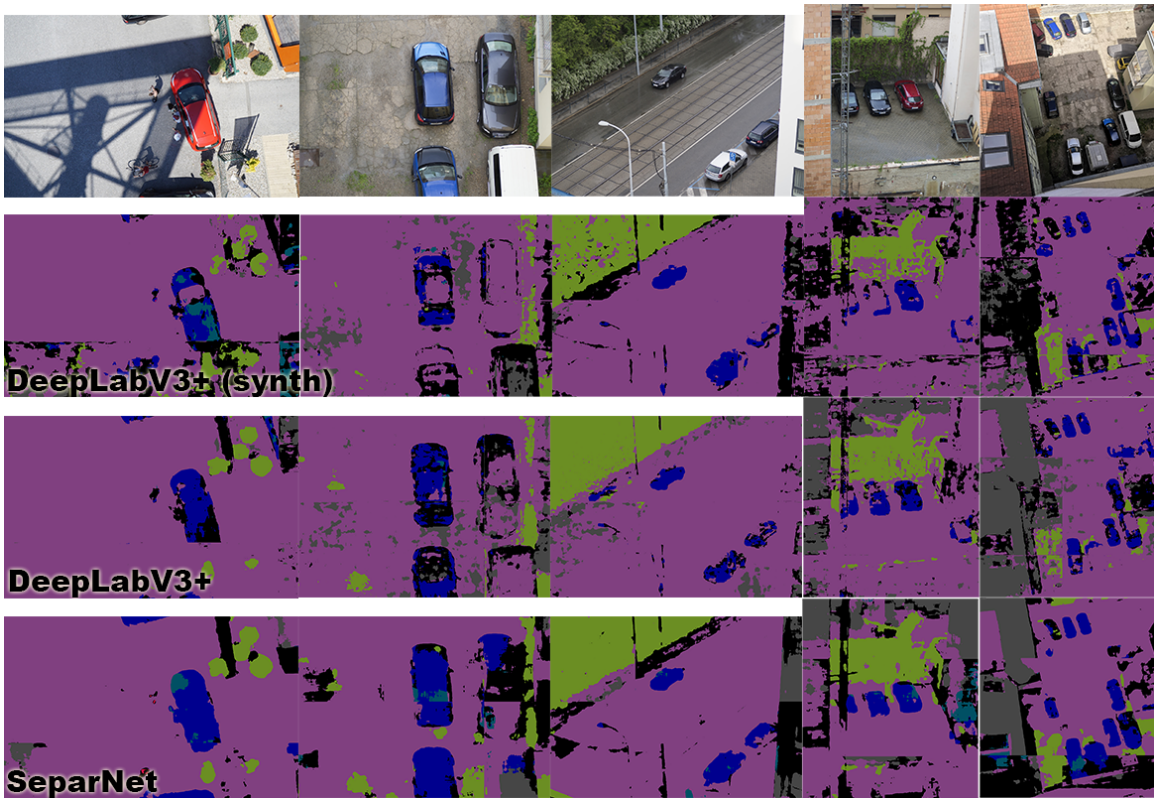
Obrázek 8.13 zobrazuje výstupy sítí trénovaných na syntetických datech. Je zde patrné, že v některých případech sítě dokážou segmentovat základní tvary, ale může se stát, že v některých případech úplně selžou.



Obrázek 8.13: Sítě trénované na syntetické datové sadě testovány na reálných datech.

Při závěrečném testování sítí z ptačího pohledu na reálných datech (obrázek 8.14) bylo zjištěno, že síť, která se trénovala od začátku do konce pouze jen na syntetických datech (DeepLabV3+(synth)) byla schopná segmentovat tyto pohledy. Toto zjištění otevírá možnosti základní segmentace různých pohledů bez nutnosti vlastnictví datových sad z reálného prostředí.

Dále podle tohoto experimentu bylo zjištěno, že síť SeparNet se dokázala nejlépe přizpůsobit jinému pohledu a dávala nejkvalitnější výsledky.



Obrázek 8.14: Sítě dotrénované na syntetické datové sadě testovány na reálných datech.

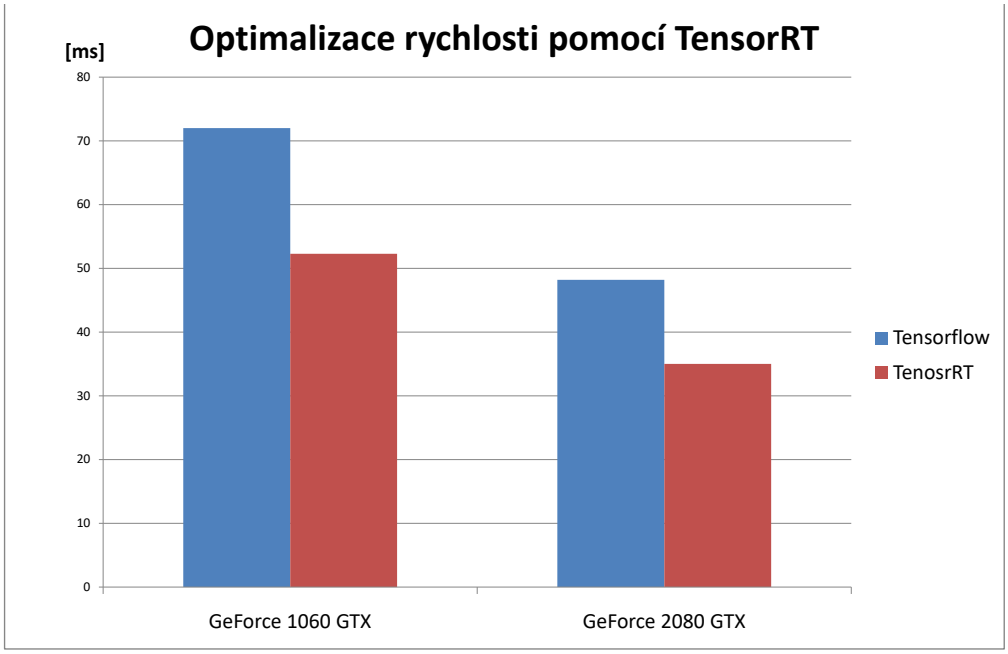
Kapitola 9

Optimalizace

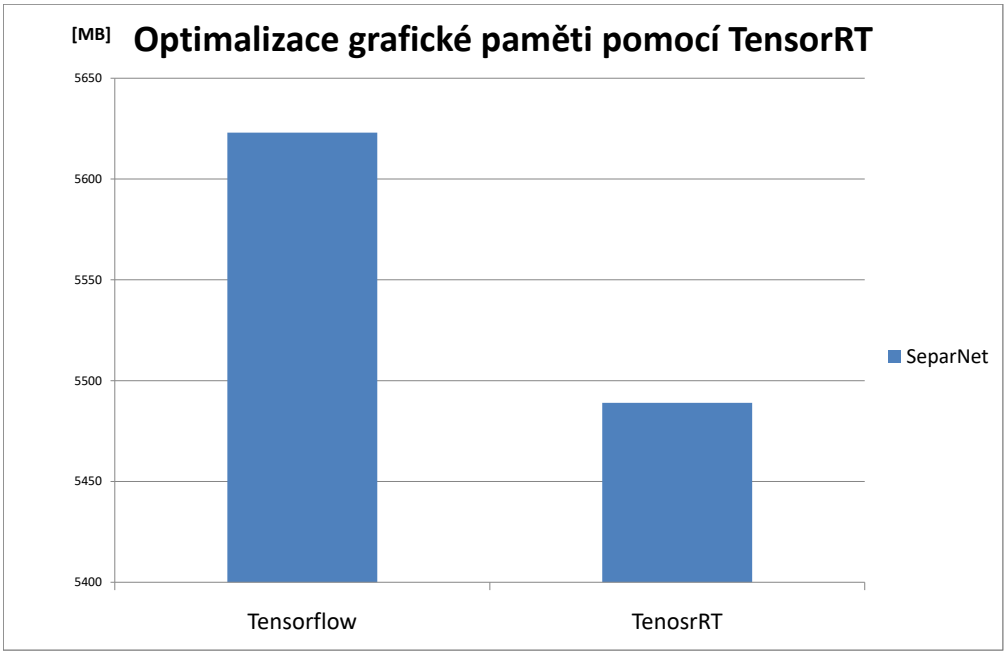
V této kapitole bude uvedena optimalizace provedená na výsledném modelu sítě.

9.1 TensorRT

TensorRT je platforma, která slouží k optimalizaci modelů neuronových sítí pro rychlé vyhodnocení na grafické kartě. Je velice vhodné tento nástroj využít na natrénovaný model sítě. Jak je patrné z grafu 9.1, bylo docíleno vysokého zrychlení a snížení potřebné grafické paměti k vyhodnocení sítě (graf 9.2). Tato optimalizace nepatrně snížila kvalitu segmentace scény.



Obrázek 9.1: Optimalizace rychlosti pomocí TensorRT.



Obrázek 9.2: Optimalizace paměťových nároků sítě pomocí TensorRT.

Kapitola 10

Závěr

Cílem této práce bylo seznámení čtenáře s konvolučními neuronovými sítěmi, které provádí sémantickou segmentaci scény z dopravního prostředí. Byly zde popsány základní vrstvy a bloky, ze kterých se skládají konvoluční neuronové sítě a vrstvy a moduly, které jsou specifické pro segmentační neuronové sítě.

Na základě teoretické znalosti byla vytvořená vlastní architektura neuronové sítě s názvem SeparNet.

Dále byla vytvořena datová sada ze silničního prostředí, která se skládala z reálných i syntetických dat z pohledu řidiče auta. Pro vytvoření vlastní specifické datové sady byl vytvořen automatický nástroj pro generování syntetických dat ze hry GTAV. Pomocí této datové sady bylo možné vytvořit neuronovou síť, která segmentuje scénu z ptačího pohledu.

Pro kompletní práci se segmentačními neuronovými sítěmi byl vytvořen systém, který umožňuje trénování, validaci, testování, optimalizaci, analýzu a vytváření segmentačních neuronových sítí. Byly zde popsány všechny součásti systému i části, které se budou ještě implementovat.

Dále byly zvoleny state-of-the-art architektury sítí, které byly trénovány a validovány na připravené datové sadě. S výslednými sítěmi byly prováděny experimenty.

Výstupem této práce je vlastní segmentační neuronová síť, která porazila state-of-the-art architektury hlavně v kvalitě výstupu, v paměťové náročnosti a ve schopnosti přizpůsobit se reálnému prostředí. Další přínos této práce je praktická ukázka toho, že lze segmentovat reálné scény pouze na základě syntetických dat. To implikuje možnost razantně zredukovat lidskou práci na anotování reálných scén, která je velice nákladná a časové náročná.

Tato práce může být základem pro velké množství projektů. Může se jednat o systém pro segmentaci plevelu na poli pro zemědělské stroje, které proti němu kropí pole chemickými přípravky. Tento systém by umožňoval přesnou lokalizaci plevelu a mohly by se chemické přípravky aplikovat přímo na něj bez nutnosti globálního postřiku. Dále se výstup této práce může využít pro záložní navigační systém pro drony, který by byl využit v případě nouzového přistání bez navigačních systémů. Pomocí segmentace scény by nalezl bezpečné místo pro přistání.

Vyvinutý systém pro práci se segmentačními sítěmi se bude dále rozvíjet. Budou se implementovat náměty na vylepšení trénování, které jsou uvedené v kapitole 7.3. Dále se do systému bude integrovat modul umožňující práci s klasifikátorem a detektorem objektů. Tato integrace bude umožňovat jednotnou práci s těmito nástroji. Dále se bude pokračovat na rozšiřování datové sady o kvalitní vzorky, které jsou nedílnou součástí při vývoji.

Literatura

[1] OpenIV Documentation. http://docs.openiv.com/doku.php, 2019.

[2] Badrinarayanan, V.; Kendall, A.; Cipolla, R.: SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, ročník 39, č. 12, Dec 2017: s. 2481–2495, ISSN 0162-8828, doi:10.1109/TPAMI.2016.2644615.

[3] Badrinarayanan, V.; Kendall, A.; Cipolla, R.: SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, ročník 39, č. 12, Dec 2017: s. 2481–2495, ISSN 0162-8828, doi:10.1109/TPAMI.2016.2644615.

[4] Blade, A.: Community Script Hook V .NET. https://github.com/crosire/scripthookvdotnet, 2019.

[5] Bowles, C.; Chen, L.; Guerrero, R.; aj.: GAN Augmentation: Augmenting Training Data using Generative Adversarial Networks. *CoRR*, ročník abs/1810.10863, 2018, 1810.10863.
URL http://arxiv.org/abs/1810.10863

[6] Brostow, G. J.; Fauqueur, J.; Cipolla, R.: Semantic object classes in video: A high-definition ground truth database. *Pattern Recognition Letters*, ročník 30, 2009: s. 88–97.

[7] Chen, L.; Zhu, Y.; Papandreou, G.; aj.: Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation. *CoRR*, ročník abs/1802.02611, 2018, 1802.02611.
URL http://arxiv.org/abs/1802.02611

[8] Chen, L.-C.; Zhu, Y.; Papandreou, G.; aj.: Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation. In *ECCV*, 2018.

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

[10] dexyfex: CodeWalker. https://github.com/dexyfex/CodeWalker, 2019.

[11] Geiger, A.; Lenz, P.; Stiller, C.; aj.: Vision meets Robotics: The KITTI Dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[12] Gidaris, S.; Singh, P.; Komodakis, N.: Unsupervised Representation Learning by Predicting Image Rotations. *CoRR*, ročník abs/1803.07728, 2018, 1803.07728.
URL http://arxiv.org/abs/1803.07728

[13] Gotmare, A.; Keskar, N. S.; Xiong, C.; aj.: A Closer Look at Deep Learning Heuristics: Learning rate restarts, Warmup and Distillation. In *International Conference on Learning Representations*, 2019.
URL https://openreview.net/forum?id=r14EOsCqKX

[14] Hamaguchi, R.; Fujita, A.; Nemoto, K.; aj.: Effective Use of Dilated Convolutions for Segmenting Small Object Instances in Remote Sensing Imagery. *CoRR*, ročník abs/1709.00179, 2017, 1709.00179.
URL http://arxiv.org/abs/1709.00179

[15] He, K.; Zhang, X.; Ren, S.; aj.: Deep Residual Learning for Image Recognition. *CoRR*, ročník abs/1512.03385, 2015, 1512.03385.
URL http://arxiv.org/abs/1512.03385

[16] Hu, J.; Shen, L.; Sun, G.: Squeeze-and-Excitation Networks. *CoRR*, ročník abs/1709.01507, 2017, 1709.01507.
URL http://arxiv.org/abs/1709.01507

[17] Hu, J.; Shen, L.; Sun, G.: Squeeze-and-Excitation Networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[18] Iglovikov, V.; Shvets, A.: TernausNet: U-Net with VGG11 Encoder Pre-Trained on ImageNet for Image Segmentation. *CoRR*, ročník abs/1801.05746, 2018, 1801.05746.
URL http://arxiv.org/abs/1801.05746

[19] Jégou, S.; Drozdzal, M.; Vázquez, D.; aj.: The One Hundred Layers Tiramisu: Fully Convolutional DenseNets for Semantic Segmentation. *CoRR*, ročník abs/1611.09326, 2016, 1611.09326.
URL http://arxiv.org/abs/1611.09326

[20] Keskar, N. S.; Socher, R.: Improving Generalization Performance by Switching from Adam to SGD. *CoRR*, ročník abs/1712.07628, 2017, 1712.07628.
URL http://arxiv.org/abs/1712.07628

[21] Leibo, J. Z.; de Masson d'Autume, C.; Zoran, D.; aj.: Psychlab: A Psychology Laboratory for Deep Reinforcement Learning Agents. *CoRR*, ročník abs/1801.08116, 2018, 1801.08116.
URL http://arxiv.org/abs/1801.08116

[22] Lin, G.; Milan, A.; Shen, C.; aj.: RefineNet: Multi-Path Refinement Networks for High-Resolution Semantic Segmentation. *CoRR*, ročník abs/1611.06612, 2016, 1611.06612.
URL http://arxiv.org/abs/1611.06612

[23] Lin, T.-Y.; Goyal, P.; Girshick, R.; aj.: Focal Loss for Dense Object Detection. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[24] Loshchilov, I.; Hutter, F.: SGDR: Stochastic Gradient Descent with Restarts. *CoRR*, ročník abs/1608.03983, 2016, 1608.03983.
URL http://arxiv.org/abs/1608.03983

[25] Luo, W.; Li, Y.; Urtasun, R.; aj.: Understanding the Effective Receptive Field in Deep Convolutional Neural Networks. *CoRR*, ročník abs/1701.04128, 2017, 1701.04128.
URL http://arxiv.org/abs/1701.04128

[26] Martinez, M.; Stiefelhagen, R.: Taming the Cross Entropy Loss. *CoRR*, ročník abs/1810.05075, 2018, 1810.05075.
URL http://arxiv.org/abs/1810.05075

[27] mathworks: Convolutional Neural Network. 2019, [Online; navštíveno 16-01-2019].
URL https://www.mathworks.com/solutions/deep-learning/convolutional-neural-network.html

[28] Nekrasov, V.; Shen, C.; Reid, I. D.: Light-Weight RefineNet for Real-Time Semantic Segmentation. *CoRR*, ročník abs/1810.03272, 2018, 1810.03272.
URL http://arxiv.org/abs/1810.03272

[29] Neuhold, G.; Ollmann, T.; Bulò, S. R.; aj.: The Mapillary Vistas Dataset for Semantic Understanding of Street Scenes. *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017: s. 5000–5009.

[30] Peng, C.; Zhang, X.; Yu, G.; aj.: Large Kernel Matters – Improve Semantic Segmentation by Global Convolutional Network. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[31] Pohlen, T.; Hermans, A.; Mathias, M.; aj.: Full-Resolution Residual Networks for Semantic Segmentation in Street Scenes. *CoRR*, ročník abs/1611.08323, 2016, 1611.08323.
URL http://arxiv.org/abs/1611.08323

[32] 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.

[33] 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.

[34] Ronneberger, O.; Fischer, P.; Brox, T.: U-Net: Convolutional Networks for Biomedical Image Segmentation. *CoRR*, ročník abs/1505.04597, 2015, 1505.04597.
URL http://arxiv.org/abs/1505.04597

[35] Ros, G.; Sellart, L.; Materzynska, J.; aj.: The SYNTHIA Dataset: A Large Collection of Synthetic Images for Semantic Segmentation of Urban Scenes. 2016.

[36] Russakovsky, O.; Deng, J.; Su, H.; aj.: ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, ročník 115, č. 3, 2015: s. 211–252, doi:10.1007/s11263-015-0816-y.

[37] Sandler, M.; Howard, A.; Zhu, M.; aj.: MobileNetV2: Inverted Residuals and Linear Bottlenecks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[38] Sherstinsky, A.: Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network. *CoRR*, ročník abs/1808.03314, 2018, 1808.03314.
URL http://arxiv.org/abs/1808.03314

[39] Siam, M.; Gamal, M.; Abdel-Razek, M.; aj.: RTSeg: Real-Time Semantic Segmentation Comparative Study. In *2018 25th IEEE International Conference on Image Processing (ICIP)*, Oct 2018, ISSN 2381-8549, s. 1603–1607, doi:10.1109/ICIP.2018.8451495.

[40] Smith, L. N.: No More Pesky Learning Rate Guessing Games. *CoRR*, ročník abs/1506.01186, 2015, 1506.01186.
URL http://arxiv.org/abs/1506.01186

[41] Szegedy, C.; Ioffe, S.; Vanhoucke, V.: Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. *CoRR*, ročník abs/1602.07261, 2016, 1602.07261.
URL http://arxiv.org/abs/1602.07261

[42] Szegedy, C.; Liu, W.; Jia, Y.; aj.: Going Deeper with Convolutions. *CoRR*, ročník abs/1409.4842, 2014, 1409.4842.
URL http://arxiv.org/abs/1409.4842

[43] Szegedy, C.; Liu, W.; Jia, Y.; aj.: Going Deeper with Convolutions. *CoRR*, ročník abs/1409.4842, 2014, 1409.4842.
URL http://arxiv.org/abs/1409.4842

[44] Szegedy, C.; Vanhoucke, V.; Ioffe, S.; aj.: Rethinking the Inception Architecture for Computer Vision. *CoRR*, ročník abs/1512.00567, 2015, 1512.00567.
URL http://arxiv.org/abs/1512.00567

[45] Valada, A.; Vertens, J.; Dhall, A.; aj.: AdapNet: Adaptive Semantic Segmentation in Adverse Environmental Conditions. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2017, s. 4644–4651.

[46] Zhao, H.; Shi, J.; Qi, X.; aj.: Pyramid Scene Parsing Network. *CoRR*, ročník abs/1612.01105, 2016, 1612.01105.
URL http://arxiv.org/abs/1612.01105

[47] Zhou, Z.; Siddiquee, M. M. R.; Tajbakhsh, N.; aj.: UNet++: A Nested U-Net Architecture for Medical Image Segmentation. *CoRR*, ročník abs/1807.10165, 2018, 1807.10165.
URL http://arxiv.org/abs/1807.10165

Příloha A

Obsah přiloženého paměťového média

- **dataset_examples**: ukázky datových sad
- **src**: zdrojové soubory systému pro trénování SS-CNN
- **src_latex**: zdrojové soubory dokumentace
- **documentation.pdf**: dokumentace diplomové práce
- **SeparNet.avi**: video reprezentující výsledky práce

Příloha B

Slovník

B.1 Seznam pojmů

1. **Hyper-parametry** - statické parametry, které např. při trénování určují rychlost trénování, počet iterací, případně počet epoch, apod. Nebo se může jednat o statické hodnoty nastavení jednotlivých vrstev (u konvoluční sítě je to velikost jádra, kroku, dilatace nebo zarovnání).
2. **Přeučení sítě** - stav sítě, kdy síť nedokáže klasifikovat nová data, pouze jen ta, na kterých byla učena.
3. **Feature map** - Mapa aktivací, tensor významných vlastností vstupu sítě.
4. **Feature extractor** - Konvoluční část sítě, která získává z vstup(ů) mapu aktivací.
5. **Podvzorkování:**
   - Poměr velikosti vstupního obrazu k výstupní aktivační mapě.
   - Definuje kolikrát se vstupní obraz zmenší když projde sítí.
   - **Příklad:** Při podvzorkování 16 bude mít vstupní obraz o rozměru 224x224x3 aktivační mapu velikosti 224/16 = 14 (14x14).
6. **Dense prediction** - jedná se o predikování popisku pro každý pixel scény obrázku.
7. **Batch** - počet vzorků zpracovávaných naráz při jednom průchodu sítí.
8. **Iterace** - průchod jednoho batche sítí při trénování.
9. **Epocha (Trénování)** - udává počet iterací nutných pro průchod všech vzorků datové sady sítí při trénování.

B.2 Seznam zkratek

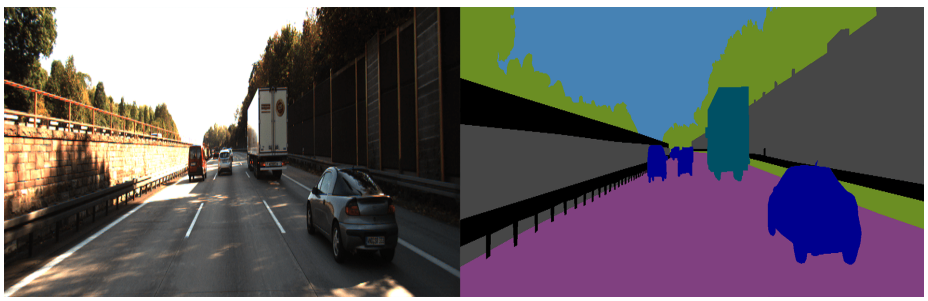
1. **CNN** - Convolutional Neural Network
2. **SS-CNN** - Convolutional Neural Network for Semantic Segmentation
3. **ReLU** - Rectified Linear Unit

Příloha C

Ukázky datových sad

Obrázek C.1: Ukázka datové sady CamVid [6].

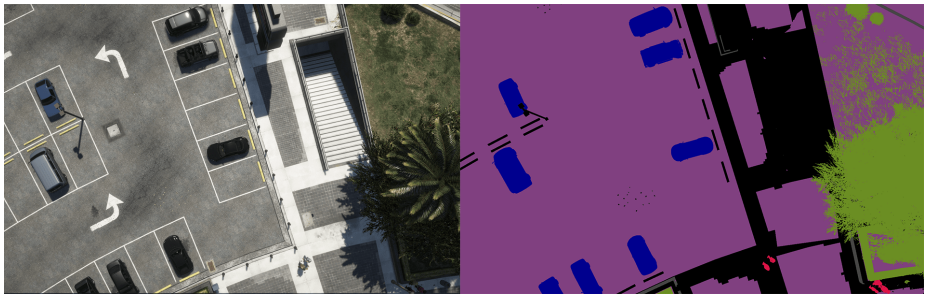
Obrázek C.2: Ukázka datové sady CityScapes [9].



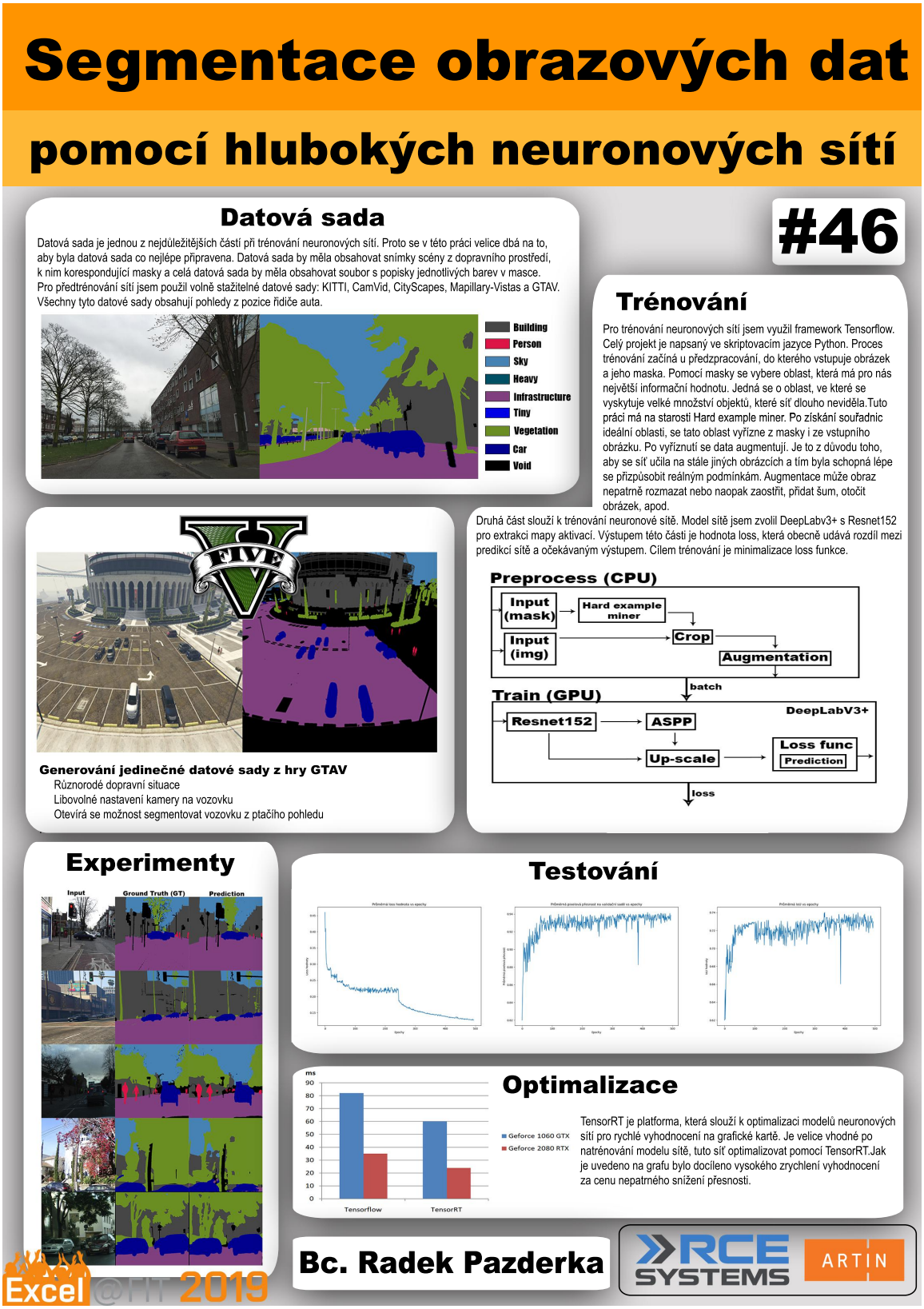
Obrázek C.3: Ukázka datové sady KITTI [11].

Obrázek C.4: Ukázka datové sady Mapillary Vistas [29].

Obrázek C.5: Ukázka datové sady GTAV street [33].



Obrázek C.6: Ukázka datové sady GTAV-custom.



Obrázek C.7: Plakát reprezentující výsledky práce.