VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

ROZPOZNÁNÍ HRANIC JÍZDNÍHO PRUHU V ZÁBĚRECH PALUBNÍ KAMERY

RECOGNITION OF DRIVING LANE BORDERS IN VIDEO FROM ON-BOARD CAMERA

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **DAVID FRIDRICH**
AUTHOR

VEDOUCÍ PRÁCE **prof. Ing. ADAM HEROUT, Ph.D.**
SUPERVISOR

BRNO 2022

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2021/2022

Zadání bakalářské práce

Student: **Fridrich David**
Program: Informační technologie
Název: **Rozpoznání hranic jízdního pruhu v záběrech palubní kamery**
Recognition of Driving Lane Borders in Video from On-Board Camera
Kategorie: Zpracování obrazu

Zadání:

1. Seznamte se s problematikou počítačového vidění a strojového učení, zaměřte se na techniky relevantní pro řešenou úlohu.
2. Vyhledejte a popište aktuální techniky a datové sady pro detekci jízdních pruhů.
3. Experimentujte s technikami detekce a lokalizace jízdních pruhů v obraze z palubní kamery, iterativně vylepšujte vlastní řešení.
4. Pořiďte vlastní datovou sadu vhodnou pro vyhodnocení vlastností vytvořeného řešení.
5. Experimentujte s vlastními daty a iterativně vylepšujte svoji datovou sadu a své řešení.
6. Zhodnoťte dosažené výsledky a navrhněte možnosti pokračování projektu; vytvořte plakátek a krátké video pro prezentování projektu.

Literatura:

- Goodfellow, Bengio, Courville: Deep Learning, MIT Press, 2016
- Xu et al.: SALMNet: A Structure-Aware Lane Marking Detection Network, IEEE T-ITS, 2021
- Bharath Ramsundar, Reza Bosagh Zadeh: TensorFlow for Deep Learning: From Linear Regression to Reinforcement Learning, O'Reily Media, 2018
- Gary Bradski, Adrian Kaehler: Learning OpenCV; Computer Vision with the OpenCV Library, O'Reilly Media, 2008
- Richard Szeliski: Computer Vision: Algorithms and Applications, Springer, 2011

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 a 2, značné rozpracování bodů 3 až 5.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Herout Adam, prof. Ing., Ph.D.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2021
Datum odevzdání: 11. května 2022
Datum schválení: 1. listopadu 2021

Abstrakt

Tato práce se zabývá detekcí jizdních pruhů. Konkrétně vlastního generátoru syntetických dat, jeho využití při trénování neuronových sítí, testování výsledků na konvoluční neuronové síti (CNN) modelu UNet a možnosti rozšíření tohoto modelu na SALMNet model (Structure-Aware Lane Marking Detection Network) pomocí přidání SGCA modulu (Semantic-Guided Channel Attention) a pyramidového modulu PDC (Pyramid deformable convolution). Výsledky trénování syntetických dat ukazují že síť umí rozpoznávat silniční pruhy velmi dobře, s přesností kolem 95 % (na jednodušších obrázcích dosahuje i 99 %). Nad reálným datasetem se výsledky lišily pro jednotlivé datasety, TuSimple dosahoval větší přesnosti kvůli menší obtížnosti obrázků, a sice kolem 62 %. Datová sada CuLane dosahovala pouze kolem 37 % velmi nestabilně.

Abstract

This paper talks about lane detection. Specifically custom generator of synthetic images, usage during training of neural networks, testing on convolutional neural network (CNN) UNet model and possibilities of extension of this model to SALMnet (Structure-Aware Lane Marking Detection Network) via addding SGCA module (semantic-guided channel attention) and PDC module (pyramid deformable convolution). Training results from synthetic datasets show very accurate results, reaching around 95 % in accuracy (even 99 % for easier images). Trainings with real datasets show lower accuracy, depending on the difficulty of the dataset itself. TuSimple has easier and clearer images and reaches about 62 %. CuLane is much more complex and results show accuracy around 37 %.

Klíčová slova

generátor, počítačové učení, rozpoznávání jízdních pruhů, neuronová síť, UNet

Keywords

generator, computer learning, lane detection, neural network, synthetic dataset, UNet

Citace

FRIDRICH, David. *Rozpoznání hranic jízdního pruhu v záběrech palubní kamery.* Brno, 2022. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce prof. Ing. Adam Herout, Ph.D.

Rozpoznání hranic jízdního pruhu v záběrech palubní kamery

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Adama Herouta a uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
David Fridrich
10. května 2022

Poděkování

Chtěl bych poděkovat svému vedoucímu za skvělé vedení, rady a hlavně trpělivost, kterou se mnou měl v celém průběhu dvou semestrů.

Obsah

Kapitola 1

Úvod

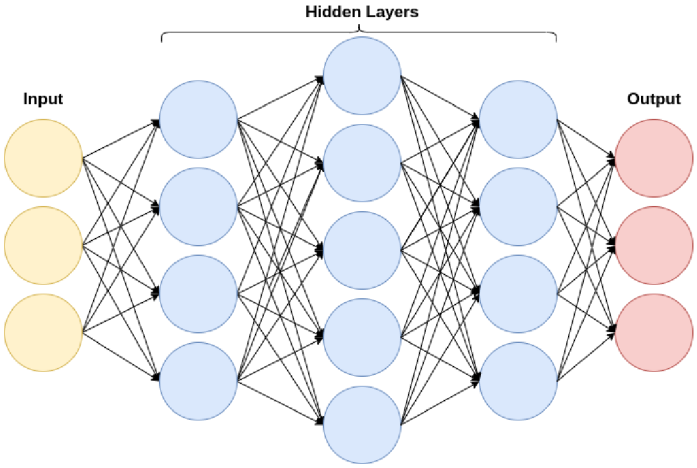
Cílem této práce je seznámit se s modely pro detekci jízdních pruhů, vytvořit generátor syntetických obrázků, experimentovat s možnostmi použití tohoto syntetického datasetu pro samostatné trénování nebo předtrénování a otestovat model i nad reálnými daty. Generátor se může využít pro generaci obrázků, které budou lehčí na naučení pro neuronové sítě než při využití reálného datasetu, tedy třeba pro novou vlastní neuronovou síť. Generátor dále umí generovat obrázky se stupňující obtížností dané vstupními parametry zadané uživatelem, je tedy možné zkoumat přesnost sítě při použití různých kombinací parametrů, nebo konvertovat aktuální reálné datové sady (CuLane a TuSimple) a testovat nad nimi.

Kapitola 2

Neuronové sítě pro detekci jízdních pruhů

Neuronové sítě, nebo kompletně Umělé neuronové sítě (Artificial Neural Networks) jsou výpočetní systémy, které jsou inspirovány biologickými neuronovými sítěmi z lidského mozku. Je to tedy pokus o napodobení fungování a učení nás, lidí.

Při detekci jízdních pruhů se hojně využívají konvoluční neuronové sítě (anglicky CNN – Convolution neural network). Patří mezi takové tři základní typy. Další jsou ANN (Artificial neural network) a RNN (Recurrent neural network). V této práci se zaměřuji na CNN právě protože se používají ve větší míře pokud jde o detekci jizdních pruhů. Celkově jsou vhodné pro klasifikaci a zpracování obrázků kvůli tomu, že dosahují větší přesnosti. Můžeme se samozřejmě setkat i s „mutacemi“ různých modelů dohromady, jako např. ConvLSTM [9].



Obrázek 2.1: Ukázka neuronové sítě neuronů hlubokého učení [3], několik skrytých vrstev zasebou

2.1 Konvoluční neuronová síť

Konvoluční neuronové sítě (dále jen CNN – convolutional neural network) [6] se využívají např. pro obrazovou segmentaci, klasifikaci, detekci objektů nebo rozpoznávání obličejů. Díky využití manipulacemi s maticemi, které jsou v CNN využívány (jako maticové násobení), jsou tyto sítě lepší pro identifikaci, nebo filtraci různých specifických vzorů v obraze.

Na druhou stranu jsou zase výpočetně náročnější. Standardně se takové sítě rozdělují do tří vrstev: Konvoluční (convolutional layer), sdružovací (pooling layer) a aktivační (activation function).

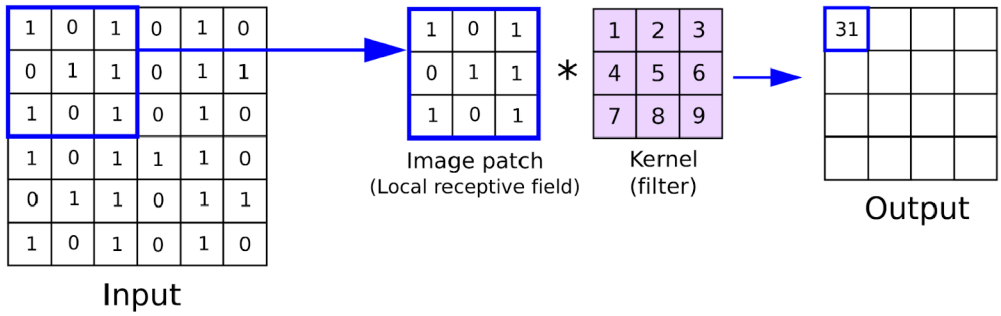
2.1.1 Konvoluční vrstva

Konvoluční vrstva je základní vrstva v konvoluční síti (také se podle toho jmenuje) a obecně slouží pro extrakci vlastností (feature extraction) z obrázku. Zde se provádí skalární součin dvou matic. Jedna matice obsahuje parametry, které chceme získat. Je to filtr (také se tomu říká kernel (kapitola 2.1.1)), který použijeme opakovaně na nějakou vstupní matici většího rozměru (např. když obrázek rozložíme na pixely) abychom vytvořili novou matici, která bude reprezentovat výskyt parametru který jsme filtrovali. Tato nová matice se nazývá aktivační mapa.

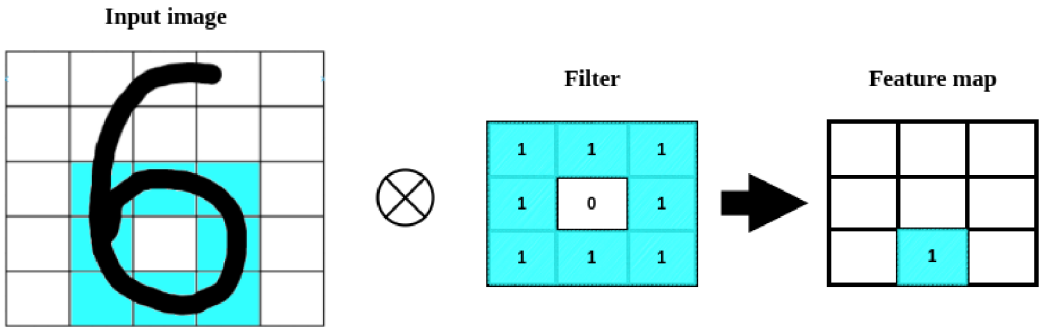
Mapa vlastností (feature map) je to stejné jako aktivační mapa (activation map). Je to výsledek aplikace filtru na vstupní obrázek – může být i předchozí aktivační mapa, v reálných případech se celý proces opakuje vícekrát.

Kernel

Kernel neboli filtr je matice např. 3×3, která detekuje nějakou vlastnost (obrázek 2.3). Filtr „přejede“ přes celý vstupní obrázek, vždy s posunutím o krok (anglicky stride) a vytvoří tím aktivační mapu.



Obrázek 2.2: Ukázka filtru. Filtr je postupně posouván přes celý vstupní obrázek (input) a je z něho vytvořena aktivační mapa (output). Převzato z: https://analyticsindiamag.com/what-is-a-convolutional-layer/

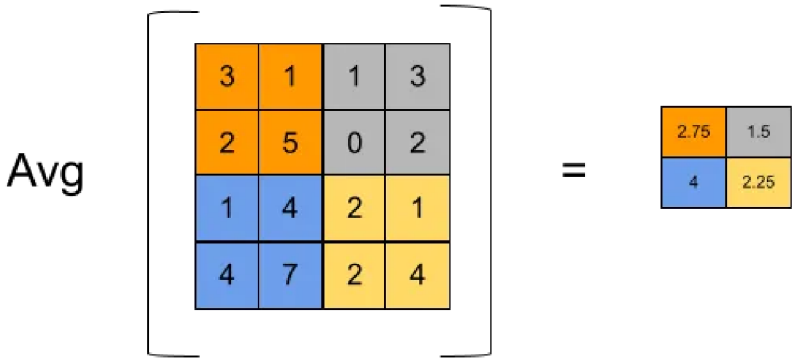


Obrázek 2.3: Konkrétní ukázka filtru. Vstupní obrázek je obrázek čísla 6. Filtr je „kruh“, hledám tedy kruh v vstupním obrázku (např. v čísle 6, 8 nebo 9). Jestliže na vstupním obrázku je hodnota 0 všude, kde není černá čára a 1 všude, kde je černá čára, potom aplikuji filtr na spodní střední část (podle modré barvy), tak získám přesnou shodu – hodnota 1 v aktivační mapě

2.1.2 Sdružovací vrstva

Sdružovací vrstva (lépe známá jako pooling layer) je další vrstva v konvolučním modelu. Hlavní smysl této vrstvy je zmenšování velikosti. Konvoluce je výpočetně náročnější, proto se snažíme zmenšit obrázek ale zanechat vlastnosti. Tuto vrstvu aplikujeme např. po filtru, když dostaneme aktivační mapu, můžeme sloučit několik okolních hodnot a vytvořit novou, menší aktivační mapu.

Jsou standardní typy: Průměrná sdruženost (average pooling) a maximální sdruženost (maximum pooling) a minimální sdruženost (Min pooling). Další možné rozdělení je globalní a lokální, které určují na co se shlukování bude aplikovat.



Obrázek 2.4: Average pooling. Shlukuje několik hodnot dohromady tím, že udělá jejich průměrnou hodnotu

Podobně jako na obrázku 2.4 (obrázek převzat z androidkt.com[1]), Max pooling vybere maximální hodnotu z množiny a Min pooling tu nejmenší.

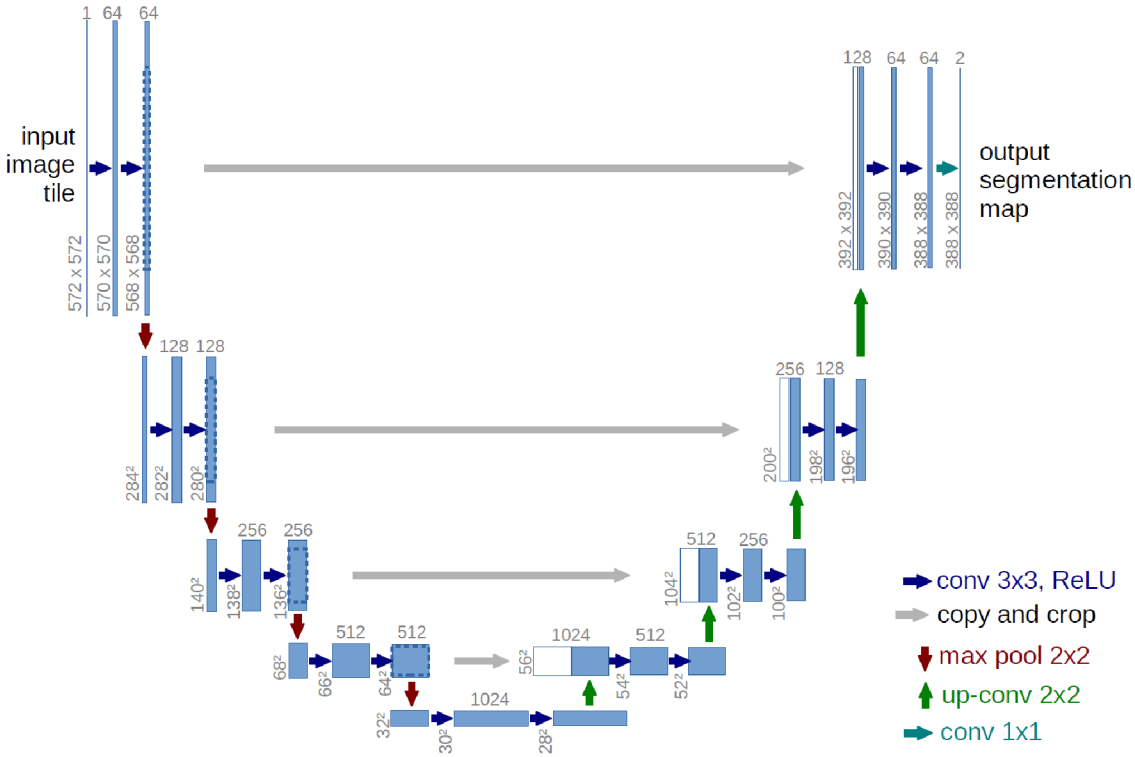
[1] https://androidkt.com/explain-pooling-layers-max-pooling-average-pooling-global-average-pooling-and-global-max-pooling/

2.1.3 Aktivační funkce

Aktivační funkce je většinou poslední krok nebo vrstva neuronové sítě. Přehled a popis aktivačních funkcí je více popsán v článku [2].

2.2 U-Net model

V této práci jsem použil model U-Net [8] konvoluční neuronové sítě[2].



Obrázek 2.5: Architektura modelu UNet. Ukázka s velikostí 32×32 pixelů v nejmenším rozlišení. Každý modrý rámeček symbolizuje více kanálovou mapu vlastností (počet kanálů je uveden nad rámečkem). V levém dolním rohu je uvedena velikost $x \times y$. Bílý rámeček označuje pouze zkopírovanou mapu vlastností a šipky označují různé operace popsané vpravo vedle modelu [8].

Architektura modelu ilustrovaná na obrázku 2.5 má celkově 23 konvolučních vrstev a skládá ze dvou částí.

Levá klesající část je stejná jako standardní architektura konvoluční sítě. Skládá se ze dvou konvolucí o velikosti 3×3, za kterými následuje aktivační funkce ReLU a vrstva max pooling (kapitola 2.1.2) o velikosti 2×2 s krokem (stride) o velikosti 2. Tento proces se několikrát opakuje. Při každém tomto kroku se zdvojnásobí počet kanálů aktivační mapy.

V pravé části se expanzuje. Nejprve se zvětší počet vzorků aktivační mapy, potom se provede konvoluce 2×2, která zmenší počet kanálů aktivační mapy na polovinu (opak oproti levé části), zřetězí se s aktivační mapou z levé části, která je na stejné úrovni a provedou se dvě konvoluce o velikosti 3×3 (za každou následuje ReLU funkce). V poslední

[2]zdroj modelu: https://github.com/milesial/Pytorch-UNet

vrstvě se provede konvoluce 1×1 kvůli namapování každého vektoru (z mapy vlastností) na požadovaný počet tříd.

2.2.1 Augmentace

Augmentace slouží pro přidání další úrovně obtížnosti pro neuronovou síť. Mezi plusy pro použití augmentací patří zvýšení velikosti datové sady, různorodost, zamezení *overfitting*[3] problému a zvýšení přesnosti. Pro vytvoření augmentací byla použita knihovna *Albumentations* (kapitola 2.3.3). Samotné augmentace jsem implementoval až při učení tzn. obrázky jsou uloženy po vygenerování v normálním stavu. Augmentace jsou aplikovány každou epochu zvlášť pro vytvoření co nejvíce rozmanité datové sady (ilustrační obrázky v kapitole 6.1). Rozsáhlé informace o augmentacích pro hluboké učení mohou být nalezeny v článku [10].

2.3 Použité implementační nástroje

2.3.1 Python

Programovací jazyk Python je nejpoužívanější jazyk pro účely strojového učení [7]. Z velké části je to díky knihovnám, které je možné použít, např. `OpenCV` (kapitola 4.2.1) pro počítačové vidění, `numpy` pro práci s vícerozměrnými poly a maticemi nebo `PyTorch` (kapitola 2.3.2) pro strojové učení a práci s GPU.

2.3.2 Pytorch

PyTorch je open-source framework postavený na knihovně Torch, která se používá pro počítačové vidění, strojové učení primárně vyvíjená týmem z Facebooku. První verze byla zveřejněna v roce 2016 a od té doby mnohokrát vylepšena. Velmi cenným přínosem této knihovny je integrace CUDA a možnost využití GPU pro strojové učení, protože ve většině případů je potřeba obrovské množství dat jejichž zpracování zabere hodně času.

Dobrou alternativou je také knihovna TensorFlow s kombinací TensorBoard

2.3.3 Albumentations

Knihovna Albumentations [1] je velmi užitečná pro práci s augmentacemi obrázku. Knihovna obsahuje více než 70 různých možných augmentací mezi které patří např. rotace, rozmazání, světelná alternace nebo simulace slunce a vše je možné aplikovat s určitou pravděpodobností a také na více obrázků zároveň, což je naprosto nezbytné v případě rotace, kdy je potřeba použít identickou augmentaci jak na vstupní obrázek, tak na jeho masku.

[3] https://www.ibm.com/cloud/learn/overfitting

Obrázek 2.6: Ukázka augmentací použitých při trénování datasetu TuSimple (kapitola 5.2.1). Na prvním obázku je aplikována rotace na obrázek i jeho masku, a potom odraz slunce pouze na vstupní obrázek. Na druhé ukázce je rotace a náhodné stínování. Na třetím je pouze rozmazání, maska zůstala nezměněna

Výsledky jsou frekventovaně testovány pro rychlost a efektivitu a knihovna funguje s Python frameworky jako Pytorch nebo Tensforflow.

2.3.4 Wandb

Wandb je nástroj pro experimentování při úlohách, které se zabývají strojovým učením. Obsahuje snadné implementace zaznamenávání experimentů, výkonu a systémových metrik nebo výstupních souborů, porovnávání výsledků a jejich sdílení. Zaznamenané informace je možné vizualizovat na webové stránce[4], kde je možné běhy ukládat, organizovat a porovnávat mezi sebou.

[4] https://wandb.ai/site

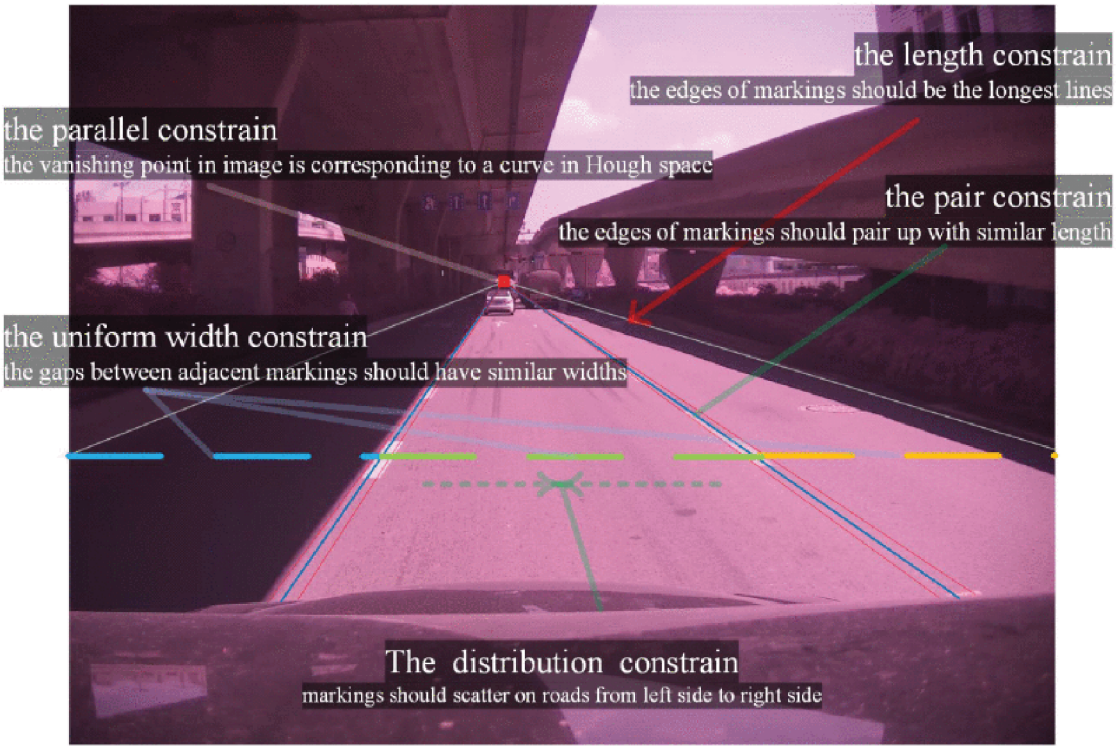
Kapitola 3

Jiné modely a řešení detekce jízdních pruhů

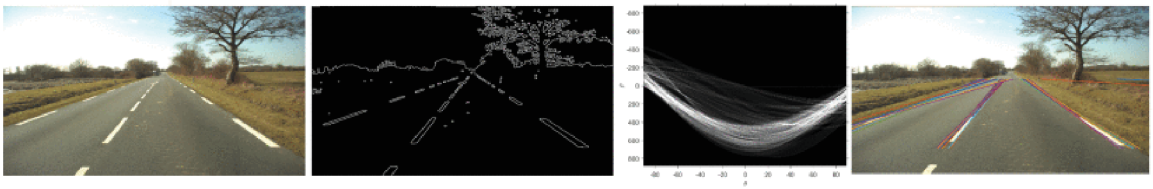
3.1 Hough transform space

Existuje spousta studií na téma detekce jednoho jízdního pruhu, narozdíl od detekce více pruhů. Toto je výhodnější pro autonomní navigaci, jízdy bez potřeby řidičů nebo využití tempomatu. Článek z roku 2020 [4] se zabývá novým a robustním algoritmem pro detekci více pruhů za jízdy, který je založený na získávání informací ze struktury silnice, která obsahuje 5 doplňkových omezení: délkové, paralelní, distribuční, párové a jednotné omezení šířky (uniform width constraint). Všech 5 omezení je začleněno do Houghovy transformace (HT) pro výběr kandidátních pruhů. Experimenty demonstrují, že tato metoda může fungovat lépe než jiné state–of–the–art metody jak v přesnosti tak v efektivitě. HT je klasicky metoda pro detekci rovných čar. Rozptýlí parametry pruhu jako diskrétní akumulátory v Houghově prostoru, kde každý pixel v prostoru „hlasuje“. Jeden hlas je přidán každému akumulátoru, pokud pro dané x a y následující rovnice platí.

$$\rho = x \cos\theta + y \sin\theta$$



Obrázek 3.1: 5 strukturálních omezení pro detekci pruhů v komplexních scénářích. Kvůli stínu jsou vytvořeny neexistující pruhy, ty jsou ale správně eliminovány.



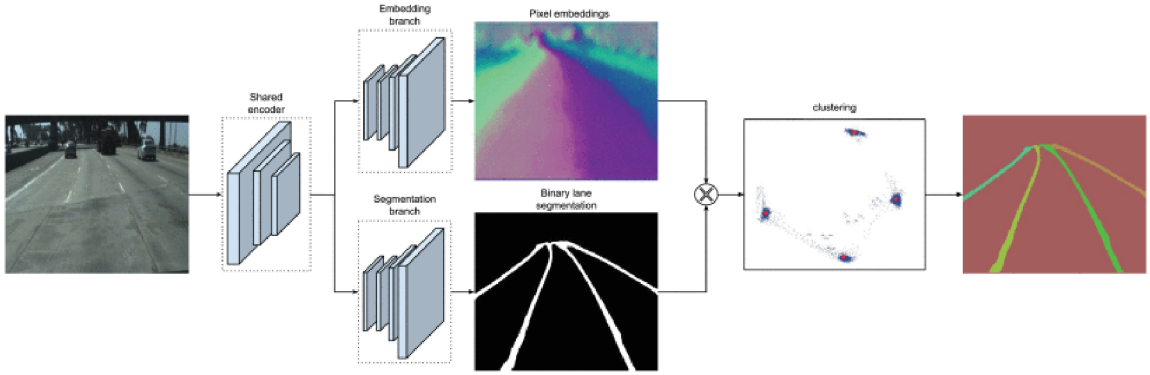
Obrázek 3.2: Originální obrázek (a), binární obrázek který obsahuje detekované hrany (b), akumulátor v Huoghovém prostoru se spoustou špatných detekcí(c), silniční označení identifikováno vrcholy v Houghovém prostoru (d).

Experimentování s modelem a porovnání s dalšími podobnými modely ukazuje větší přesnost tohoto modelu nad datasetem ROME.

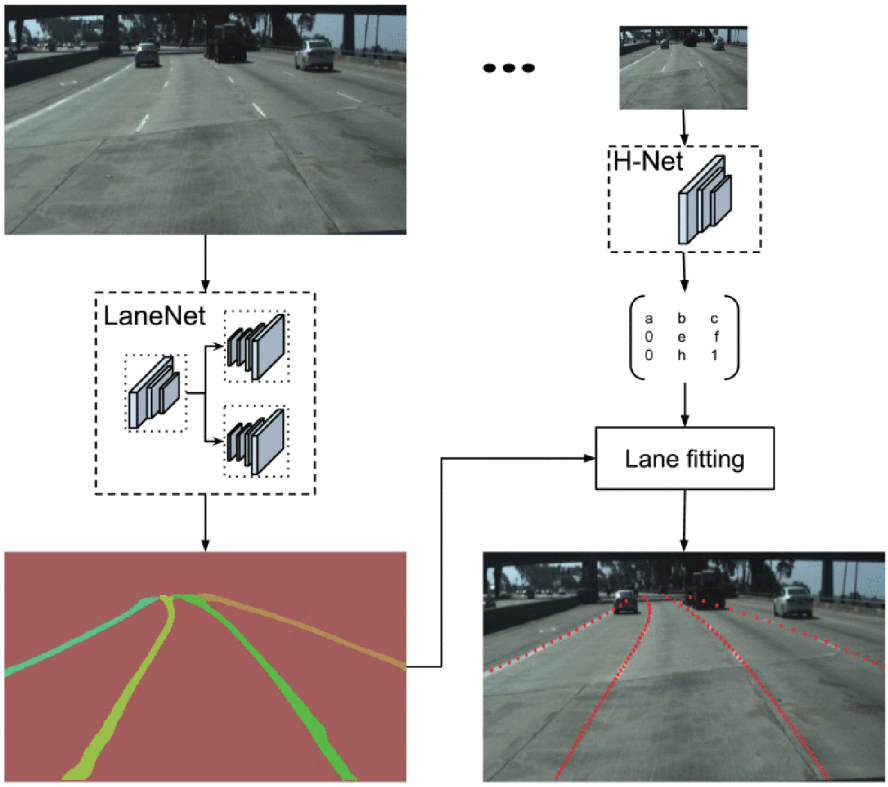
3.2 Lane-net

Další model, který je implementován pro detekci jízdních pruhů je Lane-net [5]. Tradiční metody detekce jízdních pruhů spoléhají na specifické, ručně vyrobené vlastnosti a heurestiky, po kterých následují techniky určené pro post–processing. Tento přístup je výpočetně velmi náročný. Novější přístupy se zabývají učení pro segmentaci každého pixelu i přesto, že v obraze nejsou žádné pruhy. Tyto metody jsou limitovány detekováním předdefinovaným, fixovaným počtem pruhů, nedokáží se vypořádat se změnou počtu pruhů. Tato metoda je

propozice pro chápání problému detekcí pruhů jako segmentační problém, ve kterém každý pruh je svoje vlastní instance, které může být vytrénovaná.



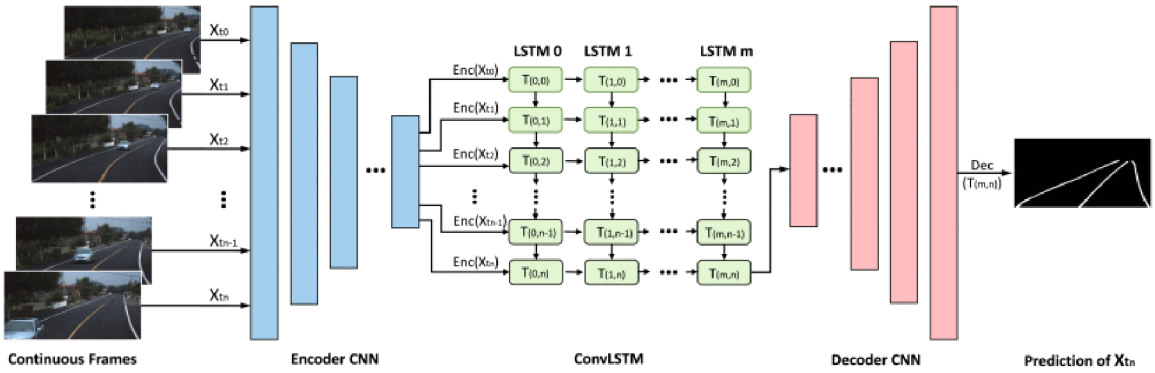
Obrázek 3.3: LaneNet architektura. Na spodní části je segmentační větev, která je trénována pro výpočet binárních masek. Vrchní část generuje N–dimenzionální vkládání co jeden pixel – takže pixely ze stejného pruhu jsou u sebe a od ostatních pruhů daleko.



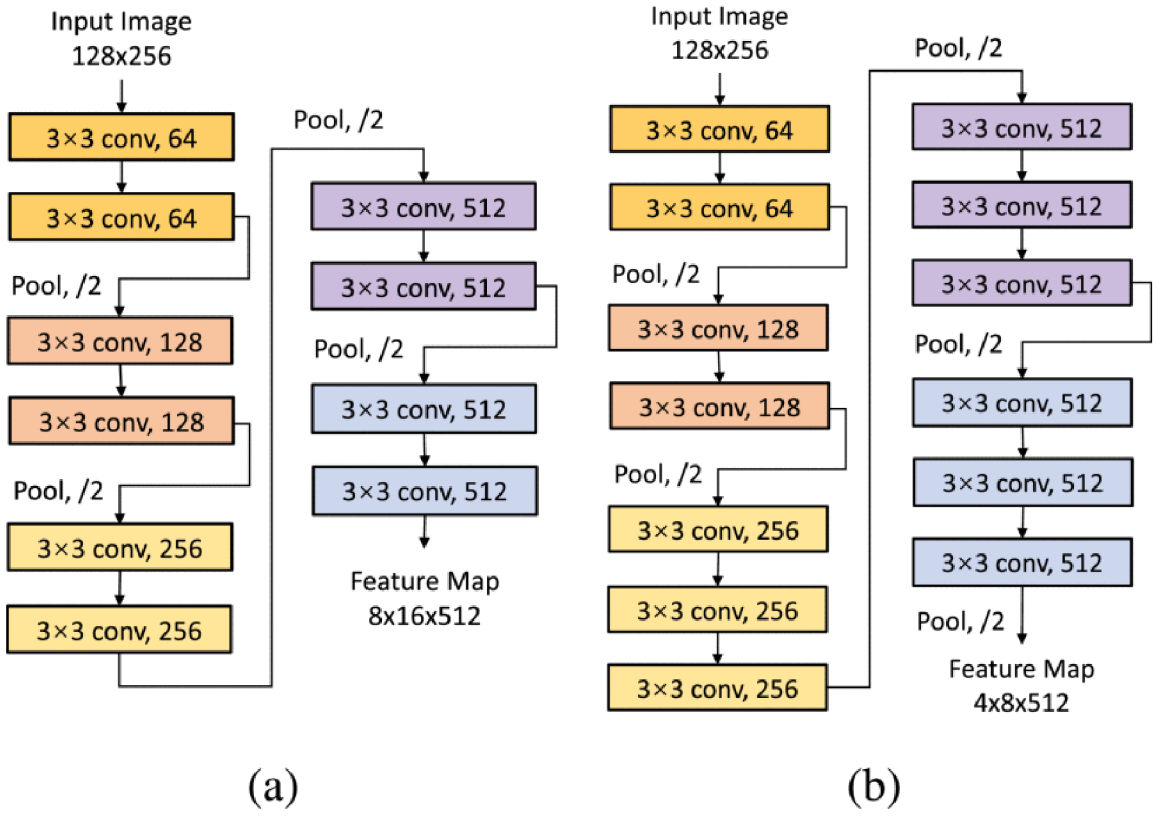
Obrázek 3.4: Přehled systému. Z vstupního obrázku LaneNet vyextrahuje instanci mapy tím, že označí každý pixel, který je součástí silničního pruhu podle ID pruhu. Pixely pruhů jsou transformovány pomocí transformační matice. Jsou výstupem H–Net modelu, který se učí perspektivní transformace podle vstupníhch obrázků.

3.3 ConvoLSTM

Článek [13] z roku 2019 povídá o možnosti detekce pruhů z více snímků jedné společné scény a vytváří hybridní architekturu kombinací konvoluční sítě (CNN) a rekurentní sítě (RNN). Konkrétně blok konvoluční sítě zaznamenává informace o každém snímku (v několika snímcích jdoucích po sobě). Tyto informace jsou následně použty jako vstup do RNN bloku pro učení vlastností (feature learning) a predikce pruhu. Experimenty na 2 velkých datasetech demonstrují, že tato metoda překoná podobné metody v detekci pruhů a to hlavně v komplexních situacích.



Obrázek 3.5: Architektura popisované sítě



Obrázek 3.6: Zakódovací síť v UNet–ConvLSTM (a) a SegNet–ConvLSTM (b).

Kapitola 4

Syntetický generátor datové sady

4.1 Funkcionalita

Tento generátor je program napsaný v jazyce Python (kapitola 2.3.1) pomocí knihovny OpenCV (kapitola 4.2.1) a několika pomocných knihoven. Program má 2 funkce, generátor a „konvertor“, který slouží pro konverzi anotací reálných datasetů (kapitola 5.2) na masky pro vstup programu neuronové sítě.

Generování

Primární funkce programu je samotný generátor, který dokáže generovat syntetické obrázky a jejich klasifikační masky. Využívá se zde tzv. „učení s učitelem“ (*supervised learning*)[1], proto jsou nutné vstupní masky. Klasifikační masky mohou být generovány jednobarevné či vícebarevné(rozlišujeme každý pruh zvlášť, např. pro rozlišování krajního pruhu a standardního přerušovaného pruhu nebo pozice pruhů v obraze vůči sobě)(kapitola 6.1), avšak při experimentech a trénování jsou v tomto projektu využity pouze jednobarevné masky.

Parametr	**Popis**
-m, --mirror	Vytvoří silniční pruhy obrazně kolem středu
-c, --count	Počet, kolik obrázků se má vytvořit
-s, --sky	Pokud je *true*, použij náhodou texturu nebe (výchozí: true)
-g, --grass	Pokud je *true*, použij náhodnou texturu trávy (výchozí: true)
-r, --road	Pokud je *true*, použij náhodnou texturu silnice (výchozí: true)
-d, --distortions	Určuje úroveň obtížnosti, možnosti: [0,1,2]
-t, --thickroad	Určuje tloušťku silničních čar
-g, --gauss	Pokud je větší než 0, vytvoří Gaussovo rozmazání pro masku
-o, --out	Složka do které se vytvoří struktura a uloží generované obrázky

Tabulka 4.1: Tabulka zobrazuje parametry generátoru, které lze využít pro generování nových obrázků

[1] https://www.ibm.com/cloud/learn/supervised-learning

Generátor je tedy možné využít pro generaci obrázků jako vstup neuronové sítě. Obrázky je možné generovat se škálující obtížností. Obrázky mají určité parametry, které jsou náhodně vybírány z intervalu pro zvýšení náhodnosti a různorodosti (obrázky zde 5.1). Pomocí parametru `--distortions` je možné regulovat úroveň obtížnosti.

Úplně nejjednodušší verze (úroveň 0) má pouze tenký pruh který označuje silniční čáru a k tomu 3 barvy symbolizující nebe, silnici a zemi/trávu (mimo silnici) (ukázka zde 5.5).

Při nastavení parametrů `-g -r -s` na `False` se využijí textury, které jsou uložené v podsložce `components`. Textury jsem vyhledal a získal na internetu (google images), snažil jsem se vybrat obrázky, které nemají autorská práva. Náhodně je pro každý obrázek vybrána textura pro oblast nebe, silnice a mimo silnice. (ukázka zde 5.6).

Úroveň 1 parametru `-d` je navíc zakřivení osy, která rozděluje horní a dolní část obrázku. Je přidán interval ze kterého se náhodně vybírá pro zvýšení náhodnosti a odlišnosti generovaných obrázků (ukázka zde 5.7).

Úroveň 2 mění tvar silničního pruhu. Použije se Gaussova křivka místo rovné čáry. Náhodnost je zde způsobena intervalem mezi středem obrázku, která určí jakým směrem a jak moc bude křivka zahnutá. (ukázka na obrázku zde 5.8).

Konverze

Parametr	**Popis**
dir	cesta složky, kde jsou uložené anotace pro konverzi
-d, --dataset_type	jednoslovný string s názvem datasetu
-y, --yes	pokud je zadán, automaticky odpověď na otázky ano (pro automatizaci)
-g, --gauss	pokud je větší než 0, určuje velikost kernelu gaussove masky
--dynamic_mask	určuje hodnotu, o kolik se bude měnit velikost dynamické masky
-n, --number	Max počet obrázků pro konverzi
-o, --out	Složka do které se uloží struktura s obrázkami a maskami (výchozí je .)
-r, --roadthickness	Určuje tloušťku silničních čar

Tabulka 4.2: Tabulka s parametry které lze použít při konverzi reálných datasetů na masky akceptovatelné použitým modelem U–Net

Vedlejší účel je konverze anotací aktuálních datasetů jako CULane[2] nebo TuSimple[3]. Každý dataset má specifický a lehce odlišný způsob ukládání informací o anotacích, které určují pozice silničních čar. Konverzí se vytvoří klasifikační masky, které sedí pro zde použitou implementaci U–Net modelu.

[2] https://xingangpan.github.io/projects/CULane.html
[3] https://github.com/TuSimple/tusimple-benchmark

4.2 Implementační nástroje

4.2.1 OpenCV

OpenCV[4] je knihovna, která implementuje funkce, které jsou převážné cílené na počítačové vidění. Knihovna má rozhraní pro jazyky Python, C++, Java a MATLAB, přičemž ji lze použít jak na Linuxu, Windowsu tak MacOS. Obsahuje více než 2500 optimalizovaných algoritmů pro state–of–the–art počítačové vidění a strojové učení. CUDA a OpenCL rozhraní jsou právě vyvíjeny a pravděpodobně bude možné tuto knihovnu brzy použít zcela samostatně pro účely ohledně strojového učení. Knihovna mimo jiné funguje skvěle pro zpracování obrazu, a proto ji v této práci využívám na čtení a zapisování obrázků, případnou manipulaci s nimi např. kvůli maskám, aplikaci Gaussova rozmazání, kreslení samotných čar nebo pro úpravy textur které používám.

[4] https://opencv.org/

Kapitola 5

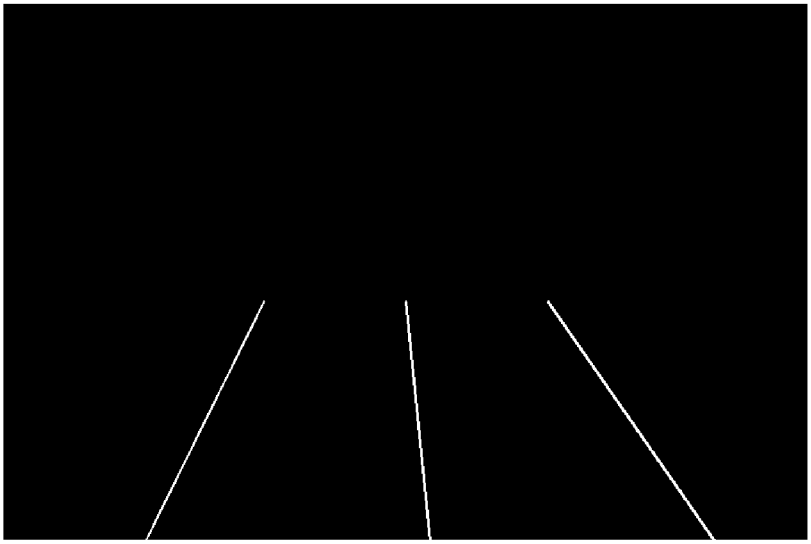
Použité datové sady

5.1 Generované syntetické obrázky

Tyto syntetické, generované obrázky jsou získány z generátoru (kapitola 4). Jsou seřazeny podle obtížnosti trénování, neboli komplexnosti obrázků. Úroveň se určuje parametrem *distortions*.

Ke každému obrázku se generuje jeho maska, jednou v RGB (pro možnost nahlédnutí) a podruhé pro klasifikaci (bez RGB složky). Při použití deformací (v generátoru je to parametr `--distortions`) je možné využít Bezierovu křivku na masky, aby se pruhy rozmazaly, tedy číslo blíž 1 znamená větší shodu a číslo blíž 0 znamená neshodu. Všechny parametry použité v následujících obrázcích jsou modifikovatelné, tedy velikost silniční čáry samotné, velikost kernelu gaussovského rozostření nebo dynamická maska.

5.1.1 Masky obrázků



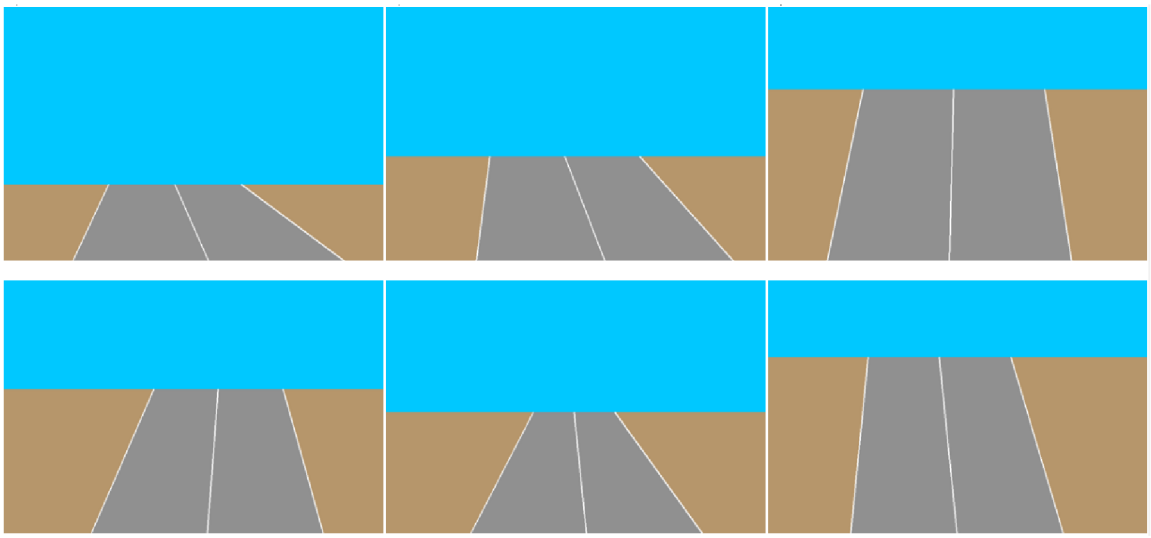
Obrázek 5.1: Ukázka nejjednodušší verze masky. Velikost silniční čáry je 2. Tuto velikost jsem po začátku již moc nepoužíval

Obrázek 5.2: Ukázka standardní trochu silnější silnice velikosti 5. Velikost 5 nebo 7 jsem použil jako standard.

Obrázek 5.3: Ukázka rozostřené masky. Velikost silniční čáry je 10. Je použito Gaussovské rozostření s kernelem o velikosti 17.

Obrázek 5.4: Ukázka masky, velikost 10, gaussovské rozostření s kernelem 17 a použití dynamické masky o velikosti 10. Silniční čára se zvětšuje o svoji velikost čím blíže je.

5.1.2 Generované obrázky



Obrázek 5.5: První generace obrázků, velmi jednoduché na trénování a naučení, obzvlášť s komplexnějším modelem jako je právě U–Net. Jediné čím se obrázky liší je „tvar“ silnice a výška linie která rozděluje nebe a zemi

Obrázek 5.6: Druhá generace. Přidání náhodných textur z výběru. Nebe, silnice a země, od každého cca 40 možností.

Obrázek 5.7: Třetí generace. Přidání nerovnosti v linii nebe – země.

Obrázek 5.8: Čtvrtá generace. Použití Bezierové křivky pro silniční pruhy

5.2 Reálné datasety

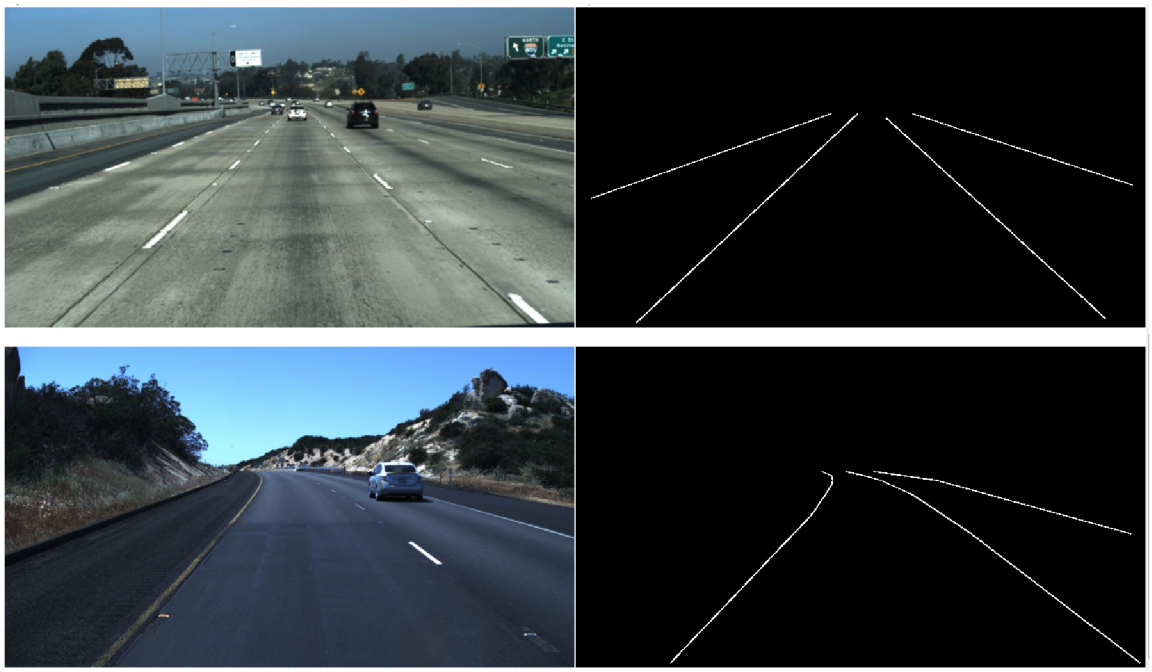
Vybral jsem 2 standardní datové sady Tusimple a Culane používané pro detekci jízdních pruhů. Obsahují obrázky a anotace v textové podobě. Jsou to souřadnice bodů/pixelů na obrázku, které označují silniční pruhy v daném obrázku. Konverze těchto textových souborů na masky se dá provést pomocí programu generátoru (jeho vedlejší funkce je konverze masek, popsána v kapitole 4).

5.2.1 TuSimple

TuSimple[1] je datová sada obsahující převážně dálniční silnice. Obrázky jsou převážně čisté, pruhy jsou různorodé: plné čáry, přerušované, někdy i bez bílých čar úplně, v tom případě jsou hranice pruhu označeny řezem v asfaltovém bloku a přerušovaným vysázením železných koulí přes které se dá přejet, ale řidič by měl poznat že přejíždí pruh.

[1] https://paperswithcode.com/dataset/tusimple

Obrázek 5.9: Náhodný výběr obrázků. Celá složka obsahuje sérii podsložek s klipy. Každý klip má 20 snímků a z toho vždy poslední snímek je anotován. Při použití všech anotací je 3626 použitelných obrázků.



Obrázek 5.10: Přiřazené masky po konverzi. Anotace má přibližně 50 bodů, které konvertor spojí a přidá do černého obrazu jako bílou čáru s volitelnou velikostí (případně i s Gaussovým rozmazáním a dynamickou maskou)

5.2.2 CuLane

Datová sada CuLane [11] obsahuje obtížnější scénáře oproti předchozí TuSimple. Jsou to fotografie z města Beijing, obsahují více než 130 000 extrahovaných snímků. Silniční pruhy jsou anotovány v některých případech i přesto že na fotografii linie není vidět (např. kvůli

autu). Anotace jsou zaměřené na nejbližší 4 linie od řidiče, ostatní nejsou brány v úvahu vůbec (tedy aktuální a vedlejší pruhy z leva a z prava). Datová sada je volně dostupná[2] na internetu.

Obrázek 5.11: Ukázka obrázků a jejich konvertovaných masek připravené pro trénování z datové sady CuLane. Anotace jsou textového formátu. Jeden řádek v souboru obsahuje dvojice x – y souřadnic, který určují jednu silniční čáru v obraze – soubory obsahují většinou 3 až 4.

[2] https://paperswithcode.com/dataset/culane

Kapitola 6

Experimenty a trénování

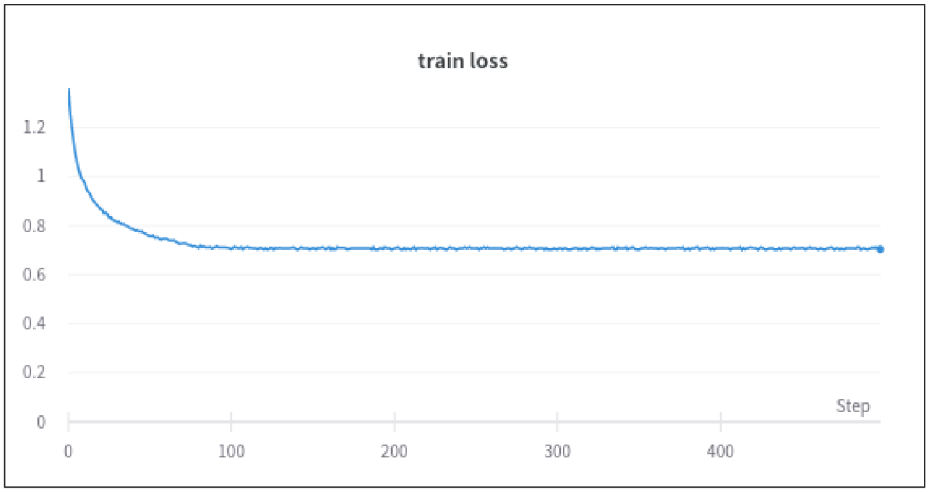
Trénování bylo nejdříve prováděno na mém systému Acer Predator s 8GB RAM a grafickou kartou NVIDIA GeForce GTX 1060. Později i na školním serveru *sophie* s 4 24GB grafickými kartami NVIDIA RTX A5000, kde proces trénování probíhal cca 8-10x rychleji a bylo možné trénovat mnohonásobně větší objem dat.

6.1 Nad syntetickým datasetem

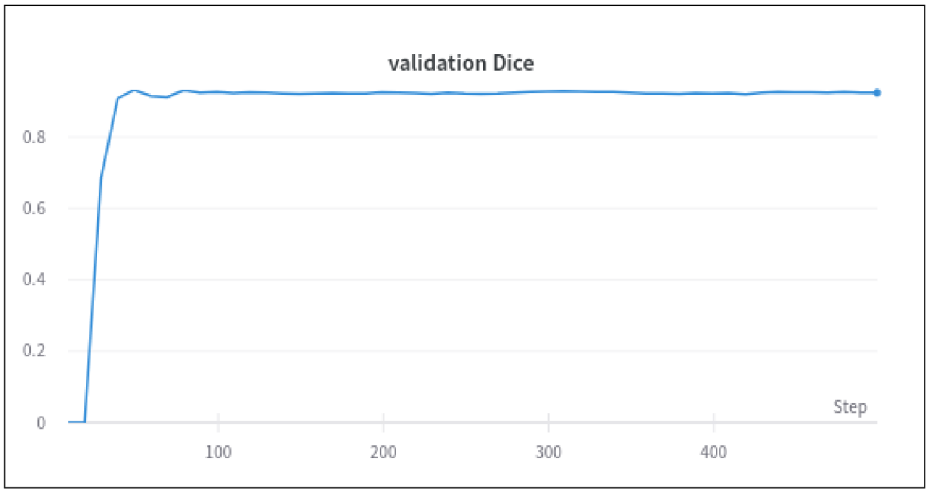
Prováděl jsem postupné trénování nad generátorem s různými parametry pro zjištění co dělá síti problém a naopak co jí pomáhá s přesností. Výsledky nad generátorem byly převážně velmi dobré, až výborné, tedy neuronová síť neměla problém s daty z generátoru. Narazil jsem i na nějaké speciální případy jako malá různorodost datasetu a přetrénování, které ukazuji níže.

V této sekci se používají termíny `training loss` a `validation score`. Training loss určuje hodnotu chyby trénovací datové sady – chceme se blížit nule. Validační skóre určuje přesnost předpokládané (naučené) masky vůči masce opravdové – chceme se blížit jedničce.

První trénování bylo provedeno s parametry `--distortions 0 -r -g -s -t 2`, tedy úroveň obtížnosti 0 a bez použití textur s tloušťkou silnice 2 (obrázek 5.5). Počet vygenerovaných obrázků použitých pro trénování byl 100. (Všechny obtížnosti a parametry jsou popsané v sekci 4.1)



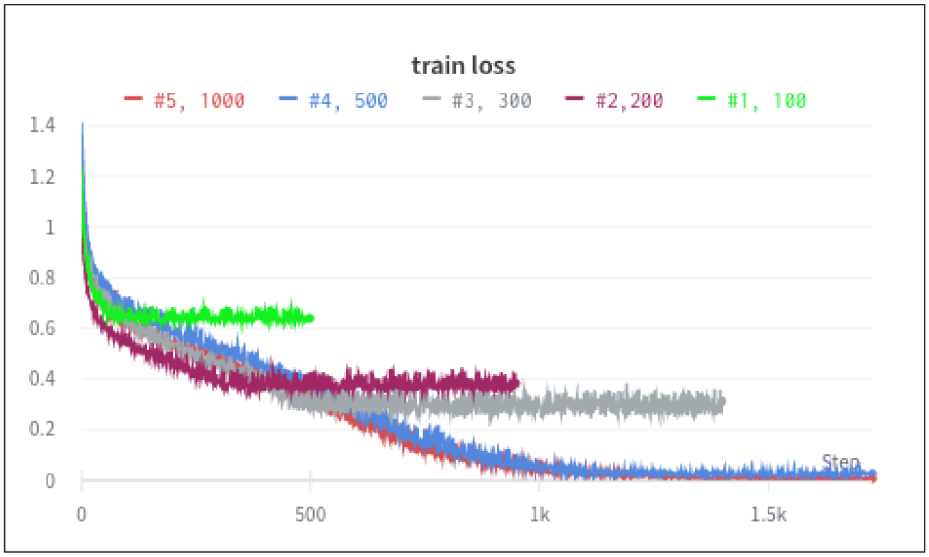
Obrázek 6.1: Train loss - ze začátku klesá, potom osciluje mezi 0,701 a 0,712 a dále už neklesá



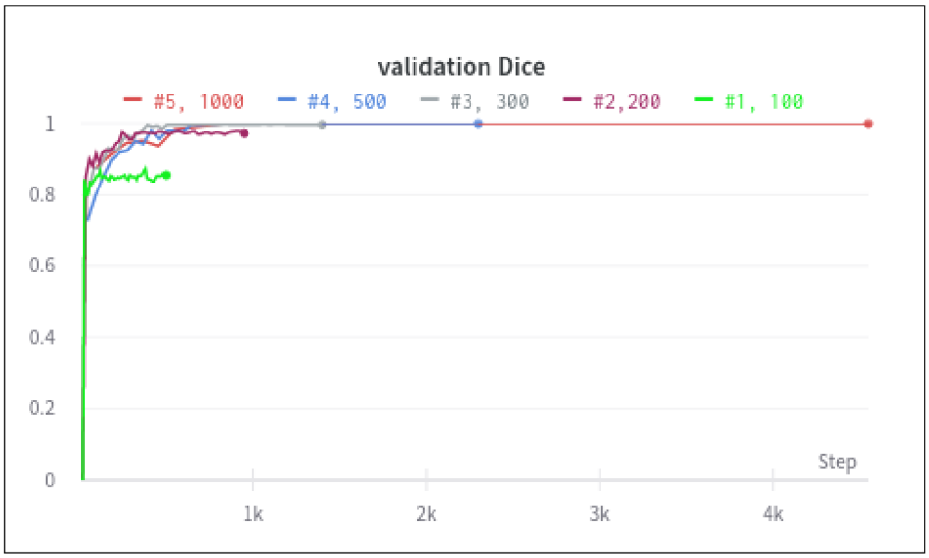
Obrázek 6.2: Validační skóre osciluje mezi 0,920 a 0,928

Z obrázku 6.1 je vidět, že chyba při trénování neklesá. Toto je způsobeno nedostatkem obrázků. Datová sada je příliš malá a obrázky jsou příliš podobné, a proto již není možné neuronovou síť nic naučit.

Druhý experiment navazuje na první. Testuje co se stane s neuronovou sítí při zvyšování počtu generovaných obrázků datové sady se stejnými parametry a kolik jich je potřeba pro překonání stagnace při trénování (kolik je potřeba obrázků aby trénovací chyba klesala dále). Obrázky 6.3 a 6.4 obsahují 5 běhů trénování. Všechny běhy spuštěny se stejnými parametry, liší se jen ve velikosti datasetu. Sada se 100 obrázky je označena zelenou barvou, s 200 obrázky vínovou, s 300 šedou, s 500 modrou a 1000 oranžovou.



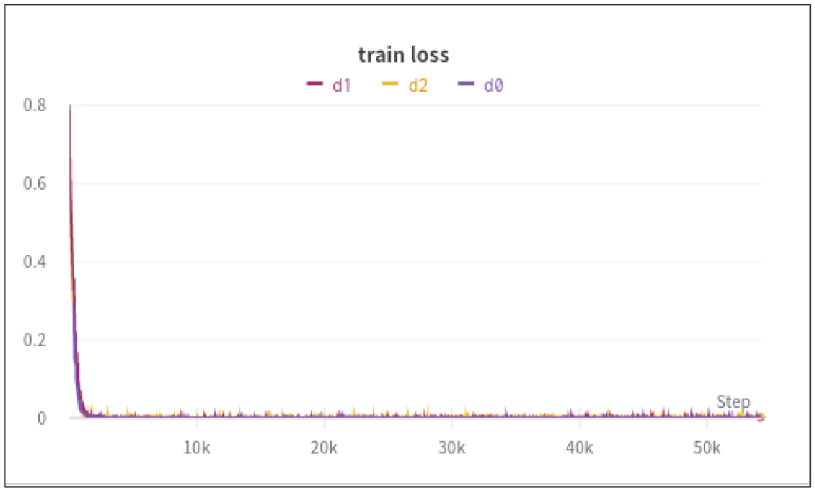
Obrázek 6.3: Všechny běhy běžely 5 epoch. Snižující křivka znamená zmenšování chyby. Běhy zelené, vínové a šedé barvy v určitém bodu stagnují z důvodu malé různorodosti datové sady.



Obrázek 6.4: Validační skóre určuje, jak moc se podobá předpovídaná maska masce opravdové. Dataset pouze se 100 obrázky má skóre menší, ostatní skóre kolem 0,9. Po kroku číslo 1000 všechny kolem 99 % přesnosti a sada s 1000 prvky dosáhla i čistě 100 %.

Porovnání běhů při použití všech úrovní `distortions` mezi sebou

Běhy spuštěné bez textur, s nejnižší úrovní obtížností jsou příliš jednoduché, dosahují přesnosti 100 % a proto je zde neukazuji. Každý běh má 3000 obrázků a běžel 20 epoch.

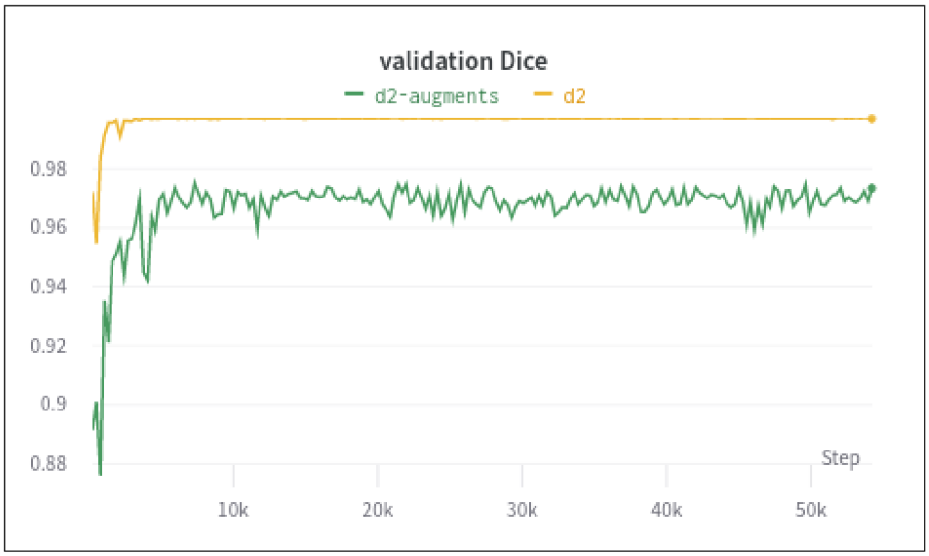


Obrázek 6.5: Train loss – označuje level obtížnosti pro běh. Train loss dosahuje konečné hodnoty kolem tisíciny.

Obrázek 6.6: Všechny běhy spolehlivě dosahují přesnosti 99 %.

První větší změny jsou vidět po aplikaci augmentací. Obrázky 6.7 a 6.8 ukazují rozdíly při použití augmentací pro trénování datové sady. Žlutá křivka zobrazuje běh s parametry `-d 2 -t 7`, stejný jako v obrázku 6.5

Obrázek 6.7: Chyba při trénování na konci u běhu s augmentacemi (zelená) je 0,01



Obrázek 6.8: Přesnost viditelně odlišná, stále velmi dobrá, spád o cca 2 desetiny na 0,97. Masky stále predikuje bez jakýchkoliv potíží.

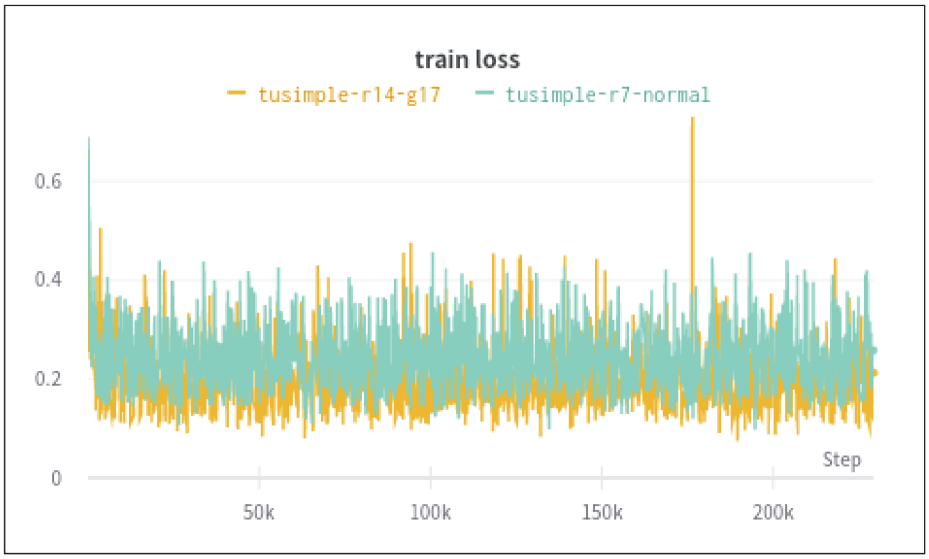
Obrázek 6.9:

Obrázek 6.10:

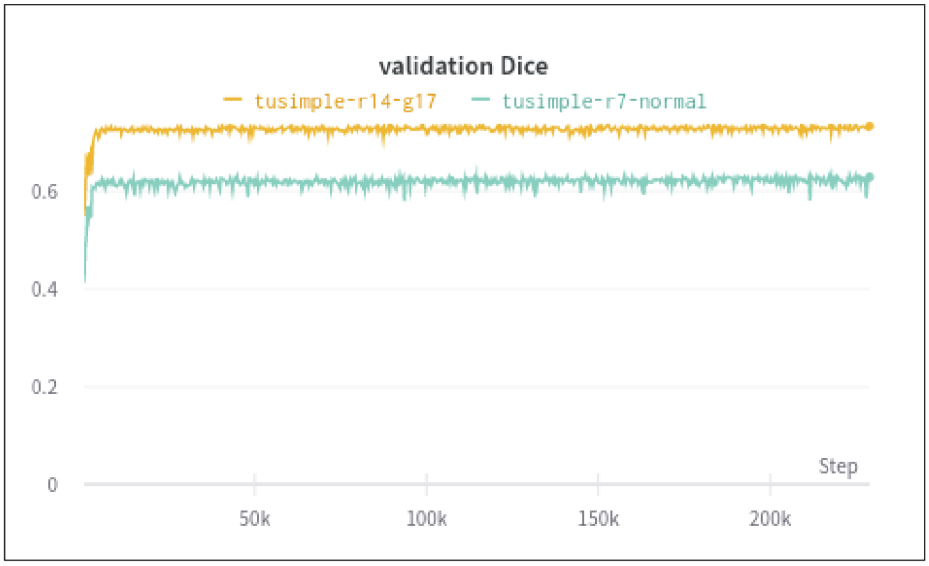
6.2 Nad reálnými datasety

6.2.1 TuSimple

Běhy s použitím datové sady TuSimple mají velikost maximální pro trénování 3626 a běžely s počtem epoch 70. V porovnání na obrázcích 6.11 a 6.12 je experiment trénování pro porovnání Gaussového rozmazání. Normální obrázky (zelená) mají velikost pruhu masky 7 a žádná další specializace není aplikována. Druhý běh (žlutá) má velikost pruhu 14, tedy dvojnásobek, protože je následně aplikováno Gaussovo rozmazání na tyto masky s očekáváním zlepšení přesnosti.



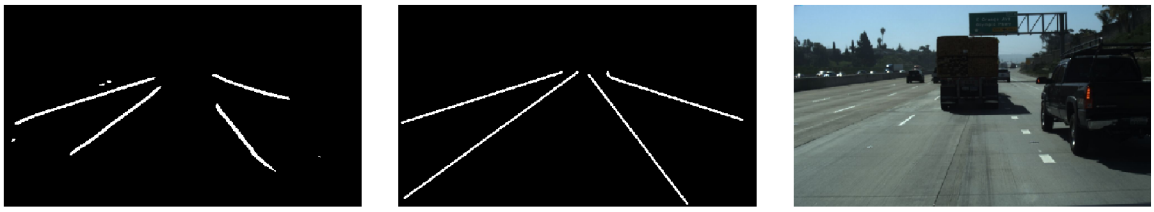
Obrázek 6.11: Trénovací chyba ukazuje hodnotu 0.285 ve finále pro normální běh (zelená) a hodnotu až 0.13 pro běh s Gaussově rozmazanou maskou (žlutá)



Obrázek 6.12: Validační skóre dosahuje 0.629 pro běh s normální maskou(zelená) o velikosti 7 bez Gaussova rozmazání a 0.733 s maskou velikosti 14 Gaussově rozmazanou s kernelem o velikosti 17

Dle výsledků je možné vidět, že běh s maskou na kterou byla aplikováno Gaussovo rozmazání ukazuje přesnější výsledky, přibližně o jednu desetinu.

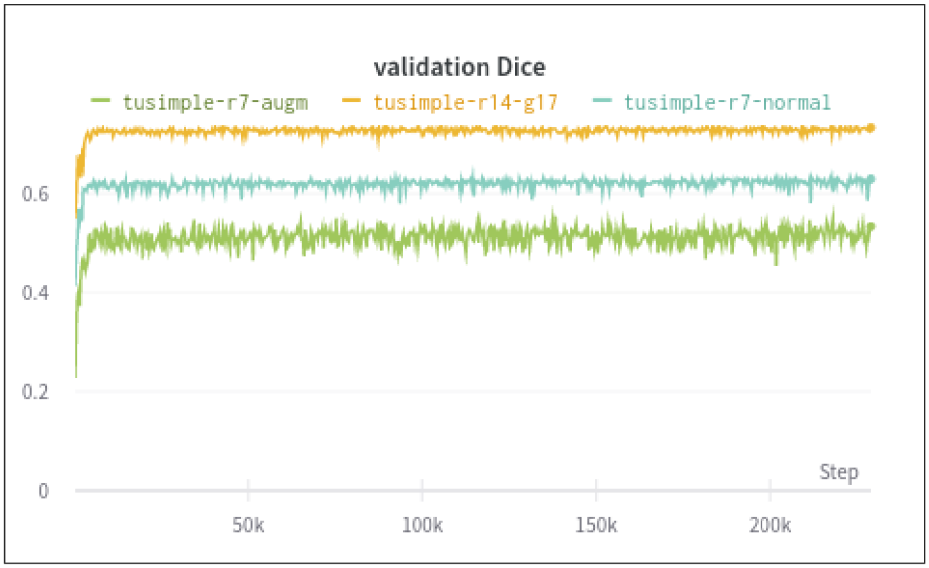
Mezi problémy, které se vyskytovali v datové sadě TuSimple patří např. přerušované čáry (obrázek 6.13). Nějakou část nepřesnosti také určuje anotace pruhů i přes to, že v obrázku nejsou zcela nebo úplně vidět (např. jede auto, tudíž chybí polovina čáry). Toto je případ který se ukazuje být obtížný i u CuLane datové sady. S tímto problémem by mohla pomoci implementace v následující práci, která je součástí SALMNet modelu, a sice PDC modul (kapitola 7.1, obrázek 7.3).



Obrázek 6.13: Ukázka výstupu neuronové sítě při učení nad datovou sadou TuSimple. Vlevo je maska predikovaná, uprostřed je maska opravdová, vpravo je vstupní originální obrázek.

Použití augmentací

Při použití augmentací klesla hodnota validačního skóre, podle očekávání. Avšak při použití Gaussova rozmazání a vytvoření dynamické masky experimenty ukazují přesnost výsledku porovnatelný s během pouze standardní masky velikosti 7 bez dalších modifikací. Běhy mají opět 70 epoch s počtem 3626 v datové sadě.



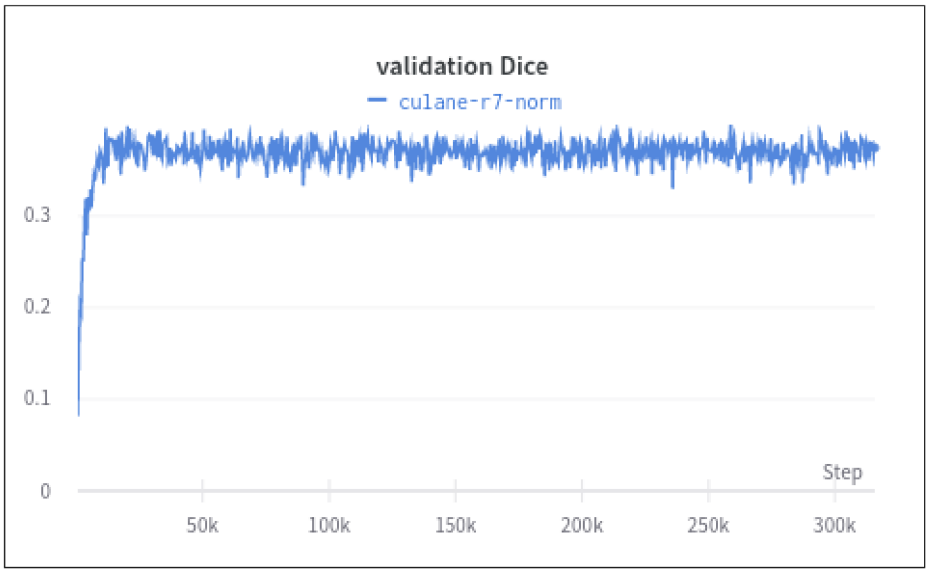
Obrázek 6.14: Nejnižší křivka, světle zelené barvy, je běh s velikostí pruhu 7 a aplikací augmentací. Validační skóre ukazuje na konci hodnotu 0.533, což je přibližně o desetinu nižší hodnota než při běhu bez augmentací.

Při dalším experimentování s augmentacemi a různými parametry jsem potvrdil, že Gaussovo rozmazání opět zvedne přesnost i augmentovaných obrázků (obrázek 6.15, avšak přidání dynamické masky přesnost téměř vůbec neovlivnilo.

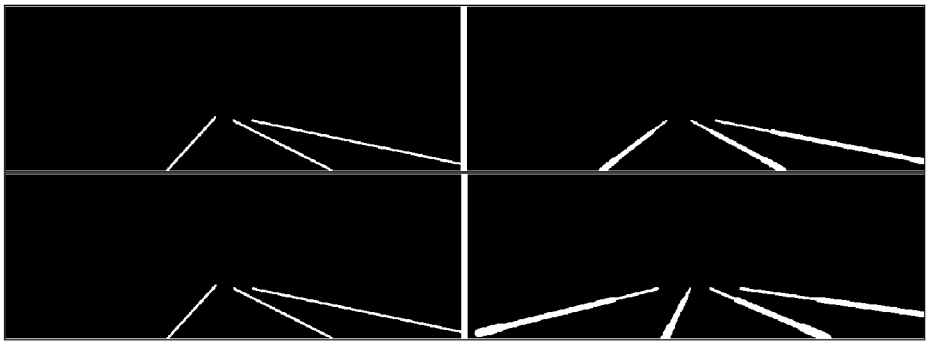
Obrázek 6.15: Běh s augmentacemi a Gaussově rozmazanou křivkou (vínová) má porovnatelnou přesnost s normálním během (světle modrá) a s poloviční velikosti pruhu v masce (bez rozmazání).

6.2.2 Culane a další porovnání

Datová sada CuLane byla vůbec nejtěžší pro neuronovou síť z důvodu typu obrázků, které v ní jsou. Standardní běh nad touto datovou sadou dosahuje maximálně 38 % přesnost. Toto je výrazně málo oproti ostatním (obrázek 6.19)

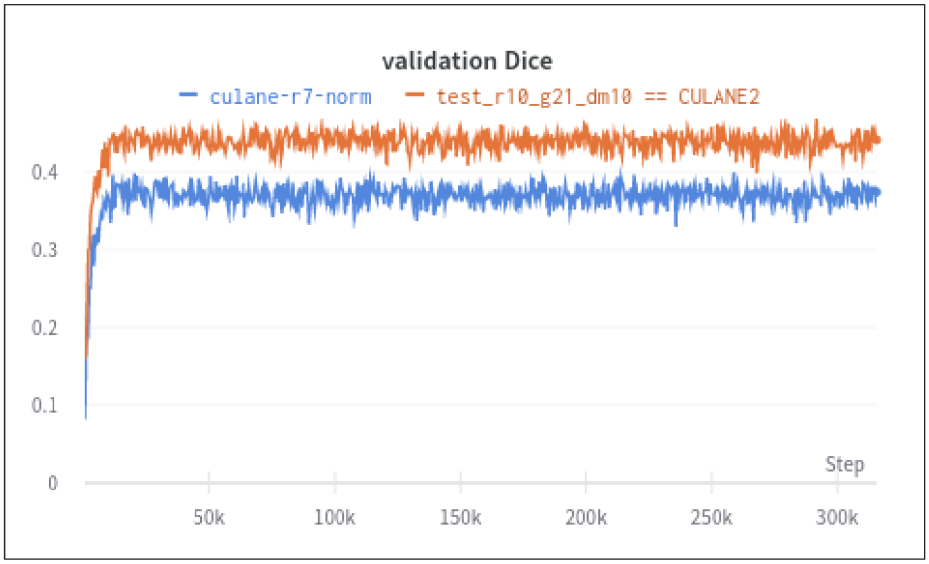


Obrázek 6.16: Ukázka grafu validačního skóre pro dataset CuLane. Má největší potenciál pro zlepšení přesnosti.



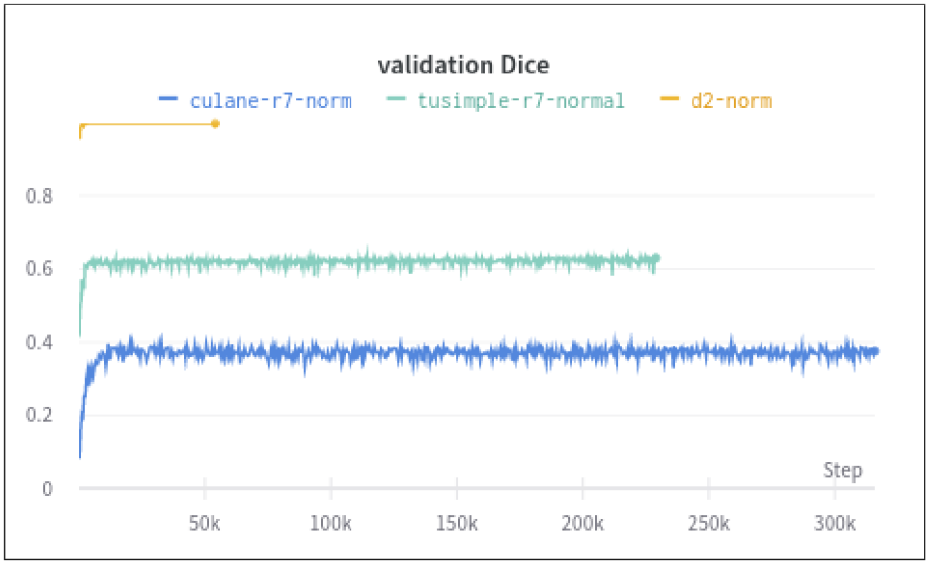
Obrázek 6.17: Porovnání normálních masek a dynamických masek ze CuLane datasetu

Při použití dynamické masky (6.18) i s Gaussovým rozmazáním se prokázalo, že přesnost lehce vzrostla, ikdyž méně než u ostatních datových sad.

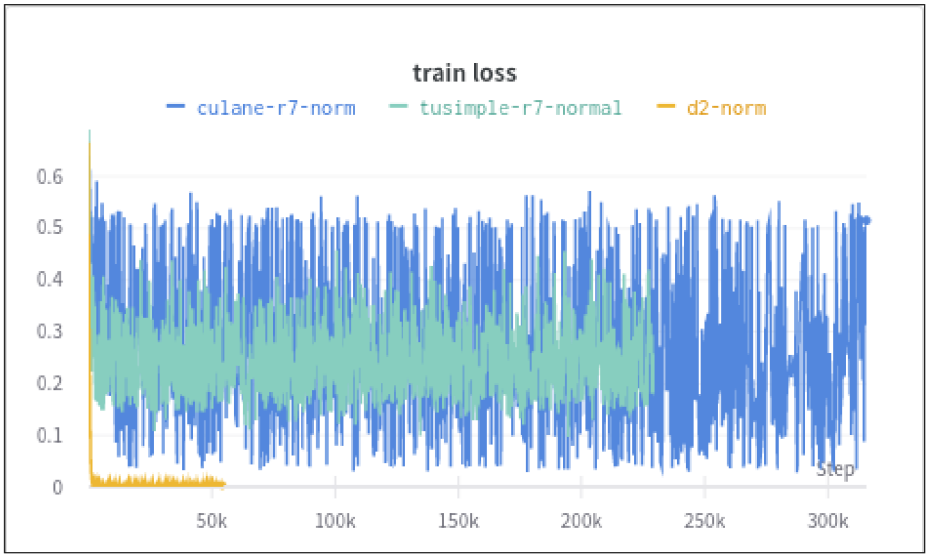


Obrázek 6.18: Ukázka zlepšení validačního skóre při použití Gaussova rozmazání s dynamickou maskou

Porovnání výsledků mezi datasety



Obrázek 6.19: Horní křivka (žlutá) je standardní běh nad syntetickým datasetem s druhým stupněm obtížnosti. Prostřední křivka (zelená) je normální běh nad TuSimple datasetem a spodní křivka (modrá) je nad CuLane datasetem.



Obrázek 6.20: CuLane (modrá) křivka má velkou amplitudu a je nestabilní. Toto je pravděpodobně způsobeno velkou různorodostí a komplexností samotné datové sady. TuSimple (zelená) křivka má mnohem nižší amplitudu a její trénovací chyba se drží mezi 0.2 a 0.3. Syntetický dataset (žlutá) je mnohem jednodušší a prakticky úplně minimalizoval trénovací chybu.

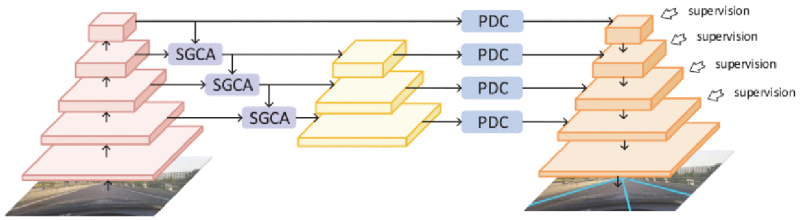
Kapitola 7

Možné rozšíření

Mezi možné rozšíření patří implementace SALMNet modelu a případně implementace více třídové klasifikace – rozlišovat i pruhy mezi sebou. Tato možnost generace mask je implementována v generátoru, avšak není využita ani ještě nebyla nijak testována.

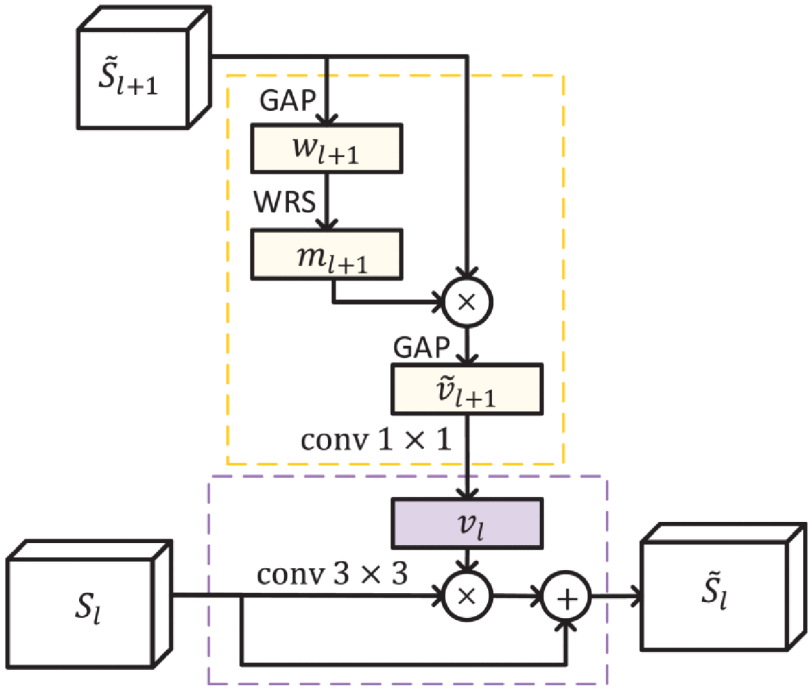
7.1 SALMNet model

Mezi hlavní možnost rozšíření patří implementace modelu SALMNet [12], který vychází z článku z roku 2020. Jednalo by se o rozšíření stávajícího UNet modelu, konkrétně o implementaci modulů SGCA (Semantic-guided channel attention) a PDC (pyramid deformable convolution). SGCA modul je vyvinut pro výběr nízkoúrovňových vlastností (features) konvoluční sítě za pomocí vysokoúrovňových vlastností.



Obrázek 7.1: Schématická ilustrace SALMNet modelu

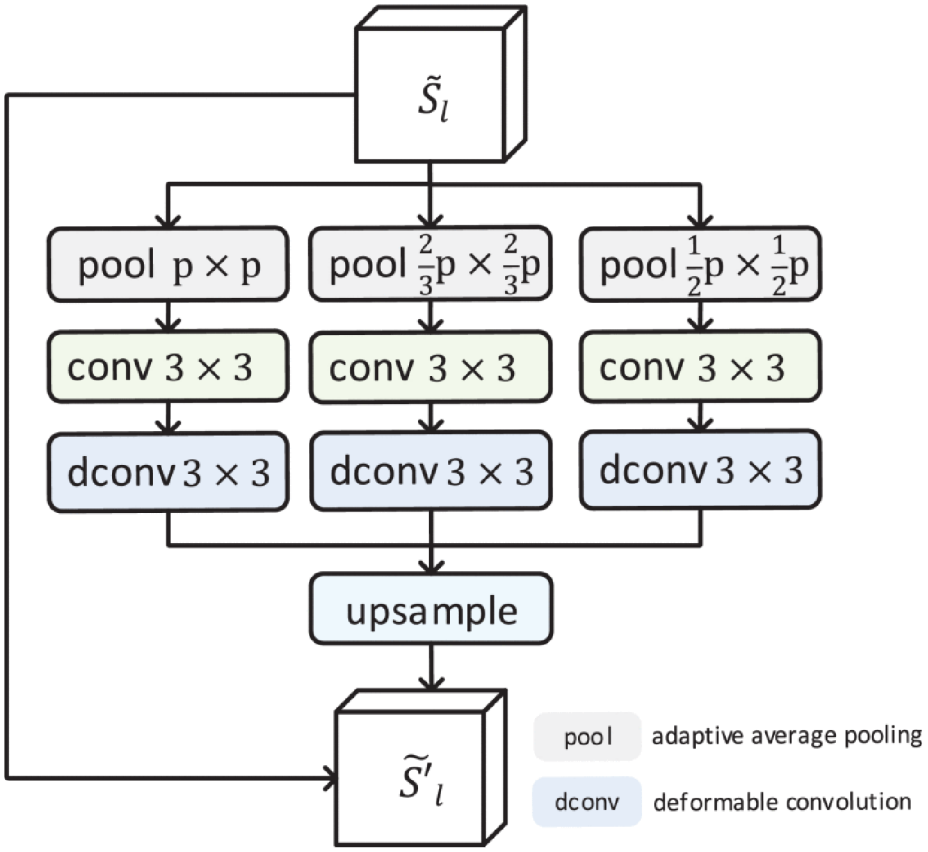
7.1.1 SGCA - semantic-guided channel attention



Obrázek 7.2: Detailní struktura SGCA modulu [12]

Díky vlastnostem a informacím získaných z vyšších vrstev, které jsou využity pro lepší výběr nízkoúrovňových detailních vlastností je zlepšena reprezentace prvků ve strukturách pro detekci jízdních pruhů a jsou potlačovány nežádoucí, rušivé prvky z pozadí. Samotný SGCA modul je vložen do tří prostředních stádií(obrázek 7.1). Aktivační mapy v nejvyšších a nejnižších vrstvách jsou ignorovány z důvodu nedostatku informací ve vyšších vrstvách a velkému využití paměti v nižších vrstvách.

7.1.2 PDC - Pyramid deformable convolution



Obrázek 7.3: Detailní struktura PDC modulu

Standardní konvoluční operace v SGCA modulu je lokální a používá čtvercové kernely s fixovanými poly, je tedy limitována možnost zachytit dlouhé a nespojené struktury jízdních pruhů. Kvůli tomuto je využita deformovatelná konvoluce, pro naučení offsetu počítaného bodu v tradičním kernelu konvoluce a následnou augmentaci lokality tohoto bodu. Deformovaná konvoluce tak pomáhá redukovat nerelevantní informace v tradiční konvoluci a zaměřuje se více na vlastnosti jízdních pruhů.

Kapitola 8

Závěr

Cílem této práce bylo prostudovat literaturu a seznámit se s problematikou týkající se počítačového vidění, strojového učení, konkrétněji na detekci jízdních pruhů, vytvořit dataset určený k trénování a experimentování neuronové sítě, porovnat a poukázat na rozdíly ve vyhodnoceních a zhodnotit dosažených výsledků.

První teoretická část je úvod do problematiky konvolučních neuronových sítí, použitých při detekci jízdních pruhů. Popisuje U–Net model, který byl součástí mé práce a následně použité implementační nástroje jako Python, framework Pytorch a knihovna Albumentations. Dále uvádím možné obdobné modely a jejich řešení pro danou problematiku detekce jízdních pruhů za použití modelů jako ConvoLSTM nebo Houghova transformačního prostoru.

Velká část této práce se zabývá syntetickým generátorem dat, který slouží pro generování obrázků pro trénování neuronových sítí. Generátor je možné parametrizovat několika způsoby pro docílení větší různorodosti, případně škálovatelnosti obtížnosti obrázků. Začal jsem s úplně základním obrázkem tří barev a bílých čar a postupně přidával složitost. Přidal jsem textury místo čistých barev, následně přidal zakřivení linií a Gaussovým zakřivením změnil rovné silnice. Toto je možné využít např. při práci s vlastní neuronovou sítí předtím, než se člověk pustí na reálná data. Pro další možnost obtížnosti a zvýšení datasetu je možné využít augmentace, které se aplikují při každé epoše trénování zvlášť pro maximální rozšíření. Výsledky trénování na tomto datasetu jsou převážně jednoznačné, a sice dosahují velmi dobrých výsledků, až 99 %.

Další krok byl přechod na reálná data. Využil jsem k tomu moderní datasetu TuSimple a CuLane. TuSimple obsahuje převážně dálniční situace a CuLane naopak velmi městskou část. Trénování nad reálnými daty bylo viditelně obtížnější než nad syntetickým datasetem, avšak detekec pruhů byla stále vcelku dobrá. Co se týče datasetu CuLane, bylo to horší, jelikož tyto situace byly velmi komplexní. Není jednoduché v obrázcích poznat, kde silniční pruhy jsou, a kde ne. Některé obrázky byly anotované i tam, kde silniční pruh vidět vůbec nebyl, např. když auto blokovalo pruh, což zhoršilo preciznost detekování a projevilo se na výsledcích tak, že třeba tato část pruhu nebyl detekována vůbec.

Jako možné menší rozšíření bych chtěl udělat více–třídovou klasifikační masku, protože to přesně navazuje na práci kterou jsem dělal do teď a je to částečně implementované v samotném generátoru. Jako velká možnost rozšíření je implementace nebo částečná implementace modelu SALMNet, resp. jeho části o deformované konvoluci a SGCA modulu. Každá tato část by měla zlepšit aktuální řešení a přesnost neuronové sítě. Deformovaná konvoluce na detekci dlouhých a nespojených silničních pruhů, tedy např. pro přerušované limie a nebo situace kdy je pruh v část zcela překryt. SGCA modul následně pro zaměření

hledání čar a šetření výkonu, zabývat se jen částmi obrázku, které jsou zajímavé a vyrušení nechtěných nebo falešných detekcí v pozadí.

Literatura

[1] Buslaev, A., Iglovikov, V. I., Khvedchenya, E., Parinov, A., Druzhinin, M. et al. Albumentations: Fast and Flexible Image Augmentations. *Information*. 2020, sv. 11, č. 2. DOI: 10.3390/info11020125. ISSN 2078-2489. Dostupné z: https://www.mdpi.com/2078-2489/11/2/125.

[2] Ding, B., Qian, H. a Zhou, J. Activation functions and their characteristics in deep neural networks. In: *2018 Chinese Control And Decision Conference (CCDC)*. 2018, s. 1836–1841. DOI: 10.1109/CCDC.2018.8407425.

[3] Goodfellow, I., Bengio, Y. a Courville, A. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[4] Luo, S., Zhang, X., Hu, J. a Xu, J. Multiple Lane Detection via Combining Complementary Structural Constraints. *IEEE Transactions on Intelligent Transportation Systems*. 2021, sv. 22, č. 12, s. 7597–7606. DOI: 10.1109/TITS.2020.3005396.

[5] Neven, D., Brabandere, B. D., Georgoulis, S., Proesmans, M. a Gool, L. V. Towards End-to-End Lane Detection: an Instance Segmentation Approach. In: *2018 IEEE Intelligent Vehicles Symposium (IV)*. 2018, s. 286–291. DOI: 10.1109/IVS.2018.8500547.

[6] O'Shea, K. a Nash, R. An Introduction to Convolutional Neural Networks. 2015. Dostupné z: arxiv.org/pdf/1511.08458.pdf.

[7] Raschka, S., Patterson, J. a Nolet, C. Machine Learning in Python: Main Developments and Technology Trends in Data Science, Machine Learning, and Artificial Intelligence. *Information*. 2020, sv. 11, č. 4. DOI: 10.3390/info11040193. ISSN 2078-2489. Dostupné z: https://www.mdpi.com/2078-2489/11/4/193.

[8] Ronneberger, O., Fischer, P. a Brox, T. U-Net: Convolutional Networks for Biomedical Image Segmentation. *CoRR*. 2015, abs/1505.04597. Dostupné z: http://arxiv.org/abs/1505.04597.

[9] Shi, X., Chen, Z., Wang, H. a Yeung, D.-Y. Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting. 2015. Dostupné z: https://arxiv.org/abs/1506.04214.

[10] Shorten, C. a Khoshgoftaar, T. M. A survey on Image Data Augmentation for Deep Learning. *Journal of Big Data*. Jul 2019, sv. 6, č. 1, s. 60. DOI: 10.1186/s40537-019-0197-0. ISSN 2196-1115. Dostupné z: https://doi.org/10.1186/s40537-019-0197-0.

[11] Xingang Pan, P. L. X. W. a Tang, X. Spatial As Deep: Spatial CNN for Traffic Scene Understanding. In: *AAAI Conference on Artificial Intelligence (AAAI)*. February 2018.

[12] Xu, X., Yu, T., Hu, X., Ng, W. W. Y. a Heng, P.-A. SALMNet: A Structure-Aware Lane Marking Detection Network. *IEEE Transactions on Intelligent Transportation Systems*. 2021, sv. 22, č. 8, s. 4986–4997. DOI: 10.1109/TITS.2020.2983077.

[13] Zou, Q., Jiang, H., Dai, Q., Yue, Y., Chen, L. et al. Robust Lane Detection From Continuous Driving Scenes Using Deep Neural Networks. *IEEE Transactions on Vehicular Technology*. 2020, sv. 69, č. 1, s. 41–54. DOI: 10.1109/TVT.2019.2949603.