VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY
DEPARTMENT OF CONTROL AND INSTRUMENTATION

DETEKCE SEMAFORU VE SNÍMKU DOPRAVNÍ SITUACE
TRAFFIC LIGHT DETECTION IN IMAGE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Václav Boček**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Karel Horák, Ph.D.**
SUPERVISOR

BRNO 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní obor **Automatizační a měřicí technika**
Ústav automatizace a měřicí techniky

Student: Václav Boček ***ID:*** 182589
Ročník: 3 ***Akademický rok:*** 2017/18

NÁZEV TÉMATU:

Detekce semaforu ve snímku dopravní situace

POKYNY PRO VYPRACOVÁNÍ:

Prostudujte možnosti detekce semaforu v obraze monokulární kamery z běžného silničního provozu za použití metod strojového učení. Body zadání:
1. Prostudujte metody detekce semaforu v obrazu a rozpoznání stavu jeho světel.
2. Vyberte vhodnou metodu založenou na konceptu strojového učení a implementujte ji pro zadanou množinu vstupníc h dat.
3. Ověřte funkčnost a stanovte úspěšnost klasifikace na reálných datech ze silničního provozu.
Téma zpracujte ve spolupráci s firmou Artin s.r.o. v rámci projektu Roboauto. Při implementaci použijte podle potřeb nástroje C++, OpenCV, Tensorflow, Keras, Caffe, ROS apod.

DOPORUČENÁ LITERATURA:

1. GONZALES, R.C., WOODS, R.R.: Digital image processing (3rd edition). Prentice-Hall, Inc., 2008. 954 pages. ISBN 978-0131687288.

2. SZELINSKI, R.: Computer Vision: Algorithms and Applications. Springer,2010. ISBN 978-1848829343.

3. Zapletal, O. Rozpoznávání obrazů konvolučními neuronovými sítěmi - základní koncepty. Brno 2017. Diplomová práce. VUT FEKT v Brně. Ústav automatizace a měřicí techniky. Vedoucí práce Karel HORÁK.

Termín zadání: 5.2.2018 ***Termín odevzdání:*** 21.5.2018

Vedoucí práce: Ing. Karel Horák, Ph.D.
Konzultant:

doc. Ing. Václav Jirsík, CSc.
předseda oborové rady

Abstrakt

Tato bakalářská práce se zabývá detekcí semaforu a rozpoznáním stavu jeho světel na snímku z dopravní situace pomocí metod strojového učení. V teoretické části textu jsou zmíněny rozdílné přístupy k řešení tohoto problému. V praktické části je popsán vlastní návrh systému s využitím konvolučních neuronových sítí a výsledky testování navrženého systému.

Klíčová slova

Počítačové vidění, strojové učení, konvoluční neuronová síť, detekce semaforu, stav semaforu, dataset

Abstract

This bachelor thesis deals with the traffic lights detection and recognition of the displayed colour using methods of machine learning. In the theoretical section, different methods to solve this problem are described. The practical part describes the proposed system which was realized using convolutional neural networks. Furthermore, results of performed tests are shown.

Keywords

Computer vision, machine learning, convolutional neural network, traffic light detection, state of traffic light, dataset

Bibliografická citace

BOČEK, V. *Detekce semaforu ve snímku dopravní situace.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2018. 56 s. Vedoucí bakalářské práce Ing. Karel Horák, Ph.D..

Prohlášení

Prohlašuji, že svou bakalářskou práci na téma Detekce semaforu ve snímku dopravní situace jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce. Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb

V Brně dne: **20. května 2018** ……

podpis autora

Poděkování

Děkuji vedoucímu bakalářské práce Ing. Karlu Horákovi, Ph.D. za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé bakalářské práce.

V Brně dne: **20. května 2018**

……
podpis autora

OBSAH

1. ÚVOD

Detekce semaforů v provozu je důležitá pro bezpečí lidí ve vozidle i pro další účastníky silničního provozu. Systém detekce semaforu může pomoci řidiči včas a za ztížených podmínek zaznamenat tyto důležité silniční signály nebo se dokonce může v autonomních automobilech podílet na aktivním řízení vozidla.

Cílem této bakalářské práce bylo vytvořit program, který pomocí některé z metod strojového učení bude detekovat semafory a také určí jeho barvu. Dále jsem měl ověřit funkčnost a stanovit úspěšnost klasifikace na reálných datech ze silničního provozu.

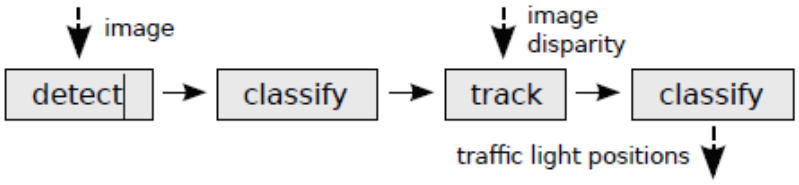
Druhá kapitola se zabývá rešerší prací a přístupů, které se věnují tématu detekce semaforu, aby se mi povedlo dále vybrat vhodnou metodu k řešení tohoto problému. Ve třetí kapitole jsou popsány veřejně dostupné datasety a také je tu popsán vlastní Boček dataset, který byl vytvořen jako součást této bakalářské práce. Ve čtvrté kapitole je uvedeno základní vysvětlení pojmů z oblasti konvolučních neuronových sítí, protože konvoluční sítě následně používám v praktické části. Pátá kapitola obsahuje vlastní návrh systému detekce a klasifikace stavu semaforu a výsledky testů na reálných datech z provozu.

2. VYBRANÉ METODY DETEKCE SEMAFORŮ

Tato kapitola se věnuje průzkumu přístupů a metod využívající strojové učení, kterými byl v posledních letech řešen problém detekce semaforu a určení jeho signalizované barvy v obraze. Jsou zde také popsány výsledky, kterých bylo daným přístupem dosaženo. Názvy podkapitol jsou odvozeny od názvů článků, ze kterých bylo čerpáno.

2.1 A Deep Learning Approach to Traffic Lights: Detection, Tracking, and Classification

Následující přístup řešení problému detekce semaforu a určení barvy světla, které na něm svítí, aplikovali výzkumníci Karsten Behrendt, Libor Novák a Rami Botros. Výsledky své práce prezentovali v [1] článku IEEE v květnu 2017.
Tento systém detekce semaforu je rozdělen na čtyři fáze. V první fázi detektor nalezne ve snímku semafor a oblast semaforu označí ohraničujícím rámečkem, neurčuje ale stav semaforu. V druhé fázi klasifikátor odstraní falešné detekce semaforu (pokud semafor byl v předchozím kroku detekován na místě, kde ve skutečnosti není) a také prvotně určí, jaké světlo svítí na detekované semaforu. Ve třetí fázi vyhodnocovacího procesu je oblast semaforu sledována v časové řadě více snímků pro zajištění větší stability a přesnosti detekce. Tím je nakonec ověřen výskyt semaforu a potvrzena, popřípadě opravena barva na semaforu, tedy jeho stav. Posloupnost kroků je zobrazena na obr. 2-1.



Obr. 2-1: Posloupnost kroků systému [1]

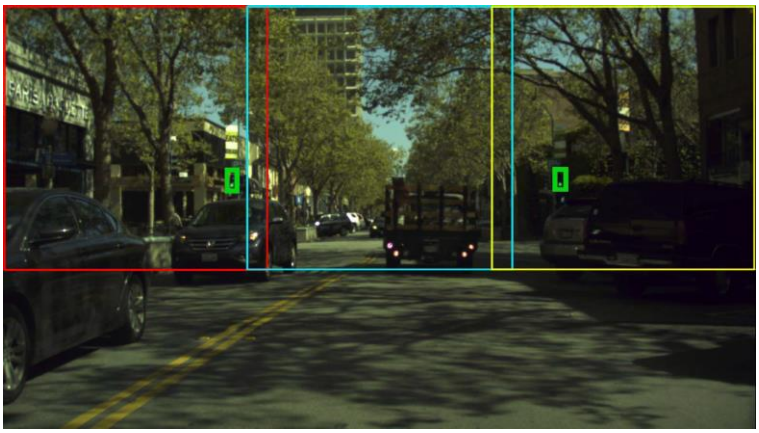
2.1.1 Galerie dat

Tento aplikovaný přístup strojového učení potřebuje k naučení trénovací data. Autoři projektu používají vlastní galerii dat Bosch Small Traffic Lights Dataset.
Galerie obsahuje tzv. R-I-I-B snímky pro trénování a RGB snímky, které jsou pouze pro ladění. Snímky mají rozlišení 1280 x 720 pixelů. K trénování bylo vybráno více než 5000 snímků ze silničního provozu v okolí San Francisca.

Obr. 2-2: Typické snímky z použité galerie [2]

2.1.2 Detekce semaforu

K detekování semaforu je použit systém s architekturou You Look Only Once (YOLO). Místo toho, aby na vstupy vstupních neuronů sítě byly přivedeny jasové hodnoty jednotlivých pixelů z celého snímku, tak se systém zabývá pouze vrchními třemi pětinami pixelů snímku. To proto, že semafor se nejčastěji vyskytuje právě v horní části snímků. Tato oblast je pak dále rozdělena na tři samostatné obrazové podoblasti. Z původního rozlišení snímku 1280 x 720 tedy jsou tři podoblasti o 448 x 448 pixelech. To je znázorněno v obr.2-3. Každá ze tří podoblastí je rozdělena na 11 x 11 buněk. Jasové hodnoty pixelů z každé oblasti jsou přivedeny na vstup jedné z paralelních neuronových sítí, která v dané oblasti detekuje semafory.



Obr. 2-3: Zobrazení rozdělení snímku, kde každou z částí zpracovává jedna YOLO síť[1]

V tomto případě neuronová síť nemá sloužit ke klasifikování objektů, pouze k nalezení objektu jedné třídy – semaforu s různými barvami světel.
Tato neuronová síť vrací z každého snímku 3 x 11 x 11 = 363 ohraničujících rámečků. Ke každému ohraničujícímu rámečku je také přidán odhad sítě o přesnosti jeho umístění.

2.1.3 Klasifikace stavu semaforu

Klasifikační část slouží k tomu, aby se určil stav semaforu – stav semaforu je dán světlem/světly, které na něm svítí, popř. bliká. Tento problém je řešen malou konvoluční neuronovou sítí, která rozlišuje mezi jednotlivými stavy a zároveň odstraňuje falešnou pozitivní detekci. Falešná detekce je výsledek algoritmu, který prohlašuje skupinu pixelů za oblast semaforu, ačkoliv v reálné scéně dané pixely představují jiný objekt anebo pozadí. Jedná se tedy o tzv. misklasifikaci neboli mylnou inferenci klasifikátoru. To nastává například, když je jako semafor detekováno brzdové světlo automobilu.
Ohraničující rámečky jsou normalizovány na jednotnou velikost a mají upravené měřítko. Ohraničující rámeček má poté rozměry 64 x 64 pixelů, z toho semafor je široký 20 pixelů, po každé straně semaforu – horizontálně - tedy zůstává 22 pixelů pozadí. Toto pozadí je důležité, zasazuje semafor do kontextu, který je pro klasifikaci důležitý. Klasifikátor rozlišuje stavy: "pozadí", "zelená", "žlutá", "červená" a "off". Stav "off" je nezbytný při kontrole správnosti klasifikace bez procházení přes více snímků, protože na některých snímcích se semafor jeví jako by byl vypnutý. To je způsobeno rozdílnými časy snímání kamery a obnovovací dobou LED semaforů [1].

Konvoluční neuronová síť klasifikátoru obsahuje 6 váhových vrstev, 3 z nich jsou konvoluční a 3 plně propojené. Její struktura je uvedena v Tab.2-1.

Tab. 2-1: Struktura sítě klasifikátoru, převzato z [1]

Input	$3 \times 64 \times 64$
Convolution	filters: 32, kernel: (7,7), padding: 0, ReLU
Max-pooling	kernel: (2,2)
Convolution	filters: 64, kernel: (3,3), padding: 0, ReLU
Max-pooling	kernel: (2,2)
Convolution	filters: 128, kernel: (3,3), padding: 0, ReLU
Fully connected	units: 256, ReLU
Dropout	p: 0.5
Fully connected	units: 128, ReLU
Dropout	p: 0.5
Fully connected	units: 5, softmax

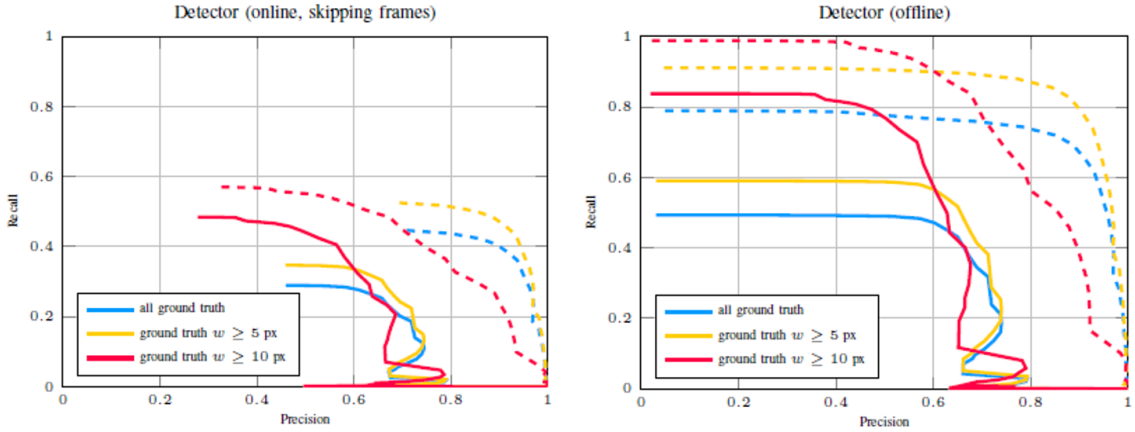
Trekování oblastí semaforu

Účelem trekování místa semaforu na snímku je korigovat občasné výpadky detektoru. V tomto řešení se využívá pohybového modelu založeného na principu odometrie a triangulace k odhadnutí pohybu semaforu a neuronové sítě ke zlepšení trekovaných pozic.

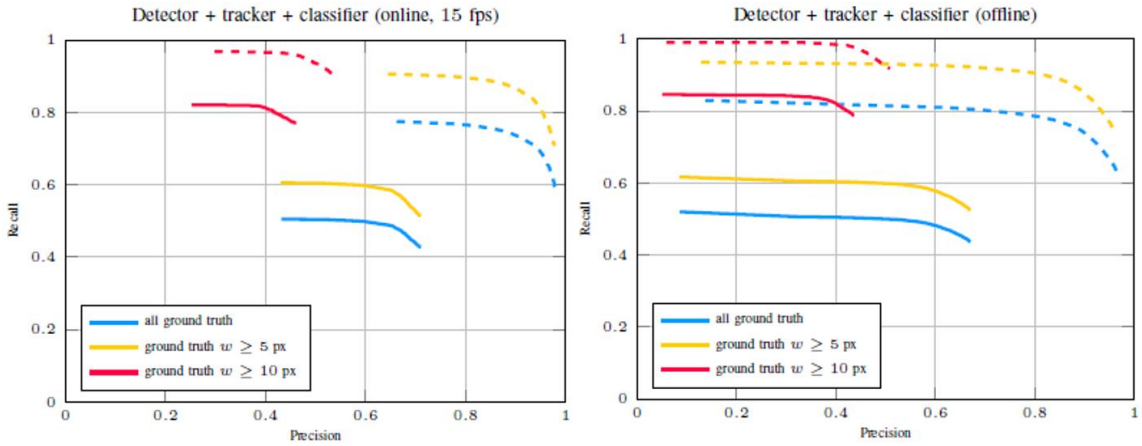
2.1.4 Vyhodnocení a výsledky

Vyhodnocování míry přesnosti bylo provedeno na testovacím souboru snímků z Bosch Small Traffic Light Dataset.

Detektor
Detektor vyhodnocuje přítomnost semaforů na snímku s frekvencí 10 snímků/s. To je nižší frekvence než je frekvence snímání kamery. Proto jsou k ověření funkčnosti a přesnosti systému dva testy – online a offline. (při offline testu detektor vyhodnotí každý snímek)



Obr. 2-4a,4b: Výsledky detektoru při online testu a offline testu [1]



Obr. 2-5a,5b: Výsledky celého systému při online testu a offline testu [1]

Pro vyhodnocení jednoho snímků je třeba doby 100 ms, proto při online režimu některé snímky z kameru musí být detektorem vynechány a jsou pak poslány pouze do trekovací části. To vede k nižší přesnosti. Výsledky obou testů jsou zobrazeny na obr.2-4 a obr.2-5.

Plná křivka grafu značí průběhy při Intersection Over Union (IOU) > 0,5
Přerušovaná křivka grafu značí průběhy při IOU > 0,3

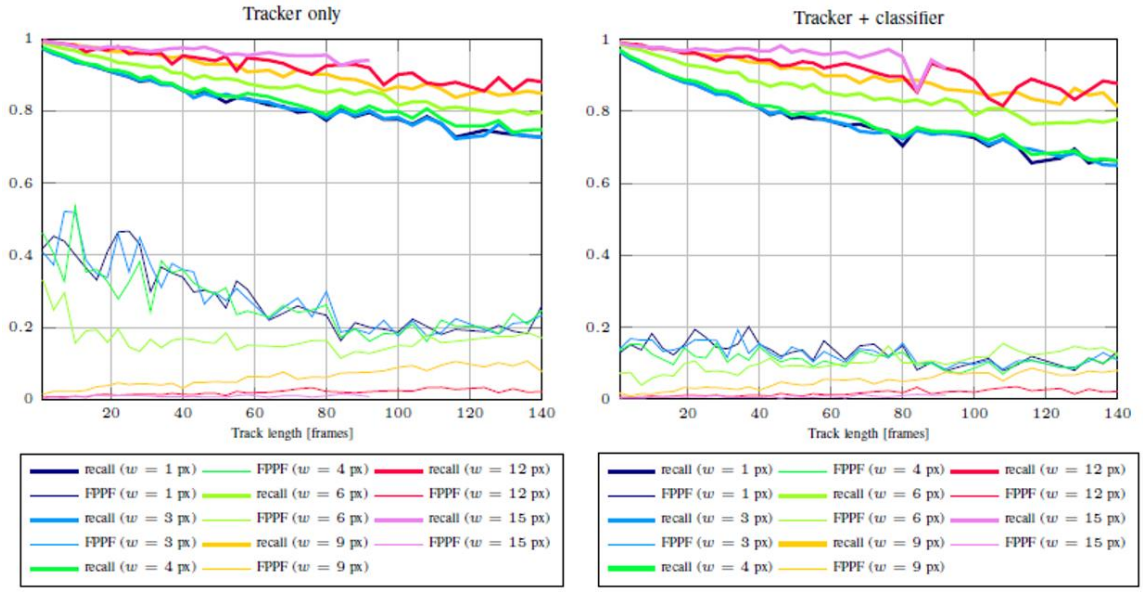
$$IOU = \frac{BBdet \,\&\&\, BBgt}{BBdet \,''\, BBgt} \quad (1)$$

kde BBdet je detekovaný ohraničující rámec a BBgt je správný (anotovaný) ohraničující rámec

Recall na y ose je poměr správně detekovaných semaforů vůči všem semaforům.
Precision na x ose je poměr správně detekovaných semaforů vůči všem detekcím systému.

Klasifikátor
Klasifikátor dosahoval na snímcích z testovacího setu přesnosti 95,1 %. Největší chybovost byla v klasifikaci stavu „off“, který byl často určen jako “pozadí“.

Tracker
Tracker byl testován samostatně bez detektoru na snímcích, kde byly ručně vyznačeny hraniční oblasti semaforu. Je testováno po kolik snímků dokáže tracker sledovat daný ohraničující rámeček. Symbol w v legendě grafů značí minimální šířku ohraničujícího rámečku v pixelech.



Obr. 2-6a,6b: Výsledné hodnoty trackeru [1]

2.2 DeepTLR: A single Deep Convolutional Network for Detection and Classification of Traffic Lights

Tento přístup využili k detekci semaforu a určení barvy, kterou signalizuje, němečtí výzkumníci Michael Weber, Peter Wolf a Marius Zölner. Výsledky své práce publikovali v červnu 2016 v článku [3].

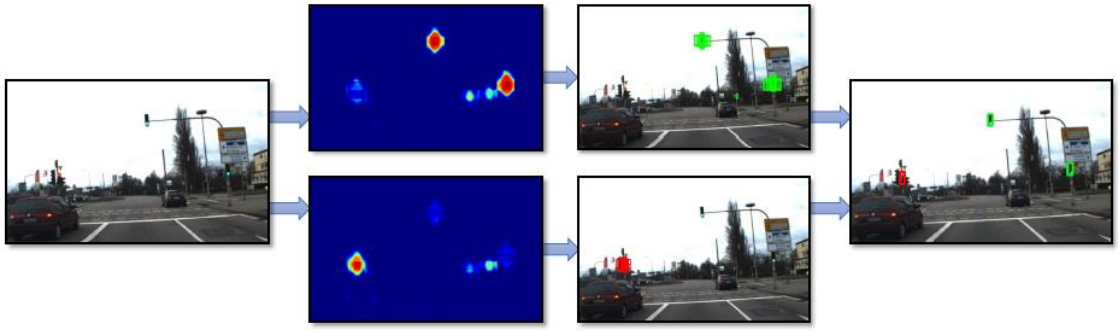
2.2.1 Popis systému

Systém pro detekci a klasifikaci stavu semaforu DeepTLR je realizován hlubokou konvoluční neuronovou sítí (ConvNet). Architektura této sítě je uvedena v Tab.2-2. V tomto přístupu se nevyužívá žádných apriorních znalostí prostředí, ve kterém se automobil pohybuje, jako jsou například předem zjištěné GPS souřadnice jednotlivých semaforů. To dělá tento systém více robustní a flexibilní vůči prostředí, ve kterém je schopen správně pracovat.

Tab. 2-2: Architektura hluboké konvoluční sítě systému DeepTLR [3]

Layer	conv1	pool1	conv2	pool2	conv3	conv4	conv5	pool5	conv6	conv7	conv-prob	conv-bb
# channels	96	96	256	256	384	384	384	384	4096	4096	192	256
Filter size	11×11	3×3	5×5	3×3	3×3	3×3	3×3	3×3	6×6	1×1	1×1	1×1
Stride	4	2	1	2	1	1	1	2	1	1	1	1
Padding	0	0	2	0	1	1	1	0	3	0	0	0

Systém rozpoznávání semaforu je dvoufázový, v první fázi jsou vytvořeny pravděpodobnostní mapy výskytu semaforu pro každý z jeho stavů, v druhé fázi pak je vymezena oblast semaforu a je označena hraničním rámečkem. Proces je názorně vyobrazen na obr.2-7.



Vstupní snímek Pravděpodobností mapy Vymezení semaforu rámečkem Finální detekce

Obr. 2-7:Náhled na jednotlivé kroky systému [3]

Na vstupní vrstvu sítě ConvNet jsou přivedeny jasové hodnoty pixelů z celého vstupního obrázku. Tok informací v této konvoluční síti postupuje přirozeně od nižší vrstvy k vyšší, čili vyšší vrstva má na vstupu matice výstupních hodnoty z nižší vrstvy a tím dochází s postupem informace do vyšších vrstev ke stálému zesložiťování příznaků. Výstupem této konvoluční sítě je vygenerovaná pravděpodobnostní mapa pro každý z možných stavů semaforu a set hraničních rámečků.
Ty hraniční rámečky, které nemají dostatečnou pravděpodobnost v pravděpodobnostní mapě jsou pak odfiltrovány. K dalšímu omezení počtu hraničních rámečků se použije algoritmus groupRectangles z openCV, tím se z pravděpodobných hraničních rámečků vytvoří/vybere pouze jeden pro jeden skutečný semafor [3].

2.2.2 Dataset

Snímky v datasetu jsou pořízeny převážně ve městě a za různého počasí. Snímky mají rozlišení 1280 x 960 pixelů. Některé snímky byly vybrány a ručně na nich byl anotován hraniční rámeček semaforu, čili jedná se o semi – supervised učení. Parametry tohoto základního datasetu jsou uvedeny v tab.2-3.

Tab. 2-3: Trénovací a testovací dataset [3]

Name	Images	Annotations	Green	Red	Amber
Training Data	10.533	13.253	8.172	5.081	219
Evaluation Data	343	625	427	198	29

Toto základní množství snímků by nestačilo na kvalitní natrénování sítě, proto byl tento dataset rozšířen vertikálním ozrcadlením a dalšími úpravami, aby bylo možné učit síť na větším množství dat. Parametry výsledného datasetu po úpravách je uveden v tab.2-4.

Tab. 2-4: Rozšířený trénovací a testovací dataset [3]

Name	Images	Annotations	Green	Red
ConvNet Train	104.360	106.834	68.500	38.334
ConvNet Val	791	1.114	421	228

2.2.3 Výsledky testování systému

Jako správně detekovaný semafor je uznán ten, který má IOU větší než 0,25.

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \qquad (2)$$

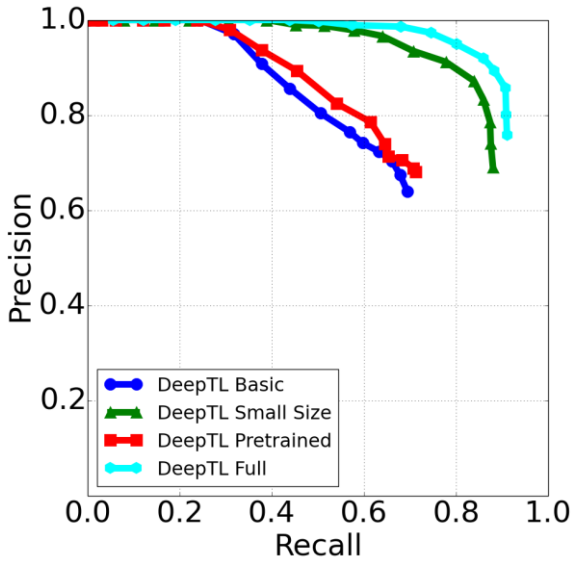
Síť byla naučena čtyřmi způsoby:

Base – síť byla učena pouze ze zmíněného datasetu, váhy konvolučních jader byly inicializovány náhodnými hodnotami; z anotace datasetu byly odstraněny ohraničující rámečky menší než 16 pixelů

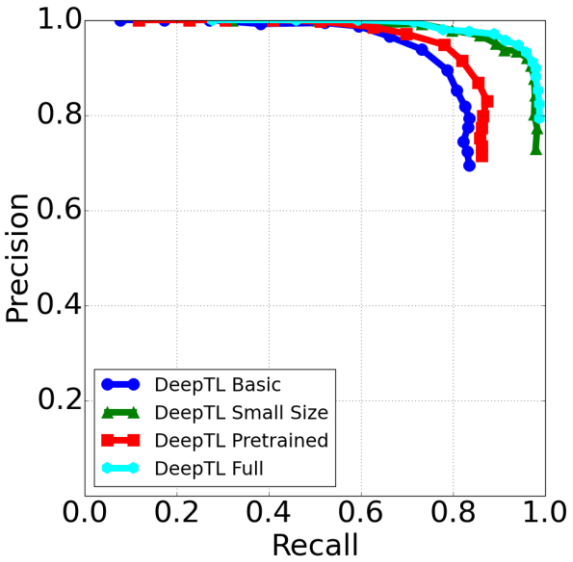
Pretrained – síť byla předtrénována váhami z Caffe BVLC AlexNet, která byla trénována na datasetu ImageNet data

Small Samples – z anotace datasetu byly odstraněny ohraničující rámečky menší než 8 pixelů

Full – kombinace Pretrained a Small Samples



Obr. 2-8: Výsledný recall / precision graf pro snímky o rozlišení 640 x 480 px [3]



Obr. 2-9: Výsledný recall / precision graf pro snímky o rozlišení 1280 x 920 px [3]

V tabulce Tab.2-5 jsou zobrazeny výsledky všech čtyř modelů pro snímky o rozlišení 640 x 480 a 1280 x 960 pixelů.

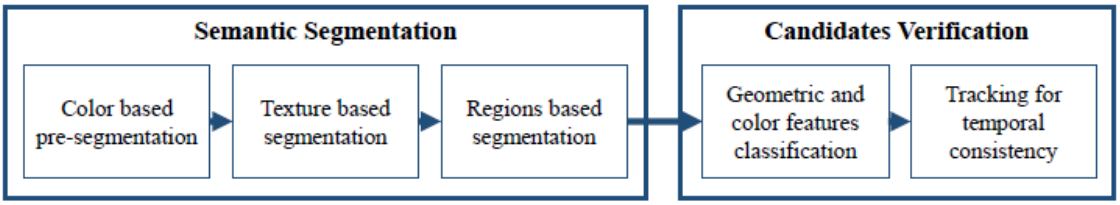
Tab. 2-5: Porovnání výsledků různě učených sítí [3]

Model	Resolution	TP	FP	FN	FC	Precision	Recall	F1
DeepTLR Base	640 x 480	413	42	80	132	70,4%	66,1%	68,2%
DeepTLR Pretrained		427	38	58	140	70,6%	68,3%	69,4%
DeepTLR Small Samples		546	99	29	50	78,6%	87,4%	82,7%
DeepTLR Full		567	59	22	36	85,6%	90,7%	88,1%
DeepTLR Base	1280 x 960	457	15	153	15	93,8%	73,1%	82,2%
DeepTLR Pretrained		488	21	131	6	94,8%	78,1%	85,6%
DeepTLR Small Samples		569	38	56	0	93,7%	91,0%	92,4%
DeepTLR Full		571	25	53	1	95,6%	91,4%	93,5%

Nejlepších výsledků tedy bylo dosaženo na přednaučené síti a s přidáním semaforů o malých rozměrech, kde F1-score dosáhlo hodnoty 93,5 %.
Tento sytém byl testován na počítači s grafickou kartou Nvidia GeForce GTX 980, při rozlišení 640 x 480 px bylo dosaženo frekvence 33 snímků/s, při rozlišení 1280 x 920 pak byla frekvence 13 snímků/s [3]. Tento systém má tedy dostatečně krátkou vyhodnocovací dobu k tomu, aby byl použit jako real-time systém.

2.3 Semantic segmentation based traffic light detection at day and at night

Tento přístup k detekci semaforu a určení jeho stavu aplikovali výzkumníci Vladimir Haltakov, Jakob Mayr, Christian Unger a Slobodan Ilic a výsledky této práce prezentovali v článku [4] nakladatelství Springer v roce 2015.
Jejich systém obsahuje dvě hlavní části. První je segmentace důležitých kandidátních oblastí, u kterých je vysoká pravděpodobnost, že obsahují semafor. Tato část pracuje se snímky na úrovni jednotlivých pixelů. Tou druhou částí je následná verifikace těchto nalezených oblastí, která vychází z již předurčených kandidátních oblastí a operuje tedy na této vyšší úrovni. Schéma systému je uvedeno na obr.2-10.



Obr. 2-10: Schéma posloupnosti kroků systému [4]

2.3.1 Semantická segmentace kandidátních oblastí

U segmentace kandidátů je důležité mít co nejméně “false negative“, protože pokud se reálný semafor odfiltruje jako pozadí, tak zbývající části systému už ho nemají šanci detekovat. Naopak větší počet “false positive“ sice více snižuje vyhodnocovací rychlost systému, protože je třeba později verifikovat tyto oblasti, které by tedy měl systém vyřadit, ale nemá vliv na finální kvalitu detekce, tedy na množství chyb systému.
Cílem této segmentace je najít potenciální regiony, které by mohly být semafory.

Obr. 2-11: Původní snímek – výstup z kamery [4]

Segmentace založená na barvách

Do tohoto kroku vstupují jasové hodnoty pixelů originálních snímků. Pixely obrázku jsou prahovány hodnotou, která je určena tak, aby nebylo odfiltrováno příliš mnoho kandidátních oblastí. Výstupem je naprahovaný obrázek, příklad takového výstupu je v obr.2-12.

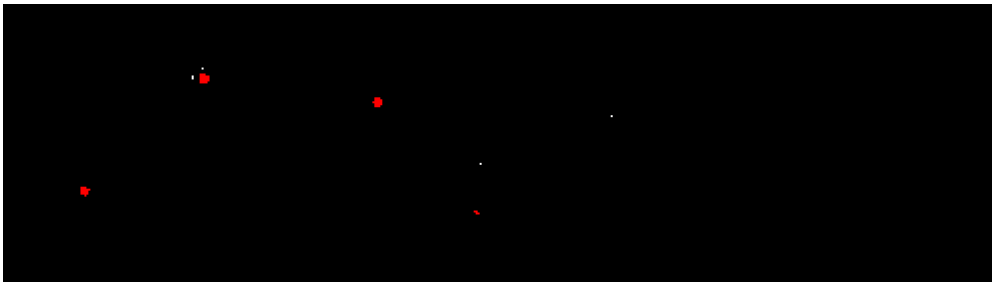
Obr. 2-12: Obrázek po segmentaci barev [4]

Segmentace založená na textuře

Vstupem je naprahovaný obrázek. Jednotlivé pixely obrazu jsou v tomto kroku klasifikovány jako pozadí nebo jako kandidátní podle textury v jejich okolí. K tomu se využívá vektoru příznaků, který je spočítán pomocí 2D Wasch-Hadamard transformace. Výstupem je snímek s kandidátními oblastmi, příklad je zobrazen na obr.2-13.

Obr. 2-13: Obraz po segmentaci textury [4]

Segmentace regionů
Vstupem je binární matice obrazu po segmentování textur, kde jsou kandidátní oblasti a okolí. V tomto kroku se dále redukuje množství kandidátních oblastí. Zda patří daný pixel ke kandidátní oblasti nebo k pozadí se určuje podle jeho osmiokolí. Použitý agoritmus bude podobný rotující masce. Výstupem jsou zpřesněné kandidátní oblasti, příklad zobrazen na obr.2-14.



Obr. 2-14: Obraz po segmentaci regionů a po verifikaci těchto kandidátních regionů (červeně) [4]

2.3.2 Verifikace kandidátních oblastí

Klasifikace podle geometrie a barevných příznaků
Na rozdíl od minulých kroků, kde se rozlišovaly pouze kandidátní oblasti od pozadí, zde jsou čtyři možné třídy, do kterých se kandidátní oblasti klasifikují. Těmi jsou stav pozadí, zelená, žlutá a červená. Klasifikace se provádí metodou JointBoostova klasifikátoru. Vektor geometrických a barevných příznaků, který vstupuje do klasifikátoru, je uveden v tab.2-6. Obrázek obr.2-14 pak zobrazuje příklad výsledku této klasifikace, kde barevné body jsou klasifikovaná světla semaforu odpovídající barvy.

Tab. 2-6: Příznaky vektoru JointBoost klasifikátoru [4]

Feature	Values	Description
Mean (RGB) Mean (Lab)	3 3	Mean of the region pixels computed separately for each color channel.
Std. deviation (RGB) Std. deviation (Lab)	3 3	Standard deviation of the region pixels computed separately for each color channel.
Image position	2	The pixel coordinates of the center of the region.
Area	1	Area of the region.
Orientation	1	Angle between the x-axis and the major axis of the region.
Aspect ratio	1	Aspect ratio of the two sides of the region's bounding box.
Ratio of areas	1	Ratio of the areas of the region and its bounding box.
Y-coordinate-area ratio	1	Ratio between the region's center y-coordinate and its area.
Solidity	1	Ratio between the areas of the region and its convex hull.
Eccentricity	1	Ratio of the distance between the foci and the major axis length of an ellipse that has the same second moment as the region.

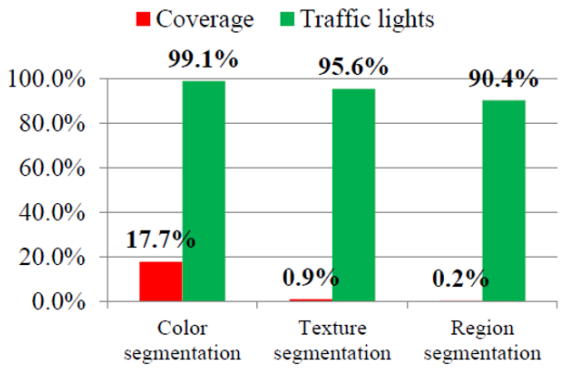
Trekování
Trekováním je myšleno procházení a kontrola přítomnosti detekovaného semaforu v řadě několika po sobě následujících snímků a až po detekci daného semaforu v dostatečném množství snímků je finálně detekován. Stává se totiž, že detekovaný semafor náhle po krátkou dobu nesvítí (hlavně LED semafory) nebo naopak vlivem šumu a objektům podobným semaforům je krátkodobě nesprávně detekován. Tento krok tedy dává systému jistou setrvačnost, která potlačuje krátké změny a tím je celý systém více robustní a stabilní.

2.3.3 Dataset

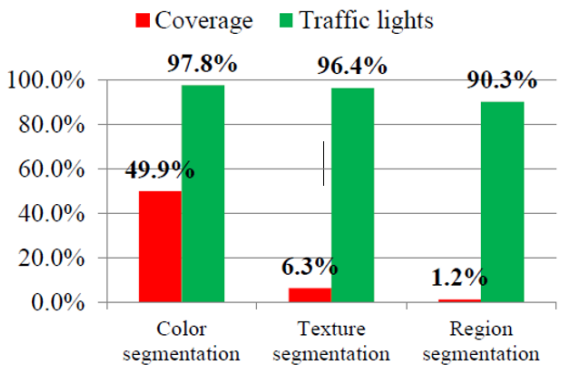
Jsou využity dva datasety. První je získaný v Pařížských ulicích, tato sekvence snímků obsahuje záznam dopravního provozu o délce 17 minut, kde jsou ručně anotovány hraniční rámečky. Druhý pak je z Německa za slunečného počasí, snímky jsou pořízeny 1 megapixelovou kamerou. Je zde zaznamenáno 17 km trasy v městském prostředí, která obsahuje 57 křižovatek se semafory [4].

2.3.4 Výsledky

Na obr.2-15 a obr.2-16 jsou zobrazeny výsledky semantické segmentace kandidátních oblastí. Termín "Coverage" reprezentuje poměrné množství pixelů v kandidátních oblastech v jednotlivých fázích segmentace vůči všem pixelům obrazu. Termín "Traffic lights" v grafech reprezentuje poměr množství pixelů, které jsou označeny jako kandidátní oblasti a všech pixelů, které by měly správně být obsaženy v kandidátních oblastech.



Obr. 2-16: Semantická segmentace – dataset pořízený ve Francie [4]



Obr. 2-15: Semantická segmentace – dataset pořízený v Německu [4]

Celkové kvantitativní výsledky systému jsou zobrazeny v tab.2-7

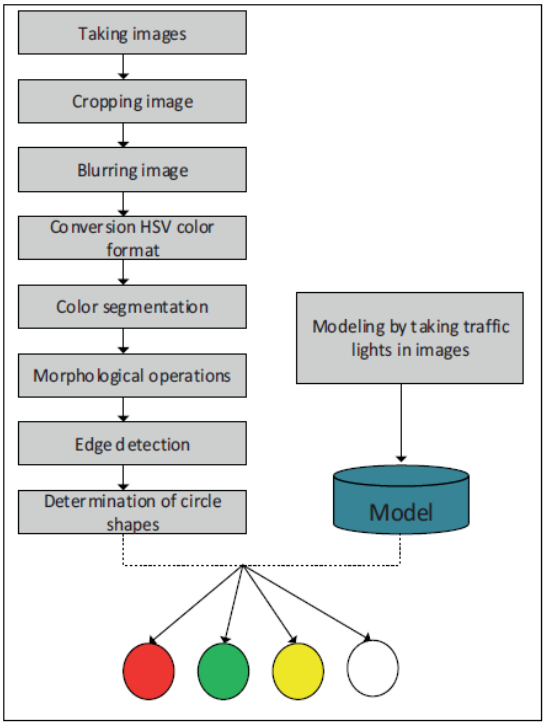
Tab. 2-7: Výsledky systému [4]

Stage	France Day		Germany Day		Germany Night	
	Recall	Precision	Recall	Precision	Recall	Precision
Without tracking	76.1%	63.3%	91.6%	61.3%	91.5%	57.4%
With tracking	71.7%	73.2%	84.3%	71.5%	84.4%	73.8%

Systém byl testován na počítači s dvěma procesory Intel Xeon X5690 s taktem 3,5 GHz. Vyhodnocení jednoho snímků pak trvalo 65 ms, čili frekvence byla přibližně 15 Hz.

2.4 A Vision Based Traffic Light Detection and Recognition Approach for Intelligent Vehicles

Tento přístup k řešení problému detekce semaforu a klasifikace jeho barvy aplikovali Ziya Ozcelik, Canan Tastimur, Mehmet Karakose a Erhan Akin z Firat Univerzity v Elazig v Turecku. Článek o tom vyšel v roce 2017 v IEEE [5]
Tento systém detekce semaforu je založen na hledání kruhového objektu v obraze, který by mohl představovat světlo semaforu. K tomu se používají klasické metody zpracování obrazu. Klasifikace je pak provedena pomocí SVM.
Diagram celého systému je zobrazen na snímku níže.



Obr. 2-17: Diagram celého systému [5]

Ze všeho nejdříve byl snímek semaforu filtrován průměrovacím filtrem, Gaussovým filtrem a mediánem, kde jádra měla velikost 9x9 pixelů.

Průměrovací filtr má matematický předpis:

$$K = \frac{1}{K_{width} x K_{height}} \begin{bmatrix} 1 & 1 & \dots & 1 \\ 1 & 1 & \dots & 1 \\ 1 & 1 & \dots & 1 \end{bmatrix} \quad (3)$$

Gaussův filtr má předpis:

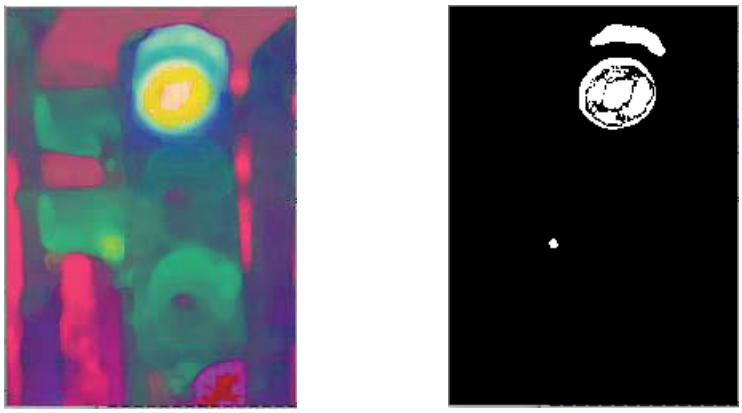
$$g(x) = ae^{-\frac{(x-b)^2}{2c^2}} \quad (4)$$

Příklady výsledků filtrací jsou na obrázcích níže.

(a) (b) (c)

Obr. 2-18a,18b,18c: Výsledky operací průměrovacího filtru (a), Gaussova filtru (b) a mediánu (c), [5]

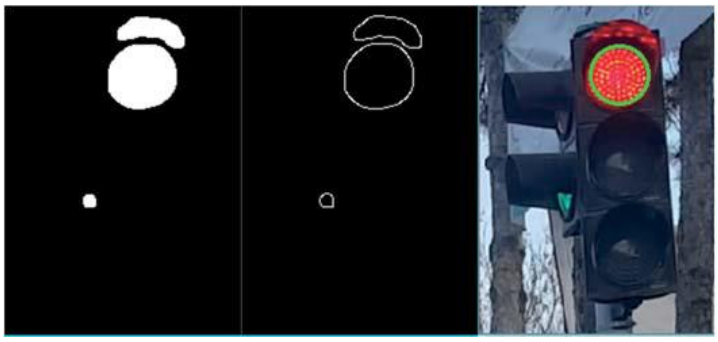
Srovnáním zjistili, že nejlepší výsledek je dosažen filtrací mediánem s jádrem 9 x 9, proto ho nakonec použili.
Následně byl převeden obraz do HSV prostoru barev, protože tento prostor definuje barvy více přirozeně a lépe přijatelně lidskému vnímání, a tak se v tomto prostoru pracuje v některých případech lépe než v klasickém RGB. Následně bylo prahováním odstraněno pozadí světla semaforu jako například lidé, ostatní části semaforu a další objekty.



(a) **(b)**

Obr. 2-19a,19b: Zobrazení obrázku v HSV se zvýrazněním některých barev (a), naprahovaný binární obraz (b), [5]

Dále byly na binární obraz aplikovány morfologické operace dilatace k potlačení drobných elementů ve světle semaforu, následně také otevření a uzavření k detekci hran světla semaforu. Poté, co byly detekovány hrany binárního obrazu, tak se nad těmito hranami provede Houghův kruhový algoritmus, který rozpozná v obrazu kruhy. Výsledky těchto tří operací jsou uvedeny v příkladu na obr.2-20



Obr. 2-20a,20b,20c: Obraz po provedení dilatace (a), detekce hran pomocí morfologických operací otevření a uzavření (b), detekce kruhových tvarů pomocí Houghova kruhového algoritmu (c), [5]

Když jsou v obraze detekovány kruhové oblasti, můžou se klasifikovat stavy semaforu. K tomu byla použita metoda strojového učení SVM. V tomto případě je objekt klasifikován do tří stavů – Červená, Žlutá, Zelená.
Trénovací dataset obsahoval přibližně 8000 snímků o rozlišení 900 x 506 pixelů.
Původní dataset byl rozdělen na tři trénovací sety, podle denní doby, ve které byly snímky pořízeny. Jejich parametry jsou zobrazeny v tabulce Tab.2-8

	Dataset 1	Dataset 2	Dataset 3
The number of total images	3945	1896	2159
The number of the current image	3577	1825	1986
The number of the invalid image	368	71	173
Success Rate	94.67%	92.33%	86.88%

Tab. 2-8: Datasety pro trénování SVM [5]

V tabulce Tab.2-9 jsou uvedeny dosažené výsledky systému.

Experiment Areas	Urban Areas	Traffic areas
Total Traffic Light	8.239	5.785
Correct Detection	7.827	5090
Incorrect Detection	412	695
Success Rate	%95	%88

Tab. 2-9: Výsledná výkonnost systému [5]

(a) (b) (c)

Obr. 2-21a,21b,21c: Ukázky klasifikace systému [5]

U tohoto systému se mi zdají nedůvěryhodné dosažené výsledky. Zavádějící je také klasifikace pouze do tří tříd, když se ve skutečnosti na semaforu mohou vyskytnout čtyři stavy – červená, žlutá, žlutá a červená, zelená.

3. DATASETY

V této kapitole uvádím a blíže popisuji tři veřejně dostupné galerie snímků a jednu vlastní galerii pro trénování a testování systému detekce semaforu a rozpoznání stavu jeho signalizačních světel.

3.1 Bosch Small Traffic Light Dataset

Tento dataset byl vytvořen výzkumníky Liborem Novákem a Karstenem Behrendtem ve spolupráci s oddělením Bosch North America Research jako součást projektu "A Deep Learning Approach to Traffic Lights: Detection, Tracking, and Classification ", který byl prezentován na konferenci ICRA 2017. Dataset je ke stažení na webových stránkách [2]. Dataset je rozdělen na část trénovací a testovací. Celkem obsahuje 13427 snímků s rozlišením 1280 x 720 pixelů, přičemž v trénovací množině je 5093 snímků, v testovacím setu je pak 8334 snímků. V těchto setech je ručně anotováno přes 24 000 semaforů (trénovací set – 10756 semaforů, testovací set – 13486 semaforů), které mají průměrnou šířkou 8.5 pixelů. Anotací se myslí zahrnutí semaforu do tzv. bounding boxu, který určuje semafor a souřadnice jeho polohy v obrázku + přiřazení aktuálního stavu semaforu, a to buď stavu červená, žlutá, zelená nebo stav off, kdy na semaforu nesvítí žádné světlo.
Snímky datasetu jsou zachyceny za různorodých podmínek, které mohou běžně při detekci a klasifikaci semaforu ve venkovním prostředí nastat a s nimiž může mít systém problém se vyrovnat. Ať už se jedná o různý charakter počasí jako je měnící se dynamický rozsah osvětlení scény nebo slabý déšť. Nebo také problém detekce v hustě osídlených městských oblastech, kde jsou víceproudé silnice nebo kde se nachází mnoho falešných potenciálních objektů, které připomínají semafory, jako jsou například brzdová světla automobilů, reklamní billboardy atd.
Na snímcích níže jsou ukázány příklady snímků z tohoto datasetu.

Obr. 3-1: Příklady snímků z Bosch Small Traffic Light Dataset [2]

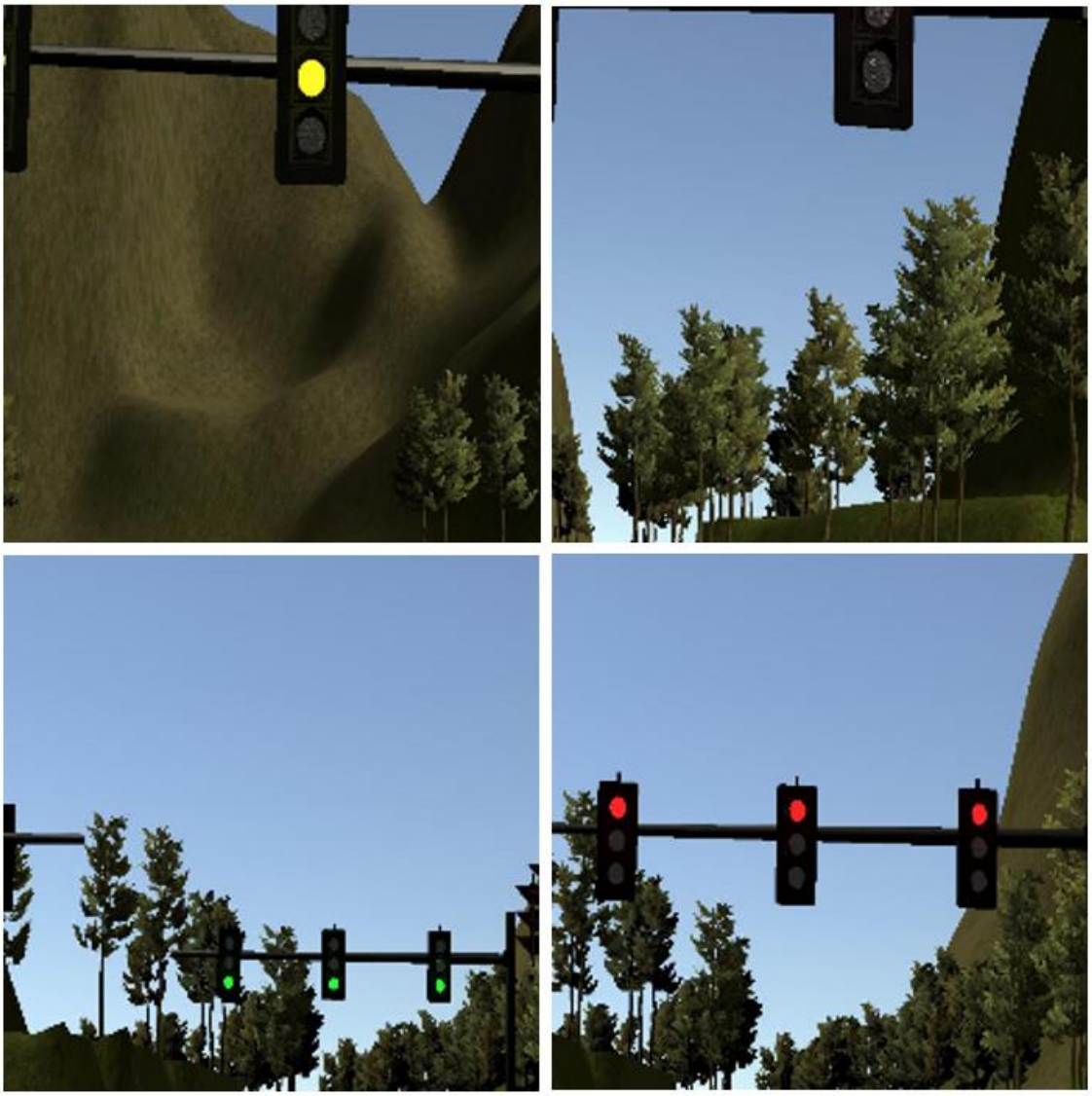
3.2 LaRA Dataset

Tento dataset pro rozpoznávání semaforu byl vytvořen v roce 2010 výzkumníky z francouzských organizací Robotics Centre of Mines Paris Tech a Imara Team of INRIA. Dataset tvoří téměř devítiminutová sekvence obsahující 11179 snímků, které jsou pořízenou v hustě zastavěné centrální části Paříže z kamery umístěné za čelním sklem automobilu. Automobil při natáčení dosahoval rychlostí menších než 50 km/h. Kamera byla vybavena čipem Marling F-046C a pořídila snímky s frekvencí 25 fps o rozlišení 640 x 480 pixelů v RGB, kde jeden pixel má hloubku 8 bitů. V sekvenci snímků bylo ručně anotováno 9168 výskytů semaforu, u kterých se klasifikoval stav zelená, oranžová, červená a stav neurčitý. Stavem neurčitý byly anotovány sporné regiony, např. pokud je daný region pouze odrazem skutečného semaforu, pokud je semafor příliš rozmazaný, pokud je semafor ve snímku příliš malý, atd. Stav zelená byl anotován v 3381 případech, oranžová v 58 případech, stav červená má 5280 výskytů a nejednoznačný stav byl určen v 449 případech.
Dataset je veřejně přístupný na stránkách [6] a je možné si ho stáhnout ve formátech MPEG-2, JPEG, JSEQ a RTMaps.
Na obrázcích níže jsou uvedeny příklady snímků z tohoto datasetu.

Obr. 3-2: Příklady snímků z LaRA datasetu [6]

3.3 Level5-Engineers Traffic Lights Datasets

Tento celek je složen ze dvou samostatných datasetů Small Dataset a Larger Dataset. Oba datasety obsahují obrázky vygenerované simulátorem a jsou volně dostupné na internetových stránkách [7]. Obrázky v obou datasetech jsou barevné a jsou uloženy ve formátu PNG. Small Dataset obsahuje 775 snímků o rozlišení 800 x 600 pixelů. Snímky nejsou anotovány přesnými souřadnicemi výskytu semaforu, ale jsou roztřízeny do složek podle stavu semaforu, a to buď stavu zelená, žlutá nebo červená. V Larger datasetu je pak 4499 obrázků o rozlišení 224 x 224 pixelů, které jsou podle stavu zařazeny ve složce červený, žlutý, zelený nebo neznámý. Do složky neznámý jsou zařazeny obrázky, na nichž se semafor vyskytuje jenom částečně a nejde tedy u něj určit stav nebo obrázky na nichž semafor není vyobrazen vůbec. Do složky červený je zařazeno 1733 obrázků, do žlutý 253, do zelený 645 a do složky neznámý 1868 obrázků.
Příklady obrázků z datasetů je uveden níže.



Obr. 3-3: Příklady snímků z level5-engineers datasetů [7]

3.4 Boček dataset

3.4.1 Výběr kamery

K pořízení videa pro vytvoření vlastního datasetu bylo potřeba vybrat a koupit kameru. V úvahu připadaly dvě kategorie kamer – outdoorové akční kamery anebo kamery do auta. Kamery do auta mají výhodu ve vyřešeném uchycení těla kamery k čelnímu sklu automobilu, outdoorové kamery jsou zase univerzálnější a tím lépe použitelné i v jiných projektech a pracích.
Při výběru jsem extrahoval důležité parametry, jako je rozlišení videa, FPS – počet snímků za sekundu, světelnost, velikost snímače. Důležitou podmínkou byl display, aby bylo možné při natáčení přesně zaměřit kameru do požadovaného prostor před automobilem. Ze zvolených parametrů jsem vytvořil kriteriální funkci. Příspěvky jednotlivých parametrů do konečné sumy kriteriální funkce jsem normalizoval, poměr velikostí maximálních příspěvků z jednotlivých parametrů jsem určil odhadem tak, aby odpovídaly důležitosti daných parametrů. Ne všechny parametry byly v popisech produktů udávány, a tak kvůli nemožnosti srovnání bylo nakonec několik parametrů z kriteriální funkce odebráno. Parametry jsem uvedl v tab.3-1, kriteriální zhodnocení v tab.3-2.

Tab. 3-1: Seznam modelů kamer s parametry

	cena [kč]	FPS [sn./s]	rozlišení	světelnost	kapacita baterie	formát souborů	display
Sony HDR-AS50 Action Cam	5990	60	1920x1080	2,8		MPEG-4	ano
GoPro HERO5 Black	8990	30	3840x2160		1220 mAh	MP4, JPEG, RAW	ano
NICEBOY VEGA 5	5990	30	3840x2160		1050 mAh	MP4, JPEG	ano
SJCAM SJ6 Legend	4990	30	2560x1440			MP4	ano
SJCAM SJ5000	3690	30	1920×1080			JPEG, MOV	ano
Olympus TG-Tracker	4999	30	3840x2160	2		DCF, MOV	ano
GoPro HERO6 Black	11490	60	3840x2160		1220 mAh	MP4, JPEG, RAW	ano
TrueCam A7s	3990	45	1920x1080		400 mAh	JPEG, MOV	ano
TrueCam A6	3490	30	1920x1080			JPEG, MOV	ano

Tab. 3-2: Kriteriální zhodnocení modelů kamer

	fps	rozlišení	RAW form	úchyt na sklo	suma
Sony HDR-AS50 Action Cam	2	0,5			2,5
GoPro HERO5 Black	1	1	0,5		2,5
NICEBOY VEGA 5	1	1			2
SJCAM SJ6 Legend	1	0,66667			1,666667
SJCAM SJ5000	1	0,5			1,5
Olympus TG-Tracker	1	1			2
GoPro HERO6 Black	2	1	0,5		3,5
TrueCam A7s	1,5	0,5		0,3	2,3
TrueCam A6	1	0,5		0,3	1,8

$$\text{fps} = \frac{2xFPS}{\max(FPS)} \qquad (1)$$

$$\text{rozlišení} = \sqrt{pocet\ px}/2880 \qquad (2)$$

RAW formát jsem ohodnotil 0,5 body a úchyt na sklo 0,3 body

Kameru GoPro HERO6 Black, která byla vyhodnocena jako nejlepší, jsme nakonec odmítli kvůli vysoké ceně. Dále jsme se tedy rozhodovali mezi Sony HDR-AS50 a GoPro HERO5 Black. Po shlédnutí řady videí, jejichž výstup byl kvalitativně srovnatelný, a po přihlédnutí k ceně jsme koupili kameru Sony HDR-AS50.

3.4.2 Popis datasetu

Boček dataset jsem vytvořil jako součást této bakalářské práce. Tento dataset jsem poté využil k trénování a testování konvolučních neuronových sítí pro systém detekce semaforu a určení signalizované barvy.
Snímky byly pořízeny začátkem jara na křižovatkách v městské části Brna Královo pole za jasného počasí.
K pořízení snímků byla použita akční kamera HDR-AS50. Video bylo natočeno v rozlišení 1920 x 1080 pixelů v RGB s frekvencí 30 snímků za sekundu a s délkou asi 12 minut. Video jsem z formátu .mov převedl na jednotlivé snímky o stejném rozlišení do formátu .jpg. Z těchto všech snímků byly vybrány pouze ty, které obsahovaly semafor a ten na nich nebyl dále než cca 60 metrů. Ručně jsem tedy nakonec vybral 1399 snímků, u kterých jsem ručně anotoval vymezení semaforu, tedy nejen signální světlo, ale i celý objekt semaforu. K anotaci jsem použil volně dostupný program LabelImg. Celkem bylo anotováno 2247 semaforů s poměrně rovnoměrným výskytem všech stavů. Rozlišil jsem stavy semaforu: zelená, oranžová, červená a oranžová současně, červená. Anotace jsou ve formátu .xml pro každý snímek zvlášť nebo hromadně v .csv souboru.
Na obrázcích níže jsou uvedeny příklady snímků z tohoto datasetu.

Obr. 3-4: Příklady obrázků z Boček dataset

3.5 Porovnání datasetů

Pro trénování neuronových sítí nebo jakékoli jiné metody strojového učení, kde je třeba systém natrénovat nebo otestovat jeho výkonnost na reálných snímcích, je třeba galerie snímků. V souhrnné tabulce Tab.3-3 uvádím kvantitativní parametry jednotlivých datasetů, které jsem popsal v předchozích podkapitolách. Spolu s těmito datasety srovnávám také vlastní Boček Dataset.

Tab. 3-3: Srovnávací tabulka základních parametrů popisovaných datasetů

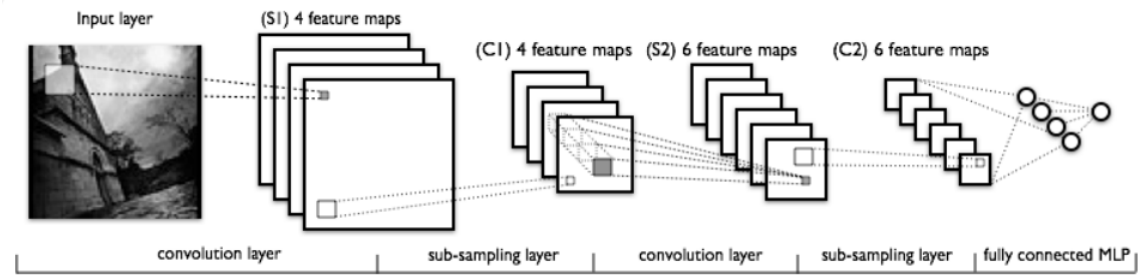
	celkem snímků	trénovací snímky	testovací snímky	anotované semafory	rozlišení
Bosch Small Traffic Lights Dataset	13427	5093	8334	24242	1280 x 720
LaRA dataset	11179	-	-	9168	640 x 480
level5 engineers - Small Dataset	775	-	-	neurčeno	800 x 600
level5 engineers - Larger Dataset	4499	-	-	neurčeno	224 x 224
Boček Dataset	1399	-	-	2247	1920 x 1080

4. KONVOLUČNÍ NEURONOVÉ SÍTĚ

Tato kapitola se věnuje konvolučním neuronovým sítím, náhledu do jejich struktury a vysvětlení významu a funkcí jednotlivých druhů vrstev, které se v konvolučních sítích používají.
Zmíněny jsou především ty termíny, které jsem využil a nastavoval u svého navrženého systému této práce.
Konvoluční neuronové sítě jsou speciálním druhem neuronových sítí. Využívají se především v aplikacích zpracování obrazu k rozpoznávání vzorů v obraze. Obecně jsou vhodné pro jakékoli dvourozměrné signály, resp. signály s danou strukturou.
Jejich vznik se datuje do roku 1980, kdy K. Fukishima vymyslel Neocognitron. Ten totiž obsahuje mnoho mechanismů, které se využívají v dnešních konvolučních neuronových sítích. Největší rozvoj a nástup konvolučních neuronových sítí ale nastal po roce 2012, kdy Krizhevsky a jeho tým vyhrály prestižní soutěž v klasifikaci ImageNet Challenge. Rozvoj konvolučních neuronových sítí je, na rozdíl od minulých desetiletí, umožněn výpočetním výkonem dnešních počítačů s výkonnými grafickými kartami [8].

4.1 Struktura

U konvolučních sítí nejsou přesně daná pravidla, jak sestavit jednotlivé vrstvy sítě, ale většinou jsou strukturovány na dvě části. V první části jsou pomocí konvolučních a subsamplingových vrstev ze vstupního obrazu extrahovány příznaky. V druhé části jsou pak vrstvy plně propojené a zde dochází ke klasifikaci [9].

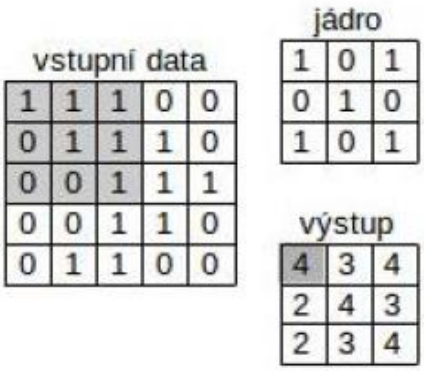


Obr. 4-1: Příklad struktury konvoluční neuronové sítě [10]

Konvoluční neuronové sítě se skládají ze tří základních druhů vrstev. Těmi jsou konvoluční, plně propojené a subsamplingové vrstvy. Mezi těmito vrstvami se navíc můžou provádět různé operace, které danou konvoluční síť formují anebo pomáhají zvyšovat úspěšnost učení. Jsou to například operace flatten a dropout. Výsledky výpočtů v těchto základních vrstvách lze ovlivnit velikostí jádra, aktivační funkcí dané vrstvy, množstvím filtrů v dané vrstvě, parametrem stride nebo zero padding.

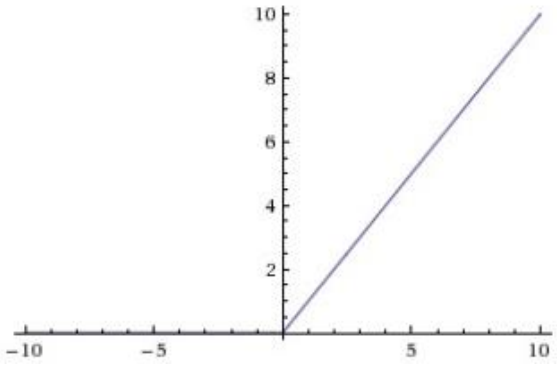
4.2 Konvoluční vrstva

Je to to dopředná neuronová vrstva, která provádí operaci konvoluce se vstupním obrazem nebo v hlubších vrstvách s příznakovými mapami. Výstup této vrstvy se nazývá mapa příznaků.
Příklad 2D konvoluce vstupu a jádra filtru je ukázán na obr.4-2.



Obr. 4-2: Ukázka 2D konvoluce [11]

U konvoluce se dá nastavit parametr stride. Stride určuje o kolik pozic se bude konvoluční jádro při konvoluci posunovat. Při klasické konvoluci se stride rovná 1, čili projde všechny možné pozice v matici obrazových elementů. To ale znamená, že jádro o velikosti 3x3 projde jeden pixel celkem 9x, což může být zbytečné a neefektivní, proto se podle velikosti jádra často parametr stride nastavuje vyšší.
Dále lze aktivovat parametr zero padding, ten před provedením konvoluce doplní vstupní matici o vnější rámec nul, aby se provedením konvoluce nezmenšil rozměr obrazové matice. Z obr.4-2 vidíme, že pokud vrstvu nul vně vstupní matice nepřidáme, výstupní mapa příznaků se nám zmenší v obou osách o velikost filtru v dané ose mínus jedna.
Stejně jako u plně propojených vrstev i u konvolučních vrstev se používají aktivační funkce. U konvolučních vrstev se ve velké většině případů používá Rectified Linear Unit – ReLU. Její charakteristika je zobrazena na obr.4-3.

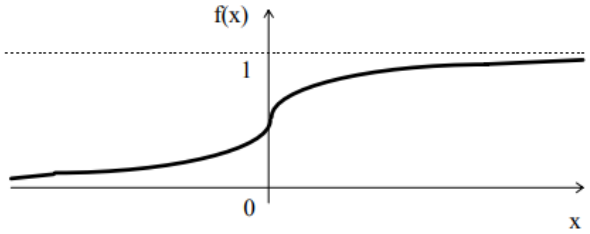


Obr. 4-3: Aktivační fukce - ReLU [12]

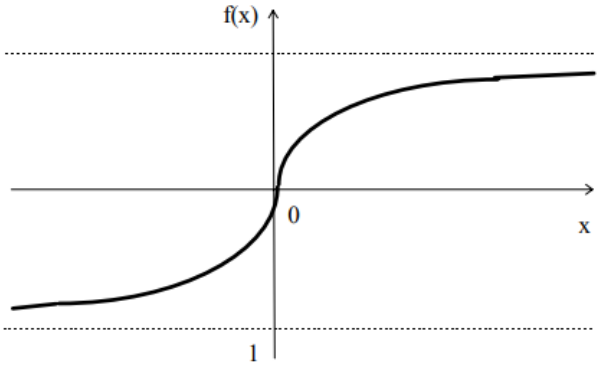
$$f(x) = \max(x, 0) \tag{5}$$

4.3 Plně propojená vrstva

Je to vrstva neuronů, kde jsou neurony propojeny jako v klasické neuronové síti. Takže každý neuron v dané vrstvě je propojen se všemi neurony ve vrstvě následující. Tyto vrstvy bývají umístěny v koncové části celé konvoluční neuronové sítě, kde klasifikují objekty v obraze do různých tříd.
Výstup z poslední plně propojené vrstvy je většinou následně ještě přepočítán funkcí soft-max. Neurony v této finální vrstvě vyjadřují míru pravděpodobnosti, že klasifikovaný objekt patří do dané třídy. Funkce softmax má tu funci, že přepočítá výstupy neuronů tak, aby celková suma těchto výstupů byla rovna jedné.
U plně propojených vrstev se na rozdíl od konvolučních aktivační funkce ReLU příliš často nepoužívá, ale využívají se nelineární aktivační funkce hyperbolická tangenta nebo sigmoida. Jejich charakteristiky uvádím na obrázcích níže.



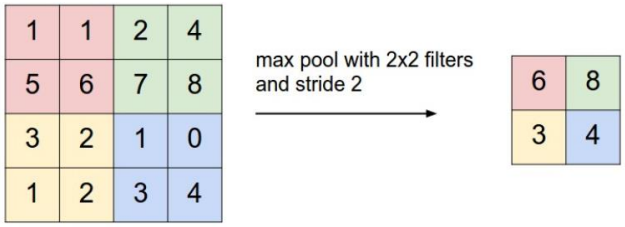
Obr. 4-4: Aktivační funkce – sigmoida [13]



Obr. 4-5: Aktivační funkce – hyperbolická tangenta [13]

4.4 Subsamplingová (Poolingová) vrstva

Tyto vrstvy se umísťují mezi jednotlivé konvoluční vrstvy. Jejich úkolem je zredukovat velikost vstupu do další konvoluční vrstvy kvůli snížení výpočetní náročnosti a odstranění přebytečných dat. Existuje několik typů funkcí jako například střední hodnota nebo nejvyšší hodnota, kterými je možno poolingovou vrstvu realizovat.
V praxi se nejčastěji využívá maxpoolingová vrstva, která z dané oblasti vymezeném velikostí poolingového jádra vybere pixel s největší hodnotou.

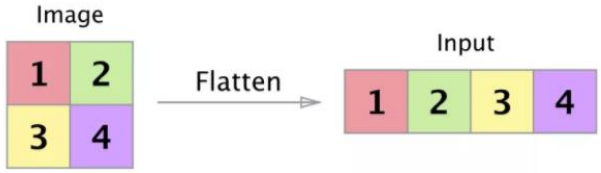


Obr. 4-6: Ukázka funkce maxpoollingu [14]

Na příkladu maxpoolingu na obr.4-6 je velikost poolingového jádra 2x2 a stride je roven dvěma, to znamená, že se žádná vstupní hodnota nemůže být na výstup zapsána vícekrát. Někdy se ale využívá toho, že se poolingové hodnoty překrývají, toho by bylo dosaženo zvolením většího jádra nebo menšího stride.

4.5 Flatten operace

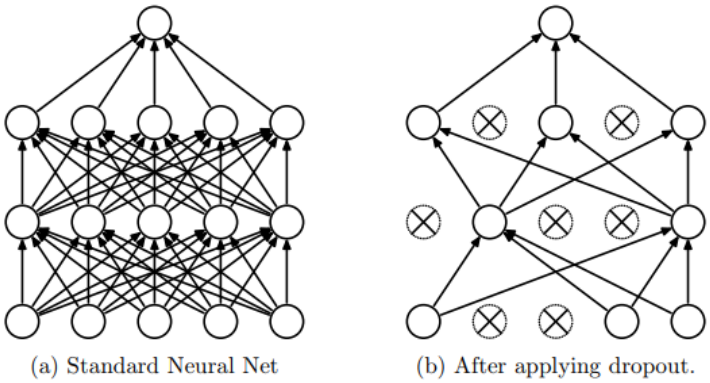
Tato operace zformuje data z vícerozměrné příznakové mapy do jednorozměrného vektoru. Operace flatten se provádí před první plně propojenou vrstvou, kde klasická neuronová síť požaduje vstup ve tvaru vektoru.



Obr. 4-7: Ukázka operace flatten [15]

4.6 **Dropout**

Dropout je operace, která slouží k tomu, aby se předešlo přeučení neuronové sítě. V každé iteraci učení je deaktivována určitá část neuronů – většinou 0,05 – 0,5 z celkového počtu. Tím se zajistí, že konečný výstup neuronové sítě nebude ovlivňovat pouze malá část neuronů a že ostatní neurony budou mít pouze malý vliv, ale na výsledku se budou podílet všechny neurony rovnoměrně. Příklad operace dropout je uveden na obrázku níže.



Obr. 4-8: Model dropoutu. Vlevo standardní neuronová síť. Vpravo neuronová síť, kde je aplikován dropout, škrtnuté neurony jsou pro danou iteraci vyřazeny [16]

5. NÁVRH VLASTNÍHO DETEKTORU SEMAFORŮ

Tato kapitola se zabývá seznámením s průběhem vývoje detektoru a klasifikátoru, dále popisuje strukturu celého systému a nakonec jsou také uvedeny výsledky testování, kterých bylo dosaženo.

5.1 Práce v jazyku Python

Minulý semestr jsem se zabýval v semestrální práci průzkumem metod, pomocí kterých se různí výzkumníci, ať už na různých světových univerzitách nebo ve významných společnostech zabývajících se počítačovým viděním, snažili detekovat semafor a určit stav jeho světel. Tento úkol byl ve většině případů řešen buď klasickou způsobem pomocí předzpracování obrazu, následné segmentaci, popisem a na konci tohoto řetězce byla použita ke klasifikaci metoda SVM (Support Vector Machine) anebo jiná podobná metoda. Anebo druhou možností, která byla často používána, je detekovat a klasifikovat stavy semaforu pomocí konvolučních neuronových sítí. Výhody aplikování první zmíněné možnosti by pravděpodobně byla rychlost učení klasifikátoru, byla by potřeba menší galerie dat a nejspíš by tento úkol byl i jednodušší na realizaci. Avšak podle prezentovaných výsledků ve vědeckých článcích by systém dosahoval obecně nižších úspěšností F1 score a kvůli metodám předzpracování obrazu, kde se často segmentuje objekt a pozadí podle jasových hodnot pixelů v obraze, by nebyl tak robustní při detekci semaforů za zhoršených světelných podmínek. Proto jsem se rozhodl řešit zadaný úkol pomocí konvolučních neuronových sítí.

Konvoluční síť jsem si nechtěl od začátku psát sám, protože k tomu je potřeba mnoho hlubokých znalostí a zkušeností s volbou uspořádáním struktury sítě a navíc je trénování konvolučních neuronových sítí časově velmi náročné – některé sítě se trénují i po dobu několika měsíců - a je k tomu zapotřebí mnoho tréninkových obrázků, kterých mohou být desítky i stovky tisíc. Proto jsem chtěl jít cestou tzv. transfer learningu. Na různých IT webových stránkách a repositářích se dají najít již vytvořené a předtrénované konvoluční sítě. Předtrénovaná síť je síť, kterou už někdo vytvořil a natrénoval na detekci nebo klasifikaci určitých obrázků. Tato síť má tedy již nějak nastavené váhy v jednotlivých vrstvách a konvolučních filtrech. Využívá se toho, že v počátečních vrstvách se detekují jednoduché elementy jako jsou hrany, barvy, a tyto elementy se v těchto vrstvách hledají víceméně při rozpoznávání jakékoliv třídy. Potom když je potřeba takovou síť natrénovat na jiné objekty, tak se pak nejčastěji nově vytvoří a nahradí pouze posledních několik vrstev, které bývají většinou plně propojené, a síť se doučí na klasifikaci nebo detekci objektů daných tříd. Toto dodatečné doučení už není tak časově náročné a je potřeba podstatně méně trénovacích snímků, než když se síť učí od začátku.

Já jsem si tedy našel na webovém repozitáři GitHub článek [17], v kterém se pracovalo s veřejně dostupnou konvoluční neuronovou sítí Fast R-CNN a bylo zde i detailně vysvětleno, co je potřeba vykonat, aby se síť natrénovala. Velikost této sítě se všemi náležitostmi byla 142 MB. Kód souborů sítě byl psán v jazyku Python a využíval

TensorFlow, což je framework využívaný pro aplikace strojového učení vyvíjený společností Google.
Na pracovišti UAMT v laboratoři počítačového vidění mi byl umožněn přístup na počítač s výkonnou grafickou kartou Nvidia Tesla C2075.

Obr. 5-1: Nvidia Tesla C2075 [18]

Na počítač jsem nainstaloval Python 3.6 a další potřebné balíčky, softwarovou architekturu CUDA 9.0 pro práci s grafickou kartou, dále bylo třeba také nainstalovat TensorFlow. Na počítači mi nefungovala nejnovější verze 1.7, bylo třeba použít starší verzi 1.5. K urychlení tréninku konvoluční sítě bylo potřeba využít TensorFlow, který provádí výpočty na grafické kartě. Při tom mi ale vždy byly hlášeny chyby. K jejich odstranění jsem vyzkoušel přeinstalovat ovladače grafické karty, přeinstalovat CUDu, a realizovat další rady, které doporučovali na internetových fórech, ale problém se tím nevyřešil. Po třech dnech práce strávených hledáním těchto chyb jsem zjistil, že TensorFlow při využití grafické karty vyžaduje, aby grafická karta měla parametr Compute capability větší nebo roven 3.0. To grafická karta Nvidia Tesla C2075 s Compute capability rovným 2.0 nesplňovala.
Hodnota tohoto parametru odpovídá tomu, jak rozsáhlou má grafická implementovanou instrukční sadu.
Poté, co tato karta nevyhovovala jsem tedy sháněl jinou grafickou kartu, našla se herní Nvidia GeForce GTX 570, která měla dobré výpočetní parametry, ale rovněž měla Compute capability 2.0. Proto mi byl přidělen jiný počítač vybavený grafickou kartou Nvidia GeForce GT 630, která je sice oproti předchozím dvěma o něco méně výkonná, ale je novější, a tak má Compute capability 3.5. Na tomto počítači již Tensorflow, byť jeho starší verze, fungoval. Přednaučenou konvoluční síť se již povedlo začít trénovat, ale místo toho, aby chyba sítě konvergovala k nulové hodnotě, tak se vývoj tréninku nijak zásadně nezlepšoval. S programováním Pythonu a opravováním rozdílností mezi jednotlivými verzemi TensorFlow a Pythonu nemám moc zkušeností, navíc u takto složité sítě, a tak jsem si po nějaké době, co jsem pracoval s touto předtrénovanou sítí uvědomil, že to není cesta vedoucí k cíli a že zadaný úkol budu muset vyřešit jiným způsobem.

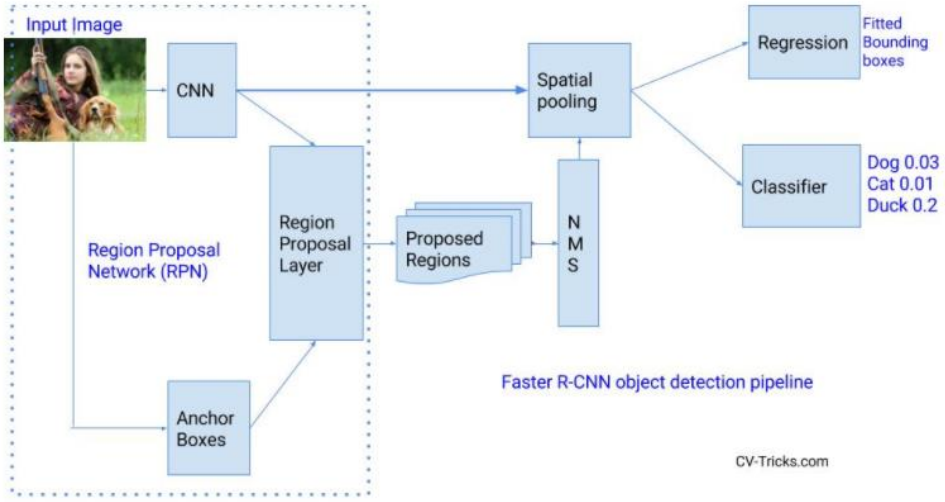
5.2 Popis systému

Po počátečních problémech, kdy jsem se snažil natrénovat předtrénovanou síť Fast R-CNN psanou v Pythonu, jsem hledal jinou možnost, jak úkol detekce semaforu úspěšně vyřešit pomocí konvolučních neuronových sítí. Z možných programovacích jazyků jsem si vybral Matlab, pro jeho výhody jako je například rychlá implementace nových metod, snadná interpretace výsledků, široká podpora metod strojového učení včetně konvolučních neuronových sítí a dobře zpracovaná internetová dokumetnace.
Pro trénování sítí v Matlabu, stejně jako v Tensorflow, při využití grafické karty bylo nutné, aby parametr grafické karty Compute capability byl minimálně 3.0, proto jsem ani zde nevyužil výkonnou Nvidii Teslu.
Ze všeho nejdříve bylo potřeba na počítači, který jsem využíval k tréninku sítí, nainstalovat programovací prostředí Matlab od společnosti Mathworks. Na webových stránkách Centra výpočetních a informačních služeb VUT jsem stáhl a posléze nainstaloval nejnovější dostupnou verzi, tedy Matlab 2017b. Pro práci s konvolučními neuronovými sítěmi dále bylo třeba doinstalovat toolboxy Computer Vision System Toolbox™, Image Processing Toolbox™ a Neural Network Toolbox™.

Celý navržený systém detekce semaforu a rozpoznání stavu jeho světel se skládá ze základních dvou kroků. Prvním je detekce semaforu, kde výstupem jsou pouze souřadnice semaforu, které určují jeho pozici v obraze, a v druhém kroku se klasifikuje stav semaforu, případně se v tomto kroku také odstraní případy False positive, tedy pokud byl chybně detekován semafor na místě v obraze, kde ve skutečnosti není.

5.2.1 Detektor

K detekci semaforů jsem použil konvoluční neuronovou síť Faster R – CNN. Je to složitá speciální konvoluční síť, která obsahuje ve skutečnosti sítě dvě. Jedna z nich je "Region proposal network", která generuje potenciální bounding boxy, a následující sít spolu s dalšími rozhodovací algoritmy vybírají ty nejpravděpodobnější bounding boxy a ty dává na výstup. Faster R-CNN se využívá k detekci a někdy i ke klasifikaci objektů. Já ji však používám pouze k detekci semaforu, protože tím získám lepší kontrolu nad celým systémem a nad výkonností obou jednotlivých částí – klasifikátoru a detektoru.
Výhodou použití této sítě je, že se v Matlabu nemusí vytvářet celá se všemi potřebnými částmi, ale stačí napsat pouze klasifikační síť pro výběr nejpravděpodobnějších bounding boxů a zbylé části Faster R-CNN jsou doplněny automaticky při překladu trénovací funkce.



Obr. 5-2: Architektura Faster R-CNN [19]

K trénování tohoto detektoru jsem použil Boček dataset, který je součástí této bakalářské práce. V něm je anotováno téměř 1400 snímků, na kterých je vyznačeno přes 2200 semaforů. Anotace k těmto obrázkům je uložena v csv souboru ve tvaru, který není po načtení není v tom správném tvaru potřebném pro vytvoření datastoru, proto se na začátku skriptu zabývám upravením uspořádání názvů obrázků a souřadnic semaforů do správného tvaru. Poté jsem rozdělil dataset na trénovací a testovací set v poměru 0,6. Dále bylo třeba vytvořit jednotlivé vrstvy konvoluční neuronové sítě. Se strukturou sítě jsem se inspiroval u článku na internetových stránkách dokumentace Matlabu. U vstupní vrstvy, čím je menší velikost jádra, tím je síť schopna detekovat menší objekty. Velikost tohoto jádra by měla být menší než velikost nejmenších objektů, které chci detekovat. Protože semafor, který je snímaný z větší vzdálenosti, je v porovnání s ostatními proporcemi obrázku malý, tak se mi nejlépe osvědčilo malé vstupní jádro 5 x 5 x 3. To je ale zaplaceno delším trénováním. Na obrázku obr.5-3 je uvedena struktura vrstev sítě detektoru.

```
 1   ''   Image Input             5x5x3 images with 'zerocenter' normalization
 2   ''   Convolution             32 3x3 convolutions with stride [1  1] and padding [1  1  1  1]
 3   ''   ReLU                    ReLU
 4   ''   Convolution             32 3x3 convolutions with stride [1  1] and padding [1  1  1  1]
 5   ''   ReLU                    ReLU
 6   ''   Max Pooling             3x3 max pooling with stride [2  2] and padding [0  0  0  0]
 7   ''   Fully Connected         64 fully connected layer
 8   ''   ReLU                    ReLU
 9   ''   Fully Connected         2 fully connected layer
10   ''   Softmax                 softmax
11   ''   Classification Output   crossentropyex
```

Obr. 5-3: Uspořádání vrstev ve Faster R-CNN

Nastavení různých parametrů trénování se provádí pomocí matlabovské interní funkce trainingOptions, kde na místě prvního parametru se uvádí optimizační algoritmus, dále se pak v této funkci na místech dalších parametrů můžou aktivovat různé operace, jako například ukládání průběžných výsledků trénování, nebo se dá z defaultních hodnot

přenastavit například velikost učícího kroku, velikost dávky, počet epoch tréninku a další. Důležitými parametry jsou NegativeOverlapRange a PositiveOverlapRange. Při detekci je objekt málo kdy určen přesně těmi souřadnicemi, které byly zadány v datasetu jako správné – ground truth. Aby bylo při tréninku možné vyhodnotit správnost detekce nastavují se těmito parametry meze, kdy objekt je, případně není detekován. Výstupem této funkce je struktura nastavení, která se vloží jako parametr do trénovací funkce.
Samotné trénování Faster R-CNN je realizováno ve funkci trainFasterRCNNObjectDetector. Na pozici prvního parametru se dá trénovací dataset, na pozici druhého argumentu je vytvořená struktura sítě, na třetí pozici pak nastavení tréninku. Výstupem této fukce je natrénovaná detekční síť.

Výsledné testování systému se provádí pomocí matlabovské funkce detect, kde na prvním místě vstupních argumentů této funkce je natrénovaná síť Faster R-CNN a na pozici druhého argumentu je vyhodnocovaný obrázek. Výstupem této funkce jsou souřadnice bounding boxů detekovaných objektů a skóre, čili pravděpodobnost, se kterou byl semafor správně detekován. V cyklu se takto detekují semafory na všech snímcích testovacího datasetu a výstupy této funkce se ukládají do výsledné struktury. Tato struktura se následně vloží jako první vstupní argument do funkce evaluateDetectionPrecision, druhým vstupním argumentem jsou správně souřadnice semaforů v obrázcích. Výstupem této vyhodnocovací funkce jsou vektory precision a recall, které následně tisknu do grafu. Na závěr jsem ještě napsal algoritmus pro vyhodnocení detektoru pomocí čtyřpolní tabulky. Vyhodnocuje se zde porovnáním predikovaných souřadnic a správných souřadnic semaforů, zda byl semafor určen správně – true positive, chybně v místě, kde ve skutečnosti není – false positive, nebo pokud nebyl detekován v místě, kde ve skutečnosti je – false negative. Stav true negative v případě detektoru nenastane.

Při spuštění tréninku jsem se potýkal s problémy ohledně velikosti snímků. Původní snímky, které byly v HD rozlišení, nešly použít, protože nároky kladené na výkonnost grafické karty, přesahovaly její možnosti. Rozlišení snímků jsem snižoval až na rozlišení 400 x 223, které už vyhovovalo. Tímto krokem se však ztratilo mnoho informací z obrazu. Velice se tím omezila spolehlivost detekce vzdálenějších semaforů přibližně od 40 metrů a dále, protože na snímcích poté nebyly téměř viditelné. Také to přineslo chybu do tréninku, protože po přepočtu souřadnic na nové rozměry snímků se detektor učí v některých případech detekovat semafory, které nejsou téměř rozpoznatelné a tím je síť při učení mystifikována.

5.2.2 Klasifikátor

Vstupem do klasifikátoru jsou výřezy z obrázků potenciálních semaforů, které určil detektor. Výstupem je pak pro každý výřez jeden ze stavů green, orange, red_and_orange, red nebo none. Stav none slouží k odstranění falešných detekcí detektoru. Tento stav je dán na výstup, pokud byl výřez klasifikován jako pozadí.
Klasifikátor je realizován konvoluční sítí AlexNet. Tato síť byla poprvé použita v roce 2012 na klasifikační soutěži ImageNet Challenge, kde se soutěží v klasifikaci objektů do 1000 různých tříd, a tento ročník s náskokem vyhrála.

```
 1   'data'     Image Input                  227x227x3 images with 'zerocenter' normalization
 2   'conv1'    Convolution                  96 11x11x3 convolutions with stride [4  4] and padding [0  0  0  0]
 3   'relu1'    ReLU                         ReLU
 4   'norm1'    Cross Channel Normalization  cross channel normalization with 5 channels per element
 5   'pool1'    Max Pooling                  3x3 max pooling with stride [2  2] and padding [0  0  0  0]
 6   'conv2'    Convolution                  256 5x5x48 convolutions with stride [1  1] and padding [2  2  2  2]
 7   'relu2'    ReLU                         ReLU
 8   'norm2'    Cross Channel Normalization  cross channel normalization with 5 channels per element
 9   'pool2'    Max Pooling                  3x3 max pooling with stride [2  2] and padding [0  0  0  0]
10   'conv3'    Convolution                  384 3x3x256 convolutions with stride [1  1] and padding [1  1  1  1]
11   'relu3'    ReLU                         ReLU
12   'conv4'    Convolution                  384 3x3x192 convolutions with stride [1  1] and padding [1  1  1  1]
13   'relu4'    ReLU                         ReLU
14   'conv5'    Convolution                  256 3x3x192 convolutions with stride [1  1] and padding [1  1  1  1]
15   'relu5'    ReLU                         ReLU
16   'pool5'    Max Pooling                  3x3 max pooling with stride [2  2] and padding [0  0  0  0]
17   'fc6'      Fully Connected              4096 fully connected layer
18   'relu6'    ReLU                         ReLU
19   'drop6'    Dropout                      50% dropout
20   'fc7'      Fully Connected              4096 fully connected layer
21   'relu7'    ReLU                         ReLU
22   'drop7'    Dropout                      50% dropout
23   'fc8'      Fully Connected              1000 fully connected layer
24   'prob'     Softmax                      softmax
25   'output'   Classification Output        crossentropyex with 'tench', 'goldfish', and 998 other classes
```

Obr. 5-4: Uspořádání vrstev v konvoluční neuronové síti AlexNet

Použitím této sítě jsem se inspiroval z článku [3], který podrobněji rozebírám v kapitole rozboru metod. Tuto předtrénovanou konvoluční neuronovou síť jsem nainstaloval z nabídky doplňkových toolboxů a rozšíření Matlabu.
Protože se jedná o klasifikační síť, rozměry vstupní vrstvy jsou shodné s rozměry klasifikovaného obrázku. Síť je trénována pouze na výřezech semaforu. K vytvoření datasetu pro tento klasifikátor jsem napsal funkci, která vyřeže výřezy semaforů a uloží je do složky. Rozměry těchto výřezů jsou určeny anotovanými souřadnicemi, a proto mají tyto rozměry nejednotnou velikost. Protože vstup této konvoluční neuronové sítě akceptuje matici obrazových elementů o rozměrech 227 x 227 x 3, tak vždy, když se načítá snímek z datastoru jsou funkcí resize upraveny jeho rozměry na požadovanou velikost.
Pomocí funkce splitEachLabel se následně rozdělí datastore na trénovací a testovací set ve zvoleném poměru počtů obrázků.
Protože struktura sítě byla již hotová, tak ji nebylo nutné vytvářet od začátku. Pouze bylo potřeba změnit některé vnitřní vrstvy. Poslední plně propojenou vrstvu, která obsahovala 1000 neuronů jsem nahradil plně propojenou vrstvou s pěti neurony z důvodu klasifikace do pěti tříd. Tyto koncové neurony vyjadřují pravděpodobnost, že na daném výřezu je objekt dané třídy.
Při spuštění trénování na grafické kartě jsem narazil na problém, že nestačila velikost její paměti. Správně jsem zjistil, že k chybě dochází ve dvou prvních plně propojených vrstvách sítě, proto jsem problém vyřešil snížením počtu neuronů v obou vrstvách z 4096 na 3072 neuronů. Struktura sítě po úpravách je na obrázku níže.

1	'data'	Image Input	227x227x3 images with 'zerocenter' normalization
2	'conv1'	Convolution	96 11x11x3 convolutions with stride [4 4] and padding [0 0 0 0]
3	'relu1'	ReLU	ReLU
4	'norm1'	Cross Channel Normalization	cross channel normalization with 5 channels per element
5	'pool1'	Max Pooling	3x3 max pooling with stride [2 2] and padding [0 0 0 0]
6	'conv2'	Convolution	256 5x5x48 convolutions with stride [1 1] and padding [2 2 2 2]
7	'relu2'	ReLU	ReLU
8	'norm2'	Cross Channel Normalization	cross channel normalization with 5 channels per element
9	'pool2'	Max Pooling	3x3 max pooling with stride [2 2] and padding [0 0 0 0]
10	'conv3'	Convolution	384 3x3x256 convolutions with stride [1 1] and padding [1 1 1 1]
11	'relu3'	ReLU	ReLU
12	'conv4'	Convolution	384 3x3x192 convolutions with stride [1 1] and padding [1 1 1 1]
13	'relu4'	ReLU	ReLU
14	'conv5'	Convolution	256 3x3x192 convolutions with stride [1 1] and padding [1 1 1 1]
15	'relu5'	ReLU	ReLU
16	'pool5'	Max Pooling	3x3 max pooling with stride [2 2] and padding [0 0 0 0]
17	''	Fully Connected	3072 fully connected layer
18	'relu6'	ReLU	ReLU
19	'drop6'	Dropout	50% dropout
20	''	Fully Connected	3072 fully connected layer
21	'relu7'	ReLU	ReLU
22	'drop7'	Dropout	50% dropout
23	''	Fully Connected	5 fully connected layer
24	'prob'	Softmax	softmax
25	''	Classification Output	crossentropyex

Obr. 5-5: Struktura vrstev klasifikační sítě po úpravě z AlexNet

Nastavení parametrů trénování se nastavuje stejně jako u Faster R-CNN pomocí funkce traininigOptions. Síť jsem trénoval pomocí funkce trainNetwork, kde prvním argumentem je trénovací datastore, druhým je struktura vrstev konvoluční sítě a třetím argumentem jsou nastavení tréninku. Výstupem této trénovací funkce je natrénovaná síť a informace o tréninku, které jsou reprezentovány průběhy přesnosti během učení a průběh chybové funkce během učení.
U případu klasifikace do těchto pěti stavů mohou nastat případy, kdy bude výsledek True Positive, to je, když bude stav určen správně a True Negative, když je stav určen chybně. False negative a True negative případy nenastanou, protože klasifikátor musí vždy klasifikovat výřez do jednoho ze stavů.

Obr. 5-6: Příklady výřezů z Boček datasetu použitých k trénování klasifikátoru

Prvně jsem navrhl klasifikátor, který rozlišoval pouze čtyři stavy “green”, “orange”, “red” a “red_and_orange”. Takže mohl nastat případ, že pokud je detekční sítí chybně jako semafor detekován objekt v pozadí, čili něco co reálně není semafor, tak se tento objekt také klasifikuje do jedné z tříd stavu semaforu. Předpokládal jsem totiž, že v tomto chybném případě bude výsledek klasifikace dost nejednoznačný. Například, že každému ze všech čtyř stavů bude v poslední plně propojené vrstvě přidělena nízká pravděpodobnost v rozsahu od 0 do 40 %. Takovéto nejasné klasifikace jsem pak s přihlédnutím na pravděpodobnost správné detekce odstranil prahováním. Při kontrole funkčnosti systému jsem ale zjistil, že takto systém nepracuje tak spolehlivě, jak by bylo

potřeba, a nejsem schopen prostým prahováním výstupů neuronů odstranit nepřesnosti klasifikátoru způsobené tímto přístupem. Proto jsem v Boček datasetu, který jsem k trénování používal, ručně označil třídou "none" objekty a části ve snímcích, které semafor neobsahují. Tyto objekty jsem vybíral jednak náhodně, pouze aby jich byl dostatečný počet, ale také jsem se snažil v dostatečné míře zahrnout kritické objekty jako jsou zadní světla automobilů, zadní desky semaforů a další objekty, které by se mohly jevit jako semafor, aby se je síť naučila rozlišovat. Poměr "none" objektů a ostatních vyznačených semaforů v galerii dat byl v poměru přibližně 3:4.
Po této úpravě, kdy se k vyřazení falešných detekcí semaforu využije samostatná klasifikační třída místo jednoduché metody prahování, se souhrnné výsledky detektoru a klasifikátoru zlepšily.

5.3 Vyhodnocení experimetů

Tato podkapitola se věnuje výsledkům a zhodnocení testování výkonnosti natrénovaných detektorů a klasifikátorů. Zaměřuje se na porovnání výsledků a průběhů tréninku sítí s různými trénovacími parametry, kterými jsou počet epoch tréninku, velikost dávky, velikost vstupního konvolučního jádra a rozlišení trénovacích snímků.
Nakonec byla experimentálně zvolena vhodná kombinaci detektoru a klasifikátoru a na vzorku snímků z Boček datasetu a snímcích z Google map byla ověřena výkonnost systému.

5.3.1 Detektor

Navrhl a natrénoval jsem 9 detektorů s různými trénovanými parametry a s různou velikostí vstupní vrstvy sítě. Mezi tyto parametry patří velikost trénovacího setu, počet epoch tréninku a počet snímků v jedné trénovací dávce. Seznam detektorů s jejich parametry uvádím v tabulce níže.

Tab. 5-1: Seznam detektorů s jejich parametry.

označení	trénovací set -počet	rozlišení snímků [px]	epochy tréninku	velikost dávky	vstupní vrstva
detektor_1	839	400 x 223	30	32	5 x 5 x 3
detektor_2	839	400 x 223	30	32	10 x 10 x 10
detektor_3	839	400 x 223	30	32	32 x 32 x 32
detektor_4	839	400 x 223	15	128	5 x 5 x 3
detektor_5	839	400 x 223	30	128	5 x 5 x 3
detektor_6	839	400 x 223	90	128	5 x 5 x 3
detektor_7	489	400 x 223	20	128	6 x 6 x 3
detektor_8	489	400 x 223	70	128	6 x 6 x 3
detektor_9	489	400 x 223	50	128	5 x 5 x 3

Těchto devět detekčních konvolučních sítí jsem testoval na 561 testovacích snímcích z Boček datasetu o rozlišení 400 x 223 pixelů, na těchto snímcích nebyly detektory trénovány. Čtyřpolní tabulku výsledků a uvádím níže.

Tab. 5-2: Čtyřpolní tabulka výsledků testovaných detektorů

označení	TP	FP	FN	TN	recall	precision
detektor_1	681	3218	251	-	0,731	0,175
detektor_2	593	2959	339	-	0,636	0,167
detektor_3	19	6463	913	-	0,020	0,003
detektor_4	691	3639	241	-	0,741	0,160
detektor_5	581	1002	351	-	0,623	0,367
detektor_6	684	1922	248	-	0,734	0,263
detektor_7	631	2994	301	-	0,677	0,174
detektor_8	667	1495	265	-	0,716	0,309
detektor_9	625	1290	307	-	0,671	0,326

$$citlivost\ (recall) = \frac{TP}{TP+FN} \quad \textbf{(6)}$$

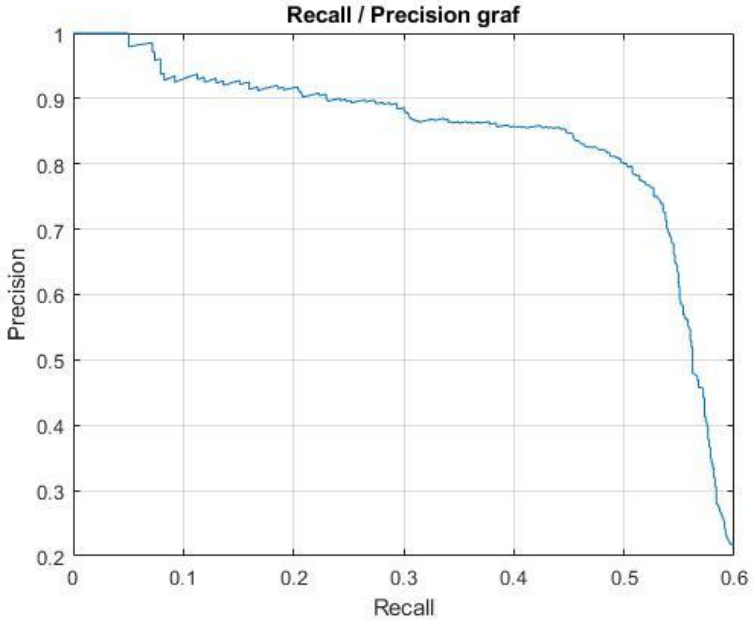
$$přesnost(precision) = \frac{TP}{TP+FP} \quad \textbf{(7)}$$

Z tabulky tab. 5-2 můžeme u prvních tří detektorů, které se liší rozměry vstupního filtru sítě, pozorovat, že úspěšnost detekce je vyšší u detektorů s menšími rozměry vstupní vrstvy. To je dáno tím, že s menším vstupem je síť schopna se naučit rozpoznávat menší objekty.

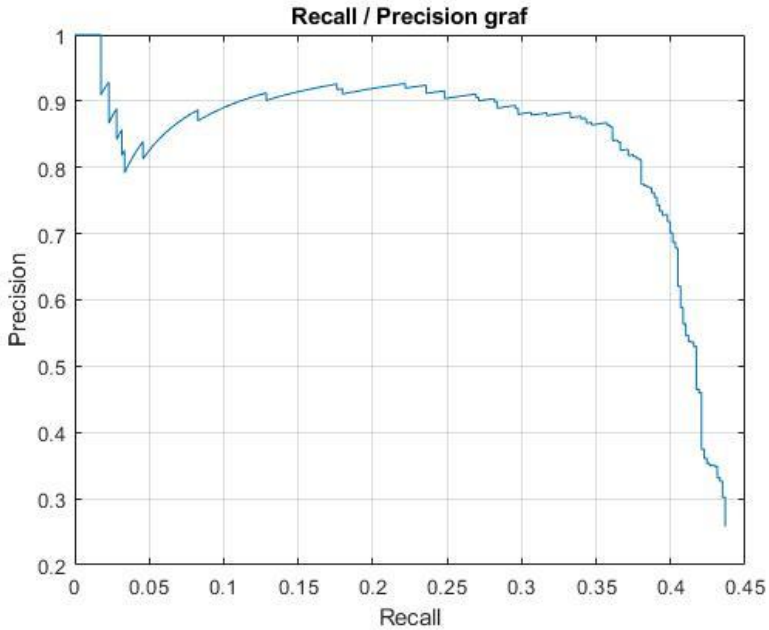
U výsledků detektorů 4, 5, 6, které se liší v počtu trénovacích epoch jsem očekával, že s vyšším počtem trénovacích epoch se bude detektor zpřesňovat, bohužel výsledky to nedokazují, i když při vizuální kontrole bounding boxů v obraze bylo vidět, že je zde patrná vyšší přesnost umístění. Tyto výsledky jsou tedy nejspíš ovlivněny větší toleranci při posuzování, kdy je semafor ještě detekován správně a kdy už nikoliv.

Nízké hodnoty precision, které vypovídají o tom, že bylo mylně navíc detekováno velké množství semaforů, nejsou kritické, protože za předpokladu, že bude klasifikátor správně fungovat, budou tyto falešné detekce odstraněny. Důležité je to, aby bylo co nejméně false negative, protože tyto chyby už dále nelze napravit.

Na následujících grafech uvádím charakteristiku recall – precision pro detektory 6 a 9, které přestože v samostatném testování nedosáhly nejlepších výsledků, tak jejich úspěšnost v kooperaci s klasifikátorem byla experimentálně ověřena jako nejlepší.



Obr. 5-7: Graf závislosti precision (přesnost) a recall (citlivost) u detektoru 6



Obr. 5-8: Graf závislosti precision (přesnost) a recall (citlivost) u detektoru 9

5.3.2 Klasifikátor

Navrhl jsem celkem devět klasifikátoru s různými trénovacími parametry jako jsou počet epoch trénování, počet snímů v jedné tréninkové dávce a rozlišení původních snímků, ze kterých podle příslušně anotovaných souřadnic byly výřezy pořízeny. Koeficient učení byl pro všechny klasifikátory 0,001. Velikost tohoto koeficientu jsem zvolil relativně malou, protože v případě klasifikátoru se jedná o doučení předtrénované sítě, kde už se nepředpokládají výrazné změny vah. Soupis klasifikátorů spolu s jejich parametry jsou uvedeny v tabulce níže.

Tab. 5-3: Natrénované klasifikátory a jejich nastavení při tréninku

označení	trénovací set -počet snímků	rozlišení původních snímků [px]	počet epoch tréninku	velikost dávky
klasifikator_1	1781	1920 x 1080	5	32
klasifikator_2	1781	1920 x 1080	10	32
klasifikator_3	1781	1920 x 1080	20	32
klasifikator_4	1781	1920 x 1080	40	32
klasifikator_5	1781	400 x 200	2	32
klasifikator_6	1781	400 x 200	4	32
klasifikator_7	1781	400 x 200	30	32
klasifikator_8	1781	400 x 200	30	64
klasifikator_9	1781	400 x 200	30	128

Testování klasifikátoru jsem prováděl na 764 výřezech semaforů a výřezech pozadí ze snímků Boček datasetu. Mezi výřezy pozadí jsou například světla aut, okna domů a další. Snímky, z kterých byly vyříznuty testované oblasti, měly v testu číslo jedna rozlišení 400 x 223 pixelů, výřezy pro test dva byly získány ze snímků v původním plném rozlišení, tedy 1920 x 1080.

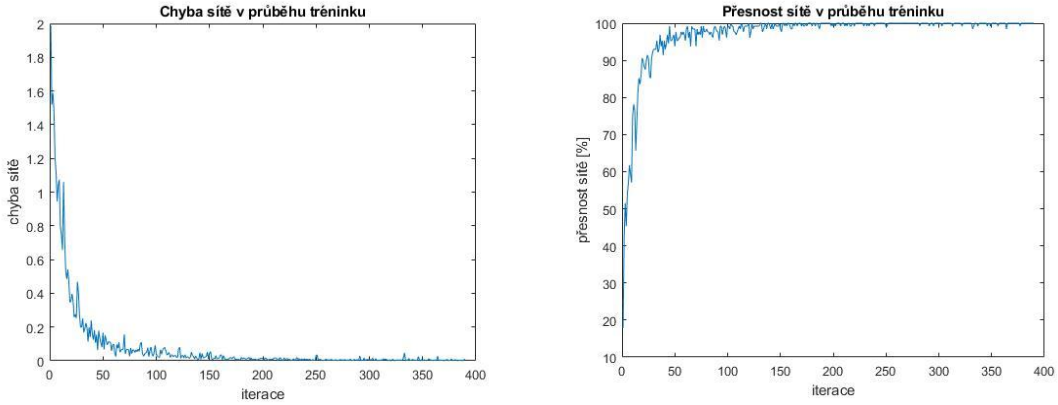
Tab. 5-4: Výsledky testů klasifikátorů, test 1 – výřezy z obrázků o rozlišení 400 x 223 px, test 2 – výřezy z obrázků o rozlišení 1920 x 1080 px

	test 1			test 2		
	TP	FP	Správnost	TP	FP	Správnost
klasifikator_1	527	237	0,690	760	4	0,995
klasifikator_2	590	174	0,772	763	1	0,999
klasifikator_3	487	277	0,637	763	1	0,999
klasifikator_4	457	307	0,598	764	0	1,000
klasifikator_5	726	38	0,950	689	75	0,902
klasifikator_6	752	12	0,984	736	28	0,963
klasifikator_7	753	11	0,986	689	75	0,902
klasifikator_8	758	6	0,992	721	43	0,944
klasifikator_9	754	10	0,987	703	61	0,920

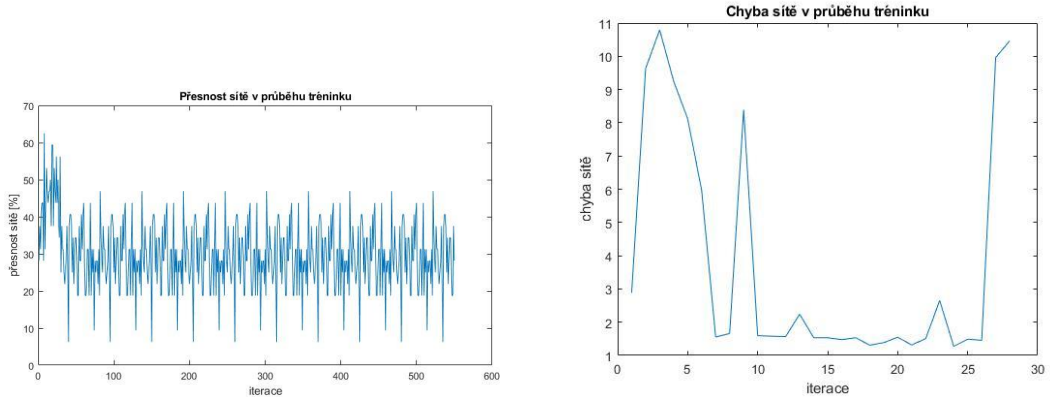
$$\textbf{správnost}\ (\textbf{accuracy}) = \frac{TP+TN}{TP+TN+FP+FN} \quad (8)$$

Z tab.5-4 lze vyčíst, že k natrénování klasifikátoru není potřeba příliš mnoho trénovacích epoch, neboť už po čtyřech epochách učení byl klasifikátor 6 schopen správně klasifikovat 96 % testovacích semaforů.

Dále je patrné, že klasifikátory trénované na větších výřezech, dosahují větší správnosti právě při testování na větších výřezech a klasifikátory trénované na menších výřezech mají i přes menší množství informace v menším výřezu na těchto výřezech větší správnost klasifikace.



Obr. 5-9: Průběh vývoje přesnosti a chyby klasifikační sítě během trénování u klasifikátoru číslo 9



Obr. 5-10: Průběh vývoje přesnosti a chyby klasifikační sítě během učení, kde byl nastaven příliš vysoký koeficient učení (0,01)

5.3.3 Detektor a klasifikátor současně

Celkovou úspěšnost detekce semaforu a určení barvy signalizované barvy jsem provedl na dvou párech detektor – klasifikátor.
Každý tento pár jsem testoval ve třech testech, jejichž vyhodnocení jsem prováděl manuálně. První test byl proveden na 30 snímcích o rozlišení 400 x 223 pixelů z Boček datasetu. Tyto snímky jsou tedy podobného charakteru jako ty, které byly použity k jejich učení. Druhý test obsahoval 45 snímků o rozlišení také 400x 223 pixelů, ale snímky byly vykopírovány z Google map. Třetí test byl proveden na stejných snímcích z Google map, akorát s rozlišením 800 x 447 pixelů.

První testovaný pár detektor - klasifikátor má následujícími parametry:

Tab. 5-5: Parametry detektoru a jeho tréninkové nastavení

Parametry detektoru	
trénovací set	489 snímků z Boček dataset
rozlišení snímků	400 x 233 pixelů
epochy tréninku	50
vstupní vrstva	5 x 5 x 3
velikost dávky	128
koeficienty učení	0,00001 a (pro zpřesnění 0,000001)

Tab. 5-6: Tréninkové nastavení klasifikátoru

Parametry klasifikátoru	
trénovací set	1781 kusů z Boček dataset
rozlišení snímků	400 x 223 pixelů
epochy tréninku	30
velikost dávky	32
koeficienty učení	0,001

Výsledky testu uvádím ve tvaru čtyřpolní tabulky:

Tab. 5-7: Výsledky testování prvního celku detektor - klasifikátor

	True Positive	False Positive	False Negative	chybné určení stavu semaforu	True Negative	Recall	Precision	F1 Score
Boček dataset	45	10	6	2	-	0,882	0,818	0,849
Google - 400 x 223	53	12	30	1	-	0,639	0,815	0,716
Google - 800 x 447	56	24	30	0	-	0,651	0,700	0,675

Vzorec F1 score uveden pod pořadovým číslem (2)

Druhý test byl proveden se stejným klasifikátorem jako v prvním testu, ale rozdílně trénovaným detektorem:

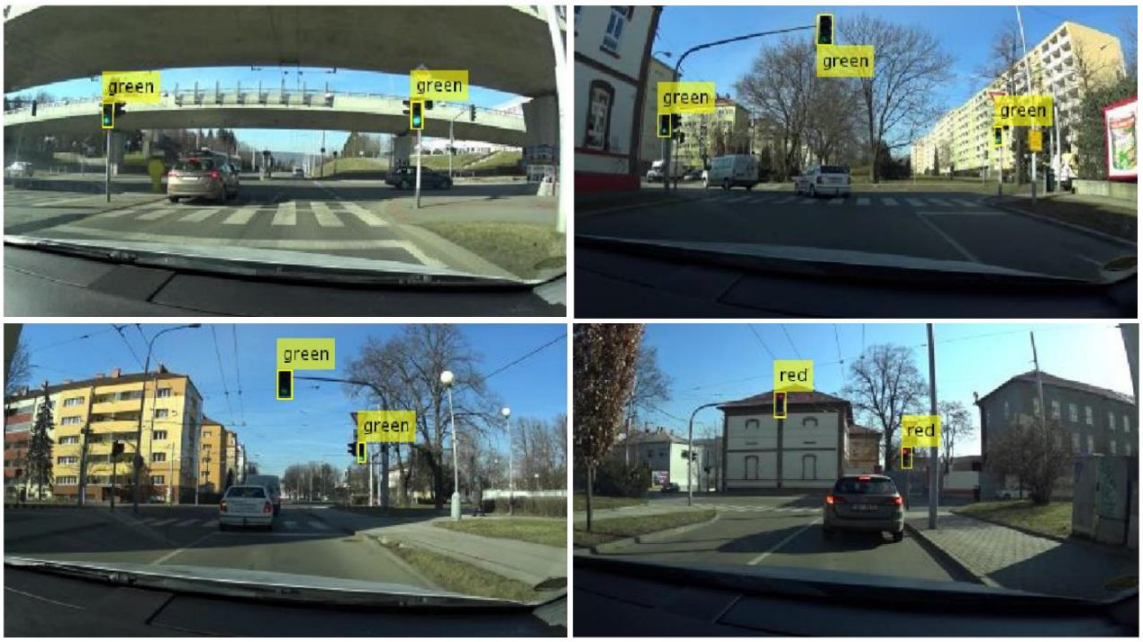
Tab. 5-8: Parametry detektoru a jeho tréninkové nastavení

Parametry detektoru	
trénovací set	839 snímků z Boček dataset
rozlišení snímků	400 x 233 pixelů
epochy tréninku	90
vstupní vrstva	5 x 5 x 3
velikost dávky	128
koeficienty učení	0,00001 a (pro zpřesnění 0,000001)

Tab. 5-9: Výsledky testování druhého celku detektor - klasifikátor

	True Positive	False Positive	False Negative	chybné určení stavu semaforu	True Negative	Recall	Precision	F1 Score
Boček dataset	41	5	12	1	-	0,774	0,891	0,828
Google - 400 x 223	49	15	33	1	-	0,598	0,766	0,671
Google - 800 x 447	52	30	29	1	-	0,642	0,634	0,638

Lepší úspěšnost detekce a klasifikace má na všech trénovacích setech první testovaný pár detektor – klasifikátor, který byl trénován na nižším počtu snímků a také méně trénovacích epoch. Tento výsledek jsem nepředpokládal. Způsobilo to nejspíš mírné přeučení druhého z detektorů, který pak v tomto důsledku nebyl nová testovací data schopen správně vyhodnotit. Z výsledků obou testovaných párů na vzorcích z Google map vyplývá to, že i celý systém má vyšší úspěšnost na snímcích s nižším rozlišením, na němž byly obě části trénovány, než na vyšším rozlišení, byť obsahuje více užitečných informací.



Obr. 5-11: Ukázka správné detekce i určení barvy semaforu

6. ZÁVĚR

V rámci bakalářské práce jsem zpracoval rešerši přístupů a metod k řešení úkolů detekce semaforů a rozpoznání stavu jeho světel. Na základě výsledků tohoto průzkumu jsem se rozhodl k realizaci vlastního systému využít konvoluční neuronové sítě. V samostatné kapitole také přidávám úvod do jejich problematiky a vysvětlení základních pojmů a typů vrstev sítě. Dále jsem provedl analýzu veřejně dostupných anotovaných galerií snímků, které se používají pro učení a testování detektorů semaforů. Pro tento účel jsem také vytvořil vlastní dataset, který obsahuje 1399 anotovaných snímků v rozlišení 1920 x 1080 pixelů.

Vlastní navržený systém jsem řešil pomocí dvou stupňů. Tvoří jej detekční část realizovaná pomocí Faster R-CNN, která lokalizuje ve snímku potenciální semafory a klasifikační část vytvořená modifikací konvoluční neuronové sítě AlexNet, která určí signalizovaný stav semaforu a případně také odstraní „false positive“ detekce. Na snímcích z Boček datasetu jsem natrénoval s různým nastavením parametrů několik variant každé z obou funkčních částí systému. Jejich specifikace uvádím v Tab.5-1 a Tab.5-3. Výsledky jejich testování jsou uvedeny v Tab.5-2 a Tab.5-4. Poté jsem podle dosažených výsledků klasifikátorů a detektorů a s přihlédnutím k experimentálnímu ověření úspěšnosti kooperace obou částí vybral dva vhodné páry a ty jsem testoval jednak na testovacích snímcích z Boček datasetu a také na snímcích z provozu, které byly vykopírovány z Google map. Na snímcích z Boček datasetu u výkonnějšího páru byla dosažena celková úspěšnost systému F1 score 84,9 %, na snímcích z Google map pak 71,6 %. Tyto páry se lišily v tom, že detektor prvního systému byl trénován na 489 snímcích po 50 epoch oproti 839 snímkům a 90 epochám druhého detektoru, přesto byl první detektor úspěšnější. Předpokládám, že tento výsledek je zapříčiněn přeučením druhého detektoru, který nebyl schopen nová data správně vyhodnocovat.

Kvůli nedostatečnému výkonu grafické karty jsem pro trénování musel snížit velikost snímků z původního rozlišení 1920 x 1080 na 400 x 223 pixelů. Kvůli tomuto kroku se v trénovacích datech ztratilo mnoho potřebných informací. Bohužel tímto zmenšením byla síť učena rozpoznávat i vzdálené semafory, které ve skutečnosti na snímcích již nebyly téměř patrné, což vedlo k její mystifikaci a ke snížení celkové úspěšnosti systému.

Do budoucna se nabízí rozšíření portfolia možných klasifikovaných stavů i o signalizované směry na semaforech. Možnosti pokračování vidím také ve zvýšení přesnosti systému pomocí ověřování správné detekce a klasifikovaného stavu semaforu pomocí procházení lokací nalezených semaforů přes více snímků anebo asi největší výzvou by bylo vytvoření systému detekující semafory v reálném čase.

Seznam obrázků

Seznam tabulek

Literatura

[1] BEHRENDT, Karsten, Libor NOVÁK a Rami BOTROS. A Deep Learning Approach to Traffic Lights: Detection, Tracking, and Classification. *IEEE* [online]. 2017, 8 [cit. 2018-03-06]. Dostupné z: http://ieeexplore.ieee.org/document/7989163

[2] BEHRENDT, Karsten a Libor NOVÁK. A Deep Learning Approach to Traffic Lights: Detection, Tracking, and Classification: Robotics and Automation (ICRA), 2017 IEEE International Conference on. *Bosch Small Traffic Lights Dataset* [online]. [cit. 2018-03-04]. Dostupné z: https://hci.iwr.uni-heidelberg.de/node/6132

[3] WEBER, Michael, Peter WOLF a Marius ZOLLNER. *2.2 DeepTLR: A single Deep Convolutional Network for Detection and Classification of Traffic Lights* [online]. IEEE, 2016, 8 [cit. 2018-03-08]. Dostupné z: http://ieeexplore.ieee.org/document/7535408

[4] HALTAKOV, Vladimir, Jakob MAYR, Christian UNGER a Slobodan ILIC. *Semantic segmentation based traffic light detection at day and at night* [online]. Springer International Publishing, 2015, 12 [cit. 2018-03-09]. Dostupné z: https://link.springer.com/content/pdf/10.1007%2F978-3-319-24947-6_37.pdf

[5] OZCELIK, Ziya, Canan TASTIMUR, Mehmet KARAKOSE a Erhan AKIN. A Vision Based Traffic Light Detection and Recognition Approach for Intelligent Vehicles. *IEEE* [online]. 2017, 6 [cit. 2018-04-08]. Dostupné z: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8093430

[6] DE CHARETTE, Raoul a Fawzi NASHASHIBI. *Traffic Lights Recognition (TLR) public benchmarks* [online]. Robotics Centre of Mines ParisTechRobotics Centre of Mines ParisTech, 2010 [cit. 2018-03-04]. Dostupné z: http://www.lara.prd.fr/benchmarks/trafficlightsrecognition

[7] UDACITY, *level5-engineers dataset.* In: Github [online]. [cit. 2018-03-02]. Dostupné z: https://github.com/level5-engineers/systém integration/wiki/Traffic-Lights-Detection-and-Classification

[8] HRADIŠ, Michal. *Konvoluční neuronové sítě* [online]. 7.11.2015, 80 [cit. 2018-05-08]. Dostupné z: https://www.superlectures.com/openalt2015/konvolucni-neuronove-site

[9] ZAPLETAL, Ondřej. *Image Recognition by Convolutional Neural Networks - Basic Concepts*. Brno, 2017, 68 p. Master's Thesis. Brno University of Technology,

Faculty of Electrical Engineering and Communication, Department of Control and Instrumentation. Advised by Ing. Karel Horák, Ph.D.

[10] GIBIANSKY A., *Convolutional Neural Networks* 2014, Dostupné z URL: http://andrew.gibiansky.com/blog/machine-learning/convolutional-neural-networks/

[11] KOLAŘÍK, M.Hluboké učení pro klasifikaci textů. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2017. 50 s. Vedoucí Ing. Lukáš Povoda.

[12] KARPATHY, Andrej.: Convolutional networks for visual recognition. In: Github [online]. [cit. 2016-12-13]. Dostupné z: http://cs231n.github.io/neural-networks-1/

[13] VOLNÁ, Eva: Neuronové sítě 1, Ostravská univerzita, Ostrava 2002

[14] KARPATHY, Andrej.: Convolutional networks for visual recognition. In: Github [online]. [cit. 2016-12-13]. Dostupné z: http://cs231n.github.io/convolutional-networks

[15] YUNG, Jessica. *Explaining Tensorflow Code for a Convolutional Neural Network* [online]. [cit. 2018-05-08]. Dostupné z: https://www.jessicayung.com/explaining-tensorflow-code-for-a-convolutional-neural-network/

[16] Dropout. http://lamda.nju.edu.cn/weixs/project/CNNTricks/imgs/dropout.png, 2015 (přístupné Květen 10, 2018).

[17] How to train a TensorFlow Object Detection Classifier for multiple object detection on Windows, [online] https://github.com/EdjeElectronics/TensorFlow-Object-Detection-API-Tutorial-Train-Multiple-Objects-Windows-10

[18] Compeve Dostupné z: http://www.compeve.com/gpu/nvidia-tesla-c2075-6gb-gddr5-pci-e-x16-gpu-computing-module, (přístupné Květen 12, 2018)

[19] SINHAL, Koustubh, SACHAN Ankit, *Zero to Hero: Guide to Object Detection using Deep Learning: Faster R-CNN, YOLO, SSD*, [online].[cit.16.5.1018], Dostupné z: http://cv-tricks.com/object-detection/faster-r-cnn-yolo-ssd/

Seznam příloh

Příloha 1.: DVD se zdrojovými kódy a databází fotografií