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

POŘÍZENÍ A ZPRACOVÁNÍ SBÍRKY REGISTRAČNÍCH ZNAČEK VOZIDEL

OBTAINING AND PROCESSING OF A SET OF VEHICLE LICENSE PLATES

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **Bc. ANETA KVAPILOVÁ**
AUTHOR

VEDOUCÍ PRÁCE **prof. Ing. ADAM HEROUT, PhD.**
SUPERVISOR

BRNO 2019

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2018/2019

Zadání diplomové práce

Studentka: **Kvapilová Aneta, Bc.**
Program: Informační technologie Obor: Počítačová grafika a multimédia
Název: **Pořízení a zpracování sbírky registračních značek vozidel**
Obtaining and Processing of a Set of Vehicle License Plates
Kategorie: Zpracování obrazu

Zadání:

1. Seznamte se s problematikou strojového učení pro detekci a rozpoznání registračních značek vozidel v obrazech a videu.
2. Vyhledejte, popište a kriticky zhodnoťte existující dostupné datové sady registračních značek.
3. Pořizujte obrázky a videa s automobily a s pomocí dostupných nástrojů je zpracujte - strojově i ručně - pro pořízení datové sady pro učení a vyhodnocování systémů pro detekci a rozpoznávání registračních značek.
4. Vyhodnocujte dostupná řešení pro detekci a rozpoznání registračních značek vozidel na pořízené datové sadě a podle potřeby datovou sadu dále rozšiřujte.
5. Zhodnoťte dosažené výsledky a navrhněte možnosti pokračování projektu; vytvořte plakátek a krátké video pro prezentování projektu.

Literatura:

- 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
- Jakub Špaňhel et al.: Holistic Recognition of Low Quality License Plates by CNN using Track Annotated Data, IWT4S-AVSS 2017
- Jakub Sochor et al.: BrnoCompSpeed: Review of Traffic Camera Calibration and A Comprehensive Dataset for Monocular Speed Measurement, arXiv:1702.06441

Při obhajobě semestrální části projektu je požadováno:

- Body 1 a 2, značné rozpracování bodů 3 a 4.

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

Vedoucí práce: **Herout Adam, prof. Ing., Ph.D.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 22. května 2019
Datum schválení: 7. listopadu 2018

Abstrakt

Diplomová práce se zabývá problematikou pořízení a zpracování datové sady, která obsahuje částečně automaticky zpracované snímky registračních značek vozidel. Cílem je jak pořízení samotných videí, tak i vytvoření nástrojů, které budou schopné videa zpracovávat do výsledné datové sady určené k učení neuronových sítí pro monitorování provozu. K realizaci je využit programovací jazyk Python3 s využitím grafické knihovny OpenCV a frameworku PyTorch pro tvorbu neuronových sítí.

Abstract

This master thesis focuses on creating and processing a dataset, which contains semi-automatically processed images of vehicles licence plates. The main goal is to create videos and a set of tools, which are able to transform input videos into a dataset used for traffic monitoring neural networks. Used programming language is Python3, graphical library OpenCV and framework PyTorch for implementation of neural network.

Klíčová slova

SPZ, registrační značka, vozidlo, neuronová síť, dataset, anotace, poloautomatické zpracování, MTCNN, Python, OpenCV, PyTorch

Keywords

LP, licence plate, vehicle, neural network, dataset, anotation, semi–automatic processing, MTCNN, Python, OpenCV, PyTorch

Citace

KVAPILOVÁ, Aneta. *Pořízení a zpracování sbírky registračních značek vozidel.* Brno, 2019. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce prof. Ing. Adam Herout, PhD.

Pořízení a zpracování sbírky registračních značek vozidel

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracovala samostatně pod vedením prof. Ing. Adama Herouta, PhD. Další informace mi poskytli Ing. Lukáš Teuer a Ing. Jakub Špaňhel. Uvedla jsem všechny literární prameny a publikace, ze kterých jsem čerpala.

.....
Aneta Kvapilová
21. května 2019

Poděkování

Ráda bych poděkovala vedoucímu mojí práce, panu prof. Ing. Adamu Heroutovi, Ph.D za vedení a cenné rady při konzultacích. Rovněž bych chtěla poděkovat panu Ing. Lukáši Teuerovi a panu Ing. Jakub Špaňhelovi za odbornou pomoc. V neposlední řadě děkuji mojí rodině a příteli za podporu.

Obsah

Kapitola 1

Úvod

Strojové učení a zvláště pak neuronové sítě se stávají nejen v posledních letech hojně se rozvíjejícím odvětvím informačních technologií a jejich uplatnění lze nalézt téměř ve všech oborech lidské činnosti. K vytvoření neuronové sítě zaměřené na řešení konkrétního problému (bez ohledu na její složitost) nestačí mít pouze znalosti o teorii neuronových sítí a matematiky. Kromě znalostí je nutné také mít připravena kvalitní data, ze kterých se neuronová síť bude učit. Proces učení je nejdůležitějším krokem při vytváření neuronových sítí a právě volbou trénovacích dat autor sítě rozhoduje o tom, co se síť naučí a jak bude následně schopná reagovat na nová data.

V případě konvolučních neuronových sítí pro zpracování obrazu je nezbytné mít připravenou datovou sadu, která obsahuje nejen obrazová data samotná, ale také nejčastěji textová data, neboli anotace, která představují správnou odezvu sítě na daný vstup. Vytvoření anotací je vzhledem k množství trénovacích dat, které konvoluční neuronová síť potřebuje, mnohdy náročnější (na čas i zdroje) než vytvoření sítě samotné. Nikde není přesně řečeno, jaké množství a jaká data bude daná síť potřebovat, ale obecně platí, že by trénovací datová sada měla co nejvíce korespondovat s daty, která bude síť dostávat posléze při reálném nasazení. Jaké vlastnosti by měly datové sady určené pro učení neuronových sítí splňovat je blíže představeno v kapitole 2, kde je také rozvedena problematika sítí pro zpracování obrazu.

Právě na tvorbu datové sady vhodné pro učení konvolučních sítí je zaměřena tato diplomová práce. Sítě, které budou při učení vzniklou sadu využívat, jsou určené pro monitorování provozu na pozemních komunikacích – konkrétně pak na detekci a rozpoznávání registračních značek vozidel. Ačkoli se jedná o běžné úlohy a potřebná data je možné vytvořit volně prakticky bez problémů, existujících datových sad, které by byly dostupné a daly se přímo použít pro učení sítě, není k dispozici mnoho. Přehled datových sad, které lze nalézt na webu, je uveden v kapitole 4.

Cílem práce bylo vytvoření vlastní datové sady, která obsahuje snímky registračních značek vozidel pořízených za reálného provozu na silnicích. Zpracování prvotních vstupních videí ve výslednou datovou sadu obsahující anotace probíhá poloautomaticky – zpracování je tedy do jisté míry strojové a vyžaduje následnou ruční korekci. Všemi kroky procesu vytvoření takovéto sady včetně použitých technologií se zabývají kapitoly 5 a 6. Kapitola 7 obsahuje podrobné informace o způsobu a rozsahu ruční korekce. Na ni dále navazuje kapitola 8 se zhodnocením dosažených výsledků, popisem prováděných experimentů nad vytvořeným řešením a návrhy na případná další vylepšení.

Kapitola 2

Neuronové sítě pro zpracování obrazu

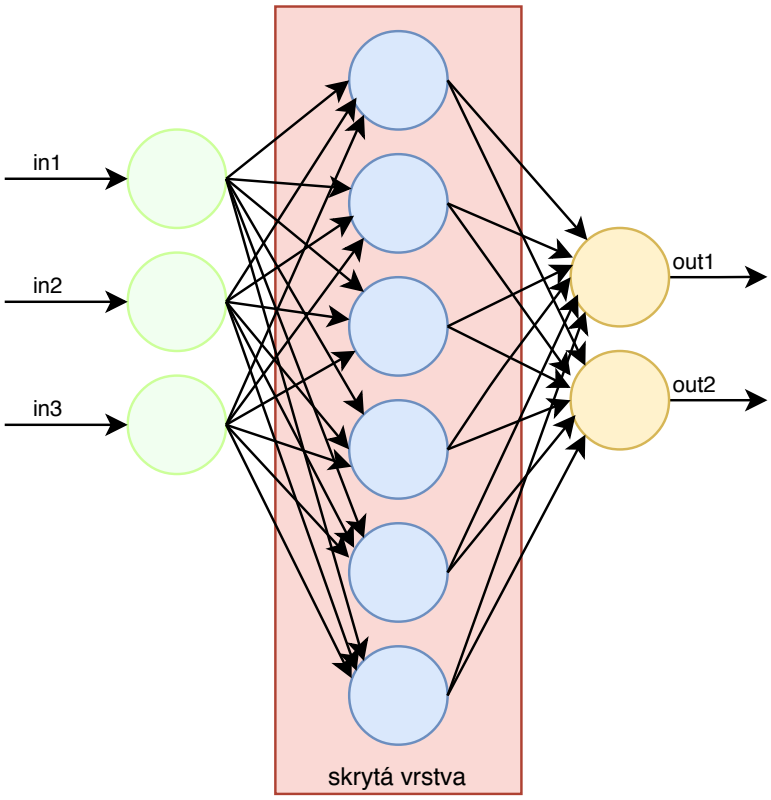
S ohledem na zaměření práce je nutné nejprve uvést problematiku neuronových sítí, motivaci k jejich využívání a také typické problémy spojené s jejich užíváním. Vzhledem k rozsahu tématu budou uvedeny pouze tzv. konvoluční neuronové sítě, zejména varianta *Multi-task Convolutional Network*, která byla použita pro programové řešení této diplomové práce. Další část je potom věnována datovým sadám, které slouží pro trénování, validaci a testování neuronových sítí.

2.1 Neuronové sítě obecně

Neuronové sítě představují výpočetní model pro řešení širokého spektra problémů. Základem modelu je umělý neuron, který podobně jako jeho vzor v lidském mozku reaguje na vstupní signály a při překročení určité meze generuje výstup. Umělým neuronem je rozuměna matematická funkce produkující na základě svých vstupů vypočtený výstup. Volba matematické funkce, kterou neuron představuje, pak ovlivňuje složitost úlohy, kterou je schopen řešit. Počet neuronů v jedné síti se může pohybovat od jednotek až po statisíce v závislosti na typu úlohy.

Neurony se rozdělují do vrstev, kde výpočet mezi neurony jedné vrstvy probíhá paralelně. Vrstva, na kterou přicházejí vstupní data, se nazývá vstupní vrstvou, naproti tomu ta, která produkuje výstup, se nazývá výstupní. Všechny vrstvy mezi nimi jsou vrstvy skryté. V případě, že je skrytých vrstev více, jedná se o hluboké sítě (*Deep Networks*). Všechny neurony jedné vrstvy provádějí stejnou matematickou operaci a na základě druhu této operace jsou pojmenovávány i druhy jednotlivých vrstev (konvoluční vrstva provádí operaci konvoluce atd.).

Podle toho, které neurony jsou mezi sebou propojeny, se rozlišují typy sítí. Dopředné sítě (*Feedforward Networks*) mají vazby od neuronů vstupní vrstvy směrem k neuronům vrstvy výstupní – tedy výstup neuronů jedné vrstvy je připojen na vstup neuronů následující vrstvy. Naproti tomu rekurentní sítě (*Feedback Networks*) mohou obsahovat i zpětné vazby mezi vrstvami. Síť, která neobsahuje vazby mezi neurony stejné vrstvy, je síť acyklická. V opačném případě jde o síť cyklickou. Jsou-li propojeny všechny neurony jedné vrstvy se všemi neurony následující (není to ovšem nutností), pak se jedná o plně propojenou síť. Ukázka plně propojené dopředné sítě je na obrázku 2.1.



Obrázek 2.1: Velmi jednoduchá ukázka dopředné plně propojené sítě. Zelené barvy jsou neurony vstupní vrstvy, v červeném rámečku pak neurony skryté vrstvy a žlutě znázorněny jsou neurony výstupní vrstvy.

Počet vrstev, počet neuronů v jednotlivých vrstvách a způsob propojení mezi neurony udává architekturu sítě. Ta musí být zvolena vhodně pro daný typ úlohy, který má být schopna řešit. Sítě obsahující nejjednodušší neuron, perceptron, jsou schopné řešit pouze klasifikaci lineárně separovatelných vektorů. Oproti tomu hluboká konvoluční síť může být schopná samostatného „myšlení“. Kromě těchto tří zmíněných parametrů je klíčová i vhodná volba tzv. aktivačních a chybových funkcí, případně typů jednotlivých vrstev (plně propojená, konvoluční, shlukovací (*Pooling Layer*) apod.). Tyto jsou podrobněji rozebrány dále. Typů sítí od jednoduchých po opravdu komplexní je nepřeberné množství a jelikož jejich studium není předmětem této práce, je ponecháno na případných zájemcích.

2.1.1 Proces učení

Ze správně navržené architektury ještě síť nedokáže sama produkovat očekávaný výstup. Aby toho byla schopna, musí se nejdříve naučit to, jak reagovat na který vstup. Jak mohlo vyplynout z úvodu, nejsou neuronové sítě nic jiného než matematický model a tak i proces učení je jen výpočet. Konkrétně se jedná o minimalizaci hodnoty tzv. chybové funkce (*Loss Function*). Chybová funkce udává, jak moc byla odezva sítě na daný vstup daleko od očekávané odezvy. Typy chybových funkcí jsou rozebrány níže.

Vazba mezi dvěma neurony sítě je vazbou váhovanou. To znamená, že vstup, který je směrem k neuronu odeslán je vynásoben příslušnou váhou a až potom je zahrnut do dalších

výpočtů. Váhy jsou zpravidla inicializovány náhodně a procesem učení dochází k jejich úpravám.

Princip učení může být mírně odlišný pro každý typ sítě, ale obecný postup je možné najít téměř všude. Po vytvoření sítě a náhodné inicializaci vah mezi neurony jsou na vstup sítě poslána data. Síť na základě jejího aktuálního stavu vypočítá odezvu. Porovnáním požadovaného výstupu s odezvou sítě se spočítá chyba sítě. Ta je následně od výstupních neuronů směrem zpět ke vstupním propagována celou sítí a neurony si na základě velikosti této chyby přepočítávají váhy. Až je vše skončeno, na vstup sítě jsou poslána další data a celý proces se opakuje. Opakování probíhá zpravidla tak dlouho, dokud je odchylka požadovaného výstupu a odezvy sítě dostatečně malá. Tento způsob zpětné úpravy vah na základě chyby sítě se nazývá algoritmus zpětného šíření chyby (*Back Propagation*). Protože byly předem známé požadované výstupy sítě, jedná se o tzv. učení s učitelem (*Supervised Learning*). Opakem je potom učení bez učitele (*Unsupervised Learning*), kde předem není známé nic a síť sama zjišťuje, zda je výsledek správný či špatný. Příkladem sítí, které se učí bez učitele mohou být autoenkodéry, nebo sítě, které řeší problémy shlukování (*Clustering*). Třetím typem učení je učení posilováním (*Reinforcement Learning*). Zde opět nejsou předem známé žádné požadované výstupy, ale síť se snaží určit, jaké by byly přínosy ze všech jejích výstupů a posléze vybere ten, ze kterého bude přínos největší. Problémem při tomto přístupu může být obrovské až nekonečné množství možných výstupů, což může výpočet značně zesložitit.

Je také možné použít libovolnou z již existujících obecných sití a odebrat z ní jen některé vrstvy, ty nahradit novými a novým procesem učení ovlivnit jen tyto nové vrstvy. Zbytek vrstev pak zůstane i při procesu učení nezměněn. Tento přístup značně urychlí celý proces učení hlavně u velkých a složitých sítí, kde by učení zabralo velké množství času i zdrojů.

2.1.2 Chybové funkce

Účelem chybových funkcí [5] je zjistit chybu, kterou síť udělala při odhadu výsledku. Tato chyba je pak nejčastěji pomocí algoritmu zpětného šíření chyby rozšířena celou sítí. Druhů chybových funkcí je mnoho a liší se podle typu problému, který je sítí řešen. Zde budou představeny pouze dvě takové funkce, jedna pro úlohy klasifikační a jedna pro úlohy regresní.

Mezi nejběžnější chybové funkce patří ***Cross Entropy Loss***, která primárně porovnává chybu při klasifikaci mezi dvěma třídami. Lze ji však použít i pro výpočet chyby mezi více třídami – v tom případě je počítána jako součet chyb všech existujících tříd. Matematický vzorec pro výpočet *Cross Entropy Loss* pro diskrétní klasifikaci do n tříd je ukázán pomocí rovnice (2.1) (t značí pravdivý výsledek, p výsledek predikovaný sítí pro vstupní data x).

$$L(t,p) = -\sum_{c=1}^{n} t(x) log(p(x)) \tag{2.1}$$

Z dalších klasifikačních chybových funkcí lze jmenovat např. *Hinge Loss*, logistickou chybovou funkci nebo exponenciální chybovou funkci.

Naproti tomu příkladem chybové funkce vhodné pro řešení regresních úloh je ***Mean Square Error*** (*MSE*). Z rovnice

$$L = \frac{1}{n}\sum_{c=1}^{n}(p(x) - t(x))^2 \tag{2.2}$$

lze vidět, že se jedná o průměr druhé mocniny rozdílu predikované a pravdivé hodnoty. Výpočet výsledku je rychlý, což je pro složité neuronové sítě bezesporu důležitým para-

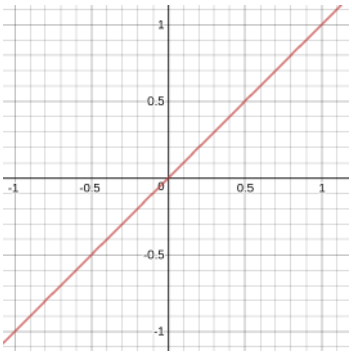
metrem, nicméně je velmi snadno ovlivnitelná odlehlými hodnotami. Dalšími regresními chybovými funkcemi jsou potom *Huber Loss* nebo *Mean Absolute Error*. Obě tyto zmíněné jsou mnohem robustnější a méně citlivé na odlehlé hodnoty.

2.1.3 Aktivační funkce

Aktivační funkce jsou běžnou součástí neuronových sítí jakožto další vrstvy, které se mohou objevit kdekoli v síti. Na základě výstupní hodnoty aktivačních funkcí je rozhodnuto o tom, zda bude daný neuron aktivní, či nikoli. Podle druhu zvolené aktivační funkce může zároveň i docházet k normalizaci dat.

Lineární aktivační funkce [19] je standardní lineární funkce s rozsahem hodnot na obou osách $\langle -\infty, \infty \rangle$. Protože není nijak omezená rozsahem hodnot, které může nabývat, není možné s její pomocí data normalizovat. Graf této funkce je znázorněn na obrázku 2.2, matematický předpis pak ukazuje rovnice

$$f(x) = x\,. \tag{2.3}$$



Obrázek 2.2: Graf lineární aktivační funkce.

Oproti tomu **sigmoida** neboli logistická aktivační funkce už dokáže svůj výstup normalizovat pro libovolné hodnoty osy x na výstupní rozsah hodnot $\langle 0, 1 \rangle$. Vizuálně znázorněný tvar funkce je na obrázku 2.3, matematický předpis představuje rovnice

$$f(x) = \frac{1}{1 + e^{-z}}\,. \tag{2.4}$$

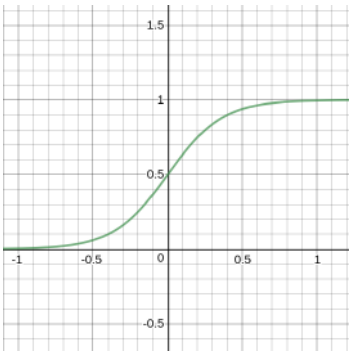
Logistická funkce je používána především pro binární klasifikaci a to právě díky tomu, že může nabývat hodnot, které mohou odpovídat pravděpodobnosti. Její nevýhodou je pomalá konvergence a také to, že síť může při učení uváznout v jednom bodě [19]. Pokud je potřeba klasifikace do více než dvou tříd, pak je nutné použít tzv. ***Softmax***, který je zobecněním logistické aktivační funkce. Vizuálně i vlastnostmi velmi podobnou funkcí logistické je hyperbolický tangens (**TanH**). Jeho rozsah hodnot osy y je $\langle -1, 1 \rangle$. Matematický předpis ukazuje rovnice

$$\frac{e^{2x} - 1}{e^{2x} + 1}\,, \tag{2.5}$$

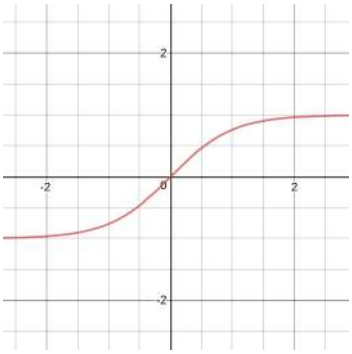
vizuální znázornění pak obrázek 2.4. Funkce je opět používána pro binární klasifikaci.

Poslední z blíže představených aktivačních funkcí je *Rectified Linear Unit* (***ReLU***) [19]. Obor jejích hodnot spadá do intervalu $\langle 0, \infty \rangle$. Jak lze vidět z rovnice

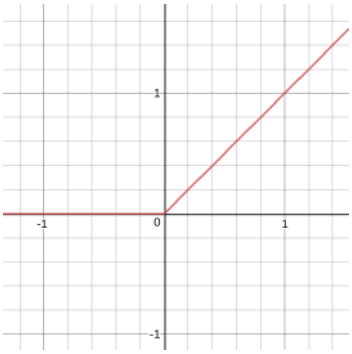
$$y = max(0, x) \tag{2.6}$$



Obrázek 2.3: Graf logistické funkce – sigmoida.



Obrázek 2.4: Graf funkce TanH.



Obrázek 2.5: Graf funkce *ReLU*.

a grafického znázornění (obr. 2.5), funkce nechává beze změny všechny kladné hodnoty a všechny záporné převádí na nulu. To přináší jednu podstatnou nevýhodu – omezuje to schopnost sítě se z těchto (záporných) hodnot učit. Z tohoto důvodu vznikly další upravené verze *ReLU*. Jmenovat lze například *Leaky ReLU* (***LReLU***), *Randomized ReLU* (***RReLU***), nebo *Parametric ReLU* (***PReLU***), které jistým způsobem upravují právě hodnoty menší než nula. *Leaky ReLU* používá místo konstantní nuly parametr $a = 0.01$, kterým vynásobí záporné číslo na vstupu. *RReLU* se od *LReLU* odlišuje pouze hodnotou parametru a. *PReLU* funguje rovněž na stejném principu, jen parametr a není jedno reálné číslo, nýbrž je stejného tvaru jako vstup (tedy nejčastěji tenzor). Díky největší schopnosti reflektovat vstupní data je právě varianta *PReLU* tou nejvíce používanou.

2.2 Konvoluční neuronové sítě

Konvoluční neuronové sítě jsou pouze jedním z mnoha typů neuronových sítí. Jsou obzvláště vhodné pro řešení těch problémů, jejichž vstupní data mají maticovou topologii. Patří sem převážně obrazová a zvuková data, nebo obecně data, která obsahují časové řady [5].

Konvoluční sítě musí alespoň v jedné ze svých vrstev aplikovat operaci konvoluce. Konvoluce je lineární matematická operace, obvykle značená symbolem $*$, která ze dvou vstupních argumentů produkuje jeden výstup. Matematický vzorec pro diskrétní konvoluci v čase t pro výstup o při vstupu i a tzv. konvolučním jádře c je znázorněna rovnicí [5]

$$o(t) = (i * c)(t) = \sum_{a=-\infty}^{\infty} i(a)c(t-a)\,. \tag{2.7}$$

Konvoluce nemusí být pouze funkcí diskrétní, ale po nahrazení sumy integrálem může být i funkcí reálnou. Konvoluční jádro může mít různé velikosti podle účelu, mezi nejběžnější patří 2D jádro 3×3px, případně 5×5px. Není nutné, aby jádro bylo čtvercového tvaru – nic neomezuje použití jádra 3×9px.

Jádro představuje filtr, který je použit zároveň na obě osy (při zpracování 2D obrázků) a na celý vstup najednou. Jádro by mělo být vždy menší, než jsou vstupní data. Právě konvoluční jádra představují hodnoty, které se síť během trénovací fáze učí. Síť se tedy pomocí úpravy jader učí vzory, které se na vstupních datech vyskytují. Ty pak identifikuje v datech nových.

2.2.1 Obvyklé problémy řešené konvolučními sítěmi

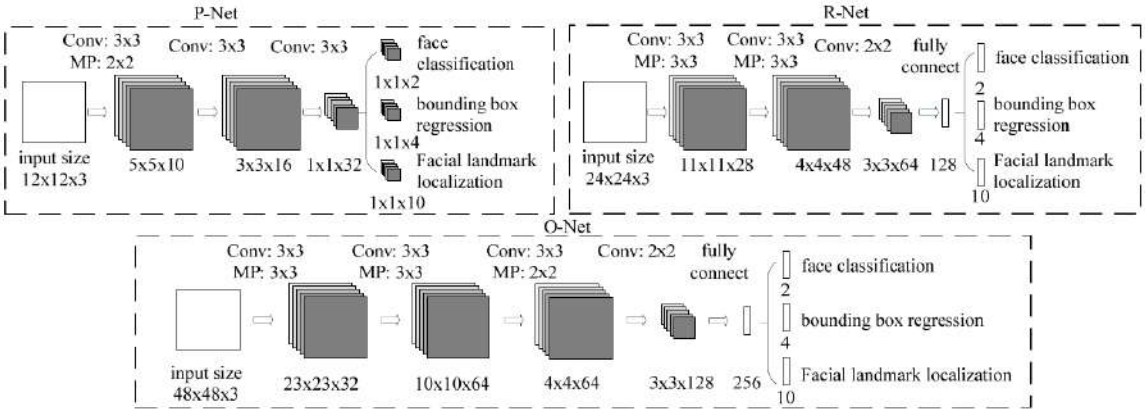
Konvoluční sítě mají rovněž celou škálu využití. Mezi tři nejtypičtější se řadí klasifikace, detekce objektů či vzorů a regrese, přičemž všechny spolu úzce souvisí.

Klasifikací se rozumí zařazení určitého objektu na základě jeho vlastností do jedné z předem definovaných kategorií. **Detekce** a klasifikace jsou úzce propojenými problémy (než je možné cokoli zařadit do příslušné třídy, je nutné zařazovaný objekt detekovat). V mnoha případech je požadován výstup ve formě tzv. bounding boxu (ohraničující box). Jedná se o pomyslný box, do kterého by se dal detekovaný objekt uzavřít. Příklad bounding boxu kolem lidského obličeje je na obrázku 2.7. Mnohdy se stává, že je detekováno velké množství bounding boxů kolem stejného objektu. K eliminaci nadbytečných boxů se používá algoritmus potlačení nemaxim (*Non Maxima Suppression – NMS*). Princip metody je založen na eliminaci hodnot, které nepředstavují maximum ve svém okolí. Problém **regrese** (predikce neznámé veličiny) může být rovněž řešen při detekci. Regresním problémem je například odhad souřadnic bounding boxu.

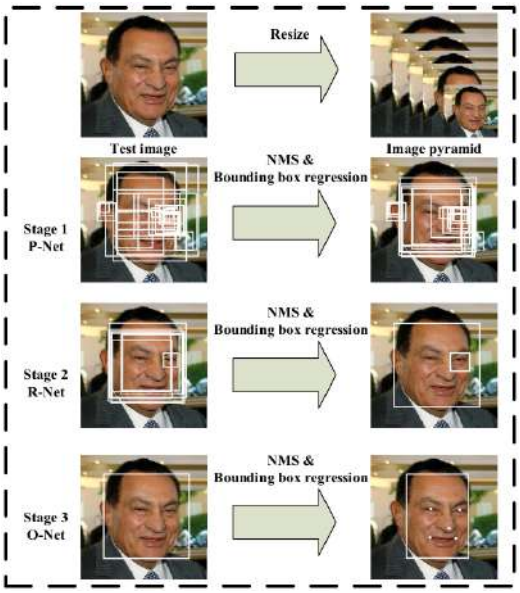
2.3 Multi-task Convolutional Neural Network

Jednou ze známých konvolučních neuronových sítí je *Multi-task Convolutional Neural Network* (*MTCNN*), na jejímž principu je založené i programové řešení této diplomové práce.

Síť byla původně navržena za účelem vyhledávání bounding boxů lidských obličejů a jejich pěti význačných znaků – špičky nosu, očí a koutků úst. Jedná se o kaskádovitě poskládané tři sítě – tzv. *Proposal Network* (*P-Net*), *Refine Network* (*R-Net*) a *Output Network* (*O-Net*) (viz. obrázek 2.6), kde výstup jedné sítě jde na vstup další z nich.



Obrázek 2.6: Architektura sítě *MTCNN*. Po každé konvoluční a plně propojené vrstvě je použita nelineární aktivační funkce *PReLU*. Zdroj [21].



Obrázek 2.7: Ukázka výstupů jednotlivých sítí *MTCNN* a jejích fází. Zdroj [21].

Originální vstupní obrázek je nejdříve přeškálován na různé velikosti. Výsledkem je pyramida obrázků, která jde na vstup sítě *P-Net*. Jedná se o plně konvoluční síť, jejímž výstupem jsou možní kandidáti na bounding boxy a jejich příslušné regresní vektory. Protože kandidátů na bounding boxy je obvykle velké množství a vzájemně se překrývají, jsou vybrány jen ty, jejichž překryv je největší. K této eliminaci je použit algoritmus *Non-Maximum Sup-*

pression (*NMS*) a regrese bounding boxů. Už eliminovaní kandidáti z *P-Net* jsou předáni na vstup druhé síti, *R-Net*. Ta provádí další eliminaci kandidátů, kalibraci za pomocí regrese bounding boxů a rovněž *NMS*. Třetí síť, *O-Net*, pak provádí finální eliminaci a přidává pozice zmíněných pěti význačných bodů. Celkové schéma zpracovávání vstupu a výsledky po zpracování jednotlivými sítěmi je graficky ukázáno na obrázku 2.7.

Sítě se musí během své učící fáze naučit řešit tři problémy – binární klasifikaci toho, zda je na snímku obličej, regresi bounding boxů a lokalizaci význačných bodů. Jako ztrátové funkce jsou použity *Cross Entropy Loss* pro binární klasifikaci a euklidovská vzdálenost pro zbylé problémy [21].

2.4 Datové sady

Jak již bylo zmíněno v úvodu této práce, mít kvalitní datovou sadu je pro učení neuronové sítě velmi důležité. Tato kapitola je proto věnována tomu, jaké vlastnosti musí taková datová sada splňovat, aby jí učená neuronová síť měla co nejlepší výsledky v reálném využití. Rovněž je zmíněn i proces získání a zpracování vstupních dat ve vhodnou datovou sadu.

2.4.1 Důležité vlastnosti

Pro všechny datové sady platí, že data, která budou sloužit jako trénovací, by měla být co nejvíce podobná datům, která bude síť dostávat na vstup při reálném nasazení. Není to však jediné kritérium, které by měla sada splňovat.

Datová sada by měla být dosti **variabilní** – tzn. měla by obsahovat všechny možné případy, které se má naučit. Pokud se jedná o klasifikační problém, potom by v datech měly být zastoupeny všechny třídy, které se má síť naučit rozpoznávat. Poměr zastoupení jednotlivých tříd by měl přibližně odpovídat poměrnému zastoupení v reálných datech.

Další klíčovou vlastností je **velikost**. Je téměř nemožné určit přesný počet vzorků, které musí datová sada obsahovat. Dokonce ani nemusí platit, že pokud je k dispozici obrovská datová sada obsahující miliony vzorků, je to klíčem k úspěchu. Obecně by se ale dalo říci, že (alespoň v případě konvolučních neuronových sítí) je vhodné mít v datové sadě alespoň řádově desetitisíce vzorků.

Pokud je množství vzorků malé, může se síť na první pohled naučit stejně dobře jako na velkém množství. Dá se ovšem očekávat, že síť pak bude hůře reagovat na nová data, případně může být nestabilní, nebo vykazovat občasné výkyvy v chování. To je dané vysokou citlivostí na inicializační parametry a pořadí vzorků při trénování [18]. Do jisté míry by mohl v případě menších množství dat pomoct princip křížové validace (*Cross Validation*) – data se rozdělí do více menších podmnožin, jedna z nich se vybere a prohlásí za testovací, ostatní jsou trénovací. Celý proces je znovu opakován s jinou testovací množinou, dokud není dosaženo požadované přesnosti.

Rovněž je nezbytné mít korektní **anotace**. Jakákoli chyba, která se vyskytne v anotacích, bude po celou dobu učení propagována sítí. Nedá se očekávat, že budou všechny anotace vždy stoprocentně správné, ale chyby v nich by měly být minimalizovány na nejnižší možnou úroveň.

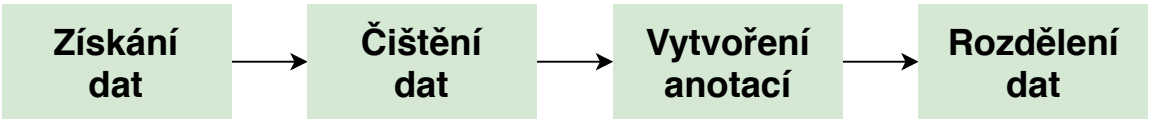
2.4.2 Proces vytvoření datové sady

Samotný proces vytvoření datové sady od vstupních dat až po kompletní datovou sadu vhodnou pro neuronové sítě je znázorněn na obrázku 2.8. Prvním krokem je získání vstup-

ního videa, obrázků nebo třeba hlasové nahrávky, podle toho, k čemu má výsledná síť sloužit. Jak již bylo řečeno – je nutné, aby v datech byly zastoupeny všechny případy, které má síť umět zpracovávat. Pokud budou data chybět již po této fázi, nemusí být jejich pozdější doplnění možné.

Druhou částí je pak čištění dat. Sem patří nahrazení chybějících hodnot, nalezení a kontrola hodnot, které se výrazně vychylují od hodnot standardních apod. V případech, kdy je to žádoucí, sem může patřit i normalizace.

Pravděpodobně nejproblematičtější částí pak může být anotace získaných dat. Anotace mohou mít v podstatě libovolný formát. Nejběžněji je možné se setkat s textovým souborem (či soubory) ve formátech *JSON*, *pickle*, *YAML* nebo *CSV*. Vždy je pouze na autorovi, jaký formát pro svá data zvolí. Nejproblematičtější část je to z toho důvodu, že je velmi těžké proces plně automatizovat. Mezi používané postupy patří částečná automatizace s následnou ruční korekcí. Nakonec je vždy potřebné zpracovaná data rozdělit na část trénovací, testovací a validační podle účelu, ke kterému budou následně použita. Data by měla být zastoupena ve všech třech částech rovnoměrně.



Obrázek 2.8: Fáze vytváření datové sady pro učení, testování a validaci neuronové sítě.

Kapitola 3

Existující algoritmy pro detekci a rozpoznání registračních značek

V této kapitole jsou představeny některé existující přístupy k detekci a rozpoznávání registračních značek. Tyto metody by se daly rozdělit na dvě skupiny. První a chronologicky starší z nich jsou metody, které ke zpracování používají algoritmy pro segmentaci obrazu – tedy rozdělení obrazu na části, které jsou si sémanticky podobné. Druhou skupinou jsou potom metody založené na neuronových sítích.

Zástupců obou skupin je mnoho [8, 10, 9, 20] a není předmětem této práce je všechny představit. Níže je tedy od každé z nich vybrán pouze jeden algoritmus jako ukázka rozdílnosti přístupů k problému.

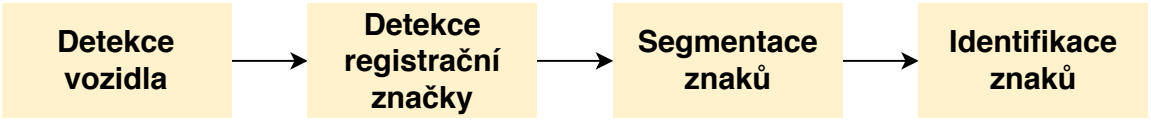
3.1 Metody založené na segmentaci obrazu

Ačkoli nejsou všechny metody, které nevyužívají neuronových sítí, stejné, lze v nich najít přinejmenším podobnou posloupnost jednotlivých úkolů, které je potřeba zvládnout pro úspěšnou detekci a rozpoznání registrační značky. Jednotlivé kroky jsou znázorněny na obrázku 3.1. Z přístupů, které se v jednotlivých částech využívají, lze jmenovat např.:

- binarizace obrazu
- *CCA – Connected Component Analysis*
- Houghovy transformace
- detekce hran
- fuzzy logika
- normalizace

Pro bližší představení toho, jak segmentační metody pracují, bude v podkapitole 3.1.1 blíže rozebrána jedna z metod, kterou představili Chen a Luo [2], využívající detekci hran.

Značná část segmentačních metod včetně vzájemného srovnání z hlediska výkonnosti i náročnosti byla rozebrána Patelem a spol. [16]. Níže je představen přístup využívající detekci hran jako ukázka toho, jak mohou tyto metody pracovat.



Obrázek 3.1: Obvyklé schéma pro detekci a rozpoznání registračních značek vozidel. Zdroj: inspirováno článkem [16].

3.1.1 Přístup využívající detekci hran

Tato metoda byla poprvé představena v roce 2012. Vzhledem k rychlosti vývoje informačních technologií byla ukázána velice dávno a také je to jedna z posledních metod, které nevyužívají neuronové sítě.

První částí celého procesu je předzpracování zachycených snímků o velikosti 640×480 pixelů a barevné hloubce *True Color*. Ty jsou převedeny do odstínů šedi a následně je za účelem zvýšení kontrastu dynamicky roztažen histogram. Posléze jsou zvýrazněny hrany pomocí rozšířeného Prewittova operátoru. To znamená, že je provedena konvoluce vstupního obrazu s osmi jádry (viz tabulku 3.1). Výsledkem je, že budou zvýrazněny všechny hrany, nejen horizontální a vertikální, jak je tomu u standardního Prewittova operátoru. Výsledné hodnoty jednotlivých konvolucí se mezi sebou vynásobí a výsledek je porovnán s prahem. Pokud je hodnota větší než práh, pixel je hranou. Následně je odhadnuta pozice registrační značky na základě součtu hodnot pixelů nejdříve v horizontálním a následně ve vertikálním směru. Jako oblasti obsahující značku jsou nakonec označeny pouze ty v předchozím kroku označené části, které mají stejný poměr stran, jako má registrační značka daného státu [2].

-1	-1	-1	-1	-1	0	-1	0	1	0	1	1
0	0	0	-1	0	1	-1	0	1	-1	0	1
1	1	1	0	1	1	-1	0	1	-1	-1	0
1	1	1	1	1	0	1	0	-1	0	-1	-1
0	0	0	1	0	-1	1	0	-1	1	0	-1
-1	-1	-1	0	-1	-1	1	0	-1	1	1	0

Tabulka 3.1: Použitá konvoluční jádra pro rozšířený Prewittův operátor. Zdroj: vytvořeno na základě informací v původním článku [2].

3.2 Metody založené na neuronových sítích

Princip konvolučních neuronových sítí byl vysvětlen v kapitole 2. Stejně jako v případě segmentačních metod, i zde je možné za posledních pár let najít nepřeberné množství variací neuronových sítí detekujících a rozpoznávajících registrační značky vozidel. Mnohé z nich do značné míry vycházejí z obecných sítí pro detekci objektů, jako jsou varianty *Regions with CNN features* (*R-CNN* [3]), *Single Shot MultiBox Detector* (*SSD* [12]) nebo *You Only Look Once* (*YOLO* [17]).

I nyní tedy bude představena jedna metoda jako ukázka odlišného přístupu k řešení daného problému.

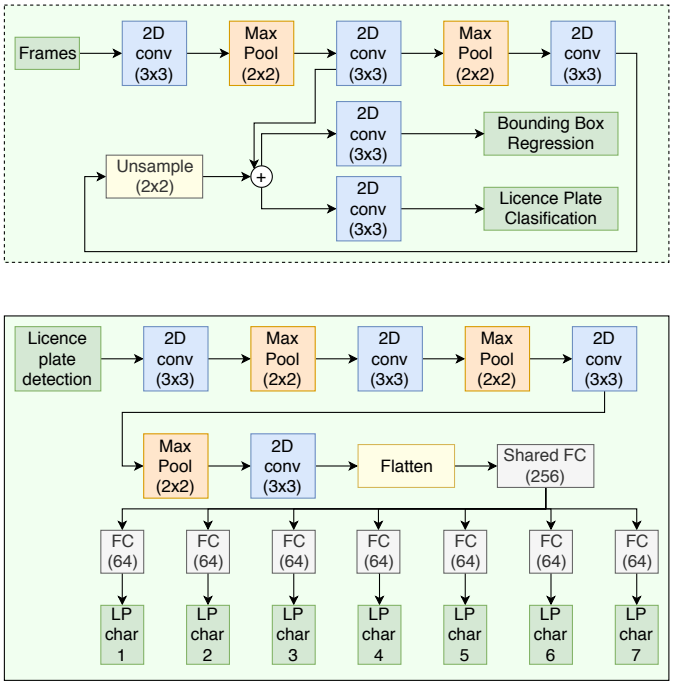
3.2.1 Přístup využívající hluboké neuronové sítě

Tato metoda, kterou představili Gonçalves a spol. [4], byla uvedena v roce 2018 a patří tedy k nejnovějším. Základem jsou dvě neuronové sítě kaskádovitě poskládané za sebe, kde první síť má za úkol rozpoznat registrační značku v obraze, zatímco druhá na jejích výstupech rozpoznává znaky. Architektura sítí je ukázána na obrázku 3.2. Oproti segmentačním metodám, které obsahovaly čtyři kroky pro rozpoznání registrační značky, tato obsahuje pouze dva kroky, kde každý z nich je reprezentován jednou sítí.

Úkolem první sítě je určit bounding box registrační značky. Při učení je tento úkol chápán jako binární klasifikace toho, zda má daný výřez obrázku dostatečně velké překrytí s pravdivým (*Ground Truth*, anotovaným) boxem. Pro správné fungování další sítě musí být zajištěno, že bounding box spolehlivě obsáhne všechny znaky značky a nejen její části. Aby byla tato podmínka splněna, byla navržena speciální chybová funkce, která zvýhodňuje ty bounding boxy, kde souřadnice jejich levého horního rohu jsou menší a pravého dolního rohu větší, než u *Ground Truth* bounding boxů.

Úkolem druhé sítě je pak rozpoznávání jednotlivých znaků registrační značky, která je výstupem první sítě. Toto rozpoznání je rozděleno jako sedm jednotlivých úkolů (i když mají mnoho podobností) – jeden úkol na rozpoznání jednoho znaku, přičemž je všech sedm úkolů prováděno zároveň. Aby byla zajištěna dostatečná obecnost sítě, byla vstupní trénovací data uměle upravována a na těchto datech byly pak znovu dotrénovávány pouze plně propojené vrstvy [4].

Datová sada, která byla použita pro učení těchto dvou sítí, je podrobně popsána v kapitole 4.4.



Obrázek 3.2: Architektura dvou navržených sítí. Horní síť slouží k detekci značky, dolní potom k jejímu rozpoznávání.

3.3 Existující kompletní řešení

Při bližším zkoumání detekce a rozpoznání registračních značek lze využít nejen algoritmy, ale i na celý funkční software, případně i hardware. V této části je představeno několik zástupců jak otevřeného, tak i komerčního software. Vzhledem k tomu, že se ve většině případů jedná o komerční software, dohledat alespoň stručné informace o jeho vnitřním fungování není možné. Z tohoto důvodu nejsou některé informace u některých zástupců uvedeny. Kapitola rovněž neslouží jako kompletní seznam všech existujících řešení, jsou zde uvedeni jen někteří zástupci jako ukázka.

OpenALPR[1] je jeden z prvních, které lze najít. Jedná se o otevřenou knihovnu napsanou v programovacím jazyce C++ s možností využití i v dalších jazycích jako jsou C#, Java, Python či Node.js. Knihovna je schopná analyzovat vstupní videa a rozpoznávat v nich registrační značky. Kromě knihovny je k dispozici i komerční rozhraní, které dokáže za stejným účelem analyzovat jak obrázky, tak i streamované video. Rovněž je poskytována i cloudová služba, pomocí které je možné analyzovat videa z IP kamer. Celé řešení je založené na neuronových sítích. Online je dostupná rovněž kvalitní dokumentace k celému projektu.

Řešení založené na hlubokých neuronových sítích, poskytuje ***Plate recognizer***[2]. Podle informací na oficiálním webu byla síť trénována na registračních značkách z více než sto zemí. Vše je dostupné formou API, kde je 2500 rozpoznání zdarma a další jsou placené měsíčně. API je poskytováno standardně přes *HTTP*, a je tudíž možné jej využít v široké škále programovacích jazyků.

Maďarská firma ***Adaptive Recognition Hungary (ARH Inc.)***[3] oproti předchozím zmíněným nabízí nejenom software, ale i hardware, a to nejen kamery a rychlostní radary, ale například i terminály pro řízení přístupu. Dodaná řešení mohou pak být využívána pro automatickou kontrolu parkování či pro řízení přístupu ať už do soukromých oblastí, nebo na státních hranicích, měření rychlostí apod.

Společnost ***Axis communications***[4] se zaměřuje nejen na monitoring provozu, ale na audio a video systémy obecně. Rovněž nabízí komplexní řešení pro automatickou detekci a rozpoznání registrační značky v reálném čase – tedy kamery i příslušný software. Řešení může podle dostupných informací fungovat jak pouze na kameře samotné, tak využívat i vzdálený server.

Americká společnost ***Riverland Technologies***[5] rovněž patří k těm, které nabízí komplexní řešení včetně analogových, inteligentních nebo IP kamer, příslušného software, či dokonce solárního přívěsu. Používaný software se skládá z vyhledávače značek a části, která dokáže nalezenou značku rozpoznat. Podle informací uvedených na webu byly zkoušeny metody založené na neuronových sítích i rozpoznávání vzorů, ale neměly dostatečné výsledky. Bližší informace dostupné nejsou.

Z dalších firem, které nabízí podobný sortiment hardware i proprietárního software po celém světě lze jmenovat ***Group Mobile***[6], ***Vision Components***[7], ***INEX Technologies***[8], ***CINTEL***[9] a další.

[1] https://www.openalpr.com/
[2] https://platerecognizer.com/
[3] https://www.arh.hu/
[4] https://www.axis.com/
[5] http://www.riverland-tech.com/
[6] https://groupmobile.com/
[7] https://www.vision-components.com/
[8] http://www.inexzamir.com/
[9] http://crm.cintelusa.com/

Kapitola 4

Analýza existujících datových sad

V této kapitole budou postupně ukázány datové sady, které je možné najít na webu dostupné volně, příp. je možné je získat při požádání autorů osobně. Na konci kapitoly v části Shrnutí je porovnání takovýchto datových sad a zhodnocení jejich výhod i nevýhod.

V rámci srovnání je možné hodnotit vlastnosti jak datové sady jako celku, tak i jednotlivých snímků. Mezi vlastnosti celku patří např. množství snímků, variabilita, obsah anotací, obsah snímků (jen značky, celá auta, více aut na snímek), reálnost apod. Vlastnostmi jednotlivých snímků mohou být kvalita, světelné podmínky, natočení nebo zpracování.

4.1 UFPR-ALPR Dataset

Datová sada známá jako *UFPR-ALPR* byl pořízen Laboratoří pro vidění, robotiku a zobrazování (*Laboratório Visão Robótica e Imagem*) spadající pod Federální univerzitu v Paraná (*Universidade Federal do Paraná*). Veškeré informace o datové sadě pochází z článku Larocy a spol. [11].

Datová sada byla pořízena právě ve státě Paraná a obsahuje snímky výhradně brazilských registračních značek, které mají svá specifika. Značky aut jsou o rozměrech 40×13cm, značky motocyklů potom 20×17cm. Soukromá vozidla mají značky šedé barvy, zatímco taxi a autobusy mají pozadí červené. Výjimečně se mohou vyskytovat i jiné barvy či typy.

Samotná datová sada obsahuje 4 500 plně anotovaných snímků s více než třiceti tisíci znaky. Na snímcích je zachyceno celkem 150 vozidel v reálném provozu – tedy za pohybu a kamerou v autě, kde na každém snímku je vždy viditelná pouze jediná značka. Snímky pak byly získány z celkem 150 videí o délce 1s se snímkovací frekvencí 30 *FPS* a jsou dostupné ve formátu *PNG* s velikostí 1920×1080 pixelů. Ukázky snímků jsou znázorněny na obrázku 4.1. Záznamy byly zachycovány na celkem tři zařízení – GoPro Hero4 Silver, Huawei P9 Lite a iPhone 7 Plus. Z každého zařízení bylo získáno 1500 snímků, které jsou rozděleny následovně:

- 900 snímků s šedou registrační značkou
- 300 snímků s červenou registrační značkou
- 300 snímků registrační značky motocyklu

Datová sada je rozdělena na 40% dat pro trénování, 40% pro testování a 20% pro validaci. Pro každý ze snímků je dostupná anotace v textovém souboru, který obsahuje následující informace:

- informace o záznamovém zařízení, kterým byl snímek pořízen
- pozice vozidla
- informace o vozidle (např. typ vozidla, výrobce, model, rok výroby)
- pozice registrační značky jako celku
- pozice jednotlivých znaků registrační značky

Celá datová sada je volně ke stažení pro akademické účely a nekomerční použití. Pro přístup k sadě je ovšem nutno požádat autora osobně elektronicky z platného univerzitního emailu.

Obrázek 4.1: Ukázky snímků z datové sady známé jako *UFPR-ALPR Dataset*. Na ukázkách jsou vidět jak odlišnosti ve světelných podmínkách, tak i ve vzdálenostech vozidel. V poslední řadě jsou pak ukázky anotovaných snímků [11].

Nevýhodou této sady je absence fotografií vozidel zepředu, což může být problém např. pro sítě, jejichž úkolem je monitorování příjezdu vozidel. Další nevýhodou může být to, že snímky byly získávány ve dne – nejsou zde žádné, které by byly natáčené za šera nebo tmy.

4.2 Dataturks – Indian Number plates

Web Dataturks[1] nabízí uživatelům po přihlášení možnosti využití jejich nástrojů či přímo API pro anotaci vlastních snímků a videí. Využití Dataturks ovšem zavazuje k tomu, že se zde anotovaná data stanou veřejně dostupná a uživatel nemá možnost data smazat. Z tohoto důvodu je zakázaný upload dat s citlivými údaji nebo dat, která porušují autorská práva. Získaná data je pak možné exportovat ve formátu Pascal VOC, nebo přímo jako data pro framework TensorFlow, Keras nebo spaCy.

[1] https://dataturks.com

Jednou z takových otevřených datových sad je i sada registračních značek indických vozidel uživatele Devika Mishra [14]. Po stažení souboru ve formátu JSON z tohoto projektu jsou dostupné tyto údaje:

- URL, na které lze snímek nalézt
- souřadnice bounding boxu
- rozměry snímku
- informace o autorovi
- záznam o posledním datu úpravy

Sada obsahuje 511 záznamů, přičemž nejsou všechny hodnoty vždy vyplněné (některé položky obsahují pouze *URL* snímku, ale žádné další informace). Ne všechny snímky pocházejí přímo od uživatele – značné množství představují snímky stažené z webu obsahující copyright či přímo autora, nebo se jedná o snímky celých aut stažené pravděpodobně přímo ze stránek výrobců. Velikost a kvalita snímků je od 200×140 pixelů po Full HD snímky.

Variabilita sady je opravdu značná – vozidla (výhradně auta a motocykly) jsou focena z blízka i z dáli pod odlišnými úhly, jednotlivé značky jsou různých tvarů, formátů a barev. Auta byla focena zepředu i zezadu. Počasí i světelné podmínky se taktéž liší – chybí sice snímky za deště, ale je možné najít snímky nasvícených značek za tmy, šera nebo pouze v přítmí. Část snímků obsahuje registrační značky popisující typ vozidla (viz ukázka na obrázku 4.2), část je umělá. Převážnou většinu ovšem tvoří reálné fotografie (ovšem nikoli z kamerového záznamu).

Obrázek 4.2: Ukázky snímků z databáze registračních značek indických vozidel volně dostupné na webu Dataturks [14].

4.3 MLBLR Datasets

Cílem komunity *MLBLR*[2] je vytvořit datovou sadu, která bude obsahovat sto tisíc uměle vytvořených a stejný počet reálných snímků registračních značek. Všechny snímky by měly obsahovat výhradně indické registrační značky. Po vytvoření, pročištění a anotaci dat by měla být data volně přístupná.

Simulovaná sada by měla obsahovat:

- různé úhly pohledu pro stejnou značku
- čisté snímky bez šumu na pozadí
- snímky bez jakéhokoli pozadí
- snímky značek ze všech indických států

Proces generování sestává z několika částí. První z nich je použití knihovny *OpenCV* pro vygenerování více než tisíce snímků pro jeden konkrétní stát, přičemž jsou použity dva a více nejběžněji používané fonty. Jakmile jsou takové snímky vytvořeny, jsou importovány do software pro 3D animace *Blender*, který má za úkol snímky vyrenderovat na obdélníkový podklad. Jakmile je tato část hotová, je možné značkami otáčet a získávat různé úhly a natočení. Posléze je přidáno rozmazání pohybem. Takto je vytvořeno dvacet vzorků z jednoho obrázku ve vysoké kvalitě.

Sada obsahující reálné snímky by měla obsahovat fotografie registračních značek více než čtyř set vozidel (auta, motocykly, skútry, nákladní auta) pod různými úhly a světelnými podmínkami. Cílem projektu je vytvořit největší datovou sadu indických registračních značek na světě [15]. Žádná z výše zmiňovaných sad ovšem není veřejně dostupná (ověřeno v prosinci 2018).

4.4 SSIG-ALPR Database

Datová sada *SSIG-ALPR Database* byla vytvořena jako podklady pro článek *Real-time Automatic License Plate Recognition Through Deep Multi-Task Networks* [4] a pro ostatní vědce při řešení úkolů zahrnujících rozpoznávání registračních značek.

Snímky byly získávány ve dne ve městě za pomocí dvou kamer – jedné stacionární a jedné umístěné uvnitř pohybujícího se vozidla. Datová sada obsahuje snímky ve Full HD rozlišení o velikosti 1920×1080 pixelů. Každý snímek má průměrně 2,4MB a velikost kompletní databáze je 35GB. Sada celkem obsahuje 3 595 trénovacích, 2 360 testovacích a 705 validačních snímků. Dohromady se jedná o 6 660 snímků, které obsahují 8 683 registračních značek 815 vozidel. Velikosti registračních značek se pohybují od 5×12 do 86×196 pixelů s poměrem stran 1:0.38 [4].

Datová sada je dostupná pouze pro akademické účely. Při žádosti o přístup je nutné podepsat licenční smlouvu podepsanou příslušnými autoritami dané organizace, které mají oprávnění ji zastupovat v právních záležitostech (nikoli tedy studenty nebo pouze členy fakulty) [4].

[2] https://mlblr.com

Obrázek 4.3: Ukázka fotografie z datové sady známé jako *SSIG-ALPR Database* [4].

4.5 Snímky bez anotací

V mnohem větší míře volně dostupné jsou snímky, kde jsou registrační značky vozidel vyfocené, ale neobsahují žádná další data ani anotace. Takové snímky mohou po následné ruční anotaci sloužit například pro testování toho, jak schopná je síť rozpoznat i atypické značky odlišných barev, tvarů i velikostí. Webů, které se zaměřují na registrační značky vozidel je nespočet. V této části je zmíněno několik z nich jako příklad.

4.5.1 License Plate Detection, Recognition and Automated Storage

Projekt *License Plate Detection, Recognition and Automated Storage* [7] Fakulty elektrotechniky a výpočetní techniky Univerzity v Záhřebu (*Fakultet elektrotehnike i računarstva Sveučilište u Zagrebu*) obsahuje databázi pěti set deseti snímků registračních značek převážně chorvatských vozidel.

Snímky registračních značek jsou pouze ze zadní části osobních i nákladních aut a autobusů, focené výhradně ve dne. K pořízení snímků byl použit digitální fotoaparát OLYMPUS CAMEDIA C-2040ZOOM. Snímky jsou dostupné v rozlišení 640×480 pixelů. Celý projekt se zabývá detekcí znaků v registračních značkách a výsledný program je volně ke stažení na webových stránkách [7] včetně dokumentace v chorvatském jazyce.

Obrázek 4.4: Ukázky snímků z databáze registračních značek pořízeného studenty Fakulty elektrotechniky a výpočetní techniky Univerzity v Záhřebu [7].

4.5.2 Medialab LPR database

Medialab[3] poskytuje rovněž volně dostupné fotografie registračních značek vozidel pro akademické účely [13]. Celkem je k dispozici cca 720 snímků o velikosti 640×480 nebo 1792×1314 pixelů rozdělených do celkem dvanácti složek podle typu fotografií.

Rozdělení do složek je podle toho, zda se jedná o snímky focené ve dne vs. noci, barevné či černobílé, ostré a rozmazané, značka je focena z blízka nebo z dáli, čistá či špinavá, se stíny nebo bez, případně jestli je na jednom snímku jedno nebo i více aut. Vozidla jsou focena zepředu i zezadu a lze zde najít fotografie osobních i nákladních aut. Variabilita sady je tedy značná. Všechny snímky jsou reálné, nikoli programově generované.

Obrázek 4.5: Ukázky snímků z databáze registračních značek z datové sady Medialab LPR dataset [13].

4.5.3 Caltech Cars

Dvě menší datové sady jsou dostupné i na webu institutu *Caltech* (*California Institute of Technology*). Obě byly pořízeny studenty v letech 1999 a 2001.

Starší z nich pojmenovaná *Cars 1999 (Rear)* 2 obsahuje 126 snímků z kalifornského parkoviště, kde jsou všechna vozidla focena zezadu. Snímky mají velikost 896×562 pixelů a jsou v JPEG formátu [1]. Značky na vozidlech jsou všechny přímo viditelné. Na každém snímku je pouze jedna registrační značka. Snímky byly pořizované výhradně ve dne a za slunečného počasí.

Druhá sada, pojmenovaná *Cars 2001 (Rear)* obsahuje 526 snímků. Auta jsou focena opět zezadu a byla pořízena za provozu na silnicích na jihu Kalifornie. Všechny snímky jsou o velikosti 360×240 pixelů, stejně jako v předchozím případě ve formátu JPEG [1]. Snímky byly pořizovány za slunečného počasí, což má za následek to, že na některých snímcích není registrační značka čitelná – viz poslední z ukázek na obrázku 4.6.

Obrázek 4.6: Ukázky snímků z databáze registračních značek z datových sad Cars 1999 (Rear) 2 (první dva snímky) a Cars 2001 (Rear) (třetí a čtvrtý snímek) [1]. Na posledním snímku není registrační značka vozidla čitelná.

[3] http://www.medialab.ntua.gr

4.5.4 AOLP Database

Datová sada *AOLP* (*Application Oriented Licence Plate*) *Database* byla představena v článku Hsua a spol. [6]. Sada je dostupná pro akademické účely a na vyžádání a je rozdělená – stejně jako aplikace, pro kterou byla pořizována – do třech kategorií.

První z nich je pro řízení přístupu (např. vjezd na parkoviště, do soukromého areálu apod.). Zde byly snímky pořizovány statickou kamerou s malou vzdáleností od vozidla a úhlem kamery od vozidla maximálně 30 stupňů. Registrační značka na těchto snímcích zabírá od pětiny do čtvrtiny šířky snímku. Tato kategorie obsahuje celkem 681 snímků. Další kategorií jsou fotografie vozidel za provozu ve městě. Účelem má být identifikace vozidel, která porušila dopravní předpisy jako je nedodržení rychlosti nebo průjezd na červenou. Tyto snímky byly zachycovány kamerou, která se nacházela na kraji silnice. Z toho důvodu je i úhel značky oproti kameře větší než v předchozím případě. Celkem bylo tímto způsobem pořízeno 757 snímků. Poslední částí jsou snímky vozidel na parkovištích. Tato část obsahuje snímky registračních značek z mnohem rozmanitějších úhlů a vzdáleností než v předchozích dvou případech.

Sada jako celek obsahuje snímky registračních značek vozidel zepředu i zezadu, ve dne i v noci s odlišnými osvětleními značek a za různého počasí.

Obrázek 4.7: Ukázky snímků z datové sady *AOLP Database* [6].

4.5.5 Olav's License Plate Pictures

Dalším zdrojem snímků registračních značek je *Olav's License Plate Pictures*[4], který obsahuje fotografie pouze registračních značek vozidel více než devadesáti států, které byly získány v Norsku. Většina fotografií byla pořízena na stojících vozidlech. Součástí webu jsou i statistiky o barvách značek, národních kódech jednotlivých států, historii apod.

4.5.6 PlatesMania

Jedním z nejrozsáhlejších webů věnovaných registračním značkám je *PlatesMania*[5], kde je možné snímky nahrávat bez omezení. Fotogalerie na tomto webu obsahuje několik milionů fotografií registračních značek vozidel z desítek států po celém světě.

4.6 Shrnutí

Problém detekce a rozpoznávání registračních značek vozidel za pomocí neuronových sítí je – jak naznačuje stáří některých zmíněných datových sad – dlouhodobou záležitostí. Není tedy velkým problémem najít datové sady obsahující fotografie ať už celých vozidel, či pouze

[4] http://www.olavsplates.com/
[5] http://platesmania.com/

Datová sada	Den	Noc	Reálné	Umělé	Přední z.	Zadní z.
UFPR-ALPR	✓	✗	✓	✗	✗	✓
Dataturks	✓	✓	✓	✓	✓	✓
MLBLR	✓	?	✓	✓	?	?
SSIG-ALPR	✓	?	✓	✗	✓	?
Projekt LPDRAS	✓	✗	✓	✗	✗	✓
Medialab LPR	✓	✓	✓	✗	✓	✓
Caltech Cars	✓	✗	✓	✗	✗	✓
AOLP Database	✓	✓	✓	✗	✓	✓
Olav's LP Pictures	✓	✓	✓	✗	✓	✓
PlatesMania	✓	✓	✓	✗	✓	✓

Tabulka 4.1: Srovnání snímků jednotlivých datových sad.

Datová sada	Počet	Velikost snímků [px]	Dostupnost	Anotace
UFPR-ALPR	3 500	1920 × 1080	Na požádání	✓
Dataturks	511	různá	Volně	✓
MLBLR	200 000	?	Volně	✓
SSIG-ALPR	6 660	1920 × 1080	Na požádání	✓
Projekt LPDRAS	510	640 × 480	Volně	✗
Medialab LPR	720	640 × 480, 1792 × 1314	Volně	✗
enCaltech Cars	652	896 × 562, 360 × 240	Volně	✗
AOLP Database	757	?	Na požádání	✗
Olav's LP Pictures	?	různá	Volně	✗
PlatesMania	?	různá	Volně	✗

Tabulka 4.2: Srovnání jednotlivých datových sad jako celků.

jejich registračních značek. V této kapitole bylo blíže rozebráno celkem deset datových sad, z nichž ovšem pouze čtyři obsahují i anotace snímků. Zbylých šest zdrojů obsahuje pouze snímky registračních značek, ale neobsahují žádná data, která by se dala použít bez předchozí přípravy přímo pro učení neuronových sítí.

Shrnutí vlastností datových sad jako celků je znázorněno v tabulce 4.2. Srovnání vlastností jednotlivých snímků je pak ukázáno v tabulce 4.1. V tabulkách jsou použity značky s následujícími významy:

✓ sada dané snímky obsahuje

✗ sada dané snímky neobsahuje

? není o dané sadě známo

Kapitola 5

Sběr dat a návrh jejich zpracování

Tato kapitola je zaměřena na způsob, jakým byla pořizována a dále zpracovávána vstupní videa. Část kapitoly je věnována použitým záznamovým zařízením a výsledným videím, která byla z příslušných zařízení získána. Další části pak popisují princip zpracování a technologie, které budou použity.

5.1 Záznamová zařízení

S dnešními možnostmi není problém natočit jakékoli video na téměř každý mobilní telefon, ale odpovídá tomu posléze i kvalita videa. Aby byla videa co nejkvalitnější a poskytovala více možností v části předzpracování, byla natáčena digitální kamerou Panasonic HDC-TM700 s Full HD rozlišením a snímkovací frekvencí 50FPS. Vozidla byla natáčena ze stativu umístěného vedle silnice v odlišné vzdálenosti od pár desítek centimetrů po přibližně dva metry. Odlišné jsou i úhly, pod kterými byla auta natáčena, a to jak ve směru vertikálním, tak horizontálním.

Velmi malá část videí byla natáčena s pomocí digitálního zrcadlového fotoaparátu Nikon D3200. Tato videa jsou ovšem o poznání méně kvalitní – fotoaparát je mnohem citlivější na špatné zaostření a světelné podmínky. Už jen natáčení na křižovatkách po západu slunce mělo za následek pouze špatně zaostřená a velmi tmavá videa. Z tohoto důvodu byla více využívána již zmíněná kamera Panasonic.

Část snímků byla získávána také jako fotografie a videa na mobilní telefon LG G6. Jednalo se převážně o fotografie a videa, která byla získávána za chůze kolem parkujících vozidel. Poslední část pak tvoří videa, která byla pořízena pomocí více druhů kamer za jízdy zevnitř auta.

5.2 Variabilita datové sady

Jelikož není předem známo, k jakému účelu přesně bude vzniklá datová sada využívána, je vhodné zajistit co největší variabilitu dat. Tu lze zajistit co nejvíce kombinacemi následujících parametrů:

- použité záznamové zařízení
- světelné podmínky
- počasí

- vzdálenost vozidla
- úhel mezi vozidlem a záznamovým zařízením
- rychlost projíždějícího vozidla

Mezi použitelná záznamová zařízení lze zařadit mobilní telefony, stacionární kamery, fotoaparáty, nebo kamery v autě. Protože je žádoucí, aby výsledný software dokázal rozpoznávat v jakýchkoli světelných podmínkách, je nutné mít k dispozici vzorky snímků získaných za všech kombinací světelných podmínek a počasí. Je tedy potřeba zohlednit světlo ve dne, v noci, za východu i západu slunce spolu s kombinací deště, mlhy, jasného dne i zatažena. Rovněž je nutné brát v potaz odlišnost ve vzdálenostech aut od kamery a jejich vertikálním i horizontálním úhlem.

5.3 Získaná data

Celkem bylo natočeno přibližně 12 hodin záznamu ze stacionární kamery zmiňované v části 5.1 umístěné v různých vzdálenostech a pod různými úhly vzhledem k projíždějícím autům. Všechny tyto záznamy pocházejí z několika oblastí města Brna. Konkrétně se jedná o Slovanské náměstí, křižovatku ulic Kotlářská a Lidická, Koliště, Poříčí, Slovanské náměstí, Palackého třída a Úvoz. Tyto záznamy byly natáčeny v době dopravní špičky, proto rychlost projíždějících aut je nízká. Ze světelných podmínek je zde zastoupeno přímé slunce svítící proti projíždějícím autům, jasný den, podvečer a večer. Auta byla natáčena jak z levé, tak i pravé strany. Dále bylo získáno 8 hodin záznamu z kamery v autě, točených především v Brně a blízkém okolí. Zde se úhel mezi kamerou a registrační značkou z pochopitelných důvodů příliš nemění. Ze světelných podmínek je zde zastoupeno denní světlo i tma. Počasí bylo slunečné, déšť i pod mrakem. Přibližně 120 snímků pak bylo pořízeno pěší chůzí kolem zaparkovaných aut při tmě.

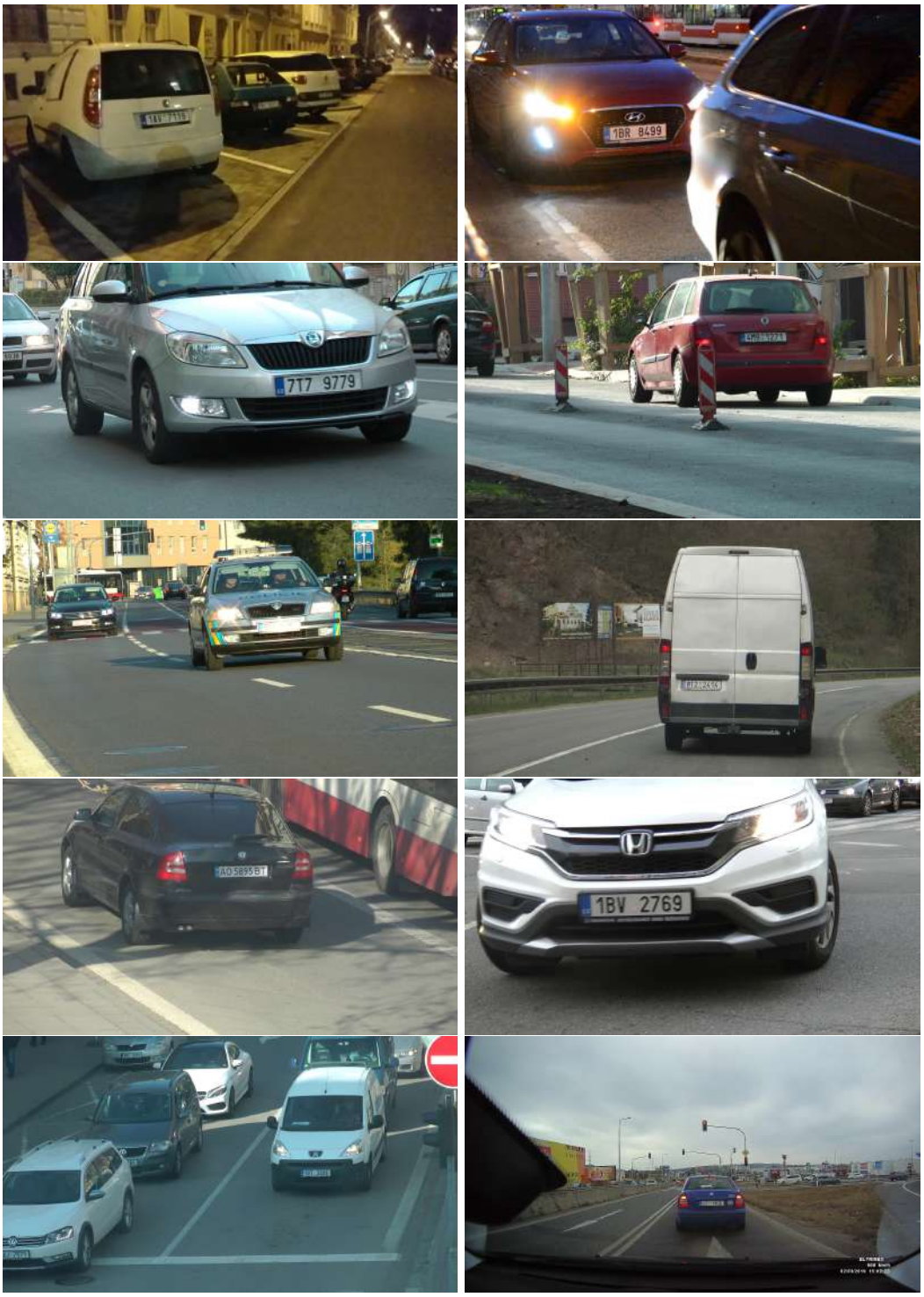
Ukázky snímků, které byly pořízeny v rámci této diplomové práce jsou k nahlédnutí na obrázku 5.1. Ze všech těchto videí bylo získáno více než 17 000 snímků, které byly určeny k dalšímu zpracování.

5.4 Způsob zpracování

Již od samotného návrhu nebylo počítáno s tím, že systém nebude veliký celek spustitelný jedním příkazem. Naopak, bylo zamýšleno, že se zpracování rozdělí na více menších částí, které se budou zpracovávat i pouštět každá zvlášť. Pomyslné části jsou následující

- zpracování videa
- detekce registračních značek
- rozpoznání registračních značek
- ruční korekce automaticky zpracovaných částí

Takovýto přístup je výhodnější v tom, že je možné zpracování více paralelizovat (byť ručně) – v jednu chvíli se při dostatečném výkonu stroje může zpracovávat video ve snímky, detekovat i rozpoznávat registrační značky, a to na různých datech.



Obrázek 5.1: Ukázka snímků pořízených v rámci této práce.

5.5 Použité technologie

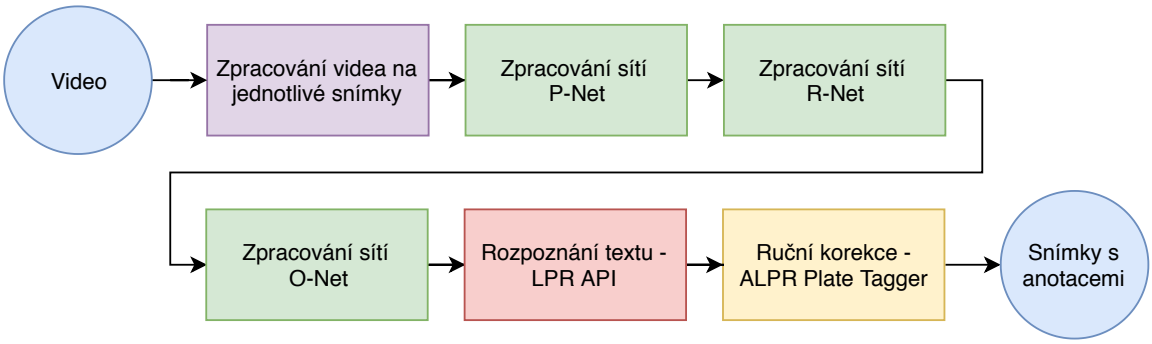
V celém systému bylo využito hned několik technologií a programovacích jazyků. Hlavním programovacím jazykem je Python3 s využitím grafické knihovny OpenCV[1]. Ten je využit ve všech částech kromě ručního zpracování. Pro detekci registračních značek byly použity s využitím frameworku PyTorch[2] neuronové sítě. Rozpoznávání už detekovaných značek zajišťuje *LPR API – Licence Plate Recognition Application Programming Interface*[3]. Jakmile je vše z automatické části detekováno, je možné výsledky nahrát v modifikované verzi aplikace OpenALPR Plate Tagger[4] a ručně opravit případné chyby ve výsledcích automatické části. Tato aplikace byla vyvinuta společností OpenALPR a je dostupná pod *GNU Affero General Public*[5] licencí.

[1] https://opencv.org/
[2] https://pytorch.org/
[3] https://lprapi.docs.apiary.io/#reference/0/license-plates/recognize-lp
[4] https://github.com/openalpr/plate_tagger
[5] https://www.gnu.org/licenses/agpl-3.0.html

Kapitola 6

Implementace automatické části

V této kapitole je podrobně popsáno vytvořené řešení. Celý systém je, jak již bylo zmíněno v předchozí kapitole, rozdělen do několika částí, kde každá z nich může zpracovávat svůj úkol samostatně. Obecné schéma zpracování je znázorněno na diagramu v obrázku 6.1.



Obrázek 6.1: Obecné schéma zpracování vstupního videa ve výsledné snímky s anotacemi. Každá z částí může s vhodně nastavenými volbami pracovat jako separátní celek.

Kapitola se věnuje hlavně těm částem schématu, které zahrnuje automatická část zpracování. Vysvětlen zde bude také způsob nastavování veškerých parametrů, které jsou konfigurovatelné, včetně podrobného výkladu o principu fungování. V kapitole 7 je pak detailně uveden způsob ručního zpracování.

6.1 Zpracování videa

Protože je z hlediska uživatele pohodlnější vstupní data získávat jako video a nikoli přímo jako snímky, bylo nutné vytvořit skript, který dokáže z videa snímky získat. Knihovna OpenCV má tuto funkcionalitu přímo v sobě, nebylo tedy nutné ji implementovat.

Ve zdrojových souborech je rozdělení videa do snímků realizováno pomocí skriptu `split_into_frames`. Tomu je pomocí parametru `-f/--folder` předána cesta ke složce obsahující videa k rozdělení. Druhým parametrem `-n/--numframe` je určeno, každý kolikátý snímek si uživatel přeje uchovat. Vlastní snímky budou uloženy do stejné lokace jako jsou vstupní videa. Pro každé video je pak vytvořena nová složka se stejným jménem jako video s řetězcem `_done` na konci.

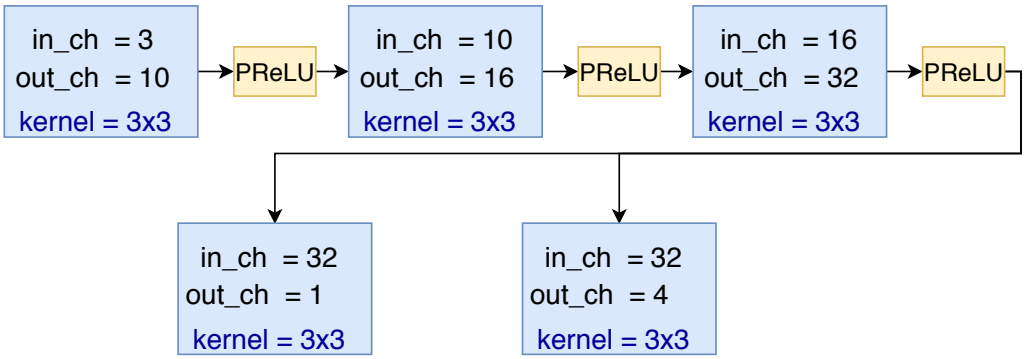
6.2 Detekce registračních značek

K detekci a extrakci registračních značek je využitá neuronová síť založená na síti *MTCNN – Multi-task Convolutional Neural Network*, která byla dopodrobna popsána v kapitole 2.3. Principem je tedy kaskáda složená ze třech neuronových sítí, které jsou lineárně za sebou. Oproti *MTCNN* je ovšem upravena jejich architektura tak, aby byla více přizpůsobena právě detekci registračních značek.

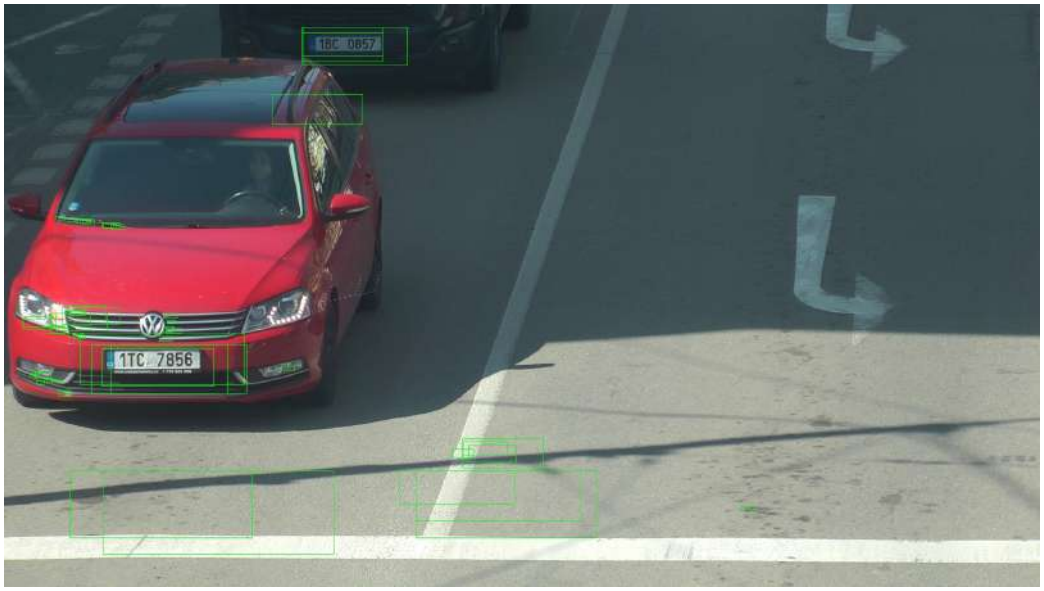
V této kapitole budou dále rozebrány jednotlivé modifikované sítě z původní *MTCNN*. Ačkoli o nich bude hovořeno pouze jako o *P-Net*, *R-Net* a *O-Net*, budou tím myšleny jejich modifikované verze pro detekci registračních značek. V případě, že bude myšlena jejich původní verze, bude to vždy explicitně zmíněno.

6.2.1 P-Net

Architektura první z trojice sítí, *P-Net* neboli *Proposal Network*, je ukázána na obrázku 6.2. Oproti původní síti se učí na výřezech 21×7 pixelů, které lépe reflektují tvar registrační značky a zároveň jejich plocha nejvíce odpovídá ploše používané při učení v původní *P-Net*. Rovněž je z ní odstraněna jedna shlukovací (*pooling*) vrstva a jsou změněny požadované výstupy. Výsledkem *P-Net* sítě by měly být boxy, které s menším okolím ohraničují registrační značky – tzn. návrhy míst, kde by se mohly registrační značky nacházet. Reálným výstupem je pětice hodnot určujících to, s jakou pravděpodobností se v daném místě nachází značka, a přesné určení parametrů boxu – x a y souřadnice levého horního rohu, výška a šířka (v tomto pořadí). Ukázka výsledků sítě je znázorněna na obrázku 6.3.



Obrázek 6.2: Struktura *P-Net* sítě. Modré boxy označují konvoluční vrstvy, žluté pak aktivační funkce. Oproti původní verzi z ní byla odstraněna jedna *pooling* vrstva a učení probíhalo na výřezech 21×7 pixelů (v původní verzi šlo o výřezy s velikostí 12×12 pixelů).



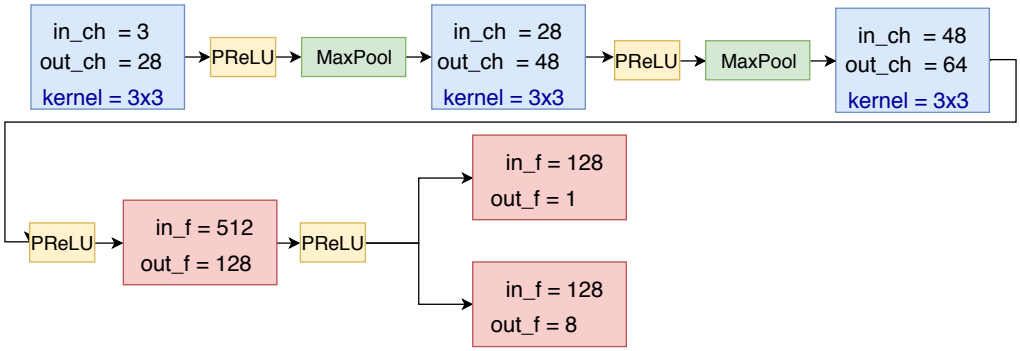
Obrázek 6.3: Ukázka výsledků sítě *P-Net*. Jejím cílem je detekovat s malým okolím místa, kde se nachází registrační značky. Počet detekovaných boxů se pohybuje v řádech desítek až stovek na snímek. Eliminaci falešně pozitivních nálezů a detekci bounding boxu realizují další dvě sítě –*R-Net* a *O-Net* .

6.2.2 R-Net

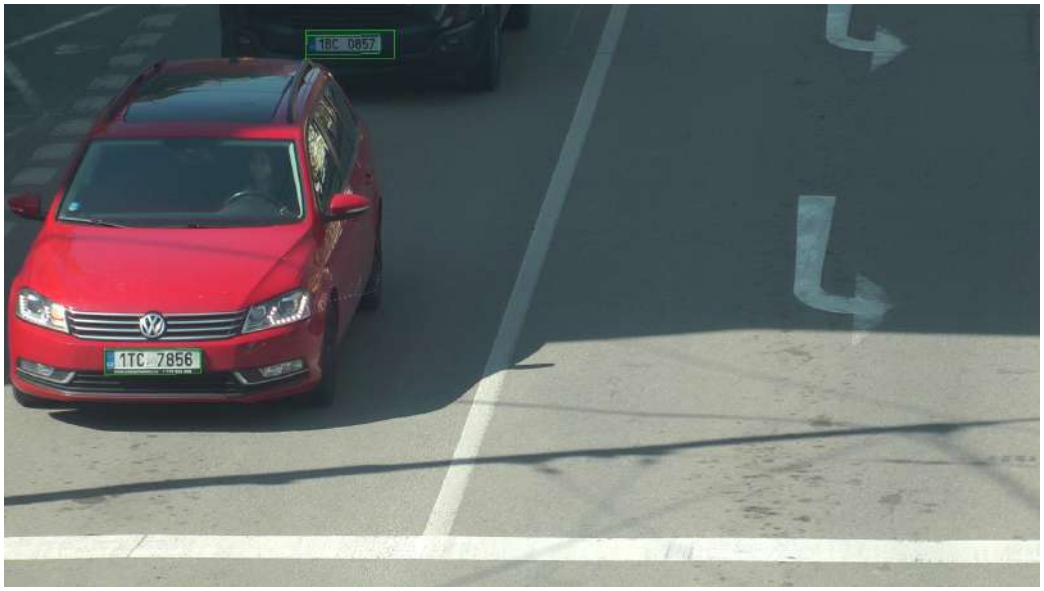
Druhá v pořadí je *R-Net* neboli *Refine Network*. Architektura upravené sítě je znázorněna na obrázku 6.4. Proti původní verzi je učena na výřezech o velikosti 42×14 pixelů (opět z důvodu lepší reflexe tvaru registrační značky). Počet i typ vrstev zůstává zachován, až na poslední plně propojené vrstvy. Vstupem sítě jsou výřezy, které označila předchozí síť jako pozitivní. Cílem je určit pravděpodobnost, s jakou box z předchozí sítě obsahuje registrační značku. Druhým výstupem jsou pak parametry bounding boxu v daném výřezu, ale tentokrát ve formátu x a y souřadnic jeho čtyřech rohů (v předchozí síti se jednalo o x a y souřadnice jednoho rohu, výšku a šířku). Určení přímo čtyřech bodů umožňuje přesně označit i značky, které nejsou přímo rovně proti kameře, ale je mezi nimi a kamerou větší úhel. To v případě předchozí sítě není možné – pokud je registrační značka pod úhlem, box je sítí *P-Net* zvětšen tak, aby značku obsáhl. Po zpracování výsledků předchozí sítě sítí *R-Net* by mělo dojít k výrazné eliminaci detekovaných boxů a to pouze na boxy kolem registračních značek. Rovněž by mělo dojít k úpravě dříve detekovaného boxu tak, aby ohraničoval opravdu pouze registrační značku. Ukázka zpracování snímku 6.3 sítí *R-Net* je znázorněna na snímku 6.5.

6.2.3 O-Net

Poslední v řadě sítí je *O-Net* neboli *Output Network*. Její architektura je velmi podobná předchozí síti. Rozdíl je pouze v tom, že obsahuje jednu konvoluční a jednu *pooling* vrstvu navíc a učí se na dvojnásobně velkých výřezech, tedy o velikosti 84×28 pixelů. Architektura zůstala zachována stejná jako v původní verzi, jen opět se změnou výstupních vrstev. Zde síť, stejně jako *R-Net*, predikuje pravděpodobnost, s jakou box z předchozí sítě opravdu obsahuje registrační značku, a parametry bounding boxu této značky ve formátu x a y

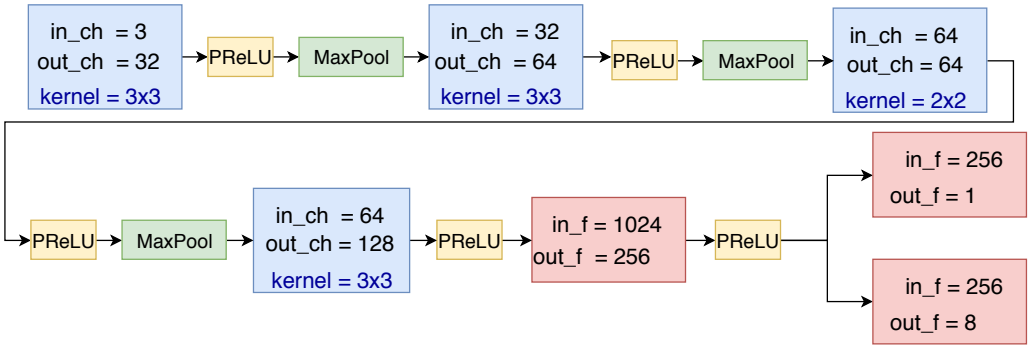


Obrázek 6.4: Struktura *R-Net* sítě. Modré boxy označují konvoluční vrstvy, zeleně jsou označeny *pooling* vrstvy, červeně plně propojené vrstvy a žlutě pak aktivační funkce. V *pooling* vrstvách je použito jádro o velikosti 3×3 s krokem dva – tedy stejné parametry jako v původní verzi. Síť je oproti původním 24×24 učena na výřezech o velikosti 42×14 pixelů. Výsledkem sítě je pravděpodobnost, s jakou box z předchozí sítě opravdu obsahuje registrační značku, a parametry bounding boxu této značky ve formátu x a y souřadnic jeho čtyřech bodů.

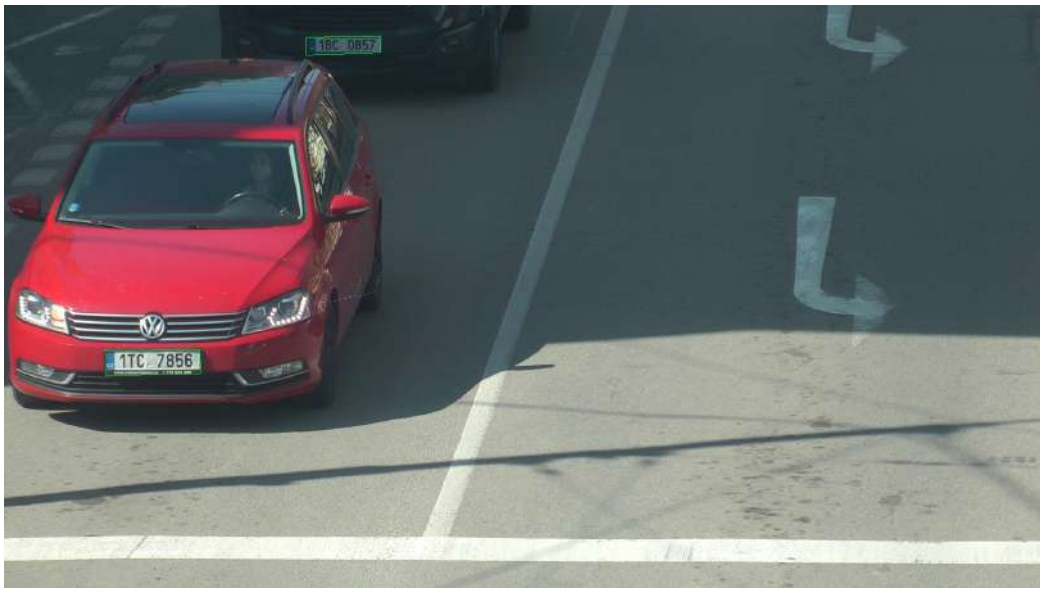


Obrázek 6.5: Ukázka výsledků sítě *R-Net*. Jejím úkolem je eliminovat počet výsledků z předchozí sítě, *P-Net*, tak, aby pokud možno zůstaly pouze boxy, které ohraničují registrační značky – bounding boxy. Nepřesné odhady při regresi bounding boxu (moc malý, nebo naopak příliš veliký box), či případné falešně pozitivní boxy by měla následně doopravit další síť, *O-Net*.

souřadnic jeho čtyřech bodů. Protože předchozí síť produkuje poměrně dobré výsledky a negativních vzorků, které produkuje není mnoho, byla síť *O-Net* učena na podstatně větším množství pozitivních vzorků s cílem co nejsprávněji určovat parametry bounding boxu.



Obrázek 6.6: Struktura *O-Net* sítě. Modré boxy označují konvoluční vrstvy, zeleně jsou označeny *pooling* vrstvy, červeně plně propojené vrstvy a žlutě pak aktivační funkce. V *pooling* vrstvách bylo použito jádro o velikosti 3×3 s krokem dva – tedy stejné jako v původní verzi. Síť byla oproti původním 48×48 učena na výřezech o velikosti 84×28 pixelů. Výstupy sítě jsou stejné jako v předchozím případě – tedy pravděpodobnost registrační značky a parametry bounding boxu ve formátu x a y souřadnic jeho čtyřech bodů.



Obrázek 6.7: Ukázka výsledků poslední v kaskádě sítí, *O-Net*. Jejím cílem je opravit případné chyby v predikci předchozí sítě – *R-Net*. Rozdíl oproti výsledkům předchozí sítě je viditelný hlavně na značce více vzadu – zde *R-Net* stále ještě detekovala dva boxy, zatímco *O-Net* nesprávný již eliminovala a zůstal pouze jediný.

6.2.4 Proces učení sítí

Podkladem pro učení všech tří sítí byla datová sada *ZoomLPDataset* poskytnutá FIT VUT Brno a také část anotací vlastnoručně zpracovaných v rámci této práce. Sada *ZoomLPDataset* obsahuje 20 131 snímků, kde se na každém z nich vyskytuje přibližně 1–8 anotovaných registračních značek. Anotace snímků jsou ve formátu *pickle* a obsahují např. parametry bounding boxu, regresního boxu (box ohraničující bounding box), textu na značce, míry jistot aj. Většina snímků této sady byla získávána pomocí kamery umístěné na mostě snímající auta projíždějící na silnici vedoucí pod mostem. Auta byla natáčena jak zepředu, tak i zezadu, za dne při slunci, standardním světle, i za přítmí. Variabilita světelných podmínek je tedy dobrá. Ovšem úhel, pod kterým byla auta natáčena, je stále velmi podobný. Odlišné je pouze přiblížení.

Pro učení *P-Net* byly z datové sady využity regresní boxy. Ty jsou zde uloženy jako čtyři hodnoty označující x a y souřadnice levého horního rohu, jeho výšku a šířku. Pro *R-Net* a *O-Net* pak byly využity bounding boxy, které jsou ve formátu x a y souřadnic čtyřech rohů v pořadí levý horní, pravý horní, pravý dolní a levý dolní. Protože se pracuje s více typy boxů, byla v rámci řešení vytvořena samostatná třída `Box`, která mezi těmito formáty dokáže převádět, či dokonce určit, o který z nich se jedná, a s ním dále pracovat.

K načítání dat z *ZoomLPDataset* sady byla vytvořena `ZoomLPPickleDatasetLoader`, která konvertuje data z původního *pickle* formátu do interní struktury `Sample`. Rovněž je zde třída `PlateTaggerYamlDatasetLoader`, která dokáže vytvořit stejnou interní strukturu `Sample` ze souborů typu *YAML*. S těmito pracuje aplikace pro ruční zpracování, která je rozebrána podrobně v kapitole 7. Druhá zmíněná třída byla vytvořena proto, aby bylo možné sítě doučit i na datech, která *ZoomLPDataset* neobsahuje. Typickým příkladem jsou registrační značky, které jsou z většího bočního úhlu.

Třídy a metody, které jsou společné pro všechny tři sítě a zabývají se vytvořením datové sady, se nachází v modulu `dataset_common`. Jedná se především o třídy reprezentující jednotlivé části datové sady. Z těch důležitých lze jmenovat:

- `Box`: reprezentující jeden bounding box nebo regresní box
- `DatasetLicensePlate`: reprezentující jednu značku datové sady
- `BoundingBoxManager`: zastřešující všechny operace nad boxy – přeškálování, výřez ze snímku, ukládání, *NMS –Non Maxima Suppression*, *IOU – Intersection over Union*
- `MTCNNDataset`: třída držící seznam všech položek v datasetu, včetně metod pro načtení jednotlivých položek. Je společným předkem pro obdobné třídy specializované pro potřeby jednotlivých sítí a sama vychází z třídy `Dataset` poskytované knihovnou PyTorch.

O úpravu dat pro každou konkrétní síť se pak starají třídy v konkrétních modulech – `p_net_dataset`, `r_net_dataset` a `o_net_dataset`. Tyto třídy generují pozitivní i negativní vzorky správné velikosti a formátu. Při vytváření instancí specializovaných tříd je nutné předat jako parametr slovník obsahující pole cest k souborům datasetu a jméno třídy, která se má použít pro čtení anotací - tedy buďto `ZoomLPPickleDatasetLoader`, nebo `PlateTaggerYamlDatasetLoader`. V jednom datasetu lze obě možnosti libovolně kombinovat a lze tedy načítat data v obou formátech zároveň. Z původního jednoho pravdivého (*Ground Truth*) vzorku získaného z trénovacích dat jsou náhodnými posuvy vytvořeny celkem čtyři pozitivní vzorky nejdříve pro *P-Net*. Posléze jsou ještě většími posuvy vytvořeny

pro *P-Net* i čtyři negativní vzorky. Při tvorbě negativních vzorků je vycházeno z předpokladu, že auto má určitou výšku i šířku a není tudíž možné, aby se posuvem např. o délku jedné registrační značky doleva označila jako negativní vzorek registrační značka vedlejšího vozidla.

Tímto postupem je vytvořena datová sada (nejprve pro *P-Net*) obsahující přibližně stejný počet pozitivních i negativních vzorků. Vzorky obsahují výřez obrázku o velikosti 21×7 pixelů, informaci o tom, zda se jedná o pozitivní či negativní vzorek a parametry regresního boxu. V případě negativního vzorku obsahuje regresní box samé nuly. Z takto připravené datové sady jsou použity dvě třetiny dat jako trénovací a jedna třetina jako testovací.

K vlastnímu učení pak slouží moduly `p_net`, `r_net` a `o_net`. Metody či třídy, které by se jinak opakovaly ve více modulech, byly přesunuty do modulu společného – `net_common`. Zde jsou definovány metody pro učení i testování, použité chybové funkce, normalizace vzorků z datové sady či ukládání výsledků i stavů sítí. Výchozí společnou třídou zajišťující tuto funkcionalitu je `MTCNNModule`. V sítích *R-Net* a *O-Net* je pak využit vylepšený potomek, `MTCNNPerfectionModule`. Potřebné úpravy společných částí pro danou síť jsou realizovány za pomocí dědičnosti a přetěžovaní metod v konkrétním modulu. Každý z nich obsahuje definici stejnojmenné sítě, přetížené metody z `net_common` i proměnné udávající, zda se bude síť spouštět pro trénování, testování, či obojí. K tomu slouží logické proměnné `DO_TRAINING` a `DO_TESTING` na začátku všech modulů, které umožňují učit sítě odděleně.

Ke všem modulům existuje i stejně pojmenovaný konfigurační soubor s příponou *.conf*, ve kterém je možné nastavovat parametry pro učení i samotnou detekci. Soubor je ve formátu *JSON* a může obsahovat následující parametry:

- `trained_state_dir`: cesta k adresáři s už naučenými sítěmi (výchozím je adresář `trained_nets`)
- `training_epochs`: počet průchodů přes vytvořenou datovou sadu při trénování
- `training_mode`: 1 pokud má být síť učena od znova, 2 pokud jde pouze o doučení
- `batch_size`: velikost dávky (*batch*) při učení
- `report_positive_counts`: při nastavení na logické *true* bude vypisovat počet nalezených boxů ve snímku
- `certainty_threshold`: určuje práh, od kterého výše bude síť považovat svoje výsledky za pozitivní
- `dataset_workers`: počet vláken (*workers*), které se budou používat pro vytvoření datové sady (čím větší číslo, tím rychlejší, ale náročnější na zdroje bude zpracování)
- `dataset_shuffle`: určuje, zda se data ve vytvořené datové sadě budou míchat, či nikoli
- `results_save`: udává, zda má daná síť do separátní podsložky v `data/positives/` ukládat snímky s vyznačenými nalezenými boxy
- `nms_threshold`: práh pro *NMS* (*Non Maxima Suppression*) – hodnota mezi 0 a 1, čím blíže k 1, tím více překrývajících se boxů bude po provedení *NMS* zůstávat

- `iou_max_negative_threshold`: práh pro *IOU* (*Intersecrion Over Union*) – při učení *R-Net* a *O-Net* určuje, do jaké hodnoty *IOU* s výsledky předešlé sítě je vzorek brán jako negativní
- `iou_min_negative_threshold`: práh pro *IOU* – při učení *R-Net* a *O-Net* určuje, od jaké hodnoty *IOU* s výsledky předešlé sítě je vzorek brán jako negativní (při hodnotě 0 a učení *R-Net* docházelo k tomu, že v datové sadě bylo několik set tisíc negativních vzorků na nízké tisíce pozitivních, proto je vhodné tuto hodnotu lehce zvýšit aby byl poměr pozitivních a negativních vzorků více vyvážený)
- `iou_min_positive_threshold`: práh pro *IOU* – při učení *R-Net* a *O-Net* určuje, od jaké hodnoty *IOU* s výsledky předešlé sítě je vzorek brán jako pozitivní
- `pyramid_max_width`: maximální šířka snímků v pyramidě, které vytváří *P-Net* při zpracování dat
- `pyramid_min_width`: minimální šířka snímků v pyramidě, které vytváří *P-Net* při zpracování dat
- `pyramid_step`: krok při přeškálovávání snímků v pyramidě, které vytváří *P-Net* při zpracování dat

Pokud některý z parametrů není definován, použijí se výchozí hodnoty definované v třídě `NetConfig` v modulu `net_common`.

Použitou chybovou funkcí pro určení toho, zda je na daném výřezu registrační značka, je *Binary Cross Entropy Loss*. K regresi boxů je použita *Mean Square Error*. Obě tyto funkce byly podrobněji popsány v kapitole 2.1.2.

Každá ze sítí musí být učena zvlášť, přičemž proces učení je možné spustit v každém z modulů `p_net`, `r_net` či `o_net`. Před spuštěním je nutné nastavit alespoň proměnnou `DO_TRAINING` a předat slovník ve formátu:

```
[{
    'paths': ZOOMLP_PICKLE_PATHS,
    'loader_class': "ZoomLPPickleDatasetLoader"
}]
```

Hodnota klíče `paths` obsahuje pole cest k souborům s anotacemi ve formátu *pickle* či *YAML*. Výchozím adresářem obsahujícím data k učení je adresář *data* nacházející se na stejné úrovní jako adresář *src*. V `loader_class` je pak `ZoomLPPickleDatasetLoader` nebo `PlateTaggerYamlDatasetLoader` (podle typu anotací).

Již v průběhu učení sítě ukládají svůj stav, a to každou stou úpravu koeficientů (*batch*) a na konci každého průchodu datové sady do složky `trained_nets` pod názvem sítě, tedy například `PNet.pt`. Na konci každého průchodu datovou sadou je síť přepnuta do vyhodnocovacího režimu a je spočítána její chyba na testovacích datech. Pokud byla chyba nižší než předešlá, je stav sítě uložen do `<název_sítě>.pt.best`. V případě, že byla chyba vyšší, je uživatel upozorněn, že mohlo dojít k přeučení sítě a v `<název_sítě>.pt.best` zůstane stav sítě po nejmenší dosažené chybě na testovacích datech. Již v průběhu učení je na standardní výstup vypisována každý stý *batch* hodnota chybové funkce v dané *batch*, celém cyklu i celková hodnota.

Pro učení *R-Net*, resp. *O-Net* je nutné mít ve stejné složce jako jsou snímky uloženy rovněž soubory obsahující název snímku s koncovkou `PNet.pkl`, resp. `RNet.pkl`. Tyto soubory obsahují výsledky příslušné sítě pro daný snímek. Tedy například k obrázku s názvem *im_00001.jpg* bude existovat soubor *im_00001.jpg.PNet.pkl* v němž jsou uloženy výsledky sítě *P-Net* pro tento snímek, které budou použity sítí *R-Net*.

Ke zjednodušení procesu učení slouží modul `net`, který při spuštění s parametry příkazové řádky `--mode=train` a `--input=<anotovaná_data>` provede výše uvedené kroky v požadovaném pořadí. Tedy provede učení sítě *P-Net*, zpracování datové sady sítí *P-Net*, učení sítě *R-Net* na výsledcích sítě *P-Net*, zpracování datové sady sítí *R-Net* a učení sítě *O-Net* na výsledcích sítě *R-Net*.

6.2.5 Zpracování snímku a výsledné anotace

Pro zpracování jednotlivého snímku, či složky se snímky, slouží modul `net`. Přes parametr `-i/--input` příkazové řádky je možné předat cestu k souboru, složce či seznamu složek (použitím několika parametrů `-i/--input`) obsahující snímky ke zpracování. Parametry tohoto zpracování je možné nastavit ve formátu *JSON* v konfiguračním souboru `Net.conf`. Mezi volby dostupné v tomto souboru patří následující:

- `yaml_save`: určuje, zda se budou ukládat výsledky sítí ve formátu *YAML*
- `yaml_overwrite_existing`: při nastavení této volby na *true* budou soubory ve formátu *YAML*, které už v dané složce existují, přepsány (pozor - může dojít k přepsání ručně anotovaných dat!)
- `pickle_save`: udává, zda bude každá síť ukládat své výsledky v *pickle* formátu, či nikoli
- `use_remote_plate_recorgnizer`: pokud má systém rozpoznávat text na detekovaných registračních značkách, je nutné tuto volbu nastavit na *true*
- `force_processing`: v případě, že ve složce již existují *pickle* soubory s mezivýsledky sítí, je možné je touto volbou ignorovat a zpracovat znova
- `nets_to_process`: pole s názvy tříd sítí, které se mají použít pro zpracování dat. Například pro učení sítě *R-Net* je potřeba data zpracovat pouze sítí *P-Net*.
- `p_net_config`, `r_net_config`, `o_net_config`: slovníky s konfigurací pro jednotlivé sítě - lze použít stejné parametry, jako byly popsány výše.

Nastavením vhodných voleb je možné zpracování provést naráz – tzn. zpracovat všemi sítěmi a nakonec rozpoznat text v detekovaných boxech, nebo naopak zpracování rozdělit na více etap. Rozdělení může být vhodné v případě, že je dočasně nedostupné *API* pro rozpoznávání, nebo se zdá, že některá ze sítí výsledky více zhoršuje než naopak. Při zpracování po částech sítě ukládají do již zmiňovaných souborů `<název_snímku>.<název_sítě>.pkl`. Ukládání těchto souborů je možné nastavit či vypnout, stejně jako je možné určit, zda se případné již existující soubory s tímto názvem budou či nebudou přepisovat.

Ze vstupního snímku, určeného ke zpracování nejdříve sítí *P-Net*, je vytvořena tzv. pyramida. Vstupní snímek je tedy přeškálován na větší množství snímků s různými velikostmi. Maximální, resp. minimální šířka snímku a krok v pyramidě jsou dány příslušnými parametry v konfiguračním souboru. Ze vzniklé pyramidy je vytvořena datová sada vhodná pro

zpracování sítí. Části snímků, které síť označí jako registrační značky a mají větší velikost než 2×6 pixelů, jsou uloženy. Omezení na velikost je přidáno proto, aby se předešlo zbytečným detekcím, neboť takto malé výřezy, i kdyby byly registračními značkami, s velkou pravděpodobností nebudou rozpoznatelné. Než jsou výřezy definitivně uloženy, jsou přeškálovány zpět na rozměry odpovídající velikosti snímku. Výsledky předchozí sítě pak dostane na vstupu další z kaskády a provede příslušné korekce a předá výsledky další síti, *LPR API* pro rozpoznání znaků, nebo metodě pro uložení do výstupního souboru.

Konečné výsledky jsou pak uloženy ve formátu *YAML*, se kterým je schopen dále pracovat software pro ruční korekci. Pro každou detekovanou registrační značku je vytvořen samostatný soubor obsahující informace pouze o této značce. V případě, že je na snímku značek více, je vytvořeno více souborů. Níže je možné vidět příklad formátu a informací, které jsou v tomto souboru uloženy:

```
image_file : im_00001.jpg
image_width : 2080
image_height : 1560
plate_corners_gt : 1003 520 1450 537 1432 884 988 857
region_code_gt : CZ
plate_number_gt : 3L8575
plate_inverted_gt : false
plate_certainty : 1
plate_recognition_confidence : 1
```

Kde jednotlivé položky mají následující význam:

- `image_file`: název snímku, na kterém je daná registrační značka detekována
- `image_height`: výška snímku
- `image_width`: šířka snímku
- `plate_certainty`: jistota, se kterou jde o registrační značku
- `plate_corners_gt`: x a y souřadnice čtyř rohů bounding boxu registrační značky v pořadí levý horní, pravý horní, pravý dolní a levý dolní
- `plate_inverted_gt`: *false* v případě, že je na značce tmavý text na světlém pozadí, *true* jinak
- `plate_number_gt`: text, který byl *LPR API* rozpoznán
- `plate_recognition_confidence`: jistota, se kterou byl tento text rozpoznán
- `region_code_gt`: kód země, ze které registrační značka pochází

Položky `plate_recognition_confidence` a `plate_certainty` jsou před fází ručního zpracování menší než jedna a označují reálné míry jistoty, které vrátily příslušné nástroje. Po fázi ručního zpracování jsou ovšem nastaveny na hodnotu 1. Vychází se totiž z předpokladu, že po ručním zpracování byly výsledky všech sítí zkontrolovány a jsou tedy opravdu správně.

6.3 Rozpoznání registračních značek

K rozpoznání textu na registračních značkách je použit nástroj *LPR API* provozovaný FIT VUT Brno. Tomu jsou k rozpoznání posílány výřezy, které sítě detekovaly jako poznávací značky. Tyto výřezy jsou zakódovány do formátu base64 a jako *HTTP POST request* odeslány k rozpoznání. Přesné formáty dotazů i odpovědí jsou popsány v dokumentaci[1] a nebudou zde již podrobně rozebírány.

Modul, který zajišťuje komunikaci s *LPR API* v rámci této práce, se nazývá `lpr_fit_api`. Metodě `recognize` v tomto modulu lze předat výsledky zpracování neuronovými sítěmi a ta provede příslušné kontroly, odeslání požadavku a zpracování odpovědi. Do výsledku od neuronových sítí přidá informace o rozpoznaném textu a jistotě rozpoznání a vrátí jej zpět. V požadavku je nutné definovat klíč, respektive uživatelský identifikátor, kde byla použita hodnota „FIT“, resp. „xkvapi12-LPdatabase“.

Další funkcionalita, která je modulem poskytována, je režim kontroly *YAML* souborů ve složce. Ta je mu předána přes argument `-f/--folder` příkazové řádky. Modul v takovém případě projde všechny *YAML* soubory a pokud v nich chybí některé položky, automaticky je doplní. Mezi tyto položky patří text registrační značky, míra jistoty jejího rozpoznání, kód země a jistota toho, zda jde o značku. Primární účel tohoto režimu je hromadné doplňování textu registrační značky do výsledků sítě, neboť se může stát, že použití *LPR API* v rámci hromadné detekce velmi významně prodlouží dobu potřebnou pro získání kompletních výsledků. Je-li k dispozici časově omezené množství výkonných zdrojů, je výhodnější nejprve vše zpracovat sítěmi a později na méně výkonném hardware doplnit text dotazováním na *LPR API*.

[1] https://lprapi.docs.apiary.io/#reference/0/license-plates/recognize-lp

Kapitola 7

Ruční zpracování

Protože automatická část nefunguje vždy spolehlivě a trénovací data nepokryla všechny světelné podmínky a úhly kamery, je nutné chyby v predikcích ať už neuronových sítí či rozpoznávacího *LPR API* manuálně opravit. K tomuto účelu byl použit nástroj vyvinutý společností OpenALPR. Popis funkcionality tohoto nástroje včetně jeho vylepšení pro zjednodušení ručních korekcí je podrobně popsán v této kapitole.

7.1 Uživatelské rozhraní

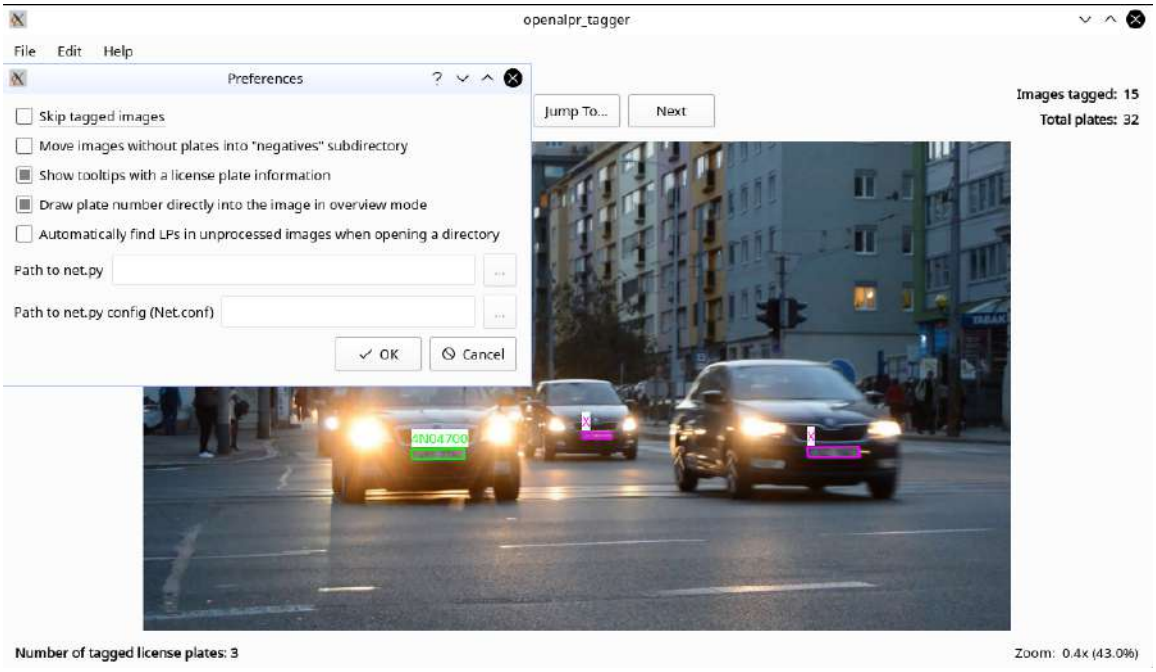
Pro přehledné zobrazení snímků i jeho anotací byl využit nástroj OpenALPR Plate Tagger[1]. Jedná se o program napsaný v programovacím jazyce C++ s využitím grafické knihovny Qt[2]. Vyžaduje, aby jednotlivé snímky měly ve stejné složce, jako jsou ony, dostupné i soubory ve formátu *YAML* a to tak, že pro každý snímek může (a nemusí) být dostupný jeden či několik souborů obsahující informace o registračních značkách přesně tak, jak jsou ve výsledku ukládány neuronovými sítěmi. Podrobné informace k těmto souborům jsou uvedeny v předchozí kapitole. V případě, že k snímku neexistuje soubor, program jej sám vytvoří na základě informací dodaných přes grafické uživatelské rozhraní. Snímky neobsahující žádnou registrační značku mohou ve složce zůstat, nebo mohou být přesunuty do složky *negatives* nacházející se ve stejném adresáři. Tuto volbu lze nastavit pomocí konfiguračního dialogu přímo v grafickém rozhraní aplikace.

Po spuštění programu si uživatel vybere složku (obsahující snímky a anotace), se kterou si přeje pracovat. Program automaticky vše načte a zobrazí náhled prvního ze snímků. Uživatel má možnost načtené anotace zkontrolovat a případné chyby opravit, falešné detekce smazat, či chybějící data doplnit. Až je vše hotovo, může se uživatel přesunout na další snímek a celý proces opakovat, dokud není zpracovaná celá složka.

Program byl mírně upraven tak, aby poskytoval veškerou funkcionalitu, která je nutná pro pohodlnou obsluhu a opravu automaticky vytvořených anotací. První modifikací bylo přidání vykreslení textu registrační značky přímo do náhledu snímku. Možnost vykreslování textu byla přidána i do voleb v nastavení programu. Původně totiž bylo nutné pro zobrazení textu značky na ni najet myší, nebo označený box přiblížit pro úpravy, což přináší zdržení. Následovalo barevné rozlišení bounding boxů okolo značek, kde je rozpoznatelný text (zelená barva) a kde je text natolik rozmazaný či v dáli, že není rozpoznatelný ani lidským okem (růžová barva). Druhé zmíněné jsou označovány z toho důvodu, že případné

[1] https://github.com/openalpr/plate_tagger
[2] https://www.qt.io/

další sítě, které tuto sadu budou využívat, by neměly takové výřezy používat ani jako pozitivní (protože na nich nelze text rozpoznat), ale ani jako negativní (protože se o registrační značku jedná). Poslední úpravou bylo dodání automatických anotací do vybrané složky obsahující pouze snímky. Tuto volbu je však vhodné použít pouze v případě, že je dostupný dostatečně výkonný hardware, nebo je ve složce pouze malé množství snímků. Automatické zpracování trvá poměrně dlouhou dobu a je náročné na využití zdrojů. Náhled modifikovaného grafického rozhraní aplikace včetně všech možností nastavení je ukázán na obrázku 7.1. Na obrázku 7.2 je ukázáno dialogové okno pro ruční anotaci značky.



Obrázek 7.1: Grafické uživatelské rozhraní aplikace OpenALPR Plate Tagger. Přidáno bylo zobrazování detekovaného čitelného textu registračních značek (zelená barva) i značek, které nejsou čitelné ani lidským okem (růžová barva, označeno písmenem X). Do možností nastavení byly přidány volby pro automatickou detekci a rozpoznání značek a vykreslování textu přímo do náhledu snímku.

7.2 Způsob zpracovávání

Celé zpracování včetně automatické části probíhá přesně tak, jak bylo popsáno na začátku kapitoly 6. Prvně je tedy nutné použít skript `split_into_frames` pro převod videa na snímky. Následně předat skriptu `net` cestu se snímky určenými k automatickému zpracování. Zde budou doplněny automatické detekce bounding boxů registračních značek i textu na nich pomocí *LPR API*. Jakmile bude vše hotové, je možné načíst složku s automatickými anotacemi v nástroji OpenALPR Plate Tagger a ručně je zpracovat.

Protože konvoluční neuronové sítě potřebují pokud možno co největší množství dat, byla vyvinuta snaha rozšířit vytvořené nástroje mezi co největší počet lidí, kteří by mohli být schopni pomoct s ručními korekcemi. Pro tyto účely (primárně tedy ruční korekci), byl vytvořen obraz virtuálního stroje, ve kterém je kompletně připravené celé pracovní prostředí pro práci s nástrojem OpenALPR Plate Tagger i neuronovými sítěmi. Rovněž byl

Plate number: 1B65938 Plate region: Light-on-Dark letters

Help Reset corners Delete & Save Save Cancel

Obrázek 7.2: Náhled dialogového okna pro ruční anotaci značky. Je zde možné změnit souřadnice rohů registrační značky a její text, doplnit zkratku státu, odkud registrační značka pochází, případě celý box smazat.

vytvořen podrobný návod obsahující jak popis instalace programu Virtual Box a importu připraveného virtuálního stroje, tak i návod k obsluze nástroje OpenALPR Plate Tagger. V obrazu virtuálního stroje je naklonovaný veřejný repozitář nacházející se na fakultním serveru. Tento repozitář obsahuje adresář `plate_detector` s kompletními zdrojovými soubory k automatickému zpracování a adresář `plate_tagger` s upraveným nástrojem OpenALPR Plate Tagger. Při jakékoli změně v repozitáři je dostupný ve virtuálním stroji skript pro snadné stažení a přeložení nové verze nástrojů. Obraz virtuálního stroje ovšem není vhodné používat k automatickému zpracování a to z důvodu náročnosti výpočtu neuronovými sítěmi na potřebný hardware.

Kapitola 8

Dosažené výsledky

Kapitola shrnuje výsledky a ukázky nejprve automatického a posléze i ručního zpracování snímků. Na konci kapitoly jsou pak uvedeny návrhy na zlepšení převážně automatické části a zvětšení efektivity zpracování.

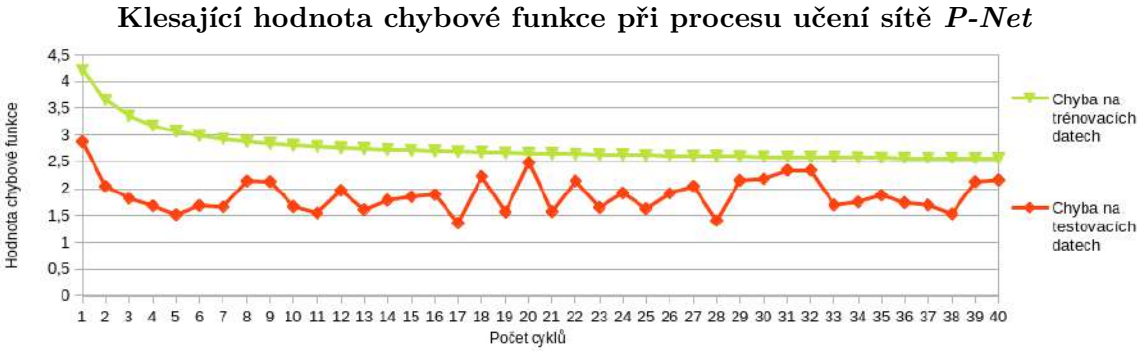
8.1 Shrnutí výsledků automatické části

Na grafech 8.1, 8.2 a 8.3 jsou ukázány rychlosti klesání hodnoty chybové funkce v závislosti na počtech průchodů přes všechna data v trénovací datové sadě pro každou síť. Hodnota chybové funkce je součet chyb při detekci značky a při regresi bounging boxu. Pro připomenutí – chybová funkce pro detekci značky je *Cross Entropy Loss* využivající logaritmus, zatímco pro regresi bounding boxu byla použita *Mean Square Error*, kde chyba stoupá s druhou mocninou rozdílů predikovaných a pravdivých hodnot. Na ose x je vždy počet průchodů přes celou datovou sadu, na ose y pak hodnota chybové funkce.

V případě všech tří sítí se stalo, že chyba na testovacích datech je menší, než na trénovacích, což je poněkud nečekané. Obvykle se naopak předpokládá, že chyba na testovacích datech bude vyšší. Opakovaným zkoumáním výpočtu chyby jsem neobjevila žádný problém v kódové části a zdá se, že se síť opravdu takto chová. Domnívám se, že možností vysvětlení by mohlo být hned několik. Zaprvé by to mohla zapříčinit rozdílnost dat v trénovací a testovací sadě. Při vytváření datové sady jsou použity standardní metody frameworku PyTorch umožňující sadu náhodně promíchat. Mohlo se tedy stát, že v testovací části sady byly mnohem jednodušší vzorky než v trénovací. Dalším případem, kdy se stává, že je testovací chyba menší než trénovací, je při použití regularizačních prostředků v síti, jako je například vrstva *Dropout*, která je použita pouze při trénovací, ale nikoli už testovací fázi. Ta je však použita pouze u sítě *O-Net* a nevysvětluje to tedy chování ostatních sítí. Třetí možností by byl samotný způsob výpočtu chyby. Zatímco v trénovacích datech se chyba měří jako průměr chyby nad trénovací sadou přes celý jeden průchod daty, projeví se (nejvýrazněji pak v prvním průchodu) mnohem více větší chyba ze začátku učení. Naproti tomu chyba v testovacích datech je spočítána sice jako průměr nad testovací sadou, ale už po skončení jednoho průchodu učení. Což tedy znamená, že počáteční největší chyby v predikci už nejsou v hodnotě testovací chybové funkce zahrnuty, kdežto v trénovací ano.

U sítě *P-Net* bylo pro trénování použito celkem 40 průchodů nad datovou sadou čítající 100 664 pozitivních a stejný počet negativních vzorků. Z celkového počtu 201 328 vzorků byly dvě třetiny použity jako trénovací a zbylá třetina jako testovací. V grafu 8.1 je zná-

zorněna hodnota chybové funkce na trénovacích datech (zelená křivka) i testovacích datech (červená křivka) po každém průchodu trénovacími daty.

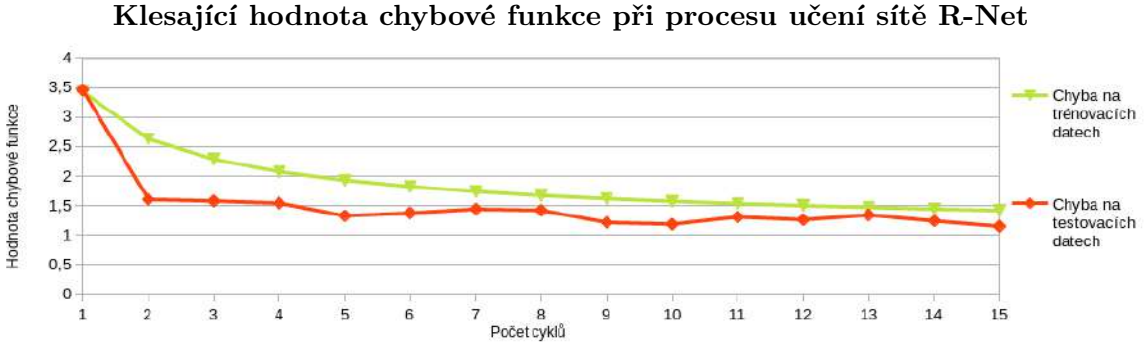


Obrázek 8.1: Graf zobrazující hodnotu chybové funkce při učení sítě *P-Net*. Hodnota chyby na trénovacích datech (zelená křivka) konstantně klesá, zatímto chyba na testovacích datech (červená křivka) osciluje mezi hodnotami 1.5 a 2.5. Vzhledem k tomu, že většinu hodnoty chybové funkce tvoří chyba při regresi bounding boxu a primárním účelem sítě není co nejpřesnější regrese, nepředstavuje toto zásadní problém při celkovém zpracování všemi sítěmi.

Dle očekávání chyba na trénovacích datech konstantně klesá po celou dobu učení. Oproti tomu chyba na testovacích datech konstantně klesá do pátého průchodu daty a dále její hodnota osciluje přibližně mezi hodnotami 1.5 a 2.5. Většinu hodnoty chyby tvoří nepřesnosti při regresi bounding boxu a rozdíl mezi těmito hodnotami, který je roven 1 v tomto případě s použitím *MSE* jako chybové funkce znamená, že síť, která pracuje s výřezy o velikosti 21×7 pixelů mezi jednotlivými průchody predikuje rohy boxu v průměru o 0.5 pixelu posunuté. Pro představu – snímky z datové sady *ZoomLPDataset* obsahovaly registrační značky o přibližné velikosti 150×50 pixelů, takže v přepočtu na původní velikost vstupního snímku se predikce sítě *P-Net* mezi jednotlivými průchody liší o necelé čtyři pixely. Vzhledem k tomu, že primární cíl sítě je eliminace míst, které registračními značkami nejsou (pro přesnou regresi slouží další sítě), není takto oscilující chyba problémem a na výsledcích sítě prakticky není k rozeznání. Optimální počet průchodů datovou sadou se na základě výsledků experimentu zdá být 5.

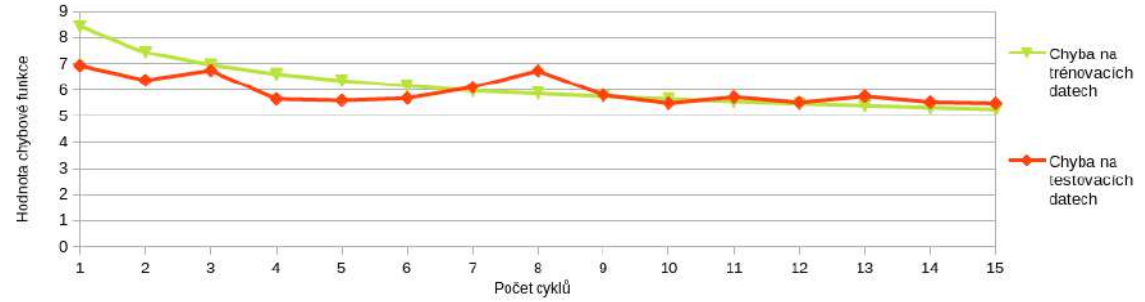
U sítě *R-Net* už existuje menší množství vzorků, které může dostat na vstupu, neboť ty jsou již redukovány sítí *P-Net*. K učení této sítě bylo použito celkem 88 838 pozitivních a 57 570 negativních vzorků. Jako pozitivní vzorky byly brány ty boxy, které měly překryv s anotacemi alespoň 40%, jako negativní pak ty, které měly překryv menší než 20%. Architektura sítí je sice odlišná, ale už učení *P-Net* ukázalo, že trénování na několika desítkách průchodů trénovacími daty nepřináší výrazné zlepšení. Proto bylo pro *R-Net* zvoleno průchodů 15. Z grafu 8.2 lze vidět, že nejlepších výsledků na testovacích datech síť dosahovala po skončení desátého průchodu. Na trénovacích datech stejně jako v případě předchozí sítě celková chyba dle očekávání klesá bez ohledu na počet průchodů.

Síť *O-Net* měla při učení k dispozici vzorky zpracované předešlými sítěmi. Vzhledem k tomu, že *R-Net* produkuje velmi kvalitní výsledky, je problém zajistit negativní vzorky. Po zpracování všech dat touto sítí bylo tedy k dispozici celkem 24 866 vzorků, které měly překryv s anotacemi alespoň 50% a pouhých 31 vzorků, které měly překryv méně než 20%. Protože hlavním cílem této sítě je korekce parametrů bounging boxu, nepředstavuje tak malé procento negativních vzorků zásadní problém. Pokud by bylo produkováno mnoho



Obrázek 8.2: Graf zobrazující hodnotu chybové funkce při učení sítě *R-Net*. Oproti *P-Net* je menší počet vzorků, které dostává na svém vstupu. S menším množstvím dat dochází i rychleji k přeučení sítě a je proto zvolen výrazně menší počet průchodů datovou sadou oproti předchozí síti. Od desátého průchodu už navíc ani nedochází k vylepšování výsledků sítě.

falešně pozitivních detekcí, je možné síť doučit na více negativních vzorcích. Ze zmiňované sady byly opět dvě třetiny použity na trénování a třetina na testování. Primárním účelem této sítě už není eliminace falešně pozitivních detekcí, ale co nejpřesnější regrese bounding boxů. Proto datová sada obsahuje vysoké procento pozitivních vzorků, na kterých je možné regresi učit. Graf 8.3 znázorňuje průběh učení a hodnoty chybových funkcí na trénovacích i testovacích datech na celkem 15ti průchodech datovou sadou. Je zde vidět, že od čtvrtého průchodu začíná chyba na testovacích datech pozvolna stoupat. Protože je sada poměrně malá, není překvapivé, že poměrně rychle došlo k přeučení sítě.



Obrázek 8.3: Graf zobrazující hodnotu chybové funkce při učení sítě *O-Net*. Je možné vidět, že chyba sítě na testovacích datech stabilně klesá do čtvrtého průchodu. Pak už pravděpodobně došlo k přeučení sítě a chyba dále stoupá. Po skončení osmého průchodu se opět trochu snižuje, ale stále je vyšší než po zmiňovaném čtvrtém průchodu.

Pro správné fungování celé kaskády sítí je nejdůležitější správné nastavení sítě *P-Net*, protože pokud ona značku nedetekuje, už se dále nikdy neobjeví. Síť zpracovává největší objem dat, z čehož plyne, že je nejpomalejší z nich. Pro rychlost zpracování je kritické nastavení parametrů pyramidy, která se má ze vstupního snímku udělat. Pyramida je nastavitelná v konfiguračních souborech nejmenší a největší šířkou delší strany snímku a krokem, který se použije při zmenšování. Zpracování Full HD snímku s krokem 1.3, nejmenší velikosti delší strany snímku 50px a největší velikosti delší strany 1000px trvá přibližně 18s (CPU Intel(R) Core(TM) i7-4500U CPU @ 1.80GHz). Při zvětšení delší strany na 1800px už trvá zpraco-

Překryv s anotacemi alespoň 40%

Pyramida, jistota	Pozitivní detekce	Pozitivní detekce [%]	Falešně pozitivní detekce	Celkem detekcí	Chybějících detekcí
1800, 0.90	245	24%	790	1035	2
1000, 0.90	389	44%	492	881	2
1800, 0.99	391	42%	528	919	2
1000, 0.99	376	58%	270	646	2

Překryv s anotacemi alespoň 50%

Pyramida, jistota	Pozitivní detekce	Pozitivní detekce [%]	Falešně pozitivní detekce	Celkem detekcí	Chybějících detekcí
1800, 0.90	143	14%	882	1025	23
1000, 0.90	220	25%	657	877	22
1800, 0.99	218	23.8%	695	913	23
1000, 0.99	215	33.6%	425	640	22

Překryv s anotacemi alespoň 60%

Pyramida, jistota	Pozitivní detekce	Pozitivní detekce [%]	Falešně pozitivní detekce	Celkem detekcí	Chybějících detekcí
1800, 0.90	69	10.7%	578	647	106
1000, 0.90	71	7.45%	811	952	107
1800, 0.99	73	8%	847	920	106
1000, 0.99	69	10.7%	578	647	108

Tabulka 8.1: Počet detekcí sítě *P-Net* v závislosti na míře jistoty, od které je výsledek sítě považován za pozitivní, a na velikosti největšího obrázku vstupní pyramidy. Každá z tabulek ukazuje hodnoty pro různou míru překrytí s anotovanými daty. Pro úplné vysvětlení tedy v případě první tabulky a jejího prvního řádku měl největší snímek v pyramidě velikost delší strany 1800px a v potaz byly brány boxy, které síť označila jako pozitivní s jistotou alespoň 90%. V tomto případě bylo celkem označeno 1 035 boxů jako pozitivních, z nichž 245 mělo překryv s anotacemi alespoň 40%, což je přibližně 24% detekovaných boxů. Zbylých 790 boxů mělo překryv menší. Ze všech 171 anotací pak dva boxy síť vůbec nedetekovala.

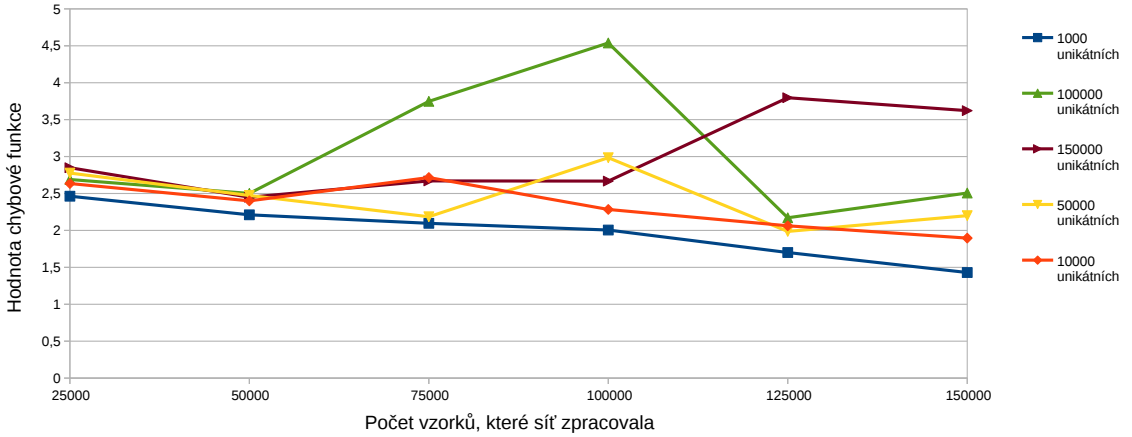
vání přibližně 43s. Velikost pyramidy má také vliv na množství výsledků, které síť označí jako pozitivní. V tabulce 8.1 je ukázáno porovnání počtu detekcí sítě v závislosti na velikosti pyramidy a jistotě sítě. Sloupec Pyramida v tabulce 8.1 označuje maximální velikost snímků v pyramidě, kterou musela síť zpracovat. Jistota pak udává míru jistoty sítě – tedy od jaké jistoty výše je box označen sítí *P-Net* jako pozitivní. Dále je počtem a procentuálně vyjádřen počet správných detekcí sítě a počet falešných detekcí sítě. Daná čísla byla získána na validační sadě obsahující 100 snímků s ručně anotovanými 171 registračními značkami. Je pochopitelné, že čím vyšší míra překrytí je požadována, tím méně pozitivních detekcí síť produkuje. Při překrytí alespoň 60% už je míra pozitivních detekcí nízká a vzhledem k faktu, že na snímcích bylo 171 registračních značek, je tedy označena sotva polovina. Je proto výhodnější nastavit míru překrytí menší a mít více falešně pozitivních detekcí.

Kromě velikosti pyramidy má vliv na počet výsledků této sítě i práh ve funkci provádějící potlačení nemaxim – *Non Maxima Suppression*. Čím je tento práh nižší, tím méně boxů zůstává po jejím provedení ve výsledcích. Stává se ovšem, že při příliš nízkém prahu jsou eliminovány boxy, které sice mají menší překrytí s ostatními detekovanými boxy, ale lépe se hodí pro korekce dalšími sítěmi. Pro síť *P-Net* je tento práh nastaven na hodnotu 0.7, což je sice vyšší hodnota, která má za následek více boxů, ale to je, jak už bylo řečeno, u této sítě žádoucí. Další dvě sítě mají tento práh shodně nastaven ve výchozím stavu na 0.5.

Byly také prováděny pokusy s velikostmi datových sad pro učení každé jednotlivé sítě zvlášť. Pro síť *P-Net* bylo vybráno po řadě 1000, 10 000, 50 000, 100 000 a 150 000 unikátních vzorků dat a bylo nad daným počtem vzorků provedeno trénování. V každém cyklu trénování bylo určeno, že síť musí zpracovat 150 000 vzorků. To znamená, že při velikosti dávky (*batch size*) 5 proběhlo 30 000 úprav koeficientů sítě (*back propagation*). Při 1000 unikátních vzorcích byly tyto zpracovány sítí 150krát dokola, při 10 000 unikátních vzorcích 15krát atd. V grafu 8.4 lze vidět průběhy chybových funkcí při učení pro každý počet

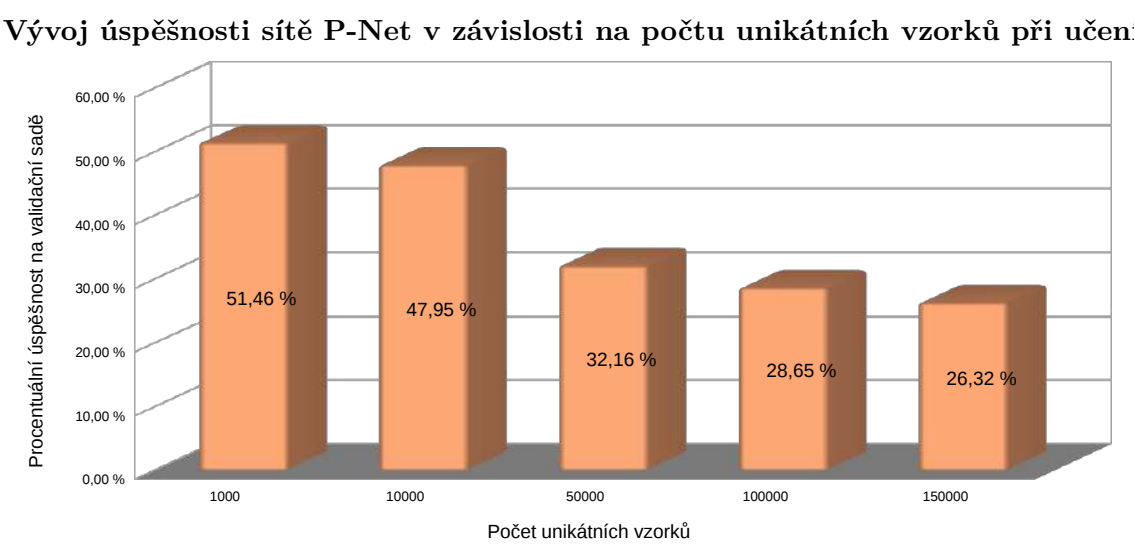
unikátních vzorků zvlášť. Chyba byla vypočítávána každých 25 000 vzorků, které síť zpracovala, na testovací datové sadě čítající 67 025 unikátních vzorků. Je zde možné vidět, že čím méně unikátních vzorků a tedy více průchodů stejnými daty, tím stabilněji funkce klesá. Je to dáno tím, že síť data už několikrát zpracovávala a stále lépe se jim přizpůsobuje. Naproti tomu chyba na velkých počtech unikátních vzorků je velmi nestabilní a také mnohem větší. Je to proto, že síť se stále ještě přizpůsobuje novým variabilním datům, která ještě nezpracovala. Vzhledem k tomu, že ani v jednom případě chyba nestoupá, lze říci, že síť ani v jenom případě ještě není přeučená.

Vývoj hodnoty chybové funkce sítě P-Net pro různé počty unikátních vzorků na vstupu

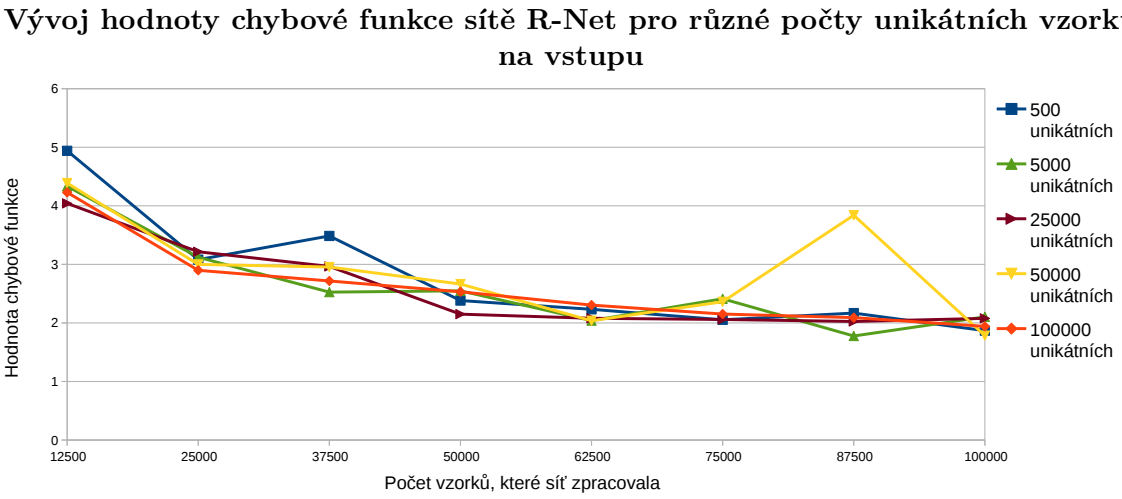


Obrázek 8.4: Graf znázorňuje průběhy chybových funkcí při trénovací fázi. Zvášť je zde průběh pro učení na 1000, 10 000, 50 000, 100 000 a 150 000 unikátních vzorcích, přičemž ve všech případech byl celkový počet vzorků, které síť zpracovala, roven 150 000. V případě 1000 unikátních vzorků tak síť udělala 150 průchodů všemi daty, při 10 000 unikátních vzorků bylo provedeno 15 průchodů atd. Je zde vidět, že čím více průchodů stejnými daty síť provede, tím stabilnější průběh funkce je. Naopak při prvním průchodu variabilními daty je průběh velmi nestabilní.

V grafu 8.5 lze vidět vliv variability unikátních dat na výsledky sítě při predikci. Zatímco při malém množství unikátních dat při učení byla sice chyba co do velikosti malá, tak výsledné detekce sítě jsou velmi špatné. Procentuální úspěšnost, kterou lze v grafu vidět, byla počítána na validačních datech, čítajících 171 ručně anotovaných registračních značek. Protože primárním účelem sítě *P-Net* je detekovat potenciální místa se značkami, je nejkritičtějším parametrem počet značek, které síť nebyla schopná detekovat. Právě procentuální podíl nenalezených značek z celkového počtu 171 je znázorněn v grafu. V případě 1000 a 10 000 vzorků je prakticky polovina značek nedetekována. Čím více unikátních dat na vstupu je, tím se procento snižuje a to i přesto, že například v případě se 150 000 vzorky síť data zpracovala pouze jednou. Při větším počtu průchodů by toto procento dále klesalo například až na hodnotu 1.17%, což jsou dva vynechané vzorky (této hodnoty síť dosáhla po zmiňovaných 40 průchodech všemi daty).



Obrázek 8.5: Procentuální podíl značek, které síť *P-Net* nebyla schopna detekovat v závislosti na počtu unikátních vzorků, na kterých byla trénována. Výsledná procenta byla počítána jako podíl nedetekovaných značek vůči celkovému počtu 171 anotovaných. Je zde možné vidět, že při nízké variabilitě dat (v případě prvního sloupe pouze 1000 unikátních vzorků) při fázi trénování je potom celková úspěšnost sítě nízká. S vyšší variabilitou klesá procento nedetekovaných značek a stoupá tak úspěšnost sítě. Po zpracování všech dostupných vzorků a 40 průchody nimi je procento nedetekovaných značek na stejné validační sadě 1.17%.



Obrázek 8.6: Graf znázorňuje průběhy chybových funkcí sítě *R-Net* při trénovací fázi. Zvášť je zde průběh pro sítě, učení na 500, 5000, 25 000, 50 000 a 100 000 unikátními vzorky na vstupu, přičemž ve všech případech byl celkový počet vzorků, které síť zpracovala roven 100 000. V případě 500 unikátních vzorků tak síť udělala 200 průchodů všemi daty, při 10 000 unikátních vzorků bylo provedeno 15 průchodů atd. Oproti předchozí síti je rozdíl v tom, že nejméně stabilní není průběh při největším počtu vzorků, ale při druhém největším, tedy 50 000. Chyba se zvyšuje v polovině druhého průchodu trénovací množinou dat.

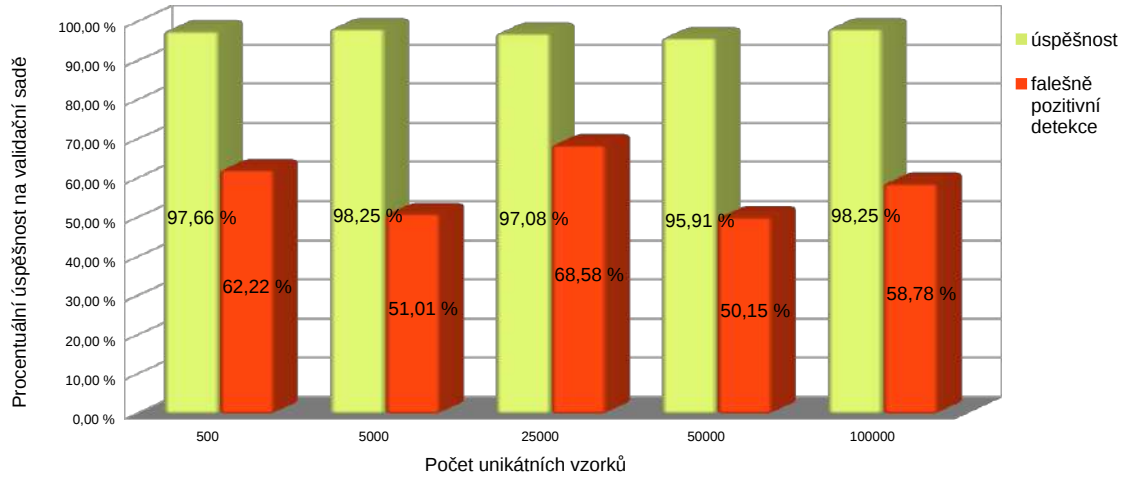
Pro síť *R-Net* bylo kvůli menšímu počtu vzorků zvoleno 500, 5 000, 25 000, 50 000 a 100 000 unikátních vzorků při maximálním počtu 100 000, které mohly být sítí zpracovány. V případě s 5 000 tedy síť tyto vzorky zpracovala 20krát, v případě se 100 000 pouze jednou. Testovací sada použitá pro znázornění průběhu chyby čítala 49 165 vzorků, přičemž chyba byla počítána každých 12 500 zpracovaných vzorků. Graf 8.6 znázorňuje vývoj této chyby pro každý počet unikátních vzorků zvlášť (stejně jako tomu bylo v grafu předchozí sítě). Zde je oproti předchozí síti vidět rozdíl v tom, že nejméně stabilní není průběh při největším počtu vzorků, ale při druhém největším, tedy 50 000. Chyba se zvyšuje v polovině druhého průchodu trénovací množinou dat. Vysvětlení by mohla být dvě. I zde by se to dalo vysvětlit tím, že síť stále ještě nemá všechna svá trénovací data naučená dostatečně. Druhou možností je, že data se při každém novém průchodu sítí zamíchají. Mohlo se náhodně stát, že data, která byla v první polovině tohoto druhého průchodu se dost lišila od testovací sady a tudíž se nastavily koeficienty podle nich a na testovací sadě se to projevilo jako vysoká chyba.

Graf 8.7 pak zachycuje procentuální úspěšnost sítě *R-Net* naučené na různých počtech unikátních dat na vstupu při fázi učení. Je zde zvlášť zobrazena úspěšnost sítě (zelený sloupec) – tedy kolik procent anotovaných registračních značek byla síť schopna detekovat. Úspěšnost 96 - 98% odpovídá tomu, že v nejhorším případě síť nedetekovala 7, v nejlepším pak pouze 3 značky. Druhý (oranžový) sloupec pak udává poměr falešně pozitivních detekcí z celkového počtu detekcí. Při 50 000 unikátních vzorků bylo sice procento falešně pozitivních nejmenší, ale také byl i největší počet nedetekovaných značek. Zde trochu překvapivě nejlépe vypadají sloupce při 5 000 vzorcích, protože mají malé procento falešně pozitivních detekcí a velké procento úspěšných detekcí. To není standardní chování, které by se od sítě očekávalo, naopak bylo předpokládané, že čím větší variabilita, tím lepší schopnosti detekce. Neznamená to ovšem, že při největší variabilitě jsou výsledky špatné, jen síť v tomto případě produkuje více falešně pozitivních detekcí. V tomto případě to není až takovým problémem, neboť další síť by tyto detekce měla dále eliminovat. Plně naučená síť produkovala na validační sadě přibližně 37% falešně pozitivních detekcí a nebyla schopna detekovat pouze jednu značku.

Stejné pokusy jako v předchozích dvou případech byly provedeny i se sítí *O-Net*. Zde bylo kvůli omezenému počtu dat zvoleno 101, 1000, 5000, 10 000 a 15 000 unikátních vzorků a maximální počet vzorků pro průchod 20 000. Což znamená, že síť provedla 4 000 přepočtů koeficientů. V grafu 8.8 lze vidět vývoj chyby pro jednotlivé unikátní vzorky. Chyba byla počítána vždy po dosažení 2 500 zpracovaných vzorků na testovacích datech, které čítaly vzorků 8 835. Je možné pozorovat stejnou vlastnost jako měly předchozí dvě sítě, kterou je mnohem plynulejší klesající průběh pro menší počet unikátních vzorků. Je tomu tak opět z toho důvodu, že síť se stále ještě velmi přizpůsobuje každému novému vzorku.

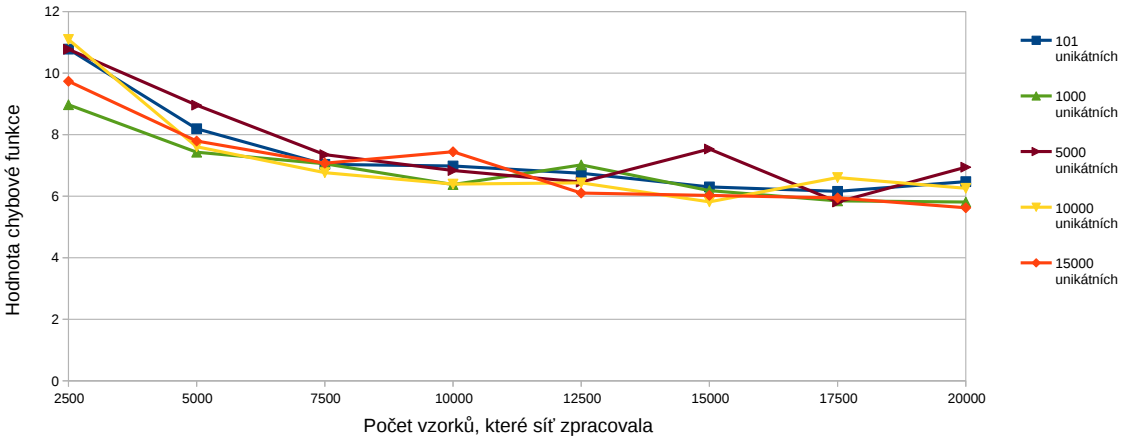
Graf 8.9 zachycuje úspěšnost sítě *O-Net* na validační sadě čítající 171 anotovaných značek. Zvlášť je znázorněno množství falešně pozitivních detekcí (červený sloupec) a úspěšnost – tedy kolik registračních značek síť detekovala. Ani v jednom případě síť nebyla schopna detekovat 10–11 (94.15%–93.57% úspěšnost) boxů a přibližně 17–22% všech detekcí byly falešně pozitivní. Pro bližší představu – plně naučená síť nebyla schopna detekovat 5 boxů, což odpovídá více než 97% úspěšnosti.

Vývoj úspěšnosti sítě R-Net v závislosti na počtu unikátních vzorků při učení

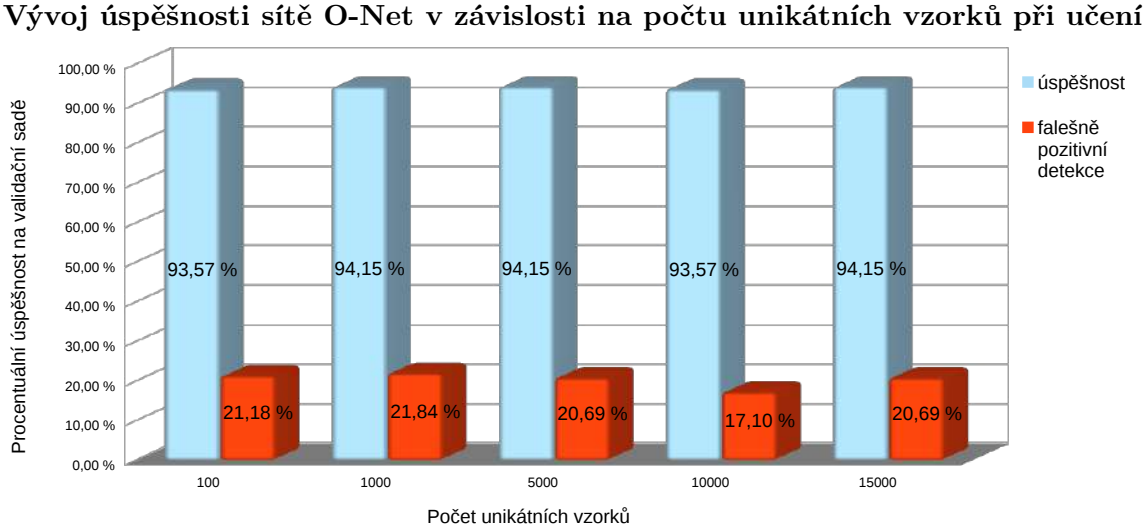


Obrázek 8.7: Graf zachycuje v zeleném sloupci procentuální podíl značek, které síť *R-Net* nebyla schopna detekovat v závislosti na počtu unikátních vzorků, na kterých byla trénována. V oranžovém sloupci je pak znázorněn procentuální podíl falešně pozitivních detekcí z celkového počtu detekcí. Zde trochu překvapivě nejlépe vypadají sloupce při 5 000 vzorcích, protože mají malé procento falešně pozitivních detekcí a velké procento úspěšných detekcí. Při největší variabilitě lehce stoupl počet falešně pozitivních detekcí. Pro představu – plně naučená síť produkovala na stejné sadě přibližně 37% falešně pozitivních detekcí a nebyla schopna detekovat pouze jednu značku.

Vývoj hodnoty chybové funkce sítě O-Net pro různé počty unikátních vzorků na vstupu



Obrázek 8.8: Vývoj chyby sítě *O-Net* v závislosti na počtu unikátních vzorků, které síť dostala v rámci trénovacích dat. Barevně odlišeny jsou průběhy učení na 101, 1 000, 5 000, 10 000 a 15 000 unikátními vzorky na vstupu, přičemž ve všech případech byl celkový počet vzorků, které síť zpracovala, roven 20 000. V případě s 1000 unikátními vzorky tak síť udělala 20 průchodů všemi daty, při 10 000 unikátních vzorků byly provedeno 2 průchody atd. Stejně jako v případě předchozích dvou sítí je průběh tím plynulejší, čím méně jsou data variabilní.



Obrázek 8.9: V grafu je v červeném sloupci znázorněno procento falešně pozitivních z celkového počtu detekcí, které síť *O-Net* predikovala na validační sadě dat. V modrém sloupci je znázorněna úspěšnost – tedy kolik procent ze všech pozitivních vzorků, které validační sada obsahovala, byla síť *O-Net* schopna detekovat. Vše v závislosti na počtu unikátních vzorků, které byly na vstupu při fázi učení. Ani v jednom případě síť nebyla schopna detekovat 10 - 11 boxů a přibližně 17 - 22% všech detekcí byly falešně pozitivní. Pro srovnání plně naučená síť nebyla schopna detekovat 5 boxů.

Na obrázcích 8.10 lze vidět, že sítě zvládají detekovat registrační značky i v případech, kdy není značka vidět úplně celá, ale část je překrytá. Rovněž je možné vidět, že na místě překryvu nezáleží. Ani *LPR API* pro rozpoznání textu nemá v těchto případech problémy s rozpoznáváním.

Obrázek 8.10: Ukázky zajímavých detekcí sítí. I přesto, že drobné části značek jsou překryté různými objekty, sítím to nepřekáží a jsou schopné i tak značky detekovat. Pokud chybí jen drobná část textu, ani *LPR API* nemá problém správně detekovat text.

Původní datová sada *ZoomLPDataset* neobsahovala žádné snímky značek v noci a zároveň z blízka či úhlu, takže sítě měly problém s takovýmito detekcemi a na snímcích většinou značku nenacházely. Ručně jsem proto anotovala celkem 164 registračních značek, které tyto podmínky reprezentovaly a sítě s nimi doučila. Byť se jednalo o malé množství vzorků, sítě zvládaly tyto detekce po doučení mnohem lépe. Podobně na tom byly snímky točené kame-

rou stojící hned vedle silnice, po které auta projížděla. I zde měly sítě s detekcemi problém a bylo tedy nutné na nově anotovaných datech sítě doučit. V tomto případě jsem anotovala 876 nových registračních značek.

Obrázek 8.11: Ukázky detekcí, se kterými měly sítě problémy a byly proto cíleně vylepšovány. Jednalo se hlavně o snímky, kde jsou projíždějící auta blízko kameře, mezi kamerou a značkou je větší úhel a také snímky získávané za tmy. V červeném rámečku je ukázka detekce před doučením, v modrém pak po něm.

8.2 Shrnutí výsledků manuální části

Ručně jsem provedla anotaci celkem 3 453 snímků, které byly určeny k manuální korekci. Z nich bylo nakonec získáno celkem 2 348 snímků obsahujících 3 557 registračních značek. Ukázky hotových anotací po provedení manuálních korekcí je možné vidět na obrázku 8.12.

Obrázek 8.12: Ukázky hotových anotací graficky znázorněných pomocí nástroje Plate Tagger.

Celkem 13 276 snímků s celkem 20 014 registračními značkami predikovanými automatickou částí bylo určeno k ruční korekci anotátorům. Z nich 1 359 bylo prozatím ručně zpracováno. Těchto 1 359 snímků obsahuje 2 317 anotovaných registračních značek. Zpracování tohoto množství snímků trvalo přibližně deset hodin. Zpracování dalších registračních značek stále probíhá. Rychlost zpracování velmi záleží na rychlosti anotátora, kvalitě automatických anotací a počtu snímků, které opravdu obsahují registrační značku. Při průměrné rychlosti anotátora 135 snímků za hodinu (experimentálně zjištěno) je pro ruční anotaci potřeba přibližně 98 hodin ruční práce.

8.3 Návrhy na zlepšení

Vzhledem k tomu, že automatickou část zpracovávají neuronové sítě, je bezpochyby možné je naučit ještě lépe. Největší vzorek dat, který byl k dispozici, představovala vozidla, která byla natáčena zepředu, z dálky a z výšky bez jakéhokoli bočního úhlu. I z výsledků sítí lze vidět, že tyto případy zvládají označit velice dobře. Problém však nastal u snímků s naopak velkým bočním úhlem mezi kamerou a registrační značkou, či snímky za tmy nebo obecně špatných světelných podmínek. Pokud by tedy bylo k dispozici větší množství anotovaných snímků za těchto odlišných podmínek, mohla by výrazně klesnout míra špatných či nepřesných detekcí. S tím se pochopitelně váže i míra ručních korekcí, která by mohla výrazně klesnout.

Rovněž by bylo vhodné zapojit například i automatickou detekci státu, ze kterého registrační značka pochází. Pro specifický formát registrační značky s modrým pruhem a zkratkou státu na levé straně by například mohla být využita rovněž neuronová síť. Obecně by však tento úkol mohl být velmi komplikovaný, protože typů registračních značek je mnoho a bylo by nutné mít od každé z nich mnoho vzorků.

Automatická část detekce by šla dále vylepšit podporou pro zpracování grafickou kartou (*GPU*) pomocí knihovny *CUDA*, což je přímo podporováno frameworkem PyTorch. Doplnění automatické části o *REST API* by umožnilo lepší dávkové zpracování a snadné napojení na další systémy, které by automatickou detekci mohly využívat.

Nástroje pro ruční zpracování by šly vylepšit napojením na centrální sklad vzorků, odkud by se mohly automaticky stahovat nezpracovaná data a zpět by se mohly nahrávat hotové anotace.

Kapitola 9

Závěr

Diplomová práce shrnovala problematiku a proces vytvoření datové sady obsahující registrační značky vozidel vhodné pro učení neuronových sítí. V první části byly shrnuty důležité poznatky o konvolučních neuronových sítích zaměřených na zpracování obrazu. Tyto poznatky obsahovaly především informace o procesu učení a s ním spojených chybových a aktivačních funkcích. Zvláštní část byla věnována vlastnostem datových sad vhodných pro učení těchto sítí. Blíže byla také představena síť známá jako *Multi-task Convolutional Neural Network – MTCNN*, která byla použita jako základ pro praktickou část práce.

Další část pak byla věnována konkrétním algoritmům, které jsou vhodné k detekci a rozpoznání registračních značek. Zde byly zmíněny dva možné přístupy. Prvním z nich jsou metody založené na segmentaci obrazu, druhým je pak využití (hlubokých) neuronových sítí. Představena byla rovněž i existující kompletní řešení zabývající se touto problematikou.

Značná část práce potom byla věnována podrobné studii existujících datových sad obsahujících registrační značky. Byly zde uvedeny ukázky a informace o čtyřech sadách obsahujících anotované registrační značky. Sem patří datové sady *UFPR-ALPR*, *Dataturks*, *MLBLR* a *SSIG-ALPR*. Další zmíněné – *Projekt LPDRAS*, *Medialab LPR*, enCaltech Cars, *AOLP Database*, *Olav's LP Pictures* a *PlatesMania* obsahují pouze snímky bez anotací.

V rámci této práce byla za pomoci frameworku PyTorch a programovacího jazyka Python3 vytvořena poloautomaticky zpracovaná datová sada obsahující snímky registračních značek z Brna a blízkého okolí. Celý proces vytvoření sady zahrnoval získání vstupních videí na různé druhy záznamových zařízení. Z těchto vstupních videí byly s použitím knihovny OpenCV vytvořeny snímky, které byly vstupem pro neuronové sítě mající za úkol detekci značek.

Řešení pro detekci vycházelo ze zmiňované sítě *MTCNN*. Obsahuje tedy kaskádu tří konvolučních sítí zpracovávajících vstup tak, že výstup první z nich je vstupem do další v pořadí atd. První ze sítí je plně konvoluční a pracuje s výřezy snímků o velikosti 21×7 pixelů a jejím primárním cílem je omezení vstupu na pouze ty části, které by s předem definovanou mírou jistoty mohly obsahovat registrační značky. Výstupem je množina výřezů původního vstupního snímku s přibližnými informacemi o tom, kde se nachází registrační značka a její parametry – souřadnice levého horního rohu, výška a šířka. Další síť v pořadí pracuje s dvakrát většími výřezy a je určena na eliminaci falešných detekcí předchozí sítě a co nejlepší regresi tzv. *bounding boxu*, tedy boxu, který co nejpřesněji ohraničuje registrační značku. Výstupem této sítě jsou už přesnější odhady souřadnic každého z rohů registrační značky. Poslední síť v pořadí pak pracuje s výřezy o velikosti 84×28 pixelů a definitivně eliminuje případné falešné detekce a určuje finální parametry bounging boxu do podoby souřadnic jeho čtyř rohů. Učení sítí probíhalo na datové sadě pořízené přímo FIT VUT v Brně. Další

doučení případů, které tato sada neobsahovala, pak probíhalo na ručně anotovaných datech získaných v rámci této práce. Neuronové sítě jsou určeny k detekci registračních značek, nikoli však k rozpoznávání textu, který obsahují. Pro tyto účely bylo použito *LPR API – Licence Plate Recognition Application Programming Interface* vytvořené a spravované rovněž FIT VUT v Brně. Vzhledem k tomu, že vytvořené řešení je do značné míry konfigurovatelné, je věnována samostatná kapitola podrobnému popisu programového řešení včetně popisu všech nastavitelných voleb a výstupů řešení.

Protože automatické zpracování nefunguje se stoprocentní spolehlivostí, je nutné případné chyby ve zpracování ručně zkontrolovat a opravit. Jako grafické uživatelské rozhraní pro manuální korekci dat automaticky zpracovaných neuronovými sítěmi byla použita volně dostupná aplikace Plate Tagger společnosti OpenALPR. Ta byla pro potřeby práce mírně poupravena.

Získáno bylo celkem téměř 17 000 snímků, které byly určeny k dalšímu ručnímu zpracování. Vzhledem k časové náročnosti byly přibližně čtyři pětiny snímků předány ke zpracování dalším anotátorům. Za účelem zjednodušení přípravy pracovního prostředí byl vytvořen virtuální stroj obsahující naučené neuronové sítě, nástroj Plate Tagger a další pomocné skripty usnadňující aktualizace a práci s daty pro anotaci. Prozatím bylo ručně zpracováno přibližně 5 000 snímků obsahujících přes 6 000 anotovaných registračních značek. Výsledná datová sada se skládá z obrazových dat ve formátu *JPEG* a z textových anotací ve formátu *YAML* obsahujících informace o snímku samotném (název, výška, šířka), rozpoznaném textu a jistotě, s jakou byl rozpoznán. Dále pak souřadnice bounding boxu, jistotu, s jakou byla značka detekována, a kód země, ze které registrační značka pochází.

Literatura

[1] California Institute of Technology: *Computational Vision Archive*. March 17, 2005, [Online; navštíveno 16.12.2018].
URL http://www.vision.caltech.edu/archive.html

[2] Chen, R.; Luo, Y.: An Improved License Plate Location Method Based On Edge Detection. *Physics Procedia*, ročník 24, 2012: s. 1350 – 1356, ISSN 1875-3892, doi:j.phpro.2012.02.201, international Conference on Applied Physics and Industrial Engineering 2012.

[3] Girshick, R.; Donahue, J.; Darrell, T.; aj.: Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 11 2013, doi:10.1109/CVPR.2014.81.

[4] Gonçalves, G. R.; Diniz, M. A.; Laroca, R.; aj.: *Real-time Automatic License Plate Recognition Through Deep Multi-Task Networks*. In *Conference on Graphic, Patterns and Images (SIBGRAPI)*, 2018, s. 1–8.

[5] Goodfellow, I.; Bengio, Y.; Courville, A.: *Deep Learning*. MIT Press, 2016, ISBN 978-0-262-03561-3.

[6] Hsu, G.; Chen, J.; Chung, Y.: Application-Oriented License Plate Recognition. *IEEE Transactions on Vehicular Technology*, ročník 62, č. 2, Feb 2013: s. 552–561, ISSN 0018-9545, doi:10.1109/TVT.2012.2226218.

[7] Kalafatić, Z.: *License Plate Detection, Recognition and Automated Storage*. November 10, 2003, [Online; navštíveno 16.12.2018].
URL http://www.zemris.fer.hr/projects/LicensePlates/

[8] Khalifa, Othman and Khan, Sheroz and Islam, Md and Suleiman, Ahmad: Malaysian Vehicle License Plate Recognition. *Int. Arab J. Inf. Technol.*, ročník 4, 01 2007: s. 359–364.

[9] Kocer, H. E.; Cevik, K. K.: Artificial neural networks based vehicle license plate recognition. *Procedia Computer Science*, ročník 3, 2011: s. 1033 – 1037, ISSN 1877-0509, doi:j.procs.2010.12.169, world Conference on Information Technology.

[10] Laroca, R.; Severo, E.; Zanlorensi, L.; aj.: A Robust Real-Time Automatic License Plate Recognition Based on the YOLO Detector. 07 2018, doi:10.1109/IJCNN.2018.8489629.

[11] Laroca, R.; Severo, E.; Zanlorensi, L. A.; aj.: *A Robust Real-Time Automatic License Plate Recognition Based on the YOLO Detector*. In *2018 International Joint Conference on Neural Networks (IJCNN)*, July 2018, ISSN 2161-4407, s. 1–10, doi:10.1109/IJCNN.2018.8489629.

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

[13] Medialab: *Medialab LPR database*. [Online; navštíveno 12.12.2018].
URL http://www.medialab.ntua.gr/research/LPRdatabase.html

[14] Mishra, D.: *Indian Number Plates*. July 07, 2018, [Online; navštíveno 16.12.2018].
URL https://dataturks.com/projects/devika.mishra/Indian_Number_plates

[15] MLBLR: *Datasets*. [Online; navštíveno 12.12.2018].
URL https://mlblr.com/includes/dataset/index.html

[16] Patel, C.; Shah, D.; Patel, A.: Automatic Number Plate Recognition System (ANPR): A Survey. *International Journal of Computer Applications (IJCA)*, ročník 69, 05 2013: s. 21–33, doi:10.5120/11871-7665.

[17] Redmon, J.; Divvala, S.; Girshick, R.; aj.: You Only Look Once: Unified, Real-Time Object Detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, ročník 00, June 2016, ISSN 1063-6919, s. 779–788, doi:10.1109/CVPR.2016.91.

[18] Shaikhina, T.; Khovanova, N. A.: Handling limited datasets with neural networks in medical applications: A small-data approach. *Artificial Intelligence in Medicine*, ročník 75, 2017: s. 51 – 63, ISSN 0933-3657, doi:j.artmed.2016.12.003.

[19] Sharma, S.: Activation Functions: Neural Networks. September 6, 2017, [Online; navštíveno 03.01.2019].
URL https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

[20] Türkyılmaz, I.; Kaçan, K.: License Plate Recognition System Using Artificial Neural Networks. *ETRI Journal*, ročník 39, č. 2, 2017: s. 163–172, doi:10.4218/etrij.17.0115.0766.

[21] Zhang, K.; Zhang, Z.; Li, Z.; aj.: *Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks*. *IEEE Signal Processing Letters*, ročník 23, č. 10, Oct 2016: s. 1499–1503, ISSN 1070-9908, doi:10.1109/LSP.2016.2603342.

Příloha A

Obsah přiložené SD karty

- **finally_processed/** - obsahující ručně zpracované snímky včetně anotací
- **for_processing/** - obsahující prozatím pouze automaticky zpracované snímky včetně anotací
- **plate_detector.zip** - obsahující zdrojové soubory k neuronovým sítím
- **plate_tagger.zip** - obsahující zdrojové soubory k nástroji OpenALPR Plate Tagger
- **openalpr_tagger** - spustitelný nástroj OpenALPR Plate Tagger
- **doc/** - zdrojové soubory k tomuto textu
- **xkvapi12.pdf** - tento text
- **xkvapi12-video.mp4** - video prezentující tuto práci
- **xkvapi12-plakat.pdf** - plakát prezentující tuto práci
- **LP_database.zip** - obraz virtuálního stroje s připraveným prostředím pro detekci i ruční anotace (dostupný také na https://drive.google.com/open?id=1Ws79Lp9LPEJHMcMz_gMbV98ZB3q8clDS)
- **LP_database_návod.pdf** - dokumentace k virtuálnímu stroji a nástroji OpenALPR Plate Tagger sloužící pro anotátory (dostupná také na https://drive.google.com/open?id=1qZpVl8V1etI22VH7zAztudozApdbaAlDisbwhrINf_k)