TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Kamerový systém s automatickým rozpoznáváním

Bakalářská práce

Studijní program: B2646 Informační technologie
Studijní obor: Informační technologie

Autor práce: **Lukáš Zeman**
Vedoucí práce: Ing. Jana Kolaja Ehlerová, Ph.D.
Ústav nových technologií a aplikované informatiky

Liberec 2022

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Zadání bakalářské práce

Kamerový systém s automatickým rozpoznáváním

Jméno a příjmení: **Lukáš Zeman**
Osobní číslo: M18000101
Studijní program: B2646 Informační technologie
Studijní obor: Informační technologie
Zadávající katedra: Ústav nových technologií a aplikované informatiky
Akademický rok: **2021/2022**

Zásady pro vypracování:

1. Proveďte rešerši kamerových systémů s automatickým rozpoznáváním obrazu. Zaměřte se na rozpoznávání registračních značek a otevřené systémy.
2. Sestavte a nainstalujte kamerový systém sestávající se z minipočítače Raspberry Pi a kamery, zajistěte jeho SW zabezpečení.
3. Navrhněte, implementujte a otestujte aplikaci pro automatické snímání obrazu s rozpoznáváním značek případně čísel.
4. Otestujte aplikaci v ostrém provozu.
5. Zamyslete se nad návrhy pro zlepšení uživatelského prožitku, zejména z pohledu automatického běhu systému a jeho přístupnosti pro uživatele.

Rozsah grafických prací:	dle potřeby dokumentace
Rozsah pracovní zprávy:	30-40 stran
Forma zpracování práce:	tištěná/elektronická
Jazyk práce:	Čeština

Seznam odborné literatury:

[1] BEZDEK, James C. *Fuzzy models and algorithms for pattern recognition and image processing*. New York: Springer, 2005. The handbooks of fuzzy sets series. ISBN 0-387-24515-4.

[2] ŠONKA, Milan, Václav HLAVÁČ a Roger BOYLE. *Image processing, analysis, and machine vision*. 3rd ed. Toronto: Thomson Learning, 2008. International student edition. ISBN 0-495-24438-4.

Vedoucí práce:	Ing. Jana Kolaja Ehlerová, Ph.D. Ústav nových technologií a aplikované informatiky
Datum zadání práce:	12. října 2021
Předpokládaný termín odevzdání:	16. května 2022

L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

Ing. Josef Novák, Ph.D.
vedoucí ústavu

V Liberci dne 19. října 2021

Prohlášení

Prohlašuji, že svou bakalářskou práci jsem vypracoval samostatně jako původní dílo s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Jsem si vědom toho, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu Technické univerzity v Liberci.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti Technickou univerzitu v Liberci; v tomto případě má Technická univerzita v Liberci právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Současně čestně prohlašuji, že text elektronické podoby práce vložený do IS/STAG se shoduje s textem tištěné podoby práce.

Beru na vědomí, že má bakalářská práce bude zveřejněna Technickou univerzitou v Liberci v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů.

Jsem si vědom následků, které podle zákona o vysokých školách mohou vyplývat z porušení tohoto prohlášení.

16. května 2022 Lukáš Zeman

ABSTRAKT:

Tato bakalářská práce se zabývá tvorbou kamerového systému s automatickým rozpoznáváním objektů, konkrétně registračních značek vozidel a jejich následného čtení. V rámci bakalářské práce je nejdříve popsána rešerše a analýza systémů pro rozpoznávání objektů. Následně je zde popsán způsob instalace a zabezpečení systému Raspberry Pi, který je dále využit jako hardware pro testování systému. Posledním bodem je poté implementace algoritmů pro rozpoznávání objektů, jejich natrénování, testování a přenos kódu na Raspberry Pi. Praktickým výstupem práce je poté statistika z provedených testů systému v různých případech.

Klíčová slova: rozpoznávání objektů, strojové učení, hluboké učení, Raspberry Pi, OCR, YOLO, Faster R-CNN

ABSTRACT:

This bachelor thesis deals with the development of a camera system with automated recognition of objects, specifically vehicle license plates and their subsequent reading. The bachelor thesis first describes the search and analysis of systems for object detection. Then it describes how to install and secure Raspberry Pi system, which is then used as hardware for the system testing. The last point is then the implementation of algorithms for object detection, their training and testing and therefore moving that code to Raspberry Pi. The practical outcome of this thesis is then statistics from various test cases.

Keywords: object detection, machine learning, deep learning, Raspberry Pi, OCR, YOLO, Faster R-CNN

Poděkování:

Na tomto místě bych rád poděkoval paní Ing. Janě Kolaja Ehlerové Ph.D.

za připomínky, odborné rady a pomoc při zpracovávání bakalářské práce.

Obsah

Seznam zkratek

HTML	Hypertext Markup Language
RZ	Registrační značka
SSD	Single Shot MultiBox Detector
IoU	Intersection over Union
RoI	Region of Interest
OCR	Optical Character Recognition
YOLO	You Look Only Once
PDF	Portable Document Format
PNG	Portable Network Graphics
XML	Extensible Markup Language
CSV	Comma-separated values
JPEG	Joint Photographic Experts Group
TSV	Tab-Separated Values
R-CNN	Region-based Convolutional Neural Network

1 ÚVOD

Systémy s automatickým rozpoznáváním jsou již využívané v mnoha reálných situacích. Využívají se například na dálnicích pro rozpoznávání RZ vozidel, rozpoznávání osob na obraze apod. Cílem této práce bylo vytvořit podobný systém, který bude navíc možno přesouvat podle potřeby uživatele a také bude v menším a levnějším provedení. V tomto konkrétním případě se rozpoznávání zaměřovalo na RZ značky.

V rešerši a analýze možných řešení jsou popsány způsoby, kterými je možné rozpoznávat objekty v obraze. Konkrétně dva algoritmy hlubokého učení využívající neuronové sítě k rozpoznávání. Druhá část této rešerše je o algoritmech, které dokáží rozpoznat text z obrázku. Spojení těchto dvou algoritmů nám dá požadovaný výsledek.

Aby bylo možné dané algoritmy zprovoznit, je k dispozici databáze snímků RZ pro natrénování konkrétních modelů neuronových sítí. Tyto snímky jsou charakterizovány detailním záběrem na RZ a také anotací, která udává, kde se na daném snímku RZ nachází. V části o implementaci aplikace je popsán celkový postup při instalaci aplikace a systému Raspberry Pi, na kterém byl finální test realizován. Také jsou zde informace o jednotlivých úspěšnostech algoritmů a důvod, proč byl zvolen algoritmus YOLO.

Pro testování se využilo Raspberry Pi verze 4, veškerý kód byl napsaný v jazyce Python a využilo se kombinace YOLO algoritmu a EasyOCR. Tento systém byl poté podroben testům, při kterých určité testy měly horší podmínky pro snímání obrazu, aby bylo možné zjistit, jakým způsobem ovlivňují různé degradace obrazu samotný systém.

.

2 Rešerše

2.1 Detekování registračních značek

2.1.1 Registrační značka

Státní poznávací značka (zkratka SPZ, hovorově espézetka, též rozeznávací, evidenční, rejstříková nebo policejní značka; podle legislativní zkratky v českém zákoně také registrační značka, RZ) je jednoznačné písmeno-číselné označení motorového vozidla nebo jeho přívěsu či návěsu, zaregistrovaného v určitém státu. Tabulka s tímto označením, nejčastěji ve formě bílé obdélníkové destičky s černými písmeny a čísly, je povinně umístěna na každém motorovém vozidle podléhajícím registraci a na přípojných vozidlech k motorovým vozidlům. (1)

V každé zemi se RZ trochu liší a z toho důvodu musí být systém na detekci těchto značek velice robustní a záleží zde na mnoha faktorech. V České republice a Evropské Unii se používají podobné značky. Může se stát, že mají jiný vzor, ale barevná kombinace a znak Evropské Unie je na každé z nich. Např.

Obrázek 2.1-1 RZ Černá Hora (2)

Obrázek 2.1-2 RZ Finsko (2)

.

Obrázek 2.1-3 RZ Německo (2)

Registrační značky po celém zbytku světa mohou být velice odlišné, např.

Obrázek 2.1-4 RZ INDIANA (3)

2.1.2 Dnešní systémy

Jak již bylo uvedeno, RZ je jednoznačný identifikátor vozidla v provozu. Automatickým detekováním RZ se dá mnoho věcí dělat bez zásahu člověka. V nynější době jsou tyto systémy na automatické rozpoznávání poměrně časté. Využívají se na větších parkovištích, kde mohou automaticky pustit auto z parkoviště, pokud nepřesáhlo časový limit, nebo v pokročilejších systémech i navést auto na volné parkovací místo. Tyto systémy se také využívají na dálnicích, kde mohou být spojeny s dalšími systémy, například kontrolovat rychlost jízdy nebo kontrolovat, zda vozidla mají dálniční známky.

2.1.3 shrnutí

Tyto systémy jsou ovšem finančně náročné a cílem této práce je vytvořit podobný systém v menším provedení, který bude možné přenášet a využít na jakékoliv místě, díky instalaci na Raspberry Pi. Systém není tak přesný, ale úlohu plní stejně jako ostatní. Je velice pravděpodobné, že s omezenými prostředky náš systém nebude tak přesný jako několik let vyvíjené aplikace.

.

Z dodaných informací lze říct, že jakékoliv algoritmy, které pracují s barvami nebo kontrastem v obraze, zde nebude možné použít z důvodu rozmanitosti různých registračních značek. Je zde možné použít algoritmy pro zjišťování hran v obraze, jelikož tvar je pokaždé velice podobný. Také je zde možnost pro větší přesnost využít rozpoznávání objektů v obrazu a následné čtení textu z obrazu. Obě metody jsou podrobněji vysvětleny v následujících kapitolách.

2.2 Algoritmy a detekce objektů

Detekování objektů je jedna z technik počítačového vidění, využívaná pro detekci různých objektů v obraze. Algoritmy pro detekci objektů většinou využívají „machine learning“, tedy strojové učení, nebo „deep learning“, respektive metodu strojového učení využívající hluboké neuronové sítě, pro zobrazení smysluplných dat. Lidé jsou schopni při pohledu na obrázek nebo video rozpoznat objekty v daném obraze. Cílem detekce objektů je replikovat tento jev pomocí různých algoritmů. (4)

K realizaci detekce objektů se mohou používat různé techniky. Populární techniky využívají hluboké učení, tedy konvoluční neuronové sítě (CNN), patří mezi ně například R-CNN a YOLO v2. Automaticky se učí detekovat objekty v obraze. (5)

Mezi metody strojového učení patří například typy algoritmů, které v obraze rozpoznávají určité konkrétní vlastnosti (Features) daného obrazu, jako například histogram hran nebo skupiny pixelů. Tyto vlastnosti jsou poté předány do regresivního modelu, který predikuje umístění konkrétních objektů a slouží jako klasifikátor. (6)

2.2.1 Tradiční metody pro zpracování obrazu

V určitých případech je použití hlubokého učení zbytečné náročné a přehnané. V těch případech se dá využít tradičních metod pro zpracování obrazu, kde je použito méně řádků kódu a je zde větší přesnost. Hluboké učení využívá model, který se trénuje na specifických datech. Pokud tato data nejsou zpracována dobře, model je poté nepřesný. Na druhou stranu, tyto metody jako hledání barev nebo práce s pixely jsou jednoduché, nespecifikované pro přesný objekt a dají se použít v každém případě dané úlohy. (7)

Typickým příkladem pro zpracování obrazu je rozpoznání produktu, kde je zapotřebí rozeznat červenou krabici od zelené. Tato úloha by byla možná provést pomocí hlubokého učení, ale v tomto případě je to zbytečné. Bylo by zde zapotřebí vytvořit

.

nejdříve trénovací data, a poté natrénovat model, zatímco v případě tradičních metod by stačilo jednoduché rozpoznání barev. Na tomto příkladu je vidět, že hluboké učení má problém v generalizování problému. Zobecnění problému je zde náročné z důvodu potřeby vlastních trénovacích dat na jakýkoliv rozpoznávaný objekt. Tradiční metody nepotřebují data a tím, že se zaměřují například jen na barvu v obraze, dají se aplikovat na jakýkoliv obraz pomocí několika řádků kódu.

Jedna z dalších výhod tradičních metod je, že programátor má možnost, jakkoliv upravit kód pro lepší a přesnější výsledky, zatímco u hlubokého učení je náročné upravovat model.

2.2.2 Metody pro hluboké myšlení

Při použití těchto technik jsou zde dva klíčové přístupy uvedené v podkapitolách 2.2.3 a 2.2.4, pomocí nichž se je možné rozpoznávat objekty.

2.2.3 Vytvořit a natrénovat vlastní model pro detekci objektů

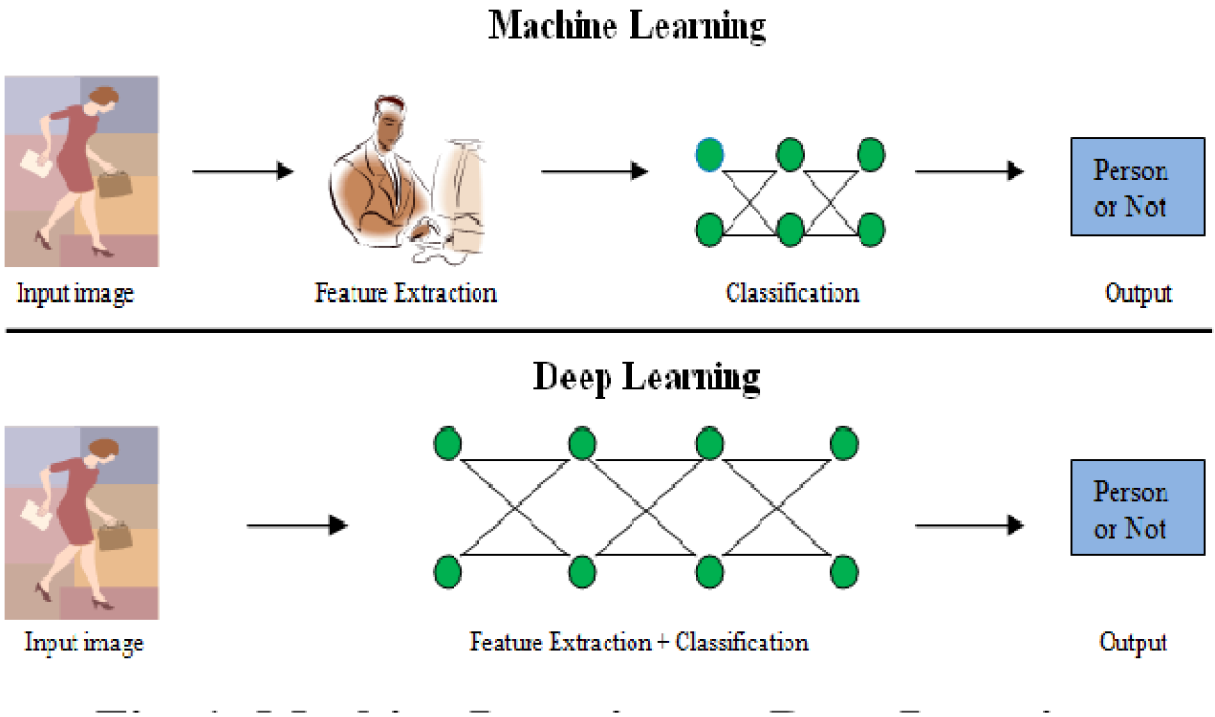
Aby bylo vůbec možné trénovat tento model, je potřeba navrhnout architekturu sítě, která se naučí vlastnosti detekovaných objektů. (4)

Pro trénování CNN je také potřeba sestavit velký soubor označených dat. Tedy mít k dispozici dostatek materiálu, na kterém se model může učit. Výsledky vlastního detektoru objektů mohou být často lepší. Ovšem je zde potřeba nastavit veškeré parametry ručně, což vyžaduje mnoho času a vyšší náročnost na tréninková data. Vlastním detektorem je myšleno naprogramovaný model od nuly včetně veškerých konfigurací a všech trénovacích procesů.

2.2.4 Použít předem připravený detektor objektů

Mnoho pracovních postupů detekce objektů využívajících hluboké učení využívá učení přenosu, což je přístup, který umožňuje začít s předem připravenou sítí a poté ji doladit pro konkrétní příklad v aplikaci. Tato metoda může poskytnout rychlejší výsledky, protože detektory objektů již byly natrénovány na tisících nebo dokonce milionech snímků.

.



Obrázek 2.2-1 Porovnání strojového učení (8)

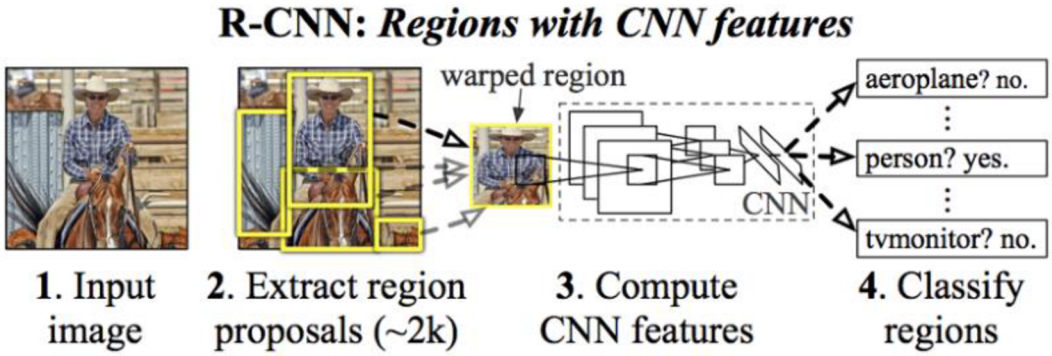
2.3 Typy algoritmů

V této kapitole jsou popsány nejpoužívanější algoritmy využívající hlubokých neuronových sítí. Dva z těch algoritmů jsou poté využity v praktické části. Mezi tyto algoritmy patří již zmíněný YOLO, R-CNN a také DETR a SSD. (9)

2.3.1 R-CNN

R-CNN je popsán v "Rich feature hierarchies for accurate object detection and semantic segmentation" z roku 2014. Jedná se o jeden z prvních úspěšných modelů pro detekci a lokalizaci objektů založených na neuronových sítích.

Detekce objektů se skládá ze dvou rozdílných úkolů. Jedním z nich je klasifikace a druhým lokalizace objektů. R-CNN znamená regionálně založené konvoluční neuronové sítě. Klíčový koncept tohoto algoritmu je navrhování potencionálních regionů. Navrhování se používá pro lokalizaci objektů v obraze.



Obrázek 2.3-1 R-CNN postup (8)

Na obrázku Obrázek 2.3-1 R-CNN postup lze vidět, že předtím, než obraz projde skrz neuronovou síť, potřebujeme vyextrahovat oblasti zájmu pomocí algoritmu jako je selektivní hledání. Poté, co jsou regiony nalezeny, je potřeba jim změnit velikost, tedy zabalit je a přeposlat do neuronové sítě.

Následně neuronová síť vytvoří kategorie pro dané oblasti zájmu, které jí byly odeslány. Navíc předpoví delty X a Y tvaru pro dané objekty.

2.3.2 Selektivní algoritmus

Selektivní hledání je algoritmus pro hledání regionů zájmu pro následnou lokalizaci objektů. Seskupuje skupiny pixelů na základě intenzity. V R-CNN je vždy navrhnuto 2000 oblastí. Mezi nimi musíme zvolit ty, které chceme rozpoznat. K tomu slouží IoU, na kterém je nastavený konkrétní parametr, podle něhož se dané oblasti označí.

Problémem tohoto algoritmu je, že vždy musí vygenerovat 2000 oblastí zájmu. Není na to žádným způsobem trénován, a proto ne vždy vygeneruje správné oblasti. Také natrénovat tento model je náročné, a i poté se nedá použít pro rozpoznávání objektů v reálném čase, jelikož trvá okolo 45 sekund na testovaný obraz.

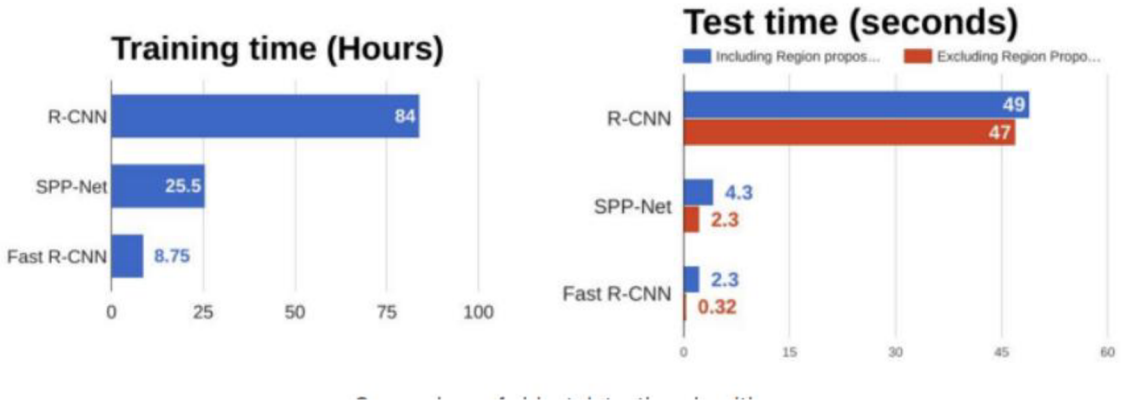
2.3.3 FAST R-CNN

Stejný autor, který vymyslel R-CNN, vymyslel i jeho nástupce Fast R-CNN, kde řeší hlavní problémy R-CNN. Průběh je podobný jako u původního algoritmu. Hlavní rozdíl je zde, že namísto navrhnutí regionu a odeslání daných regionů do neuronové sítě se nejprve odešle obraz do neuronové sítě. Poté se vytvoří konvoluční mapa prvků. Z této

.

mapy se identifikují regiony a následně se zabalí do čtverců a pomocí tzv. RoI se upraví jejich tvar na fixní. Dále se použije softmax vrstva pro určení třídy, do které daný objekt patří a také k nastavení offsetu jeho ohraničení. Výsledkem softmax vrstvy je vektor, který obsahuje pravděpodobnost jednotlivých detekovaných tříd pro daný objekt.

Důvodem, proč je tento algoritmus rychlejší, je fakt, že nemusíme do neuronové sítě posílat 2000 oblastí zájmu. Namísto toho zašleme celý obraz a až z něho se vytvoří dané oblasti.



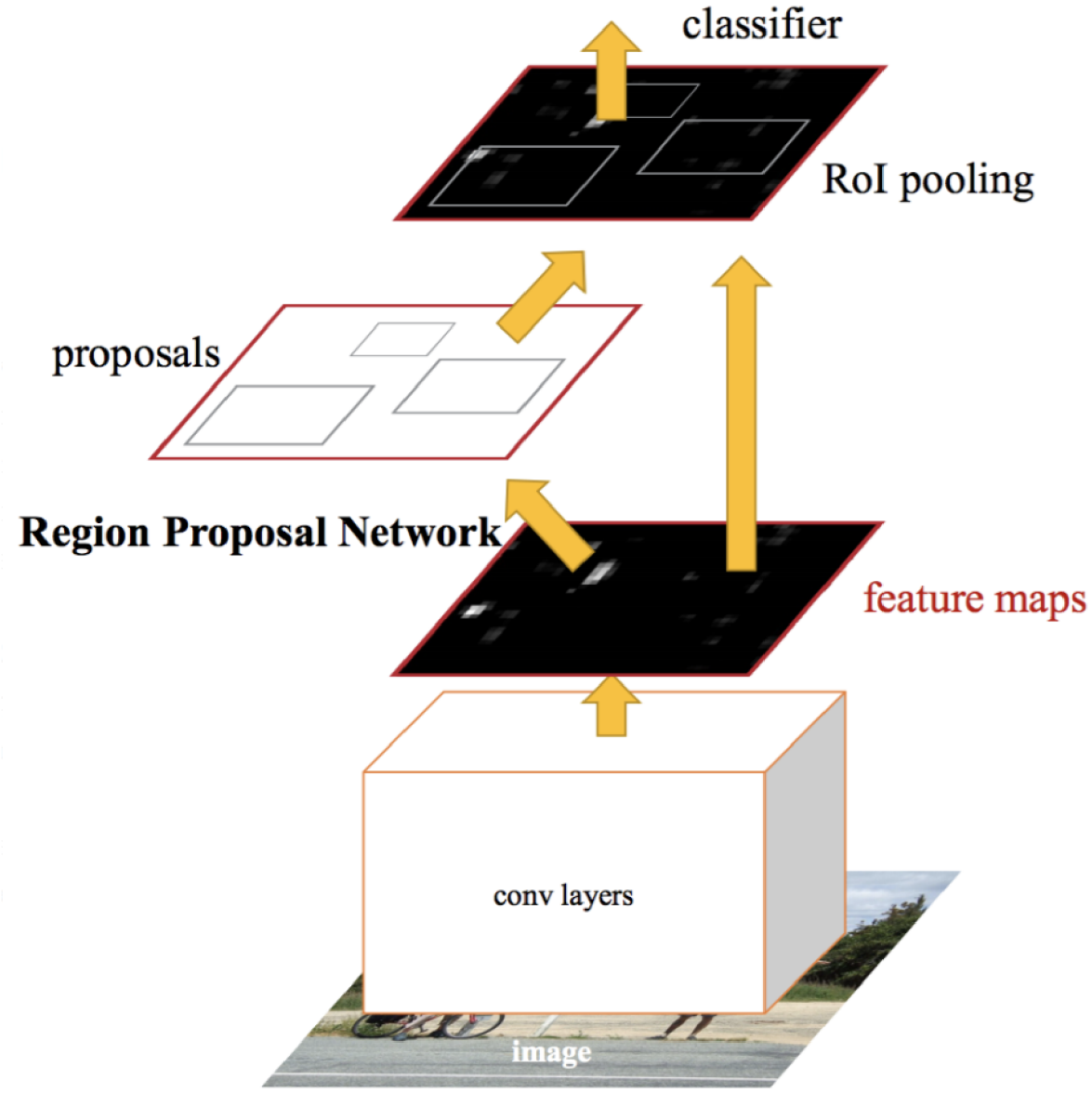
Obrázek 2.3-2 časy modelů (10)

Z grafu Obrázek 2.3-2 časy modelů lze vidět rozdíl rychlostí těchto modelů, jak v testovací, tak v trénovací části.

2.3.4 Faster R-CNN

Oba ze zmíněných algoritmů používají selektivní algoritmus pro nalezení regionů zájmu. Tento algoritmus je pomalý a trvá poměrně dlouho, tím pádem omezuje jinak rychlou práci dané neuronové sítě. Proto Shaoqing Ren et al. vymyslel detekci objektů, při které není zapotřebí selektivního algoritmu, ale samotné regiony zájmu se zde učí neuronová síť. (10)

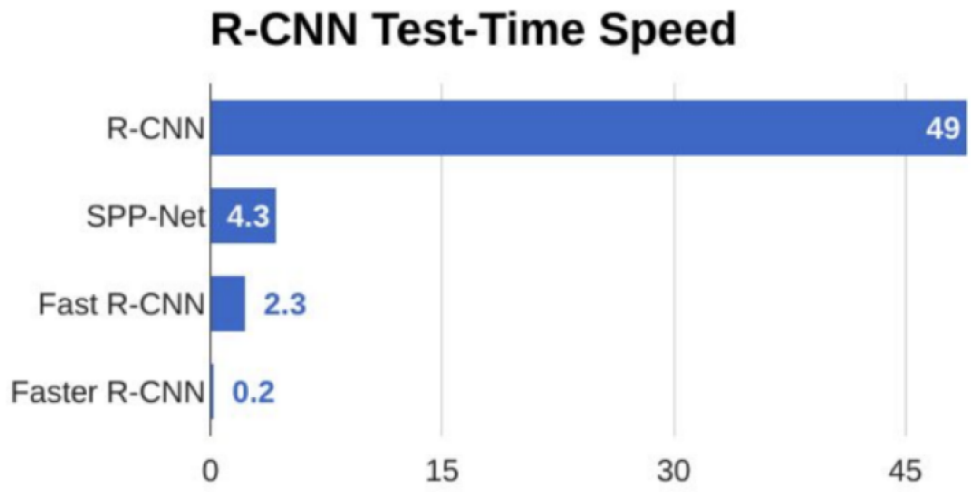
.



Obrázek 2.3-3 Faster R-CNN (10)

Stejně jako u FAST R-CNN je obraz poslán jako vstup pro neuronovou síť. Vytvoří se mapa oblastí zájmu, která je následně zaslána do neurální sítě, tato síť navrhne a předpoví regiony. Tyto regiony jsou poté opět upraveny pomocí RoI vrstvy a následně klasifikovány. Jako poslední se určí offset jejich ohraničení, to jsou souřadnice, mezi kterými se daný objekt nachází. V grafuObrázek 2.3-4 lze vidět porovnané rychlosti algoritmů.

.



Obrázek 2.3-4 R-CNN testovací časy (10)

2.3.5 YOLO

YOLO bylo vytvořeno v roce 2015 a svou výkonností předčilo všechny své předchůdce. YOLO neboli „you look only once“, je první jednofázový model. Tedy jak jeho název napovídá, není rozdělen na více částí, ale pracuje najednou s celým obrazem. Hlavní vylepšení oproti starším algoritmům je rychlost, možnost rozpoznávat objekty v reálném čase.

YOLO algoritmus má veškeré parametry lepší než jiné dosavadní algoritmy. Je to algoritmus založený na regresi, namísto vybírání oblastí, které by mohly obsahovat objekt, předpovídá třídy a ohraničení pro celý obraz v jednom běhu algoritmu. (11) (12)

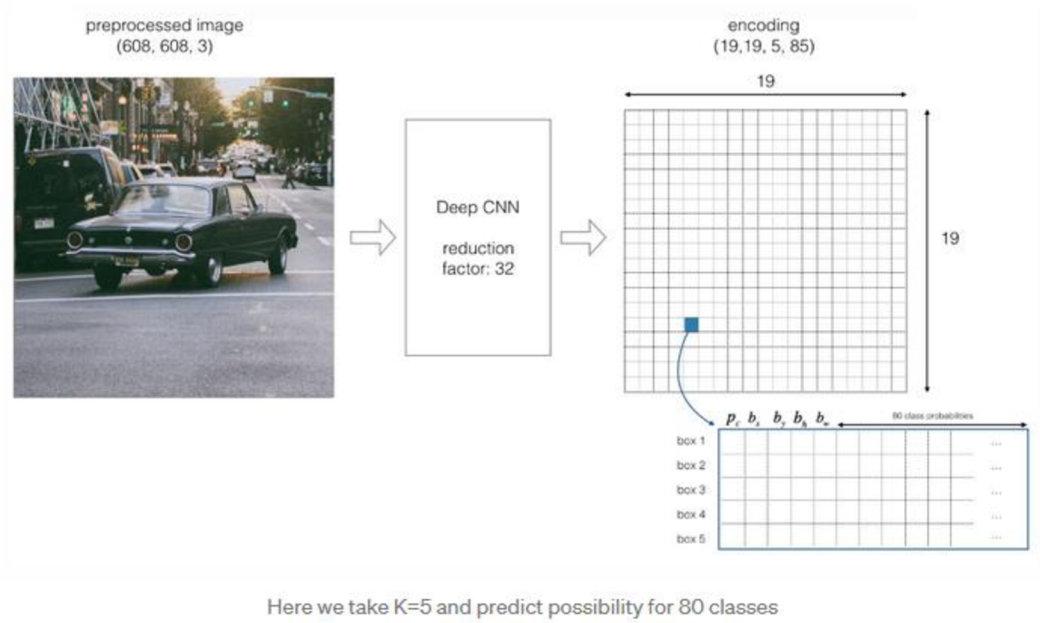
Pro pochopení YOLO algoritmu je zapotřebí vědět, co je reálně predikováno. Každé ohraničení objektu může být popsáno následujícími vlastnostmi.

1. Střed objektu, tedy souřadnice (bx, by)
2. Šířka (bw)
3. Výška (bh)
4. Hodnota c, která udává třídu objektu

Ještě existuje hodnota „pc“, která určuje pravděpodobnost výskytu objektu v ohraničení.

.

YOLO nehledá oblasti zájmu v obraze, které by mohly obsahovat objekt. Místo toho rozděluje obrázek mřížkou, nejčastěji 19x19. Každé pole této mřížky je zodpovědné za nalezení K počtu ohraničení.



Obrázek 2.3-5 YOLO algoritmus (11)

Pokud leží střed objektu v poli mřížky, pouze tehdy je dané pole označené za objekt. Souřadnice středu jsou vždy počítány relativně k poli mřížky, zatímco šířka a výška boxu je počítána k celému obrazu.

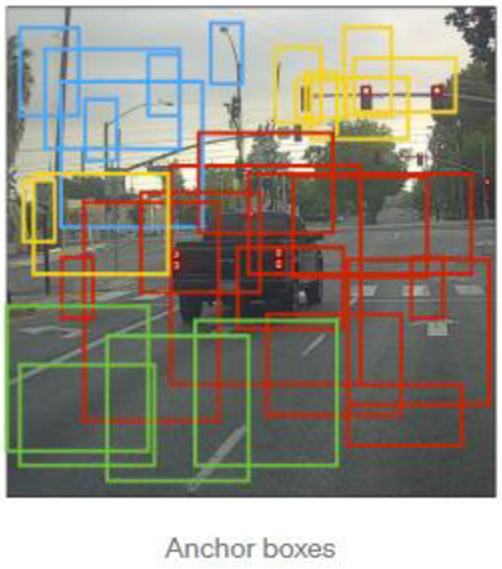
V průběhu algoritmu YOLO určuje pravděpodobnost třídy, která se v mřížce nachází. Třída s největší pravděpodobností je poté vybrána. Tento proces je pro každé pole v mřížce.

Poté, co je dopočítána pravděpodobnost třídy, může obrázek vypadat následovně:

.

Obrázek 2.3-6 Mřížka YOLO (11)

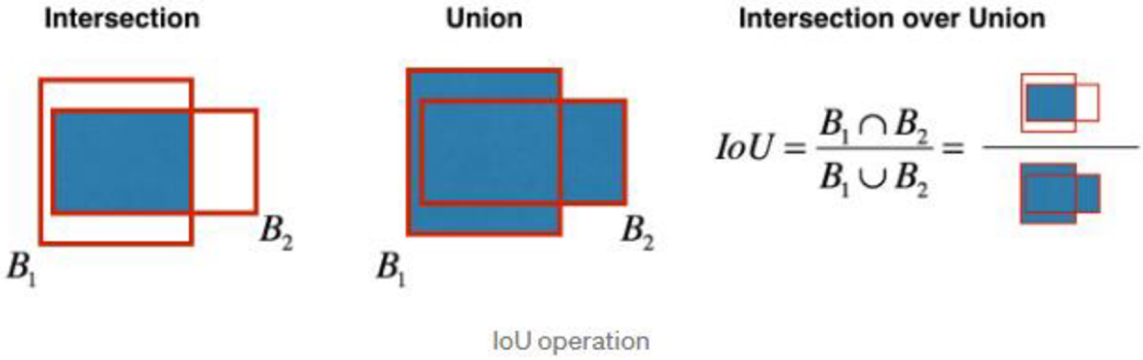
V obrázku je vidět, jak algoritmus prošel všemi poli a vyhodnotil jejich třídy. Dalším krokem je tzv. Non-max supression. V tomto kroku se algoritmus zbavuje zbytečných ohraničení, jako můžete vidět v následujícím obrázku.



Obrázek 2.3-7 Rozpoznané objekty (11)

Pro vyřešení tohoto problému je použita metoda IoU (Intersection over Union), eliminují se blízké ohraničení a porovnávají se s tím, který má největší pravděpodobnost na vyskytnutí objektu.

.



Obrázek 2.3-8 IoU operace (11)

Tento proces počítá hodnotu IoU pro všechny ohraničení v porovnání s ohraničením, který má nejvyšší pravděpodobnost výskytu objektu. Poté všechny výpočty, které mají hodnotu IoU větší než daný limit, se odstraní. Znamená to, že dané ohraničení překrývá stejný objekt, ale má menší pravděpodobnost, tím pádem zaniká. (13)

Jakmile je proces u konce, algoritmus najde další ohraničení s nejvyšší pravděpodobností výskytu objektu a zopakuje proces pro dané ohraničení. Opakuje se tolikrát, dokud nedostaneme nepřekrývající se ohraničení různých objektů.

Poslední krok YOLO algoritmu je odeslání finálních vektorů s výsledky. Součástí YOLO algoritmu je také „Loss Funkce“, která umožňuje algoritmu učit se parametry pro odhadování objektů v obraze.

2.4 Čtení textu RZ

Bylo zapotřebí také využít nástroje k přečtení textu po nalezení samotné RZ. Dále jsou v textu rozepsány tři nástroje, které byly v rámci bakalářské práce využity. Jedním z nich byl Tesseract OCR druhým EasyOCR a třetím ALPR, který je přímo upravený pro hledání RZ.

2.4.1 OCR

Zkratka OCR (z anglického Optical Character Recognition), neboli optické rozpoznávání znaků, je mechanická nebo elektronická přeměna ručně psaného nebo tištěného textu do strojově kódovaného textu, ať už ze skenovaného dokumentu, vyfoceného dokumentu nebo fotografie s textem. (14) Také se dá použít v digitální podobě, kde je hojně využíván mnoha softwary. Existují komerční i volně dostupné

.

zdroje. Mezi nejznámější volné dostupné systémy patří Tesseract OCR, GOCR, CuneiForm.

2.4.2 Historie

Prvotní optické rozpoznávání znaků lze sledovat na technologiích zahrnujících telegrafii a vytváření čtecích zařízení pro nevidomé. V roce 1914 vynalezl Emanuel Goldberg stroj, který četl znaky a přeměnil je na standardní kód telegrafu. Edmund Fournier d'Albe vyvinul současně Optophone, ruční skener, který při přesunutí potištěné stránky vytvořil tóny, které odpovídaly konkrétnímu písmenu nebo znaku.

2.4.3 Tesseract OCR

V rámci bakalářské práce byla zvolena knihovna Tesseract OCR. Důvodů pro toto rozhodnutí bylo několik. Hlavním důvodem je velikost této knihovny, která podporuje více než 100 jazyků, a má velikou přesnost, co se týče rozpoznávání jednotlivých znaků. Je volně dostupná pro stažení a podporuje vícero operačních systémů. Je to volně dostupný software pod licencí Apache a je sponzorován a propagován firmou Google od roku 2006. Software je možné rozšiřovat různými prvky například podporou dalších jazyků. Samotný software nemá grafické rozhraní, a pokud ho daná aplikace vyžaduje, je zapotřebí aplikací třetích stran. Tesseract také podporuje mnoho typů výstupů pro soubory jako je čistý text, HTML, PDF, TSV. Celé ovládání softwaru je řízeno přes příkazový řádek. Veškerý kód je dostupný z oficiálního githubu. Zdrojový kód byl původně psán v jazyce C a následně přepsán do jazyku C++ z důvodu udržitelnosti kódu.

Tesseract byl původně vytvořen v Hewlett-Packard Laboratories Bristol a v Hewlett-Packard Co, Greeley Colorado v letech 1985 až 1994. Následován dalšími změnami provedenými v roce 1996 pro port na Windows a v roce 1998 s některými úpravami v C++. V roce 2005 byl Tesseract otevřen jako open-source. Od roku 2006 do listopadu 2018 byl vyvíjen společností Google. (15)

2.4.4 EasyOCR

EasyOCR je balíček Pythonu, který umožňuje programátorovi využít optické rozpoznávání znaků a čísel. EasyOCR je velice přímočará knihovna pro používání. Tento balíček je možné nainstalovat pomocí jednoho pip příkazu. Poté již stačí jen import a pouhé 2 řádky kódu pro základní funkčnost. Tento balíček je pod správou

.

firmy Jaided AI, která se specializuje v optickém rozpoznávání textu. Podporuje více než 80 jazyků. (16)

2.4.5 OpenALPR

OpenALPR je volně dostupná knihovna pro rozpoznávání poznávacích značek. Je napsána v jazyce C++ s možností využití v Jazycích C#, Java, Node.Js, Go a Python. Tato knihovna analyzuje obrázky a videa a v nich identifikuje poznávací značky. Výstupem je poté textová reprezentace znaků z rozpoznaného objektu.

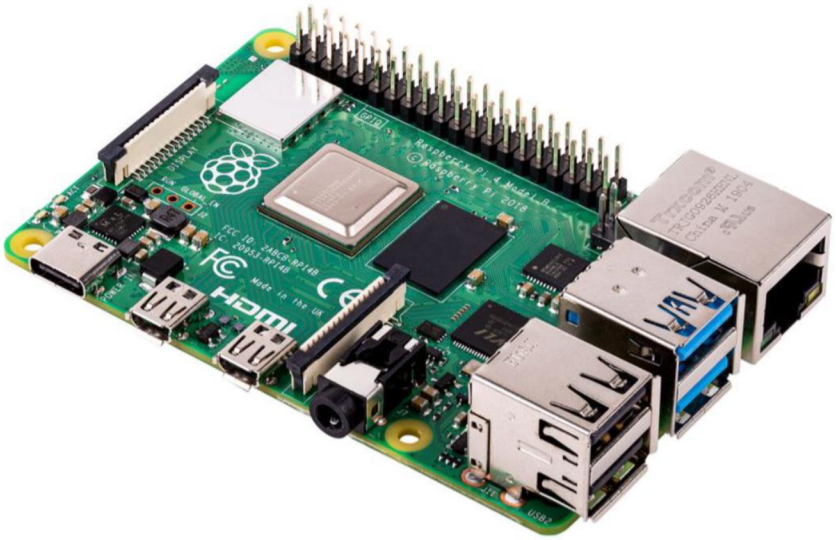
.

3 Kamerový systém Raspberry Pi

Jedním z cílů této práce bylo sestavit funkční systém pomocí kamery a Raspberry Pi. Raspberry Pi je malý jednodeskový počítač s deskou plošných spojů o velikosti zhruba platební karty. V roce 2012 byl vyvinut britskou nadací Raspberry Pi Foundation s cílem podpořit výuku informatiky ve školách a seznámit studenty s tím, jak mohou počítače řídit různá zařízení (např. mikrovlnná trouba, automatická pračka). Primárním operačním systémem je Raspbian. (1)

3.1 Raspberry Pi 4

Pro účely této práce byl vybrán nejnovější model Raspberry Pi 4 s následujícími parametry: Vnitřní paměť 1 GB RAM, napájení 5 V, procesor 1.5GHz Cortex-A72 quad Core a operační systém Raspbian.



Obrázek 3.1-1 Raspberry Pi 4 (17)

Kamera byla zvolena Arducam OV5647 5Mpx, která má rozlišení 2592×1944 a nahrává v 1080p 30fps nebo 720p 60fps.

Pro kameru a Raspberry Pi byl zvolen kryt proti poškození. Výsledný produkt je viděn na obrázku Obrázek 3.3-1.

.

Obrázek3.1-2 Arducam OV5647 (17)

3.2 Instalace Raspberry Pi

Z oficiálních stránek Raspberry Pi lze stáhnout instalaci systému NOOBS (Offline and network install) ve formátu ZIP. Tento obsah se následně rozbalí a přehraje na SD kartu.

Druhou možností je stažení programu Raspberry Pi Imager a následná instalace tohoto programu. Po otevření programu již stačí jen zvolit operační systém a SD kartu, na kterou se daný systém má nainstalovat. Nainstalovaná verze Raspbian systému je „Bullseye“, tedy Debian 11. Kernel verze jádra je 5.10.92.

Jako další se SD karta vsune do Raspberry Pi a připojí se zde i potřebné příslušenství (kamera, monitor). Systém se zapne a nakonfiguruje.

3.3 Zabezpečení Raspberry Pi

Jedním z dalších úkolů bylo zabezpečit přístup na Raspberry Pi. Ze základu Raspberry Pi není skoro vůbec zabezpečené. Je to převážně stroj určený k testování a k malým projektům. Ovšem ne do produkce. Proto je zapotřebí udělat několik kroků k jeho zabezpečení, pokud ho chceme používat na síti.

.

3.3.1 Nastavení účtu

Jako první je potřeba vytvořit nového uživatele a zakázat výchozího. Pomocí toho zajistíme, aby se nikdo nepřipojil přes výchozí údaje Raspberry Pi. Tomuto uživateli nastavíme administrátorská práva.

3.3.2 Aktualizace Raspberry Pi

Raspberry Pi při instalaci má určitou verzi. Je proto potřeba tuto verzi aktualizovat kvůli možným chybám ve starších verzích. Tato aktualizace se dá provádět ručně, ale také automaticky. Raspberry se většinou využívá pro konkrétní činnost a již málokdy se k němu uživatel poté připojuje. Proto je lepší variantou automatická aktualizace. Nastavit se dá pomocí balíčku „unattended-upgrades“, ve kterém je následně nutnost přepsat verzi na Raspbian.

3.3.3 Vypnutí nepotřebných programů

Když poprvé naběhne systém Raspbian, běží na něm spousty programů v pozadí, které nejsou pro projekt potřeba. Z bezpečnostního hlediska je dobré tyto programy vypnout a také uzavřít veškeré porty, které se pro komunikaci používat nebudou.

Pomocí příkazu „systemctl –type=service –state=active“ lze vyhledat aktivní služby a vypnout ty, jež nepoužíváme a mohly by představovat riziko napadení (Bluetooth, alsa-state).

3.3.4 Instalace Firewall

K zabezpečení portů je nutná instalace a konfigurace firewallu. Poté jeho samotné nasazení. Pro přihlašování na Raspberry bylo zvoleno SSH, při konfiguraci je potřeba nezablokovat si přístup, jinak se do systému nedostane ani uživatel. Pro jednoduchost nastavení byl zvolen balíček „uncomplicated firewall“. Pomocí „ufw allow“ se povolil port pro přístup SSH, obvykle tento port bývá 22 a používá tcp protokol. Pro lepší zabezpečení je možné tento port změnit. Jako poslední je potřeba zapnout samotný firewall.

3.3.5 Zabezpečení SSH

Samotné SSH je nejčastějším způsobem připojování k Raspberry Pi. Je zapotřebí ho zabezpečit více než jen heslem.

- Nastavit, jací uživatelé přesně mají dovolený přístup přes SSH.

.

- Blokovat přístup, respektive povolit přístup z konkrétních IP adres.
- Blokovat přístup po několika neúspěšných pokusech o přihlášení.
- Vygenerování SSH klíče.

K blokování IP adres po neúspěšných pokusech o přihlášení se využilo balíčku fail2ban. Tento balíček blokuje veškeré IP adresy, kterým se X krát nepodaří přihlásit do systému.

Pro vyšší zabezpečení SSH je dobré vygenerovat si SSH klíč. SSH klíč slouží k přihlášení přes SSH. Na zařízení, ze kterého se uživatel chce přihlašovat, je zapotřebí vygenerovat tento SSH klíč a poté ho nahrát do Raspberry Pi. Poté je možné vypnout přihlašování pomocí hesla a do systému se dostane pouze zařízení, které má tento klíč.

3.3.6 Použití zabezpečených protokolů pro přenos dat

Pokud by bylo zapotřebí odesílat data přes internet, doporučuje se používat zabezpečené verze protokolů, např. HTTPS, SFTP. V této práci nebylo zapotřebí odesílat data přes internet, pouze se ukládala lokálně.

3.3.7 Použití VPN

Více radikálním přístupem do Raspberry Pi je použití VPN. VPN, neboli virtuální soukromá síť, je zabezpečené šifrované připojení mezi dvěma sítěmi nebo mezi konkrétním uživatelem a sítí. Všechny toky mezi zařízením uživatele a Raspberry Pi je poté enkryptováno pomocí silného protokolu.

V rámci bakalářské práce nebylo zapotřebí použití VPN.

.

Obrázek 3.3-1 Kryt na Raspberry Pi a kameru (17)

.

4 Návrh a implementace aplikace

Návrh aplikace byl poměrně přímočarý, byly zde jasně dané parametry. Z hlediska uživatelského prostředí zde nebylo potřeba mnoho věcí. Důležité bylo mít výstup s výsledky rozpoznávání v dobrém formátu pro následné vytváření statistik apod. Pro uživatele zde bylo možné přidat displej pro zobrazování aktuálního rozpoznávání, ale to se do výsledné aplikace nedostalo z důvodu toho, že nebylo nutné data zobrazovat hned ve chvíli detekování, ale pouze se ukládali na pozdější využití. Při návrhu a implementaci aplikace se postupovalo následujícím způsobem.

1. Instalace potřebných balíčků pythonu, jejich následná konfigurace a nastavení, naprogramování aplikace.
2. Získání potřebných dat pro trénování modelů a testování.
3. Trénování modelů.
4. Rozpoznávání a testování úspěšnosti modelů.
5. Aplikování OCR na text RZ pro přečtení.
6. Uložení dat do konkrétního formátu.

Bakalářská práce byla rozdělena do několika částí, kde nejprve bylo zajištěno testovací prostředí pro aplikaci. Následovalo získání potřebných dat pro natrénování modelů pro rozpoznávání RZ. Poté bylo zapotřebí vybrat modely, natrénovat je a otestovat jejich funkčnost.

4.1 Prvotní instalace

Veškerý kód bakalářské práce byl napsán v jazyce Python a testován v prostředí Windows na osobním počítači. Následně přenesen na Raspberry Pi. Parametry počítače byly následující:

- Procesor AMD Ryzen 3600
- Grafická karta Radeon GT 5600
- 16 GB RAM

.

Po instalaci jazyku Python bylo potřeba stáhnout potřebné balíčky pro rozpoznávání objektů, vykreslování obrazu a podobně. Pro tento systém byly využity dvě klíčové knihovny. První z nich bude rozpoznávat objekty v obraze a druhá z objektu číst text.

4.1.1 Tensorflow Object Detection

Tensorflow je repositář již natrénovaných modelů připravených pro následné úpravy a konkrétní natrénování. Veškeré materiály jsou k dispozici otevřeně pro veřejnost. Tato knihovna vznikla pod záštitou firmy Google.

Nejprve se vytvořilo virtuální prostředí, do kterého se nainstalovala knihovna Tensorflow. Většina kódu byla konfigurována přes prostředí Jupyter Notebook. Pro instalaci této knihovny bylo zapotřebí mnoho dodatečných balíčků. Po nainstalování balíčku byly staženy již před trénované modely z Tensorflow Model Zoo, ze stránek https://www.tensorflow.org/install/source_windows. Pro tento krok bylo nutné stáhnout balíček wget pro následné stažení modelů.

Tento balíček umožňuje stáhnout Model Garden z oficiálních stránek Tensorflow. Model Garden je repositář mnoha rozdílných modelů a modelových řešení pro uživatele používající Tensorflow. Z tohoto repositáře je poté možné využít jakýkoliv model a obsahuje mnoho dalšího materiálu pro detekci objektů. Při instalaci balíčků se nevyskytly žádné problémy. (18)

4.1.2 Získání dat k trénování a testování modelů

Aby mohly být využity algoritmy pro rozpoznávání RZ, bylo nejprve zapotřebí sestavit trénovací a testovací data pro tyto modely. Jako první byly vytvořeny tři databáze fotek RZ. Trénování bylo zajištěno pomocí databáze dostupné ze stránek kaggle od uživatele Larxel, která obsahuje dostatečný počet obrázků pro natrénování modelů a je pod otevřenou licencí. Obsahuje konkrétně 433 obrázků RZ různých zemí, úhlů kamery a kvality obrazu. Obrázky jsou již také anotovány, to znamená, že jsou připravené pro trénování modelů. Dále byly pro testovaní modelů vytvořeny dvě databáze RZ, které byly nafoceny v různých místech po České republice, které měly převážně české RZ. Databáze A obsahuje 154 obrázků nafocených za denního světla. Databáze B obsahuje 128 obrázků nafocených za horších podmínek, také upravených pro horší podmínky

.

rozpoznání. K dispozici bylo také video pro rozpoznávání RZ a následně se testovalo v reálném čase.

Obrázek 4.1-1 Příklad obrázku pro trénování modelu

4.1.3 Trénování modelů

Zde vždy platí, že daný model musí mít nejlépe co nejvíce trénovacích dat. Trénovací data byla ve formátu PNG a ke každému obrázku musí být vždy soubor anotace. Soubory anotace byly ve formátu XML, kde vždy bylo popsáno, na jakém místě se RZ v obraze nachází, aby model vždy věděl, na co se má v obraze zaměřit. Na začátku procesu učení má model vždy velikou trénovací chybu, která se postupně s cykly tréninku zmenšuje. Trénovacích cyklů bylo vždy 10000. Trénování bylo nutné provádět na počítači z důvodu slabého výkonu Raspberry Pi. Trénovány byly dva modely pro následné porovnání úspěšnosti a zvolení lepšího z nich pro realizaci tohoto projektu. Zvolené modely využívaly algoritmy Faster R-CNN a YOLO. (19)

.

```
<annotation>
    <folder>images</folder>
    <filename>Cars0.png</filename>
    <size>
        <width>500</width>
        <height>268</height>
        <depth>3</depth>
    </size>
    <segmented>0</segmented>
    <object>
        <name>licence</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <occluded>0</occluded>
        <difficult>0</difficult>
        <bndbox>
            <xmin>226</xmin>
            <ymin>125</ymin>
            <xmax>419</xmax>
            <ymax>173</ymax>
        </bndbox>
    </object>
</annotation>
```

Obrázek 4.1-2 Příklad souboru anotace

Pro úspěšné trénování modelu bylo nutné vytvořit označení pro dané objekty, vytvořit TF záznamy a připravit konfiguraci modelů. TF záznamy jsou určené pro modely k trénování, je to upravený formát vstupních dat. K vytvoření TF záznamů se nejprve vytvořila tzv. label map, tedy reference, která popisuje, co za třídu se nachází na obrázku. Tato mapa je samostatný zdroj pro anotaci třídy objektu. Reprezentuje všechny možné objekty, které tento model může detekovat. V tomto případě pouze jeden, a to poznávací značku. Formát je podrobněji vidět na Obrázek 4.1-3 Label mapa. (20) (21)

```
item {
      name:'licence'
      id:1
}
```

Obrázek 4.1-3 Label mapa

.

TF záznamy jsou potřebné a nezbytné pro trénování modelu. K vytvoření těchto záznamů bylo nutné stáhnout oficiální skript na jejich vytváření a upravit ho pro formát používaných dat, jelikož anotační soubory byly v jiném formátu, než oficiální skript je napsán. Tento skript převádí formáty materiálů pro trénování na TF záznamy.

Modely, které byly použity v rámci bakalářské práce, byly ve formátu konfiguračních souborů. V těchto souborech bylo nutné upravit veškeré nastavení cest k souborům pro trénování, testování k label mapě apod.

Ve chvíli, kdy vše potřebné bylo zajištěno, mohlo začít trénování samostatných modelů, kde se nastavil počet celkových cyklů trénovaní, v případě použitých modelů to bylo 10000. Čím více těchto kroků, tím déle se daný model bude trénovat. Výstupem trénování jsou vždy soubory (záchytné body), kdy ten s nejvyšším číslem je poslední vytvořený při trénování. Tento soubor se poté používá jako model. Po natrénování modelů se oba modely vyzkoušely na testovacích datech testovacích sad A, B.

průběh trénování modelu YOLO se velice přibližuje limitě blížící se nule. Jeho chyby se postupně zmenšují a ke konci trénovací fáze se blíží hodnotě 0,03. To stejné platí i pro model Faster R-CNN, který však neměl tak plynulý průběh.

4.1.4 Rozpoznávání a testování úspěšnosti modelů.

Pro zobrazení detekování objektu byla použita knihovna matplotlib.

Obrázek 4.1-4 rozpoznaná RZ

V přiložených tabulkách Tabulka 1 a Tabulka 2 jsou zaznamenány výsledky úspěšností obou modelů na konkrétních datech z databází obrázků, které byly pro účely testování

.

vytvořeny. Pro oba modely jsou vyhodnoceny všechny obrázky z daných databází. Pro výpočet úspěšnosti se nejprve počítaly správně detekované poznávací značky a počet nenalezených poznávacích značek. Z těchto dvou hodnot je poté vypočítán Recall, tedy hodnocení samotné úspěšnosti. Dále jsou v tabulce také počítány případy, kdy daný model vyhodnotil objekt špatně, a v daném místě žádná poznávací značka nebyla.

Použití rozpoznávání objektu v reálném čase bylo na podobném principu jako na obrázku Obrázek 4.1-4. Zde se využilo knihovny opencv pro zobrazení videa.

4.1.5 Aplikování OCR na text RZ

Po úspěšném natrénování a odzkoušení modelu pro rozpoznání poznávacích značek následovalo jejich přečtení. Pro tento účel bylo potřeba OCR. Konkrétně knihovna EasyOCR podporující Python. Tato knihovna využívá PyTorch, je zde proto důležité rozdělit využití grafické karty pro PyTorch, který bude využívat EasyOCT, a Tensorflow, které využívá model pro rozpoznávání RZ. Při aplikování OCR je potřeba nastavit si omezující hodnotu, při které chceme RZ přečíst, například při 90% jistotě modelu, že je na daném obrázku RZ.

Při čtení textu RZ se muselo vyřešit několik konkrétních problémů, které se vyskytly. Poznávací značky, které měly více textu než samotné označení, velkým způsobem ovlivňovaly výsledky OCR knihoven. Proto zde byl zaveden systém, který umí vyfiltrovat výsledky z OCR a vzít si ten, který zabírá největší plochu RZ.

4.1.6 Uložení dat

Jako poslední v rámci návrhu a implementace aplikace bylo vytvořit metodu pro ukládání dat a formát dat, ve kterém se budou ukládat. V metodě se ukládaly následující parametry: text RZ, region, respektive rozpoznaný oříznutý objekt, a čas, ve kterém se daná RZ objevila. Obraz se ukládá ve formátu jpg, vždy s unikátním názvem. Zbytek dat, zároveň s názvem daného objektu, se ukládá do csv souboru, který má veškeré záznamy o rozpoznaných objektech.

.

	A	B
1	c65cbdd8-d13f-11ec-b7b0-b42e99611fe1	[5L6 2171]
2	d353a6e7-d13f-11ec-9eea-b42e99611fe1	[4AK 7253]
3	d8df41e5-d13f-11ec-acc5-b42e99611fe1	[7J0 2092]
4	df9af6e7-d13f-11ec-a81f-b42e99611fe1	[5L8 4101]

Obrázek 4.1-5 Uložená data

Obrázek 4.1-6 Rozpoznaný objekt

4.2 Výsledky testování

Pro samotné testování se nejprve využily dvě databáze již vytvořených fotografií. V těchto testech se vyhodnotilo, který model bude poté využit pro test v reálném čase. Také byly modely využity s různými OCR balíčky pro porovnání jejich úspěšnosti a celkové rychlosti. V tabulce můžeme vidět úspěšnost rozpoznání RZ na databázích modelu YOLO v porovnání s modelem Faster R-CNN. Rychlost nalezení RZ se pohybovala kolem 300 ms.

Tabulka 1: Úspěšnost YOLO algoritmu

Testovací sada	Práh rozpoznání	Odhalené RZ	Falešné poplachy
A	0.5	93.80%	58
	0.6	92.15%	25
	0.7	90.51%	16
	0.8	87.78%	6
	0.9	85.42%	2
B	0.5	95.12%	63
	0.6	91.13%	31
	0.7	89.65%	18
	0.8	86.12%	9
	0.9	84.89%	3

.

Pro porovnání rozdílu mezi rozpoznáváním textu pouze z oříznuté části RZ, tedy části, kterou rozpozná model YOLO nebo Faster R-CNN, se testovalo jen využití EasyOCR a Tesseract OCR pro nalezení RZ. Úspěšnost se blíží spíše k 0 %, jak je možné vidět v tabulce Tabulka 3. Důvodem pro to je, že tyto knihovny se zaměřují na veškerý text v obraze a nejsou specifické pro RZ. Byla zde také delší doba průměrného času pro nalezení. Zavedena zde byla i tzv. Levenshteinova vzdálenost.

Tabulka 2 Faster R-CNN

Testovací sada	Práh rozpoznání	Odhalené RZ	Falešné poplachy
A	0.5	95.45%	91
	0.6	94.25%	48
	0.7	92.51%	29
	0.8	84.78%	15
	0.9	75.42%	8
B	0.5	96.53%	102
	0.6	93.48%	69
	0.7	90.21%	31
	0.8	82.76%	18
	0.9	77.89%	6

4.2.1 Levenshteinova vzdálenost

Levenshteinova vzdálenost je metrika pro měření rozdílů mezi dvěma sekvencemi. Rozumíme tím, že Levenshteinova vzdálenost mezi dvěma slovy je minimální počet změn znaků (vložení, smazání, nahrazení jiným) potřebných pro změnu jednoho slova na druhé. Je pojmenována po matematikovi Vladimírovi Levenshteinovi, který ji vymyslel v roce 1965.

Tabulka 3 EasyOCR a Tesseract OCR

Testovací sada	Tesseract OCR			EasyOCR		
	Celá RZ	Levenshteinova vzdálenost	Průměrný čas rozpoznání textu	Celá RZ	Levenshteinova vzdálenost	Průměrný čas rozpoznání textu
A	0.87%	16.25%	1543 ms	0.67%	15.12%	1748 ms
B	0.74%	17.11%	1612 ms	0.59%	15.98%	1692 ms

.

4.2.2 Testování EasyOCR a modelu

V následujícím porovnání lze vidět úspěšnost využití modelů YOLO a Faster R-CNN s využitím knihovny EasyOCR. Testování probíhalo na dvou testovacích databázích. Hodnotila se úspěšnost nalezení a následného rozpoznání textu na RZ. Tyto testy probíhaly stále na počítači. (22)

Tabulka 4 YOLO s použitím EasyOCR

Testovací sada	Celá RZ	Levenshteinova vzdálenost	Průměrný čas (YOLO)	Průměrný čas (OCR)
A	26.11%	46.42%	265 ms	185 ms
B	27.56%	47.89%	272 ms	192 ms

4.2.3 Testování TesseractOCR a modelu

Testování TesseractOCR a modelů bylo stejné jako u testování EasyOCR. V tabulce Tabulka 5 je možné vidět samostatnou úspěšnost.

Tabulka 5 YOLO s použitím Tesseract OCR

Testovací sada	Celá RZ	Levenshteinova vzdálenost	Průměrný čas (YOLO)	Průměrný čas (OCR)
A	28.01%	48.15%	265 ms	177 ms
B	29.32%	49.76%	272 ms	184 ms

4.2.4 Vyhodnocení

V předchozích testech modelů a knihoven OCR vycházela kombinace TesseractOCR a modelu YOLO dle úspěšnosti nejlépe. Tato kombinace se využila pro následné testování na Raspberry PI.

.

5 Testování na Raspberry Pi

V předchozí kapitole bylo vysvětleno, jak aplikace byla navržena. Také zde bylo uvedeno, jaké řešení bylo nejúspěšnější dle rychlosti a úspěšnosti rozpoznávání RZ. V této kapitole je popsáno, jakým způsobem byl kód implementován na Raspberry Pi a jakých dosahoval výsledků.

5.1 Implementace na Raspberry Pi

Jedním z cílů bakalářské práce bylo implementovat aplikaci na Raspberry Pi a zjištění, zdali tato platforma je schopná konkrétní systém zvládnout provozovat. Z rešerše je jasné, že to nebude schopné v reálném čase stíhat v porovnání s počítačem, s větším výkonem, ale i na Raspberry Pi je možné dosahovat určitých výsledků.

Raspberry Pi umožňuje nainstalovat knihovnu Tensorflow, ovšem nedoporučuje se to z důvodu větší náročnosti. Proto zde byla zvolena varianta Tensorflow lite, tato verze je upravená pro menší počítače, a proto je méně náročná. Pro tuto variantu se musel konvertovat model z Tensorflow do Tensorflow Lite. Pro tuto činnost byl využit oficiální converter z webových stránek Tensorflow. Ostatní knihovny byly použity stejné a funkční i pro tento systém. (23)

YOLO je plně konvoluční neuronová síť. To znamená, že nejsou plně propojené vrstvy sítě. To znamená, že YOLO může vyvozovat závěry z obrázků s různými velikostmi. Tedy použilo se zde upravování vstupního obrazu pro správu kompromisu mezi rychlostí a přesností detekce RZ.

Stejně jako u počítače, bylo potřeba rozdělit paměť jak Tensorflow, tak zbytku systému. Pro Tensorflow se využilo 75 % a zbytek byl pro ostatní procesy v systému. Grafické rozhraní by zde taky určitou část paměti zabíralo, proto bylo po dobu testování vždy vypnuté.

5.2 Testovací sady

Stejně jako v případě testování na počítači se zde nejprve otestovala funkčnost na testovacích sadách, kde testy vycházely velice podobně. Po otestování funkčnosti se mohl systém vyzkoušet v reálném čase. Testů probíhalo více, konkrétně 4. První dva testy ze čtyř jsou testy, kdy Raspberry Pi bylo pouze ve statické poloze vedle silnice, kde projížděla auta. Druhý test byl stejný, ovšem se sníženou viditelností (deštivé

.

počasí). Druhé dva testy byly za pohybu vozidla, kdy Raspberry Pi bylo uvnitř auta a snímalo vozidla v obou směrech. Zde byl jeden problém. Vždy, když na záběru kamery bylo vozidlo jedoucí stejným směrem, tedy stále na záběru, měl s tím systém problém, jelikož stále rozpoznával tento objekt a stále si o něm ukládal data, tedy jedna RZ mohla být vícekrát rozeznána.

Obrázek 5.2-1 Detekování RZ

Výsledky tohoto testování jsou možné vidět v tabulkách Tabulka 6 a Tabulka 7. Je zde vidět, jakou rychlostí průměrně byla RZ rozpoznána. Z výsledků je jasné, že pokud bychom po takovém systému opravdu vyžadovali operace v reálném čase i s dalšími prvky jako ukládání obrazu, nahrávání na server a podobné, bylo by třeba výkonnějšího zařízení, než je Raspberry Pi. Zároveň co se týče kvality obrazu, určitě by změna kamery vylepšila konkrétní statistiky pro rozpoznávání objektu. Výsledky jsou celkově očekávané, s menším počtem snímků za sekundu nebylo pro Raspberry Pi možné detekovat veškeré RZ. Přesto byla úspěšnost okolo 80 %, ovšem se čtením RZ je to již horší. Díky kvalitě obrazu a dalším problémům byla úspěšnost přečtení všech

.

znaků okolo 20 %. V rychlostech po městech stačilo využít každý druhý snímek za sekundu z kamery.

Tabulka 6 Statické testy

Aut celkově	Rozpoznaných RZ	Celková úspěšnost	Falešných poplachů	Celé RZ
100	87	87%	3	28.73%
80	54	67.50%	6	24.21%

Tabulka 7 Dynamické testy

Aut celkově	Rozpoznaných RZ	Celková úspěšnost	Falešných poplachů	Celé RZ
55	38	69%	9	18.73%
63	45	71.43%	11	21.21%

Aplikace splňuje do jisté míry požadavky systému dle zadání. Je přenosná, může se tedy použít kdekoliv. Jako napájení je zde zapotřebí 5V. Celková cena systému v tomto konkrétním provedení byla přesně 2150,- Kč i s krytem na Raspberry Pi a kameru.

.

6 Vylepšení aplikace

Z pohledu posledního bodu zadání, zamyšlením se nad uživatelským rozhraním a přívětivostí aplikace se může navrhnout několik vylepšení. Určitě by bylo možné odesílat data přes síť do databáze, odkud by si je mohla následně převzít například webová aplikace, kde by byly všechny statistiky podrobně zobrazeny.

Důvod, proč systém není tak výkonný ve vyšších rychlostech a v horších podmínkách, je určitě i po stránce hardwaru, kde by bylo možné použít výkonnější počítač. Výkonnější počítač by určitě zajistil možnost snímání blíže k reálnému času. Také by mohla být použita knihovna Tensorflow. Dále by bylo možné pořídit lepší kameru pro vyšší kvalitu obrazu. Při vyšší kvalitě obrazu je potom o dost vyšší šance na rozpoznání většiny znaků RZ. Na obrázku 23, kde druhá RZ v pozadí již nebyla rozpoznána, se dá pozorovat, že z důvodu kvality obrazu, tak i z důvodu vzdálenosti od kamery nebyla druhá RZ rozpoznána na 80 % a víc. Určitě by také šla vylepšit přesnost modelu v případě natrénování na českých RZ a větším počtu trénovacích dat, který v tomto případě byl necelých 450.

.

7 Závěr

Cílem bakalářské práce bylo nastudovat problematiku rozpoznávání objektů v obraze, konkrétně RZ, dále čtení textu z obrazu. Dalším cílem bylo tyto metody implementovat a navrhnout aplikaci, která je bude využívat. Natrénovat modely a otestovat v ostrém provozu a následně vytvořit funkční přenosný kamerový systém s automatickým rozpoznáváním, který bude levnější než možné alternativy na trhu. Systém byl složen z Raspberry Pi, které bylo také nutné zabezpečit dle jednoho z cílů.

V praktické části bakalářské práce bylo popsáno, jakým způsobem byl systém Raspberry Pi nainstalován a zabezpečen proti případným napadením systému. Následně byl popsán proces implementace samotné aplikace, kde byly nejprve popsány algoritmy pro rozpoznávání objektů, konkrétně YOLO a Faster R-CNN, které bylo nutné natrénovat a otestovat. Trénování probíhalo pomocí předem vytvořené databáze snímků RZ a jejich anotací, testování poté probíhalo na vytvořených databázích českých RZ nafocených především v Liberci. Druhou sadu algoritmů pro optické rozpoznávání textu z obrazu bylo také nutné otestovat, tyto algoritmy byly EasyOCR a Tesseract OCR. V těchto testech se hodnotila přesnost a úspěšnost rozpoznání jednotlivých znaků RZ. Po těchto testech a jejich výsledcích byla zvolena kombinace těchto algoritmů, které měly největší úspěšnost. Nejlepší kombinací byl algoritmus YOLO pro rozpoznávání RZ spolu s algoritmem Tesseract OCR pro optické rozpoznávání textu.

V další části se veškerý kód aplikace přesunul na systém Raspberry Pi, kde bylo nutné nainstalovat knihovny přímo pro tento systém, aby bylo možné použít modely z počítače. Na tomto systému poté proběhlo několik testů, ze kterých byla vytvořena statistika a záznam o úspěšnosti. Testy byly celkem 4, kdy dva z nich byly ve statické poloze systému, zatímco druhé dva byly za jízdy automobilu. Vždy se snímal jeden jízdní pruh.

Systém je možné přenášet a využít ho tedy kdekoliv. Systém také není drahý a pořizovací cena potřebných součástí je 2150,- Kč. Tento systém samozřejmě nedosahuje kvalit velkoprůmyslových řešení, ovšem při možnosti větší investice do možného hardwaru a lepší kamery by výsledky byly značně lepší. Finální statistiky vychází průměrně 80 % úspěšnosti na rozpoznání RZ v obraze a dále průměrně 20 %

.

na přečtení celé RZ správně, bez jediné chyby. U čtení RZ je to z velké části kvůli špatné kvalitě obrazu. Výsledek by také mohl být zlepšen využitím více dat pro trénování modelu, například různě natočených RZ, různě kvalitních snímků RZ a také RZ v různých situacích (počasí, viditelnost).

.

Seznam obrázků

.

Seznam Tabulek

.

Citovaná literatura

1. **wikipedia.org. *Státní poznávací značka.* [Online] [Citace: 5. Leden 2022.] https://cs.wikipedia.org/wiki/St%C3%A1tn%C3%AD_pozn%C3%A1vac%C3%AD_zna%C4%8Dka.**

2. ježek, Martin. finance.cz. *Jak vypadají SPZ.* [Online] 22. Září 2018. [Citace: 15. Leden 2022.] https://www.finance.cz/514594-zahranicni-spz/.

3. [Online] https://www.gatenachod.cz/anglictina/z014_indi.jpg.

4. Boesch, Gaudenz. viso.ai. *Object Detection in 2022.* [Online] 2022. [Citace: 23. Duben 2022.] https://viso.ai/deep-learning/object-detection/?fbclid=IwAR2PNUldCJQSsa8zuyz9jp6BQqSsBVZAZ0OClznjhmr6acQhr-61cXD3FU8.

5. Solawetz, Jacob. roboflow.com. *Object Detection - The Ultimate Guide.* [Online] 1. Únor 2021. [Citace: 14. Prosinec 2021.] https://blog.roboflow.com/object-detection/?fbclid=IwAR0vmIZ_Ywt5Bxdk7VuzMvJnqy-EzIx2_84s3pcqSrhhDZ78kECjrAvuErs.

6. Fritz Labs. Fritz.ai. *Object Detection Guide.* [Online] 2021. [Citace: 24. Listopad 2021.] https://www.fritz.ai/object-detection/?fbclid=IwAR32LfhMVtAylHGvnJmHFVLJuI1RirwKqi7lMqEs0zWxCM8Iwj1ZBHxJgaQ#:~:text=Object%20detection%20is%20a%20computer,all%20while%20accurately%20labeling%20them.

7. Burns, Ed. techtarget.com. *Machine learning.* [Online] [Citace: 25. Listopad 2021.] https://www.techtarget.com/searchenterpriseai/definition/machine-learning-ML?fbclid=IwAR1Bi0_Zs7ezXnpIuA8GUnHGD7J2JL3bHJgZ0COehUKBHpZWTqEtLUrY_Bc.

8. Kalluri, H. semanticscholar.org. *Deep learning.* [Online] 2019. [Citace: 10. Prosinec 2021.] https://www.semanticscholar.org/paper/Deep-learning-and-transfer-learning-approaches-for-Krishna-Kalluri/6c2a0e3a798d59655732980d2b03b9e89f586548.

9. Khandelwal, Renu. towardsdatascience.com. *A basic Introduction to TensorFlow Lite.* [Online] 15. Červen 2020. [Citace: 12. Duben 2022.] https://towardsdatascience.com/a-basic-introduction-to-tensorflow-lite-59e480c57292.

10. Gandhi, Rohith. towardsdatascience.com. *R-CNN, Fast R-CNN, Faster R-CNN, YOLO.* [Online] 9. Červenec 2018. [Citace: 21. leden 2022.] https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e.

11. Manishgupta. towardsdatascience.com. *YOLO - You Only Look Once.* [Online] 30. Květen 2020. [Citace: 5. leden 2022.] https://towardsdatascience.com/yolo-you-only-look-once-3dbdbb608ec4.

12. Karmini, Grace. section.io. *Introduction to YOLO.* [Online] 15. Duben 2021. [Citace: 12. prosinec 2021.] https://www.section.io/engineering-education/introduction-to-yolo-algorithm-for-object-

.

detection/?fbclid=IwAR2hOP4uF302MERjEtTGistSzQ1nSuzEVGtp9asZeB5Rlg3x0dI0eaLb7y4.

13. Subramanyam, Vineeth S. medium.com. *IOU.* [Online] 17. Leden 2021. [Citace: 20. Prosinec 2021.] https://medium.com/analytics-vidhya/iou-intersection-over-union-705a39e7acef.

14. wikisofia.cz. *OCR.* [Online] [Citace: 24. Leden 2022.] https://wikisofia.cz/wiki/OCR.

15. github.com. *Tesseract-OCR.* [Online] 7. Leden 2022. [Citace: 10. únor 2022.] https://github.com/tesseract-ocr/tesseract.

16. Rosebrock, Adrian. pyimagesearch.com. *Getting started with EASYOCR.* [Online] 14. září 2020. [Citace: 17. Leden 2022.] https://pyimagesearch.com/2020/09/14/getting-started-with-easyocr-for-optical-character-recognition/.

17. https://rpishop.cz/. *https://rpishop.cz/.* [Online] https://rpishop.cz/.

18. Tensorflow. tensorflow.org. *Tensorflow Object Detection Colab.* [Online] [Citace: 18. únor 2022.] https://www.tensorflow.org/hub/tutorials/tf2_object_detection.

19. Renotte, Micholas. github.com. *Tensorflow Object Detection walkthrough.* [Online] 3. Duben 2021. [Citace: 18. Prosinec 2021.] https://github.com/nicknochnack/TFODCourse.

20. Larxel. kaggle.com. *Car License Plate Detection.* [Online] 2019. [Citace: 13. Leden 2022.] https://www.kaggle.com/datasets/andrewmvd/car-plate-detection?resource=download.

21. Solawetz, Jacob. towardsdatascience.com. *Label Map.* [Online] 2. Říjen 2020. [Citace: 12. květen 2022.] https://towardsdatascience.com/what-is-a-label-map-f1066af6df70.

22. JaidedAI. github.com. *EasyOCR.* [Online] JaidedAI, 10. červenec 2020. [Citace: 21. Leden 2022.] https://github.com/JaidedAI/EasyOCR/issues?page=5&q=is%3Aissue+is%3Aopen.

23. Gunnarsson, Adam. diva-portal.org. *Real time object detection on a Raspberry Pi.* [Online] 2019. [Citace: 24. březen 2022.] https://www.diva-portal.org/smash/get/diva2:1361039/FULLTEXT01.pdf.

24. Jędrzej Świeżewski, Ph.D. appsilon.com. *YOLO algorithm and YOLO Object Detection.* [Online] 22. květen 2020. [Citace: 15. červen 2022.] https://appsilon.com/object-detection-yolo-algorithm/.

25. Praveen. medium.com. *License Plate Recognition using OpenCV Python.* [Online] 7. Červenec 2020. [Citace: 15. říjen 2021.] https://medium.com/programming-fever/license-plate-recognition-using-opencv-python-7611f85cdd6c.

26. Ferm, Oliwer. diva-portal.org. *Real-time Object Detection on Raspberry Pi 4.* [Online] 2020. [Citace: 27. Březen 2022.] https://www.diva-portal.org/smash/get/diva2:1451946/FULLTEXT01.pdf.

.

.