Ekonomická fakulta
Faculty of Economics

Jihočeská univerzita v Českých Budějovicích
University of South Bohemia in České Budějovice

Jihočeská univerzita v Českých Budějovicích

Ekonomická fakulta

Katedra aplikované matematiky a informatiky

Diplomová práce

Systém rozpoznávání SPZ pro řízení a správu parkování

Vypracoval/a: **Bc. Dominik Kutil**

Vedoucí práce: **Ing. Ondřej Budík**

České Budějovice 2024

JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH

Ekonomická fakulta

Akademický rok: 2023/2024

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Dominik KUTIL**
Osobní číslo:	**E21048**
Studijní program:	**N0613A140025 Aplikovaná informatika**
Specializace:	**Softwarové inženýrství**
Téma práce:	**Systém rozpoznávání SPZ pro řízení a správu parkování**
Zadávající katedra:	**Katedra aplikované matematiky a informatiky**

Zásady pro vypracování

Cíl práce:

Seznámení se s dostupnými hardwarovými možnostmi k rozpoznávání SPZ (Kamera, výpočetní HW,)
Návrh řešení rozpoznávání SPZ pomocí strojového vidění (CNN) na low-cost zařízení.
Návrh propojení vyvinutého softwaru s databázovým systémem. Vyhodnocení povolených SPZ, logování průjezdů.
Návrh webového prostředí k ověření povolení parkování dané SPZ se zobrazením zbývajícího volného času.

Metodický postup:

1. Proveďte rešerši dostupných řešení a vyberte vhodný HW (RPi, Jetson, ESP, ...) pro běh vašeho softwaru, včetně výběru vhodného kamerového řešení.
2. Zajistěte technické zázemí instalace a zdokumentujte nutný postup nasazení.
3. Navrhněte skelet aplikace pro rozpoznání SPZ z kamery s propojením na databázový systém povolených SPZ. Aplikace musí běžet na zvoleném HW řešení. Aplikace musí provádět logování průjezdů.
4. Navrhněte webové prostředí pro vzdálenou validaci již zaparkovaných aut.

Rozsah pracovní zprávy:	**30 – 50 str.**
Rozsah grafických prací:	**dle potřeby**
Forma zpracování diplomové práce:	**tištěná**

Seznam doporučené literatury:

[1] SINGH, Himanshu. Practical Machine Learning and Image Processing: For Facial Recognition, Object Detection, and Pattern Recognition Using Python [online]. Berkeley, CA: Apress, 2019 [vid. 2024-01-05]. ISBN 978-1-4842-4148-6. Dostupné z: doi:10.1007/978-1-4842-4149-3
[2] WONG, Mei. Neural Networks with Python: Design CNNs, Transformers, GANs and capsule networks using tensorflow and keras. B.m.: GitforGits, 2023. ISBN 978-81-19-17789-9.
[3] LOPEZ, J.M., Javier GONZÁLEZ-JIMÉNEZ, Cipriano GALINDO a J. CABELLO. A versatile low-cost car plate recognition system [online]. 2007. ISBN 978-1-4244-0778-1. Dostupné z: doi:10.1109/ISSPA.2007.4555412
[4] MAURYA, Richa. Application of Restful APIs in IOT: A Review. International Journal for Research in Applied Science and Engineering Technology

[online]. 2021, 9, 145-151. Dostupné z: doi:10.22214/ijraset.2021.33013

Vedoucí diplomové práce:	**Ing. Ondřej Budík** Katedra informatiky
Konzultanti diplomové práce:	**Mgr. Jakub Geyer** Katedra informatiky **Ing. Rudolf Vohnout, Ph.D.** Katedra informatiky
Datum zadání diplomové práce:	**7. února 2024**
Termín odevzdání diplomové práce:	**14. dubna 2024**

doc. RNDr. Zuzana Dvořáková Líšková, Ph.D.
děkanka

JIHOČESKÁ UNIVERZITA
V ČESKÝCH BUDĚJOVICÍCH
EKONOMICKÁ FAKULTA
Studentská 13
370 05 České Budějovice

doc. RNDr. Jana Klicnarová, Ph.D.
vedoucí katedry

V Českých Budějovicích dne 7. února 2024

Prohlášení

Prohlašuji, že svou diplomovou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury. Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své diplomové práce, a to – v nezkrácené podobě vzniklé vypuštěním vyznačených částí archivovaných Ekonomickou fakultou – elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

Datum Podpis studenta

Poděkování

Tímto bych rád poděkoval panu Ing. Ondřeji Budíkovi za odborné rady a trpělivost při psaní této diplomové práce. Dále bych rád poděkoval panu Petru Zichovi z Technického úseku a správy budov Přírodovědecké fakulty za pomoc s přípravou brány pro testování vyvíjeného systému.

Obsah

1 Úvod

Moderní technologie a automatizace stále více ovlivňují každodenní život, výjimkou není ani oblast parkování. Tato závěrečná práce se zaměřuje na vývoj a implementaci systému rozpoznávání registračních značek pro řízení a správu univerzitního parkoviště. Cílem tohoto systému je efektivně a spolehlivě řídit přístup na parkoviště, snížit počet neoprávněných vjezdů a zvýšit celkové zabezpečení parkoviště.

V teoretické části práce byl proveden podrobný průzkum nízkonákladového výpočetního hardwaru s důrazem na technologie jako jsou Raspberry Pi, NVIDIA Jetson a Asus Tinker Board. Dále se práce zabývá analýzou různých typů kamer, od kamerových modulů například od Raspberry Pi, přes IP kamery až po specializované kamery s funkcí automatického rozpoznávání registračních značek (ANPR). Následující kapitola se věnuje detekci objektů pomocí neuronových sítí, jejich principům a dostupným frameworkům, jako jsou TensorFlow a PyTorch. Poslední část teoretické části se zaměřuje na zpracování obrazu pomocí optického rozpoznávání znaků (OCR), včetně vysvětlení principu fungování a přehledu vhodných frameworků.

Praktická část práce je zahájena popisem technického zázemí a procesu implementace. Nejprve je představena brána zvoleného parkoviště spolu s popisem stávajícího stavu řešení vjezdu a výjezdu z parkoviště, jakož i stávajícího napájení a síťové infrastruktury. Následují návrhy pro nasazení, kde jsou specifikovány technické požadavky a procesy testovacího nasazení. v této části je také podrobně představen vybraný hardware. Po této úvodní části se práce věnuje akvizici dat a následně kapitole o trénování neuronové sítě pro detekci objektů. Dále je prezentována volba OCR, včetně nastavení kamery a image processingu potřebného pro získání čistého obrazu, který OCR efektivně zpracuje. v této části je rovněž obsažena kapitola o testování vybraných OCR systémů a jejich výsledcích. Následující kapitola se zabývá samotnou aplikací, kde je představen vybraný softwarový základ a jazyky použité pro vývoj, a popis aplikace je doplněn schématem a popisem webového prostředí pro validaci parkovacích oprávnění. Předposlední kapitola pojednává o implementaci celého řešení na vybraném Raspberry Pi. Práce je zakončena srovnáním vyvinutého systému s komerčním systémem vybrané kamery, což ilustruje klíčové přínosy a výsledky této práce.

Tato práce úzce souvisí s projektem "Campus parking management", který realizoval tým z předmětu IS projekt na Jihočeské univerzitě (JU). Cílem projektu bylo vytvořit přístupový ekosystém pro parkovací prostory JU, který na základě různých identifikátorů (osobní/univerzitní) povoluje nebo zamítá vjezd na parkoviště. Součástí projektu je také vytvoření webové aplikace pro uživatele a administrátory, která umožňuje správu uživatelských účtů a parkovišť. Tento projekt a diplomová práce se vzájemně doplňují a podporují, přičemž tato práce se zaměřuje specificky na hardware a rozpoznávání registračních značek, což je klíčová komponenta celého systému.

2 Dostupné hardwarové možnosti zpracování

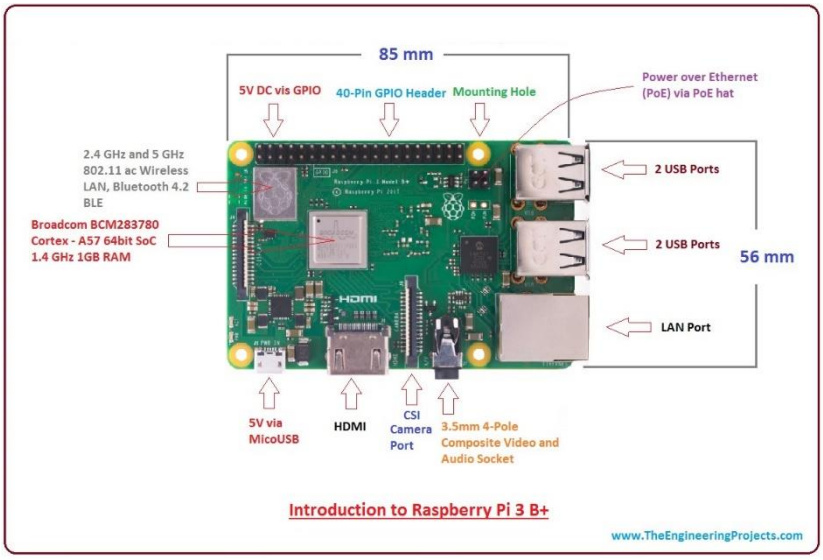
Při výběru vhodného hardwaru pro zpracování dat pro účely tohoto projektu je klíčové vybrat vhodný jednodeskový počítač. Tato kapitola se zaměřuje na analýzu dostupných možností zpracování dat s využitím jednodeskových počítačů. Zmíněné ceny jednotlivých produktů jsou datovány k únoru 2024.

2.1 Raspberry

Jednou z možností je jednodeskový počítač od nadace Raspberry Pi Foundation. Tato nadace poskytuje širokou škálu modelů jednodeskových počítačů využitelných pro různorodé projekty prvotně zaměřené na vzdělávání v oblasti informatiky. Raspberry Pi je hojně využíváno pro učení základních programovacích dovedností nebo domácí automatizaci, popřípadě může být využit i pro průmyslové aplikace. Tento velmi malý počítač běžící na systému Linux, je levným počítačem poskytujícím rozmanitý výkon [1].

2.1.1 Raspberry Pi 3 B+

Tento model bylpředstaven v roce 2018 a byl nástupcem verze Raspberry Pi 3 B, které nemělo dvoupásmové Wi-Fi s podporou 2,4 i 5 GHz, zvýšení rychlosti čtyřjádrového 64bitového CPU, a to z původních 1,2 GHz na 1,4 GHz. Tento jednodeskový počítač je vybaven GPU VideoCore IV operující na frekvenci 400 MHz. Port Ethernet dosahuje rychlosti 300 Mbit/s oproti předchozí verzi, která poskytovala pouze 100 Mbit/s. Bluetooth také dostalo vylepšení, a to z verze 4.1 LE na verzi 4.2 s podporou BLE [2, 3].

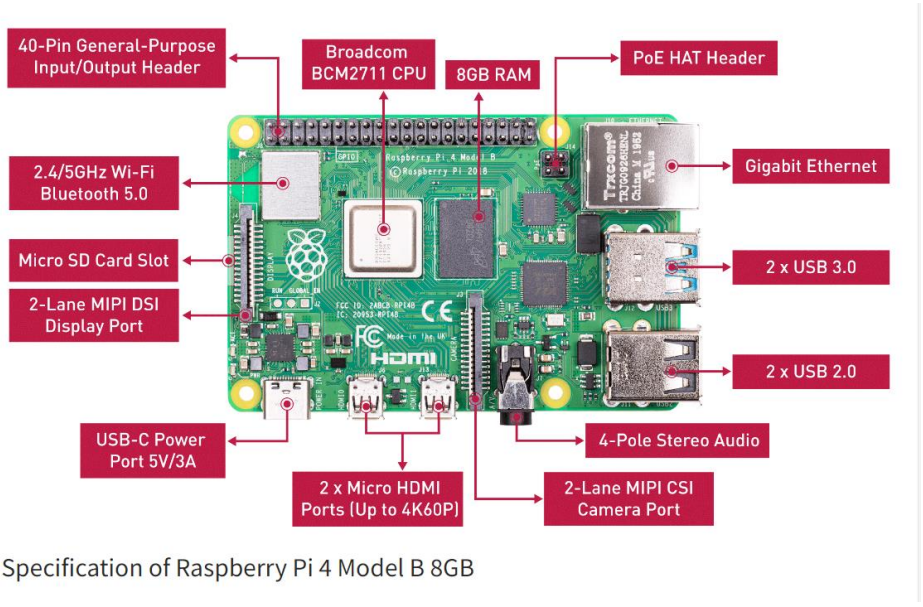


Obrázek 1: Schéma Raspberry Pi 3B+ (převzato z [2])

Model obsahuje slot pro microSD kartu a 4 USB 2.0 porty. Jako standard pro tyto modely je zde 40pinové GPIO a porty CSI pro připojení fotoaparátu nebo DSI pro připojení displeje. Pro připojení monitoru je tu klasické HDMI a pro audio se používá 3,5 mm audio jack. Velkým omezením tohoto modelu je velikost RAM pouze 1 GB LPDDR2 SDRAM, přičemž v jiných variantách se neprodává ani standardní model Raspberry Pi 3 B. Napájení může být prostřednictvím HAT modulu přes PoE nebo zdrojem s 2,5 v přes 5V MicroUSB konektor [2]. Tento model je cenově velmi dostupný, a to do 1000 Kč. [3]

2.1.2 Raspberry Pi 4 B

Do nedávna nejnovější model této řady. Tento počítač má čtyřjádrový 64bitový procesor s frekvencí 1,5 GHz s GPU VideoCore VI s 600 MHz. Tento model, se prodává ve 4 verzích s rozdílnou pamětí a to s 1, 2, 4 a 8 GB DDR4 SDRAM [4].



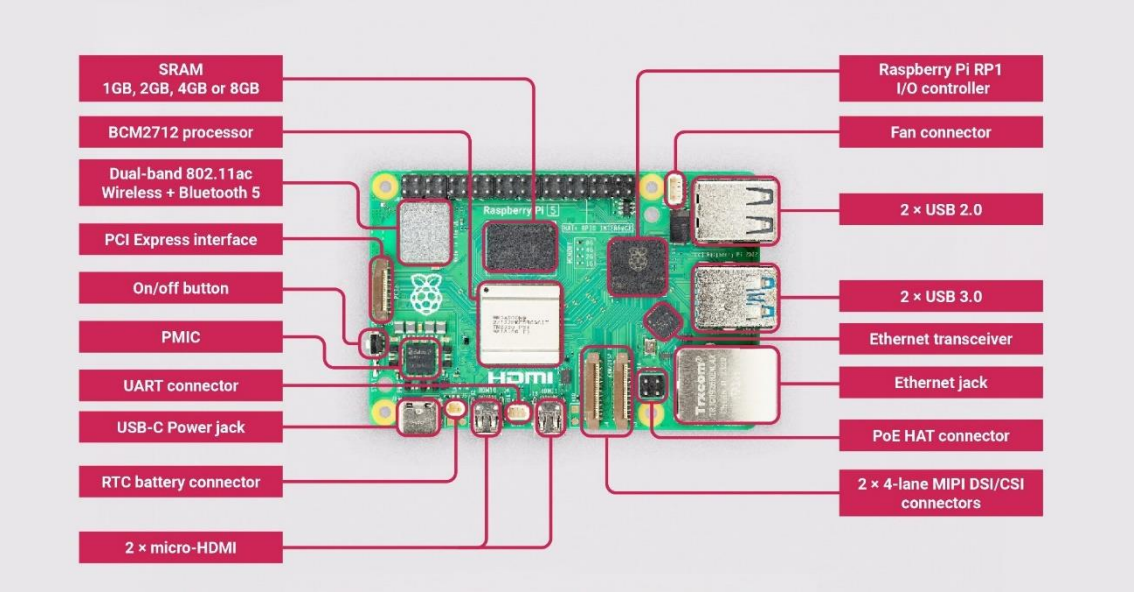
Obrázek 2: Specifikace Raspberry Pi 4 (převzato z [5])

Model má k dispozici dva USB porty 3.0 a dva porty 2.0 pro připojení například klávesnice nebo myši. Samozřejmostí jsou dvoupásmový Wi-Fi standardu 802.11ac, Bluetooth 5.0 a Gigabitový Ethernet. Pro video výstup má dva konektory micro HDMI s možností 4K při 60Hz, avšak při použití obou konektorů se frekvence sníží o polovinu. Audio výstup může být v rámci zmíněného HDMI nebo dostupného 3,5 mm jack. Uložiště je řešeno pomocí MicroSD karty a pro napájení je použit port USB-C, ale je také možné napájení prostřednictvím PoE. Specifické porty pro Raspberry jsou zde

40 GPIO pinů, CPI pro připojení modulu kamery nebo popřípadě DSI port pro displeje [4, 5]. Cenová dostupnost tohoto modelu je velmi příznivá, avšak záleží na typu a zvolené velikosti RAM, od čehož se odvíjí cena od 900 do 2 000 Kč [3].

2.1.3 Raspberry Pi 5

Nyní je tu nejnovější verze, která přišla po 4 letech od předchozího modelu. Zásadní změnou tohoto modelu je opravdu značné zvýšení výkonu. Tento model je oproti Raspberry Pi 4 až 3krát rychlejší a také 2–3krát rychlejší ve výkonu CPU a GPU.



Obrázek 3: Specifikace Raspberry Pi 5 (převzato z [7])

Raspberry Pi 5 přichází s novém 64bitovým čtyřjádrovým procesorem pracujícím na frekvenci 2,4 GHz, což je značný rozdíl oproti 1,5 GHz staršího modelu. Grafický výkon byl zlepšen na 800MHz VideoCore VII GPU s možností obsluhy dvou 4K displejů připojených přes HDMI s 60 snímky za sekundu současně [6]. Tato verze Raspberry je cenově podobná svému předchůdci a lze ji zakoupit v cenách do 2200 Kč [3].

Tento model je dostupný pouze ve dvou variantách, a to s 4 nebo 8 GB paměti. Největší inovací je čip RP1 plně vyhrazený pro I/O, čímž se značně sníží zátěž CPU. Součástí Raspberry Pi 5 je také v dnešní době velmi používaný PCIe konektor, avšak s rozdílným připojením, kde se používá plochý kabel a připojení k HAT namísto standardního M.2 konektoru. Přidány byly také porty MIPI, ke kterým se připojují kamery nebo displeje. Nelze opomenout fakt, že na tomto modelu jsou zrychleny i microSD karty, a to z čtecí rychlosti 40–50 Mb/s na 80-90 Mb/s, což je dvojnásobné zrychlení oproti Raspberry Pi 4. Tentokrát je vynechán audio port 3,5 mm, ale je

přidáno tlačítko na napájení a také je zabudované RTC tudíž není potřeba načítat aktuální čas z internetu jak tomu bylo u starší verze [6, 7].

2.2 NVIDIA Jetson

V oblasti velmi výkonných jednodeskových počítačů vyniká skupina NVIDIA Jetson, která je speciálně navržená pro práci s neuronovými sítěmi a umělou inteligencí a využívá grafické procesory od tohoto amerického giganta. [8]

2.2.1 NVIDIA Jetson Nano

Mikropočítač Nano z řady NVIDIA Jetson je vhodnou volbou pro práci s neuronovými sítěmi zaměřenou na klasifikaci obrazu, detekci objektů, segmentaci či zpracovávání řeči. Dostupný je čtyřjádrový procesor ARM A57 s frekvencí 1,43 GHz. Jak název tohoto mikropočítače naznačuje, je zde použitý čip NVIDIA, a to 128jádrový Maxwell, který poskytuje značný výkon pro neuronové sítě [9].

Obrázek 4: Jetson Nano Developer Kit (převzato z [10])

K dispozici je ve dvou verzích, a to s 2 a 4 GB operační paměti LPDDR4. Mezi I/O komponenty patří 4 porty USB 3.0 a jeden USB 2.0 Micro-B, port HDMI a jako většina těchto sad 40 pinů GPIO. Součástí je také Gigabitový Ethernet a podle zvolené verze RAM je dostupný buďto adaptér USB 802.11ac, nebo M.2 E pro Wi-Fi. Značnou nevýhodou může být skutečnost, že napájecí zdroj je větší než samotné zařízení a napájení 10 W nezbytné pro kompletní výkon, protože se při 5 W vypnou 2 ze čtyř jader. [8, 9]. Cenově se tato řada pohybuje mnohem výše než jiné uvedené značky, a tak cena přesahuje hranici 7500 Kč. [10]

2.2.2 NVIDIA Jetson Orin Nano

Tento kompaktní modul se vyznačuje až 80krát větším výkonem než jeho předchůdce Jetson Nano. Tato vývojová sada je vybavena 64bitovým 6 jádrovým procesorem RM Cortex A78AE s frekvencí 1.5 GHz. Model Orin je dostupný s velikostí 8 GB a 1024 jádrovým Nvidia Ampere s 32 jádry Tensor. Operační paměť má 8 GB 128bit LPDDR5 [11, 12].

Obrázek 5: Nvidia Jetson Orin Nano (převzato z [12])

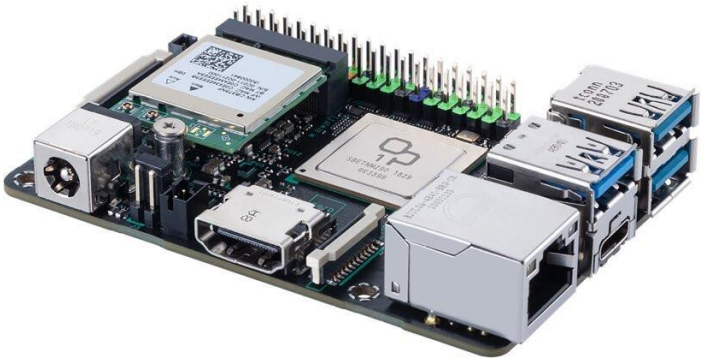
Orin Nano, jako ostatní vývojářské sady od společnosti NVIDIA, mají slot pro microSD kartu a vnitřní uložiště 128 GB NVMe SSD. Mezi konektorové vybavení této desky patří Gigabitový Ethernet a 4 USB konektory 3.2 a jeden USB-C, 40pinový GPIO. Bezdrátový modul poskytující Wi-Fi a Bluetooth je přeinstalován v jednom z M.2 dostupných modulů a několik dalších portů či konektorů. Napájení je zde zprostředkováno prostřednictvím DC 19V zásuvky. Tato vývojářská sada je již značně drahá, a to přibližně za 19200 Kč [11, 12].

2.3 Asus Tinker Board

Zajímavou alternativou jednodeskových počítačů může být i rodina od společnosti ASUS, a to modely z řady Tinker Board. Tyto jednodeskové počítače jsou navrženy jako velmi spolehlivá a schopná zařízení oslovující hlavně nadšence do internetu věcí (IoT) [13].

2.3.1 Asus Tinker Board 2S

Tinker Board 2 je vybaven čtyřjádrovým 64bitovým procesorem Cortex-A53 pracujícím s frekvencí 1.5 GHz a dvoujádrovým jádrovým 64bitovým procesorem Cortex-A72 s operující frekvencí 2.0 GHz. Tinker Board 2S je vybaven 800 MHz ArmMali-T860 GPU, který podporuje frameworky OpenGL ES 3.1/3.0 a mnoho dalších pro 2D/3D grafické aplikace [14].

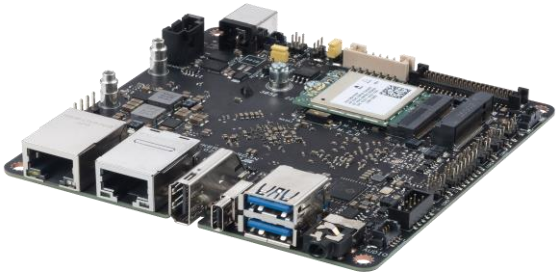


Obrázek 6: ASUS Tinker Board 2S (převzato z [3])

Tento model podporuje připojení až dvou displejů s rozlišením 4K v UHD. Možnosti připojení jsou přes HDMI nebo DisplayPort DSI či USB-C. Bezdrátové technologie zde nesmí chybět. Jsou k dispozici Wi-Fi 802.11ac s dvěma dvoupásmovými anténami s podporou 2,4 a 5 GHz a podpora Bluetooth 5.0. Tinker Board 2S se prodává ve dvou verzích, a to s 2 nebo 4 GB dvoukanálová operační paměť LPDDR4. Uložiště je zprostředkováno přes 16 GB eMMC s možností přidání MicroSD karty. Součástí je i hlava se 40 piny a 3 USB 3.2 konektory a 1 USB-C. Přes GPIO piny je možnost připojení pro audio. Napájení přes 12~19V DC Jack konektor 5.5/2.5 mm [14]. Cena takového jednodeskového počítače se pohybuje přibližně okolo 2700 Kč [3].

2.3.2 Asus Tinker Board 3N

Další jednodeskový počítač od společnosti ASUS je vybaven 64bitovým 4jádrovým procesorem ARM Cortex A55 s integrovanou GPU ARM Mali G52 s pracující frekvencí 850 MHz. Operační paměť je zde již standardní dvoukanálová LPDDR4, která je u tohoto modelu dostupná v různých velikostech, a to ve 2, 4 nebo 8 GB. Totéž platí o velikosti uložiště, kde buď to 32 nebo 64 GB s možností slotu MicroSD karty [15, 16].



Obrázek 7: Tinker Board 3N (převzato z [16])

Mezi I/O komponenty této desky patří dva standartdní Ethernet porty, z nichž jeden podporuje napájení PoE, dalším běžným portem je i HDMI pro připojení monitoru nebo televize s možností 4K s 60Hz. Wi-Fi a Bluetooth je stejně jako u předchozího modelu jsou připojeny přes M.2 avšak modul pro tyto bezdrátové připojení se musí dokoupit odděleně. Mezi USB porty jsou zde dostupné běžné USB 3.2 nebo 2.0 porty a jeden USB-C. Pro audio má přítomný audio Jack. Zajímavostí je zde přítomnost slotu pro SIM kartu velikosti nano SIM. Součástí na této desce je i hlava se sadou pinů, běžně dostupnými napříč jednodeskovými počítači ať se jedná o GPIO nebo UART piny. Napájení je zajištěno pomocí DC 5.5/2.5 mm Jack konektoru nebo prostřednictvím 4 napájecích pinů s napětím 12~24 v [15].

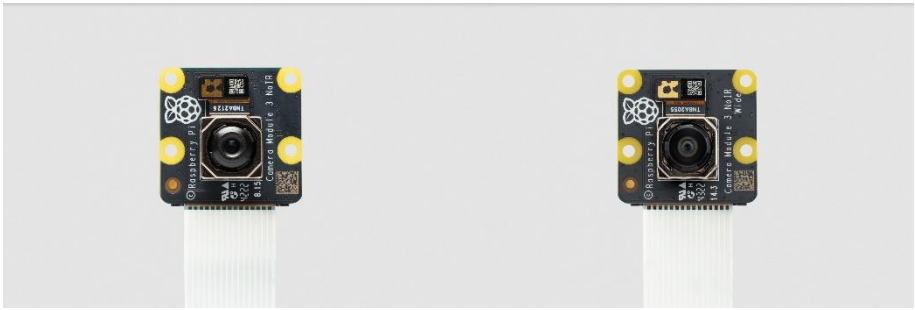
3 Dostupné hardwarové možnosti akvizice snímků

Pro návrh systému, jehož jednou ze základních funkcionalit je akvizice snímků, hraje zásadní roli volba kamerového systému a zařízení. v této kapitole budou popsány různé dostupné možnosti, počínaje kamerovými moduly dostupnými například od Raspberry Pi přes IP kamery až po specializované ALPR kamery pro detekci registračních značek automobilů. Cílem této kapitoly je poskytnout celistvý pohled na dostupný hardware. Zmíněné ceny jednotlivých produktů jsou datovány k únoru 2024.

3.1 Kamerové moduly

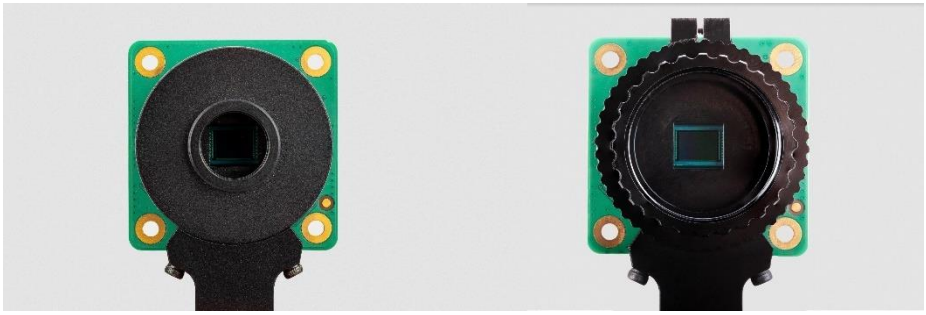
Vhodnou možností pro akvizici snímků, je-li zohledněna hlavně cena, jsou kamerové moduly s konektory CSI z rodiny Raspberry Pi, které podporuje většina jednodeskových počítačů.

Na trhu jsou dostupné tři hlavní řady kamerových modulů z rodiny Raspberry Pi od původního pěti megapixelový model předcházející modelu Camera Module 2 s osmi mega pixely až po dvanácti megapixelovým Camera Module 3 uvedený na trh v průběhu roku 2023. Poslední modul z této hlavní řady je vybaven snímačem od společnosti Sony IMX708 s rozlišením 11,9 megapixelů a poskytuje vysokou kvalitu obrazu. Tento model je k dispozici ve variantách pro viditelné světlo i infračervené záření, což je užitečné při různých světelných podmínkách. Má také rozšíření Field of View (FoV) umožňující různý úhel záběru kamery.



Obrázek 8: Ukázka kamerového modulu 3 (převzato z [17])

Nejnovější kamerový modul pro Raspberry Pi, Global Shutter, disponuje speciální závěrkou, která umožňuje zachytit světlo ze všech pixelů současně. Tato vlastnost jej činí ideální volbou pro snímání pohybujících se objektů bez rozmazání, ideální volbou pro **precizní** snímání pohybujících se objektů bez rozmazání. Pokud však upřednostňujeme maximální kvalitu obrazu, pak může být tou správnou volbou HQ Camera Module.



Obrázek 9: Ukázka modulu HQ kamera (převzato z [17])

Tento modul s rozlišením dvanácti megapixelů (4056x3040 pixelů) a dynamickým rozsahem 65 dB umožňuje pořizovat detailní a živé snímky. s podporou výměnných objektivů typu C a CS je možné vybrat si ten, který nejlépe vyhovuje potřebám konkrétního projektu. Bez ohledu na to, zda je preferován širokoúhlý objektiv pro zachycení rozsáhlých scén, teleobjektiv pro přiblížení vzdálených objektů nebo makro objektiv pro detailní snímky zblízka, HQ Camera Module poskytuje maximální flexibilitu.

S HQ Camera Module a vhodným objektivem lze dosáhnout profesionálních výsledků v široké škále aplikací, od fotografování a natáčení videa ve vysokém rozlišení až po průmyslové a vědecké použití. Jsou cenově poměrně dostupné, a to od 570 do 1200 Kč [17].

3.2 IP kamery

Pro přenos video a audio dat přes počítačovou síť jsou vhodné digitální IP kamery. Zásadní rozdíl oproti analogovým kamerám spočívá v nepoužívání koaxiálního kabelu, ale v použití ethernet kabelu či bezdrátového připojení k síti. Tento způsob připojení umožňuje vzdálený přístup ke kameře a sledování záznamu v reálném čase odkudkoliv. IP kamery také nabízejí výrazně vyšší kvalitu obrazu ve srovnání s analogovými kamerami. Bohužel zde je již vyšší cena a mohou vyžadovat složitější konfiguraci

kamery. IP kamery mají širokou škálu použití, včetně monitorování budov, areálů, vnitřních prostor a dopravy [18, 19].

Do této kategorie spadá velká spousta typů kamer. Prvním příkladem by mohla být kamera Hikvision DS-2CD2346G2-ISU/SL. Velmi kompaktní kamera s rozlišením 4MP a snímačem 1/3 Progressive Scan CMOS. Díky vysoké citlivosti (barevný lux 0,0028 a černobílý 0,001) a kompresi obrazu H.265+ zachytí kamera detailní obraz i při zhoršených světelných podmínkách. Vyvážený obraz v oblastech s velkým kontrastem světla a stínu zajišťuje funkce WDR [1](120 dB) a také funkce 3D Digital Noise Reduction umožní snížení šumu v obraze a tím zlepšit jeho kvalitu.

Obrázek 10: Kamera Hikvision DS-2CD2346G2-ISU/SL (převzato z [20])

Varifokální objektiv s ohniskovou vzdáleností 2,8 – 12 mm přizpůsobuje zorné pole podle potřeb uživatele. Kamera je vybavena nočním viděním až do 30 metrů pro zajištění čitelnosti snímků i za tmy. Pokud by kamera měla pokrýt oblast vstupů do budov, jsou dostupné funkce i pro detekci pohybu či rozpoznání tváře. Kamera je chráněna kovovým pouzdrem proti poškození a disponuje voděodolností IP67 [20]. Pořizovací cena zařízení se pohybuje v cenové relaci okolo 5000 Kč. [21]

[1] WDR – Wide Dynamic Range slouží pro vyrovnávání kontrastu v obraze

Obrázek 11: Dahua IPC-HFW5442E-ZE-S3(převzato z [22])

Další příklad kamery z této skupiny je Dahua IPC-HFW5442E-ZE-S3. Jedná se o kompaktní bullet kameru s čtyř megapixelovým rozlišením a 1/1,8 Progressive Scan CMOS snímačem. Oproti předchozímu příkladu kamery má tato zařízení ještě vyšší citlivost (0,00014 lux černobílé a 0,00028 barevné). Opět je zde komprese obrazu H.265+ pro možnost zachycení detailnějšího obrazu během zhoršených podmínek. Součástí jsou i zmiňované funkce WDR a 3D Digital Noise Reduction.

U této kamery je podpora napájení přes PoE pomocí Ethernet kabelu a ochranným krytem proti poškození s vodotěsností IP67. Pro přizpůsobení zorného pole je kamera vybavena motorizovaným objektivem s ohniskovou vzdáleností 2,7 – 12 mm. Noční vidění s technologií Starlight+ dosahuje až 50 metrů. Kamera disponuje funkcemi pro detekci pohybu, rozpoznání tváří či inteligentní analýzu obrazu (IVS) [22]. Cena této kamery začíná od 13 000 Kč. [23]

3.3 Specializované ANPR kamery

Kamery vybavené speciální funkcí ANPR, tedy automatickým rozpoznáváním registračních značek automobilů, jsou schopné detekovat a číst tyto registrační značky a odesílat zpracovaný výsledek do systémů řízení kontroly přístupu. Někdy se označení ANPR zaměňuje i za ALPR[2] nebo jen LPR[3].

Fungují na principu snímání obrazu vozidel a zachycení jejich registračních značek. Po zachycení registrační značky použijí software pro analýzu textu (OCR) a detekují její informace. Získané informace se porovnávají s databází a dále vyhodnocují. Kamery s touto funkcionalitou často umí také rozpoznat barvu, značku či rychlost vozidla. Využití těchto kamer může být v dopravní bezpečnosti, logistice nebo pro správu parkování [24, 25].

[2] ALPR – Automatic License Plate Recognition
[3] LPR – License Plate Recognition

První ukázka může být kamera Hikvision IDS-2CD7A46G0/P-IZHS s čtyř megapixelovým rozlišením a motorizovaným objektivem 2,8 - 12 mm s clonovým číslem f/1.6 pro poskytnutí dostatku světla. Důležité je také zmínit, že kamera umožnuje regulaci zesílení signálu, a to v rozmezí 0-27 dB, pro zlepšení kvality snímků za nízké úrovně světla. Kompresi zde zajišťují kodeky H.265 nebo H.264 s nastavitelným bitratem až 32 Mbps. Pro zachycení ostrého snímku i za pohybu objektu se stará závěrka s rychlostí od 1/25–1/100000 sekundy. Kamera je vybavena širokou škálou funkcí pro rozpoznávání registračních značek pomocí technologie Deep Learning a tak dosahuje velmi vysoké přesnosti při detekci registračních značek i za náročných povětrnostních či světelných podmínek.

Obrázek 12: Hikvision iDS-2CD7A46G0/P-IZHS (převzato z [26])

Kamera má rychlé rozpoznávání registračních značek a to až 20 značek za sekundu při maximální rychlosti vozidla 60 km/h a s přesností vyšší než 95 %. u této detekce lze nastavit i různé parametry, jako je detekce specifických čísel a státních příslušností. Velkou výhodou u této kamery je snadná integrace do již existujícího systému či volitelnost funkcí, jako je rozpoznávání barvy a typy vozidla, nebo hlášení o podezřelých aktivitách [26]. Cena této kamery se pohybuje okolo 25 000 Kč. [27]

Dalším příkladem kamery spadající do této kategorie je Dahua IPC-HFW71242H-Z-X. Jedná se o síťovou kameru s 12MP 1/1.7 CMOS snímačem, který zajišťuje ostré a detailní snímky. Funkční výbavou přesahuje předchozí kameru například detekcí a rozpoznáváním tváří a počítáním osob. Funkce ANPR je téměř totožná jako s výše uvedenou kamerou od společnosti Hikvision, s výjimkou procentuální úspěšnosti detekcí, která je vyšší než 97 %. Tato kamera také zvládá detekci 20 značek za sekundu při maximální rychlosti vozidla 60 km/h [28].

Obrázek 13: Dahua IPC-HFW71242H-Z-X (převzato z [28])

Kamera využívá komprese H.265/H.264 s nastavitelným bitratem a jako tomu bylo i u předchozí kamery, pro zachycení ostrého a detailního snímku je kamera vybavena závěrkou s rychlostí až 1/100000 sekundy. Výhodou této kamery oproti předchozí je zahrnutí funkcí správy parkovacích míst, umožňující sledování obsazenosti parkoviště nebo možností detekce ochranných pomůcek na pracovišti, jako jsou helmy, ochranné brýle či respirátory. Také je vhodné zmínit systém Intelligent Video System (IVS) detekující specifické události, na které je možné nastavit alarm či jiná notifikace. [28] Díky spoustě dodatečných funkcí se cena této kamery vyšplhala až k hranici 40 000 Kč. [29]

4 Zpracování obrazu – detekce objektů

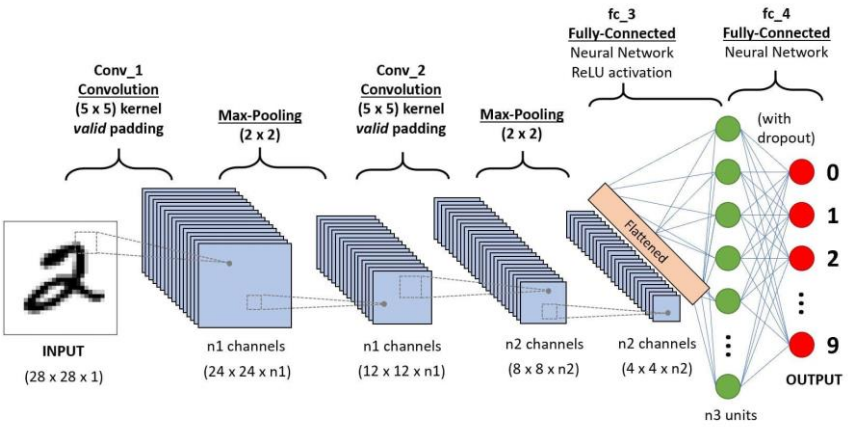
Detekce objektů je technologií počítačového vidění pro nalezení určitých objektů v obraze a určení jejich pozice. Tato technologie je klíčová pro širokou škálu aplikací, od autonomního řízení až po sledování bezpečnosti [30].

Princip detekce objektů

Detekcí objektů rozumíme proces, který zahrnuje identifikaci či lokalizaci objektů na snímku. Nejprve se snímek rozloží na individuální pixely a poté se zpracuje pomocí různých algoritmů a technik, aby se identifikovaly objekty a určily jejich pozice. Tato technologie využívá sofistikovaných metod z oblasti strojového učení a hlubokých neuronových sítí, které umožnují vysokou přesnost a efektivitu detekcí [30].

Neuronové sítě

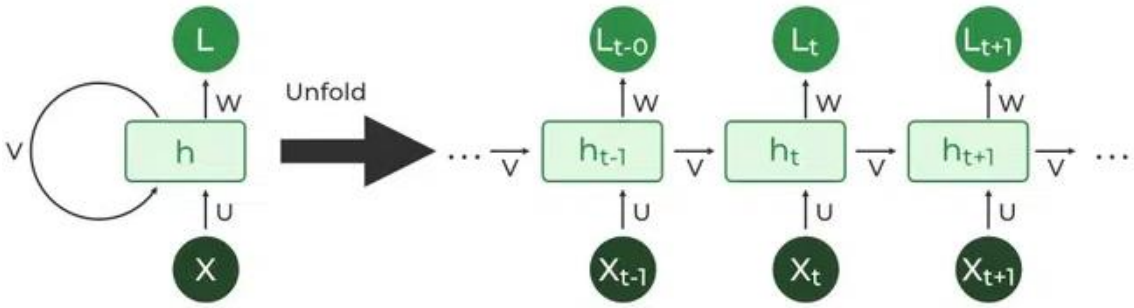
Prvním typem hlubokých neuronových sítí jsou **konvoluční neuronové sítě (CNN),** které jsou speciálně navržené pro správu a analýzu dat v podobě obrázků a videí. Jejich hlavní funkcí je využití konvolucí, tedy matematických operací, které umožnují síti extrahovat místní znaky v datech. Využití tyto sítě naleznou především při rozpoznávání obličejů, detekci objektů či segmentaci obrazu. [31, 32]



Obrázek 14: CNN sít (převzato z [31])

Znázornění na obrázku výše popisuje proces zpracování dat přes tento typ sítě. Na vstupní vrstvu vstoupí obraz uspořádaný do matice v reprezentaci pixelů. Dále postupuje na konvoluční vrstvu, která může používat filtr o velikosti 5 x 5 a provádí matematickou operaci konvoluce, jejíž schopností je získávání specifických rysů obrázku. Dále se zde objevuje Max-Pooling o velikosti filtru 2 x 2, jejíž funkcí je snižování dimenzí výstupu z konvolučních vrstev. Těchto dvou vrstev může být více za sebou s různými filtry. v konečné fázi procesu vznikne plně propojená vrstva reprezentující vrstvu, která již reprezentuje vrstvu neuronů, ve které je každý neuron je propojen se všemi ostatními z předchozí vrstvy. [33]

Druhým typem jsou **rekurentní neuronové sítě (RNN),** které jsou schopné pracovat se sekvenčními daty. Vhodným využitím této sítě může být při analýze časových řad nebo textu, protože využívá zpětných vazeb v síti k zachycení informací z předchozích krocích. Tento typ sítí se často používá pro rozpoznávání řeči, analýzu sentimentu či ve strojovém překladu. [34, 35] Jednou významnou variantou RNN sítí jsou LSTM a GRU sítě, které řeší problém zmizení gradientu a učení dlouhodobých závislostí [36].



Obrázek 15: RNN síť (převzato z [35])

Třetím typem jsou méně běžné **reciproční neuronové sítě**, které umožňující dvousměrné zpracování informací. Tento typ je vhodný pro aplikace, které vyžadují přístup k obousměrné sekvenci, například při zpracování textu, kde se význam může lišit v závislosti na předchozím i následujícím kontextu. To může být vhodné pro zpracovávání přirozeného jazyka [36].

Frameworky

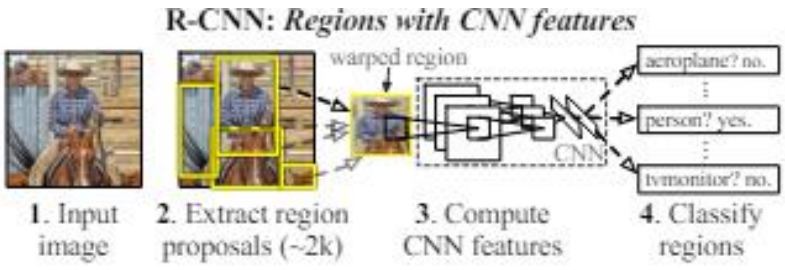
- **TensorFlow**: TensorFlow je open-source platforma pro strojové učení vyvinutá společností Google. Podporuje širokou škálu úloh strojového učení, včetně detekce objektů. Tento framework je založen na výpočetním grafu, kde uzly reprezentují matematické operace a hrany představují multidimenzionální datové matice, což umožnuje efektivní paralelní výpočty na různém hardwaru. [37].
- **PyTorch**: PyTorch je open-source platforma pro strojové učení vyvinutá společností Facebook. PyTorch podporuje širokou škálu úloh strojového učení, včetně detekce objektů. PyTorch je populární díky svému dynamickému výpočetnímu grafu a snadnému použití, což ho činí ideálním pro výzkum a vývoj nových modulů či snadnějšímu ladění v reálném čase. Velkou výhodou tohoto frameworku je i silná komunita a velká podpora různých knihoven, což z něj činní preferovanou volbou jak pro výzkum, tak pro komerční použití [38].

4.1 Modely NN k detekcím

Modely pro detekci objektů můžeme rozdělit do dvou hlavních kategorií:

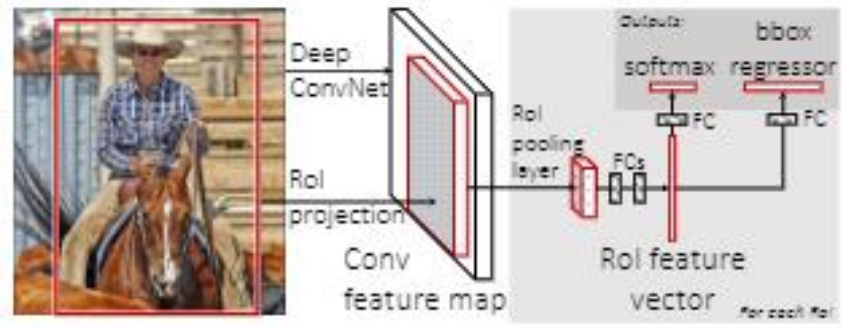
4.1.1 Region proposal-based metody

- Model **Region-based Convolutional Neural Network (R-CNN)** se používá k detekcím objektů a integruje selektivní vyhledávání k extrakci oblastí zájmu (ROI) ze snímku, a následně je klasifikuje pomocí CNN. R-CNN je schopný dosáhnou velké přesnosti, ale je výpočetně náročný a nelze ho efektivně použít [39].



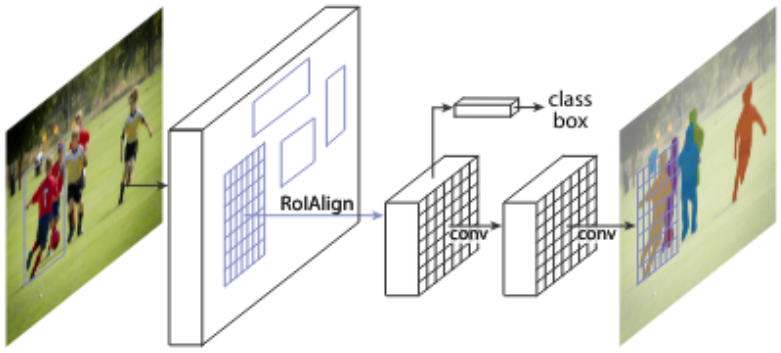
Obrázek 16: R-CNN (převzato z [39])

- **Fast R-CNN** je metoda pro detekci objektů, která oproti výše uvedené metodě značně zvyšuje rychlost a přesnost. Namísto zpracování každého regionu zájmu (ROI) individuálně tato metoda používá sdílenou konvoluční vrstvu pro extrahování vlastností z celého snímku najednou, čímž se snižuje čas potřebný k analýze. Poté se extrahují ROI ze sdílené reprezentace prostřednictvím ROI pooling vrstvy pro následnou klasifikaci. Tento proces urychluje proces detekce a zvyšuje její rychlost díky vylepšenému využití výpočetních zdrojů. [40]



Obrázek 17: Fast R-CNN (převzato z [40])

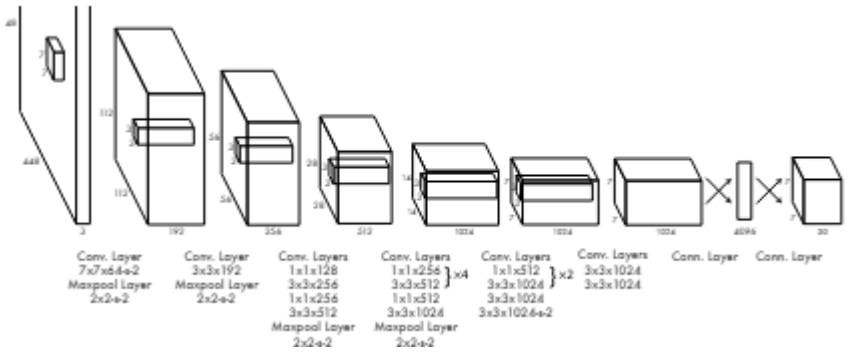
- **Mask R-CNN** je rozšířením modelu Fast R-CNN, které umožňuje detekci objektu a navíc jejich přesnou segmentaci. Zatímco jeho předchůdce ohraničuje boxy okolo objektů, tento model přidává větve předpovědí binárních mask, které detekují přesné hranice detekovaných objektů. Model je specifický použitím vrstvy ROIAlign namísto používané RIOPool, čímž se zlepšuje přesnost. Mask R-CNN je složen ze dvou struktur detekční části, která je zodpovědná za detekci objektů, a maskové části, která generuje pixelově přesné masky pro každý detekovaný objekt. [41]



Obrázek 18: Mask R-CNN (převzato z [41])

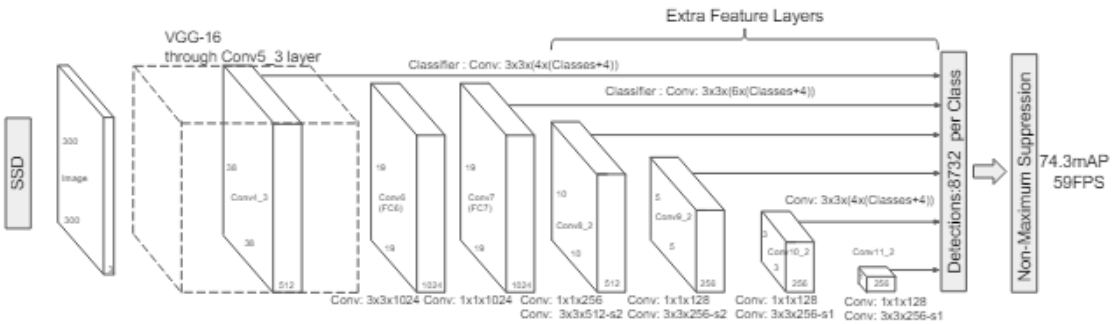
4.1.2 Classification-based metody

- **You Only Look Once (YOLO)** přichází s revolučním přístupem k detekci objektů v reálném čase, přičemž se zaměřuje hlavně na efektivitu a rychlost. Tato metoda se liší od tradičního způsobu detekce objektů tím, že namísto skládání různých částí obrazu aplikuje jednu neuronovou síť na celý zkoumaný snímek. YOLO model rozděluje vstupní obraz do mřížky a pro každou buňku této mřížky předpovídá bounding boxy a pravděpodobnosti jednotlivých tříd. Tento model se stal nejpoužívanějším pro detekci objektů v reálném čase z důvodů své vynikající přesnosti a rychlosti. [42]



Obrázek 19: YOLO architektura (převzato z [42])

- **Single Shot MultiBox Detector (SSD)** je pokročilý model poskytující kombinaci vysoké přesnosti a rychlosti detekcí, což ho činí vhodným pro širokou škálu aplikací od mobilních zařízení po složité systémy počítačového vidění v reálném čase. SSD využívá konvoluční neuronové sítě k získání vlastností z různých úrovní abstrakce a simultánně odhaduje bounding boxy a jejich příslušné třídy. Tento přístup umožňuje detekovat objekty různých velikostí a tvarů. Velkou výhodou tohoto modelu je efektivní využívání výpočetních zdrojů, čímž se dosahuje zrychlení detekčního procesu bez snížení přesnosti. [43]



Obrázek 20: SSD architektura (převzato z [43])

4.2 Principy trénování

Optimalizace je klíčová pro trénování neuronových sítí s cílem minimalizovat chybovost (loss) úpravou váhy modelu. Existuje mnoho algoritmů pro optimalizaci, avšak mezi nejpoužívanější patří Stochastic Gradient Descent (SGD), RMSprop, Adagrad a Adam. [44]

4.2.1 Stochastic Gradient Descent (SGD)

Jedná se o jeden z nejstarších algoritmů optimalizace neuronových sítí. Principem je v každém kroku aktualizovat váhy modelu podle gradientu chyby jednoho či více vzorků z tréninkových dat. Výhodou algoritmu je jeho jednoduchost a s tím i nízké výpočetní nároky, avšak nevýhodou je možnost oscilace. Vzorec pro výpočet aktualizace vah:

$$\theta = \theta - \eta \nabla\_\theta J(\theta; x^{(i)}; y^{(i)})$$

kde η reprezentuje rychlost učení, θ jsou váhy modelu, ∇θJ vyjadřuje gradient chyby, J je ztrátová funkce a prvky x^i a y^i zastupují tréninkové vzorky. [44]

4.2.2 Adaptive Moment Estimation (Adam)

Adam je optimalizační algoritmus kombinující výhody algoritmů RMSprop a Adagrad. Je navržen tak, aby přizpůsoboval rychlost učení podle každé váhy modelu. Využívá odhady prvních dvou momentů gradientů k dynamické úpravě rychlosti učení každé váhy. Vzorec je:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

$$m'_t = \frac{m_t}{1 - \beta_1^t}$$

$$v'_t = \frac{v_t}{1 - \beta_2^t}$$

$$\theta = \theta - \eta \frac{m'_t}{\sqrt{v'_t + \epsilon}}$$

Přičemž m_t a v^t reprezentují první dva odhady momentů gradientu, β_1 a β_2 jsou hyperparametry, g_t značí gradient, η je učební rychlost a ϵ značí konstantu pro stabilitu. [44]

4.2.3 Adaptive Gradient Algorithm (Adagrad)

Algoritmus Adagrad dynamicky přizpůsobuje rychlost učení pro váhy na základě sumy minulých gradientů. Tento přístup je vhodný pro problém s řídkými daty, kdy se sníží učební rychlost pro často se vyskytující parametry, a naopak se zvýší u méně častých.

$$\theta = \theta - \frac{\eta}{\sqrt{G_t + \epsilon}} g_t$$

Kde G_t je diagonální matice reprezentující součet druhých mocnin gradientů, g_t je gradient, η značí učební rychlost a ϵ je numerická konstanta. [44]

4.2.4 Root Mean Square Propagation (RMSprop)

Optimalizační algoritmus s podobnými vlastnostmi jako Adgrad, ale umí zamezit problémům s neustálým snižováním rychlosti a tím udržuje klouzavý průměr druhých mocnin gradientů s použitým průměrem k normalizaci učební rychlosti, což zajistí vetší stabilitu a rychlost konvergence:

$$E[g^2]_t = \beta E[g^2]_{t-1} + (1 - \beta) g_t^2$$

$$\theta = \theta - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t$$

$E[g^2]_t$ je klouzavý průměr druhých mocnin gradientů, β reprezentuje koeficient pro klouzavý poměr, η je učební rychlost a ϵ je zde numerická konstanta pro stabilitu. [44]

4.3 Metriky vyhodnocení funkčnosti modelu

Vhodný výběr metriky pro vyhodnocení modelu může být klíčový. Každá metrika nám poskytne jiný pohled na výkonost modelu. Pro řešení detekce registračních značek automobilů pro nás může být podstatné vyvážení mezi správnou detekcí a minimalizací chyb ale také přesností.

4.3.1 Accuracy (přesnost)

Jedná se o jednu z nejjednodušších metrik, která uvádí podíl správně klasifikovaných případů, tedy „True Positives“ a „True Negatives“, z celkového počtu daných případů. v potaz se berou následující stavy:

- True Positives (TP) – správně předpovězené pozitivní případy,
- False Positives (FP) – špatně předpovězené pozitivní případy,
- True Negatives (TN) – správně předpovězené negativní případy,

- False Negatives (FN) – špatně předpovězené negativní případy.

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

Pro projekt detekce registračních značek automobilů, kterým se zabývá tato práce, by mohla být tato metrika zavádějící. Pokud by dataset obsahoval mnoho negativních vzorků a málo pozitivních, hrozilo by dosažení vysoké přesnosti i při nesprávné detekci poznávací značky vozidla. [45, 46]

4.3.2 Precision (preciznost)

Precision je metrika měřící podíl instancí klasifikovaných jako daná třída, tedy hodnoty TP z celkového počtu instancí. Matematicky je vyjádřena jako:

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

Tato metrika je vhodná v situacích, kdy je důraz kladen na minimalizaci počtu falešně pozitivních detekcí. Pro případ detekce registračních značek automobilů by to znamenalo, že model musí být schopný identifikovat značky a zamezit chybným detekcím. [45, 46]

4.3.3 Recall (senzitivita)

Další metrikou máme senzitivitu tedy podíl správně detekovaných instancí TP z celkového počtu skutečných instancí TP v součtu s FN Matematicky vyjádřeno níže. [45, 46]

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

4.3.4 F1 Score

Jedná se o harmonický průměr metrik precision a recall, čímž poskytuje vyvážené váhy mezi těmito dvěma měřeními.

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

F1 score by mohla být vhodnou volbou pro použití u detekcí registračních značek, protože se tato metrika snaží najít rovnováhu mezi správnou detekcí a minimalizací falešně pozitivních detekcí. [45, 46]

4.4 Anotace

Anotace, známá také jako labeling, je proces, během kterého se objektům na obrázcích, zvuku či videu přiřazují štítky označující názvy daných objektů. Tato anotace objektů umožnuje modelům počítačového vidění učit se identifikovat jednotlivé třídy na datech bez předchozí klasifikace. [47]

4.4.1 Princip

Hlavním cílem celého procesu anotace obrázků je vytvoření datových sad pro následné učení neuronových sítí. Tyto datové sady se posléze rozdělují na trénovací, testovací a validační sady, které se používají k trénování modelu pro detekci nebo jeho následnou validaci. Existují tři základní techniky pro anotaci dat: [47]

1. **Manuální označení dat:**

   Jedná se o velmi časově náročnou techniku, zejména pokud se aplikuje na velký dataset. Anotace zde může být relevantnější a mnohem přesnější díky precizní ruční práci datových vědců.

2. **Poloautomatická anotace:**

   Spojení lidské síly s automatizovanými technikami pro vytvoření spolehlivé datové sady. v principu je část dat označená datovými vědci a tato data slouží jako šablona pro automatizované označování. Dále se provede kontrola, kde se zjistí nepřesné anotace, a poté se opět ručně zasahuje do dat a tyto nepřesně označená data opraví. Zde je potřeba cílit na vysokou přesnost u první části ručního označování za účelem ovlivnění části automatické anotace.

3. **Automatická anotace:**

   Využívá algoritmy umělé inteligence naučené pro anotaci vybraných typů dat dle preferencí. Tato technika se hodí pro rozsáhlé datové sady a stále vyžaduje nějakou odbornou kontrolu od vyškoleného specialisty pro eliminaci nepřesností sytému pro anotaci. [48]

4.4.2 Kvalitní anotace

Výběr správné techniky pro anotaci závisí na daném úkolu a typu dat. Existuje několik druhů anotačních technik, mezi které patří:

- Bounding Boxes – označení objektů pomocí obdélníků, které je vhodné pro detekci objektů,

- Semantic Segmentation – označení každého pixelu konkrétní třídou objektu,
- Instance Segmentation – rozlišuje jednotlivé objekty v obrázku, což zvyšuje složitost oproti sémantické segmentaci,
- Landmarks – konkrétní a přesně dané body objektů v obraze, například pro určení polohy,
- Čáry a křivky – vhodné pro učení modelů zaměřujících se na tvary objektů, například pro rozpoznávání obličeje. [49, 50]

Anotace je zásadní krok v přípravě dat pro různé úkoly v oblasti počítačového vidění, jako jsou klasifikace, detekce a segmentace. Každý z těchto úkolů vyžaduje specifický přístup k anotacím.

Klasifikace

Pro klasifikaci je nejjednodušší způsob anotace rozdělení obrázků do složek podle tříd. Každá složka obsahuje obrázky jedné třídy, například složka „pes“ obsahuje všechny obrázky psů a složka „kočka“ všechny obrázky koček. Tento přístup umožňuje jednoduché trénování klasifikačních modelů, které se učí rozpoznávat rozdíly mezi třídami na základě obrázků ve složkách. Tento postup anotace je efektivní pro úlohy, kde je cílem pouze rozlišit mezi několika třídami objektů.

Detekce

Pro detekci objektů je hlavní metodou nalezení bounding boxů, tedy označení hledaného objektu pomyslným obdélníkem. Proces detekce se skládá z následujících kroků:

1. **Identifikace objektů zájmu**

   Nejprve je zapotřebí identifikovat všechny objekty zájmu na snímcích a označit jejich bounding boxy. Tyto boxy mají za úkol ohraničit celý objekt, tedy je nezbytné dbát na velmi přesné ohraničení, aby nedošlo k vynechání žádné části označovaného objektu, přičemž tyto boxy mohou přesahovat i přes jiné objekty, jestliže se překrývají.

2. Označení tříd

Po označení objektů zájmu je nezbytné přiřadit správnou třídu neboli label. Například při označování objektu zvířat musíme vhodně definovat jednotlivé třídy, jako jsou „pes“ a „kočka“. Správné označení je klíčovou částí pro přesnost detekčních modelů. [51]

Obrázek 21: anotace pro detekci (převzato z [51])

Segmentace

Tato technika rozděluje obrázky na segmenty patřící daným objektům či třídám. Existují dva hlavní typy segmentace:

1. Sémantická segmentace

Označí každý pixel v obrázku specifickou třídou objektu, do které spadá. Tímto způsobem se získá detailní mapa tříd na zkoumaném snímku. Například všechny pixely, které tvoří třídu pes, budou označeny jako „pes“. [52]

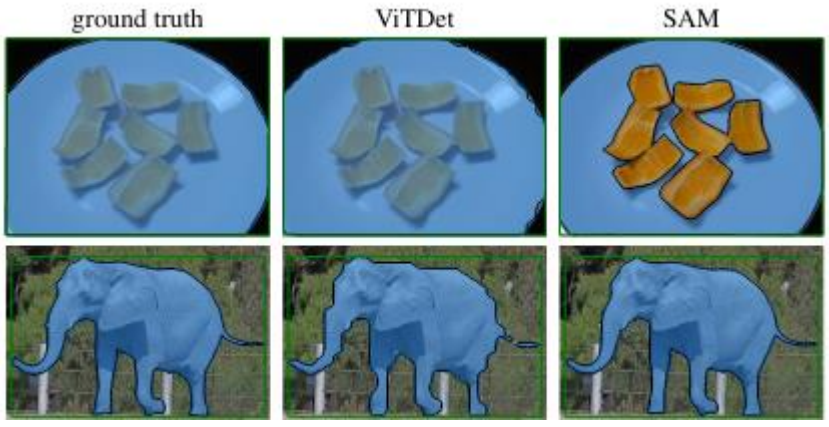
2. Instanční segmentace

Oproti sémantické segmentaci rozlišuje jednotlivé instance objektů. To znamená, že místo označení všech objektů jako „pes“ na daném snímku se jednotlivé instance označí jako „pes1“, „pes2“ atd. Tento přístup je mnohem složitější, ale poskytuje mnohem více informací a díky tomu můžeme rozlišovat mezi jednotlivými objekty stejné třídy. [52]

Obrázek 22: ukázka segmentace (převzato z [52])

SAM (Segment Anything Model)

Jedná se o moderní techniku pro asistovanou segmentaci. Princip této techniky je založen na interaktivní segmentaci, kde uživatel poskytuje návodné značky a model segmentuje objekty podle tohoto značení. Tento přístup umožňuje rychlejší a přesnější anotace pro instanční segmentaci. Model je schopný se adaptivně učit z interakcí s uživatelem, čímž výrazně zvyšuje efektivitu procesu anotace. [53]



Obrázek 23: Ukázka SAM oproti jiným technikám (převzato z [53])

4.4.3 Kvalitní anotace

Provedení kvalitní anotace je klíčovým faktorem pro úspěch modelu strojového učení zejména pak pro oblast počítačového vidění. Nesprávně označená data mohou vést ke špatným výsledkům modelu, což je hlavní důvod, úroč je nezbytné postupovat podle odsvědčených nebo doporučených postupů pro anotaci. Níže se tato kapitola zaměří na principy a metody pro zajištění kvalitní anotace. [54].

Přesnost a konzistence

Nesprávně označené nebo nekonzistentně označené objekty zájmu mohou způsobit, že model bude mít problémy s generalizací a správným rozpoznáním objektů v nových datech. Označení objektů proto musí být co nejpřesnější, tedy bounding boxy musí těsně obklopovat objekty zájmu a segmentační masky přesně odpovídat obrysům objektů. Přesnost je důležitá zejména u velmi detailních úkolů, jako je instance segmentace, kde každá chyba může výrazně ovlivnit výkon modelu.

Všichni, kteří provádí anotaci, by měli používat stejná pravidla či standardy pro označování. To znamená, že objekty by měly být označeny stejným způsobem v celém datasetu. Konzistence lze velmi dobře dosáhnout za pomoci školení pro anotátory a použitím podrobných pokynů pro anotaci. [54]

Kontrola kvality

Pravidelná kontrola kvality anotací je nezbytnou součástí pro zajištění, že data jsou správně označena. Existuje několik metod, jak tohoto cíle dosáhnout:

Prvotní anotace by měly být pravidelně revidovány a schvalovány zkušenými anotátory či supervizory, aby se zajistilo, že všechny anotace splňují požadované standardy kvality. Pro zvýšení spolehlivosti a eliminaci chyb mohou být stejné obrázky označovány více anotátory nezávisle na sobě. Následné porovnání a vyhodnocení může odhalit nesrovnalosti, které je třeba opravit.

Inter-coder agreement, neboli míra shody mezi anotátory, je klíčovou metodou pro hodnocení konzistence a kvality anotací v datasetu. Použitím této metody můžeme identifikovat a řešit problémy s nekonzistencí mezi anotátory, což je zásadní krok pro spolehlivost a validitu dat. [54, 55]

Vhodně definované třídy

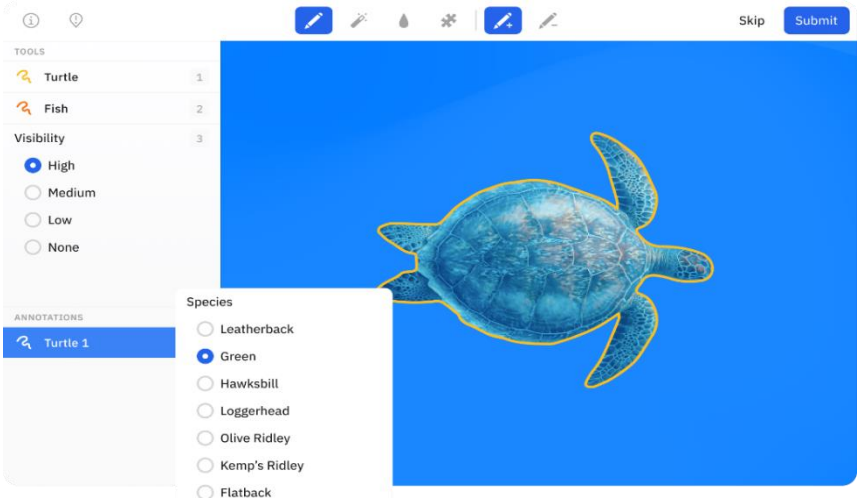
Správná definice tříd je klíčovým krokem v procesu anotací. Třídy musí být jasně definované a v zájmu zachování kvality anotací by se neměli překrývat. Každá třída musí mít jasnou definici, která vysvětluje, co do ní bude zahrnuto a co již ne. Tímto lze předcházet chybám během tvorby anotací. Cílem vhodně definovaných tříd je vytvoření jejich hierarchické struktury, například třídu „zvíře“ a její podtřídy „kočka“, „pes“ atd. Díky tomu se model učí rozpoznávat nejen obecné kategorie, ale i specifické podkategorie. [54]

4.4.4 Dostupný software

Existuje mnoho komerčních i nekomerčních programů pro anotace. Následující text představuje několik příkladů nekomerčních programů, které exportují data do formátů podporovaných frameworky jako PyTorch, TensorFlow a OpenCV.

LabelBox

Jedná se o komplexní platformu pro anotaci dat s vysokou přesností, vhodnou pro širokou škálu úloh strojového učení, včetně detekce objektů. Nutností je mít jeden z podporovaných webových prohlížečů, protože se jedná o webovou platformu, takže hardwarové požadavky nejsou náročné. [56, 57]



Obrázek 24: Ukázka LabelBox (převzato z [56])

Tento software umožňuje jednoduchý import našich dat a pro jejich zpracování je zde přívětivé uživatelské rozhraní. LabelBox poskytuje řadu pokročilých funkcí pro správu členů týmu, kteří mohou spolupracovat na práci s daty. Mezi funkce můžeme zařadit správu uživatelských rolí, práci s úkoly pro členy týmu a různé prostředky komunikace. Součástí jsou také funkce pro kontrolu kvality pro vysokou přesnost anotací.

Anotovaná data lze poté exportovat do formátu podle našich potřeb. Podporované formáty jsou například JSON, CSV ale také formáty pro strojové učení PyTorch či TensorFlow. [56]

LabelImg

Další zástupce softwaru pro anotaci je bezplatný program LabelImg s otevřeným kódem, který byl vydán v roce 2015. Jedná se o velmi jednoduchý a základní software vyvinutý v Pythonu zvládající datasety o několika stech obrázků. Výsledný dataset lze uložit v mnoha různých formátech, například XML pro formát PASCAL VOC, ale úspěšně pracuje i s formáty pro YOLO či CreateML. Tento program je spíše určen pro menší projekty. [58]

Obrázek 25: Ukázka LabelImg (převzato z [58])

Computer Vision Annotation Tool (CVAT)

CVAT je open-source program pro anotaci, který byl vyvinut speciálně pro úkoly v oblasti počítačového vidění společností Intel. Program umožňuje detekci objektů pomocí anotace bounding boxů, tedy ohraničení objektů rámečky. Podporuje ale také řadu dalších funkcí, jako je segmentace a klasifikace obrázků a anotace 3D dat.

Obrázek 26: Ukázka CVAT (převzato z [59])

Tento software je multiplatformní a uživatelsky přívětivý, tudíž je vhodný i pro začátečníky. Data dokáže exportovat do formátů JSON, COCO a PASCAL VOC, které jsou podporované běžně používanými frameworky. [59]

5 Zpracování obrazu (OCR)

Optické rozpoznávání znaků (OCR) je technologie umožňující detekovat text z tištěných či ručně psaných znaků v digitálních snímcích dokumentů, skenů, PDF dokumentech či obrázcích. Tento software může být dále použit v kombinaci s umělou inteligencí například k obnově historických nebo právních dokumentů s možností další editace. [60, 61]

5.1 Princip fungování OCR

Princip, jakým OCR funguje lze popsat v několika krocích. Prvním krokem je získání snímku pro detekci. Takovýto snímek může být fotografie či jakýkoliv dokument získaný například prostřednictvím skeneru. [60]

Předzpracování (Pre-processing)

Poté přichází zásadní krok, a to předzpracování, při kterém jde o úpravu dokumentu či fotografie pro vylepšení přesnosti detekce. Úpravy zahrnují eliminaci hluku (noise elimination) prostřednictvím různých filtrů či algoritmů, pomocí kterých zredukujeme šum ve snímku pro OCR. Šumem mohou být různé čáry či skvrny na snímku, které mohou zhoršit přesnost detekce. Tento krok výrazně zvýší přesnost při rozpoznávání znaků textu.

Další částí této úpravy jsou binarizace (binarization) a rozpoznávání/vylepšení zkreslení (Skew recognition/improvement). Binarizace je proces převodu snímku z barevného obrazu do černobílých barev. Dochází k rozpoznávání kontrastu mezi textem a pozadím, kdy text obvykle má černou barvu a pozadí bílou. Rozpoznávání/vylepšení zkreslení je proces, kdy pomocí algoritmů napravujeme geometrické deformace nebo různá zarovnání, ke kterým mohlo dojít v průběhu pořizování snímku. [62]

Segmentace a Rozpoznávání znaků (Character Segmentation and Recognition)

Třetím krokem je segmentace a následné rozpoznání znaků. Segmentace se provádí několika možnými metodami. První metodou je projekce znaků, která je však nejméně přesná. v principu se text projektuje na vertikální a horizontální osu. Nejprve se text promítne na horizontální osu a spočítá se počet pixelů v každém sloupci, což indikující základní umístění textu. Poté se text promítne na vertikální osu, rovnějž se spočítají pixely, čímž je daný text rozdělen na slova. Maximální hodnoty v řádku pak

odkazují na pozice znaků. Jedná se o rychlou metodu, avšak je náchylná na zhoršení kvality a možné šumy snímku.

Druhou metodou je dělení podle svislých a vodorovných čar, kde se text rozděluje na řádky a poté na samostatné znaky za pomocí vodorovných či svislých čar. Tato metoda je velmi efektivní u textů s čistým formátováním, avšak je náchylná na šum nebo nepravidelné rozložení textu.

Třetí a nejúčinnější metodou je segmentace znaků podle tvaru. Tato metoda je mnohem pokročilejší než předchozí dva zmíněné příklady. Podstatou je rozdělení textu na samostatné znaky a analyza jejich tvaru.

Tohoto rozdělení znaků lze dosáhnout různými metodami. Mezi ně lze zařadit porovnávání šablon, kde se porovnávají tvary, úsečky a klíčové rysy znaku se šablonou. Další metodou je rozdělení a sloučení, při které se text rozloží na menší segmenty a poté znovu skládá, což je výpočetně náročné. Běžnou metodou je naučení neuronové sítě pro rozpoznávání znaků, kde se síť učí rozpoznávat znaky pomocí příkladů. Metoda pomocí neuronové sítě je vhodná pro složité fonty či styly, dosahuje vysoké přesnosti, avšak vyžaduje velkou trénovací sadu. Další vhodnou metodou může být statistické rozpoznávání znaků, které jsou založeny na principu anylýzy vlastností zkoumaného znaku, jako je například textura. Tato metoda je odolnější vůči šumu, ale rovněž je výpočetně náročnější.

Důležité je poznamenat, že hlavní roli při detekci textu z obrázku či snímku kamery hraje hlavně kvalita zkoumaného snímku. Mezi faktory ovlivňující rozpoznávání je nejen výše zmíněná segmentace znaků, ale také normalizace znaků pro eliminaci variace tloušťky čar, velikosti či orientace prostřednictvím normalizačních metod, jako jsou normalizace výšky, šikmosti, tloušťky či středové linie znaků.

Následné zpracování (Post-processing)

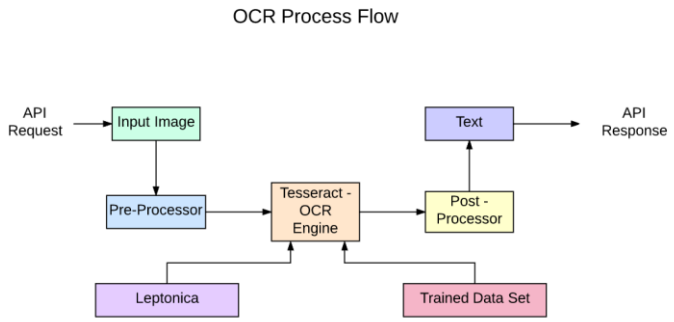
Hlavním účelem této závěrečné fáze je korekce chyb rozpoznaných znaků, které lze dosáhnou za pomoci jazykových modulů využívající gramatická pravidla pro korekci rozpoznaných slov. Je také možné využít slovníky obsahující seznam známých slov nebo algoritmy heuristiky pro korekci běžných chyb, jako je například záměna znaků („a“ a „o“ nebo „2“ a „S“). Úspešnost této fáze závisí na kvalitě a velikosti slovníku či složitosti detekovaného jazyka. [62]

5.2 OCR frameworky

OCR je poměrně podstatný komponent u práce jako je tato. v této části jsou představeni někteří zástupci volně dostupných OCR, které lze použít pro čtení textu registračních značek.

5.2.1 Tesseract

Framework Tesseract je open source nástroj pro optické rozpoznávání znaků (OCR) vyvíjený společností Google s licencí Apache 2.0. Tento framework podporuje více než 100 jazyků včetně češtiny. Podporuje také různé formáty vstupních obrázku, a to TIFF, JPEG, PNG, PDF a další. [63]



Obrázek 27: Tesseract process flow Api verze (převzato z [37])

Tesseract má více verzí. Ta nejnovější, Tesseract 4.0, má vylepšenou detekci a rozpoznávání vstupních dat prostřednictvím integrace AI LSTM[4] neuronové sítě. Tento framework podporuje několik programovacích jazyků a frameworků či Wrapper, například pro jazyk Python je dostupný Pytesseract. Oblast propojení Tesseract a Pythonu je veliká, například:

- PYOCR – Knihovna, která umožnuje použít více možností detekce u vět, číslic a slov,
- Textract – umožňuje extrakci dat z velkých PDF souborů,
- OpenCV – knihovna programovatelných funkcí pro analýzu v reálném čase,
- Leptonica – poskytuje funkce pro zpracování obrazu a prácis vlastní knihovnou zobrazování obrazu. [64]

[4] LSTM – Long Short-Term Memory

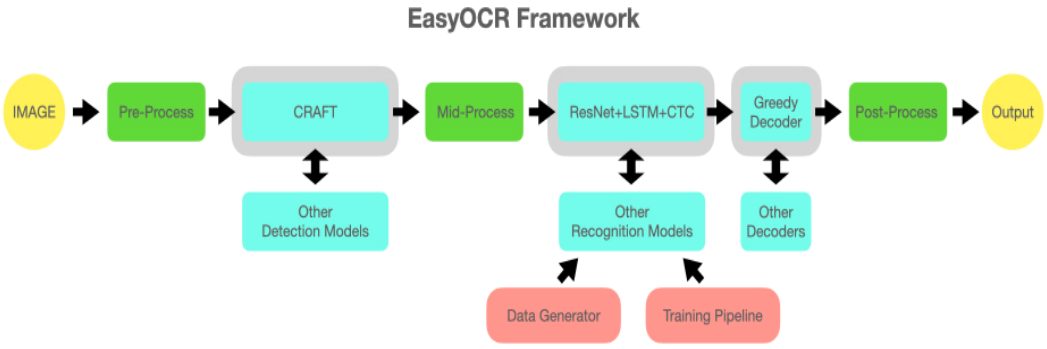
Velkou výhodou Tesseract frameworku je možnost přeučení na příklady, které tento framework nepokrývá. To, avšak vyžaduje velký dataset s tisíci příkladů obrázků nebo dokumentů. Tento proces muže být velmi komplikovaný a časově náročný. [64, 65]

Hlavním omezením tohoto OCR je kvalita obrazu, se kterou má během detekce textu problémy, pokud snímek obsahuje šum, zkreslení či nízký kontrast. v detekci textu z dokumentů může mít problémy, když jsou použity různé layouty jako formuláře či tabulky. [65]

Princip fungování Tesseract frameworku je téměř totožný s obecným principem. Ten je popsán výše v podkapitole 5.1 s názvem Princip fungování OCR.

5.2.2 EasyOCR

Další vhodnou volbou pro optické rozpoznávání znaků je volně dostupný modul EasyOCR, který je jednoduše a flexibilně použitelný pomocí jazyka Python. Tento optický rozpoznávač znaků je navržen tak, aby byl velmi snadno použitelný, a především uživatelsky přívětiví, a poskytuje širokou škálu již předučených modelů detekce a rozpoznávání textu. Ačkoli se jedná o poměrně nový framework rychle získává popularitu zejména mezi vývojáři Pythonu díky svým možnostem práce s textem a rychlosti. Velkou výhodou je podpora více než 80 jazyků včetně češtiny a vývojáři stále přidávají podporu pro další jazyky. [66]



Obrázek 28: EasyOCR framework (převzato z [66])

Princip EasyOCR lze rozdělit do tří hlavních částí, na kterých stojí kompletní funkcionalita tohoto frameworku.

Extrakce funkcí (Feature Extraction)

Jedná se o první část procesu EasyOCR. Zde se pro extrakci textu ze zkoumaného obrazu používají modely hlubokého učení, a to ResNet a VGG, které jsou potřebné pro rozpoznání textu. [66]

ResNet plným názvem Residual Networks je druh konvoluční neuronové sítě (CNN) známý svou vlastností tvorby zkratek (skip connections) či přeskakováním. Těmito zkratkami se propojují vstupní a výstupní vrstvy bloků sítě, což umožňuje efektivnější šíření a snižuje problém mizení gradientů[5]. Zkratky umožňují trénování hlubších sítí až do 1000 vrstev bez ztráty přesnosti. Tím se výrazně zlepšuje optimalizace a učení sítě. Pro EasyOCR je to velkou výhodou v prvotních fázích, tedy pro získání relevantních informací ze snímaného snímku pro další zpracování. [66–68]

Visual Geometry Group (VGG) je architektura hluboké konvoluční neuronové sítě (CNN) s 16 vrstvami, která je jednoduchou a homogenní architekturou, což umožnuje její jednoduchou implementaci. Jedná se o architekturu tvořenou z velmi malých konvolučních filtrů skládaných do sebe. To znamená, že se skládá z bloků, které se opakují a obsahují konvoluci (aplikaci filtrů na vstupní data) a max pooling (vrstvu snižující rozlišení obrázku). v principu obrázek jako vstupní data prochází skrz síť vrstvu po vrstvě. Na začátku se extrahují okraje a textury a v pozdějších vrstvách se provádí kombinace těchto vlastností a vytváří se komplexní reprezentace daného obrázku. [68]

Sekvenční značení (Sequence Labeling)

Zde je využita LSTM neuronová síť speciálně navržená pro práci se sekvenčními daty. Oproti klasickým neuronovým sítím si LSTM pamatuje informace z minulých kroků sekvence v buňkách a používá je k predikci budoucích kroků. To umožňuje řešit úkoly, jako je strojový překlad, rozpoznávání řeči a analýza časových řad. LSTM je velmi vhodná pro práci s data, která se mění v čase, jako je text, video či zvuk.

[5] Gradient – míra změny funkce v daném bodě

Základními prvky LSTM jsou paměťové buňky, které uchovávají informace z předchozích kroků sekvence a tři různé druhy brán: vstupní, výstupní a zapomenutí. Vstupní brána řídí tok informací, které se zapisují do buňky, výstupní brána řídí tok informací pro zobrazení výstupu a brána zapomenutí řídí, kolik informací se má zapomenout z předchozího kroku. [66, 69]

Dekódování (CTC)

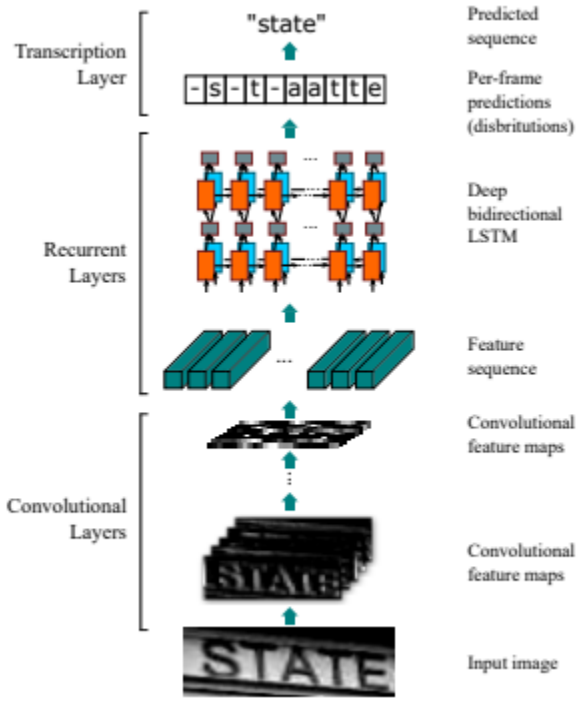
Posledním krokem pro rozpoznávání se využívá Connectionist Temporal Classification (CTC). Jedná se o ztrátovou funkci používanou v neuronových sítích pro úkony sekvenčního značení. v EasyOCR hraje roli v oblasti dekódování sekvence pravděpodobností generované LSTM sítí do sekvence znaků. [66]

Vstupem do této funkce je sekvence pravděpodobností, kde každý krok reprezentuje pravděpodobnost výskytu daného znaku. Poté dekodér prochází sekvenci pravděpodobností krok za krokem a vybírá label s největší pravděpodobností. Dekodér může vkládat i prázdný label pro zohlednění, že ne každý krok v sekvenci pravděpodobností odpovídají některému znaku. Výstupem je zde sekvence znaků reprezentující rozpoznaný text. [70]

5.2.3 PaddleOCR

Vhodným open-source frameworkem pro optické rozpoznávání je také PaddleOCR, někdy taky označován jako PP-OCR, vyvíjený společností Baidu AI. Tento OCR je specificky navržen pro rozpoznávání textu a jeho následné extrahování zaměřené na obrázky a dokumenty.

Velkým rozdílem oproti ostatním OCR, které jsou v práci analyzovány, je použití modelů hlubokého učení pro všechny činnosti, které OCR provádí, čímž je dosaženo zefektivnění rychlosti a především přesnosti. Mezi podporované formáty vstupních data patří JPEG, PNG, PDF a BMP. Součástí frameworku je řada před trénovaných modelů hlubokého učení, které byly trénovány na různých datech, tudíž zvládá detekovat text ve více než 70 jazycích. Seznam podporovaných jazyků obsahuje nejen angličtinu, ale také specifické asijské země. Pyšní se například podporou zjednodušených znaků čínského znakového písma, což činí PP-OCR vhodným pro podniky v Číně. Konkurence podporuje většinou více jazyků což by mohlo být jedno z hlavních omezení pro použití tohoto OCR. [71]



Obrázek 29: PaddleOCR vrstvy architektury (převzato z [55])

Architektura PaddleOCR je založena na konvoluční rekurentní neuronové síti (CRNN), která se skládá ze tří vrstev. První vrstva je tvořena konvoluční neuronovou sítí (CNN), která je zodpovědná za extrakci prvků z obrázku na vstupu a tvorbu map těchto prvků jako výstupu. Druhá vrstva je tvořena rekurentní neuronovou sítí (RNN), postavené na konvoluční neuronové síti. Zde jsou aplikovány dva obousměrné LSTM pro řešení problematiky mizejícího gradientu a tím se dokáže vytvořit hlubší síť. Třetí a poslední vrstvou je transkripce, která je zodpovědná za převod předpovědí podle pravděpodobností na výsledný snímek. [72, 73]

6 Technické zázemí a proces implementace

Pro implementaci bylo po dohodě s technikem, který se stará o kamerový systém a brány parkovišť, vybráno parkoviště u budovy C Přírodovědecké fakulty JU. Brána na tomto parkovišti je pouze několik let stará, a proto jsme se poměrně dlouhou dobu tvorby této práce domnívali, že by mohlo manipulací s bránou dojít k porušení záruční lhůty. Tuto domněnku technik z přírodovědecké fakulty byl schopen vyvrátit až tři týdny před odevzdáním této práce.

Obrázek 30: Brána parkoviště u budovy C přírodovědecké fakulty JU

6.1 Stav současného řešení

6.1.1 Řešení vjezdu a výjezdu parkoviště

Současné řešení spočívá v načtení univerzitní karty přes systém JIS a pokud má uživatel povolený vstup, závora se otevře. Další implementované řešení, které je zde implementováno je zvonek na vrátnici. Je však problematické, jelikož pracovníci častočasto při zazvonění otevírají bránu všem nehledě na to, kdo se pokouší o vjezd.

Výjezd z tohoto parkoviště je mnohem zajímavější. v silnici je zabudován tlakový senzor, a tak je závora otevřená každému, kdo vyjíždí a přijede blíže k bráně. Kvůli tomuto senzoru by nemělo smysl aplikovat vstupní i výstupní kontrolu například čtením registračních značek, ale lze se zaměřit pouze na rozšíření možností pro vjezd parkoviště.

6.1.2 Napájení a síťová infrastruktura

U závory je umístěna skříň s jističi pro elektroinstalaci a MikroTik switchem napojeným na optický kabel. v tomto switchi je jeden volný port, což by pro budoucí propojení Raspberry Pi a kamery nestačilo. Bude tedy zapotřebí použít další síťový prvek pro kompletní propojení.

Obrázek 31: Ukázka skříně elektroinstalace při testovacím zapojení

Dalším problémem je nedostatek místa v této skříni, což lze řešit přidáním další skříně pod stávající a protažením potřebných kabelů do nové skříně, například pro připojení ke switchi a zajištění potřebného napájení.

Ve zmíněné skříni závory je přivedeno 230V napájení, které bude třeba transformovat na 12V/1A pro napájení kamery. Pro zvolené Raspberry Pi je zapotřebí zabezpečit doporučené napájení přes 5V/3A DC adaptér pro napájení a stabilní přístup k internetu. Další natažení kabelu by však zahrnovalo výkopové práce, protože stávající řešení nepočítalo s rozšiřováním či potřebou dalšího natažení kabelů. Ideálně bude Raspberry Pi připojeno k internetu prostřednictvím ethernetu ze switche, což zajistí zabezpečí stabilní a rychlé připojení pro získávání dat od druhé části tohoto projektu zařizovaným studenty z předmětu IS projekt. Tato část projektu bude popsána v kapitole 10.

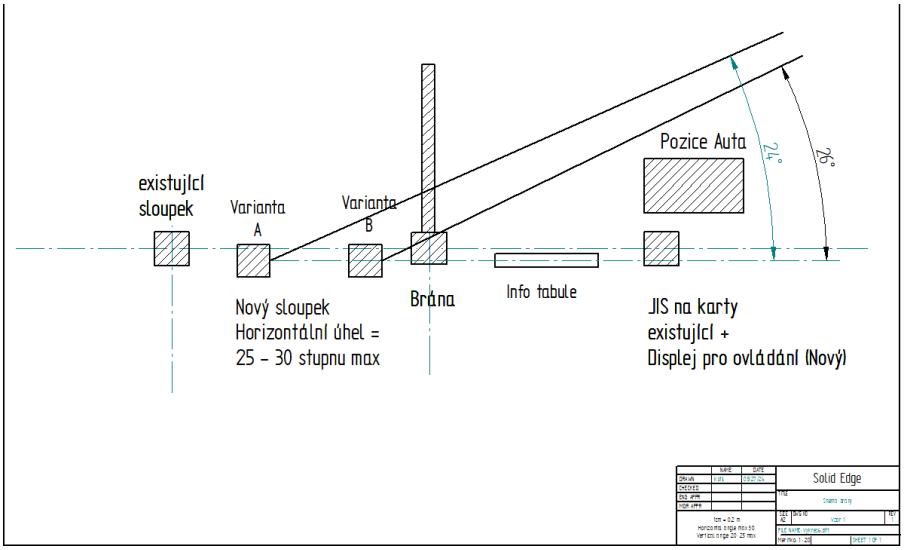
6.2 Návrhy nasazení

Instalace kamery k závoře parkoviště má několik omezení, které je třeba zohlednit před fyzickým nasazením.

6.2.1 Využití sloupku

Původní návrh počítal s využitím již existujícího sloupku, na kterém se již nachází bezpečnostní kamera a který se nachází v optimální pozici vůči závoře. Tento návrh byl po několika prvních konzultacích zamítnut. Hlavním důvodem byl požadavek, abych nezasahoval do existujícího kamerového řešení, což vedlo k nutnosti zvážení instalace nového sloupku. Hlavním omezením byl tedy zákaz zásahu do existujícího řešení.

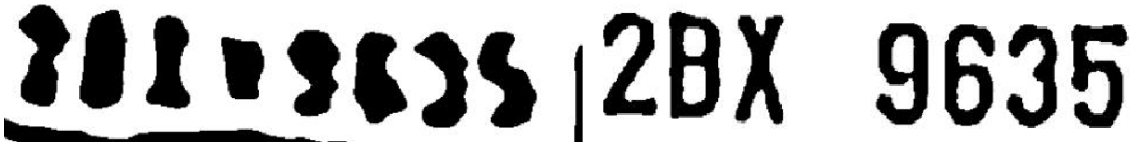
Alternativní řešení je použití **nového sloupku** pro kameru, který musí být umístěn tak, aby vertikální úhel kamery byl v rozmezí 15–30 stupňů a neztratil čistý záběr na snímaný automobil. To představuje značnou výzvu vzhledem k tomu, že v ose, na kterou lze sloupek umístit, se nachází informativní tabule Přírodovědecké fakulty, která by mohla clonit výhledu kamery. Naopak by také mohla tvořit stín a tím snížit pravděpodobnost chyby při čtení registračních značek způsobené možným odleskem.



Obrázek 32: Návrh umístění sloupku (obsahující varianty a a B)

Ideální umístění sloupku je již zabrané existujícím sloupkem, a proto by se nový musel umístit na osu či lehce za osu informační tabule, a to převážně kvůli vedení elektroinstalace k existujícímu řešení. Současné řešení je implementováno tak, že mezi sloupkem a bránou je vzdálenost 4 metry, a to samé platí pro vzdálenost brány a čtečky karet JIS. Pro zaručení přesnosti systému vyvíjeného v této práci je zapotřebí dodržet

přibližnou vzdálenost 6 metrů mezi kamerou a snímaným automobilem. Při překročení určité vzdálenosti se text upravovaného snímku registrační značky může zdeformovat kvůli aplikovaným úpravám pro zvýšení čitelnosti, které jsou popsané v kapitole 9.2.



Obrázek 33: Porovnání výsledků detekce v závislosti na vzdálenosti

Zmíněné deformace jsou reprezentovány na obrázku 33.

Do nového sloupku je nutné zavést síťový kabel k propojení kamery s Raspberry Pi a dále zajistit napájecí zdroj pro kameru. v současném řešení univerzity je používáno napájení 12V/1A dle informací poskytnutých techniky, které bude zachováno i v novém řešení. Raspberry Pi bude muset mít přístup k internetu pro aktualizaci a získávání dat z endpointu pro vyhodnocení vstupu na parkoviště.

6.2.2 Využití skříně brány

Jedním z hlavních problémů je zajištění dostatečné přesnosti čtení registračních značek (SPZ) z kamery instalované v určité vzdálenosti. Bylo zjištěno, že přesnost čtení značně klesá při vzdálenosti větší než 6 metrů. z tohoto důvodu je nezbytné minimalizovat vzdálenost mezi kamerou a SPZ.

Pro vyřešení tohoto problému bylo třeba:

1. **Optimalizovat umístění nového sloupku**: v zájmu minimalizace vzdálenosti od detekované SPZ.
2. **Zvolit vhodnou kameru**: s vysokou citlivostí a rozlišením, která umožní pořízení kvalitního snímku i na větší vzdálenost, a to vše v zájmu přesnosti čtení SPZ.

V souvislosti s těmito problémy vyvstala otázka opětovného využití nově vystaveného sloupku v případě, že se univerzita rozhodne vyvíjený systém pro otvírání závory nepoužívat. Při osobních konzultacích s techniky univerzity u závory jsem navrhl alternativní řešení, které spočívá v umístění kamery na skříň závory. Tímto způsobem by se vzdálenost mezi kamerou a SPZ snížila na přibližně 3 metry. Možnou nevýhodou tohoto řešení je, že kamera by byla pouze ve výšce jednoho metru, a tudíž by mohla být snadno odcizena, i když je v zorném úhlu bezpečnostní kamery.

Obrázek 34: Pozice pro umístnění kamery skříň závory

Navrhované řešení však nebylo přijato techniky, kteří vyjádřili obavy ohledně možného zatékání vody do skříně závory v důsledku nutnosti přivrtání kamery.

Bez ohledu na zvolené řešení pro fyzickou instalaci je nezbytné provést podkopání základní desky, na které je umístěna skříň brány. Tento krok je potřebný nejen pro uložení kabelů nezbytných k instalaci kamery, ale také pro natažení kabelů od QR čtečky nebo displeje a zajištění potřebného napájení. Tento proces by byl velmi zdlouhavý a vyžadoval by rozsáhlou organizační přípravu. Technici vyjádřili obavy z možného vážného poškození brány během instalace a vykopávkových prací. Dále odhadli, že pouhá organizační příprava by zabrala minimálně tři měsíce.

6.3 Požadavky pro navrhované řešení

Během konzultací a spolupráce s týmem z IS projektu vzešlo několik požadavků:

6.3.1 Offline režim

Jeden z požadavků na systém je schopnost pracovat offline a uchovávat základní data pro kontrolu vstupu, která se budou aktualizovat každých 15 minut. Tato data obsahují skupiny uživatelů, jednotlivé uživatele, QR kódy, povolené a blokované registrační značky a parametry pro nastavení brány.

Tento požadavek vzešel od týmu z předmětu IS projekt a přinesl zajímavé řešení. Při každé žádosti o data se posílají kompletní data z endpointu, čímž se vytváří nová databáze a stará databáze se stává zálohou. Tento přístup zajišťuje kontinuitu a bezpečnost dat i v případě dočasného výpadku internetového připojení. Bylo však nezbytné zohlednit strukturu přijímaných dat z endpointu.

6.3.2 Další možnosti vstupu na parkoviště

Dalšími dvěma požadavky bylo umožnit vstup nejenom pro detekované registrační značky, ale také zajištění vjezdu prostřednictvím naskenovaného QR kódu nebo otevření pomocí stisku tlačítka na displeji pro zásilkové a rozvozové služby.

Nicméně se zde objevuje otázka, zda by nedošlo k zneužití jednorázového vjezdu před displej pomocí stisknutí tlačítek, které jsou určené pro doručovací služby, například, například ze strany studentů, kteří by neměli oprávnění k vjezdu, jako jsou ti, kteří nemají zaregistrovanou registrační značku, nebo ti, kteří jsou na černé listině.

6.4 Volba hardwarových prostředků

Správná volba hardwaru je klíčovou částí této diplomové práce. Vybraný hardware musí být schopný poskytnout kvalitní snímek, který bude zachycen kamerou, a poté musí disponovat dostatečným výpočetním výkonem pro zpracování detekce automobilu a následného rozpoznání textu registračních značek.

6.4.1 Kamera

Prvním klíčovým prvkem hardwaru, který jsem vybíral pro tuto práci, je kamera určená pro zachycení kvalitního snímku potřebného pro následnou detekci. Výběr možných kandidátů byl výrazně omezen klíčovým požadavkem technického oddělení univerzity, aby kamera byla vyrobena společností Hikvision, a to proto, aby zařízení zapadalo do stávajícího kamerového řešení/zázemí univerzity, navzdory tomu, vybraná kamera nebude zapojena do stávajícího bezpečnostního systému.

Vzhledem k tomu, že kamera bude součástí parkovacího systému, bylo zjištěno, že musí být schopna operovat za dne i v noci. Proto je infračervený přísvit nezbytnou funkcí. Dalším specifickým parametrem je ohnisková vzdálenost kamery, která umožňuje flexibilitu záběru úseku u vjezdu na parkoviště. Pro snímání registračních značek automobilů měl být objektiv o ohniskové vzdálenosti do 12 mm dostačující, vzhledem k malé vzdálenosti od závory k parkovišti.

S ohledem na tyto požadavky jsem vybral kameru Hikvision iDS-2CD7A46G0/P-IZHS s objektivem 2,8-12 mm, která je specifikována výše v podkapitole 3.3. Značnou výzvou bude propojení kamery s jednodeskovým počítačem, který popíši v následující podkapitole.

6.4.2 Výpočetní HW

Druhým klíčovým prvkem hardwaru je výpočetní jednotka. Výběr byl obtížný s ohledem na poměr cena/výkon. Zvažoval jsem například Tinker Board, Raspberry Pi a NVIDIA Jetson, která byla vyvinuta pro práci s neuronovými sítěmi. Prvním faktorem je cena, která nevyhovovala požadovanému low-cost řešení, protože i nejlevnější model Nvidia Jetson Nano přesahoval cenovou hranici 7 500 Kč, což je téměř čtyřnásobek ceny Raspberry Pi 4 s 8 GB RAM.

Dalším důležitým faktorem je specifické napájení (jak bylo zmíněno v kapitole 2.1.1). Je pravděpodobné, že by zařízení nemělo dostatek napájení pro další připojené periferie, což by vyžadovalo použití specifického zdroje napájení. To by mohlo představovat problém z hlediska velikosti zařízení potřebného pro umístění výpočetní jednotky do skříně závory parkoviště.

Důležitou součástí je také dostatečná uživatelská základna pro řešení případných problémů a volba operačního systému. Nvidia Jetson používá softwarový balíček jako vlastní operační systém, zatímco Raspberry Pi a Tinker Board používají systémy založené na Linuxu. z těchto důvodů jsem se rozhodl pro zařízení od Raspberry Pi, které je poměrně malé, má dobrý výkon a širokou podporu. Nejnovější verze byla sice uvedena na trh začátkem roku 2024, ale vzhledem k dostupnosti jsem zvolil starší verzi Raspberry Pi 4, předpokládám, že je projekt schopen běžet i na jeho výkonnějším nástupci, který poskytuje čtyřnásobek výkonu oproti předchozí verzi.

6.4.3 Dodatečný HW

Jak již bylo uvedeno výše, byl vznesen požadavek na možnost otevírání brány prostřednictvím QR kódů a tlačítek na dotykovém displeji. Pro tento účel byl pořízen modul čtečky čárových kódů (Barcode Scanner Module B), u kterého jsem naskenováním nastavovacích QR kódů z přiloženého manuálu provedl prvotní nastavení čtečky. Sedmipalcový dotykový displej od značky Waveshare, zapůjčený univerzitou, není vhodný pro permanentní nasazení, avšak pro testovací účely je plně postačující.

Pro finální fyzické nasazení však bude nutné zajistit dodatečný hardware v podobě úchytů pro displej a QR čtečku. Kromě toho bude vyžadováno vyrobit speciální pouzdro pro QR čtečku, které zajistí její ochranu před vnějšími vlivy. Zároveň je nezbytné zabezpečit, aby jak displej, tak QR skener byly voděodolné, což je klíčové pro jejich spolehlivé fungování v exteriérovém prostředí. Tyto kroky jsou nezbytné k zajištění dlouhodobé a bezproblémové operace zařízení v rámci dané aplikace.

6.5 Postup testovacího nasazení

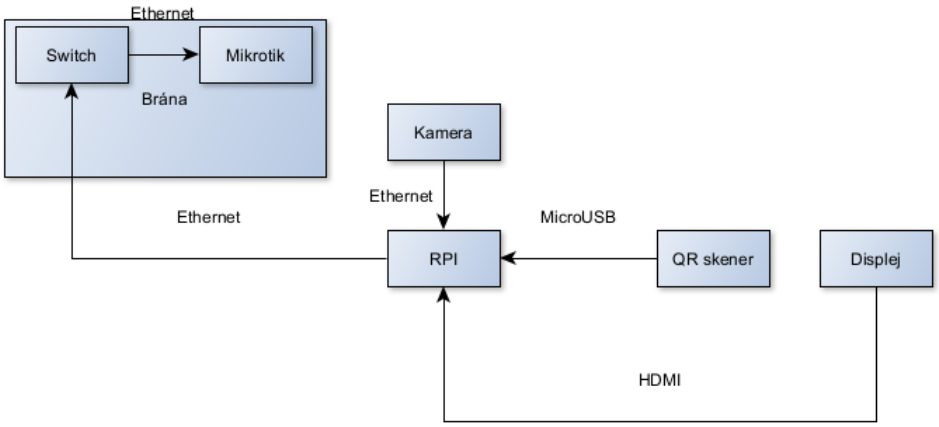
Během vývoje nastalo několik problémů, kvůli kterým nebylo možné systém implementovat takovým způsobem, aby představoval permanentní řešení. Prvním z důvodů byly časté organizační změny ze strany techniků či dlouhé čekání na jejich vyjádření k mnou navrhovaným řešením nebo jsem byl několikrát odkázán na jiného zaměstnance univerzity.

Druhým z důvodů byly nedostatky části systému, který vyvíjel tým z předmětu IS projekt. v prvé řadě je problém s tím, že v současné době není jejich část systému kompletně dokončena. Hlavním problémem je, že v systému nebyli zavedeni žádní uživatelé kromě vývojářů, což znamená, že by v případě uvedení mého systému do provozu nemohl být používán nikým jiným. Navíc by bylo nezbytné uživatele parkoviště o změnách a zavedení nového systému s dostatečným předstihem informovat.

Vzhledem k uvedeným nedostatkům jsem se rozhodl systém zapojit pouze prostřednictvím testovacího řešení. Toto řešení mi pomohl realizovat pan Petr Zich z technického úseku Přírodovědecké fakulty. Testovací scénář zahrnoval instalaci kamery na dřevěný sloupek umístěný u krajnice silnice před závorou. Cílem bylo, aby kamera zaznamenávala automobily přijíždějící k bráně a k existující čtečce karet JIS.

Obrázek 35: Testovací umístění kamery

Vzhledem k omezené dostupnosti pouze jednoho LAN portu na switchi značky MikroTik se ukázalo jako nezbytné využít další switch pro účely kompletního propojení všech komponent systému. Tento dodatečný switch umožňuje propojení Raspberry Pi s kamerou a současně zajišťuje připojení Raspberry Pi k internetu prostřednictvím univerzitní sítě. Před zahájením testovacího procesu byl zmíněný port na switchi MikroTik odpojen, což vyžadovalo kontaktování správce za účelem jeho reaktivace. Po úspěšném připojení k univerzitní síti bylo nutné upravit síťové parametry aplikace a následně provést spuštění Docker kontejneru s aplikací. Detailnější rozbor této problematiky je uveden v kapitole 11. Kompletní zapojení znázorňuje diagram níže.



Obrázek 36: Ukázka testovacího zapojení

Na konci testovací fáze jsem provedl několik klíčových testů, které měly ověřit funkčnost celého systému. Každý typ vjezdu byl podroben scénářům simulujícím různé situace příjezdu vozidel na parkoviště. Všechny testy byly úspěšné a vedly k otevření brány, i přesto, že čtečka QR kódu byla narušena nadměrným slunečním svitem

z důvodu absence krytu pro čtečku. Nicméně, testy také poukázaly na prodloužení procesu vyhodnocení vjezdu po finální implementaci systému na Raspberry Pi. Tento problém by bylo možné vykompenzovat použitím nejnovější verze Raspberry Pi pravděpodobně z nedostatku výpočetního výkonu procesoru. Videozáznamy těchto testů budou k dispozici na DVD disku v příloze A, což poskytuje jasný přehled o funkčnosti a výkonnosti systému v reálném provozu.

Během testovacího nasazení jsem si všiml, že řada uživatelů parkoviště při příjezdu k bráně autem najíždí z nesprávného úhlu, a v důsledku toho musí s autem najet znovu v jiném úhlu nebo vystoupit, aby si mohli bránu otevřít pomocí karty. Přestože se při testovacím nasazení obdobný problém mého systému neprojevil, mohl by potenciálně v budoucnu nastat. z tohoto důvodu je třeba nakreslit obdélník vedle současné čtečky karet JIS, který pomůže řidičům najet k bráně ve správném úhlu.

7 Akvizice dat

Pro tuto práci je nezbytné mít k dispozici kvalitní a rozsáhlý dataset pro trénink neuronové sítě. Vytvoření takového datasetu by však přesahovalo rámec této práce a sběr dat by mohl být komplikován Obecným nařízením o ochraně osobních údajů (GDPR). Proto jsem se rozhodl použít volně dostupné datasety sestavené jinými autory.

Prvním problémem těchto datasetů je jejich velikost, která často nepřesahuje 250 obrázků. Existují sice i rozsáhlejší datasety s přibližně 1500 obrázky, ale jejich kvalita bývá velmi nízká a jsou vzácné. z tohoto důvodu jsem se rozhodl použít více menších datasetů a následně je spojit do jednoho většího, který jsem pak rozšířil pomocí vybraných augmentačních metod na části složeného datasetu. Tento přístup mi umožnil získat kvalitnější data z veřejných zdrojů v dostatečném množství.

První dataset[6], který jsem použít, obsahuje 433 obrázků automobilů s jejich registračními značkami ve formátu PNG. Obrázky jsou anotovány ve formátu Pascal VOC, což znamená, že je nutné následně zpracovat XML soubory obsahující pozice (rámeček) hledaného objektu. Tyto anotace je potřeba převést do formátu YOLO, který jsem zvolil pro framework učení neuronové sítě.

Obrázek 37: Ukázka obrázku z datasetů (převzato ze zdroje [74])

Druhý zvolený dataset[7] má 225 obrázků ve formátu JPEG a jejich XML soubory ve formátu Pascal VOC, tudíž bude zapotřebí stejného zpracování jako u předchozího datasetu.

[6] první dataset převzatý ze zdroje [74]
[7] druhý dataset převzatý ze zdroje [75]

Vzhledem k malému počtu snímků pro učení jsem se rozhodl použít další dva datasety ze zdroje roboflow.com, který nabízí zdarma datasety od různých autorů. Třetí dataset[8] v pořadí obsahuje celkem 395 obrázků ve formátu JPG a umožňuje stáhnout dataset v různých formátech, například YOLO, JSON či Pascal VOC. Zvolil jsem formát Pascal VOC kvůli již napsanému skriptu pro zpracování předchozích dvou datasetů v tomto formátu. Čtvrtý dataset[9] obsahuje 1084 obrázků ve formátu JPG a také umožňuje volbu formátu datasetu. Zajímavostí u těchto snímků je, že autoři již provedli augmentaci expozice (25 %) a rozostření (až 2 px).

Obrázek 38: Ukázka snímku z roboflow datasetu (převzato ze zdroje [76])

Tyto obrázky se liší od obrázků snímaných kamerou například úhlem záběru a různými úrovněmi přiblížení. Některé obrázky zobrazují přímý detail automobilu s poznávací značkou, zatímco jiné zachycují automobil zezadu, což by mohlo být vhodné pro použití modelu při snímání automobilů při výjezdu z parkoviště. Ojediněle se zde nachází přímý detail na poznávací značku. Některé snímky jsou zachyceny v menších rotacích, což může zvýšit robustnost naučeného modelu.

7.1 Augmentace dat

Vzhledem k celkovému počtu 1837 obrázků ve výsledném datasetu jsem se rozhodl tento dataset rozšířit prostřednictvím augmentace. Některá data již obsahovala augmentaci, a proto jsem se rozhodl aplikovat ji pouze na část dat, konkrétně na první dva datasety. Zvážil jsem různé typy augmentací, včetně šumu a geometrických úprav. Konkrétně jsem zvažoval aplikaci šumu typu HSV a Salt-and-Pepper a dále geometrické augmentace, jako jsou rotace a převrácení (flip).

[8] třetí dataset převzatý ze zdroje [76]
[9] čtvrtý dataset převzatý ze zdroje [77]

Rozhodl jsem se nepoužívat geometrické augmentace přímo na datech z několika důvodů. Použití flip augmentace by bylo nevhodné, protože výsledný model bude použit u vjezdu na parkoviště, kde by otočení obrazu mohlo způsobit chybné detekce, například by sloupek mohl být detekován jako poznávací značka. Podobně tomu je i u rotace – velké úhly rotace by zvyšovaly chybovost detekcí. Pro model, který bude použit u vjezdu na parkoviště, jsem zvažoval maximálně patnáctistupňovou rotaci. Tuto rotaci budu aplikovat náhodně během učení pomocí frameworku YOLO, společně s náhodnými úpravami jasu a odstínu. Tento přístup jsem vyhodnotil jako vhodnější alternativu.

Druhou kategorií augmentace, pro kterou jsem se rozhodl, byla aplikace šumu na snímky za účelem zvýšení přesnosti detekce v případech, kdy by se na zachyceném snímku kamery nacházelo rušení. Aplikoval jsem augmentaci typu HSV, která mění odstín, sytost a jas, aby simulovala různé světelné podmínky. Druhou použitou technikou byl šum typu Salt-and-Pepper, který náhodně mění hodnoty pixelů a simuluje zrnitý šum v obraze.

Obrázek 39: Ukázka Sand and Pepper augmentace (převzato z [76])

Těmito augmentačními metodami jsem rozšířil dataset o dalších 1316 snímků, což navýšilo robustnost modelu při detekci v případě přítomnosti rušení.

7.2 Zpracování dat

Tato kapitola popisuje proces zpracování datasetů, který byl proveden před jejich použitím k trénování YOLO sítě. Vzhledem k různým formátům a strukturám zdrojových datasetů bylo nutné sjednotit data do konzistentního formátu vhodného pro následné učení neuronové sítě.

7.2.1 Struktura a příprava vstupních dat

V úvodu kapitoly 7 byly představeny čtyři datové sady, které budou následně zpracovány. Tyto datové sady byly umístěny v různých adresářích a bylo nutné je sjednotit do jednotného výstupního formátu. Výstupní adresářová struktura byla navržena tak, aby oddělovala trénovací a validační data.

7.2.2 Konverze anotací do formátu YOLO

Protože anotace ve všech použitých datasetech byly ve formátu VOC (XML), bylo nutné tyto anotace převést do formátu YOLO. Formát Pascal VOC popisuje objekty v obraze pomocí souřadnic rohových bodů obdélníkových bounding boxů, zatímco formát YOLO používá relativní souřadnice středu objektu a jeho šířku a výšku vzhledem k rozměru obrázku. Níže uvedený skript načte XML soubor, extrahuje rozměry obrázku a souřadnice objektů a převede souřadnice do formátu YOLO.

```
def convert_pascal_voc_into_yolo_annotation(Path: str):
    tree = xet.parse(Path)
    rootElem = tree.getroot()
    size_elem = rootElem.find("size")
    imageWidth = int(size_elem.find("width").text)
    imageHeight = int(size_elem.find("height").text)
    yolo_annotation = ''
    objects = rootElem.findall("object")
    for obj in objects:
        name = str(obj.find("name").text)
        if name not in ["license-plate", "license", "licence",
                        "num_plate", "number_plate",
                        "License_Plate", "LicensePlate"]:
            continue
        boundingBoxElem = obj.find("bndbox")
        x_min = int(boundingBoxElem.find("xmin").text)
        x_max = int(boundingBoxElem.find("xmax").text)
        y_min = int(boundingBoxElem.find("ymin").text)
        y_max = int(boundingBoxElem.find("ymax").text)
        xCenter = (x_min + x_max) / 2 / imageWidth
        yCenter = (y_min + y_max) / 2 / imageHeight
        yoloWidth = (x_max - x_min) / imageWidth
        yoloHeight = (y_max - y_min) / imageHeight
        yolo_annotation += f'0 {xCenter} {yCenter} {yoloWidth} {yoloHeight}\n'
    return yolo_annotation.rstrip()
```

7.2.3 Rozdělení dat na trénovací a validační sady

Dalším krokem bylo rozdělení datasetů na trénovací a validační sady. Poměr rozdělení byl zvolen na 80 % pro trénovací data a 20 % pro validační data. Rozdělení bylo provedeno náhodným výběrem a bylo zajištěno, že data jsou dobře promíchaná pro každou sadu.

```
# Split data to 0.8 : 0.2 train / validate
def train_validation_split(inputData: list[any]):
    totalSize = len(inputData)
    trainSize = int(0.8 * totalSize)

    np.random.shuffle(inputData)

    trainData = inputData[:trainSize]
    validationData = inputData[trainSize:]
    return trainData, validationData
```

7.2.4 Sloučení a uložení finálního datasetu

Poslední část procesu zpracování dat zahrnuje sloučení, uložení a finální zpracování datových položek. Tato fáze zahrnovala konverzi obrázků do formátu JPG, uložení YOLO anotací a správné uspořádání dat do trénovacích a validačních adresářů.

```
def save_item(img_path: str, yolo_annot: str, img_dest: str, lbl_dest: str,
              item_name: str):
    new_img_path = os.path.join(img_dest, f'{item_name}.jpg')

    if img_path.endswith(".png"):
        img = cv2.imread(img_path)
        cv2.imwrite(new_img_path, img, [int(cv2.IMWRITE_JPEG_QUALITY), 100])
    elif img_path.endswith(".jpg") or img_path.endswith(".jpeg"):
        shutil.copy(img_path, new_img_path)
    else:
        print("Out of supported formats of .jpg and .png")
        return

    new_lbl_path = os.path.join(lbl_dest, f'{item_name}.txt')
    yolo_file = open(new_lbl_path, "w")
    yolo_file.write(yolo_annot)
    yolo_file.close()
```

Tímto postupem byla data připravena a transformována do formátu vhodného pro trénování YOLO modelu. Výsledný dataset byl uložen a uspořádán do trénovacích a validačních adresářů, čímž bylo dokončeno zpracování dat. Tento proces zajistil, že data jsou konzistentní, dobře strukturovaná a připravena pro efektivní trénink neuronové sítě.

8 Trénování neuronové sítě pro detekci objektů

Pro tuto diplomovou práci jsem zvolil síť YOLOv8, která je velmi populární pro detekci objektů. Nejprve jsem využil hardware, který jsem měl k dispozici, a to procesor Intel Core i5-10300H 2.5 GHz s grafickou kartou NVIDIA GeForce GTX 1650 s 4GB VRAM, která má 896 CUDA jader a 16 GB RAM. Během učení neuronové sítě jsem zjistil, že tento hardware není dostatečně výkonný. Kvůli nedostatku paměti probíhala špatně kalkulace validace, což vedlo k tomu, že se negenerovaly validační hodnoty do grafů.

Proto jsem se rozhodl využít platformu Google Colab, která za relativně nízký poplatek 11 eur nabízí značný výpočetní výkon. Google Colab lze snadno propojit s Google Drive, kam lze nahrát dataset a data potřebná pro učení neuronových sítí. Po zakoupení příslušného balíčku je na výběr několik možností pro GPU. Já jsem zvolil NVIDIA T4 s 16 GB VRAM, která poskytuje dostatečný výkon pro výpočty. Platforma dále nabízí 15 GB RAM a 100 GB úložiště na Google Drive pro uložení naučených modelů.

8.1 První model

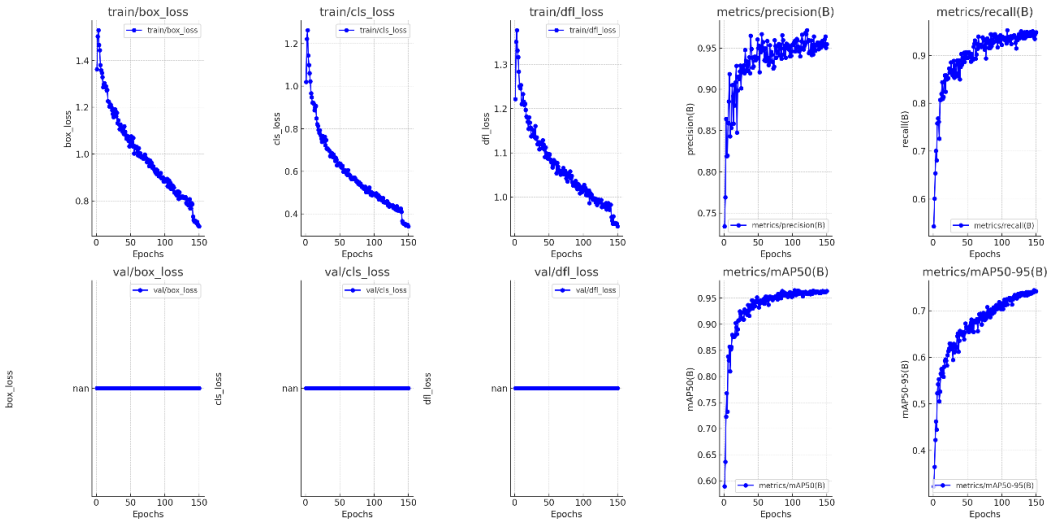
První model jsem učil pouze na datech z prvních tří datasetů bez rozšíření o augmentaci. Ve výsledku bylo tedy použito 1694 obrázků pro trénování a 424 pro validaci. Níže znázornění kód znázorňuje příkazy pro trénování modelu YOLO.

```
import os
import torch
from ultralytics import YOLO, settings

# Constants for path
MODEL_PATH = 'models/yolov8n.pt'
DATASET_DIR = "training_data_preprocessed1/"
if __name__ == "__main__": # For proper training on windows machine
    torch.cuda.empty_cache()
    settings.update({"datasets_dir": os.path.join(os.getcwd(), DATASET_DIR)})
    settings.save()
    # Loading pretrained model
    model = YOLO(MODEL_PATH)
    model.train(data='./data.yaml', epochs=150, imgsz=640, batch=8,
          amp=False, optimizer='AdamW', val=True, plots=True, device=0)
```

Při učení modelu na vlastním hardwaru jsem narazil na řadu problémů, které znemožňovaly efektivní trénink. Jedním z hlavních problémů byla nutnost zakázat automatické míchání přesnosti (AMP), pravděpodobně kvůli nekompatibilitě s mou grafickou kartou. Tento problém vedl k chybným výpočtům ztrátových hodnot a vracení hodnot „nan“ (not a number). Jak jsem zjistil, tento problém je poměrně častý a na fórech je doporučováno řešit jej snížením hodnoty „batch size“, avšak tato metoda ne vždy funguje.

Podobné potíže jsem zaznamenal i při validaci, kde rovněž docházelo k chybám způsobeným grafickou kartou počítače, patrně pro to, že karta nebyla podporována. Po přepnutí na CPU se model naučil detekovat SPZ a validace se počítala správně. Nicméně nepořizoval se záznam validace a trénink modelu pro 150 tréninkových epoch trval přes 21 hodin.



Obrázek 40: Grafické znázornění průběhu učení prvního YOLO modelu

8.1.1 Trénovací ztráty

V průběhu trénování jsem zaznamenal následující výsledky (viz obrázek 40) v jednotlivých hodnotách:

- **train/box loss** – Tento graf dokumentuje ztrátu při učení modelu na detekci ohraničení bounding boxů jednotlivých objektů. z grafu je zřetelný klesající trend, který naznačuje, že každou další epochou se model zlepšuje v lokalizaci objektů ve snímcích. z grafu je zřetelné, že při 150 epochách se hodnota ztráty snížila přibližně z 1.4 na hodnotu 0.6.
- **train/cls loss** – Tento graf reprezentuje ztrátovou hodnotu při klasifikaci objektů v průběhu tréninku. Klesající trend z hodnoty 1.2 na hodnotu nižší než 0.4

indikuje jasně zřetelné zlepšování schopnosti modelu správně klasifikovat objekty.

- **train/dfl loss** – Reprezentuje ztrátu při optimalizaci distribuce odhadů modelu. v tomto případě klesající ztrátová hodnota naznačuje efektivní učení modelu, kde hodnota klesla v průběhu učení z 1.2 na přibližně 0.9.

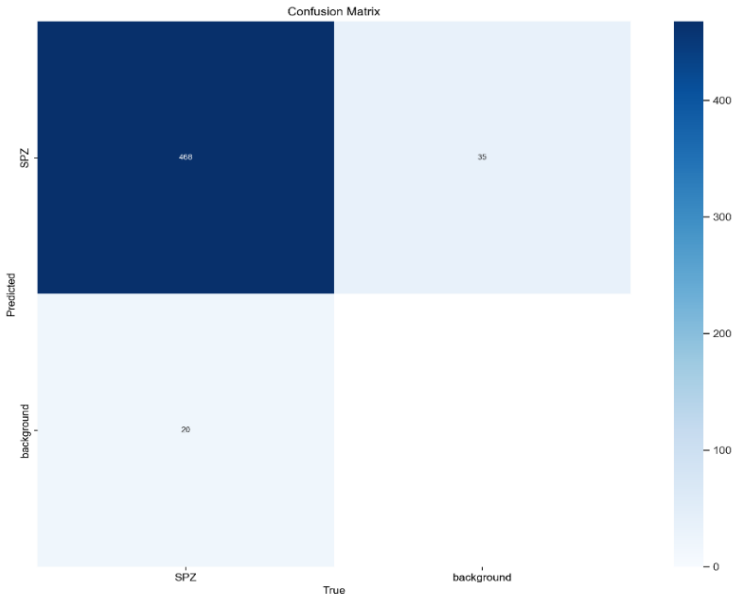
8.1.2 Evaluační metriky

Na obrázku 39 jsou dále znázorněny evaluační metriky:

- **Precision** – Graf přesnosti ukazuje, že model dosahuje vysokých hodnot přesnostI s pohybem hodnot blízko 1, což znamená nízký počet falešně negativních detekcí.
- **Recall** – Tento graf ukazuje schopnost modelu zachytávat relevantní objekty opět s hodnotami blízko hodnotě 1. Výsledek indikuje, že model má nízký počet falešně negativních detekcí.
- **mAP50** – Tento graf ukazuje celkovou přesnost modelu, která se v průběhu učení modelu zvyšovala a následně stabilizovala na vysoké hodnotě 0.96,
- **mAP50-95** – Tento graf znázorňuje podrobnější pohled na výkon modelu. Rostoucí trend jasně indikuje celkově zlepšující se výkon modelu, a to s výslednou hodnotou 0.74.

8.1.3 Matice záměn

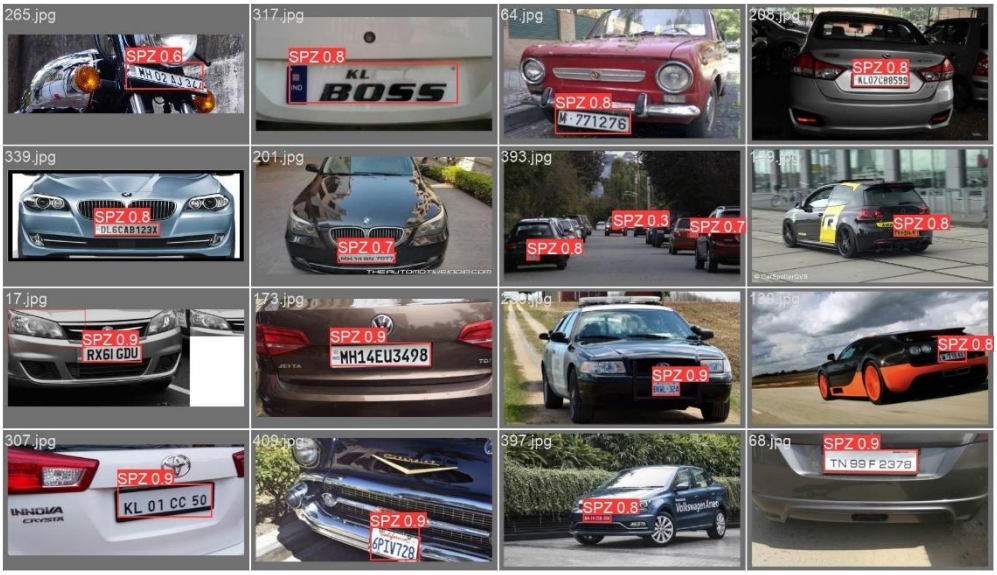
Matice záměn poskytuje přehled o klasifikačních schopnostech modelu. z obrázku 41 níže je patrné, že model správně klasifikoval 468 instancí třídy SPZ jako TP a 35 instancí bylo detekováno jako FP Třída TN nezaznamenala žádnou chybu a jako FN bylo detekováno 20 instancí třídy SPZ. Lze konstatovat, že model má vysokou přesnost při klasifikaci třídy SPZ s relativně malým počtem chyb.



Obrázek 41: Matice záměn prvního modelu (confusion matrix)

8.1.4 Predikce modelu

Na obrázku 42 jsou znázorněny predikce vytvořené naučeným modelem, přičemž je zřejmé, že model dosahuje vysoké přesnosti detekcí. Přesnost predikovaných registračních značek je značně vysoká. Přesto jsem se rozhodl vyzkoušet i model trénovaný s daty, na která byla aplikována augmentace s cílem zvýšit robustnost modelu. z tohoto důvodu jsem se rozhodl vytvořit další model.



Obrázek 42: Vizualizace predikcí prvního modelu

8.2 Druhý model

Kvůli zdlouhavému procesu učení na vlastním hardwaru jsem se rozhodl použít externí zdroje pro trénování druhého modelu. Zvolil jsem předplatné Google Colab, jak bylo

zmíněno výše v kapitole 8. Pro trénování tohoto modelu bylo nakonec použito 2746 obrázků pro trénink a 688 obrázků pro validaci.

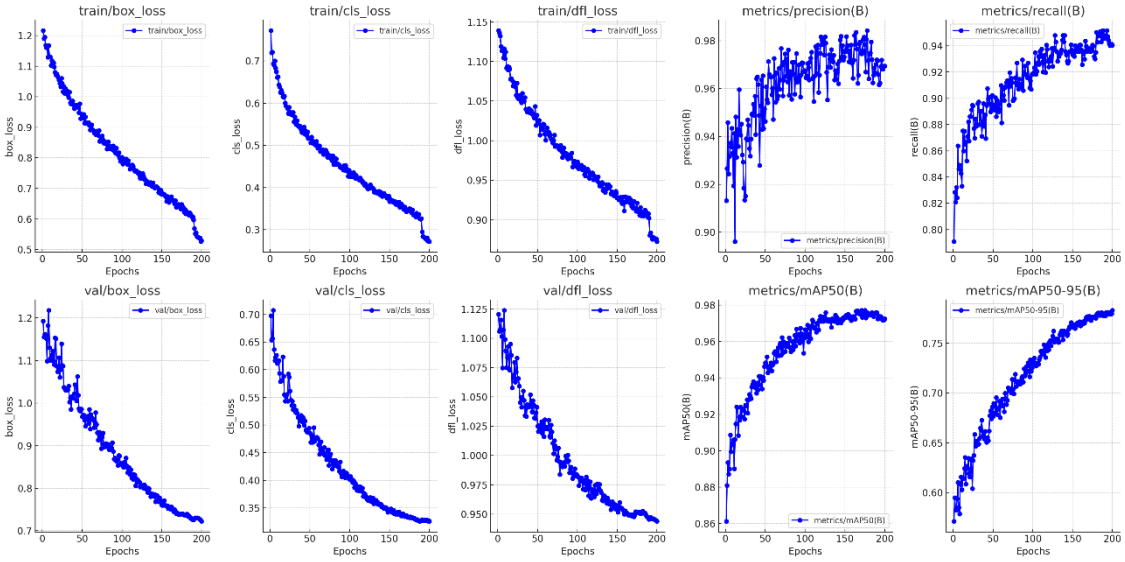
Nejprve bylo nezbytné nahrát dataset na Google Drive, ke kterému se pak připojíme z Google Colab postupem znázorněným níže.

```
from google.colab import drive
drive.mount('/content/gdrive')
import tensorflow as tf
print(tf.test.gpu_device_name())
ROOT_DIR = '/content/gdrive/My Drive/data/'
!pip install ultralytics
```

Díky možnosti využití vyššího výpočetního výkonu prostřednictvím externího hardwaru jsem mohl při trénování modelu nastavit vyšší hodnoty parametrů, což by na mém lokálním zařízení nebylo možné. Oproti tomu Google Colab umožňuje využití výkonných grafických procesorů. Díky tomu jsem mohl například zvýšit počet epoch na 200, velikost dávky (batch size) na 16 a použít optimalizátor 'Adam' s počátečním učícím koeficientem (learning rate) 0,001. Tento vyšší výpočetní výkon zkrátil dobu trénování modelu na přijatelné 4 hodiny a 36 minut, což by na mém lokálním zařízení trvalo mnohem déle. Kód níže reprezentuje trénování modelu.

```
import os
from ultralytics import YOLO
dataYAML = os.path.join(ROOT_DIR,"data.yaml")
dataModel = os.path.join(ROOT_DIR,"preTrainedYolov8n.pt")
model = YOLO(dataModel)
result = model.train(data=dataYAML, epochs=200, imgsz=640,  batch=16,
                     optimizer='Adam', val=True, plots=True, lr0=0.001)
```

8.2.1 Trénovací ztráty



Obrázek 43: Grafické znázornění průběhu učení druhého YOLO modelu

Během trénování druhého modelu byly zaznamenány následující metriky ztrát:

- **train/box loss** – Graf této ztráty ukazuje postupné snižování hodnoty z přibližně 1.2 na 0.52 v průběhu 200 epoch, což ukazuje na lepší výsledek v lokalizaci objektů ve snímcích než první model.
- **train/cls loss** – Tento graf zobrazuje ztrátu klasifikace, která klesá z hodnoty 0.77 na 0.27, což indikuje zlepšování schopnosti modelu správně klasifikovat objekty.
- **train/dfl loss** – Tato ztráta se snižovala z hodnoty 1.13 na hodnotu 0.87, což signalizuje efektivní učení modelu.

8.2.2 Validační ztráty

U tohoto modelu můžeme vyhodnotit i validační ztráty popsané níže:

- **val/box loss** – Tento graf ukazuje validační ztrátu lokalizace bounding boxů objektů. Hodnota ztráty se pohybuje okolo hranice 0.72.
- **val/cls loss** – Validační ztráta klasifikace dosahuje hodnoty 0.32.
- **val/dfl loss** – Validační ztráta distribuce odhadů modelu dosahuje 0.94.

Validační ztrátové hodnoty dosáhly o něco vyšších hodnot, ale nejsou znatelně vyšší než trénovací ztrátové hodnoty, což indikuje, že model je schopný správně se generalizovat na validačním souboru.

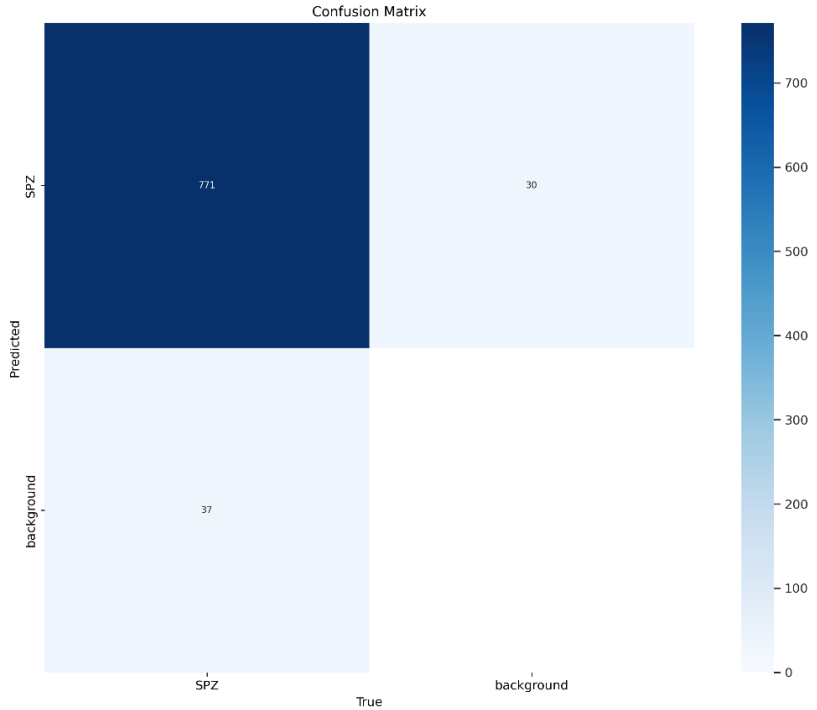
8.2.3 Evaluační metriky

Dále jsou znázorněny evaluační metriky:

- **Precision** – Pro tuto metriku graf zobrazuje hodnoty blízké hodnotě 1, což naznačuje velmi nízký počet falešně pozitivních detekcí.
- **Recall** – Tento graf zobrazuje opět hodnoty blízké 1, což nám značí velmi nízký počet falešně negativních detekcí.
- **mAP50** – Graf střední přesnosti při 50 % IoU ukazuje zvyšující se hodnotu během učení modelu, která se stabilizovala na hodnotě 0.97.
- **mAP50-95** – Poslední graf ukazují výkon modelu s rostoucím trendem, který dosáhl hodnoty 0.78.

8.2.4 Matice záměn

Matice záměn poskytuje přehled o schopnostech modelu klasifikovat různé třídy. z následujícího obrázku 44 vyplývá, že model správně klasifikoval 771 instancí třídy SPZ jako TP a detekoval 30 instancí jako FP. Počet FN pro třídu SPZ byl 37. Toto indikuje vysokou přesnost modelu v klasifikaci třídy SPZ s relativně malým počtem



chyb.

Obrázek 44: Matice záměn druhého modelu (Confusion matrix)

8.2.5 Predikce modelu

Predikce, které provedl trénovaný model, jsou zobrazeny na přiloženém obrázku 45. Je zřejmé, že model dosahuje vysoké přesnosti při detekci a klasifikaci registračních značek. Augmentace dat výrazně přispěla ke zlepšení robustnosti modelu, což se projevilo ve zvýšení celkové přesnosti a snížení chybovosti.

Obrázek 45: Vizualizace predikcí druhého modelu

8.3 Vyhodnocení

Na základě dat uvedených v kapitole 8 jsem se rozhodl vybrat druhý model pro detekci registračních značek. Důvodem je, že tento model dosahuje vyšších přesností a lepšího výkonu ve srovnání s prvním modelem. Trénink na rozšířeném datasetu o augmentaci vedl k výraznému zlepšení schopnosti modelu generalizovat na nových datech.

Evaluační metriky ukazují, že druhý model dosahuje vysoké přesnosti (precision) a citlivosti (recall), přičemž hodnoty obou metrik jsou blízké 1. To indikuje velmi nízký počet falešně pozitivních a falešně negativních detekcí. Metriky mAP50 a mAP50-95 dosáhly hodnot 0.97 a 0.78, což potvrzuje celkově vysoký výkon modelu.

Navíc, díky použití výkonnějšího hardwaru na platformě Google Colab, bylo možné model trénovat s vyššími hodnotami parametrů, což vedlo k lepším výsledkům v mnohem kratším čase. Trénovací a validační ztrátové hodnoty značí efektivní učení modelu a jeho schopnost správně lokalizovat a klasifikovat objekty.

Matice záměn a vizualizace predikcí potvrzují, že model dosahuje vysoké přesnosti při detekci registračních značek. Přidání augmentace dat výrazně přispělo ke zvýšení robustnosti a snížení chybovosti modelu.

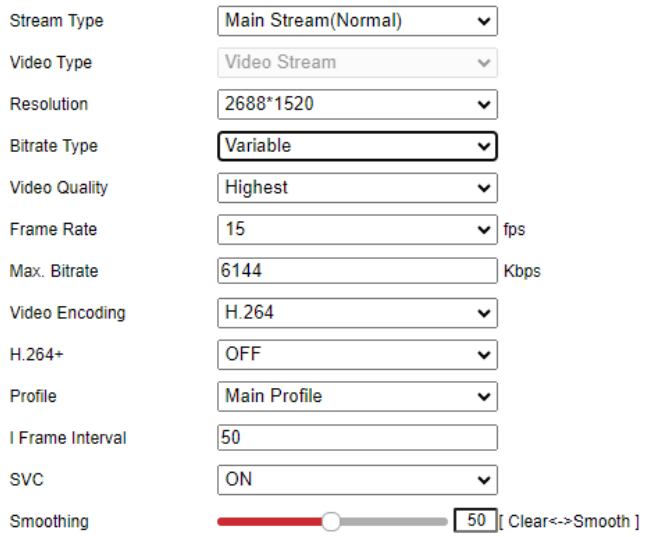
9 Volba optického rozpoznávače znaků

Dalším krokem při zpracování nalezené registrační značky je její přečtení. Tento krok vyžaduje více než jen použití zvoleného OCR, jelikož je nutné snímek nejprve očistit od nepotřebného šumu, který je na zkoumaných snímcích často přítomen. Tato úprava snímku výrazně zvyšuje přesnost použitého OCR při převodu obrazu na text.

9.1 Nastavení kamery

Prvním krokem je nastavení kamery, které zajistí kvalitní vstup pro neuronovou síť určenou k detekci registrační značky. Jak bylo popsáno v kapitole 6, volba kvalitní kamery, která je předurčena pro tuto činnost, je základním předpokladem pro zachycení kvalitního snímku pro následné zpracování.

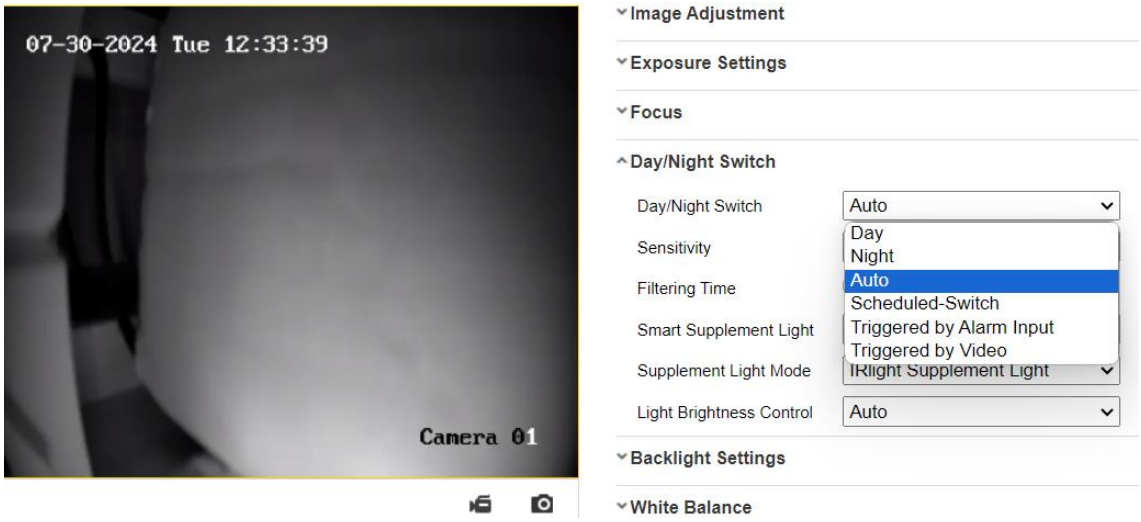
Po připojení kamery k počítači a dokončení úvodní konfigurace pro přihlášení do kamery musíme nastavit základní nastavení pro video. Zde bylo potřeba provést nastavení pro primární přenosový kanál. Mým primárním cílem bylo získat pro systém co nejkvalitnější snímk, který nám může kamera poskytnout.



Obrázek 46: Ukázka nastavení videa kamery

Rozhodl jsem se nastavit rozlišení na nejvyšší dostupné, a to 2688 x 1520, a nastavit nejvyšší kvalitu videa. Dále jsem nastavoval typ datového toku, kde bylo možné vybrat mezi konstantním či variabilním. Zvolil jsem variabilní pro využití přizpůsobivosti datového toku podle adaptivního chování kamery. Dále jsem snížil počet snímků za sekundu (frame rate) na 15, protože se neočekává vysoká rychlost přijíždějícího automobilu k bráně parkoviště, a tudíž bude tato hodnota dostačující.

S ostatními parametry jsem během práce experimentoval a doporučil bych dále nastavovat parametry kódování videa a vyhlazování (smoothing), které „zjemní“ obraz, avšak na úkor odstranění detailů snímku. Proto je zapotřebí řádně tyto parametry testovat. z mého testování vyplynulo, že pro zpracování obrazu popsané níže spíše nastavování dalších parametrů detaily spíše rozostřuje, a proto jsem je ponechal na doporučených hodnotách.

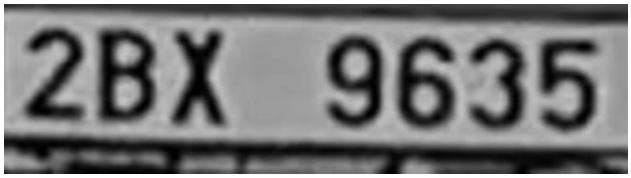


Obrázek 47: Nastavení nočního / denního režimu

Je důležité nastavit kameře denní a noční režim (viz obrázek 47), aby bylo možné číst registrační značky i za špatných světelných podmínek, například během večerních hodin. Nastavení režimu je možné provést několika způsoby: od spouštění poplachu, přes přepínání mezi nočním a denním režimem v předem stanovených časech, až po automatický režim. v automatickém režimu se kamera přepne do nočního režimu, jakmile detekuje nedostatek světla.

9.2 Image processing

V rámci zpracování obrazů pro detekci a rozpoznávání textu, zejména ve specifických aplikacích, jako je čtení poznávacích značek vozidel, se často používají pokročilé metody filtrace obrazu ke zvýšení přesnosti detekce. Metoda popsaná v uvedeném kódu představuje komplexní sekvenci filtrů a zpracovatelských kroků, které mají za cíl zvýšit kvalitu obrazu před jeho analýzou pomocí OCR. Pro demonstraci těchto úprav jsem nahrál videozáznam automobilu, který se postupně přibližuje, aby nasimuloval situaci u brány parkoviště.



Obrázek 48: Vystřižená značka před zpracováním

Na základě rozsáhlé literatury, kterou jsem prostudoval, jsem dospěl k závěru, že pokročilé metody zpracování obrazu jsou klíčové pro dosažení co nejvyšší přesnosti detekce. Použití pouze základních úprav není dostatečné pro dosažení optimálních výsledků.

Prvním krokem bylo aplikování filtru pro převod barevného obrazu na odstíny šedé, což je standardní metoda předzpracování obrazu. Následně jsem aplikoval Gaussovské rozostření za účelem redukce šumu. Je důležité pečlivě nastavit parametry tohoto filtru, protože příliš vysoké hodnoty mohou snímek znehodnotit.

Dalším krokem bylo adaptivní prahování pro segmentaci obrazu, které umožňuje lepší oddělení textu od pozadí. Následně jsem provedl inverzi barev, což změní text na bílou barvu a pozadí na černou.

```
def applying_filters(spz_cropped_array, b_width, logger):
    spz_gray= cv2.cvtColor(spz_cropped_array,cv2.COLOR_GRAY2BGR)
    # Gaussian filter
    spz_gauss = cv2.GaussianBlur(spz_gray, (5,5), 0) #11,11
    # Adaptive thresholding
    spz_threshold = threshold_local(spz_gauss,99,offset=5, method='gaussian')
    thresh = (spz_gauss > spz_threshold).astype('uint8') * 255
    # Inverting colors (white objects, black background)
    thresh = cv2.bitwise_not(thresh)
    # Resize image
    height = int(spz_gauss.shape[0] * b_width/ spz_gauss.shape[1])
    spz_gaussian = cv2.resize(spz_gauss, (b_width, height))
    thresh = cv2.resize(thresh, (b_width, height))
    # Morphological operation
    kernel = np.ones((3, 3), np.uint8)
    thresh = cv2.morphologyEx(thresh,cv2.MORPH_OPEN,kernel,iterations=1)
    # Ensure the image is single-channel
    if len(thresh.shape) == 3:
        thresh = cv2.cvtColor(thresh, cv2.COLOR_BGR2GRAY)
    # Ensure the image is single-channel
    if len(thresh.shape) != 2:
       logger.error("The image passed to findContours is not single-channel")
```

```
    # Contour detection
    contours, _ = cv2.findContours(thresh,cv2.RETR_EXTERNAL,
                                   cv2.CHAIN_APPROX_SIMPLE)
    mask = np.zeros(thresh.shape, dtype="uint8")
    for cnt in contours:
        x, y, w, h = cv2.boundingRect(cnt)
        aspect_ratio = w / float(h)

        # Filter contours likely to be part of the text
        if cv2.contourArea(cnt) < 700:
           cv2.drawContours(thresh, [cnt], -1, 0, -1)
        elif 0.2 < aspect_ratio < 5 and h > 15:  # Remove small contours
            cv2.drawContours(mask, [cnt], -1, 255, -1)
            # Bitwise AND to isolate the text
     final = cv2.bitwise_and(thresh, thresh, mask=mask)
     # Inverting colors again
     spz_invert = cv2.bitwise_not(final)
     return spz_invert
```

Poté jsem upravil velikost snímku pro zajištění konzistence rozměrů před dalším zpracováním a aplikoval morfologické operace k odstranění menšího šumu z binárního obrazu. Během testování se ukázalo, že tato korekce šumu nebyla dostatečná, a proto jsem přistoupil k detekci kontur. Při tomto kroku jsem identifikoval a odstranil menší objekty, které nelze považovat za součást textu. Tento přístup funguje velmi dobře při detekcích na krátké vzdálenosti, avšak při použití na větší vzdálenosti má nežádoucí účinky, protože se text registrační značky často stává nečitelným či zdeformovaným.

Úprava snímku byla dokončena aplikací masky a následnou inverzí barev, což umožnilo izolovat textové oblasti od pozadí a zvýraznit relevantní informace pro OCR. Následující obrázek 49 ilustruje postupné stavy úprav snímku.

Obrázek 49: Jednotlivé stavy během úpravy snímku

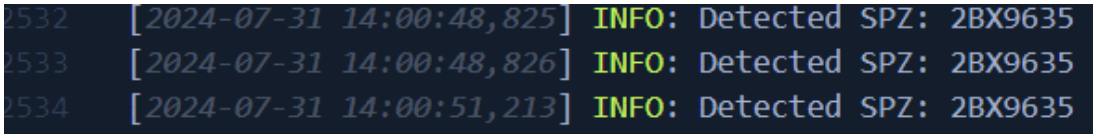
9.3 Testované OCR

Před implementací na Raspberry Pi bylo provedeno otestování na notebooku s parametry uvedenými v kapitole 8, aby se zamezilo případnému přehodnocování během finálního nasazení.

Pro otestování přesnosti a rychlosti jsem zvolil dva zástupce, a to Tesseract a EasyOCR. Oba by měli prokázat podobné výsledky při detekci, avšak pro nasazení na zařízení jako je Raspberry jde hlavně o limitovaný výkon zařízení. Pokud tedy rychlost nebude vyhovující na testovaném notebooku, nemá význam zvažovat jeho nasazení na Raspberry Pi.

9.3.1 Test Tesseractu

Získané výsledky ukázaly značné zlepšení v rychlosti, kdy identifikace automobilu s registrační značkou a její následné přečtení z kamerového záznamu bylo dokončeno během několika milisekund. Přitom čistá detekce bez přečtení znaků trvala 150 milisekund. Celková doba detekce s přečtením textu registrační značky byla okolo půl sekundy, kde určitě bude značné zpomalení v rychlosti detekcí na zvoleném Raspberry.



Obrázek 50: Ukázka úspěšné detekce se čtením registrační značky pomocí Tessaeractu

Na obrázku 50 je ilustrován tento výkon, který demonstruje úspěšnou detekci a čtení registrační značky. Přes tyto úspěchy však systém čelí omezením v situacích, kdy detekuje registrační značku, ale není schopen z ní přečíst jakýkoliv text. To vede k provádění neúčelných detekcí, které sice trvají pouze zlomek času potřebného pro kompletní detekci, ale nevedou k žádnému praktickému výsledku. Přesnost Teseractu je velmi odvozená od použitých trénovacích dat. v základním nastavení se používá model „eng“, ale lze také stáhnout model „leu“, se kterým v této práci pracuji. Tento naučený model byl dříve využíván také známým, nyní již nefunkčním, OCR systémem OpenALPR.

9.3.2 Test EasyOCR

Tento OCR systém ukázal vyšší úroveň přesnosti při detekci číslic, avšak detekce písmen se ukázala být značně problematická, což negativně ovlivňuje celkovou rychlost detekce. Každý znak je detekován přibližně za 0,3 sekundy, díky čemuž celkový čas

potřebný pro detekci standardní registrační značky dosahuje 4 až 5 sekund. Navíc, finální proces sestavování přečteného textu trvá více než 4 sekundy, přičemž jeho přesnost výrazně zaostává za výsledky dosaženými systémem Tesseract.

```
0: 384x640 1 car, 14.0ms
Speed: 5.0ms preprocess, 14.0ms inference, 4.0ms postprocess per image at shape (1, 3, 384, 640)

0: 576x640 1 SPZ, 18.5ms
Speed: 6.0ms preprocess, 18.5ms inference, 4.0ms postprocess per image at shape (1, 3, 576, 640)
Detection completed in 0.10 seconds.
Detection completed in 0.20 seconds.
Detection completed in 0.31 seconds.
Detection completed in 0.20 seconds.
Detection completed in 0.19 seconds.
Detection completed in 0.19 seconds.
Detection completed in 0.26 seconds.
28965
```

Obrázek 51: Ukázka průběhu čtení textu EasyOCR

Na obrázku 51 je zobrazen průběh čtení textu pomocí EasyOCR. Bohužel, vzhledem k jeho přesnosti vyvolal tento systém zklamání, jelikož na testovacím videozáznamu nebyl schopen správně přečíst ani jednu registrační značku a ani nedokázal identifikovat správný počet znaků.

9.4 Vyhodnocení výběru OCR

Po důkladném testování dvou OCR systémů, Tesseract a EasyOCR, byl pro finální implementaci na Raspberry Pi vybrán Tesseract. Hlavním rozhodujícím faktorem byla vyšší přesnost Tesseractu při detekci a čtení textu z registračních značek. Ačkoliv EasyOCR vykazoval srovnatelnou rychlost detekce, rozdíl v rychlosti mezi oběma systémy nebyl natolik významný, aby ospravedlnil nižší přesnost čtení. v kontextu našeho nasazení je přesnost klíčová, jelikož systém musí spolehlivě a správně identifikovat registrační značky vozidel, a to i za cenu mírně delšího času detekce.

S ohledem na technická omezení Raspberry Pi bylo očekáváno, že rychlost detekce na této platformě se zhorší přibližně tří až čtyřnásobně ve srovnání s testy provedenými na notebooku. Po nasazení systému bylo potvrzeno, že tato predikce byla přesná. Doba potřebná pro detekci registračních značek na Raspberry Pi se skutečně zvýšila a dosahuje hodnot až 1500 milisekund pouze pro neuronovou síť bez OCR. Tento pokles rychlosti je důsledkem nižší výpočetní kapacity Raspberry Pi ve srovnání s notebookem. i přes tyto výsledky bylo prioritou zachování vyšší přesnosti, kterou Tesseract během testování poskytoval.

10 Aplikace pro správu přístupu

10.1 Specifikace IS projektu

Jak již bylo zmíněno v úvodní kapitole, tato práce navazuje na projekt „Campus parking management“ vytvářený studenty z předmětu IS projekt, jejož cílem bylo vytvoření přístupového ekosystému pro parkovací prostory vlastněné JU.

10.1.1 Popis zadání projektu

Hlavním cílem IS projektu bylo vytvoření informačního systému pro správu univerzitních parkovišť, který uživatelům po přihlášení umožňuje spravovat své profily, poskytuje informace o mapě a kapacitách parkovišť a umožňuje generování jednorázových QR kódů pro vjezd. Tento systém také měl poskytovat rozličné funkce pro různé uživatelské role. Studenti by měli mít v aplikaci možnost přístupu k informacím o zbývajícím čase povoleného parkování, zatímco zaměstnanci by mohli spravovat seznam svých vozidel a generovat jednorázové QR kódy pro své hosty. Administrátoři systému by měli mít k dispozici nástroje pro nastavení rolí a práv uživatelů, správu jednotlivých parkovišť a možnost nastavení notifikací pro uživatele.

Systém navíc měl umožnit monitorování trendů zaplnění parkovišť, a zobrazit je v aplikaci. Pro zajištění bezpečnosti a správnosti provozu měly být implementovány různé bezpečnostní protokoly, včetně pravidelného testování a ladění systému. Tento komplexní informační systém byl navržen tak, aby vyhovoval potřebám univerzity a zároveň zjednodušoval každodenní správu a užívání parkovacích ploch.

10.1.2 Komunikace mezi projekty

Mezi mým systémem a systémem IS projektu probíhá komunikace, kterou lze rozlišit na dva směry:

Žádost o data

Do této kategorie lze zahrnout žádost o data, která obsahují povolené a blokované registrační značky, uživatele, uživatelské skupiny a jejich oprávnění, základní nastavení a QR kódy. o tato data se žádá při startu aplikace a posléze každých patnáct minut, avšak tato hodnota je nastavitelná prostřednictvím „.env“ konfiguračního souboru.

Dále máme žádosti o data o neznámém uživateli či QR kódu, která se využívají během vyhodnocování již získaných dat z databáze. Pokud v lokální databázi není

uživatel nebo QR kód nalezen, provede se ještě dotaz na endpoint, zda se nezměnila data od posledního natažení. Tímto přístupem se zajistí funkčnost vygenerovaných QR v časovém okně mezi aktualizací dat.

Posílání záznamů

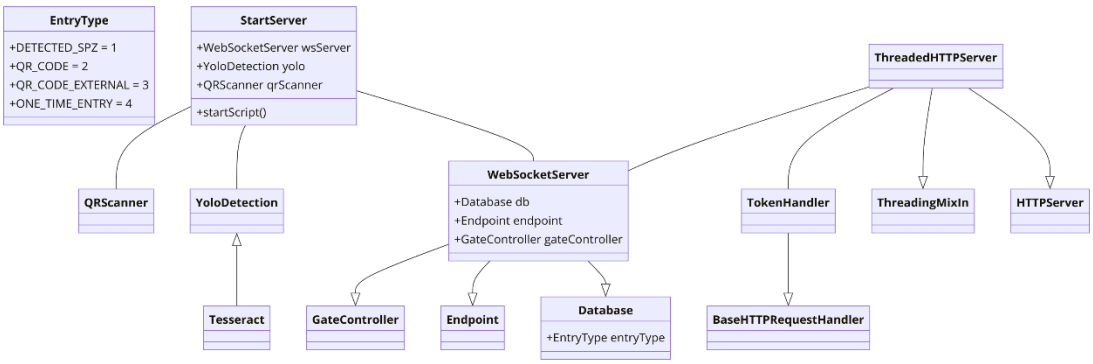
Protože oba systémy musí umožňovat obousměrnou komunikaci, jsou zahrnuty logovací HTTP POST žádosti, které umožňují zaznamenávat data pro IS projekt týkající se vjezdů a výjezdů vozidel. Tato data zahrnují komplexní informace o každém vozidle, které projede bránou, včetně textu a obrázku registrační značky, času průjezdu, údajů o uživateli a identifikátoru jeho uživatelské skupiny, pokud jsou tyto informace dostupné. Kromě toho, pokud byl pro vjezd použit QR kód, je tento odeslán jako další parametr.

10.2 Schéma aplikace

Tento systém je založen na souboru služeb běžících na serveru, které společně zajišťují všechny klíčové funkce, od zpracování obrazu po správu dat a uživatelské interakce.

Detekce registračních značek je implementována prostřednictvím naučeného modelu YOLO, který zpracovává obrazový vstup z kamery a identifikuje oblasti obsahující registrační značky. Tyto detekované značky jsou následně zpracovány technologií OCR Tesseract pro extrahování textu ze zkoumaného snímku. Paralelně s detekcí registračních značek se provádí skenování QR kódů pro alternativní vjezd.

Hlavní částí aplikace je WebSocket server, který zajišťuje komunikaci mezi serverem a klientem prostřednictvím soketů, posílaných vždy se specifickým typem pro provedení žádané operace. Podle typu zprávy poté na displej, znázorněný na obrázku 51, zobrazují informace o povolení či zamítnutí vstupu.



Obrázek 52: Class diagram systému

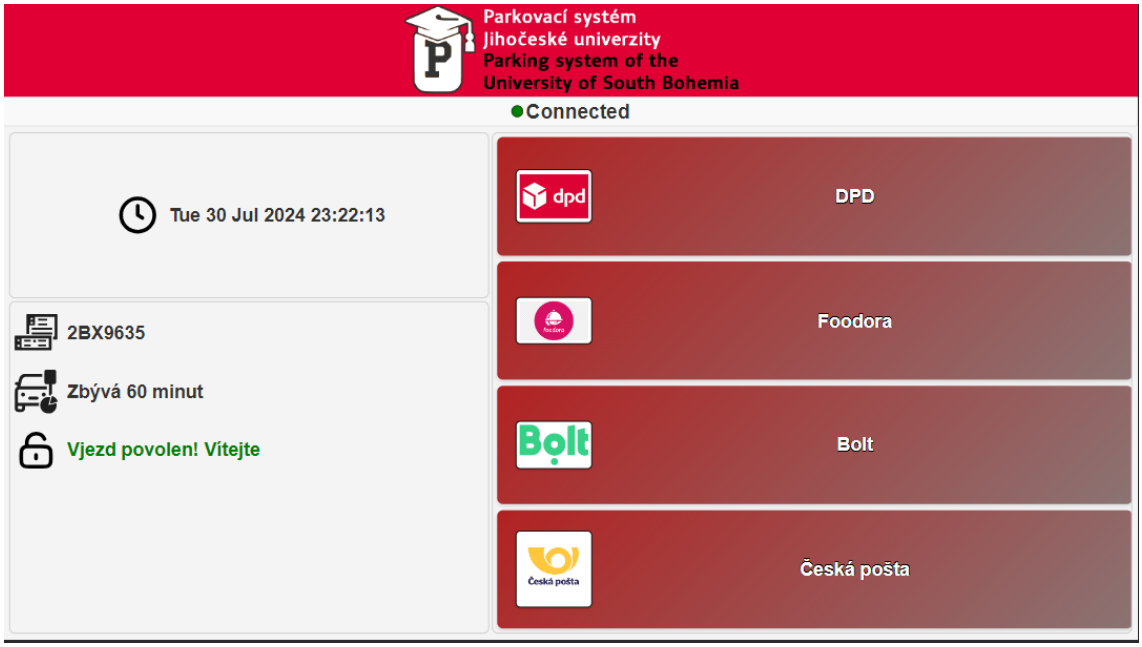
Backendové operace, včetně ověřování oprávnění vozidel a logování přístupů, jsou řízeny zvolenou databází SQLite3. Databáze je rozdělena na dvě, kde první ukládá data od endpointu a druhá ukládá data pro systém.

10.3 Řešení vstupu na parkoviště

Pro umožnění jednorázového vstupu bylo zapotřebí udělat jednoduchý, ale přehledný výstup na displej, kde se zobrazují vyhodnocení vstupu. Když je vstup povolen, zobrazí se detekovaná registrační značka, čas v minutách, jak dlouho může uživatel parkovat, a zpráva o povolení či zamítnutí vjezdu. Při zamítnutí vjezdu se zobrazí čtvrtá informace, která reprezentuje oznámení proč byl vjezd zamítnut.

10.3.1 Jednorázový vjezd

Jednorázový vstup je vyhodnocován pomocí obecného nastavení brány, které se přijímá z endpointu IS projektu. Toto nastavení má pouze dva stavy, a to povoleno či zakázáno. Při zmáčknutí tlačítka na displeji se zastaví všechny detekce vstupu, tedy čtení registračních značek i QR kódů, a zablokují se tlačítka, dokud se nevyhodnotí vjezd. Toto omezení platí při kterémkoli vstupu, aby se zabránilo duplicitním detekcím.



Obrázek 53: Ukázka zobrazení na displeji

Při jakémkoli typu vstupu se do logů ukládá jak obrázek, tak text registrační značky, kterou můžeme považovat za univerzální identifikátor. Pokud by systém chybně vyhodnotil registrační značku, stále máme k dispozici snímek, který obsahuje potřebné informace.

10.3.2 Vstup přes detekovanou registrační značku

Pro zvýšení přesnosti a minimalizaci chybných detekcí byl implementován algoritmus, který vyžaduje určitý počet shodných detekcí. Například musí být provedeny minimálně dvě detekce, kde je registrační značka identifikována stejně, než dojde k povolení vstupní kontroly.

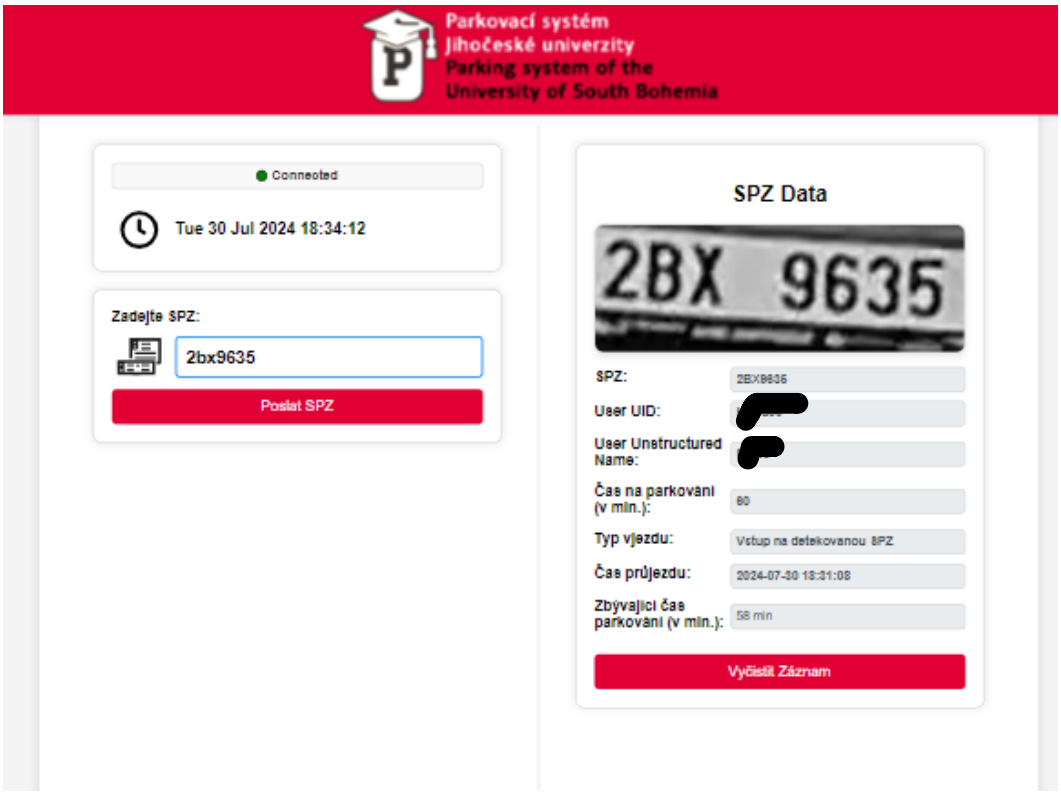
Vstupní kontrola povolení je závislá na několika faktorech, včetně existence značky v systému a jejího stavu (například zda je značka blokována). Dále kontrola zahrnuje ověřování oprávnění uživatele, která jsou specifická pro skupinu, do které uživatel patří. Tato oprávnění mohou zahrnovat omezení parkovacích dob nebo limit pro počet vozidel, která mohou být zaparkována uživatelem z dané skupiny.

10.3.3 Vstup přes QR kód

Pro tento vstup platí pravidla podobná k těm, která se vztahují na detekci registrační značky. Provádí se kontrola existence kódu v systému, stejně jako kontrola uživatele a jeho příslušnosti k uživatelské skupině. Při povolení vjezdu se navíc ukládá poslední detekovaná registrační značka, aby bylo možné zasílat logy s co nejvíce informacemi o vjezdu.

10.4 Webové rozhraní pro validaci povolení parkování

Jedním z hlavních cílů této práce bylo také umožnění ověření informací o zaparkovaném automobilu prostřednictvím jednoduchého webového rozhraní. Toto rozhraní zasílá požadavky na WebSocket server, které obsahují typ socketu a registrační značku hledaného vozidla. Systém poté prohledává databázi zaparkovaných vozidel a v případě identifikace shody poskytne kompletní informace o vozidle, což je vizualizováno na přiloženém obrázku. Aby se minimalizovala možnost chyb při zadávání registrační značky uživatelem, je zadaná registrační značka převedena do standardního formátu používaného v aplikaci.



Obrázek 54: Ukázka webového prostředí pro kontrolu vjezdů

Toto rozhraní zobrazuje kompletní informace o vjezdu, včetně typu použitého vjezdu. Například v případě, že by student zneužil jednorázový vjezd, systém automaticky zaznamenává poslední detekovanou registrační značku před aktivací tlačítka na displeji. Záznam zahrnuje jak přečtený text, tak i obrázek registrační značky. Tímto způsobem je zajištěno, že i v případě chybného vyhodnocení registrační značky zůstává snímek dostupný pro možnou korekci administrátorem.

11 Implementace řešení do Raspberry Pi

Výběr operačního systému je klíčovou volbou při nasazení aplikace na Raspberry Pi 4, zvláště pokud aplikace vyžaduje značné výpočetní zdroje. Pro tento projekt byla zvolena odlehčená verze operačního systému Raspberry Pi OS Lite 64-bit. Toto řešení bylo zvoleno převážně pro zajištění optimálního výkonu a stabilitu nasazené aplikace.

11.1 Instalace operačního systému

Instalace byla možná pomocí Raspberry Pi Imager, který po spuštění na počítači nabízí uživateli volbu požadovaného operačního systému ze seznamu dostupných obrazů, které zahrnují i oficiální Raspberry Pi OS. Připojil jsem microSD kartu k počítači a spustil Raspberry Pi Imager, ve kterém jsem nejprve vybral typ zařízení, poté jsem zvolil požadovaný operační systém a poté se zahájila instalace.

Hlavní výhodou odlehčené verze operačního systému, pro kterou jsem ho zvolil, je, že neobsahuje grafické uživatelské rozhraní a další nepotřebné služby, které by zbytečně spotřebovávaly systémové zdroje. To umožňuje alokovat větší část dostupného výpočetního výkonu přímo pro běh aplikací s využitím YOLOv8 s Tesseractem.

11.2 Konfigurace softwarového prostředí

Po prvním přihlášení jsem aktualizoval systém a poté provedl instalaci dalších součástí systému, které jsou pro tento projekt zapotřebí.

11.2.1 Instalace softwaru pro Argon case

K Raspberry Pi jsem pořídil kryt Argon s ventilátorem a rozšiřujícím slotem pro HDMI porty. Kompletní běh aplikace na tomto zařízení by se bez ventilátoru neobešel. Jsem stáhl instalační skript a poté jsem nastavil permanentní rychlost ventilátoru na 90 %. Tato hodnota je dostačující mimo jiné proto, že zátěž procesoru není 100%.

```
curl https://download.argon40.com/argon1.sh | bash
argnone-config
```

11.2.2 Instalace LightDM, X serveru a Openbox

Dalším nezbytným krokem pro spuštění aplikace bylo zobrazení uživatelského rozhraní na displeji připojeném k Raspberry Pi pro zobrazení výstupu z kontrol vjezdů a umožnění použití jednorázového vstupu. Pro zajištění zobrazení jsem použil light display manager umožňující spouštění grafických relací, zatímco X server poskytl základní grafický výstup.

```
sudo apt-get install lightdm xserver-xorg

sudo apt-get install openbox
```

Dále jsem nainstaloval Openbox, což je vysoce konfigurovatelný správce oken, který zajišťuje efektivní správu grafického prostředí. T oto řešení jsem zvolil z důvodu minimální spotřeby systémových zdrojů.

11.2.3 Instalace Chromia a displeje

Chromium je open-source webový prohlížeč, který jsem použil pro zobrazení uživatelského rozhraní aplikace. Chromium je lehký a rychlý, což z něj činí vhodnou volbu pro zařízení s omezenými zdroji.

```
sudo apt-get install chromium-browser
```

Pro správné fungování displeje bylo nutné provést několik kroků, které zahrnovaly úpravu konfigurace X serveru a instalaci potřebných ovladačů. Nejdříve jsem nastavil konfigurační soubor, abych umožnili přístup vybraným uživatelům.

```
sudo nano /etc/X11/Xwrapper.config
```

Do tohoto souboru nastavíme přístup uživatelům připojeným ke konzoli, což však může příležitostně způsobovat problémy se spouštěním aplikace v Dockeru. Vhodnější variantou by bylo nastavit přístup všem, což ale může znamenat bezpečnostní riziko.

```
allowed_users=console

allowed_users=anybody
```

Dále bylo nezbytné vytvořit, pokud neexistuje v domovském adresáři každého uživatele, který má mít přístup k X serveru specifický soubor „.Xauthority“.

```
touch ~/.Xauthority
chown $USER:$USER ~/.Xauthority
chmod 600 ~/.Xauthority

xhost +SI:localuser:$USER
```

Pro zajištění správného fungování displeje připojeného k Raspberry Pi 4 jsem nainstaloval specifické ovladače pro framebuffer a upravil konfigurační soubor X serveru. Tímto způsobem optimalizujeme grafický výstup a zajistíme, že zobrazování na displeji bude plynulé a efektivní.

```
sudo apt-get install xserver-xorg-video-fbturbo
```

Po instalaci potřebných ovladačů jsem upravil konfigurační soubor X serveru, aby využíval nainstalované ovladače a optimalizoval grafický výstup. Konfigurace zahrnuje nastavení zařízení, monitoru, obrazovky a rozvržení serveru.

```
sudo nano /etc/X11/xorg.conf

Section "Device"
    Identifier "Card0"
    Driver "modesetting"
    Option "AccelMethod" "none"
EndSection

Section "Monitor"
    Identifier "Monitor0"
    Option "DPMS" "false"
EndSection

Section "Screen"
    Identifier "Screen0"
    Device "Card0"
    Monitor "Monitor0"
    DefaultDepth 24
    SubSection "Display"
        Depth 24
        Modes "1920x1080"
    EndSubSection
EndSection

Section "ServerLayout"
    Identifier "DefaultLayout"
    Screen "Screen0"
EndSection
```

Některé hodnoty se lišily v závislosti na připojeném displeji. Jednoduše jsem to ověřil pomocí příkazu pro získání informací o připojeném displeji. Většina konfliktů vycházejících z tohoto konfiguračního souboru byly špatně zadané specifikace monitoru jako je například podporované rozlišení.

```
fbset
```

11.2.4 Instalace Dockeru

Docker je moderní platforma, která umožňuje spouštět aplikace v izolovaných kontejnerech. Nasazení aplikace na Raspberry Pi 4 prostřednictvím Docker zaručuje běh aplikace ve vlastním prostředí a tím ho činní nezávislým na ostatních aplikacích spuštěných na daném systému, čímž minimalizuje možné konflikty závislostí.

Pro instalaci Dockeru na Raspberry Pi 4 jsem nejprve zajistil, aby měl systém nainstalované nástroje pro správu certifikátů a stahování souborů. Použil jsem k tomu následující skript, který zajišťuje správnou instalaci Dockeru přímo z oficiálních repozitářů Dockeru, což zaručuje kompatibilitu s architekturou ARM a specifikacemi Raspberry Pi[10].

```
# Add Docker's official GPG key:
sudo apt-get update
sudo apt-get install ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/debian/gpg -o
/etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc

# Add the repository to Apt sources:
echo \
  "deb [arch=$(dpkg --print-architecture) signed-
by=/etc/apt/keyrings/docker.asc]
https://download.docker.com/linux/debian \
  $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update
```

Aby bylo možné spouštět Docker příkazy bez nutnosti použití administrátorských práv, je třeba přidat aktuálního uživatele do skupiny „docker“. Tento krok zajišťuje, že uživatel bude mít správná oprávnění pro manipulaci s kontejnery.

```
echo $USER

sudo groupadd docker

sudo usermod -aG docker $USER
```

[10] Dostupný z: https://docs.docker.com/engine/install/debian/.

```
newgrp docker
```

Posledním krokem byla samostatná instalace Dockeru a jeho závislostí pomocí následujícího příkazu.

```
sudo apt-get install docker-ce docker-ce-cli containerd.io
docker-buildx-plugin docker-compose-plugin
```

11.3 Sestavení Docker kontejneru s aplikací

V této kapitole se podrobně zaměřím na proces sestavení a nasazení aplikace pomocí Docker kontejneru na Raspberry Pi 4. Tento postup zahrnuje stažení aplikace z git repozitáře, sestavení Docker obrazu, konfiguraci nezbytných závislostí a služeb a spuštění kontejneru.

11.3.1 Stažení aplikace z GitHub repozitáře

Prvním krokem je stažení nejnovější verze aplikace z git repozitáře. Vzhledem k tomu, že GitHub nyní používá ověřování pomocí Personal Access Token (PAT), bylo nutné použít tento token pro klonování repozitáře. Následující příkaz ilustruje, jak toho dosáhnout:

```
git clone https://<TOKEN>@github.com/<user>/parkingJCU.git
/home/<user>/parkingJCU
```

11.3.2 Dockerfile

Pro sestavení Docker kontejneru jsem vytvořil „Dockerfile“, který obsahoval instrukce pro sestavení, tedy tento soubor obsahuje kroky nutné k vytvoření prostředí, ve kterém aplikace poběží.

V postupu je zprvu definována architektura Docker obrazu, který musí být optimalizovaný pro architekturu ARM64 podporovanou Raspberry Pi, a je použita verze Python 3.9. Dále je nastaven pracovní adresář uvnitř kontejneru a zkopírován zdrojový kód aplikace do tohoto adresáře. Následně se instalují nezbytné balíčky a knihovny, stahují jazykové sady pro Tesseract OCR a nastavují se omezení počtu vláken pro optimalizaci výkonu. Poté se zkopírují závislosti pro Python z předpřipraveného souboru requirements.txt, (viz přílohu B této práce).

Skript pokračuje nakopírováním konfigurace pro správné detekování a zpracování připojených zařízení a konfigurace pro Nginx, který funguje jako reverzní

proxy server. Nakonec skript otevře potřebné porty pro aplikaci a spustí Nginx a aplikaci po startu kontejneru. Ukázka Dockerfile je uvedena v příloze C.

11.4 Konfigurace Nginx

Konfigurační soubor „nginx.conf“ definuje nastavení serveru Nginx a specifikuje způsob, jakým budou HTTP požadavky směrovány na různé části aplikace. Soubor jsem strukturoval do několika sekcí, které upravují obecné parametry serveru a stanovují konkrétní pravidla pro směrování požadavků. Tyto požadavky mohou být například směrovány na různé backendové služby, jako je websocket server běžící uvnitř Docker kontejneru.

Při konfiguraci jsem nastavil počet pracovních procesů a počet simultánních připojení, které může každý proces obsloužit. Dále jsem v konfiguraci obsáhl základní nastavení pro HTTP server, včetně zahrnutí MIME typů, nastavení výchozího typu souboru, efektivního přenosu souborů a timeoutu pro udržování připojení. Server blok specifikuje, že server bude naslouchat na portu 80 pro všechny HTTP požadavky. Různé cesty jsou směrovány na odpovídající cíle: požadavky na kořenovou URL jsou směrovány na HTML soubory, požadavky na podadresáře /css a /js na odpovídající adresáře, Websockety jsou přesměrovány na backendový server běžící na portu 8765 a požadavky na API a administrativní rozhraní jsou směrovány na porty 8000 a 8081. Tato konfigurace se provádí při každém spuštění kontejneru s aplikací a zajišťuje, že všechny části aplikace jsou správně dostupné a optimalizuje přenos dat mezi klienty a serverem. Kompletní ukázka konfiguračního souboru je uvedena v příloze D.

11.5 Přístup pro čtečku kódů

Pro správné fungování QR skeneru, který byl připojen přes USB, bylo nezbytné zajistit, aby zařízení bylo správně rozpoznáno a jeho připojení povoleno. Tento proces vyžadoval úpravu pravidel „udev“, což je správce zařízení v systému Linux. Následující postup charakterizuje kroky, které byly třeba k identifikaci připojeného zařízení a následné vytvoření pravidel „udev“.

Nejprve je třeba zjistit, jaká zařízení jsou připojena k Raspberry Pi. To lze provést pomocí příkazu „lsusb“, který vypíše seznam všech připojených USB zařízení:

```
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 001 Device 020: ID 060e:16c7 Transmonde Technologies, Inc. NT1640S
Bus 001 Device 002: ID 2109:3431 VIA Labs, Inc. Hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

Výpis výše ukázal 4 připojená zařízení, přičemž čtečka kódů je druhé zařízení na Bus 001. Pro vytvoření pravidla, aby Docker kontejner měl přístup ke čtečce, jsem potřeboval znát identifikátor výrobce „idVendor“ a identifikátor produktu „idProduct“. Ve výše uvedeném výstupu se jedná o část za atributem „ID“, kde jsou uvedeny ve formátu „idVendor:idProduct“. Pokud bych potřeboval znát podrobnější informace, mohl bych si je vypsat pomocí následujícího příkazu:

```
lsusb -D /dev/bus/usb/001/006
```

Po zjištění potřebných informací jsem vytvořil soubor „99-usb.rules“, který se kopíruje vždy při sestavování kontejneru. Toto pravidlo je nezbytné pro zajištění správného fungování QR skeneru připojeného k Raspberry Pi.

```
# /etc/udev/rules.d/99-usb.rules

SUBSYSTEM=="usb", ATTR{idVendor}=="060e", ATTR{idProduct}=="16c7",
MODE="0666"
```

11.6 Sestavení a spuštění kontejneru

Krokem pro nasazení aplikace na Raspberry Pi bylo sestavení Docker obrazu. Tento krok vytvořil obraz obsahující všechny nezbytné závislosti a aplikaci samotnou.

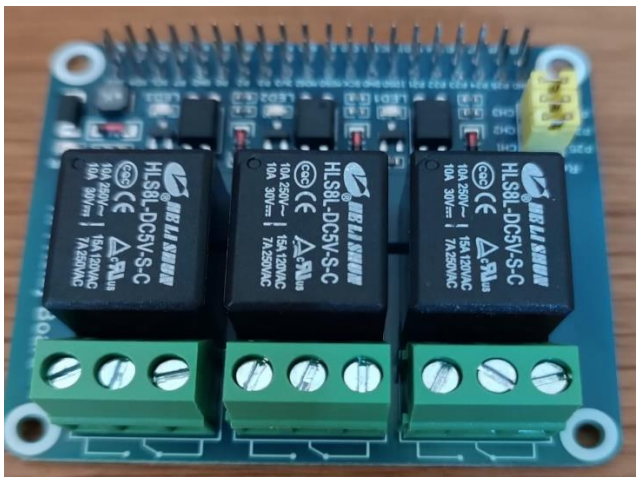
```
docker build -f /home/user/parkingJCU/serverService/DockerFile -t
parking-jcu /home/user/parkingJCU/
```

Po úspěšném sestavení Docker obrazu jsem kontejner spustil pomocí spouštěcího skriptu, který zajistil správné nastavení prostředí a spuštění potřebných služeb. Níže je ukázka části spouštějící Docker kontejner s konfigurací pro načtení environmentálních proměnných, nastavení portů, přístupu k USB zařízením nebo GPIO paměti na hostitelském Raspberry Pi a nastavením lokálního času pro správné logování vytvořeného systému.

```
docker run --env-file /home/user/parkingJCU/serverService/.env.server
\
            -p 8765:8765 -p 8000:8000 \
            --device /dev/bus/usb:/dev/bus/usb \
            --device /dev/mem:/dev/mem \
            --device /dev/gpiomem:/dev/gpiomem \
            -v /etc/localtime:/etc/localtime:ro \
            -v /etc/timezone:/etc/timezone:ro \
            -e TZ=Europe/Prague \
            --name parking-jcu \
            parking-jcu &
```

11.7 Relé deska

Z důvodu potřeby umožnění zvedání závory brány bylo nezbytné zakoupit reléovou desku, která slouží k propojení Raspberry Pi s elektromotorem závory a tím umožnit její otevírání. Otevírání závory je realizováno vysláním krátkého impulsu, který sepne elektrický obvod. Následná aktivace elektromotoru způsobí zvednutí závory. Vzhledem k vyššímu napětí použitému v bráně jsem zvolil reléovou desku, která obsahuje tři relé schopná pracovat s maximálním napětím do 30 V.



Obrázek 55: Raspberry Pi expansion board power relay

Pro zvedání závory jsem využil první relé na této desce, označené jako „Channel 1“. Toto relé je připojeno přes BCM[11] pin 26. Každé relé na desce má tři kontakty. Do prvních dvou kontaktů jsou přivedeny kabely od brány, přičemž jeden kabel vede od zdroje napětí a druhý k elektromotoru relé, které po aktivaci zvedne závoru.

[11] Pozn. Broadcom SOC channel je číslování pinu na Raspberry Pi zařízeních

12 Srovnání s existujícím Systémem Hikvision

V této závěrečné části jsem provedl porovnání přesnosti detekcí systému, který jsem vytvořil, se systémem kamery Hikvision, který je běžně využíván v komerčním prostředí. Pro tento účel jsem připravil 50 snímků, pořízených přibližně ze vzdálenosti 3-5 metrů, a následně jsem tyto snímky analyzoval jak systémem v kameře, tak i mým systémem. Kamera byla nastavena tak, aby pořizovala snímky v rozlišení 2688x1520, což zaručuje poměrně vysokou kvalitu pro detekci.

Obrázek 56: Ukázka snímku z vytvořené sady

Pro účely srovnání jsem zvolil metodu, při níž je vždy brán v potaz první výsledek při čtení registračních značek. Pro oba systémy, kameru i můj systém, byly podmínky stejné – snímky byly promítány na monitor a poté snímány kamerou. Vzhledem k výrobcem uváděné vysoké přesnosti kamery, která přesahuje 95 %, jsem nepředpokládal, že se mi podaří zaznamenat chybu v tomto kamerovém systému.

Absenci chyb při měření lze rovněž přičíst omezenému počtu testovacích dat. Při manipulaci s kamerou se mi nepodařilo zaznamenat chybné vyhodnocení kamery, avšak povedlo se mi zaznamenat vyhodnocení „unknown“, což znamená, že kamera nedokázala značku vyhodnotit. Tato situace nastala v případě, kdy došlo k náhodnému zachycení automobilu na pozadí snímku ve vzdálenosti větší než 20 metrů, kdy SPZ automobilu byla nečitelná. Tento případ však představuje výjimečnou situaci.

ID	Registrační značka	Zachycení kamerou	Můj systém
6	1SJ 9118	1SJ 9118	2S9 4325
8	4S0 7701	4S0 7701	4S0 7Q701
19	5AH 8458	5AH 8458	5IH 845B
27	5AE 5385	5AE 5385	SAE 53B5
28	5SX 8836	5SX 8836	5SX 8B36
35	6SN 8484	6SN 8484	6SN B4B4

Tabulka 1: Porovnání kamerového systému s vytvořeným systémem

Tabulka 1 prezentuje výsledky naměřené během testování. Oba systémy byly otestovány metodou zaznamenání prvního pokusu o detekci registrační značky. Oba uspěly v detekci a čtení textu všech padesáti testovacích značek. Systém kamery vyhodnotil všechny registrační značky na první pokus správně a tím dosáhl 100% úspěšnosti. Naopak mnou vytvořený systém zvládl správně přečíst pouze 44 z 50 testovacích značek, čímž dosáhl úspěšnosti 88 %. Tento výsledek přesto představuje velmi vysokou přesnost pro open-source řešení.

Je důležité zdůraznit, že se vždy jednalo o první pokus o detekci daného snímku. Ve většině případů, kdy systém chyboval, byla při dalších pokusech chyba opravena a registrační značka byla správně identifikována. Toto považuji za značný úspěch, zejména když aplikace pro vyhodnocení značek a vstup na parkoviště vyžaduje alespoň tři shody v detekcích.

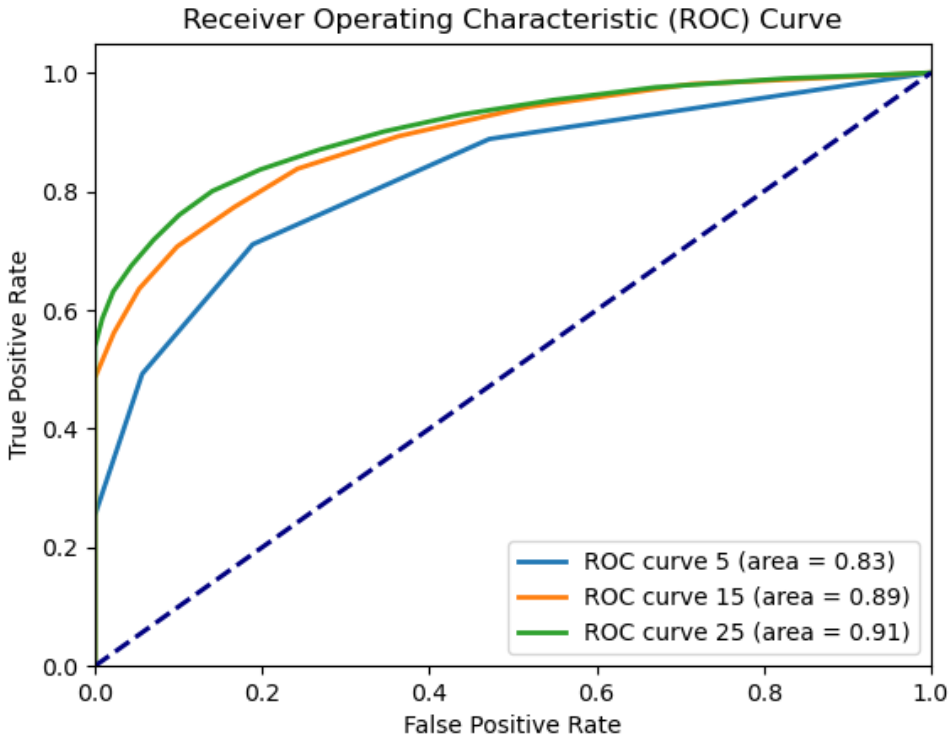
12.1 Analýza výkonnosti pomocí ROC křivky

Pro dosažení podrobnějších výsledků přesnosti vytvořeného systému jsem nechal každý snímek detekovat 25krát a zaznamenal počet správných detekcí. k výpočtu přesnosti použiji stejný vzorec:

$$Přesnost = \frac{Počet\ správně\ přečtených\ SPZ}{Celkový\ počet\ přečtených\ SPZ} \times 100 = \frac{991}{1250} \times 100 \cong 79{,}28$$

Kromě toho jsem se zaměřil na analýzu výkonu systému při různých počtech opakovaných detekcí na jednom snímku. Pro tyto účely jsem vybral 50 snímků obsahujících SPZ a na každém z nich jsem provedl detekci 25krát. Výsledky byly poté zaznamenány pro tři specifické počty opakovaných detekcí: 5, 15 a 25.

Pro vyhodnocení výkonu systému v těchto třech scénářích byl vytvořen Receiver Operating Characteristic (ROC) graf, který znázorňuje vztah mezi mírou falešně pozitivních (False Positive Rate) a pravdivě pozitivních (True Positive Rate) detekcí pro jednotlivé počty opakování detekcí.



Obrázek 57: ROC křivky výkony systému

ROC graf poskytuje detailní pohled na schopnost systému správně detekovat registrační značky vozidel při různých úrovních citlivosti.

12.1.1 Interpretace ROC grafu

Graf ROC představuje klíčový nástroj pro vyhodnocení binárních klasifikátorů, který umožnuje vizualizaci citlivosti (senzitivity) a specificity klasifikátoru. Na vertikální ose je uvedena True Positive Rate (TPR) neboli citlivost, zatímco horizontální osa zobrazuje False Positive Rate (FPR).

Výsledky pro pět detekcí

Plocha pod křivkou (AUC) znázorněná modrou barvou vyjadřuje hodnotu 0.83, což naznačuje, že systém má přiměřenou schopnost detekovat registrační značky při pěti opakováních na jeden snímek. Tato křivka je nejnižší ze všech třech znázorněných křivek, což indikuje nižší přesnost systému při menším počtu opakování.

Výsledky pro patnáct detekcí

Hodnota AUC, znázorněná oranžovou křivkou, zde dosahuje 0.89, což ukazuje na zvýšenou přesnost detekce ve srovnání s předchozím měřením pro pět detekcí. Tato křivka se posouvá blíže k levému hornímu rohu grafu, což znamená lepší citlivost a specificitu systému.

Výsledky pro dvacet pět detekcí

Poslední zkoumaný scénář znázorňuje zelená křivka dosahující AUC hodnoty 0.91, což je nejvyšší hodnota mezi všemi třemi zkoumanými scénáři. Tyto výsledky indikují nejlepší výkon systému při dvaceti pěti opakováních detekce. Tato křivka je nejblíže levému hornímu rohu grafu, a to znázorňuje vysokou přesnost a nízkou míru falešně pozitivních případů.

12.2 Zhodnocení výsledků měření

V této kapitole byla podrobně analyzována výkonnost vytvořeného systému detekce registračních značek a jeho porovnání s komerčně dostupným systémem Hikvision. Testovací sada padesáti snímků poskytla srovnatelný základ pro hodnocení přesnosti obou systémů. Snímky byly pořízeny mobilním telefonem s rozlišením 108 MP a následně prezentovány na televizi. Detekce byla provedena pomocí kamery, nejprve s použitím vytvořeného systému a poté systému kamery. Oba systémy měly stejné podmínky během testování, což zajistilo férové srovnání jejich výkonnosti.

Systém Hikvision prokázal velmi vysokou přesnost detekce, kdy správně identifikoval všechny registrační značky v prvním pokusu, což odpovídá 100% úspěšnosti. Tento výsledek potvrzuje spolehlivost a výkonnost komerčních systémů i mimo doporučené podmínky pro běh takového systému.

Nově vytvořený systém dosáhl 88% úspěšnosti při prvním pokusu, což znamená milnou detekci šesti zkoumaných registračních značek z padesáti. Tento výsledek je velmi povzbudivý, zejména vzhledem k tomu, že se jedná o open-source řešení. Navíc, při více opakovaných pokusech detekce se přesnost systému dále zlepšovala, což podtrhuje význam vícenásobné detekce pro zvýšení spolehlivosti výsledků.

Chybovost systému byla převážně způsobena znečištěním na snímcích, které vedlo k rozpoznání znaků, které tam nebyly. Často docházelo k záměně "A" za "I", pravděpodobně kvůli vzdálenosti nebo rozmazání, a také k záměně "B" za "8" nebo "5". Zajímavou situací byla záměna "0" za "6", což mohlo být způsobeno vzdáleností pořízeného snímku a nastavením zpracování obrazu. Tuto chybovost lze snížit například přeučením Teseractu na znaky na poznávacích značkách, které by ovšem zahrnovalo vytvoření obrovského datasetu obsahujícího několik stovek či tisíců snímků pro každý znak. Autoři zmíněného OCR však spíše doporučují se nejprve zaměřit na řádný image processing než se pouštět do jeho přeučení, které je velmi komplexní a náročné na výpočetní výkon.

Analýza pomocí ROC křivky poskytla hlubší vhled do výkonnosti systému při různých počtech opakování detekcí. Výsledky ukázaly, že přesnost systému se zvyšuje s počtem opakovaných detekcí, přičemž nejlepších výsledků bylo dosaženo při 25 opakováních, kde AUC dosáhla hodnoty 0.91. To indikuje vysokou schopnost systému správně identifikovat registrační značky i při vyšší citlivosti.

13 Závěr

Tato diplomová práce se zabývá vývojem a návrhem implementace systému pro rozpoznávání registračních značek určeného k řízení a správě parkovišť na Jihočeské univerzitě, kde je třeba řešit problémy s neoprávněnými vstupy.

Cílem práce bylo vytvořit systém pro detekci registračních značek automobilů a navázat na projekt týmu z předmětu IS projekt, jehož cílem bylo vyvinout informační systém pro správu univerzitních parkovišť.

V praktické části jsem zmapoval situaci kolem brány vybraného parkoviště a navrhl možnosti fyzického nasazení. Dále jsem představil vybrané hardwarové řešení a popsal testovací nasazení. Poté jsem se věnoval přípravě datasetu a jeho rozšíření pomocí augmentace, který byl následně použit pro výcvik dvou neuronových sítí. Po výcviku neuronových sítí jsem popsal upravené nastavení kamery pro získání kvalitního snímku pro následnou detekci a image processing zachycených snímků. Dále jsem testoval dvě OCR technologie, Tesseract a EasyOCR, které byly následně použity v systému, který komunikuje s projektem týmu IS projekt a zajišťuje správné uchovávání a zasílání dat, včetně webového rozhraní pro kontrolu již zaparkovaných automobilů. Následně jsem popsal detailní proces nasazení aplikace na Raspberry Pi a potřebné rozšíření pomocí Relay Board, který umožňuje ovládání závory u brány parkoviště.

Poslední kapitola je věnována porovnání vytvořeného systému s komerčním řešením od společnosti Hikvision, přičemž jejich porovnání představuje jeden z největších přínosů této práce. Výsledky drovnání ukazují, jakých úrovní přesnosti lze dosáhnout s využitím volně dostupných technologií. Mému systému se povedlo dosáhnout přesnosti vyšší než 80 % (při úspěšné detekci 44 SPZ z 50). Oproti tomu při použití komerčního řešení od Hikvision jsem dosáhl 100% přesnosti.

Výkon zvoleného OCR závisí na mnoha faktorech, jako jsou světelné podmínky, úhel záběru nebo vzdálenost kamery od snímaného objektu. Tyto problémy lze částečně eliminovat vytvořením vlastního systému pro rozpoznávání znaků nebo vytvořením rozsáhlého datasetu a pokusem o přeučení již existujících modelů, což může být u registračních značek velmi náročné. Pokud bych měl k dispozici více času, pokusil bych se vyvinout vlastní OCR a rozšířit učební datasety, což by mému systému umožnilo snáze se zdokonalit v přesnosti detekce.

Při zpětném ohlédnutí je zřejmé, že zvolené Raspberry Pi má dostatečný výkon, avšak není stoprocentně optimální. Doba detekce registračních značek se z původních několika desítek milisekund zvýšila na průměrně 1500 milisekund, což je důsledek nedostatku výpočetních vláken procesoru. Tento nedostatek by mohl být vyřešen použitím nejnovější verze Raspberry Pi 5, která však v době pořizování hardwaru nebyla dostupná kvůli vyprodání zásob.

Testovací nasazení u brány u budovy C Přírodovědecké fakulty přineslo kladné výsledky. Systém obstál ve všech třech typech vstupu, a to detekcí registrační značky, načtením QR kódu i vstupem přes zmačknutí tlačítka na displeji. Provedené testovací scénáře přinesly výsledek v delší detekční době registrační značky, než se očekávalo, a to v průměru 20 sekund, což jak bylo zmíněno výše by mohlo být vyřešeno novějším type hardwaru či snížením počtu nutných shod detekcí.

14 Summary

This thesis deals with the development and design of the implementation of a license plate recognition system for the management and administration of parking lots at the University of South Bohemia, where the issue of unauthorized entries needs to be solved.

The objective of this work was to develop a system for the detection of car registration plates and to build upon the team's project from the IS Project course, which aimed to develop an information system for the management of university car parks.

In the theoretical section of the thesis, the hardware options for license plate recognition were presented, with an emphasis on the most commonly used platforms, including Raspberry Pi, NVIDIA Jetson and Asus Tinker Board. In addition, the available options for camera solutions were presented, which are instrumental in obtaining quality input for object detection. The theoretical part also discusses image processing methods employing neural networks and optical character recognition.

In the practical part, I mapped the situation around the gate of the selected university car park and proposed options for physical deployment. Furthermore, I presented the selected hardware solution and described the test deployment. Subsequently, I prepared and augmented the dataset, and then used it to train two neural networks. Following the training of the neural networks, I described the modified camera setup to obtain a high-quality image for subsequent detection and image processing of the captured images. Moreover, I tested two OCR technologies, Tesseract and EasyOCR, which were then used in a system that communicates with the IS Project team’s work to ensure the proper data storage and transmission of data, including a web interface for checking already parked cars. Following that, I described the detailed process of deploying the application on a Raspberry Pi and the necessary extensions using a Relay Board to control the car park gate barrier.

The final section is dedicated to a comparative analysis of the developed system with a commercial solution from Hikvision. This comparison represents a significant contribution to this thesis. The results of this comparison demonstrate the levels of accuracy that can be achieved through the use of freely available technologies. Using a commercial solution developed by Hikvision, the model reached almost 100% accuracy.

In contrast, my system managed to achieve an accuracy of more than 80%, while it successfully detected 44 license plates out of 50).

The performance of the chosen OCR depends on many factors such as lighting conditions, the angle of view, and the distance of the camera from the subject. These issues can be partially eliminated by creating a custom character recognition system or by creating a large dataset and attempting to relearn existing models, which can be particularly challenging in the context of license plate recognition systems. Given additional time, I would develop my own OCR system and extend the training dataset, thereby facilitating more pronounced improvements in detection accuracy.

In retrospect, it is evident that the selected Raspberry Pi has sufficient performance, though not necessarily the optimal configuration. The license plate detection time has increased from the original few tens of milliseconds to an average of 1,500 milliseconds, which is due to the lack of computational threads in the processor. This deficiency could have been resolved by using the latest version of the Raspberry Pi 5; however, this was not available at the time of hardware acquisition, due to unavailability.

A preliminary trial of the system at the gate by building C of the Faculty of Science has produced favourable outcomes. The system demonstrated successful performance in all three entry types, namely entering via detecting the license plate, a QR code, and entering via button press on the display. The test scenarios performed resulted in a longer-than-expected license plate detection time of 20 seconds on average. However, this issue could be addressed by implementing a more advanced form of hardware or by reducing the number of detection hits required (from the current 3).

15 Seznam literatury

[1] ROB BASTIAANSEN. What is a Raspberry Pi used for? | TechTarget. *IT Operations* [online]. 17. říjen 2022 [vid. 2024-02-17]. Dostupné z: https://www.techtarget.com/searchitoperations/tip/What-is-a-Raspberry-Pi-used-for.

[2] ADNANAQEEL. *Úvod do Raspberry Pi 3 B+ – inženýrské projekty* [online]. 24. červenec 2018 [vid. 2024-02-18]. Dostupné z: https://www.theengineeringprojects.com/2018/07/introduction-to-raspberry-pi-3-b-plus.html.

[3] RPISHOP.CZ. *RPishop.cz* [online]. 27. červen 2023 [vid. 2024-02-17]. Dostupné z: https://rpishop.cz/coral/6025-recyberry-tinker-board-2s.html.

[4] DAVID SELA. Raspberry Pi Compute Module vs Raspberry Pi 4 - JFrog Connect. *JFrog* [online]. 15. srpen 2021 [vid. 2024-02-17]. Dostupné z: https://jfrog.com/connect/post/raspberry-pi-compute-module-vs-raspberry-pi-4/.

[5] ROB ZWETSLOOT. Raspberry Pi 4 specs and benchmarks. *The MagPi magazine* [online]. 24. červen 2019 [vid. 2024-02-17]. Dostupné z: https://magpi.raspberrypi.com/articles/raspberry-pi-4-specs-benchmarks.

[6] RASPBERRY PI. *Raspberry Pi Documentation - Raspberry Pi 5* [online]. 2024 [vid. 2024-02-17]. Dostupné z: https://www.raspberrypi.com/documentation/computers/raspberry-pi-5.html.

[7] VISHNU, Arjun. Raspberry Pi 4 vs. Raspberry Pi 5: 14 Key Differences. *MUO* [online]. 8. říjen 2023 [vid. 2024-02-16]. Dostupné z: https://www.makeuseof.com/raspberry-pi-4-vs-raspberry-pi-5-key-differences/.

[8] BARTOSZAK, Rafał. Nvidia Jetson - recenze a možnosti. *Botland* [online]. 23. srpen 2023 [vid. 2024-02-19]. Dostupné z: https://botland.cz/blog/nvidia-jetson-recenze-a-moznosti/.

[9] NVIDIA CORPORATION. Jetson Nano Developer Kit. *NVIDIA Developer* [online]. 2024 [vid. 2024-02-19]. Dostupné z: https://developer.nvidia.com/embedded/jetson-nano-developer-kit.

[10] BOTLAND. Nvidia Jetson Nano B01 - ARM Cortex A57 4x. *BOTLAND* [online]. 2024 [vid. 2024-02-19]. Dostupné z: https://botland.cz/moduly-nvidia/16536-nvidia-jetson-nano-b01-arm-cortex-a57-4x-143-ghz-nvidia-maxwell-4-gb-ram-5903351241519.html.

[11] BOTLAND. Nvidia Jetson Orin Nano Developer Kit – ARM Cortex A78AE 6x 1,5 GHz, Nvidia Ampere + 8 GB RAM. *BOTLAND* [online]. 2024 [vid. 2024-02-19]. Dostupné z: https://botland.cz/moduly-nvidia/22877-nvidia-jetson-orin-nano-developer-kit-arm-cortex-a78ae-6x-15-ghz-nvidia-ampere-8-gb-ram.html.

[12] NVIDIA CORPORATION. NVIDIA Jetson Orin. *NVIDIA* [online]. 2024 [vid. 2024-02-19]. Dostupné z: https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-orin/.

[13] ASUSTEK COMPUTER INC. Tinker Board | AIoT & Industrial Solutions | ASUS Global. *ASUS Global* [online]. n.d. [vid. 2024-02-21]. Dostupné z: https://www.asus.com/networking-iot-servers/aiot-industrial-solutions/tinker-series/tinker-board/.

[14] ASUS. *Tinker Board* [online]. březen 2021 [vid. 2024-02-18]. Dostupné z: https://tinker-board.asus.com/documentation/tb2.html#user.

[15] ASUSTEK COMPUTER INC. *Tinker Board* [online]. srpen 2023 [vid. 2024-02-20]. Dostupné z: https://tinker-board.asus.com/documentation/tb3n.html#user.

[16] ASUSTEK COMPUTER INC. *Tinker Board 3N* [online]. n.d. [vid. 2024-02-29]. Dostupné z: https://tinker-board.asus.com/series/tinker-board-3N.html.

[17] RASPBERRY PI. *Raspberry Pi Documentation - Camera* [online]. n.d. [vid. 2024-02-22]. Dostupné z: https://www.raspberrypi.com/documentation/accessories/camera.html.

[18] CARLO DI LEO. What are IP Cameras and How Do They Work? *Spotter Security* [online]. 24. červenec 2023 [vid. 2024-02-24]. Dostupné z: https://www.spottersecurity.com/blog/what-are-ip-cameras/.

[19] SECURIAPRO.CZ. Co je IP kamera. *securiapro.cz* [online]. 24. září 2020 [vid. 2024-02-24]. Dostupné z: https://www.securiapro.cz/clanek/co-je-ip-kamera/.

[20] HANGZHOU HIKVISION DIGITAL TECHNOLOGY CO. DS-2CD2346G2-ISU/SL. *Hikvision* [online]. 2024 [vid. 2024-02-24]. Dostupné z: http://www.hikvision.com/cz/products/IP-Products/Network-Cameras/Pro-Series-EasyIP-/ds-2cd2346g2-isu-sl/.

[21] HEUREKA.CZ. *Hikvision DS-2CD2346G2-ISU/SL(2.8mm) od 5 288 Kč - Heureka.cz* [online]. n.d. [vid. 2024-02-24]. Dostupné z: https://ip-kamery.heureka.cz/hikvision-ds-2cd2346g2-isu-sl-2_8mm/.

[22] DAHUA TECHNOLOGY. IPC-HFW5442E-ZE - Dahua International. *Dahua Technology* [online]. 2023 [vid. 2024-02-24]. Dostupné z: https://www.dahuasecurity.com/products/All-Products/Network-Cameras/WizMind-S-Series/4MP/IPC-HFW5442E-ZE=S3.

[23] HEUREKA.CZ. *Dahua IPC-HFW5442E-ZE od 13 098 Kč - Heureka.cz* [online]. n.d. [vid. 2024-02-24]. Dostupné z: https://ip-kamery.heureka.cz/dahua-ipc-hfw5442e-ze/.

[24] SURVISION GROUP. What are ANPR cameras? What are they used for? *What are ANPR cameras? What are they used for?* [online]. n.d. [vid. 2024-02-24]. Dostupné z: https://survisiongroup.com/anpr-cameras.

[25] TATTILE, Redazione. ALPR System: how work, cameras and components. *Tattile: machine vision systems & AI software for ANPR, ALPR cameras* [online]. 4. srpen 2021 [vid. 2024-02-24]. Dostupné z: https://www.tattile.com/alpr-system/.

[26] HIKVISION DIGITAL TECHNOLOGY CO., LTD. iDS-2CD7A46G0/P-IZHS(Y). *Hikvision* [online]. 2023 [vid. 2024-02-24]. Dostupné z: http://www.hikvision.com/cz/products/IP-Products/Network-Cameras/DeepinView-Series/ids-2cd7a46g0-p-izhs-y-/.

[27] ESHOP.VAKAP.CZ. Hikvision iDS-2CD7A46G0/P-IZHS(8-32mm)(C) + lepší cena po registraci. *eshop.vakap.cz* [online]. 2024 [vid. 2024-02-24]. Dostupné z: https://eshop.vakap.cz/cteni-spz/ids-2cd7a46g0-p-izhs-8-32mm--c-/.

[28] DAHUA TECHNOLOGY. IPC-HFW71242H-Z-X - Dahua International. *Dahua Technology* [online]. 2023 [vid. 2024-02-24]. Dostupné z: https://www.dahuasecurity.com/products/All-Products/Network-Cameras/WizMind-X-Series/12MP/IPC-HFW71242H-Z-X.

[29] DAHUA TECHNOLOGY CO. Dahua IPC-HFW71242H-Z-X 12 Mpx kompaktní IP kamera. *HIFI24* [online]. n.d. [vid. 2024-02-24]. Dostupné z: https://www.hifi24.cz/dahua-ipc-hfw71242h-z-x-12-mpx-kompaktn%C3%AD-ip-kamera.

[30] FRITZ AI. Object Detection Guide - Everything You Need to Know. *Fritz ai* [online]. 2024 [vid. 2024-06-29]. Dostupné z: https://fritz.ai/object-detection/.

[31] FERENČÍK, Jakub. Umělá inteligence – 6. díl. *studuj.digital* [online]. 9. prosinec 2020 [vid. 2024-03-07]. Dostupné z: https://studuj.digital/2020/12/09/umela-inteligence-6-dil/.

[32] DANIEL NELSON. *Co jsou to CNN (konvoluční neuronové sítě)? - Spojte se.AI* [online]. 23. srpen 2020 [vid. 2024-03-07]. Dostupné z: https://www.unite.ai/cs/co-jsou-konvolu%C4%8Dn%C3%AD-neuronov%C3%A9-s%C3%ADt%C4%9B/.

[33] BHATT, Dulari, Chirag PATEL, Hardik TALSANIA, Jigar PATEL, Rasmika VAGHELA, Sharnil PANDYA, Kirit MODI a Hemant GHAYVAT. CNN Variants for Computer Vision: History, Architecture, Application, Challenges and Future Scope. *Electronics* [online]. 2021, **10**(20), 2470. ISSN 2079-9292. Dostupné z: doi:10.3390/electronics10202470.

[34] ALEX GRAVES. Speech Recognition with Deep Recurrent Neural Networks. *DeepAI* [online]. 22. březen 2013 [vid. 2024-06-30]. Dostupné z: https://deepai.org/publication/speech-recognition-with-deep-recurrent-neural-networks.

[35] GEEKSFORGEEKS. Introduction to Recurrent Neural Network. *GeeksforGeeks* [online]. 3. říjen 2018 [vid. 2024-06-30]. Dostupné z: https://www.geeksforgeeks.org/introduction-to-recurrent-neural-network/.

[36] SCHMIDHUBER, Jürgen. Deep learning in neural networks: An overview. *Neural Networks* [online]. 2015, **61**, 85–117. ISSN 0893-6080. Dostupné z: doi:10.1016/j.neunet.2014.09.003.

[37] ABADI, Martín, Ashish AGARWAL, Paul BARHAM, Eugene BREVDO, Zhifeng CHEN, Craig CITRO, Greg S. CORRADO, Andy DAVIS, Jeffrey DEAN, Matthieu DEVIN, Sanjay GHEMAWAT, Ian GOODFELLOW, Andrew HARP, Geoffrey IRVING, Michael ISARD, Yangqing JIA, Rafal JOZEFOWICZ, Lukasz KAISER, Manjunath KUDLUR, Josh LEVENBERG, Dan MANE, Rajat MONGA, Sherry MOORE, Derek MURRAY, Chris OLAH, Mike SCHUSTER, Jonathon SHLENS, Benoit STEINER, Ilya SUTSKEVER, Kunal TALWAR, Paul TUCKER, Vincent VANHOUCKE, Vijay VASUDEVAN, Fernanda VIEGAS, Oriol VINYALS, Pete WARDEN, Martin WATTENBERG, Martin WICKE, Yuan YU a Xiaoqiang ZHENG. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems* [online]. B.m.: arXiv. 16. březen 2016 [vid. 2024-06-30]. Dostupné z: doi:10.48550/arXiv.1603.04467. arXiv:1603.04467 [cs].

[38] COLE STRYKER a DAVE BERGMANN. *What is PyTorch? | IBM* [online]. 4. říjen 2023 [vid. 2024-06-30]. Dostupné z: https://www.ibm.com/topics/pytorch.

[39] GIRSHICK, Ross, Jeff DONAHUE, Trevor DARRELL a Jitendra MALIK. *Rich feature hierarchies for accurate object detection and semantic segmentation* [online]. B.m.: arXiv. 22. říjen 2014 [vid. 2024-06-30]. Dostupné z: doi:10.48550/arXiv.1311.2524. arXiv:1311.2524 [cs].

[40] GIRSHICK, Ross B. Fast R-CNN. In: [online]. 2015 [vid. 2024-06-30]. Dostupné z: https://www.semanticscholar.org/paper/Fast-R-CNN-Girshick/7ffdbc358b63378f07311e883dddacc9faeeaf4b.

[41] WU, Minghu, Hanhui YUE, Juan WANG, Yongxi HUANG, Min LIU, Yuhan JIANG, Cong KE a Cheng ZENG. Object detection based on RGC mask R-CNN. *IET Image Processing* [online]. 2020, **14**(8), 1502–1508. ISSN 1751-9667. Dostupné z: doi:10.1049/iet-ipr.2019.0057.

[42] REDMON, Joseph, Santosh DIVVALA, Ross GIRSHICK a Ali FARHADI. *You Only Look Once: Unified, Real-Time Object Detection* [online]. B.m.: arXiv. 9. květen 2016 [vid. 2024-06-30]. Dostupné z: doi:10.48550/arXiv.1506.02640. arXiv:1506.02640 [cs].

[43] LIU, Wei, Dragomir ANGUELOV, Dumitru ERHAN, Christian SZEGEDY, Scott REED, Cheng-Yang FU a Alexander C. BERG. SSD: Single Shot MultiBox Detector. In: [online]. 2016 [vid. 2024-06-30], s. 21–37. Dostupné z: doi:10.1007/978-3-319-46448-0_2.

[44] RUDER, Sebastian. *An overview of gradient descent optimization algorithms* [online]. B.m.: arXiv. 15. červen 2017 [vid. 2024-07-01]. Dostupné z: doi:10.48550/arXiv.1609.04747. arXiv:1609.04747 [cs].

[45] POWERS, David M. W. *Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation* [online]. B.m.: arXiv. 10. říjen 2020 [vid. 2024-06-30]. Dostupné z: doi:10.48550/arXiv.2010.16061. arXiv:2010.16061 [cs, stat].

[46] SOKOLOVA, Marina a Guy LAPALME. a systematic analysis of performance measures for classification tasks. *Information Processing & Management* [online]. 2009, **45**(4), 427–437. ISSN 0306-4573. Dostupné z: doi:10.1016/j.ipm.2009.03.002.

[47] DATAGEN.TECH. Image Labeling in Computer Vision: a Practical Guide. *Datagen* [online]. n.d. [vid. 2024-03-06]. Dostupné z: https://datagen.tech/guides/image-annotation/image-labeling/.

[48] VIKTOR VDOVYCHENKO a VOLODYMYR ANDRUSHCHAK. *Označování dat v 8 krocích: Jak to děláme | Lemberg Solutions* [online]. 29. září 2023 [vid. 2024-03-06]. Dostupné z: https://lembergsolutions.com/blog/data-labeling-8-steps-how-we-do-it.

[49] TECHZIZOU. IMAGE DATASET LABELING/ANNOTATION. *Analytics Vidhya* [online]. 9. leden 2023 [vid. 2024-03-07]. Dostupné z: https://medium.com/analytics-vidhya/image-dataset-labeling-annotation-bec3390eda2d.

[50] ALVI, Farooq. Data Annotation – a Beginner's Guide. *OpenCV* [online]. 21. únor 2024 [vid. 2024-03-07]. Dostupné z: https://opencv.org/blog/data-annotation/.

[51] RUSSAKOVSKY, Olga, Jia DENG, Hao SU, Jonathan KRAUSE, Sanjeev SATHEESH, Sean MA, Zhiheng HUANG, Andrej KARPATHY, Aditya KHOSLA, Michael BERNSTEIN, Alexander C. BERG a Li FEI-FEI. *ImageNet Large Scale Visual Recognition Challenge* [online]. B.m.: arXiv. 29. leden 2015 [vid. 2024-06-29]. Dostupné z: http://arxiv.org/abs/1409.0575. arXiv:1409.0575 [cs].

[52] HE, Kaiming, Georgia GKIOXARI, Piotr DOLLÁR a Ross GIRSHICK. *Mask R-CNN* [online]. B.m.: arXiv. 24. leden 2018 [vid. 2024-06-29]. Dostupné z: doi:10.48550/arXiv.1703.06870. arXiv:1703.06870 [cs].

[53] KIRILLOV, Alexander, Eric MINTUN, Nikhila RAVI, Hanzi MAO, Chloe ROLLAND, Laura GUSTAFSON, Tete XIAO, Spencer WHITEHEAD, Alexander C. BERG, Wan-Yen LO, Piotr DOLLÁR a Ross GIRSHICK. *Segment Anything* [online]. B.m.: arXiv. 5. duben 2023 [vid. 2024-06-29]. Dostupné z: doi:10.48550/arXiv.2304.02643. arXiv:2304.02643 [cs].

[54] NORTHCUTT, Curtis G., Anish ATHALYE a Jonas MUELLER. *Pervasive Label Errors in Test Sets Destabilize Machine Learning Benchmarks* [online]. B.m.: arXiv. 7. listopad 2021 [vid. 2024-06-29]. Dostupné z: doi:10.48550/arXiv.2103.14749. arXiv:2103.14749 [cs, stat].

[55] ARTSTEIN, Ron a Massimo POESIO. Inter-Coder Agreement for Computational Linguistics. *Computational Linguistics* [online]. 2008, **34**(4), 555–596. ISSN 0891-2017. Dostupné z: doi:10.1162/coli.07-034-R2.

[56] EDOUARDWEBMANAGE. Getting started with LabelBox (A Comprehensive Guide). *SmartOne* [online]. 26. říjen 2023 [vid. 2024-03-07]. Dostupné z: https://smartone.ai/blog/getting-started-with-labelbox-comprehensive-guide/.

[57] LABELBOX. *Image Annotation & Labeling Tool | Labelbox* [online]. n.d. [vid. 2024-03-07]. Dostupné z: https://labelbox.com/product/annotate/image/.

[58] BOESCH, Gaudenz. LabelImg for Image Annotation. *viso.ai* [online]. 11. únor 2022 [vid. 2024-03-07]. Dostupné z: https://viso.ai/computer-vision/labelimg-for-image-annotation/.

[59] BOESCH, Gaudenz. CVAT: Computer Vision Annotation Tool - 2024 Guide. *viso.ai* [online]. 20. prosinec 2023 [vid. 2024-03-07]. Dostupné z: https://viso.ai/computer-vision/cvat-computer-vision-annotation-tool/.

[60] SIMPLILEARN. What is OCR (Optical Character Recognition): How it works & Application | Simplilearn. *Simplilearn.com* [online]. 16. listopad 2021 [vid. 2024-02-25]. Dostupné z: https://www.simplilearn.com/what-is-ocr-optical-character-recognition-article.

[61] EDUCATION, IBM Cloud. What Is Optical Character Recognition (OCR)? *IBM Blog* [online]. 5. leden 2022 [vid. 2024-02-25]. Dostupné z: https://www.ibm.com/blog/optical-character-recognition/www.ibm.com/blog/optical-character-recognition.

[62] NARENDRA SAHU a MANOJ SONKUSARE. a *Study on Optical Character Recognition.* [online]. B.m.: The International Journal of Computational Science, Information Technology and Control Engineering. leden 2017. Dostupné z: https://airccse.com/ijcsitce/papers/4117ijcsitce01.pdf.

[63] TESSERACT. Tesseract User Manual. *tessdoc* [online]. n.d. [vid. 2024-02-26]. Dostupné z: https://tesseract-ocr.github.io/tessdoc/.

[64] DOAN, Duy. Tesseract OCR: What Is It and Why Would You Choose It? *Klippa* [online]. 9. leden 2024 [vid. 2024-02-29]. Dostupné z: https://www.klippa.com/en/blog/information/tesseract-ocr/.

[65] FILIP ZELIC a ANUJ SABLE. How to OCR with Tesseract in Python with Pytesseract and OpenCV? *Nanonets Intelligent Automation, and Business Process AI Blog* [online]. 27. únor 2023 [vid. 2024-02-29]. Dostupné z: https://nanonets.com/blog/ocr-with-tesseract/.

[66] MAHAJAN, Aditya. EasyOCR: a Comprehensive Guide. *Medium* [online]. 29. říjen 2023 [vid. 2024-03-05]. Dostupné z: https://medium.com/@adityamahajan.work/easyocr-a-comprehensive-guide-5ff1cb850168.

[67] KHAN, Muhammad Rizwan. Residual Networks (ResNets). *Medium* [online]. 3. listopad 2018 [vid. 2024-03-05]. Dostupné z: https://towardsdatascience.com/residual-networks-resnets-cb474c7c834a.

[68] HE, Kaiming, Xiangyu ZHANG, Shaoqing REN a Jian SUN. *Deep Residual Learning for Image Recognition* [online]. B.m.: arXiv. 10. prosinec 2015 [vid. 2024-03-05]. Dostupné z: http://arxiv.org/abs/1512.03385. arXiv:1512.03385 [cs].

[69] CHENG, Jianpeng, Li DONG a Mirella LAPATA. Long Short-Term Memory-Networks for Machine Reading. In: Jian SU, Kevin DUH a Xavier CARRERAS, ed. *EMNLP 2016*: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing* [online]. Austin, Texas: Association for Computational Linguistics, 2016, s. 551–561 [vid. 2024-03-06]. Dostupné z: doi:10.18653/v1/D16-1053.

[70] HANNUN, Awni. Sequence Modeling with CTC. *Distill* [online]. 2017, **2**(11), e8. ISSN 2476-0757. Dostupné z: doi:10.23915/distill.00008.

[71] UFUK DAG. *What is Paddle OCR?* [online]. n.d. [vid. 2024-03-13]. Dostupné z: https://www.plugger.ai/blog/what-is-paddle-ocr.

[72] SHREYANSH JAIN. How does Paddle-OCR reads image data? *Auriga IT* [online]. 22. srpen 2022 [vid. 2024-03-14]. Dostupné z: https://aurigait.com/blog/how-does-paddle-ocr-reads-image-data/.

[73] DHANUSHKUMAR. PADDLEOCR. *Medium* [online]. 15. září 2023 [vid. 2024-03-14]. Dostupné z: https://medium.com/@danushidk507/paddleocr-439a8d92fb1a.

[74] LARXEL. *Car License Plate Detection* [online]. n.d. [vid. 2024-03-20]. Dostupné z: https://www.kaggle.com/datasets/andrewmvd/car-plate-detection.

[75] ASLAN AHMEDOV. *Automatic Number Plate Recognition* [online]. [vid. 2024-03-20]. Dostupné z: https://www.kaggle.com/datasets/aslanahmedov/number-plate-detection.

[76] MOCHOYE. License Plate Detector - v2 2023-08-31 10:01am. *Roboflow* [online]. 31. srpen 2023 [vid. 2024-03-20]. Dostupné z: https://universe.roboflow.com/mochoye/license-plate-detector-ogxxg.

[77] ROBOFLOW. License Plates - v5 Augmented License Plates. *Roboflow* [online]. 2022 [vid. 2024-03-20]. Dostupné z: https://universe.roboflow.com/samrat-sahoo/license-plates-f8vsn.

16 Seznam obrázků

17 Seznam tabulek

18 Seznam příloh

A. DVD
B. Ukázka requirements.txt souboru
C. Ukázka Dockerfile
D. Ukázka Nginx konfigurace

A Příloha

DVD – na přiložením DVD disku se nachází:

- Diplomová práce ve formátu PDF
- Kompletní projekt
- Video záznam testovacího nasazení
- Dataset pro porovnání systémů
- Záznamy z vyhodnocení porovnání systémů

B Ukázka requirements souboru

```
websockets==12.0 # this is actual version
aiohttp==3.8.5
# Enviromental vars
python-dotenv==1.0.1 # for environmental variables
imutils==0.5.4
Pillow==10.0.0 # 10.2.0 possible
pytesseract==0.3.10
numpy==1.26.2
matplotlib==3.7.2 # 3.8.3 possible
ultralytics==8.0.231 # 8.1.20 possible
dill==0.3.8 # for ultralytics
scikit-image==0.22.0
# Database
pymssql==2.2.8
# OPENCV
opencv_python==4.8.0.76
opencv_python_headless==4.8.0.76
# QR SCANNER
pyusb==1.2.1
evdev==1.7.1
# Cryptography
cryptography==36.0.0
# RPi.GPIO
RPi.GPIO==0.7.1
# Specifying torch and torchvision versions
torch==1.8.0
torchvision==0.9.1
```

C Ukázka Dockerfile

```
FROM arm64v8/python:3.9
# Setup working repository in container /app
WORKDIR /home/user/parkingJCU/
# Copy source code to the repository ./app
COPY . /home/user/parkingJCU/
# Installing dependencies (make sure all necessary packages are
installed)
RUN apt-get update && apt-get install -y \
    libglib2.0-0 \
    libsm6 \
    libxext6 \
    libxrender-dev \
    freetds-dev \
    libgl1-mesa-glx \
    libusb-1.0-0-dev \
    usbutils \
    nginx \
    tesseract-ocr \
    libleptonica-dev \
    libtesseract-dev \
    tesseract-ocr-eng \
    wget \
    python3-rpi.gpio \
    && apt-get clean

# Download and install the Tesseract language pack
RUN wget
https://github.com/openalpr/openalpr/raw/736ab0e608cf9b20d92f36a
873bb1152240daa98/runtime_data/ocr/tessdata/leu.traineddata -P
/usr/share/tesseract-ocr/5/tessdata/

# Setting environment variables to limit the number of threads
ENV OPENBLAS_NUM_THREADS=1
ENV GOTO_NUM_THREADS=2
ENV OMP_NUM_THREADS=2
```

```
# Set TESSDATA_PREFIX environment variable
ENV TESSDATA_PREFIX=/usr/share/tesseract-ocr/5/tessdata/

# Installing Python dependencies (make sure that
requirements.txt file exist)
COPY serverService/requirements.txt
/home/user/parkingJCU/serverService/requirements.txt
RUN pip install --no-cache-dir -r
/home/user/parkingJCU/serverService/requirements.txt

# Copy udev rules into the container
COPY serverService/99-usb.rules /etc/udev/rules.d/99-usb.rules
# Copy nginx configuration
COPY serverService/nginx.conf /etc/nginx/nginx.conf
# Expose necessary ports
    # Websocket server
EXPOSE 8765
    # UI display
EXPOSE 8000
EXPOSE 80
# Start the server and display the UI after the container starts
CMD ["sh", "-c", "service nginx start && python
serverService/start_server.py"]
```

D Ukázka Nginx konfigurace

```
worker_processes 1;


events {
    worker_connections 1024;
}


http {
    include mime.types;
    default_type application/octet-stream;
    sendfile on;
    keepalive_timeout 65;
    server {
        listen 80;
        location / {
            root /home/user/parkingJCU/views/html;
            index index.html;
        }
        location /css {
            root /home/user/parkingJCU/views;
        }
        location /js {
            root /home/user/parkingJCU/views;
        }
        location /ws {
            proxy_pass http://localhost:8765;
            proxy_http_version 1.1;
            proxy_set_header Upgrade $http_upgrade;
            proxy_set_header Connection "Upgrade";
            proxy_set_header Host $host;
        }
        location /api {
            proxy_pass http://localhost:8000;
        }
    }
}
```