Západočeská univerzita v Plzni
Fakulta elektrotechnická

Bakalářská práce

Systém automatické závory s rozpoznáváním registračních značek

Radek Klesa

květen 2021

ZÁPADOČESKÁ UNIVERZITA V PLZNI

Fakulta elektrotechnická

Akademický rok: 2020/2021

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Radek KLESA**
Osobní číslo:	**E18B0125P**
Studijní program:	**B2644 Aplikovaná elektrotechnika**
Studijní obor:	**Aplikovaná elektrotechnika**
Téma práce:	**Systém automatické závory s rozpoznáváním registračních značek vozidel**
Zadávající katedra:	**Katedra výkonové elektroniky a strojů**

Zásady pro vypracování

1. Prostudujte principy a algoritmy neuronových sítí a hlubokého učení se zaměřením na zpracování obrazu.
2. Prostudujte principy zpracování obrazu a vyberte vhodné knihovny.
3. Navrhněte systém určený k rozpoznávání registračních značek vozidel v reálném čase, založený na neuronových sítích vhodného typu, a s možností běhu na zabudovaném zařízení.
4. Proveďte vhodný výběr tréninkových dat a následné natrénování neuronové sítě.
5. Implementujte řešení na zvolené hardwarové platformě s podporou neuronových sítí a kamery.

Rozsah bakalářské práce: **30 – 40 stran**
Rozsah grafických prací: **podle doporučení vedoucího**
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. Chollet, F. Deep learning v jazyku Python: knihovny Keras, Tensorflow. Přeložil Rudolf PECINOVSKÝ. Praha: Grada Publishing, 2019. Knihovna programátora (Grada). ISBN 978-80-247-3100-1.
2. Cuesta, H. Analýza dat v praxi. Přeložil Jiří HUF. Brno: Computer Press, 2015. ISBN 978-80-2514361-2.
3. Dendaluce, M., Valera, J., Gómez, V., Irigoyen, E., Larzabal, E. (2014). Microcontroller Implementation of a Multi Objective Genetic Algorithm for Real-Time Intelligent Control. 10.1007/978-3-319-01854-6_8.
4. Understanding and Visualizing Neural Networks in Python. Analytics Vidhya – Learn Machine learning, artificial intelligence, business analytics, data science, big data, data visualizations tools and techniques. Analytics Vidhya [online]. Copyright 2013 [cit. 30.04.2020]. Dostupné z: https://www.analyticsvidhya.com/blog/2019/05/understanding-visualizing-neural-networks/.
5. Kim, T. S., Bae, J., Sunwoo, M. H. Fast Convolution Algorithm for Convolutional Neural Networks, 2019 IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS), Hsinchu, Taiwan, 2019, pp. 258-261.

Vedoucí bakalářské práce: **Ing. Petr Kropík, Ph.D.**
Katedra elektrotechniky a počítačového modelování

Datum zadání bakalářské práce: **9. října 2020**
Termín odevzdání bakalářské práce: **27. května 2021**

L.S.

Prof. Ing. Zdeněk Peroutka, Ph.D.
děkan

Prof. Ing. Václav Kůs, CSc.
vedoucí katedry

ANOTACE A KLÍČOVÁ SLOVA

Tato bakalářská práce se zabývá vytvořením systému automatického rozpoznávání registračních značek vozidel. Seznamuje s umělou inteligencí a konvolučními neuronovými sítěmi s použitím pro zpracování obrazu, konkrétně právě pro čtení registračních značek. Práce se tedy zabývá detekcí registračních značek, segmentací znaků a následné čtení. To vše s následnou implementací na dostupný mikrokontrolér s kamerou pro použití v provozu. V práci jsou také obsaženy aspekty výběru a zpracováním dat, trénováním neuronové sítě a průzkumem trhu se současnými řešeními.

KLÍČOVÁ SLOVA

hluboké učení, konvoluční neuronové sítě, neuronové sítě, počítačové vidění, umělá inteligence, zpracování obrazu

ANOTATION AND KEYWORDS

This bachelor thesis deals with the creation of a system of automatic license plate recognition. The work introduces artificial intelligence and convolutional neural networks used for image processing, specifically for reading registration marks. The work therefore deals with the detection of registration marks, character segmentation and subsequent reading. All this with subsequent implementation on an available microcontroller with a camera for use in operation. The work also includes aspects of data selection and processing, neural network training and market research with current solutions.

KEYWORDS

artificial intelligence, CNN, computer vision, deep learning, image processing, neural networks

PROHLÁŠENÍ

Předkládám tímto k posouzení bakalářskou práci, zpracovanou během mého studia na Fakultě elektrotechnické Západočeské univerzity v Plzni.

Prohlašuji, že jsem tuto práci vypracoval samostatně, s použitím uvedené odborné literatury a pramenů a že veškerý software, použitý při jejím řešení a zpracování, byl využit s respektováním všech jeho licenčních podmínek.

V Plzni, dne 27.5.2021 Radek Klesa

PODĚKOVÁNÍ

Na tomto místě bych chtěl poděkovat Ing. Petru Kropíkovi, Ph.D. jako vedoucímu bakalářské práce, za odborné vedení mé práce, za jeho čas, cenné poznámky a připomínky, a také za možnost častých osobních konzultací. Dále bych tímto chtěl vyjádřit poděkování všem respondentům, za jejich přínos pro výzkumnou část této bakalářské práce. V neposlední řadě patří obrovské poděkování mé rodině a přítelkyni, bez kterých bych tuto práci nemohl dokončit.

OBSAH

SEZNAM SYMBOLŮ A ZKRATEK

ALPR Automatic license plate recognition

ANPR Automatic number plate recognition

API Application Programming Interface

CNN Convolutional neural network

EU Evropská unie

GPU Graphical Processing Unit

IoU Intersection of Union

IR Infrared

FHD Full HD (1920 x 1080, 1080p)

LPR License Plate Recognition

mAP Mean Average Precision

N/D Not defined

NS Neuronová síť

OCR Optical character recognition

ReLU Rectified linear unit

RZ Registrační značka

SDK Software Development Kit

SPZ Státní poznávací značka

TPU Tensor Processing Unit

1

ÚVOD

Hlavní motivací pro výběr této problematiky pro mě bylo získat přehled a nabýt znalosti kolem umělé inteligence. Již z názvu člověku vyskočí ta složitost a náročnost. Jako programátora mě toto odvětví již dlouhodobě lákalo, avšak začít bylo vždy těžké. Zprvu bylo překážkou nedostatek informací, poté nedostatečná znalost lineární algebry a také čas. Ale nyní v době psaní bakalářské práce jsem měl za sebou několik kurzů vysokoškolské matematiky a rozmach odvětví také přinesl mnoho publikací a článků a i čas na to vymezen jako na závěrečnou práci.

Kdejakému klukovi se od dětství líbili roboti, samořídící či jinak futuristická vozidla nebo se mu obecně líbila myšlenka umělé inteligence. Nyní jsem ve fázi, kdy bych se mohl na vývoji a rozvoji těchto věcí podílet či jinak s nimi pracovat, avšak nedisponuji zkušenostmi. Při výběru bakalářské práce se mi naskytla možnost pracovat na realizaci systému na rozpoznávání registračních značek vozidel v obraze za požití umělé inteligence, a tak jsem se chopil příležitosti a pojal to jako výzvu se vše naučit na užitečném projektu, kde je jasná vidina cíle a jako bonus mám odborné vedení a oporu ze strany univerzity.

obsah a cíle práce

Úkolem je v této práci seznámit se s odvětvím umělé inteligence zabývající se zpracováním obrazu. Vše bude směřovat k vytvoření vlastní implementace automatického rozpoznávání registračních značek vozidel, které bude sloužit k automatickému otevírání vrat či závory povoleným vozidlům. Celá implementace bude cílena na dostupné zařízení Nvidia Jetson Nano, které bude opatřeno kamerou a bude tak schopno samostatného běhu.

Během cesty k tomuto cíli bude postupně natrénován detektor RZ na datech vlastního výběru, včetně seznámení datasetů na internetu a kritérii výběru a zpracování. Dále bude vytvořena segmentace znaků a jejich postupné rozpoznávání pomocí neuronové sítě a na konec finální implementace na zařízení včetně jeho oživení, řešení objevujících se problémů a testování.

2

POŽADAVKY ZÁKAZNÍKA

Jelikož závěrem a výsledkem této práce má být rozpoznávací software a dokumentace pro zákazníka univerzity, je důležité znát zadání, požadavky a očekávání zákazníka. Proto si v této kapitole popíšeme a přiblížíme tyto požadavky. Je to i pro nás pak snazší, když víme, čeho máme dosáhnout.

ZAŘÍZENÍ ZÁKAZNÍKA

Nejprve si ještě přiblížíme, kde by měl být SW nasazen a vizi finálního produktu zákazníka.

Mělo by se jednat o dostupné zařízení pro domácnosti a podniky, které bude ulehčovat každodenní rutinní činnost a to otevírání příjezdové brány s motorickým pohonem. S něčím takovým se lze setkat běžně, kdy pomocí dálkového ovladače nebo mobilního telefonu bránu otevřete. To však může být po nějaké době otravné například v případě, že odjíždíte z firmy na výjezd několikrát denně hlavně z pohledu hledání ovladače nebo používání telefonu. Také vybavit každé vozidlo ovladačem není zrovna ekonomická varianta.

Zákazník tedy přišel s nápadem aplikovat kamery na obě strany brány pro příjezd i odjezd, které budou číst RZ vozidel a brána se bude otevírat vozidlům, které budou mít umožněný přístup v databázi. Takové řešení je pohodlné, bez nutnosti používat telefon nebo ovladač. Případné rozšíření spočívá pouze v přidání RZ do databáze a již není třeba přikupovat ovladač. Jako bonus dokáže systém vést evidenci příjezdů a odjezdů podle vozidel a ne podle uživatelů.

Podobné systémy lze znát z parkovišť například u obchodních domů, kde se to používá pro zjištění doby parkování a následného případného výpočtu ceny. Tato řešení jsou často však velmi rozsáhlá, velká a nákladná na pořízení do menšího podniku nebo domácnosti.

RYCHLOST ROZPOZNÁVÁNÍ

Klíčovým parametrem pro plynulost tohoto systému je rychlost rozpoznávání. Nelze čekat 10 a více vteřin, než rozpoznávač "řekne" RZ vozidla. Proto zákazník nastavil limit 2 sekundy. Reálně a ideálně by tento výpočet měl být do 1 s, avšak za zhoršených podmínek viditelnosti lze očekávat pomalejší výstup. Zmíněné hodnoty jsou předpokládány pro běh na mikrokontroléru.

KVALITA ROZPOZNÁVÁNÍ

Bezesporu pro funkčnost takového systému nelze opomenout jeden z hlavních parametrů a tím je kvalita rozpoznání, neboli hodnota pravděpodobnosti, s jakou si je rozpoznávací systém jist svým výsledkem. Pro zajištění určité kvality, jistoty a bezpečnosti je hodnota nastavena na 95 %. Asi lze předvídat, že takových hodnot je možné dosáhnout za optimálních podmínek, kdy bude dobré světlo a RZ nebude silně znečištěna, protože to pak může znaky zkreslovat.

PLATFORMA

Mělo by se jednat o co nejmenší zařízení, které lze snadno integrovat a bude mít dostatečný výkon pro běh takového systému. Vše tedy směřuje k výkonným mikrokontrolérům s podporou kamery. Musí také být možné připojit infračervené přisvětlení pro možnost chodu za tmy. Důraz je také kladen na kvalitu z hlediska nepřetržitého běhu a podporu ze strany výrobce z důvodu dostupnosti v následujících letech.

FORMÁTY VSTUPU A VÝSTUPU SYSTÉMU

Jelikož není potřeba detekovat plynule ve videu velkou rychlostí, tak zařízení bude pořizovat snímky, které se budou předávat ve vhodném formátu rozpoznávacímu systému. Výstupem toho by měl být textový řetězec s rozpoznanou RZ a pravděpodobnost shody výsledku. Takový výstup může v pythonské syntaxi představovat například třída „dict“, se kterou se dále dobře pracuje a data se snadno parsují. Formát takového výstupu může vypadat třeba takto: {'r':'SPZ12345','q':'97,3'}, kde klíč 'r' představuje RZ a klíč 'q' pravděpodobnost. V pokročilé fázi projektu pak takto: {'r':'SPZ12345','q':'97,3','d':'289'}, kde 'd' je vzdálenost vozidla od kamery.

UMÍSTĚNÍ

Tímto je myšleno kde a jak bude zařízení umístěno. Je to důležité z důvodu oněch nočních snímků, kdy by pořízený snímek při umístění velmi nízko byl příliš negativně ovlivněn silným světlem reflektorů automobilu. Proto je požadavek na umístění do výšky přibližně 1 až 2 metry nad zemí. Pro nás je tento parametr dobré znát, jelikož při této vzdálenosti není snímek RZ přímý, ale je pod úhlem shora, a tak RZ na snímku je lehce tvarově zkreslena a my s tím budeme počítat a můžeme zkreslení eliminovat.

CENA

Jak již bylo zmíněno, mělo by se jednat o dostupné zařízení i do domácnosti, s tím tedy silně souvisí i pořizovací cena finálního produktu pro koncového zákazníka. Samozřejmě, že by měla být ideálně co nejnižší. Po průzkumu trhu s aktuálními řešeními se došlo k závěru pořízení vlastního rozpoznávacího systému na míru. Vize zákazníka je prodávat finální produkt za cenu do 15 000 Kč pro koncového zákazníka. Ceny a porovnání aktuálních řešení nalezneme v kapitole 4.

ZABEZPEČENÍ

Tento požadavek souvisí hlavně s chodem zařízení. Celé rozpoznávání musí probíhat offline, tedy bez připojení na internet nebo používat výpočetní výkon cloudových platforem. Jediný způsob komunikace bude přes GSM bránu nebo pomocí lokálně vytvořené a skryté WiFi sítě na hlavní řídící jednotce.

VÝPOČET VZDÁLENOSTI

Zákazník má v požadavku krom rozpoznávání RZ, také zjistit z obrazu, jak daleko se vozidlo nachází od kamery. K tomu potřebujeme znát již zmíněné umístění a bude se nám hodit znát parametry RZ, které si rozebereme v kapitole 3, a rozměry, které jsou vyobrazené na obr. 3.

Tento požadavek je součástí zadání projektu, avšak nebude řešen v této bakalářské práci.

3

REGISTRAČNÍ ZNAČKA

Registrační značky slouží k jednoznačné identifikaci vozidla, jsou jimi tedy opatřena všechna silniční motorová, přípojná, elektrická a zvláštní vozidla. Všechny tyto pojmy jsou vymezeny ve vyhlášce 343/2014 Sb. Pomocí registračních značek lze z online databází zjistit mnoho, například pojišťovnu, se kterou je sjednáno povinné ručení, tovární značku vozidla, typ automobilu a kraj, ve kterém vozidlo bylo registrováno. Jelikož je RZ vozidla jedinečný identifikátor vozidla, jak bylo zmíněno, nelze tedy narazit na dvě různá vozidla se stejnou RZ. Toho lze využít i při dalších aplikacích, zejména automatizovaných na bázi přístupnosti. To vše je pro nás motivací, proč chceme převádět obrazová data na textový výstup, se kterým tyto automatizované operace lze provádět. Příklad české registrační značky znázorňuje obr. 1.

PARAMETRY A VLASTNOSTI REGISTRAČNÍCH ZNAČEK

Abychom mohli co nejsnáze číst registrační značky pomocí počítačového vidění, určitě je pro nás polehčující okolnost, pokud budeme znát veškeré vlastnosti a parametry RZ, díky kterým dosáhneme lepších výsledků a vyhneme se chybám.

Znaková sada

Použité znaky jsou velká písmena latinské abecedy z množiny {A, B, C, D, E, F, H, I, J, K, L, M, N, P, R, S, T, U, V, X, Y, Z}. Lze si povšimnout, že nějaké znaky

Obr. 1: Příklad české registrační značky [1]

v množině chybí. Jsou to znaky, které by mohly být snadno zaměnitelné s jinými, např. "O"a "0"nebo "G"a "6". Tím je zajištěna vyšší bezpečnost u rozpoznávání a také možnost určité znaky z predikce vyřadit. Dále jsou znaky vyjádřeny pomocí arabských číslic z číselné množiny {1, 2, 3, 4, 5, 6, 7, 8, 9, 0}. Běžná registrační značka silničního motorového vozidla má celkem 7 znaků.

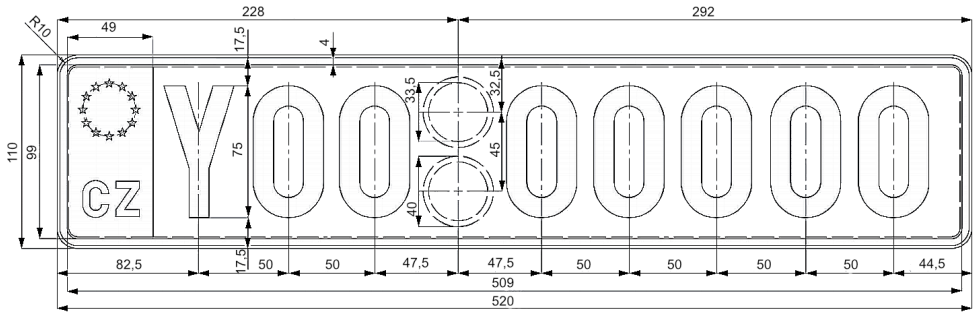
Barvy

Pro naše snažení je určitě dobré vědět, že na RZ jsou znaky černou barvou (pro většinu vozidel, výjimku tvoří veteránská vozidla, jež jsou opatřeny znaky zelené barvy) a pozadí je bílé. Dohromady nám to zajistí skvělé kontrastní vlastnosti znaků vůči pozadí a tím i lepší práci s detekcí. V levé části mimo znaky je modrý pruh se znakem EU a rozlišovací značka státu. Aby detekční systémy fungovaly i za snížené viditelnosti, je RZ opatřena reflexní vrstvou viz obr. 2. V praxi je však snadné se setkat s vozidly, jež mají RZ znečištěné (zvláště v zimních měsících), a tak se kontrastní vlastnosti snižují. S tímto problémem bude nutné se vypořádat v pokračování této práce v rámci projektu.

Obr. 2: Snímek RZ na vozidle za tmy a IR osvětlení [2].

Rozměry

Rozměry lze také brát jako pevné východisko pro naši práci. Jsou totiž neměnné a jejich hodnoty jsou známé. Úprava je zveřejněna ve vyhlášce 343/2014 Sb. Zde uvedu alespoň nejčastější, který je na většině provozovaných vozidel na našem území a to jest 520 x 110 mm. [3] Detailní pohled na RZ včetně rozpoložení a situování znaků je na obr. 3.



Obr. 3: Rozměry RZ včetně rozpoložení znaků [3].

Značky na přání

Nelze však opomenout možnost a variantu značek na přání, které je možné v České republice pořídit od ledna roku 2016. Takové značky musí splňovat požadavky výše, liší se však v počtu znaků, zde jich je 8. Rozložení takové značky je znázorněno na obr. 3. Cena za každou značku je 5 000 Kč a nesmí obsahovat hanlivé či jinak urážející texty. Vymezení pojmů upravuje rovněž vyhláška 343/2014 Sb. [3]

4

SOUČASNÁ ŘEŠENÍ

Součástí této kapitoly bude přehled a srovnání aktuálně používaných a dostupných řešení na trhu. Zaměření bude hlavně na dostupnost v České republice, jakožto z pohledu podpory rozpoznávání českých RZ. Dále pak důležitými aspekty ke srovnání jsou výsledné rozpoznání a jeho rychlost. V této kapitole pouze teoretické z poskytnutých dat výrobce či poskytovatele. A posledním srovnávacím aspektem v této části bude také cena. Závěrem tohoto poznání by měla být motivace k vytvoření vlastního detekčního řešení. Následující informace byly od společností na žádost poskytnuty prostřednictvím emailové komunikace.

OPENALPR

V současnosti se jedná o jedno z nejznámějších a nejrozšířenějších řešení, které lze sehnat. Software je také šířen pod Open-source licencí a lze ho bezplatně používat za určitých podmínek a je možné ho získat z oficiálních stránek výrobce. Dokonce i náš zákazník používá tento SW pro vývoj a testování HW. Pro komerční použití je nutné splnit copyleftové podmínky použití, které však většina firem nesplní včetně našeho zákazníka. [4] Nejlevnější tarif, který je nabízen, vyjde na 29 $ měsíčně, což je přibližně 640 Kč (přepočteno kurzem ze dne 18.3.2021) a je limitováno 2 500 rozpoznáními za měsíc. [5]

Open-source řešení je založeno na OpenCV a TesseractOCR knihovnách. OpenCV se stará o segmentaci a zpracování snímku a TesseractOCR pak jen v podstatě "převede" vysegmentované znaky z obrazu do textové formy. [6] Je dostupné formou knihovny, které se předá obrázek a výstupem je text RZ a pravděpodobnost rozpoznání, se kterými lze dále pracovat.

Řešení podporuje více jak 65 zemí včetně České republiky. Rozpoznávání probíhá do 1 000 ms a jeho kvalita je více než 80 % dle údajů z nasazení Open-source varianty na Raspberry Pi 3+ u zákazníka. [5]

PLATE RECOGNIZER

Toto řešení je asi hlavním konkurentem výše popisovaného openALPR, je to patrné i z politiky tohoto výrobce, kdy veškeré srovnání je pouze s openALPR. Oba poskytovatelé jsou si velmi podobní, avšak Plate Recognizer není šířen jako Open-source, ale nabízí možnost bezplatné verze s limitem 2 500 rozpoznání měsíčně pro nekomerční účely, pro ty je nutné zvolit tarif, který začíná s 50 000 rozpoznání měsíčně za cenu 50 $ každý měsíc (přibližně 1 100 Kč dle kurzu z 18.3.2021). Za stejnou cenu nabízí openALPR poloviční měsíční limit rozpoznání.

Podpora zde je vysoká, jelikož tento systém dokáže fungovat ve více než 90 zemích včetně České republiky. Chlubí se také výtečnou kvalitou rozpoznání za špatných světelných podmínek a méně kvalitních snímků se šumem. O rozpoznávání se starají neuronové sítě. Výsledná rychlost je deklarována na 21 ms (8-core CPU, 32GB RAM) a až 1 000 ms (Raspberry Pi 3/4). Kvalitu výstupu nelze oficiálně dohledat, ale technická podpora ujišťuje velmi vysokou kvalitou. [7]

CARMEN

Tak zní název řešení od maďarské firmy Adaptive Recognition. Firma s tradicí sahající do roku 1990 a již od svého počátku se věnují vývoji této platformy.
Společnost nabízí mnoho variant jak na cloudové bázi, tak formou enginu. Pro naše potřeby a srovnání nejvíce vyhovuje Carmen Freeflow. Za tuto variantu si společnost účtuje 1 250 € za každou licenci, ale nutno podotknout, že licence je zde brána za každé jádro procesoru, které se bude účastnit výpočtu (není nutné využít všechna jádra procesoru).

Tento engine založen na bázi neuronových sítích je podporován na procesorech od firmy ARM a je tedy možné jej provozovat na mikropočítačích založených na této architektuře. I zde je zaručena podpora českých RZ a výstupy rozpoznávání lze získat za 100-300 ms s přesností více jak 95 % v závislosti na procesoru a kvalitě obrázku. [8]

VIVOTEK

Jedná se o Tchaj-wanskou společnost, která působí na trhu od roku 2000. Je to jeden z předních výrobců, zabývající se výrobou bezpečnostních kamer a produktů týkající se kamerového zabezpečení. Krom toho se v jejich portfóliu vyskytují zařízení umožňující rozpoznávající RZ.

I toto řešení podporuje naše RZ a mnoho dalších. Doba rozpoznání je 200-300 ms a přesnost 95-98 % v závislosti na okolnostech. I tento výrobce nasadil na rozpoznávání umělou inteligenci, přesněji hluboké učení. Toto řešení je však distribuováno pouze jako zabudované v kamerovém zařízení a nelze použít na vlastní platformě.

Cenově se pak pohybuje okolo 200-500 $ za licenci v závislosti náročnosti projektu (přibližně 4 440-11 100 Kč dle kurzu z 24.3.2021). [9]

VAXTOR

Dalším hráčem na trhu se softwarem na ALPR je společnost Vaxtor. Společnost působí v tomto odvětví na trhu téměř 10 let a zabývá se pouze detekčním SW. Ten je schopen běžet na běžném počítači, mikrokontroléru, embedded kameře nebo na android zařízení.

Pro Vaxtor je podpora české RZ samozřejmostí a podporuje i dalších 150 států. Časy rozpoznávání se pohybují okolo 50-300 ms v závislosti na hostující platformě a úspěšnost dosahuje až 99 %. I zde je stavěno na nasazení moderních neuronových sítích. Zde cena startuje na 195 € za licenci pro android zařízení (přibližně 5 120 Kč dle kurzu z 24.3.2021) a 495 $ za licenci pro kameru nebo počítač (přibližně 11 000 Kč dle kurzu z 24.3.2021). Tento SW je možné vyzkoušet v trial verzi po dobu 2 týdnů pro android nebo 4 týdnů pro kameru či počítač. [10]

NEURALLABS

Další společností v segmentu nabízejících ANPR je NeuralLabs. Jedná se o společnost sídlící ve Španělsku a má více jak 20 let zkušeností s detekcí a analýzou obrazu. Jejich zaměření je na neuronové sítě a techniky deep learningu.

Jejich řešení podporuje více jak 50 států včetně české RZ, je možné jej provozovat offline na vlastní platformě, Raspberry Pi nebo podobné. Rozpoznávací rychlost je okolo 50 ms pro evropské RZ a dosahuje kvality 95-99.9 % v závislosti na kvalitě pořízeného snímku.

Toto konkrétní řešení využívá k rozpoznávání neuronové sítě a v nejnovější verzi také deep learning pro zvýšení kvality čtení, i když je to dle slov výrobce o něco pomalejší.

Cenově pak takové řešení vyjde na 1050 € (přibližně 27 500 Kč dle kurzu z 18.3.2021), pokud se jedná o samotný engine pro jedno jádro, nebo na 640 € (přibližně 17 000 Kč dle kurzu z 18.3.2021) za jednu kameru s vestavěným LPR. [11]

Ještě bych rád vychválil komunikaci ze strany NeuralLabs, jež poskytli téměř okamžitou odpověď s přívalem informací a ochotou poradit.

OSTATNÍ

Dalšími velkými hráči v oborou jsou společnosti jako Jenoptik, IntelliVision či Inex Tech, avšak ani od jednoho z nich se nepodařilo získat žádné podrobnější informace o jejich řešeních, jelikož nikdo z nich neodpovídal na technickém oddělení

jak přes komunikační protokol, tak ani na emailové zprávy. I to může mnoho vypovídat o kvalitě společnosti, protože i technická podpora a komunikace prodejce může být často dosti klíčová při výběru.

SHRNUTÍ A ZÁVĚR KAPITOLY

Závěrem lze říci, že na trhu lze sehnat mnoho kvalitních řešení nebo SW. Avšak z tohoto našeho průřezu trhem ani jedno nezapadá do našeho hlavního kritéria a tím je cena. Při vizi dostupného zařízení nelze použít ani jeden výše zmíněný systém, jelikož by se cena mnohdy více než zdvojnásobila a tím bychom nehovořili o cenově dostupném zařízení. Dále ani jeden software není schopen běhu na mikrokontrolérech, nejmenší podporované zařízení je Raspberry Pi.

Níže v tabulce 1 se lze podívat na krátké porovnání důležitých parametrů pro produkty popsané výše. Jedná se o data od poskytovatelů.

Tab. 1: Přehledová tabulka srovnávající jednotlivé platformy.

Produkt	Cena	Rychlost [ms]	Kvalita [%]	Platforma	Limit
OpenALPR	29 $/m	N/D	N/D	cloud/edge	2 500
PlateRecognizer	50 $/m	< 1000	N/D	cloud/edge	50 000
Carmen	1250 €	< 300	> 95	edge	žádný
Vivotek	200 $	< 300	> 95	kamera	žádný
Vaxtor	495 $	< 300	~ 99	edge	žádný
NeuralLabs	1050 €	< 50	> 95	edge	žádný

5

HARDWARE

Součástí zadání této práce je také výběr vhodného hardwaru, který bude schopný hostovat rozpoznávací software, který by měl vyplynout v závěru této práce.

Požadavky jsou kladeny tak, aby zvolený hardware byl co možná nejmenší pro následné zakomponování, měl by obsahovat kameru nebo alespoň zaručovat kompatibilitu s nějakou z dostupných embedded kamer, dostatečný výkon pro zpracování obrazu, podporu neuronových sítí, podporu od výrobce v podobě dokumentace a podpory knihoven a API. Na závěr by měla být také zaručena dostupnost na trhu od výrobce pro následné komerční použití.

Takto se může zdát, že hledáme ideální produkt, takový ovšem neexistuje, nejvíce by takovým potřebám vyhovoval produkt na míru stavěn těmto potřebám, avšak bohužel tuto možnost nemáme k dispozici a není to předmětem tohoto zkoumání. My se zaměříme na vývojové platformy běžně dostupné na trhu, které si v následujících subkapitolách přiblížíme a dozvíme se jaké ústupky bude třeba podniknout. Stále budeme mít při výběru na paměti požadavky popsané v kapitole 2.

Raspberry Pi 4

Z hlediska rozšíření mezi developery a nadšenci se jedná o nejpopulárnější jednodeskový počítač. Na tomto počítači běží oficiální operační systém Raspbian, jež je na linuxovém jádru, ale je možné na něm provozovat další distribuce linuxu. O výkon se stará procesor vyroben 28nm technologií a pracující pod taktem až 1,5 GHz, GPU VideoCore VI pak na taktu 500 MHz. To je dostatek výkonu na 4K video při 60 snímcích za sekundu. Nenabízí se však hardwarová podpora neuronových sítí. To je však možné řešit neural stick zařízeními. [12]

Nabízí skvěle programové vybavení, včetně podpory programovacího jazyka Python a mnoha knihoven dostupné pro klasické počítače a z toho vyplývající výhody. Tyto benefity by nám usnadnily mnoho práce u přenosu platformy.

Dostupno je také mnoho periferií, nás však zajímají kamery a ty jsou k dostání jak oficiální, tak mnoho neoficiálních.

Jako nevýhodu lze pro naše potřeby brát to, že zařízení je poměrně velké, nemá hardwarovou podporu NS a také to, že celý systém běží na paměťové kartě, což může být z hlediska spolehlivosti pro nepřetržitý provoz slabinou.

NVIDIA JETSON NANO

Asi nejlépe hardwarově i softwarově vybavený developer kit na trhu za dostupnou cenu je právě Jetson Nano od společnosti NVIDIA. Ta je obecně známa svým velkým investováním do vývoje umělé inteligence, proto i její grafické karty jsou ideálním nástrojem pro trénování neuronových sítí. Na trh se dostal i tento developer kit, který má sloužit jako základna pro projekty na bázi AI.

Je zde hardwarová podpora vývoje NN v podobě 128-jádrové GPU, softwarově pak v podobě CUDA, cuDNN a TensorRT, jež jsou součástí JetPack balíčku. Dále skvělá podpora ze strany výrobce, široká komunita a nespočet tutoriálů, věnující se implementaci právě na Jetson Nano. Práce je výrazně ulehčena vývojovým prostředím od NVIDIA a výše zmíněného JetPack SDK, kdy je možné začít na předpřipravených projektech téměř bez předchozích znalostí či seznámení s NS. [13]

Pro naše potřeby se jedná o velmi dobrého kandidáta, avšak pro nasazení v zákazníkově produktu je příliš velké a drahé. Navíc v případě použití by bylo potřeba připojit kamery na vzdálenost 10 metrů a více, což je v případě linky CSI nerealizovatelné a použití USB webkamery nepraktické. Avšak pro potřeby této práce bude použito pro testování a implementaci softwaru. Konkrétně nejmenší verze s 2 GB RAM a webkamerou Logitech C270 s rozlišením 720p.

MAIX BIT I

Vývojová platforma od společnosti Seeed studio. Poháněna je dvou-jádrovým 64bitovým procesorem K210 Kendryte, jež je postaven na architektuře RISC-V. Výhodou použití tohoto procesoru je, že obsahuje hardwarový akcelerátor konvolučních neuronových sítí KPU. Tento akcelerátor nemá určen limit počtu vrstev, lze konfigurovat parametry jednotlivých vrstev nezávisle, jsou podporována 1x1 a 3x3 konvoluční jádra (kernels). Též jsou podporovány jakékoliv aktivační funkce vrstev a velikost sítě je omezena na 5,5 - 5,9 MB pro běh v reálném čase a bez limitu velikosti (limit flash paměti) pro ostatní sítě. Flash paměť na desce má základní velikost 16 MB, ale je zde slot pro micro SD karty s podporou až do velikosti 128 GB.

Zařízení podporuje programovací jazyk microPython, jež je derivací jazyka Python pro mikrokontroléry, který je navíc obohacen o knihovní funkce od výrobce pro práci s obrazem a neuronovými sítěmi pod nadstavbou MaixPy. Z užitečných knihoven, které nejsou úplně standardní, bych vyjmenoval například ulab, jež má podobné vlastnosti jako NumPy nebo OpenMV, která v mikrokontrolérech nahra-

zuje OpenCV a stará se o zpracování obrazu. Pro neuronové sítě (konvoluční) je zde k dispozici knihovna KPU. Ze softwarové podpory hlubokého učení lze zmínit tiny-YOLO, mobile-net nebo TensorFlow Lite (verze pro mikrokontroléry). [14]

Za cenu okolo 20 $ se jedná asi o nejlepší zařízení na trhu s poměrem cena/výkon. Navíc je rovnou součástí dodávky 2Mpx kamera, ale K210 je schopen pracovat jen s VGA rozlišení (640 x 480 pixelů, poměr 4:3), avšak to je dostatečné rozlišení pro naše potřeby. [15]

MAIX BIT II

Při sepisování této práce se u ověřování informací vyskytl na trhu ještě výkonnější model a nástupce Maix Bit I a tím je právě Maix Bit II. V tuto chvíli k němu není poskytnuto příliš informací, avšak z těch dostupných lze říci, že obsahuje procesor V831 pracující na taktu až 1 GHz, což je až dvojnásobek oproti K210. Dále je zde výrazný narůst paměti z 8 MB na 64 MB, plná podpora neuronových sítí v podobě hardwarového TPU. Zařízení běží na MAIX-Linux, jehož základem je OpenWRT známý ze síťových routerů nebo jiných zařízení. To přináší i plnou podporu jazyka Python a tím i podporu NumPy a PIL knihoven nebo také adaptaci Jupyter Notebooku. Dále lze zmínit on-board Wi-Fi, slot pro paměťovou kartu nebo kameru součástí dodávky, známou z předchozí verze. [16]

Toto zařízení bylo představeno začátkem roku 2021 a nyní je k dispozici pouze u výrobce za cenu 28,80 $. [17] Bohužel však nebude dodáno zavčasu pro potřeby této práce, ale přesto jsem cítil nutnost se o něm také zmínit, jelikož odpovídá nejvíce předpokladům hledaného zařízení a je tedy velmi pravděpodobné, že pokračování projektu bude právě na něm.

ZHODNOCENÍ

Na trhu je nespočet vývojových kitů různých modifikací a cenových kategorií, avšak jakmile zahrneme do kritéria výběru také podporu AI či nějaký hardwarový prvek pro NS, hned se výběr zúží na pár zařízení. Často se jedná o velká a drahá zařízení nebo zařízení s nedostatečnou dokumentací, podporou nebo nejistou podporou do budoucna. Nejlépe si v tomto směru vede NVIDIA, jež se AI usilovně zabývá, má skvělou podporu a velkou komunitu. Proto byl jejich produkt Jetson Nano zvolen pro potřeby této práce. Na velmi dobré cestě je čínské Seeed studio a pro své parametry a proporce jsem vybral právě Maix Bit I jako kandidáta pro projekt, stavějící na této práci. Veškeré práce budou voleny z ohledem na rozvoj a následnou implementaci právě na Maix Bitu I. Na závěr lze ještě zmínit existenci tzv. neural sticks, které fungují jako hardwarové akcelerátory neuronových sítí, které je možné připojit přes USB rozhraní a zvýšit tak výkon běhového zaří-

zení např. jak bylo zmíněno výše u Raspberry Pi. Z těchto zařízení lze uvést třeba Intel Neural stick nebo Google Coral.

Tab. 2: Přehledová tabulka porovnávající zmíněný hardware, pro kameru uvažovány oficiální nebo doporučené.

Platforma	Cena	Kamera	Podpora NN	Rozměry	OS
Raspberry Pi 4 (2GB)	39 $	Ne (28 $)	Ne	Střední	Raspbian
NVIDIA Jetson Nano (2GB)	39 $	Ne (28 $)	Ano (CUDA GPU)	Velké	Ubuntu
Maix Bit I	20.90 $	Ano	Ano (KPU)	malé	- (FreeRtos)
Maix Bit II	28.80 $	Ano	Ano (TPU)	malé	Maix-Linux

6

UMĚLÁ INTELIGENCE

Umělá inteligence je v posledních pár letech na velkém vzestupu. Naleznout ji lze dnes na každém kroku ať, už si to uvědomujeme či nikoli. V dnešní době je téměř každý majitel chytrého telefonu, kde se lze setkat například s hlasovými asistenty, fotoaparátem automaticky si volící scénu pro optimalizaci ideální fotografie, převodem textu na řeč a opačně, tlumočníkem nebo seskupováním fotografií podle obličeje. Dále se lze v běžném životě setkat se systémem využívající AI například u různých chatterbotů na webu nebo v e-shopech, které Vám pomohou s výběrem produktů, nebo s telefonními automaty na infolinkách, kde sdělíte svůj problém a ony Vám jej pomohou vyřešit nebo Vás přepojí, aniž byste museli vyslechnout veškeré možnosti a volit na číselníku.

V našem zkoumání nás však bude zajímat hlavně ta část zpracování obrazu, kdy ji budeme využívat k převodu textu z obrázku na textový výstup, se kterým bude počítač moci dále pracovat a vyhodnocovat.

K masovému rozvoji přispěl hlavně výkonný hardware, který se také stal dostupnější. Již není těžké dostat se ke zdroji informací nebo sehnat HW pro vývoj. Se zájmem o toto odvětví vzrostla také podpora ze strany vývojářů programovacích jazyků a knihoven, které umožňují více přiblížit a usnadnit práci. [18]

6.1 UMĚLÁ INTELIGENCE

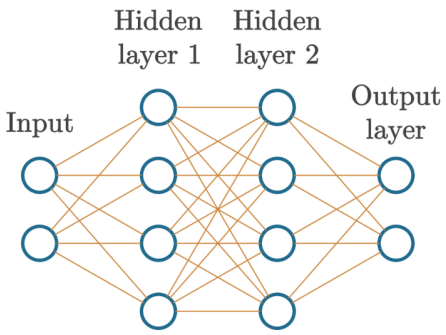
Definici umělé inteligence lze vyjádřit dle [19] takto: „*Umělá inteligence označuje systémy nebo stroje, které napodobují lidskou inteligenci k plnění úkolů a mohou se iterativně vylepšovat na základě shromážděných informací.*"“

Umělá inteligence jako taková má své podmnožiny a to stejnojmennou umělou inteligenci, strojové učení a hluboké učení [20]. My se v této práci budeme zabývat podmnožinou hlubokého učení (v literatuře spíše nepřeloženě deep learning), pouze si v určitých případech uvedeme odlišnost od strojového učení (též jako machine learning).

6.2 MACHINE LEARNING

Při klasickém programování se snažíme program naučit (naprogramovat) pomocí nějakých dat a pravidel a výstupem nám jsou odpovědi. Pokud hovoříme o strojovém učení, vstupem nám jsou data a odpovědi, výstupem jsou pak pravidla. Pomocí těchto naučených pravidel pak můžeme získávat odpovědi na nová data. Jedná se o jakousi automatizaci pravidel. Francois Chollet ve své knize uvádí definici strojového učení jako: *"...hledání užitečných reprezentací některých vstupních dat v předem definovaném prostoru možností pomocí pokynů ze zpětnovazebního signálu."* Zpětnovazební signál je zde použit pro vyhodnocování správnosti učení, kdy je měřena odchylka skutečné a predikované hodnoty. [20]

Tento typ učení je někdy označován jako mělké učení, jelikož obsahuje jednu až dvě skryté vrstvy, jak je vyobrazeno na obr. 4.



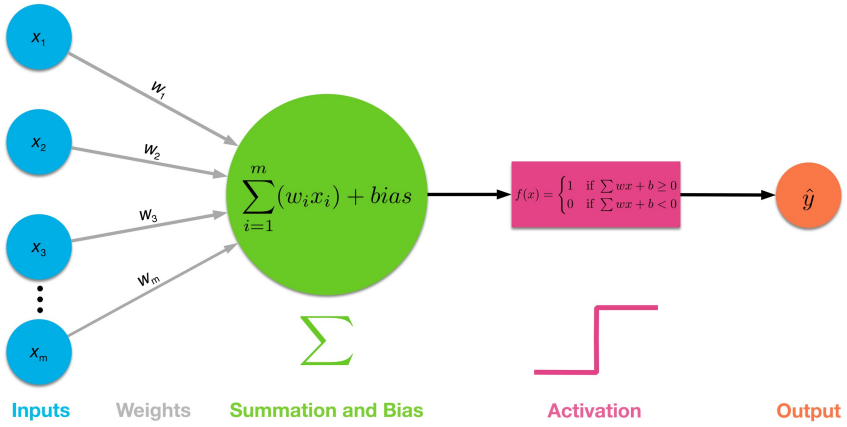
Obr. 4: Plně propojená neuronová síť [21].

6.3 DEEP LEARNING

Tato metoda je podskupinou strojového učení a jeho hlavní odlišností, jak již název napovídá, je v hloubce učení. To je realizováno počtem skrytých vrstev v modelu, kdy mělké učení u strojového učení používá jednotky těchto vrstev, zde jsou to pro velké modely i tisíce. [20] S hloubkou modelu však rostou požadavky na trénovací hardware a trénování zabere značně času. U hlubokého učení nám také odpadá nutnost manuální extrakce příznaků (jak je tomu u strojového učení), které se provedou ve vrstvách hlubokého modelu.

6.4 NEURONOVÉ SÍTĚ

Pod pojmem neuronové sítě si lze představit velké množství vzájemně propojených buněk, jak je tomu u člověka. V koncepci umělé neuronové sítě je to podobné. Než se však vrhneme na takové sítě, rád bych popsal, jak funguje jeden jediný neuron, ze kterých je pak taková síť složena. Pomůže to v představě toho, co se pak v takové síti děje a na jakém principu funguje. Základní koncepční schéma jediného neuronu, včetně popisu dílčích funkčních částí si lze prohlédnout na obr. 5.



Obr. 5: Detailní pohled na neuron [22].

Všechny vstupy (na obrázku modře s označením x) jsou napojené na náš neuron (zelený), každé cestě tohoto spoje je přiřazena váha (weight, označena w). Váhy jsou právě to, co se snažíme trénovat a co má být výstupem učení. Při trénování dochází zjednodušeně v nastavování vah pro dosažení cíle. Váha v tomto modelu udává, jak velký bude mít vstup na daný neuron vliv. Též se dá říci, že váhy zvýhodňují některé vstupy a jiné naopak penalizují. Každý neuron má pak také svou hodnotu vychýlení neboli bias, který umožňuje posun aktivační funkce vlevo či vpravo a určuje tak práh aktivity. Bias je též trénovacím parametrem.

Výsledná hodnota neuronu se pak spočte podle následujícího vztahu:

$$\sum_{i=1}^{m}(w_i x_i) + bias, \tag{1}$$

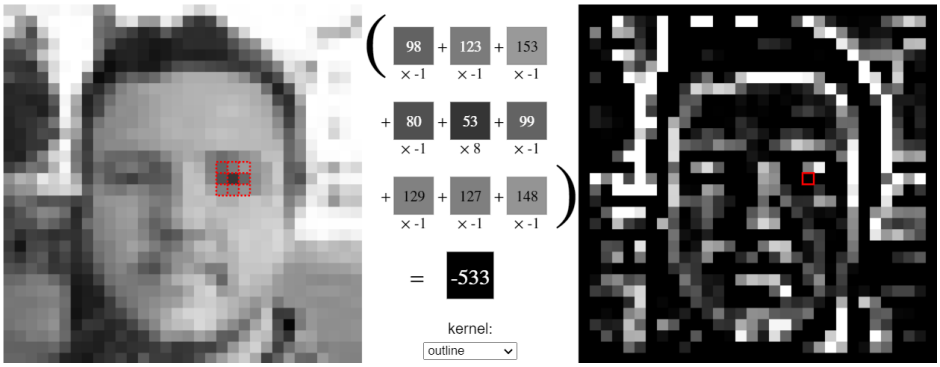
kde index m označuje počet vstupů.

Nyní v řetězci následuje aktivační funkce, která již z názvu vypovídá o tom, že sděluje, jak moc je neuron aktivován v závislosti na vstupu. Aktivačních funkcí je několik a z nejběžnějších se lze setkat s funkcí skokovou, lineární, hyperbolický tangens, sigmoid nebo ReLu. Pro přiblížení si lze nyní představit skokovou funkci, která uvádí neuron do stavu “aktivní” nebo “neaktivní” v závislosti na vstupních hodnotách a příslušných vah. Tato funkce je také znázorněna v obr. 5. [22]

6.5 KONVOLUČNÍ NEURONOVÉ SÍTĚ

KONVOLUČNÍ NEURONOVÉ SÍTĚ, též označovány zkratkou CNN nebo ConvNet z anglického convolutional neural network. Jedná se o speciální druh neuronových sítí, jejichž hlavní úkol je extrakce příznaků v obrázku. To lze popsat tak, že nám z obrázku extrahují hrany, kontury, zaostřují či jinak jej upravují pro další zpracování. [23]

Toto zpracování je prováděno postupným aplikováním jádra (též kernel) na celý obrázek. Jádro může představovat třeba matice 3x3x3 pro obrázek RGB. Příklad aplikace hranového detektoru na obrázek ve stupních šedi je znázorněn na obr. 6. Lze si povšimnout, že je zde použito jádro 3x3 o jedné dimenzi, to jest proto, že obrázek ve stupních šedi je také pouze jedno-dimenzionální, můžeme tedy vydedukovat, že dimenze jádra souvisí s dimenzí vstupního obrázku.



Obr. 6: Aplikování jádra hranového detektoru, vlevo vstupní obrázek, uprostřed výpočet na vybrané oblasti a vpravo výstup se zvýrazněnými hranami [24].

Výpočet konvoluce je matematicky zapsán v následující rovnici [25]:

$$y(t) = f \otimes x = \int_{-\infty}^{\infty} f(k).x(t-k)\,dk, \tag{2}$$

kde $\otimes$ označuje konvoluci a f a x jsou funkce přes rozsah t.

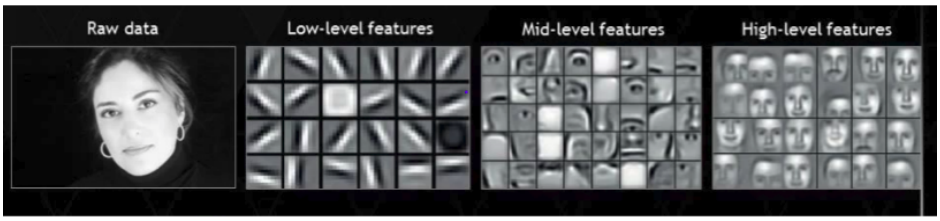
V případě zpracování obrázků, kde je aplikováno jádro na obrázek jako v případě obr. 6, probíhá takzvaná 2D konvoluce, jež je popsána následovně [26]:

$$y[i,j] = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} h[m,n] \cdot x[i-m, j-n], \tag{3}$$

kde x představuje vstupní obrázek, h je náš konvoluční filtr neboli kernel, m a n jsou jeho rozměry, pak y je výstupní obraz (též nazýván jako mapa příznaků). Z této rovnice je patrné, že je zpracovávána zmenšená část vstupního obrázku v závislosti na velikosti kernelu, proto se používají vycpávky, podrobněji popsané v subkapitole 6.5.

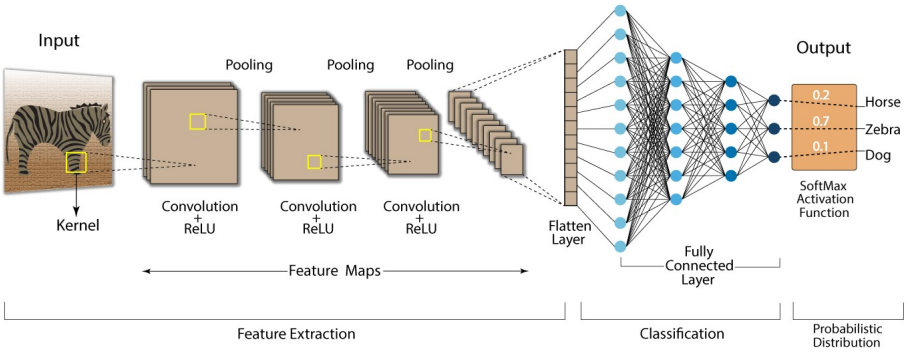
Mějme vstupní obrázek o rozměrech $W_1 \times H_1 \times C$, kde W_1 je šířka, H_1 výška vstupního obrázku a C je počet kanálů. Po aplikaci konvolučního jádra je rozměr výstupního obrázku $W_2 \times H_2 \times K$, kde $W_2 = (W_1 - F + 2P)/S + 1$ je šířka, $H_2 = (H_1 - F + 2P)/S + 1$ výška výstupního obrázku, K označuje počet filtrů, F je velikost filtru (kernelu), S je posun (stride) a P je vycpávka (padding). Potom lze určit i počet parametrů jako $(F^2C + 1)K$, kde číslo 1 vyjadřuje zahrnutí počtu bias pro každý filtr. [27]

Jak vypadají konvoluční filtry napříč vrstvami si lze prohlédnout na obr. 7, kde je patrné, že ze vstupního obrázku lidského obličeje jsou v nejnižších vrstvách nejprve extrahovány hrany, ve středních jsou to pak menší části jakožto nos, pusa nebo oko a v nejvyšší vrstvě jsou již naučené kompletní objekty jako zde třeba celý obličej.



Obr. 7: CNN features jak prochází konvolucí [28].

Výhodou a základním rozdílem konvolučních neuronových sítí v práci s obrazovými daty oproti běžným neuronovým sítím je, že jsou schopny se naučit lokální vzory, které jsou schopny nalézt kdekoliv na obrázku, kdežto plně propojená síť hledá globální vzory a pokud bychom to po ní chtěli, museli bychom ji to nechat znovu naučit nalézt v jiném místě. [20] Schéma kompletního řetězce CNN s koncovými dense vrstvami si lze prohlédnout na obr. 8.



Obr. 8: Schéma CNN s plně propojenou vrstvou [29]

Pooling

Sdružování nebo spíše známo pod anglickým názvem pooling je neodmyslitelnou součástí konvolučních neuronových sítí. S konvoluční vrstvou tvoří párovou dvojici, jež jsou řazeny hned za sebou, nejdříve konvoluce a poté pooling. Jeho hlavním úkolem je zmenšit nebo zredukovat velikost zpracovávaných dat a tím tak snížit nároky na výpočetní výkon a ušetřit trénovací čas. [30] Pooling se provádí způsoby popsanými níže.

- *Max-pooling*

  Jedná se v podstatě o jádro velikosti $M \times N$, kde nejčastěji je použito 3×3 nebo 2×2, přes které se pohlíží na vstupní matici a vybere z ní maximum. Velikost výstupní vrstvy se počítá obdobně jako u konvoluce. Posun neboli stride je zde zajištěn tak, aby se nepohlíželo na již stejné místo. Tato redukční metoda zachovává nejdůležitější příznaky, a proto je nejčastěji používána. [31]

- *Average-pooling*

  Tato metoda je principiálně stejná jako předchozí max-pooling, liší se pouze v tom, že z náhledové oblasti nevybírá maximum, ale počítá průměr, to však může mít za následek ztrátu výrazných rysů, a proto není tak používána, avšak v určitých situacích může být vhodnější.

Padding

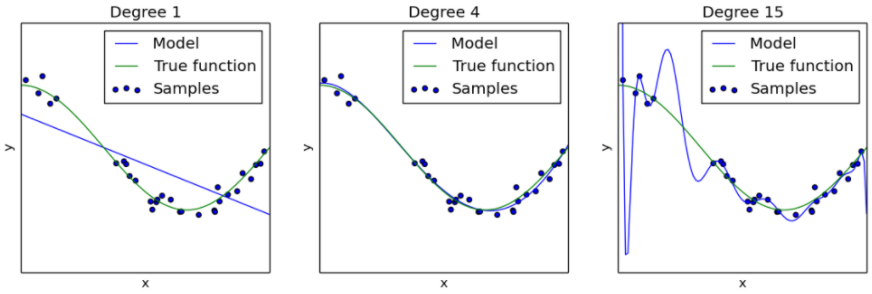
Pod českým názvem znám jako vycpávka. Jedná se o jakousi formu přípravy dat, než jdou do konvoluční vrstvy. Hlavní myšlenka je taková, že pokud použijeme kernel například o velikosti 3×3, tak nelze středem kernelu pohlížet na krajní pixely a tím nevyužije plně jejich významnost. Proto se dělá umělé rozšíření tak, aby střed kernelu byl aplikován i na krajní pixely. Nejčastěji se přidá řada pixelů, jejichž hodnota je 0 a jejich počet lze jednoduše určit ze vztahu $(F-1)/2$, kde F je velikost konvolučního jádra. Tato metoda se používá i v případě, kdy máme takový krok (stride), že kernel neprojde celý vstup. [30]

Trénování

Vlastní trénování neboli učení neuronové sítě lze definovat jako hledání zobecnění na trénovacích datech, protože od sítě nechceme, aby se naučila pracovat jen s daty totožnými v datasetu, ale i takovými, jež jsou pro ni neznámé. Tento problém se označuje termínem over-fitting neboli přetrénování.

Nejjednodušší formu trénování si lze představit například pod hledáním vhodné aproximační křivky bodů ve 2D, kde na ose X jsou časové úseky a na ose Y cena pohonných hmot. Máme dostatek dat a nyní se snažíme najít vhodné proložení, abychom byli schopni předpovědět vývoj ceny pro následující dny. Pokud bychom použili nízký stupeň aproximace, dopouštíme se velké odchylky a aproximace neodpovídá trendu dat, a tak výsledná predikce bude klesající, avšak trend je rostoucí. Pokud bychom však použili příliš vysoký stupeň, budeme mít ideálně proložená data s minimální odchylkou, avšak taková predikce je téměř nepoužitelná, protože model je příliš kmitavý a očekávaný výstup může být nahodilá hodnota. Na obr. 9 si lze prohlédnout tyto případy znázorněny graficky.

Ukazatelem toho, jak je náš model podtrénován nebo přetrénován, je takzvaná ztráta (loss) a v případě grafu se jedná o ztrátovou funkci (loss function) a vypovídá o chování modelu na validačních datech. Jedná se v podstatě o vzdálenost predikce od reálné hodnoty. Je tedy zřejmé, že se snažíme o co nejmenší ztrátu.



Obr. 9: Vlevo první stupeň aproximace (podtrénování), uprostřed čtvrtý stupeň (optimum) a vpravo patnáctý stupeň (přetrénování) [32].

Zpětné šíření

Též známo jako backpropagation. Jedná se o způsob, jakým je síť trénována a optimalizována. V podstatě se pomocí ztrátové funkce na výstupu ladí hodnoty vah při každém průchodu epochou. Též by se o tom dalo hovořit jako o zpětné vazbě, pomocí níž je model laděn tak, aby ztrátové skóre bylo co nejmenší a model tak co nejpřesnější.

7

DATA

TRÉNOVÁNÍ nebo také učení neuronových sítí je podobné tomu lidskému učení, učíme se z dat. Typickým příkladem může být třeba v této problematice neuronová síť. Pokud jste se s nimi nikdy předtím nesetkali, tak pro vás bude obtížné identifikovat obrázek či nákres plně propojené neuronové sítě jak je např. znázorněno na obr. 4. Pokud Vám nyní k obrázku řeknu, že je to neuronová síť, v řeči datových množin se toto označení nazývá "anotace" nebo případně "labeling" pro detekci, tedy přiřazení třídy nebo "labelu" k obrázku. Anotace říká, co je na obrázku a label, co je na obrázku a kde se objekt nachází.

Nyní víte, jak může vypadat neuronová síť, Váš mozek ji zanalyzuje jako řady kruhů, které jsou navzájem propojené. Tím jsme si vlastně popsali, jak taková neuronová síť funguje. Pokud bych Vám nyní ukázal obrázek neuronové sítě, která bude obsahovat více vrstev nebo více neuronů, tak předpokládám, že byste již věděli, že se jedná o neuronovou síť a dokázali tak obrázek klasifikovat.

Takové učení, jako probíhá u člověka se snažíme aplikovat i do křemíkového světa, kdy se pokoušíme naučit počítače "vidět". K tomu, abychom ale dokázali počítač dostat do stavu, kdy je schopen rozpoznávat objekty na úrovni člověka, potřebuje mnoho dat.

V této kapitole si popíšeme, jak taková data získat, na co se zaměřit, co je s nimi třeba udělat a jaké mohou nastat problémy.

7.1 VLASTNÍ DATA

Pokud se budeme zaměřovat nějakým problémem, jež je úzce zaměřený a ještě ho nikdo před námi neřešil, je téměř nevyhnutelné to, že se budeme muset zabývat sběrem a zpracováním vlastních dat. Touto problematikou se zabývá celá vědní disciplína a není jednoduché získat správná data.

Nyní budeme uvažovat sběr vlastních dat pro naši problematiku týkající se registračních značek. Prvně je nutné pořídit snímky vozidel, což bude časově náročné, jelikož množství snímků je v řádů tisíců pro kvalitní detektor. Pořizování snímků však značně stěžuje evropské nařízení o ochraně osobních údajů GDPR. Zde jsou osobní údaje velmi širokým pojmem a i RZ lze tak brát, pokud je možné identifiko-

vat vlastníka například podle místa pořízení. Podle RZ lze určit vlastníka, avšak tyto databáze jsou neveřejné. Pokud bychom však nechtěli balancovat na hraně zákona, je řešením souhlas majitelů vozidel pro zpracování, jež musí být prokazatelně doložitelný a následné bezpečné uložení dat podle GDPR.
To však sběr dat značně komplikuje. Dále pak různorodost snímků, abychom pokryli spektrum značek, s nimiž se lze setkat.

Již takto se zdá sběr dat náročný a to je dále třeba veškeré snímky anotovat pro trénování, jež si popíšeme níže. I přes svou náročnost je tato varianta pro některé problémy jediná cesta. V našem případě se této cestě vyhneme a použijeme nějaký z volně dostupných nebo komerčních datasetů popsané dále.

7.2 LABELING

Jak jsme se dozvěděli v úvodu této kapitoly, abychom umožnili počítači (neuronové síti) naučit se predikce na datech, které mu předáme, musíme mu s nimi také předat informace, které si z nich má vzít. Tedy mu říci, co na obrázcích je a kde v obrázku se objekt nachází.

Tato data se předávají v různých formách, ať už textový dokument s anotací celé datové sady nebo .xls soubor až po .json nebo .XML soubory, které mohou náležet každému obrázku zvlášť a obsahují anotace a ohraničující rámečky objektů (též nazývané jako “bouding boxes”).

Abychom tyto data správně připravili, nabízí se mnoho nástrojů a platforem. Dnes jsou k dispozici společnosti, které tuto přípravu udělají za Vás. Existují také nástroje, které vytvoří anotaci a labeling plně automaticky. Tyto nástroje fungují v principu na tom, že na nich již běží nějaký naučený klasifikátor nebo detektor a je aplikován na data, která mu předáte. Všechny tyto vymoženosti jsou však placené, tak se vydáme bezplatnou, avšak náročnější cestou.

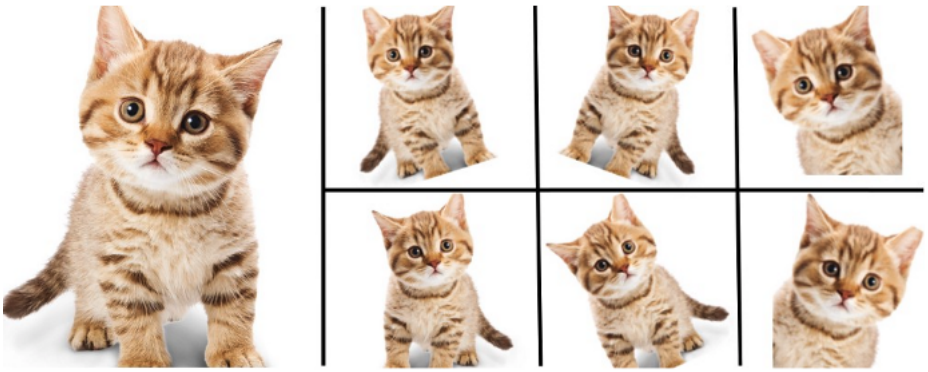
7.3 AUGMENTACE

Augmentace neboli česky rozšiřování je velmi častá praktika aplikovaná na trénovací data a je součástí přípravy dat. Jak jsme si již dříve zmínili, pro trénování je třeba velké množství dat, avšak ne vždy je správné množství k dispozici. Proto se dostupná data uměle rozšiřují o jejich modifikace a transformace.

Nejzákladnějšími rozšiřovacími úpravami jsou pootočení, zrcadlení, překlopení a jejich různé kombinace, ty si lze prohlédnout na obr. 10. Mezi dalšími lze zmínit přiblížení, oříznutí, posun v obraze, interpolaci, kdy po úpravě je obrázek s místech, kde není definován, vyplněn černou barvou. Na závěr nelze opomenout Gaussovský šum. [33]

Tyto praktiky se používají pro případy, kdy je dat dostatek, ale chceme dosáhnout větší přesnosti. Pro uvedení významnosti augmentace zmíním příklad z [33],

kde trénovaný model bez augmentace dosáhl přesnosti 76 % a model s augmentovanými daty 94.5 %. Lze si tedy udělat obrázek o její důležitosti a významnosti. Nutno však podotknout, že takových zlepšení lze dosáhnout pouze, pokud je dat velmi málo, pro velké datasety jsou to jednotky procent.



Obr. 10: Možnosti rozšíření datové množiny [34]

7.4 DOSTUPNÉ DATASETY

Pokud bychom se chtěli vyhnout potřebě manuálně anotovat data, je možné sehnat již anotovaná data. Velmi často jsou tyto datové množiny zpoplatněné, zvláště jedná-li se o úzce zaměřené odvětví. Na internetu však lze získat i mnoho datových množit bez poplatku. Obecně se jedná o datasety určené k soutěžím, na kterých se vyhodnocovala kvalita soutěžních klasifikátorů či detektorů.

Nyní se podíváme na nejpopulárnější z nich, jelikož se s nimi setká snad každý věnující se problematice neuronových sítí se zaměřením na zpracování obrazu, tak je dobré mít alespoň povědomí.

Povětšinou se jedná o obecné datasety, pomocí nichž jsou předtrénované sítě, které budeme používat, nebo se s nimi určitě setkáme. Poté si přiblížíme datové množiny vhodné na práci s RZ.

ImageNet

Nejznámější a také asi největší množinou, kterou lze volně získat je právě Imagenet. Její počátky se začaly psát již v roce 2006 a postupně s rostoucím rozvojem a zájmem o neuronové sítě rostla její popularita pro její ojedinělost. [35]

Stala se součástí mnoha soutěží a na jejích datech je natrénováno a předtrénováno mnoho neuronových sítích.

Je vhodná jak pro rozpoznávání, tak pro detekci. Je tvořena 15 mil. obrázky, které jsou rozděleny do 1000 tříd. Data jsou volně dostupná na webu.

Strukturou je zastoupeno mnoho zvířat a třídově jsou velmi detailně rozdělena, takže například u psa lze získat i plemeno. Celou hierarchii tříd si lze prohlédnout zde: https://observablehq.com/@mbostock/imagenet-hierarchy.

Na této datové množině byly předtrénovány sítě jako VGG16, VGG19, ResNet50, Inception nebo Xception pro framework Keras. [36] Tvoří tak často základ nově vznikajících sítí.

COCO

Jedná se o druhý největší, volně dostupný dataset. Název vycházející z Microsoft Common Objects in Context, proto někdy též označován jako MS COCO. Oproti předchozímu popisovanému obsahuje "pouze" 330 tis. obrázků, z toho více jak 220 tis. má přiřazený label. Obrázky jsou členěny do 80ti kategorií. Celkem bylo na datasetu vytvořeno více jak 2,5 mil. labelů. Jedná se o obrázky z každodenního světa s vícero objekty v jednom obrázku. [37]

Takový dataset je pak vhodný, pokud se snažíme natrénovat síť s multidetekcí. Také strukturou obrázků má pak natrénovaný model na této množině výrazně lepší představu o reálném světě a může dosahovat lepších výsledků.

Další výhodou tohoto datasetu je, že anotované objekty obsahují také segmentační informace. To znamená, že objekt má definované krom hraničního boxu také přesné pixelové ohraničení.

Na této množině je natrénován například i detektor YOLO. Příklad, jak mohou vypadat anotována data je znázorněn na obr. 11.

Obr. 11: Náhled, jak mohou vypadat data v COCO a jejich segmentace [38].

Ostatní obecné

Mezi další známé datasety, které jsou volně dostupné a velmi populární, lze zmínit ještě OpenImages od společnosti Google, který obsahuje přibližně 9 milionů obrázků [39]. Dále nelze opomenout PASCAL VOC dataset (Visual Object Clas-

ses), jež je populární u vydání z let 2007 a 2012 a je častým základem mnoha detektorů (např. také YOLO) a základní bází při srovnávání.

7.5 PLATESMANIA

Tak jako někdo sbírá známky, tak někdo sbírá snímky RZ a komunita těchto lidí je právě na Platesmania. Je zde přes 15 milionů snímků vozidel se zaměřením na registrační značky. Snímky pochází z celého světa, zastihují i unikátní typy značek, které nejsou tak časté, ale přesto je možné se s nimi setkat, proto data z tohoto serveru jsou vhodná pro potřeby projektu.

Přes širokou rozmanitost obsahuje také české RZ, konkrétně přes 48 tis. snímků, které zahrnují převážně běžné značky, ale je zde i nemalé množství atypických jako veteránské, převozní, sportovní nebo značky na přání. Snímky pochází z kamer, parkovišť, záběrů palubních kamer či výstav nebo závodů.

Nabízena je dokonce přímo zpoplatněná verze balíčků pro AI, kde je součástí i .XML soubor s anotovanými značkami, avšak bez bounding boxů RZ. Součástí anotace je také značka automobilu, takže je možné použít pro případné rozšíření tento dataset bez výraznějších úprav. Cenově se tato varianta pohybuje okolo 60 € za 50 000 snímků. [40] Přes svou rozsáhlost, obsah českých RZ a pro nás částečnou anotací se jedná o nejlepší možný dataset této problematiky.

7.6 KAGGLE - LICENSE PLATES DATASET

Jako alternativu lze brát data na serveru Kaggle, který čítá 433 snímků vozidel z reálného světa. Data jsou anotována včetně bounding boxů přes RZ. Anotace je však v souboru .XML a formátu VOC, to však není problém převést do formátu YOLO a .txt souboru pomocí nějakého volně dostupného scriptu z GitHubu. [41]

Množství snímků je však v této fázi nedostatečné, často různá rozlišení a opakující se snímky, proto je tento dataset vážně spíše alternativou nebo možný testovací dataset než trénovací.

7.7 AKADEMICKÁ PRÁCE

Podobnou problematikou a jí přidružených se zabývalo již několik prací, např. Bc. Aneta Kvapilová se ve své diplomové práci zabývala sběrem a zpracováním sbírky registračních značek vozidel.

Pro svou práci pořídila téměř 17 000 snímků a přibližně k 3 453 snímkům provedla ruční anotaci ve formátu YAML. Snímky jsou ve FHD rozlišení z reálného provozu českých silnic. [42] Data z její práce jsou k dispozici ke stažení, proto budou využita pro tuto práci.

8

ZPRACOVÁNÍ OBRAZU

V TÉTO kapitole se zaměříme na zpracování obrazu moderními praktikami za použití neuronových sítí. Jelikož se jedná o velmi širokou problematiku, rozhodl jsem se zaměřit primárně na detekci, kterou potřebujeme v naší práci. Dále pak na OCR (optické rozpoznávání znaků), které potřebujeme na přečtení znaků z vyznačené oblasti naším detektorem. Nebude ale v této teoretické části blíže zkoumáno, jelikož je založena na obdobném principu jako MNIST čtení znaků, který je považován jako "Hello World" pro AI. Konkrétní řešení bude však popsáno v praktické implementaci v kapitole 9.

Níže si popíšeme nejpoužívanější detekční architekturu YOLO, která nachází své uplatnění všude, kde je třeba rychlá detekce.

YOLO

Název vychází jako zkratka ze slov "You Look Only Once" jež doslovně vystihuje i princip fungování algoritmu. Na rozdíl od jednoduchého algoritmu posuvného okna (sliding window), který rozdělí obraz do několik částí a ty následně projdou detekčním algoritmem, tak YOLO má pohled na celý obrázek a je na něj algoritmus aplikován pouze jednou, tím jest vysvětleno odvození názvu. Řadí se tak do kategorie jednofázových detekčních modelů (stejně jako SSD).

První krok k porozumění tomu, jak YOLO funguje, je dekódování výstupu. Vstupní obrázek je rozdělen do mřížky velikosti $S \times S$ buněk. V každé takové buňce je predikováno B bounding boxů a C jako pravděpodobnost příslušnosti predikované třídě (C počet tříd). Každý box má pak paramery:

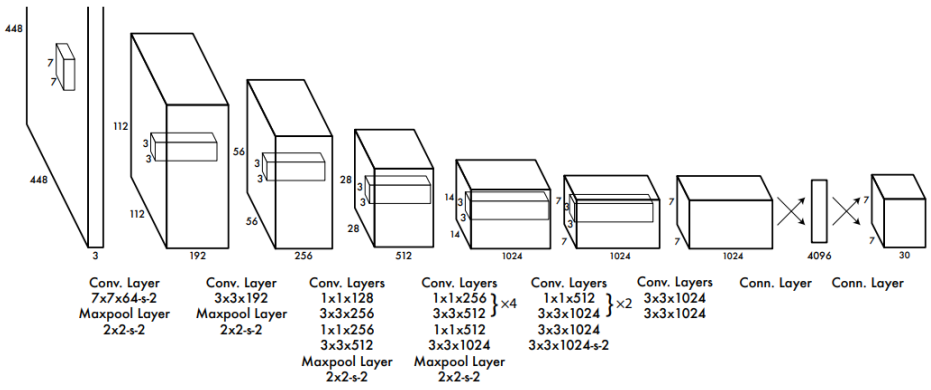
$$y = (p, x, y, h, w, c), \tag{4}$$

kde p je pravděpodobnost predikce umístění boxu, x a y je absolutní umístění středu boxu vůči rozměru obrázku (buňce), h a w jsou absolutní rozměry šířky

a výšky boxu vůči obrázku (buňce), c je pak příslušnost třídě. Velikost výstupního vektoru lze pak obecně určit následovně [43]:

$$y = S \times S \times (B * 5 + C). \tag{5}$$

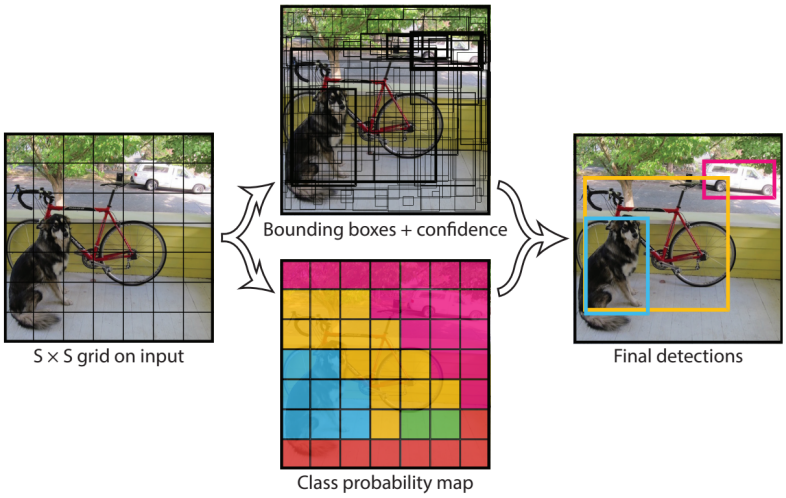
Architektura sítě vypadá jako klasická CNN, včetně konvolučních operací a poolingových vrstev. Na výstupu jsou pak dvě plně propojené vrstvy (dense). Celá architektura YOLO detektoru je znázorněna na obr. 12. Pro přiblížení pochopení toho obrázku, na vstupu se nachází obrázek s rozlišením 448×448 px v RGB formátu (dimenze 3), kernel je o rozměru 7×7 a posuv (stride) je 2, těchto filtrů je tam 64. Co zde není znázorněno, tak je aktivační funkce, ta bývá nejčastěji leaky ReLu. V dalším kroku následuje aplikace maxpoolingu a výstupem této vrstvy je tedy upravený snímek o rozměrech 112×112 a dimenze 192 (dimenze 3 původního * 64 filtrů). Obdobný postup je pro následující vrstvy. V předposlední vrstvě je tzv. zploštění (flatten) do vektoru a výstupem je formát, který byl popsán výše jako YOLO výstup. Zde se jedná o mřížku 7×7, $B = 2, C = 20$, takže po dosazení do vztahu 5 dostáváme číslo 30. Nutno také zmínit, že každá buňka může predikovat pouze jeden objekt. Z toho vychází limitace detekce vícero malých objektů.



Obr. 12: Architektura sítě YOLO [44]

Pro popsání principiální funkčnosti detektoru byl vybrán názorný obr. 13, který se také nachází v oficiální verzi dokumentu. Vlevo je první krok, tím je rozdělení do mřížky $S \times S$. V dalším kroku jsou v každé buňce predikovány bounding boxy (pokud obsahují objekt) s hodnotou pravděpodobnosti. To je vidět v horní části uprostřed, kdy je zde velké množství boxů, míra pravděpodobnosti je dána tloušťkou rámečku. Pokud bychom si vzali nyní pouze nejvyšší pravděpodobnosti příslušnosti dané třídy a vybarvili celou buňku podle barev příslušící třídě, dostaneme obrázek uprostřed dole. V posledním kroku dáme prostřední obrázky dohromady a odfiltrujeme boxy takzvanou prahovou hodnotou (treshold), tak aby nám zůstaly

pouze ty s nejvyšší pravděpodobností. [44] Nyní máme vyřešenu detekci objektů a jsme schopni říci, kde se objekt nachází a jaké přísluší třídě.



Obr. 13: Grafické znázornění funkčnosti YOLO v krocích [44]

YOLO verze

Výše zmíněné principy souvisí s obecným přístupem tohoto detektoru. V praxi se však vyskytují pod různými verzemi. Obecný přístup je použit v 1. verzi. Vylepšení se týkají rychlosti a kvality predikce, kdy s vyššími verzemi přichází lepší výsledky. Je také vylepšena detekce vícero objektů a zvětšení počtu buněk (například 2. verze má mřížku 13x13). V současnosti je možné se setkat s 5. verzí, ale ta není oficiální, jelikož každý může architekturu volně upravovat a vydávat.

IoU

Tato zkratka znamená "Intersection over Union" a pro detektor při trénování to má význam určování kvality predikcí, kdy se měří velikost překrytí predikovaného bounding boxu a anotovaného (označován jako ground truth). Její vyjádření lze zapsat následujícím vztahem:

$$IoU = \frac{P \cap G}{P \cup G}, \tag{6}$$

kde P značí predikovaný box a G anotovaný ground truth box. Je zřejmé, že čím větší překrytí, tím více se hodnota IoU blíží číslu 1 a model tak funguje lépe.

mAP

Jedná se o ukazatel kvality detektoru. Výraz znamená "mean Average Precision", tedy průměrnou přesnost. Průběh je takový, že se pro každou třídu měří AP (Average Precision), jež vyhodnocuje přesnost detekce z IoU, kde je nastavena hranice (nejčastěji 0.5) a pokud je IoU nad touto hranicí, je tato detekce jako pozitivní (TP - true positive) a pokud pod, tak jako negativní (false positive). Z těchto hodnot je vypočtena přesnost (precision), dále je třeba pro AP vypočíst recall, jež se vypočte z TP a takzvané nesprávné negativní detekce (FN - false negative).

$$precision = \frac{TP}{TP + FP}, \quad recall = \frac{TP}{TP + FN}, \tag{7}$$

pak se AP vypočte ze vztahu níže a z něj pak i mAP:

$$AP = \int_0^1 p(r)dr, \quad mAP = \sum_{i=1}^{n} \frac{AP_i}{n}, \tag{8}$$

kde funkcí je závislost precision (p) na recall (r) a n je počet tříd [45].

OSTATNÍ

Mezi další detektory se řadí R-CNN a jeho rychlejší, modernější odvozeniny FAST R-CNN a FASTER R-CNN, jež se řadí mezi dvoufázové modely založené na klasifikaci. To znamená, že v první fázi dochází k navrhování oblastí a klasifikaci v těchto oblastech, načež podle výsledků se provede detekce. Ze zástupců jednofázových modelů jako je YOLO, lze zmínit SSD detektor (Single Shot Detector).

Ani jeden z těchto detektorů nebude v této práci dále uvažován, jen je nezbytné se o nich zmínit. Důvody použití YOLO vychází ze srovnávací tabulky 3, kde na základě rychlosti je jasně nejrychlejší a tím pádem nejvhodnější pro použití na mikrokontroléru. [46]

Tab. 3: Srovnání YOLO s jinými detektory.

Model	Dataset	mAP	FPS
YOLOv2 544x544	VOC 2007+2012	78.6	40
Tiny YOLOv2	VOC 2007+2012	57.1	207
YOLO	VOC 2007+2012	63.4	45
SSD500	VOC 2007+2012	76.8	19
Faster R-CNN VGG-16	VOC 2007+2012	73.2	7
SSD500	COCO	46.5	19
YOLOv2 608x608	COCO	48.1	40
YOLOv3-320	COCO	51.5	45
Tiny YOLOv3	COCO	33.1	220

9

VLASTNÍ ŘEŠENÍ

Tímto se dostáváme k praktické implementaci vlastního řešení a uplatnění nabytých znalostí, které jsme si přiblížili v předchozích kapitolách.

Nyní je našim úkolem natrénovat vlastní model pro ANPR na datech, jež byla vybrána v kapitole 7 a zpracovaná způsoby, které popisuje tatáž kapitola.

V prvotní fázi nám půjde pouze o detekci RZ v obraze a vrátit oblast, v níž se RZ nachází. To je důležité z důvodu rychlosti zpracování, aby nedocházelo ke zpracovávání textu volně v obraze. V druhé fázi se budeme snažit ve vyznačené oblasti najít znaky, které se předají ve třetí fázi rozpoznávači znaků. V závěru se znaky seřadí v pořadí umístění v obraze a rozpoznaný text je možné předat k dalšímu zpracování.

Po úspěšné implementaci se budeme snažit zakomponovat tento ANPR script na kompaktní zařízení s podporou kamery, kterým je pro nás v této práci Nvidia Jetson Nano, jež byla vybrána a popsána v kapitole 5. Poté testování, optimalizace a závěrečné zhodnocení a prostor pro optimalizaci.

FRAMEWORKY

Nejprve bychom si osvětlili, co to vlastně "framework" je, pomůže nám to chápat další odstavce. V našem jazyce hovoříme o "aplikačních rámcích" a je to hotová struktura, která má již implementované programy, knihovny a API. Její účel krásně a přesně vystihuje tato věta: „*Cílem frameworku je převzetí typických problémů dané oblasti, čímž se usnadní vývoj tak, aby se návrháři a vývojáři mohli soustředit pouze na své zadání.*“ [47] Proto i my budeme používat frameworky při práci, nemusíme tedy vymýšlet již vymyšlené.

V této práci byl použit DarkNet. Jedná se o Open-source framework pro neuronové sítě napsán v jazyce C a CUDA, což je speciální jazyk pro práci s GPU a paralelními operacemi, to se tedy přímo vybízí pro použití ve spojení s NS. Je to také oficiální framework pro práci s YOLO detektorem, je to od stejného vývojáře. [48] Dále byl použit framework Keras, jehož autorem je François Chollet, autor také mnoha publikací, ze kterých bylo čerpáno i pro tuto práci. Framework je napsán v jazyce Python a je tak proto velmi uživatelsky přívětivý a snadno

použitelný. Často je právě Keras první věcí, s kterou se každý programátor začínající s deep learningem setká. Dnes je Keras již součástí frameworku TensorFlow od společnosti Google.

VYTVÁŘENÍ ANOTACE A ZPRACOVÁNÍ DAT

Pro trénování detektoru byla vybrána data, která bylo možné získat z diplomové práce [42], kde se autorka zabývala sběrem dat. Z nasbíraného datasetu bylo vybráno 1337 snímků ve FHD, na nichž se vozidla nacházejí v rozumné vzdálenosti pro zpracování a jsou relativně ostře vyfocena. Celkem se na snímcích nachází 2225 registračních značek a maximální počet vozidel na snímek je 5. Dále bylo nutné veškeré snímky zmenšit na VGA rozlišení, které je vhodné pro trénování, jelikož pak pro následné nasazení je toto právě rozlišení kamery hostitelského zařízení. Data nebyla nikterak augmentována. Následně bylo nutné na zmenšených snímcích vytvořit anotaci RZ.

Pro manuální anotaci a vytváření labelů jsem vybral nástroj LabelImg (dostupné z GitHub pod licencí MIT). Jedná se o jednoduchý nástroj napsaný v jazyce Python. Jeho spuštění je tedy možné přímo ze staženého repositáře, kdy stačí spustit soubor “labelimg.py” pomocí python konzole.

Práce je vskutku jednoduchá, postačí vybrat složku s našim datasetem, vybrat místo pro ukládání anotací a také formát výstupu. V našem případě, kdy budeme pracovat s YOLO detektorem, je vhodné zvolit výstupní formát YOLO, avšak moderní frameworky, knihovny a nástroje umožňují úpravy a importy napříč formáty. Nevýhodou ovšem je, že takto musíte projít opravdu každý obrázek, což může být zdlouhavé a po chvilce také nezáživné, ale na druhou stranu je zajištěna kvalita označených dat a také jejich selekce, kdy lze narazit na nevhodný nebo duplicitní příklad.

Příklad označování a vzhled prostředí si lze prohlédnout na licencovaném obr. 14.

DETEKTOR RZ

Pro detekci RZ z předkládaných snímků byl pro tuto práci vybrán detektor YOLO, jež byl popsán v kapitole 8. Konkrétněji se jedná o verzi 3. Vybrán byl z důvodu rozšíření, podpory při trénování a hlavně jeho kvalitu a rychlost. Již v teoretické kapitole si lze všimnout, že dosahuje velmi vysokých rychlostí se zachováním vysoké kvality a je tedy vhodný pro zpracování obrazu v reálném čase i na méně výkonném zařízení. Tím je právě Maix Bit I, na které toto řešení bude směřovat, proto je bráno pořád v úvahu s kompatibilitou, i když v této práci bude použita Nvidia.

Obr. 14: Vytvoření labelu registrační značky v LabelImg s výstupem pro YOLO na obrázku automobilu [49].

Trénování detektoru

Pro trénování byla zvolena cloudová platforma Gradient od společnosti Paperspace. Jsou nabízeny tři druhy předplatného, já zvolil variantu Developer G1 s tarifem 8 $/měsíčně. Oproti bezplatné variantě je zde možnost vytvářet soukromé projekty, 200 GB úložiště, emailová podpora a možnost běhu až 5 notebooků. Vše probíhá online v prostředí Jupyter. Jako výpočetní zařízení si lze vybrat mezi několika instancemi, pro svou práci jsem uznal za vhodnou variantu P4000 se stejnojmenným modelem grafické karty od Nvidia s pamětí 8 GB GPU a 1792 CUDA jader, 30 GB RAM a 8-jádrové CPU. Cena běhu této instance je 0.51 $/hod. Společnost poskytuje 20% slevu pro studenty a vzdělávací instituce. [50]

Při snaze zprovoznit virtuální stroj se však ukázalo, že je třeba nainstalovat kompletní podporu GPU v podobě cuDNN a dalšího nastavení, včetně podpůrných knihoven. Z toho důvodu bylo snazší využít vývojovou platformu Google Colab, jež pracuje na nadstavbě Jupyter notebooku. Je zde k dispozici free výpočetní výkon na grafických kartách Nvidia K80. Zde již nebylo třeba nic doinstalovávat, vše potřebné bylo k dispozici, postačilo pouze naimportovat. Nutno podotknout, že je zde omezení v podobě doby běhu instance, která je 90 minut (myšleno doba neaktivity) a pak maximální 12 hodin, po jejichž uplynutí se virtuální stroj smaže s veškerými daty. Z toho důvodu byl jako první krok napojit Google disk s tímto strojem, na který se v průběhu trénování ukládala záložní data vah každých 100 iterací a také kopie celého Jupyter notebooku.

Před trénováním bylo nutné stáhnout z GitHubu DarkNet a nastavit kompilační soubor, kde bylo třeba povolit podporu GPU, CuDNN a OpenCV, poté bylo

možné zkompilovat. Dále bylo třeba před trénováním upravit konfigurační soubor zvoleného detektoru, zde bylo potřeba nastavit hodnotu batch, počet filtrů a počet detekčních tříd. Dále bylo nutné nahrát trénovací data s anotačními soubory do složky dat přímo v DarkNetu.

Nyní již bylo vše připraveno pro trénování YOLO detektoru registračních značek. Spuštění proběhlo s předáním konfiguračního souboru a předtrénované CNN DarkNet53, která je volně dostupná na stránkách vývojáře YOLO, následujícím příkazem.

```
./darknet detector train data/obj.data cfg/yolov3-train.cfg
darknet53.conv.74 -dont_show
```

Při trénování se zálohy osvědčily, jelikož je trénování časově náročné, v tomto případě probíhalo přibližně 5 hodin a asi hodinu před koncem se iterace přerušila. Jelikož byla k dispozici záloha, bylo možné stroj pustit z této zálohy.
Dá se říci, že to funguje jako nové trénování, s tím že jako předtrénovaný model se použije částečně natrénovaný ze zálohy.

```
./darknet detector train data/obj.data cfg/yolov3-train.cfg
/content/gdrive/MyDrive/yolo_backup/yolov3-train_last.weights
-dont_show
```

Po dokončení trénování se na Google disku objeví nový soubor s finální hodnotou vah, který je naší naučenou sítí. Pro otestování funkčnosti sítě je třeba si z DarkNetu vzít konfigurační soubor dané verze s příponou “testing.cfg” a pak náš soubor s třídami, jež máme od anotace. Po načtení těchto souborů je možné detektoru předávat snímky s vozidly. Vytvoření třídy s modelem je voláno následující funkcí.

```
lpd = LPDetector(
        pth_weights='yolov3.weights',
        pth_cfg='yolov3_testing.cfg',
        pth_classes='classes.txt'
    )
```

Příklad výstupu detektoru je znázorněn na obr. 15, kde byli odfiltrováni kandidáti s pravděpodobností pod 80 %. Na zvolením příkladu je predikce 98 %, což lze pokládat za velmi kvalitní výsledek.

Detektor má velmi dobré výsledky při přímém pohledu, horší je to při pohledu z většího úhle, kdy predikce klesne až k 50 %, ale i přesto je RZ predikována na správném místě. To jest dáno charakterem trénovacích dat, kdy jsou převážně přímé pohledy nebo z malého úhlu. Tento nedostatek by bezesporu vyřešila augmentace, která by spočívala v prostorovém pootočení obrázku a tím by tak byl simulován pohled ze strany a natrénovat tak model znovu. V této práci však bude dále uvažován pouze již natrénovaný model bez augmentace.

Obr. 15: Aplikace natrénovaného detektoru s vykreslením detekované části a přesností [49].

SEGMENTACE ZNAKŮ

V tomto kroku máme již k dispozici souřadnice umístění RZ na snímku a je třeba jednotlivé znaky na RZ detekovat pro rozpoznávání. Při této práci byla použita metoda segmentace za pomocí knihovny OpenCV.

Samotné segmentaci předcházel preprocessing s obrazem. Konkrétněji došlo k převodu barevného obrazu do stupňů šedi, jelikož barvy v tomto případě nepotřebujeme a ušetří se čas na zpracování. Dále byl aplikován Gaussův šum, který má za efekt snížení šumu, a následně binární převod barev, tak že na výstupu je pouze černobílý obrázek. Takový výstup je znázorněn na obr. 16. V takovém obrázku se snadněji hledají znaky na základě kontrastu.

Byla zde použita funkce na hledání kontur, která právě hledá obrazce, jež mají stejný barevný základ a jsou ohraničeny jiným. Návratem funkce je seznam kandidátů (obrazců). Z tohoto seznamu je nutné odfiltrovat kandidáty, jež nejsou znaky RZ. To je provedeno podmínkou, která tvoří propustnost na základě výšky kandidáta ku výšce výřezu značky, jelikož tyto parametry známe z kapitoly 3. Další podmínka pro pokročilou selektivitu je omezení se na poměr stran znaků, která také můžeme určit z kapitoly 3 a je to přibližně poměr $1 : 1,5$.

Z důvodu pohledu pořizovaného snímku z různých úhlů je nutné nastavit větší volnost v propustnosti u podmínky týkající se poměru výšky znaku ku výšce RZ (výřezu z obrázku), jelikož pak je výřez vyšší a tento poměr se snižuje a pod-

Obr. 16: Zpracování v podobě odstranění šumu a převodu do černo-bílé [40]

.

mínka by nebyla splněna a znaky by neprošly selekcí. To však nepříznivě působí na RZ, které jsou pořízené z přímého pohledu, při selekci pak dochází například k propuštění u znaku “0” i vnitřního kruhu jako samotného znaku a následně k rozpoznání jako “00”, výstup je znázorněn na obr. 17. To lze však ošetřit další podmínkou na umístění středu těchto znaků. Další problém této selekce je písmeno “I”, které má naopak velmi malý poměr stran a tak jej tento selektivní filtr nepropustí.

Obr. 17: Problém při segmentaci [42].

Pro tento případ byl hledán kompromis mezi propustností pohledu ze strany a maximálnímu omezení propustnosti nežádoucích kontur. Řešením by mohlo být omezení se pouze na přímý pohled, což se dá ovlivnit umístěním kamery. Další možností je prostorová transformace obrazu, pomocí níž by došlo k “narovnání” RZ a byla by tak jako z pohledu přímo. Toto jsou však velmi náročné operace na výpočetní výkon a pro nasazení na mikrokontroléru tak neakceptovatelné. Výstup po správné segmentaci si lze prohlédnout na obr. 18.

V pokračování projektu, jež staví na této práci, tak bude z důvodu popsaném výše, použit detektor na bázi neuronových sítí, jež bude mít za úkol vyhledat znaky v předloženém výřezu. Při použití akcelerátoru bude tento proces tak trvat i kratší dobu.

Obr. 18: Segmentované znaky v oblasti RZ [40]

ČTENÍ ZNAKŮ

V této části byly vyzkoušeny dvě metody. První bylo použití open-source engine Tesseract OCR od společnosti Google s obsluhou jazyka Python za pomoci PyTesseract knihovny. Tento engine slouží k převodu textu v obraze do textové formy v počítači.

Tomuto engine byl předáván pouze výřez s RZ s odstraněním šumu a binarizací na černé znaky na bílém pozadí a bez předchozí segmentace znaků. Výsledky se však příliš vzdalovaly od toho, co bylo na snímcích. Výsledky se drobet zlepšily po předání seznamu povolených znaků enginu, ale i přesto se nedalo hovořit o kvalitním rozpoznávání. S určitou přesností byla rozpoznána čísla, písmena téměř vůbec. Z toho důvodu nebyl tento engine dále testován a byla použita jiná metoda. Níže znázorněna práce s tímto enginem:

```
import cv2
import pytesseract
pytesseract.pytesseract.tesseract_cmd =
    r'C:\Program Files\Tesseract-OCR\tesseract.exe'

img = cv2.imread('canny_edged.jpg') #nacteni vyrezu s upravami
custom_config = r'--oem 3 --psm 11 -c
↪  tessedit_char_whitelist=0123456789ABCDEFHIJKLMNPRSTUVXYZ'
lp = pytesseract.image_to_string(img, config=custom_config)
```

Druhou metodou bylo rozpoznávání za pomocí rozpoznávače na bázi neuronové sítě. Na GitHubu je k dispozici váhy neuronové sítě, jež je natrénovaná na sadě znaků, kde každý zástupce obsahuje přibližně 1 000 snímků, jsou bíle na černém pozadí o velikosti 28 x 28 px. Klasifikátor znaků je na MobileNets architektuře a byla použita předtrénovaná síť na ImageNet, jež byla popsána v kapitole 7. Trénován byl ve frameworku Keras. [51]

Tomuto klasifikátoru byly předávány segmentované znaky jeden za druhým a výsledky jsou velmi dobré. Pokud jsou znaky segmentovány správně, tak rozpoznání

probíhá s vysokou přesností. Jediný problém nastával, pokud byl vstupní obrázek v nižším rozlišení a detekovaná značka vzdálenější. Docházelo pak k záměnám znaků jako například “0” a “D”. Tento problém pomohlo vyřešit upscalling výřezu na konstantní velikost 460 x 130 px. Opět i zde je možnost omezit se pouze na RZ do určité vzdálenosti a předejít tak zvyšování rozlišení a tím zvýšení času pro rozpoznávání. To však bude řešeno až při aplikaci na konkrétní zařízení.

Obr. 19: Finální výstup po rozpoznání znaků [40]

IMPLEMENTACE NA ZAŘÍZENI

V této podkapitole se budeme zabývat implementací scriptu, který je schopen rozpoznávat RZ a byl popsán výše. Implementace bude prováděna na Nvidia Jetson Nano, jež byla popsána v kapitole 5. Budeme se zabývat od úplného oživení zařízení a jeho přípravy až po implementaci a následné využití kamery a předávání snímků na rozpoznání pro použití v reálném čase.

Uvedení do provozu

Pro běhový systém byl vybrán oficiální od Nvidia, jež je stavěn na linuxové distribuci Ubuntu 18.04 a obsahuje rovnou JetPack a další nástroje. Je navíc op-

timalizován pro běh na tomto zařízení. Během samotného uvádění do provozu se nevyskytly problémy, stačilo pouze následovat instrukce.

Problémy však nastaly při instalaci knihoven, které jsou nutné pro běh vytvořeného scriptu. Konkrétně jsem se potýkal s nekompatibilitou verzí jednotlivých knihoven, které působily problémy již při samotné instalaci, jelikož jsou závislé jedna na druhé. Bylo tedy nutné zajistit kompatibilitu jednotlivých knihoven. Zásadní podmínkou pro všechny je přítomnost jazyka Python verze 3.6. Celý postup řešení tohoto problému je řešen v návodu přiloženém v příloze. V následujícím seznamu je přehled verzí knihoven, na kterých vše funguje.

SciPy	1.5.4
NumPy	1.19.5
Sklearn	0.24.2
OpenCV	4.5.1
TensorFlow	2.4.0
Keras	2.4.3
MatPlotLib	2.1.1

Dále bylo nutné se vyvarovat chybám při importu knihoven, kdy byly vraceny hlášky jako “Cannot allocate memory in static TLS block” nebo “Core dumped”. To se podařilo vyřešit umístěním následujících příkazů před spuštěním scriptu nebo umístění do bash souboru:

```
export OPENBLAS_CORETYPE=ARMV8
export LD_PRELOAD=/usr/lib/aarch64-linux-gnu/libgomp.so.1
```

V dalších krocích byla zjišťována výkonnost a přišlo se na to, že knihovna OpenCV, jež je defaultně nainstalována v obraze systému, není kompilována s podporou CUDA, která zvyšuje výkonost a rychlost za pomocí běhu na GPU. Z toho důvodu následovalo stažení repozitáře této knihovny ve verzi 4.5.1 a upravení buildu pro podporu CUDA. To vše provedl shell script, jež je také součást přílohy. Vše potřebné se nainstalovalo či aktualizovalo bez nutnosti dohledu. Celý tento proces trval přibližně 5 hodin. Poté již bylo možné naimportovat OpenCV s podporou CUDA.

Pro zajištění maximální výkonosti a poskytnutí výkonu pouze pro běhový script bylo vypnuto GUI, které se stará o grafické prostředí systému, takto bylo ušetřeno přibližně 300-400 MB RAM, jež mohou být zásadní u 2GB verze, které jsou využity na 100 % při běhu. Takový stroj pak bez zatížení zabírá přibližně 300 MB RAM a je nutné obsluhovat jej pouze unixovými příkazy z terminálu.

TESTOVÁNÍ A SROVNÁNÍ

Script byl dosud provozován a testován na osobním laptopu s CPU Intel i7 8550u bez dedikované grafické karty, pouze integrované UHD Graphics 620, 16 GB RAM

a SSD se čtením 3 500 MB/s. Na tomto stroji bylo dosaženo času rozpoznání nejčastěji v rozmezí 600-900 ms, nejlepší čas až na hranici 580 ms.

Na Nvidia Jetson Nano bylo dosaženo rozpoznávacích časů v rozmezí 2-3 s, v průměru však 2.34 s pro 12 testovacích snímků a nejlepší čas 2.03 s. Pro stejné snímky na laptopu bylo dosaženo času v průměru 658 ms.

Zařízení bylo opatřeno kamerou Logitech c270 s rozlišením 720p a je tedy takto schopno rozpoznávat v provozu a pracovat s návratovou hodnotou RZ například za účelem omezené přístupnosti pouze některým vozidlům nebo podobné aplikace, založené na tomto principu.

Toto vše však s limity a nevyhnutelnými kompromisy, které je nutné udělat. Vyplývají s omezením úhlu a vzdálenosti pozorování a byly popsány výše v této kapitole.

10

ZHODNOCENÍ VÝSLEDKŮ A SMĚRY POKRAČOVÁNÍ PRÁCE

ZÁVĚREM této práce přichází finální zhodnocení výsledků. Úkolem bylo seznámit se s umělou inteligencí, se zpracováním obrazu a dostupným hardwarem, který tyto AI metody podporuje. Dále bylo zadáním určeno pomocí získaných znalostí vytvořit vlastní software, který bude založen na AI a následně jej implementovat na zařízení vlastního výběru, které bude vycházet z průzkumu.

ZHODNOCENÍ DOSAŽENÝCH VÝSLEDKŮ

Na základě prostudovaných teorií a průzkumů byl vytvořen detekční algoritmus RZ, který je založen na konvolučních neuronových sítí. Jedná se o detektor YOLOv3, který byl trénován na Google Colab s využitím GPU na datasetu čítající 1337 snímků, které byly manuálně anotovány a nachází se v nich celkem 2225 RZ.

V další části byly v detekované oblasti s RZ segmentovány znaky pomocí OpenCV knihovny a praktik zpracování obrazu a následně jednotlivé znaky rozpoznávány pomocí neuronové sítě MobileNets na čtení znaků. Jedná se o obdobu rozpoznávače MNIST i se znaky.

Celý tento script byl implementován na Nvidia Jetson Nano 2 GB, které bylo pro běh řádně připraveno a optimalizováno pro dosažení co nejvyššího možného výkonu pro běh scriptu. Následně byla opatřena kamerou a je tak schopna běhu v provozu.

Při běhu bylo dosaženo času rozpoznávání na testovacích snímcích v průměru 2.34 s, což je na takové zařízení s příznivou cenovkou přijatelné. Pro srovnání kvality s dostupnými řešeními by bylo nutné je mít na stejné platformě, což nebylo možné, protože většina není schopna běhu na mikrokontrolérech.

Úplným závěrem jako autor práce hodnotím zkoumání jako zdařilé, jelikož zde na konci práce mám v rukou zařízení, které je schopno hostit vlastní SW, přijímat snímky z kamery a vracet výsledky s průměrnou odezvou 2.34 s, což je odezva použitelná v reálném zařízení.

SMĚRY POKRAČOVÁNÍ PRÁCE

Při této práci byly ověřeny metody zpracování obrazu s následnou implementací na dostupný hardware. Byly tak položeny základy pro úspěšné rozběhnutí první fáze školního projektu, jež bude z této práce vycházet a směřovat tak, aby byly splněny všechny požadavky zákazníka, jak byly popsány v kapitole 2 této práce. Bude třeba dále rozpracovat a zdokonalit některé části, které byly kompromisem v této práci, ale umožnil tak vyvinout základ projektu, který bude nyní optimalizován pro použití v praxi.

NÁVRH NA OPTIMALIZACI PRO POKRAČOVÁNÍ

Jak již bylo zmíněno v kapitole 9, potenciálně nejslabším článkem tohoto zpracovávajícího řetězce je segmentace znaků, ve kterém by mohla vzniknout chyba. Proto by další postup v projektu bylo nahradit OpenCV pro segmentaci sofistikovanějším detektorem na bázi NS a výsledný SW by měl získat ještě větší robustnost, zvýšení rychlosti a přesnosti. Mohlo by to být opět YOLO, které by hledalo znaky. To by se dalo spojit rovnou i s klasifikací znaků, avšak byla by mnohem náročnější anotace a pro posouzení kvality tohoto návrhu by bylo nutné porovnat tyto dvě varianty.

Pro dosažení výkonu z vybraného zařízení bylo uděláno maximum, další možností by bylo použít totožné zařízení s větším množstvím paměti, konkrétně 4 GB RAM.

A

PŘÍLOHA

Kódy	ANPR script a další v repozitáři projektu BP
Shell script	script na kompilaci OpenCV v repozitáři projektu BP

B

PŘÍLOHA

Datasety veškeré použité datasety na Google Disku

C

PŘÍLOHA

Návod sada instrukcí na zprovoznění knihoven na GitLabu projektu BP

SEZNAM LITERATURY

1. WIKIPEDIA. *Státní poznávací značky v Česku* [online]. San Francisco (CA): Wikimedia Foundation, 2001 [cit. 2021-03-16]. Dostupné z: https://cs.wikipedia.org/wiki/St%C3%A1tn%C3%AD_pozn%C3%A1vac%C3%AD_zna%C4%8Dky_v_%C4%8Cesku.
2. *Installation Guide of Hikvision LPR cameras* [online]. Itálie [cit. 2021-03-16]. Dostupné z: https://www.televista.it/siti/TELEVISTA/img/upload/Prodotti/file_1/ZOOM_installation%5C%20guide%5C%20of%5C%20hikvision%5C%20lpr%5C%20camera.pdf.
3. *Vyhláška č. 343/2014 Sb. Vyhláška o registraci vozidel* [online]. 2014 [cit. 2021-03-18]. Č. 343. Dostupné z: https://www.zakonyprolidi.cz/cs/2014-343.
4. *Commercial-enhancements* [online]. Rekor, 2021 [cit. 2021-05-25]. Dostupné z: https://docs.rekor.ai/opensource.html%5C#commercial-enhancements.
5. *OpenALPR* [online]. Maryland, USA: OpenALPR, 2021 [cit. 2021-05-20]. Dostupné z: https://www.openalpr.com/software/carcheck.
6. *OpenALPR* [online]. San Francisco (CA): Wikimedia Foundation, 2001 [cit. 2021-05-25]. Dostupné z: https://en.wikipedia.org/wiki/OpenALPR.
7. *Plate Recognizer* [online]. Silicon Valley: Plate Recognizer, 2021 [cit. 2021-05-20]. Dostupné z: https://platerecognizer.com/.
8. *Adaptive Recognition* [online]. Budapest: Adaptive Recognition, 2021 [cit. 2021-05-20]. Dostupné z: https://adaptiverecognition.com/.
9. *Vivotek* [online]. Nová Tchaj-pej: Vivotek, 2021 [cit. 2021-05-20]. Dostupné z: https://www.vivotek.com.
10. *Vaxtor* [online]. Velká británie: Vaxtor, 2021 [cit. 2021-05-20]. Dostupné z: https://www.vaxtor.com/.
11. *NeuralLabs* [online]. Barcelona: NeuralLabs, 2021 [cit. 2021-05-20]. Dostupné z: https://www.neurallabs.net/en.
12. *Raspberry Pi* [online]. San Francisco (CA): Wikimedia Foundation, 2001 [cit. 2021-05-20]. Dostupné z: https://en.wikipedia.org/wiki/Raspberry_Pi.
13. *Jetson Nano Developer Kit* [online]. Kalifornie: Nvidia, 2021 [cit. 2021-05-20]. Dostupné z: https://developer.nvidia.com/embedded/jetson-nano-developer-kit.

14. *MaixPy API* [online]. Shenzhen: Seed studio, 2021 [cit. 2021-05-20]. Dostupné z: https://en.maixpy.sipeed.com/maixpy/en/api_reference/.

15. *Sipeed MAIX Bit* [online]. Shenzhen: Seed studio, 2021 [cit. 2021-05-20]. Dostupné z: https://www.seeedstudio.com/Sipeed-MAix-BiT-for-RISC-V-AI-IoT-1-p-2873.html.

16. *Sipeed Maix-II SoM and Devboard for AI/IoT/Machine Learning* [online]. San Francisco: Hackster, 2021 [cit. 2021-05-20]. Dostupné z: https://www.hackster.io/dmitrywat/sipeed-maix-ii-som-and-devboard-for-ai-iot-machine-learning-d2557b.

17. *Sipeed MAIX-II Dock* [online]. Shenzhen: Seed studio, 2021 [cit. 2021-05-20]. Dostupné z: https://www.seeedstudio.com/Sipeed-MAIX-Dock-p-4815.html.

18. CHOLLET, François. *Deep learning with Python* [online]. 1. vyd. Shelter Island, NY: Manning, 2018 [cit. 2021-02-02]. ISBN 978-161-7294-433. Dostupné z: https://www.manning.com/books/deep-learning-with-python.

19. *Oracle* [online]. Austin: Oracle [cit. 2021-05-20]. Dostupné z: https://www.oracle.com/cz/artificial-intelligence/what-is-ai/.

20. CHOLLET, François. *Deep learning v jazyku Python: knihovny Keras, Tensorflow*. 1. vyd. Praha: Grada Publishing, 2019. ISBN 978-80-247-3100-1.

21. KINSLEY, Harrison; KUKIEŁA, Daniel. *Neural Networks from Scratch in Python* [online]. 1. vyd. 2020 [cit. 2021-02-03]. Dostupné z: https://nnfs.io/.

22. *Multi-Layer Neural Networks with Sigmoid Function* [online]. USA: Medium, 2017 [cit. 2021-05-20]. Dostupné z: https://towardsdatascience.com/multi-layer-neural-networks-with-sigmoid-function-deep-learning-for-rookies-2-bf464f09eb7f.

23. *Convolutional neural network* [online]. San Francisco (CA): Wikimedia Foundation, 2001 [cit. 2021-05-20]. Dostupné z: https://en.wikipedia.org/wiki/Convolutional_neural_network.

24. *Image Kernels* [online]. Berkeley, CA: Victor Powell, 2015 [cit. 2021-05-20]. Dostupné z: https://setosa.io/ev/image-kernels/.

25. *Convolution* [online]. Kalifornie: Nvidia [cit. 2021-05-24]. Dostupné z: https://developer.nvidia.com/discover/convolution.

26. *2D Convolution in Image Processing* [online]. Boise: Sneha H.L., 2018 [cit. 2021-05-24]. Dostupné z: https://www.allaboutcircuits.com/technical-articles/two-dimensional-convolution-in-image-processing/.

27. *Convolutional neural network* [online]. Stanford: Stanford, 2021 [cit. 2021-05-24]. Dostupné z: http://cs231n.stanford.edu/slides/2021/lecture_5.pdf.

28. *AI Starter- Build your first Convolution neural network in Keras...* [Online]. USA: Medium, 2019 [cit. 2021-05-20]. Dostupné z: https://pallawi-ds.medium.com/ai-starter-build-your-first-convolution-neural-network-in-keras-from-scratch-to-perform-a059eaa6d4ff.

29. *Convolutional Neural Network | Deep Learning* [online]. Indie: Swapna, 2020 [cit. 2021-05-20]. Dostupné z: https://developersbreach.com/convolution-neural-network-deep-learning/.

30. *A Comprehensive Guide to Convolutional Neural Networks* [online]. USA: Medium, 2018 [cit. 2021-05-24]. Dostupné z: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

31. *Machine Learning & Deep Learning Fundamentals* [online]. Deeplizard, 2018 [cit. 2021-05-24]. Dostupné z: https://deeplizard.com/learn/video/ZjM_XQa5s6s.

32. AL., Pedregosa et. *Underfitting vs. Overfitting* [online]. Scikit-learn, 2014 [cit. 2021-05-20]. Dostupné z: https://scikit-learn.org/0.15/auto_examples/plot_underfitting_overfitting.html.

33. *Data Augmentation | How to use Deep Learning when you have Limited Data* [online]. USA: Medium, 2018 [cit. 2021-05-24]. Dostupné z: https://medium.com/nanonets/how-to-use-deep-learning-when-you-have-limited-data-part-2-data-augmentation-c26971dc8ced.

34. *Enlarge your Dataset* [online]. Brookline: Bharath Raj, 2018 [cit. 2021-05-20]. Dostupné z: https://www.kdnuggets.com/2018/05/data-augmentation-deep-learning-limited-data.html.

35. *ImageNet* [online]. San Francisco (CA): Wikimedia Foundation, 2001 [cit. 2021-05-24]. Dostupné z: https://en.wikipedia.org/wiki/ImageNet.

36. *ImageNet: VGGNet, ResNet, Inception, and Xception with Keras* [online]. Philadelphia: Adrian Rosebrock, 2017 [cit. 2021-05-24]. Dostupné z: https://www.pyimagesearch.com/2017/03/20/imagenet-vggnet-resnet-inception-xception-keras/.

37. *Microsoft COCO: Common Objects in Context* [online]. Ithaca, New York: Arxiv, 2014 [cit. 2021-05-24]. Dostupné z: https://arxiv.org/pdf/1405.0312.pdf.

38. *Microsoft COCO: Common Objects in Context* [online]. COCO, 2015 [cit. 2021-05-20]. Dostupné z: https://cocodataset.org/.

39. *Overview of Open Images V6* [online]. Silicon Valley: Google, 2020 [cit. 2021-05-24]. Dostupné z: https://storage.googleapis.com/openimages/web/index.html.

40. *Platesmania* [online]. Rusko: Platatesmania, 2021 [cit. 2021-05-20]. Dostupné z: http://platesmania.com/cz/nomer16658338.

41. *Car License Plate Detection* [online]. San Francisco: Kaggle, 2020 [cit. 2021-05-24]. Dostupné z: https://www.kaggle.com/andrewmvd/car-plate-detection.

42. KVAPILOVÁ, Aneta. *Pořízení a zpracování sbírky registračních značek vozidel*. Brno, 2019. Diplomová práce. Vysoké učení technické v Brně.

43. *Understanding YOLO* [online]. Colorado: Hackernoon, 2018 [cit. 2021-05-25]. Dostupné z: https://hackernoon.com/understanding-yolo-f5a74bbc7967.

44. *You Only Look Once: Unified, Real-Time Object Detection* [online]. Washington: Pjreddie, 2016 [cit. 2021-05-20]. Dostupné z: https://pjreddie.com/media/files/papers/yolo_1.pdf.

45. *MAP (mean Average Precision) for Object Detection* [online]. USA: Medium, 2018 [cit. 2021-05-25]. Dostupné z: https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173.

46. *YOLO9000: Better, Faster, Stronger* [online]. Washington: Pjreddie, 2017 [cit. 2021-05-20]. Dostupné z: https://pjreddie.com/media/files/papers/YOLO9000.pdf.

47. *Framework* [online]. San Francisco (CA): Wikimedia Foundation, 2001 [cit. 2021-05-24]. Dostupné z: https://cs.wikipedia.org/wiki/Framework.

48. REDMON, Joseph. *Darknet: Open Source Neural Networks in C* [online]. 2016 [cit. 2021-05-20]. Dostupné z: http://pjreddie.com/darknet/.

49. *Wartburg* [online]. Berlin: Pixabay, 2017 [cit. 2021-05-20]. Dostupné z: https://pixabay.com/photos/wartburg-toilet-paper-reflector-2456243/.

50. *Gradient-Paperspace* [online]. Brooklyn: Paperspace, 2021 [cit. 2021-05-24]. Dostupné z: https://gradient.paperspace.com/.

51. *Plate detect and recognize* [online]. San Francisco: GitHub, 2020 [cit. 2021-05-25]. Dostupné z: https://github.com/quangnhat185/Plate_detect_and_recognize/blob/master/License_character_recognition_weight.h5.

SEZNAM OBRÁZKŮ

SEZNAM TABULEK