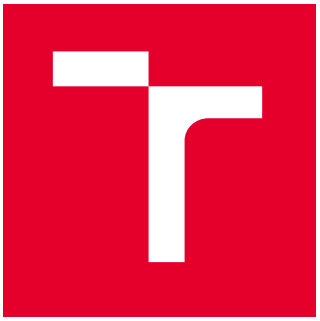


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ

FACULTY OF MECHANICAL ENGINEERING

ÚSTAV AUTOMOBILNÍHO A DOPRAVNÍHO INŽENÝRSTVÍ

INSTITUTE OF AUTOMOTIVE ENGINEERING

DETEKCE DOPRAVNÍCH ZNAČEK PRO AUTONOMNÍ VOZIDLA

DETECTION OF TRAFFIC SIGNS FOR AUTONOMOUS VEHICLES

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Lucie Kovaříková**
AUTHOR

VEDOUCÍ PRÁCE — **doc. Ing. Pavel Kučera, Ph.D.**
SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání diplomové práce

Ústav:	Ústav automobilního a dopravního inženýrství
Studentka:	**Bc. Lucie Kovaříková**
Studijní program:	Automobilní a dopravní inženýrství
Studijní obor:	bez specializace
Vedoucí práce:	**doc. Ing. Pavel Kučera, Ph.D.**
Akademický rok:	2022/23

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma diplomové práce:

Detekce dopravních značek pro autonomní vozidla

Stručná charakteristika problematiky úkolu:

Naprogramování řídicího algoritmu pro detekci dopravního značení z kamer pro autonomní řízení vozidla využitím softwaru Simulink, nebo dalších programovacích jazyků C, C++, C#, atd. Konkrétně se jedná o detekci dopravních značek, semaforů, vodorovného značení, železničních přejezdů a tím určovat vstupní informace pro autonomní vozidlo. Experimentální ověření správné detekce dopravního značení.

Cíle diplomové práce:

Rešerše o detekci dopravního značení.
Naprogramování základní komunikace s kamerou.
Naprogramování algoritmu pro detekci.
Experimentální ověření správné detekce.

Seznam doporučené literatury:

HANSEN, John H. L. Digital signal processing for in-vehicle systems and safety. 1. London: Springer, 2012. ISBN 978-1441996060.

SOLOMON, Chris a Toby BRECKON. Fundamentals of digital image processing: a practical approach with examples in Matlab. 1. Hoboken, NJ: Wiley-Blackwell, 2011. ISBN 978-0470844731.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2022/23

V Brně, dne

L. S.

prof. Ing. Josef Štětina, Ph.D. ředitel ústavu	doc. Ing. Jiří Hlinka, Ph.D. děkan fakulty

ABSTRAKT

Tato diplomová práce se zabývá detekcí dopravních značek pro autonomní vozidla s využitím programovacího jazyka Python a architektury konvoluční neuronové sítě YOLOv7. Cílem je rešerše v oblasti rozpoznávání dopravního značení a naprogramování algoritmů pro komunikaci s kamerou a detekci. Výsledkem je experimentální ověření detekce a jeho vyhodnocení.

KLÍČOVÁ SLOVA

detekce dopravního značení, autonomní vozidla, Python, YOLOv7, konvoluční neuronové sítě

ABSTRACT

This master's thesis focuses on traffic sign detection for autonomous vehicles using the Python programming language and the YOLOv7 architecture of convolutional neural network. The objective is to conduct research in the field of traffic sign recognition and implement algorithms for camera communication and detection. The results include experimental verification of the detection system and its subsequent evaluation.

KEYWORDS

traffic sign detection, autonomous vehicles, Python, YOLOv7, convolutional neural networks

BIBLIOGRAFICKÁ CITACE

Kovaříková, L. Detekce dopravních značek pro autonomní vozidla. Brno, 2023. Závěrečná práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav automobilního a dopravního inženýrství. Vedoucí závěrečné práce Pavel Kučera.

ČESTNÉ PROHLÁŠENÍ

Prohlašuji, že tato práce je mým původním dílem, zpracovala jsem ji samostatně pod vedením doc. Ing. Pavlem Kučerou, Ph.D a s použitím informačních zdrojů uvedených v seznamu.

V Brně dne 26. května 2023 ……..……..……..

Lucie Kovaříková

PODĚKOVÁNÍ

Chtěla bych poděkovat vedoucímu diplomové práce doc. Ing. Pavlu Kučerovi, Ph.D. za praktické rady při zpracování této práce a za zapůjčení kamery pro experimentální ověření. Děkuji také mému bratru a nejbližším z mého okolí za podporu během celého studia.

OBSAH

ÚVOD

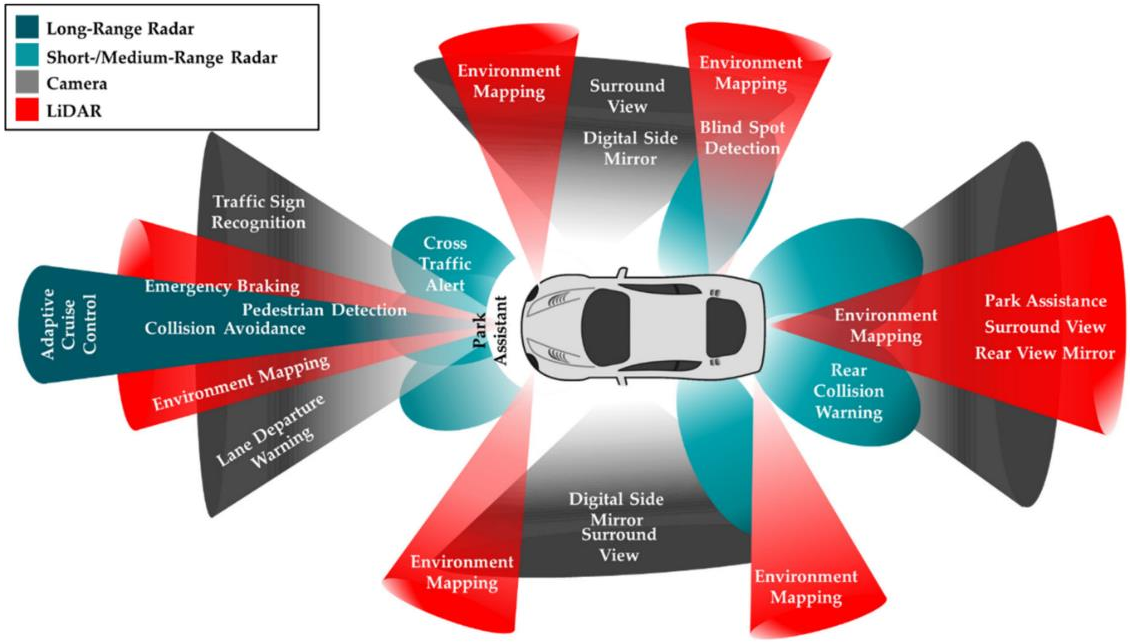
Automobilový průmysl je zejména díky vývoji nových technologií transformován více než kdy jindy. Autonomní vozidla představují inovativní technologii s potenciálem zásadně změnit dopravní systémy a způsob, jakým se pohybujeme po silnicích. Jedním z klíčových aspektů autonomního řízení je schopnost vozidel detekovat a rozpoznávat objekty v jejich okolí, včetně dopravního značení. Detekce dopravního značení je nezbytným prvkem pro bezpečnou a spolehlivou navigaci autonomních vozidel.

Autonomní řízení je reálné především díky pokročilým technikám umělé inteligence. Vyvinuté algoritmy se během trénování na velkých datových sadách učí rozpoznávat specifické rysy a vzory v datech, které odpovídají jednotlivým typům dopravního značení. V této práci bude zpracována rešerše různých přístupů k detekci dopravního značení. Bude zde také popsán samotný proces trénování neuronové sítě včetně metod, které se využívají k hodnocení schopnosti detekovat naučené rysy.

Na základě těchto poznatků bude provedena implementace a optimalizace modelu detekce založeném na architektuře YOLOv7. Dále bude navržen algoritmus pro komunikaci s kamerou a algoritmus pro detekci. Tyto algoritmy budou využity k experimentálnímu ověření, v rámci kterého bude provedeno testování a vyhodnocení modelu na reálných videích získaných z kamery. Výsledkem práce bude zhodnocení přesnosti a spolehlivosti detekce dopravního značení a případné návrhy na další vylepšení a rozšíření modelu.

1 AUTONOMNÍ VOZIDLA A JEJICH SENZORY PRO DETEKCI OKOLNÍHO PROSTŘEDÍ

Autonomní vozidlo je vozidlo, které je schopné plnit úkoly řízení vozidla bez lidského řidiče. Tato vozidla jsou vybavena různými senzory, jako jsou radarové senzory, lidarové senzory, kamery a GPS, aby mohla být zachycena široká škála informací o okolním prostředí. Tyto informace jsou pak analyzovány pomocí algoritmů a umělé inteligence, které rozhodují, jakým způsobem se vozidlo má pohybovat. [1] V této kapitole budou popsány úrovně autonomního řízení a následně senzory, které jsou v autonomních vozidlech používány ke snímání okolního prostředí.



Obr. 1 Příklad typu senzorů a jejich umístění na autonomním vozidle [2]

1.1 ÚROVNĚ AUTONOMNÍHO ŘÍZENÍ

Existuje několik úrovní autonomie vozidel, které jsou definovány podle SAE International. Na nejnižší úrovni autonomie vozidlo vyžaduje neustálé lidské řízení, zatímco na nejvyšší úrovni je vozidlo schopné řídit plně autonomně bez jakékoliv lidské interakce. [3] Tyto hlavní úrovně lze pak ještě dále rozdělit na dvě kategorie dělené podle sledování okolního prostředí. Pokud okolní prostředí sleduje řidič, jedná se o úrovně 0, 1 a 2. Druhá kategorie zahrnující zbylé úrovně platí pro případy, kdy okolní prostředí sleduje systém. [4]

- **Úroveň 0:** Žádná automatizace řízení. Vozidlo je řízeno plně člověkem, bez jakéhokoliv podpůrného zařízení.
- **Úroveň 1**: Asistované řízení. Vozidlo obsahuje omezené automatické řídící funkce, jako je např. regulace rychlosti nebo udržování jízdního pruhu, ale řízení vozidla stále zůstává plně v rukou řidiče.
- **Úroveň 2:** Částečně automatizované řízení. Vozidlo je schopné řízení v určitých situacích, jako např. na dálnici, ale řidič musí být stále připraven převzít řízení kdykoliv, kdy je to potřeba. Tato úroveň v současné době patří mezi nejrozšířenější úrovně autonomie.

- **Úroveň 3:** Podmíněně automatizované řízení. Vozidlo je schopné řídit plně autonomně ve vymezených podmínkách, jako např. na dálnici, a řidič nemusí být stále připraven převzít řízení. Nicméně, řidič musí být stále schopen převzít řízení, když je to potřeba.
- **Úroveň 4:** Vysoce automatizované řízení. Vozidlo je schopné řídit plně autonomně ve všech situacích vymezených jeho operačním prostředím. Pokud se vozidlo dostane do situace, kdy se nemůže řídit, provede bezpečné zastavení.
- **Úroveň 5:** Plně automatizované řízení. Vozidlo je plně autonomní a nevyžaduje žádnou lidskou interakci při řízení. [3]

Souhrn těchto bodů je lépe ukázán v tabulce 1. Objevuje se zde pojem „dynamický úkol řízení“. Tento úkol zahrnuje provozní úkoly jako je řízení, brždění, zrychlování a taktické reakce na události, včetně určování změny jízdního pruhu a odbočování. [4]

Tab. 1 Úrovně automatizovaného řízení: Shrnutí normy SAE [4]

Úroveň autonomie (SAE)	Typ automatizovaného řízení	Dynamický úkol řízení	Monitoring okolního prostředí	Zásah v případě potřeby
0	Žádné	Člověk	Člověk	Člověk
1	Asistované	Člověk a systém	Člověk	Člověk
2	Částečné	Systém	Člověk	Člověk
3	Podmíněné	Systém	Systém	Člověk
4	Vysoké	Systém	Systém	Systém
5	Plné	Systém	Systém	Systém

1.2 KAMERY

Kamery jsou nedílnou součástí autonomních vozidel. Díky nim je možno získat dostatečné množství informací o prostředí spolu s barvami a texturami objektů. Výhodou je nízká pořizovací cena, nicméně je potřeba zmínit i rizika nepřesnosti snímání dat. Problémy může způsobovat přímé slunce, déšť, mlha a další meteorologické jevy. [5]

Kamerové systémy autonomních vozidel využívají buď monokulární nebo stereo (binokulární) kamery, případně kombinaci obojího. Dále jsou využívány tzv. fish eye kamery, které za využití 4 kamer umístěných na autě umožňují snímání v 360°. Nevýhodou monokulárních kamer je to, že nejsou schopny zaznamenat informaci o hloubce objektu, většinou jsou proto instalovány dvě kamery na každý bok automobilu. Tento problém řeší stereo kamery, které mají dva obrazové senzory. Díky jejich stejné srovnávací základně můžeme porovnávat vzdálenost objektů na obou obrazech.

Data z těchto kamer se převážně používají pro detekci dopravních značek, semaforů a jízdních pruhů. Jak již bylo zmíněno, jelikož kamery mají kvůli problematickým podmínkám potíže zaznamenat kvalitní data, jsou často kombinovány s daty z ostatních senzorů, jako je například radar nebo LiDAR. [2]

1.3 RADAR

Radarové senzory měří vzdálenost překážek, zajišťují jejich precizní vnímání ve dne i v noci, nehledě na okolní podmínky viditelnosti. Na základě Dopplerova jevu elektromagnetických vln určují relativní rychlost a relativní pozici detekované překážky. V automobilovém průmyslu se nejvíce využívá radaru, který operuje na frekvenci 76 až 81 GHz. [6] Tyto senzory jsou umístěny na různých místech vozidla, nejčastěji za předním nárazníkem, na střeše nebo za logem výrobce automobilu. K předejití chybné detekce nebo pozdního zjištění překážky musí být radary umístěny přesně do montážních poloh již ve výrobě, zamezí se tak úhlové nepřesnosti při měření. V poslední době výrobci automobilních radarů začali přidávat vertikální antény, které slouží k odlišení objektů nad silnicí (jako jsou například dopravní značky), od objektů, které jsou na silnici. V závislosti na požadovaném rozsahu se používají různé druhy antén. Nové druhy radarů dokonce mohou přepínat mezi různými konfiguracemi antény. [2]

Obr. 2 Radary od společnosti Bosch [7]

1.4 LIDAR

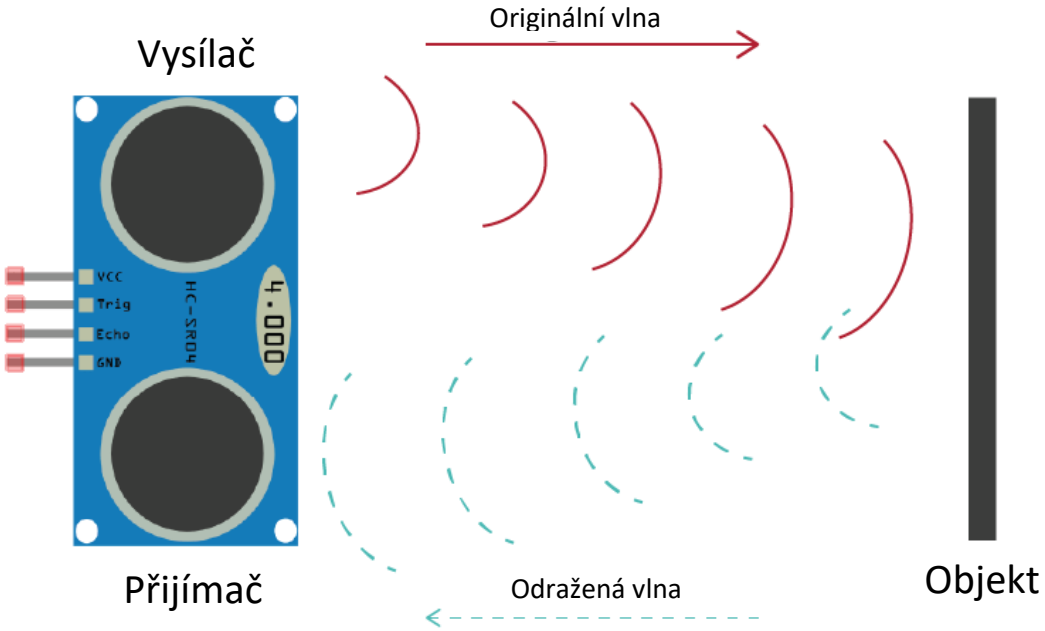
LiDAR (Light Detection And Ranging) je využíván pro celkové vnímání prostředí okolo vozidla. [8] Jeho funkce spočívá v tom, že vysílá pulzy infračervených paprsků nebo laserového světla, které se následně odráží od okolních objektů. Změřením intervalu mezi vysláním světelného paprsku a zpětného přijetí odraženého paprsku je následně vypočtena vzdálenost překážky. Výstupem může být 1D, 2D nebo 3D sken, který je zobrazen jako mračno bodů.

Obr. 3 Výstupní sken z 3D LiDARu [9]

Z hlediska záběru snímání zorného pole se LiDARy dále ještě dělí na mechanické nebo polovodičové (SSL). Mechanický LiDAR může dosáhnout horizontálního záběru 360°, využívá totiž vysoce kvalitní optiku a rotační čočky, které jsou nasměrovány ke snímání požadovaného pole pomocí elektromotoru. Oproti tomu SSL směřuje laserové paprsky pomocí množství mikro strukturovaných vlnovodů. Jejich hlavní výhodou je větší mechanická spolehlivost a nižší náklady na cenu, nicméně to vše na úkor horizontálního záběru, který je obvykle od 120° a méně. [2]

1.5 ULTRASONICKÉ SENZORY

Nejlevnější variantou ze všech dříve zmíněných senzorů jsou senzory ultrasonické. Senzory ke své funkci využívají inverzní piezoelektrický jev. Napětím vznikají oscilace, které zesílením přes kovovou membránu generují zvuk v rozsahu 40-50 kHz. Odražené vlny se pak vrací zpět na membránu, která svým rozkmitáním vytváří elektrické napětí, které je měřeno snímačem. [10] Tyto senzory jsou nejčastěji voleny pro aplikace s malým dosahem, tedy například pro parkovací senzory, na které je taky většina automobilových výrobců používá. O jejich využití k řízení autonomních vozidel se zatím vzhledem k bezpečnosti neuvažuje, a to převážně z hlediska toho, že mohou být ovlivněny akustickým rušením a kvůli jejich citlivosti na změny okolních podmínek prostředí, které mohou výrazně ovlivnit výkonnost snímače. [11] V práci jsou uvedeny pro doplnění přehledu senzorů, které využívají dnešní vozidla.



Obr. 4 Funkční schéma ultrasonického senzoru [12]

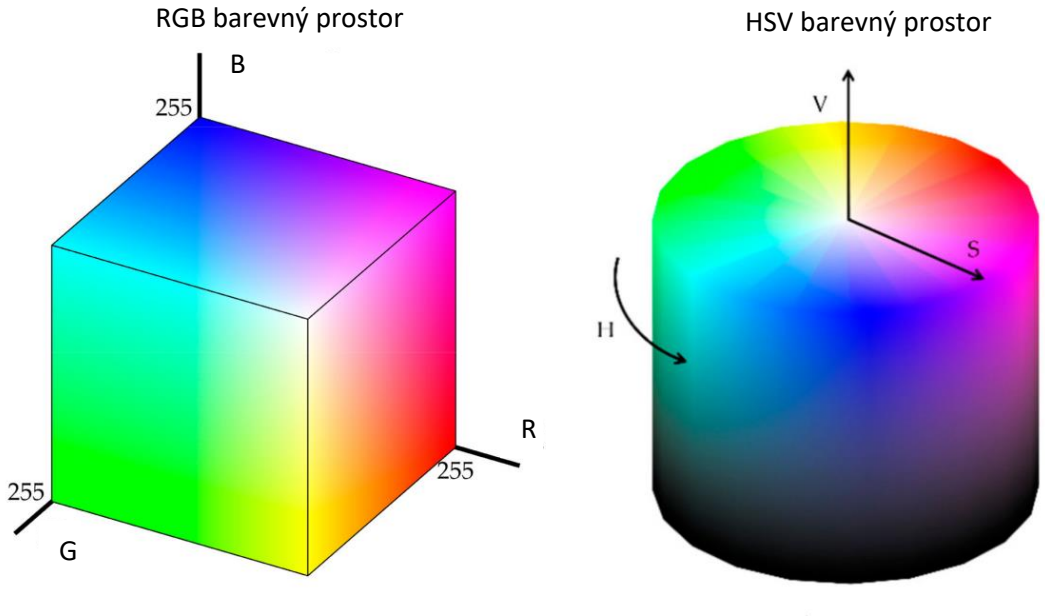
2 DETEKCE DOPRAVNÍHO ZNAČENÍ

Dopravní značení je všude kolem nás, hraje zásadní roli v určování předností na křižovatkách, udávání zákazů a omezení. A přesně tak, jak rozpoznává řidič toto značení, je třeba aby jej rozpoznávala i autonomní vozidla. Hlavním zdrojem vstupních dat pro následnou detekci jsou snímky z kamery. Od detekce založené na základě barev a tvarů se kvalita a rychlost rozpoznávání s dalším výzkumem výrazně zvýšily. Vývoj a rozvoj v oblasti hlubokého učení přinesl do detekce a rozpoznávání objektů nový směr. V této kapitole je zpracována rešerše o aktuálních přístupech k detekci dopravního značení.

2.1 DETEKCE NA ZÁKLADĚ BAREV A TVARU

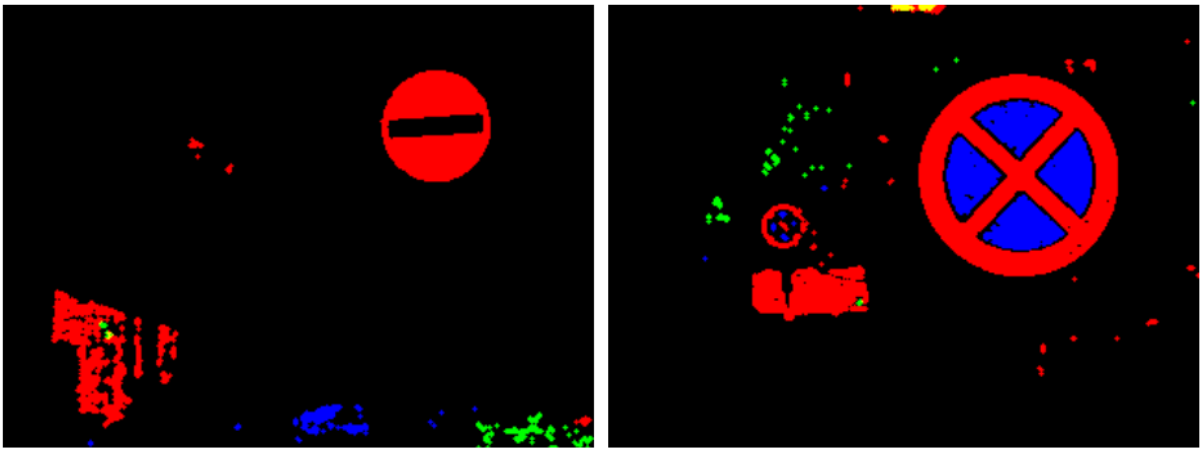
V počátcích detekce dopravního značení se algoritmy založené na detekci podle barev a na detekci podle tvaru používaly odděleně. V průběhu let se však za účelem dosažení rychlejších a přesnějších výsledků začala používat jejich kombinace pro určení zájmové oblasti a následné klasifikace. [13]

Nejčastěji jsou barvy na displejích nebo fotografiích reprezentovány pomocí RGB barevné škály. Jelikož dopravní značky většinou obsahují opakující se barvy, můžeme je na základě této skutečnosti detekovat. Barevný prostor RGB však není vhodný pro segmentaci z důvodu vysoké korelace mezi barevnými složkami. Tento problém způsobuje kolísání intenzity okolního světla, které pak posouvá shluk barev směrem k bílým nebo černým rohům, což výrazně ovlivňuje složky RGB. Proto se využívají různá barevná spektra, například HSV barevné spektrum. [14]

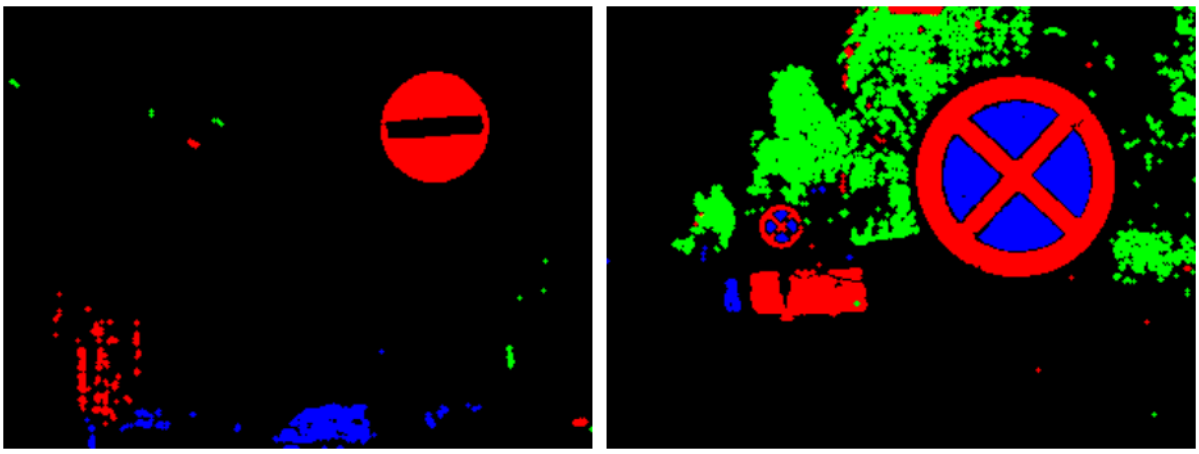


Obr. 5 RGB a HSV barevné spektrum [15]

Nutno podotknout, že existují i algoritmy, které k detekci používají RGB barevnou škálu. Harshavardhan a spol. dosáhli s navrženým algoritmem 90% úspěšnosti. [16] Aby se potvrdilo, že se jedná opravdu o značku, užívá se algoritmu, který analyzuje tvar značky a její polohu na snímku. Při detekci dopravního značení jsou využívány hlavně tvary kruhu, trojúhelníku a obdélníku. Tyto tvary se pak analyzují pomocí několika metod. Metody založené na histogramech, metody shlukování, metody detekce hran, metody duálního růstu nebo metody duálního prahování. Z hlediska rychlosti a účinnosti detekce dopravních značek je metoda prahování nejefektivnější. Kombinují-li se obě metody (analýzy barvy a tvaru), nejprve se použije filtr na základě barev, který vyhledá objekty, jejichž barvy odpovídají typickým barvám pro dopravní značky a následně se pak tyto objekty tvarově analyzují. [17]



Obr. 6 Výsledek segmentace v RGB barevném spektru. [17]

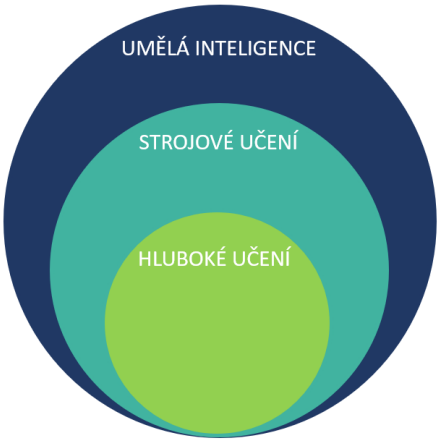


Obr. 7 Výsledek segmentace v HSV barevném spektru. [17]

Pro zrychlení detekce dopravního značení se tyto metody nejčastěji kombinují s detekcí založenou na hlubokém učení. Příkladem toho je práce autorů Lu, Gozdzikiewicz, Chang a Ciou [18], kteří použili dvoufázovou metodu, která nejprve detekuje tvar dopravní značky pomocí Houghovy transformace, který se pak následně klasifikuje využitím konvoluční neuronové sítě. Ve druhé fázi detekce konvoluční neuronové sítě používají také autoři článku Dvoustupňová detekce a rozpoznávání dopravních značek na základě SVM a konvolučních neuronových sítí [19], kde nejprve značky dělí podle kruhového nebo trojúhelníkového tvaru pomocí HOG rysů a lineárních SVM.

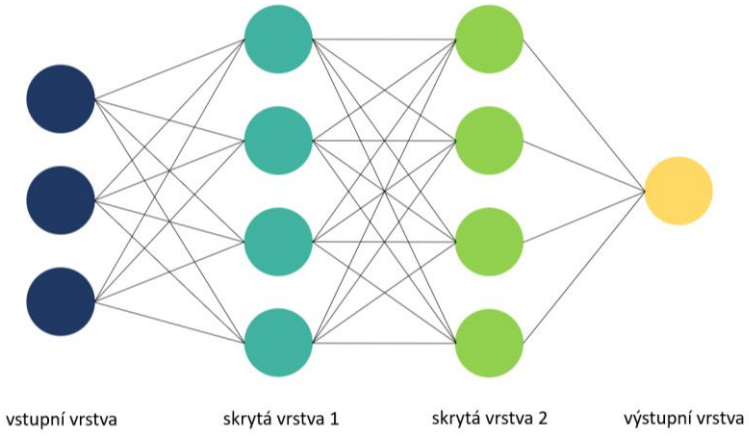
2.2 DETEKCE ZALOŽENÁ NA HLUBOKÉM UČENÍ

Hluboké učení, z anglického deep learning, je součástí umělé inteligence a strojového učení, které se zaměřuje na tvorbu komplexních modelů, které se mohou učit z velkého množství dat. Využívá se k řešení různých úkolů, například překladu textů nebo rozpoznávání obrazů. [20] Často se hluboké učení pojmenovává jako umělá inteligence nebo strojové učení, nicméně je nutno zdůraznit, že se nejedná o synonyma. Vzájemný vztah těchto termínů je zobrazen na *Obr. 8*. Umělá inteligence je definována jako schopnost strojů napodobovat lidské schopnosti. Strojové učení je podmnožinou umělé inteligence, jejíž aplikace umožňuje systému automatické učení a zlepšování na základě zkušeností. [21]



Obr. 8 Vztah mezi umělou inteligencí, strojovým učením a hlubokým učením [22]

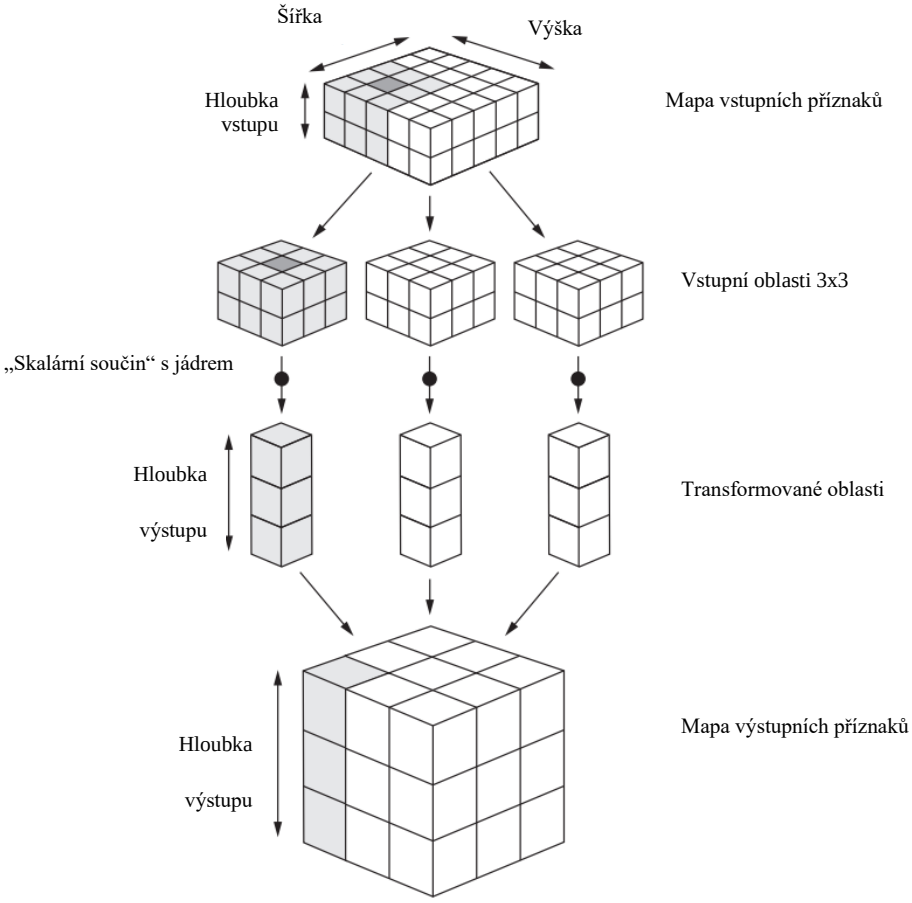
Dříve se k detekci používala kombinace umělé inteligence a strojového učení. Samotné hluboké učení k extrakci objektů používá konvoluční neuronovou síť, která simuluje zpracování informací v lidském mozku, díky čemuž dosahuje lepší přesnosti a robustnosti. [23] Neuronová síť je matematický model, který se skládá z neuronů propojených větvemi. Tyto větve přijímají vstupy, zpracovávají je a vytvářejí výstupy. Jaký výstup bude generován na základě vstupu, je určeno vstupní váhou a biasem na neuronech. Tyto parametry jsou podrobněji vysvětleny v kapitole 3.1. Výstup je také ovlivněn aktivační funkcí.



Obr. 9 Architektura neuronové sítě [24]

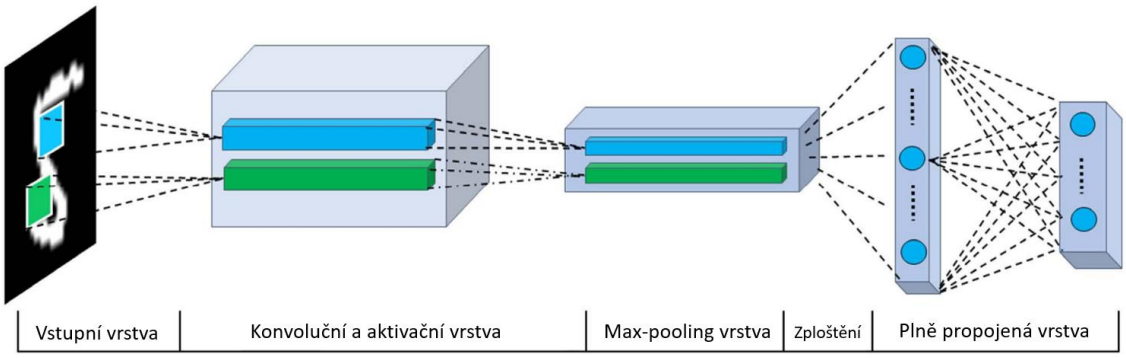
Aktivační funkce je matematická funkce, která se používá v neuronových sítích pro převod vážené sumy vstupů na výstupní hodnotu neuronu. Účelem aktivační funkce je přidání nelinearity do modelu. Bez aktivační funkce by neuronová síť byla pouze lineárním modelem, schopným aproximovat pouze lineární vztahy mezi vstupy a výstupy. Mezi nejznámější aktivační funkce patří Sigmoid nebo ReLU. [24]

Pro zpracování obrazových dat se nejvíce používají **konvoluční neuronové sítě.** [25] Na vstupu této sítě je obrázek, který je reprezentován tenzorem třetího řádu se dvěma osami vyjadřujícími šířku a výšku obrázku. Dále osou hloubky, která vyjadřuje tzv. kanály barevnosti obrázku. Barevné RGB obrázky mají tyto kanály tři. Černobílé obrázky pouze jeden, protože se dají vyjádřit v odstínech šedé. [22] Dále je umístěna skupina detektorů, kterou procházejí data ke zpracování. Tato skupina detektorů se nazývá konvoluční vrstva, která provádí operaci zvanou **konvoluce**. V konvoluční vrstvě jsou konvoluční filtry, které mají nastaveny hodnoty náhodně nebo pomocí předem naučených vah. Tyto hodnoty jsou v průběhu učení neuronové sítě neustále upravovány. Učení filtru v průběhu trénování umožňuje adaptaci na specifické vlastnosti datové sady a optimalizaci příznaků, které jsou relevantní pro danou úlohu. Tím se zajišťuje, že konvoluční vrstvy sítě jsou schopny efektivně extrahovat relevantní informace ze vstupních dat. Výstupem konvoluce je matice příznaků, která určuje, kde se v původním obrázku nacházejí dominantní části, které jsou obsaženy ve filtru. Tato operace se opakuje pro každý filtr, čímž vzniká mapa příznaků z celého obrázku. [26] Celý proces konvoluce je znázorněn na *Obr. 10*.



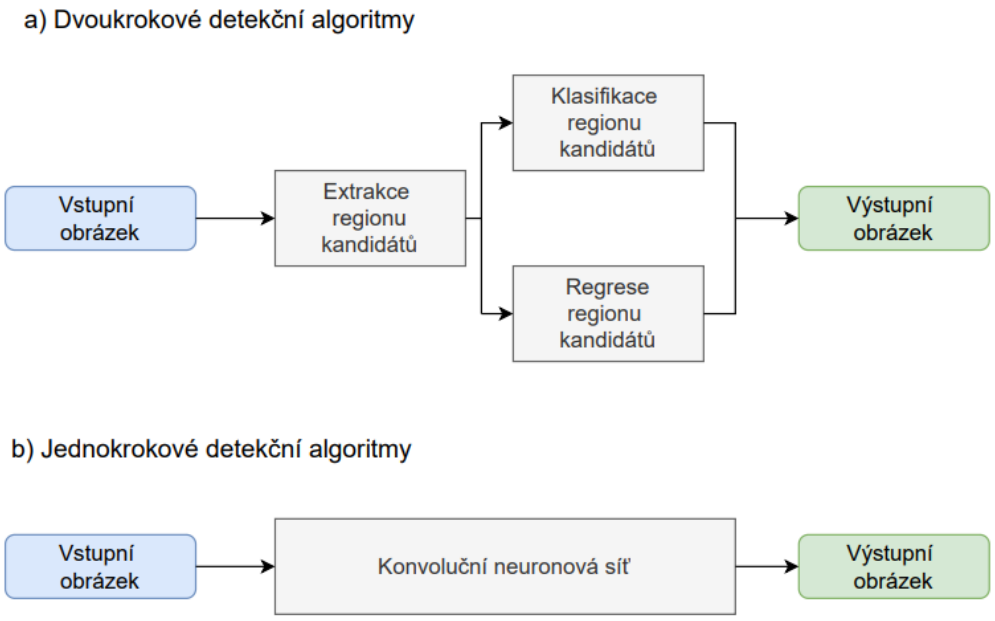
Obr. 10 Proces konvoluce [22]

Následně je použita agregační vrstva, tzv. pooling vrstva. Existují různé druhy této vrstvy, jako je například average nebo min pooling. Nejčastěji je používána max pooling, která zmenšuje počet parametrů sítě. Zároveň je ale zachován klíčový aspekt potřebný k dokončení úlohy. Během max poolingu je vstupní mapa příznaků dělena do diskrétních regionů, obvykle pomocí čtvercového okna, a z každého regionu je vybírána největší hodnota (maximální) jako výstupní hodnota pro daný region. Tím dochází ke zmenšení rozměrů mapy příznaků. [26]. Následně je datová 2D mapa zploštěná tzv. flat vrstvou na 1D mapu, které je vstupem pro plně propojenou vrstvu. [27]



Obr. 11 Architektura konvolučních neuronových sítí [28]

Detekce založená na hlubokém učení se dělí do dvou hlavních kategorií. Jedná se o dvoukrokové algoritmy a jednokrokové algoritmy.



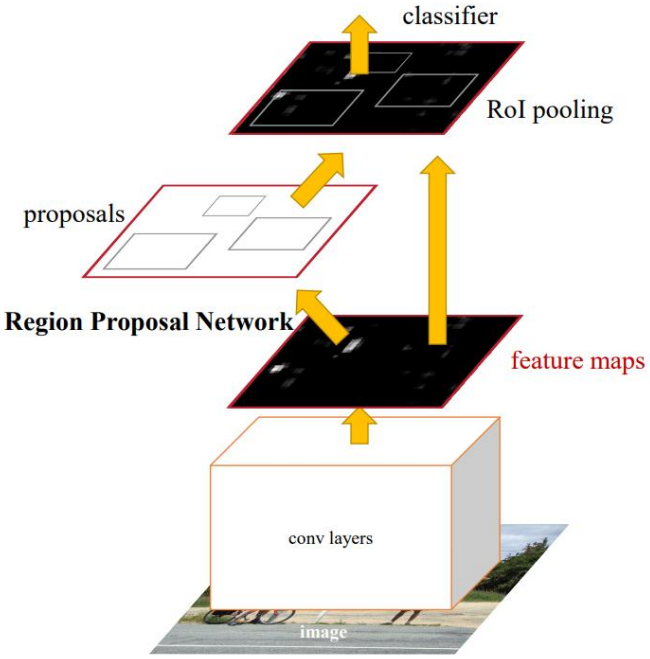
Obr. 12 Hlavní principy dvoukrokových a jednokrokových algoritmů pro detekci [29]

2.2.1 DVOUKROKOVÝ PŘÍSTUP

Hlavní myšlenkou dvoukrokového přístupu je návrh velkého počtu regionů kandidátů pro vstupní obraz, poté je provedena regresní lokalizace a klasifikace na každém z těchto regionů. [23] Z principu její funkce se tato metoda v jiných článcích nazývá jako technika založená na návrhu regionu. Návrh regionu je oblast, ve které je vyšší pravděpodobnost, že se v něm nachází potenciální objekt.

Mezi nejznámější techniky založené na tomto principu patří R-CNN, Fast R-CNN nebo Faster R-CNN. Nejrychlejší a nejpřesnější metodou je Faster R-CNN, a to hlavně z důvodu využití sítě RPN a sítě Fast R-CNN, díky čemuž Faster R-CNN generuje méně oblastí návrhu. V porovnání s R-CNN nevyžaduje tak vysokou časovou a prostorovou složitost, která u R-CNN vzniká právě generováním velkého množství návrhů regionů.

Rozšířením Faster R-CNN vznikla Mask R-CNN, která zlepšuje výkonnost systému při detekci malých objektů, což bývá mnohdy problém. V prvním kroku podobně jako Faster R-CNN využívá RPN síť, ve druhém kombinuje základní síť s FPN sítí. FPN extrahuje základní rysy z nižších vrstev sítě ještě předtím, než dojde k odstranění důležitých detailů při převzorkování na malých objektech. [30]



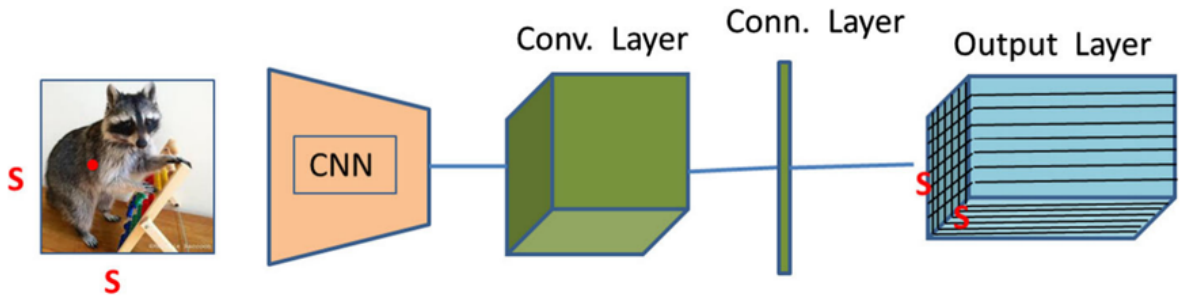
Obr. 13 Architektura Faster R-CNN [31]

Tyto architektury se při detekci dopravního značení kombinují s různými dalšími metodami. Například Qian a spol. využili kombinaci hybridního návrhu regionu, která uvažuje komplementární informace o barvě a hraně v kombinaci Fast R-CNN pro detekci vodorovného dopravního značení. S tímto modelem dosáhli celkové průměrné přesnosti 85,58 %. [32] Mask R-CNN metodu ve své práci zabývající se detekcí dopravního značení využili Tabernik a Skočaj. Zabývali se problémem učení a detekce na vlastní datové sadě, která obsahuje 200 tříd dopravního značení. Jejich úpravami v Mask R-CNN zlepšili míru chybovosti sítě RPN pro malé objekty a schopnost detekce velkých i malých objektů. Dále také došlo ke zlepšení celkového výkonu z hlediska průměrné přesnosti. [33]

2.2.2 JEDNOKROKOVÝ PŘÍSTUP

Ačkoliv dvoukrokové metody dosahují vysoké přesnosti, jejich výpočetní rychlost je nízká, což může komplikovat detekci v reálném čase. Řešení tohoto problému nabízí jednokrokové algoritmy, které provádí v jednom kroku klasifikaci i regresní lokalizaci objektu. Schematicky lze porovnání obou principů vidět na *Obr. 12*. Typickým jednokrokovým algoritmem je například YOLO. [23] První verze architektury YOLO byla představena v roce 2016. [34] Od té doby je stále vyvíjena, nejnovější verzí je momentálně YOLOv8 od Ultralytics, kteří také vytvořili verzi YOLO 3 a 5. [35] Dále mezi tyto algoritmy patří také SSD, RetinaNet, RefineNet a daší. [36]

YOLO k detekci využívá detektor s konvolučním filtrem, díky čemuž je oproti dvoukrokovým metodám rychlejší. Na celý obraz, který je rozdělen na pevné oblasti, se používá pouze jedna neuronová síť. V každé oblasti je vypočtena pravděpodobnost a ohraničení objektu. Toho je dosaženo na základě konvolučních vrstev, které využívají systém mřížek S X S. Architektura YOLO je zobrazena na *Obr. 14*.

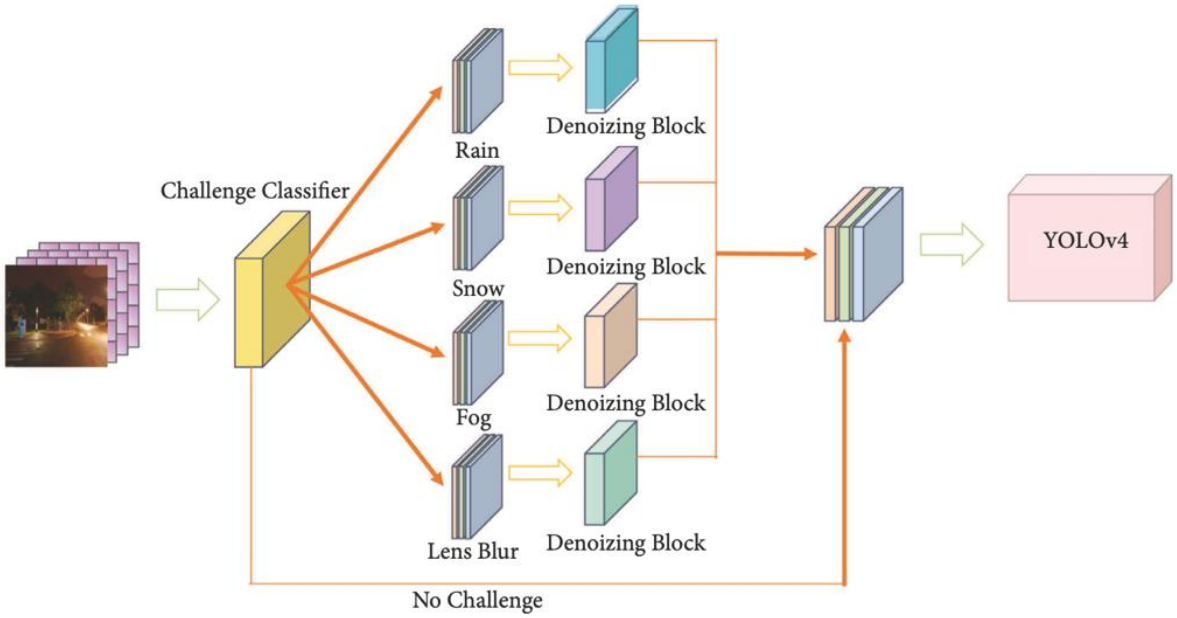


Obr. 14 Architektura sítě YOLO – první verze [37]

YOLO je značně využívaným algoritmem při řešení problémů detekce dopravního značení. I když jsou stále představovány nové verze, existují aplikace, ve kterých se používají i starší verze tohoto algoritmu. Mnoho prací porovnává různé verze mezi sebou, obzvlášť z hlediska rychlosti a přesnosti detekce. Tak jako jsou upravovány dvoukrokové metody, bývá upravována a vylepšována i základní architektura YOLO, většinou z důvodu zlepšení detekce v případě menších objektů nebo komplexního okolního prostředí (mlha, déšť, sníh, atd).

Zhongli provedl úpravy na algoritmu YOLOv3 s cílem dosáhnout přesnější detekce vzdálenějších objektů. Výsledný test byl proveden na 3 fotografiích o různých rozlišeních. Oproti originálnímu algoritmu YOLOv3 se přesnost rozpoznávání zlepšila o 8,1 %, 5,9 % a 4,6 %. Při testování na větší datové sadě se zejména zlepšila míra rozpoznávání malých a vzdálených značek a také značek, které byly v malém rozsahu zakryty. [38]

YOLO bylo také využito v práci Wanga a spol., kteří upravili algoritmus YOLOv4 za cílem dosažení lepší detekce ve zhoršených podmínkách počasí. Nejprve je využit modul předzpracování obrazu (viz *Obr. 15*). Na základě klasifikátoru je rozpoznáno, zda se jedná o déšť, sníh, mlhu nebo rozmazání obrazu. Podle určené třídy je pak použit specifický blok, který použije příslušný algoritmus pro odstranění šumu v obraze. Výsledný zpracovaný obraz se pak dále zpracovává pomocí upraveného YOLOv4. [39]



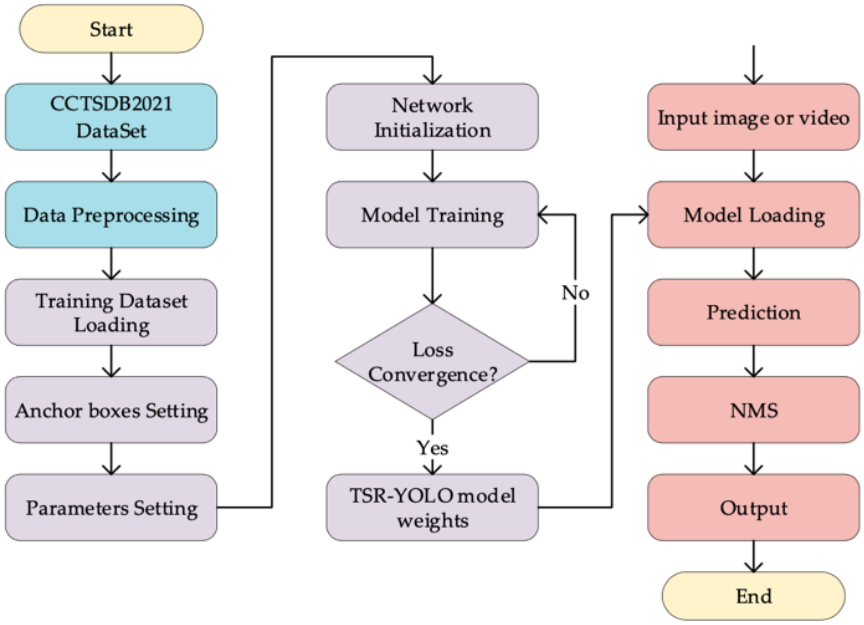
Obr. 15 Struktura modulu předzpracování obrazu [39]

Výsledná metoda pak byla testována a porovnávána s dalšími metodami. Mezi tyto metody patřila i dvoukroková metoda Faster R-CNN. Při normálních podmínkách dobré viditelnosti průměrná střední přesnost navrhované metody dosáhla 87,19 %, u metody Faster R-CNN 88,86 %. V případě potřeby využití aplikace pro mobilní zařízení je tento výsledek pořád obstojný, vzhledem k tomu, že metoda Faster R-CNN je výpočtově náročnější. [39] Více výsledků z tohoto porovnání lze vidět v *Tab. 2*.

Tab. 2 Porovnání průměrné střední přesnosti navrhované metody s dalšími metodami [39]

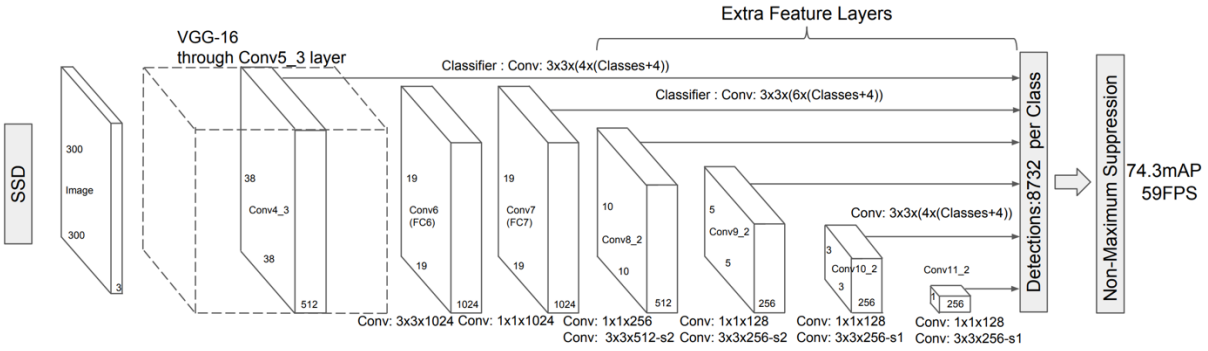
Model	$mAP_{NoChallenge}$	mAP_{Rain}	mAP_{Snow}	mAP_{Fog}	$mAP_{LensBlur}$	mAP	FPS
Faster-RCNN	88.86	70.51	71.07	69.28	73.05	82.59	24
SSD	78.71	61.54	63.56	60.97	64.58	72.12	51
RetinaNet	78.07	59.55	60.91	63.89	64.76	71.84	53
YOLOv3	80.18	63.94	63.83	62.81	66.25	74.32	58
YOLOv4-tiny	79.11	61.57	61.45	60.32	63.59	72.03	142
YOLOv4	83.92	66.62	67.14	65.59	69.26	77.25	86
Ours	87.19	73.22	74.02	72.50	76.41	81.78	74

Dalším příkladem využití YOLOv4 je práce Songa a Suandi [40], navrhující úpravu algoritmu YOLOv4-tiny s cílem zvýšit bezpečnostní rizika spojená s reálnou aplikací rozpoznávání dopravních značek, jako jsou již dříve zmíněné proměnné – intenzita světla, extrémní počasí ale také vzdálenost značky od kamery. Při snímkové frekvenci 81 FPS vytvořený algoritmus TSR-YOLO dokázal detekovat dopravní značky s přesností 96,62 % a průměrnou střední přesností 92,77 %.



Obr. 16 Proces YOLO-TSR při detekci dopravního značení [40]

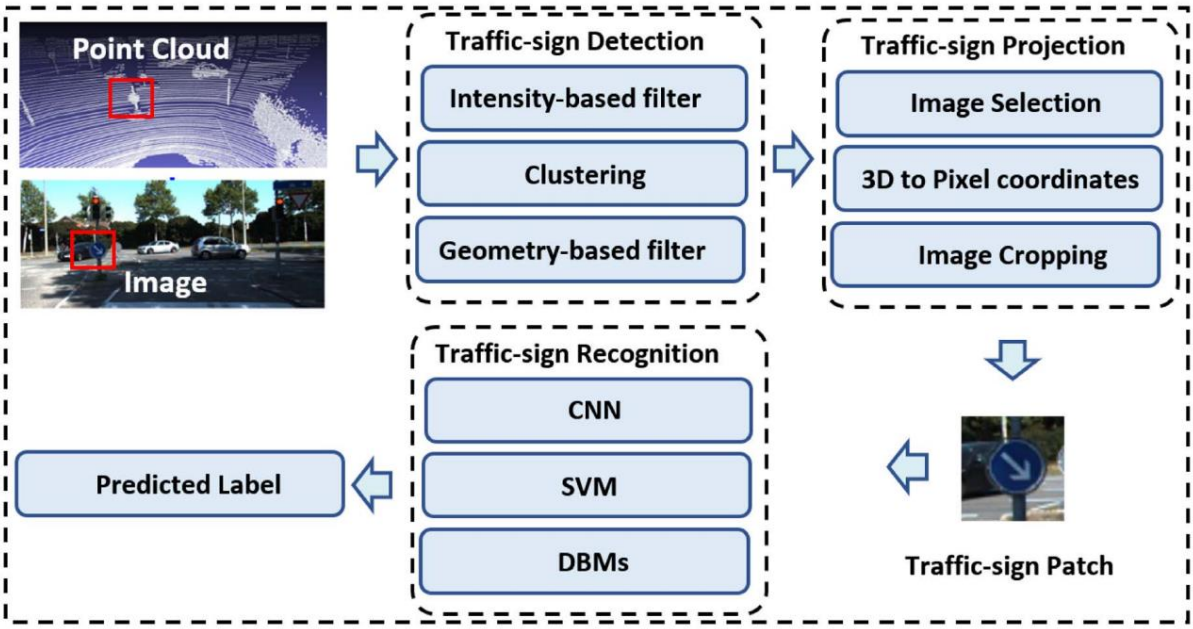
K detekci dopravních značek se také využívá algoritmu SSD. [41] Hlavní rozdíl mezi SSD a YOLO je ten, že SSD probíhá pouze v horních vrstvách, kdežto YOLO probíhá na různých vrstvách 5 měřítek. Hlavní architektura SSD je rozdělena do dvou částí. První částí je je backbone model, kterým je předtrénovaná klasifikační síť, která extrahuje mapu příznaků. Nad první částí je aplikována druhá část, která obsahuje několik konvolučních vrstev dohromady. Výstupem jsou ohraničující rámečky okolo detekovaného objektu. [42]



Obr. 17 Architektura SSD [42]

2.3 DETEKCE NA ZÁKLADĚ FÚZE DAT ZE SENZORŮ

Za účelem získání přesných informací o 3D okolí vozidla v reálném čase se novým tématem výzkumu stala fúze senzorů. Toto téma je zpracováno v rešeršní práci [43], především fúze kamery, která reálný svět promítá do obrazové roviny a LiDARu. Pro zpracování těchto záznamů je potřeba využít různých metod pro extrakci příznaků, neboť obraz z kamery je uspořádaný a diskrétní, kdežto mračno bodů je nepravidelné, neuspořádané a nespojité. Nejprve se lokalizuje dopravní značka na skenu z LiDARu na základě její retroreflexní vlastnosti. Poté je 3D pozice značky promítnuta do roviny obrazu z kamery, odkud se vystřihne pouze daná oblast, která je následně vložena do klasifikátoru obrazu, kterým mohou být například konvoluční neuronové sítě. Celý tento proces je přehledně znázorněn na *Obr. 18*.



Obr. 18 Postup při rozpoznávání dopravních značek v případě fúze dat z více senzorů [43]

3 UČENÍ NEURONOVÝCH SÍTÍ

Učení neuronové sítě je proces, při kterém se model učí rozpoznávat vzory a vzájemné vztahy v datech. Existuje několik kategorií strojového učení – řízené učení, neřízené učení, samořízené učení a posilované učení. Nejběžnější metodou je učení řízené, které je zejména používáno pro aplikace, jako je optické rozpoznávání znaků, překlady jazyka nebo rozpoznávání řeči. Základem tohoto učení je snaha naučit model předvídat správné výstupy na základě známých vstupů a příslušných cílových hodnot, které jsou předem nastaveny člověkem. V případě detekce objektů jsou na každý obrázek datové sady nakresleny ohraničující rámečky kolem zájmových objektů. Takto připravená data jsou rozdělena na trénovací, validační a testovací množinu. Validační množina je používána během procesu trénování modelu k monitorování jeho výkonu a úpravě parametrů a hyperparametrů. Testovací množina slouží k finálnímu hodnocení modelu po dokončení trénování. [44]

Pro trénování modelu se také využívá metody přeneseného učení (transfer learning). Tato metoda využívá váhy z modelů, které byly trénovány na velkých a rozmanitých datových sadách (například na datasetech s tisíci obrázky) a jsou tedy schopné zachytit obecné vlastnosti a strukturu dat. Tyto váhy jsou pak použity jako výchozí body pro trénování nového modelu na menší a specifičtější datové sadě. Tato metoda je praktická z několika důvodů. Prvním z nich je případ, kdy nashromážděná datová sada není dostatečného rozsahu pro naučení se všech potřebných rysů objektů, které mají být detekovány. Druhým důvodem je rychlejší a efektivnější trénování nového modelu. Ten tak nemusí začínat trénování od nuly, ale může využít zkušenosti a znalosti již natrénovaného modelu. To umožňuje dosáhnout lepších výsledků s menším množstvím trénovacích dat v krátkém čase. [45]

3.1 PARAMETRY

Hodnoty, které řídí chování systému se nazývají parametry. Správná inicializace a aktualizace těchto parametrů během tréninku je klíčová pro dosažení efektivního učení a dobrého výkonu modelu na konkrétním úkolu. [20] Tyto parametry jsou specifické pro každou vrstvu a architekturu sítě. Každá vrstva může mít své vlastní váhy a biasy, které se učí během tréninku. Celkový počet parametrů v síti závisí na velikosti a složitosti sítě, včetně počtu vrstev a velikosti každé vrstvy.

Mezi tyto parametry patří:

- **Váhy** – parametry, které se nacházejí na spojeních mezi jednotlivými neurony v síti. Každé spojení má svou váhu, která ovlivňuje, jakou váhu bude mít vstupní signál do daného neuronu. Váhy jsou upravovány během tréninku sítě, aby se minimalizovala chyba a dosáhlo se požadovaných výstupů. [20]

- **Biasy** – přidávají se ke každému neuronu v síti. Tyto parametry přidávají konstantní hodnotu k váženému součtu vstupů neuronu a umožňují posunutí aktivační funkce neuronu. Biasy jsou také upravovány během tréninku, aby se model co nejlépe přizpůsobil datům. [20]

3.2 HYPERPARAMETRY

Na začátku trénování neuronové sítě je potřeba nastavit hyperparametry, které ovlivňují výkon po dobu její konvergence. [46] Zde je nutno zmínit, jsou nastaveny na začátku trénování a nejsou na rozdíl od parametrů přizpůsobovány samotným učícím algoritmem. [20]

- **Batch** – označuje počet trénovacích dat, která jsou zpracovávána současně během jedné iterace trénování. Tento parametr ovlivňuje rychlost trénování a množství potřebné paměti. Čím větší je velikost batche, tím rychleji sítě konvergují a tím více paměti potřebují. Naopak menší batche potřebují méně paměti, ale trénování může trvat déle. Nastavení tohoto parametru je závislé na výkonu grafické karty. [47]
- **Epocha** – značí počet průchodů celou trénovací datovou sadou během trénování. Trénování neuronových sítí probíhá iterativně, kdy během každé epochy jsou váhy sítě upravovány tak, aby se minimalizovala chyba predikce. Čím více epoch, tím více průchodů sítě daty proběhne a tím více se váhy upraví. Příliš vysoký počet epoch může vést k přeučení, a tedy horším výsledkům na nových neznámých datech. [47]

3.3 PŘEUČENÍ A PODUČENÍ

Přeučení (overfitting) a podučení (underfitting) jsou dva jevy, které se mohou vyskytnout během tréninku neuronových sítí.

Přeučení nastává, když se model naučí příliš přizpůsobit tréninkovým datům a nezobecňuje dobře na nová, neznámá data. Přeučení může nastat, pokud máme složitý model s příliš mnoha parametry vzhledem k dostupným datům. Model se pak příliš naučí specifické vzorce a šum v tréninkových datech, které nemusí být relevantní pro všeobecný problém. V důsledku toho může model vykazovat vysokou přesnost na tréninkových datech, ale nízkou přesnost na nových datech. Ověření, zda model je či není přeučen probíhá na testovací datové sadě. [22]

Podučení nastává, když se model nedokáže dostatečně dobře naučit nebo zachytit vzorce v datech. To může být způsobeno nedostatečnou kapacitou modelu, nedostatkem tréninkových dat nebo nevhodnou volbou hyperparametrů. Výsledkem je, že model nedosahuje dostatečně dobrého výkonu ani na tréninkových, ani na nových datech. [20]

Ideálním cílem je dosáhnout vyváženého stavu mezi přeučením a podučením, kde model správně generalizuje a zobecňuje naučené vzorce na nová data. [20]

3.4 METRIKY HODNOCENÍ

Pro hodnocení modelu během trénování se používají následující metriky hodnocení:

Loss funkce (také známá jako ztrátová funkce nebo objektivní funkce) se používá k porovnání skutečných výstupů s predikcemi modelu během trénování strojového učení. Cílem je minimalizovat rozdíl mezi těmito výstupy a maximalizovat přesnost modelu. [48]

Ztrátová funkce slouží jako cílová funkce pro optimalizaci vah a parametrů modelu během tréninku. Během zpětné propagace se gradient ztrátové funkce vypočítá a použije se k úpravě vah a parametrů modelu tak, aby se minimalizovala ztráta a dosáhla se lepší predikce. [20]

Precision – udává, jak přesně model identifikuje pozitivní třídy. Vypočítá se podle vzorce následujícího vzorce:

$$Precision = \frac{TP}{TP+FP} \tag{1}$$

Kde TP je počet správně klasifikovaných pozitivních objektů a FP je počet falešně pozitivně klasifikovaných objektů. [49]

Recall – vyjadřuje schopnost modelu nacházet pozitivní predikce. Určí se ze vzorce:

$$Recall = \frac{TP}{TP+FN} \tag{2}$$

Kde TP je počet správně klasifikovaných pozitivních objektů a FN je počet falešně negativně klasifikovaných objektů. [49]

F1 Score – je vyjádřeno harmonickým průměrem recall a precision, slouží k pochopení jejich vzájemnému vztahu, pokud je potřeba zahrnout do vyhodnocení význam obou metrik. [49] Vypočítá se ze vzorce:

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision+recall} \tag{3}$$

mAP – je průměrná hodnota AP pro každou třídu. Vypočítá se z následujícího vzorce:

$$mAP = \frac{1}{n}\sum_{i=1}^{n} AP_i \tag{4}$$

Kde n je počet tříd a AP je průměrná přesnost, která zohledňuje schopnost modelu detekovat a klasifikovat objekty pro různé třídy. [50]

4 IMPLEMENTACE MODELU DETEKCE

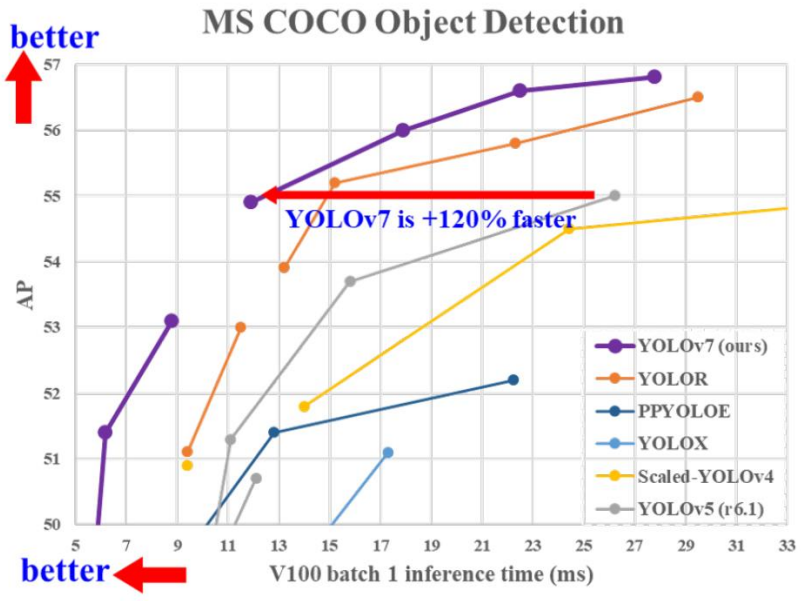
Na základě předchozí rešerše o způsobech detekce dopravního značení byl zvolen jednokrokový přístup. Pro vytvoření modelu tvořeného konvoluční neuronovou sítí, bylo nejprve potřeba vybrat programovací jazyk a architekturu předučené sítě pro dotrénování modelu na požadovanou aplikaci. Proto bylo nutné nashromáždit data s obrázky dopravních značek a připravit datovou sadu, která slouží k trénování a validaci modelu. V této kapitole budou tyto procesy popsány.

4.1 PROGRAMOVACÍ JAZYK A ZVOLENÁ ARCHITEKTURA SÍTĚ

Výběr programovacího jazyka závisí na mnoha faktorech, jako je dostupnost knihoven pro práci s neuronovými sítěmi, efektivita výpočetního prostředí, ale také vlastní zkušenosti a preference. Na základě předchozích zkušeností byl zvolen programovací jazyk Python, který patří mezi populární jazyky pro tvorbu neuronových sítí. Mezi nejznámější knihovny patří TensorFlow, Pytorch nebo Keras, které poskytují širokou podporu pro tvorbu a trénování sítí. Všechny tyto knihovny jsou volně dostupné a umožňují integraci s různými platformami. Pro trénování neuronové sítě byla zvolena platforma Google Colab od Google Reseachers. A to hlavně z toho důvodu, že poskytuje bezplatný přístup k výpočetním zdrojům včetně GPU. GPU je speciální druh mikroprocesoru, který je určen pro zpracování grafických dat. Například grafické karty NVIDIA využívají speciální architekturu nazvanou CUDA, díky které je možné provádět několik tisíc sekvencí operací paralelně. [51]

4.1.1 YOLOv7

Jako základ modelu byla zvolena architektura sítě YOLOv7, která pro strojové učení využívá Python knihovnu PyTorch. Vydali ji v roce 2022 Chien-Yao Wang, Alexey Bochkovskiy a Hong-Yuan Mark Liao. Tato architektura je navržena tak, aby zajistila rychlou a efektivní detekci objektů v reálném čase od 5 do 160 FPS. Při detekci objektů v reálném čase dosahuje nejvyšší průměrné přesnosti 56,8 % při 30 FPS. [52]



Obr. 19 Porovnání YOLOv7 s ostatními real-time detektory [52]

Architektura YOLOv7 je podobně jako jiné architektury rozdělena na tři hlavní části:

1. **Backbone (základna):**

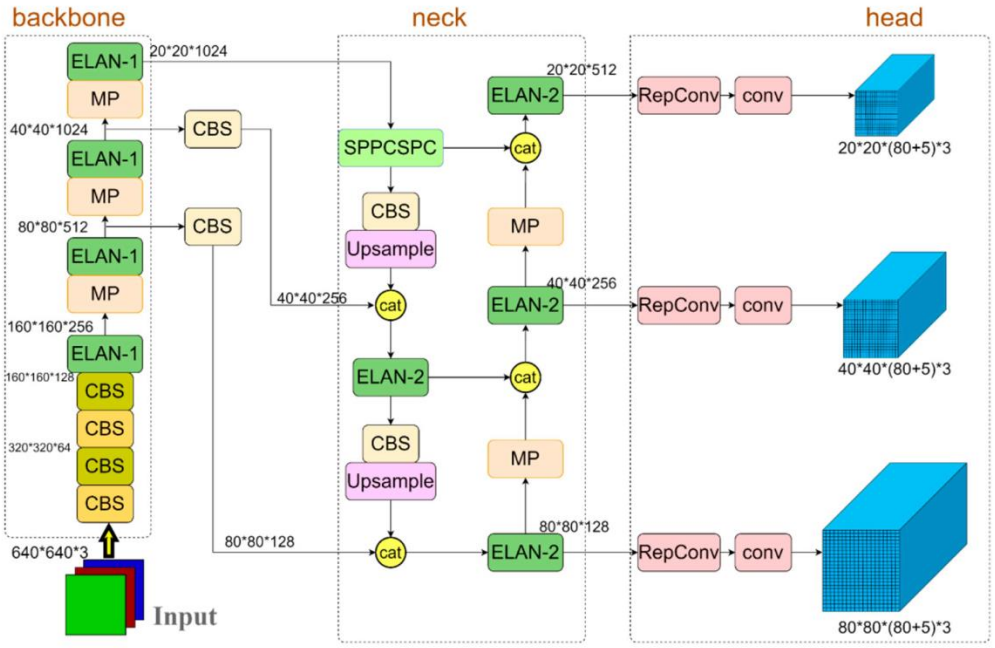
   Backbone je základní konvoluční síť architektury YOLOv7, která slouží k extrakci příznaků ze vstupních obrazů. Je založen na CNN architektuře E-ELAN. Trénování probíhalo na velkém datasetu MS COCO, díky čemuž je umožněno efektivní získání příznaků z obrazových dat. [53]

2. **Neck (krk):**

   Neck je část, která hraje důležitou roli ve spojení mezi extrakcí příznaků (backbone) a detekcí objektů (head). Neck slouží k integraci příznaků z různých vrstev backbone do pyramid příznaků, z kterých následně vznikají mapy příznaků, což umožňuje zachytit informace na různých úrovních rozlišení a kontextu. [54]

3. **Head (hlava)**:

   Head je poslední část sítě, která zpracovává příznaky přijaté z předchozích vrstev. Hlavním úkolem je převádění těchto příznaků na ohraničující rámečky, třídy objektů a pravděpodobnosti, které popisují detekované objekty na obraze. [54]



Obr. 20 Architektura YOLOv7 [55]

4.2 PŘÍPRAVA A TRÉNOVÁNÍ MODELU

Vývoj detekčního modelu dopravního značení zahrnuje nezbytné fáze, které musí být provedeny před samotným trénováním modelu. V této kapitole bude popsána příprava dat a jejich následné rozdělení do oddělených sad, které budou využity k trénování, validování a testování modelu. Dále zde bude popsán proces trénování modelu a způsoby jeho hodnocení v průběhu učení pomocí validační sady.

4.2.1 PŘÍPRAVA DATOVÉ SADY

Před vytvořením datové sady bylo nejprve potřeba rozhodnout, které dopravní značky bude zahrnovat. Pro tuto práci bylo zvoleno 28 tříd dopravních značek včetně semaforů.

Jedná se o následující:

- Hlavní silnice
- Konec hlavní silnice
- Stop
- Dej přednost v jízdě
- Křižovatka
- Světelné signály
- Železniční přejezd bez závor
- Železniční přejezd se závorami
- Pozor přechod pro chodce
- Přednost před protijedoucími vozidly
- Zákaz předjíždění
- Jednosměrný provoz
- Zákaz vjezdu (jednosměrný provoz)
- Nejvyšší dovolená rychlost 80
- Konec nejvyšší dovolené rychlosti 80
- Zákaz vjezdu
- Přednost protijedoucích vozidel
- Konec všech zákazů
- Přechod pro chodce
- Kruhový objezd
- Přikázaný směr vlevo
- Přikázaný směr přímo
- Přikázaný směr vpravo
- Zákaz zastavení
- Zákaz stání
- Semafor – zelená
- Semafor – oranžová
- Semafor – červená

Obr. 21 Přehled dopravního značení obsaženého v datové sadě [63]

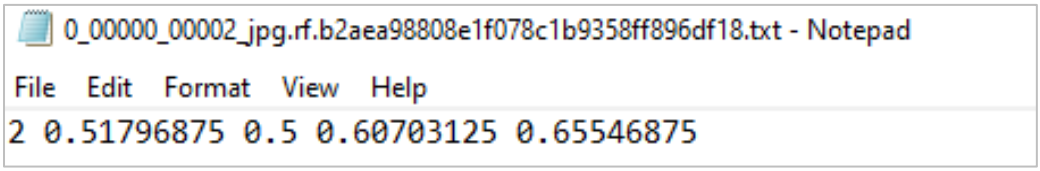
TVORBA DATOVÉ SADY

Při tvorbě datové sady bylo uvažováno o využití již hotových sad pro detekci dopravního značení. Vzhledem k podobnosti značení se jednalo převážně o belgickou a německou sadu. Nakonec však bylo využito jenom malé množství tříd, a to konkrétně – hlavní silnice, dej přednost v jízdě a zákaz vjezdu jednosměrný provoz. Data k ostatním třídám byla shromážděna z obrázků z Google Map, z vlastních záznamů z kamery a z uměle vytvořených obrázků, kde dopravní značka byla náhodně generována v různé velikosti a poté náhodně umístěna na obrázek. Ukázka z datové sady je na *Obr. 22*.

Obr. 22 Ukázka obrázků z datové sady

ANOTACE DATOVÉ SADY

Následně pak takto vytvořená datová sada musela být anotována. Proces anotace je časově velmi náročný. Spočívá v tom, že se na každém obrázku musí ručně okolo dopravní značky vytvořit ohraničující rámeček, kterému je přiřazena konkrétní třída. Tato konkrétní anotace je pak uložena do textového souboru ve specifickém formátu pro YOLOv7. Ukázkový textový soubor lze vidět na *Obr. 23* níže.



Obr. 23 Textový soubor obsahující informace o objektu na obrázku

V prvním sloupci se nachází číslo třídy na obrázku, v následujících sloupcích jsou souřadnice ohraničujícího rámečku objektu. Jedná se o x a y souřadnici středu tohoto pole, šířku a výšku. Pro lepší představu jsou tyto souřadnice zakótovány na *Obr. 24*.

Obr. 24 Souřadnice ohraničujícího rámečku objektu [56]

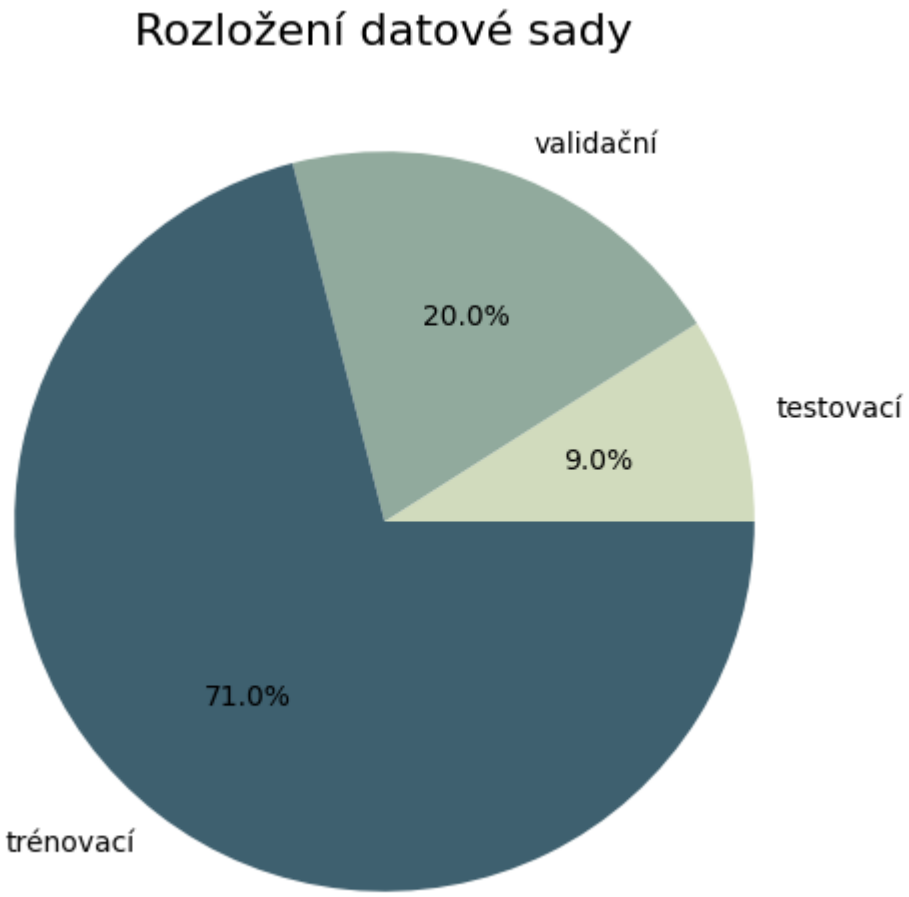
Jak vypadá konkrétní anotace datové sady dopravních značek lze vidět na *Obr. 25*. K tomuto bylo využito prostředí Roboflow. V tom je možné nahrání celé sady, poté její anotace a následný export verze přímo pro konkrétní model, ve kterém bude datová sada využita. Velkou výhodou je, že Roboflow znázorňuje přehled toho, jestli je rozložení obrázků v jednotlivých třídách rovnoměrné. Pokud je některá třída nedostatečně zastoupená, je to na první pohled vidět. Na *Obr. 26* lze vidět ukázku rozložení datové sady před augmentací.

Obr. 25 Ukázka anotovaných obrázků

Obr. 26 Zobrazení rozložení tříd v průběhu tvorby datové sady v Roboflow

ROZDĚLENÍ A ROZŠÍŘENÍ DATOVÉ SADY

Pro vyhodnocení výkonnosti modelu bylo potřeba datovou sadu rozdělit na trénovací, validační a testovací. Jak již z názvu vypovídá, trénovací sada se používá k učení modelu. Validační sada je během trénování využívána při ladění hyperparametrů modelu, čímž zajišťuje nezkreslené vyhodnocení modelu. K závěrečnému hodnocení modelu pak slouží testovací sada. Aby byla tato hodnocení opravdu nezávislá, je nutné, aby se obrázky z trénovací sady neopakovaly v sadě validační a testovací. V případě učení modelu na malé množině dat, jako je tomu v tomto případě, je potřeba datovou sadu rozšířit datovou augmentací, jako je například úprava jasu a kontrastu [22]

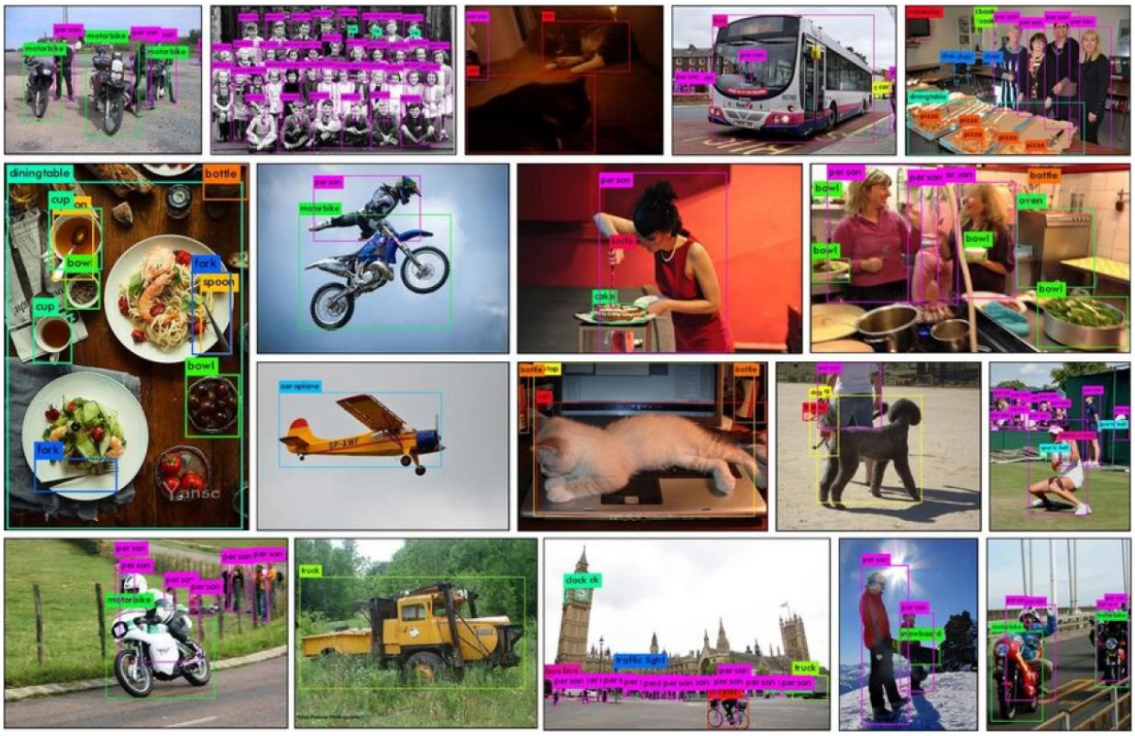


Obr. 27 Graf rozdělení datové sady dopravního značení

Na *Obr. 27* lze vidět rozložení vytvořené augmentované datové sady, která je složena z 15407 obrázků. Cílové rozložení bylo 70:20:10, výsledek se nepatrně odlišuje z důvodu pozdějšího rozšiřování sady pro zpřesnění detekce. Nutno podotknout, že dokud byla datová sada v rozsahu nižších tisíců obrázků, v Roboflow se s ní pracovalo dobře. Po překročení 10 tisíc obrázků nešlo nahrávat další soubory, které bylo potřeba anotovat pro přesnější detekci specifických tříd. Průměrné rozložení obrázků v trénovací sadě je 400 obrázků na jednu třídu, dalších 100 obrázků je pak rozděleno do validační a testovací složky.

4.2.2 TRÉNOVÁNÍ MODELU

Pro trénování modelu byla zvolena metoda přeneseného učení na modelu YOLOv7. Tento model byl předtrénován na datové sadě MS COCO, která obsahuje přes 330 tisíc obrázků s více než 2,5 miliony anotovaných objektů v 80 různých třídách, jako jsou například lidé, zvířata, nábytek, jídlo a další. [57]



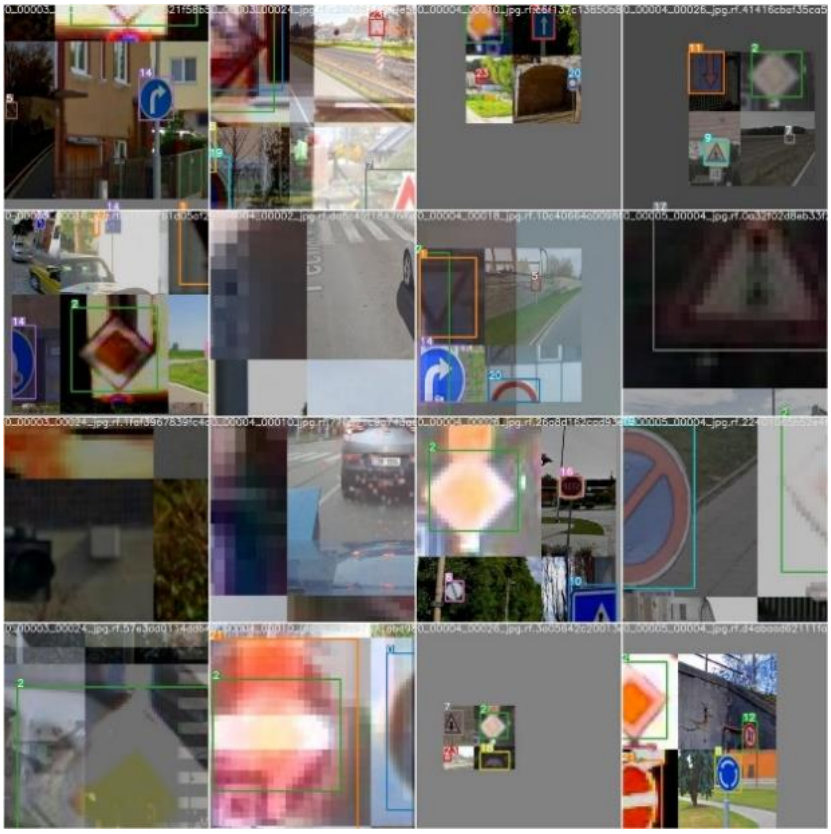
Obr. 28 Ukázka z datové sady MS COCO [57]

NASTAVENÍ PARAMETRŮ PRO TRÉNOVÁNÍ

Pro trénování modelu byl využit skript z YOLOv7 train.py, ve kterém bylo potřeba nastavit parametry uvedené v kapitole 3.2 a cesty k následujícím souborům:

- **Cfg** (configuration) - konfigurační soubor, který určuje nastavení architektury modelu, hyperparametry trénování a další parametry.
- **Data** – konfigurační soubor, který obsahuje názvy tříd a cesty ke složkám s trénovací, validační a testovací sadou.
- **Weights** – soubor s předtrénovanými váhami modelu YOLOv7.

Model pro detekci byl trénován s parametrem batch o velikosti 16 na grafické kartě NVIDIA Tesla T4, která je díky tenzorovým jádrům NVIDIA Turing™ uzpůsobená pro hluboké učení. Karta má 16 GB dedikované paměti pro výpočty, což umožňuje paralelní zpracování velkých množství dat. [58]



Obr. 29 Ukázka jedné trénovací sady (*batch = 16*)

4.2.3 HODNOCENÍ MODELU NA VALIDAČNÍ SADĚ BĚHEM TRÉNOVÁNÍ

V průběhu trénování byly průběžně do konzole vypisovány hodnoty validující aktuální epochu. Konkrétní ukázka výstupu je zobrazena na *Obr. 30*. Celková doba trénování modelu byla přibližně 2,5 hodiny.

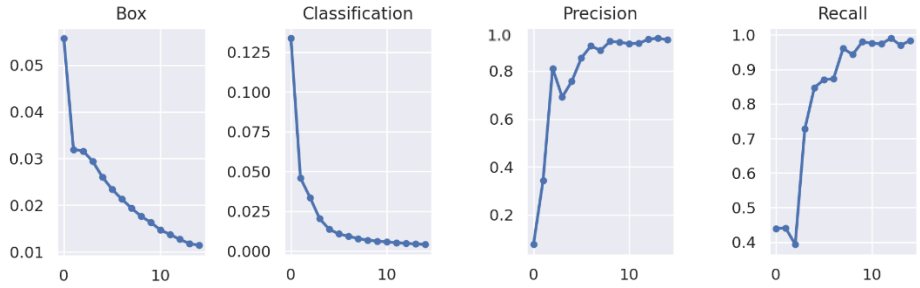
```
Epoch   gpu_mem       box       obj       cls     total    labels  img_size
 1/14     13.6G   0.04475  0.007588   0.05245    0.1048        26       640: 100% 693/693 [08:47<00:00,  1.31it/s]
           Class      Images      Labels           P           R      mAP@.5  mAP@.5:.95: 100% 93/93 [00:52<00:00,  1.76it/s]
             all        2955        3106      0.0335       0.398      0.0863      0.0472

Epoch   gpu_mem       box       obj       cls     total    labels  img_size
 2/14     11.1G   0.04011  0.007607    0.0435   0.09122        49       640: 100% 693/693 [08:44<00:00,  1.32it/s]
           Class      Images      Labels           P           R      mAP@.5  mAP@.5:.95: 100% 93/93 [00:52<00:00,  1.78it/s]
             all        2955        3106       0.236       0.422       0.163      0.0901
```

Obr. 30 Výstup do konzole při trénování modelu

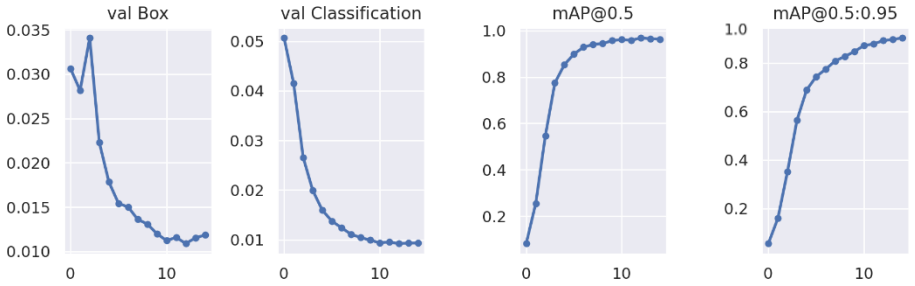
Na *Obr. 31* lze vidět hlavní metriky vyhodnocení při trénování modelu. Jedná se o ztrátovou funkci, která byla popsána v kapitole 3. Pro vykreslení výsledků byly využity následující typy této funkce:

- **Box loss** (ztrátová funkce souřadnic) – měří rozdíl mezi predikovanými a skutečnými souřadnicemi a souřadnicemi, které ohraničují detekovaný objekt.
- **Classification loss** (ztrátová funkce klasifikace) – hodnotí, jak dobře model klasifikuje objekty do různých tříd. [59]



Obr. 31 Hlavní metriky trénování modelu

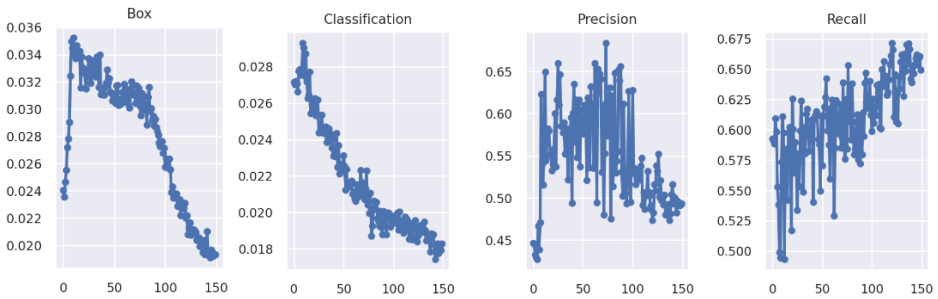
Tyto hodnotící funkce se používají i při ověřování modelu na validační sadě obrázků. Místo precision a recall je zde použita metrika mAP. Vyšší hodnota mAP značí lepší výkon modelu. Hodnota 1 znamená dokonalou detekci a 0 znamená, že žádný objekt nebyl správně detekován.



Obr. 32 Hlavní metriky validování modelu

Na základě předchozích experimentů bylo rozhodnuto, že pro natrénování modelu je optimální počet epoch 15. Toto rozhodnutí vychází z iterací, ve kterých byl model trénován na vyšším počtu epoch. Výsledky potvrdily, že model dosáhne bodu, kdy se ztrátová funkce přestává významně snižovat a metriky precision a recall dosahují stabilních hodnot. To naznačuje, že model již nezískává vylepšení výkonu při dalším prodlužování trénování.

Na následujícím obrázku jsou zobrazeny grafy vyhodnocující trénování modelu na architektuře YOLOv7, která nebyla předtrénována na datové sadě MS COCO. Na tomto příkladu lze potvrdit, že předtrénovaný model vah výrazně urychluje dobu trénování a zlepšuje jeho přesnost. Zde ani po 150 trénovacích epochách model nedosahoval precision a recall 70 %, čehož se předtrénovanému modelu podařilo dosáhnout již ve čtvrté epoše.



Obr. 33 Výsledky během trénování na architektuře YOLOv7 bez předtrénovaných vah

4.3 NÁVRH A ŘEŠENÍ ALGORITMŮ

Hlavním úkolem algoritmů, které mají být navrženy, je komunikace s kamerou a samotná detekce dopravního značení. Jelikož autonomní vozidla musí reagovat na dopravní značení okamžitě, algoritmy musí být navrženy tak, aby co nejdříve a nejpřesněji prováděly detekci v reálném čase.

V případě této práce bude experimentální ověření řešeno tak, že detekce bude probíhat na předem nahraném videu z kamery ZED 2. Snímky z videa jsou plnohodnotným nahrazením snímků, které by byly detekovány přímo z výstupu z kamery v reálném čase. Tento přístup byl zvolen především z důvodu nedostatečného výpočetního výkonu na vlastním zařízení.

Prvním krokem v návrhu algoritmů je volba vhodných Python knihoven. Samotný model byl trénován na architektuře YOLOv7, která disponuje vlastní rozsáhlou databází algoritmů pro trénování i detekci. Jak již bylo zmíněno, YOLOv7 využívá PyTorch jako hlavní knihovnu. Z důvodu zjednodušení celého procesu byla knihovna PyTorch zvolena i pro navrhovaný algoritmus detekce.

Zvažovanou možností byla také knihovna od společnosti NVIDIA TensorRT, která je speciálně optimalizována pro rychlé zpracování inferencí neuronových sítí. Nicméně pro převedení předtrénovaného modelu ve formátu PyTorch na formát pro TensorRT je nutné využití GPU, která nebyla k dispozici.

Součástí PyTorch je také knihovna Torchvision, která při detekci bude aplikována pro práci s výsledky detekce objektů. Pro práci s obrazovými daty z kamery bude použita knihovna OpenCV.

Pro další potřebné kroky při zpracování budou použity Random, Numpy, Pandas a Time. Specifické využití všech zmíněných knihoven bude popsáno v následujících podkapitolách pro konkrétní algoritmus zvlášť. Pro lepší přehlednost jsou všechny zmíněné knihovny vypsány v seznamu zde:

- YOLOv7
- PyTorch
- Torchvision
- OpenCV
- Random
- Numpy
- Pandas
- Time

4.3.1 ALGORITMUS PRO KOMUNIKACI S KAMEROU

Algoritmus pro komunikaci s kamerou byl navržen tak, aby zaznamenával obraz, který následně bude zpracován v detekčním algoritmu. Pro aplikaci v reálném čase je potřeba oba tyto algoritmy zkombinovat. To znamená, že na vstupu pro detekci by tak místo snímků z videa byly snímky přímo z kamery. Tento algoritmus je se všemi ostatními zahrnout v příloze 2. Tato příloha je v notebooku pro aplikaci Jupyter Notebook, který je pro strojové učení běžně využíván. [44]

Ke snímání obrazu byla použita stereoskopická kamera ZED 2 vyvinutá společností Stereolabs. Tato kamera nabízí vysoké rozlišení a schopnost zaznamenávat obraz se stereoskopickým efektem, což umožňuje získávání hloubkových informací a vytváření 3D percepčního vnímání prostředí v reálném čase. [60]

Obr. 34 Kamera ZED 2 [60]

Vzhledem k tomu, že pro aplikaci detekce není potřeba snímání ve stereo obraze, bylo nutné vyfiltrovat pouze jeden obraz z kamery. Jelikož kamera byla připevněna na čelním skle automobilu pomocí držáku, který neumožňoval jiné upevnění, celý obraz byl zaznamenán obráceně o 180°. Na *Obr. 35* lze vidět originální vstupní obraz, který musí být v algoritmu dále zpracován.

Obr. 35 Originální obraz z kamery ve vozidle

Na začátku algoritmu jsou importovány všechny potřebné knihovny:

```
import numpy as np
import cv2
```

Dále jsou definovány následující parametry:

- Číselný index kamery
- Název výstupního videa
- Alpha parametr pro změnu kontrastu
- Beta parametr pro změnu jasu

Číselný index kamery se může lišit v závislosti na počtu připojených kamer k zařízení. Bohužel se tyto indexy náhodně mění, proto nejlepší způsob, jak zjistit index pro požadovanou kameru, která má být používána, je vyzkoušet různé hodnoty indexů.

Výchozí hodnota parametru alpha pro nastavení kontrastu je 1. Změněním tohoto parametru mezi hodnoty větší než 0 a menší než 1 bude dosaženo snížení kontrastu. Pokud bude hodnota nastavena na hodnoty větší než 1, kontrast bude zvýšen. U parametru beta pro nastavení jasu je výchozí hodnota pro originální obraz z kamery 0. Doporučené hodnoty pro snižování a zvyšování jasu jsou v rozmezí od -100 do 100.

Pomocí OpenCV je inicializována kamera, včetně definice jejího hlavního rozlišení.

```
cap = cv2.VideoCapture(camera)
cap.set(cv2.CAP_PROP_FRAME_WIDTH, 2560)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 720)
```

V případě, že je požadavek na to, aby byl zobrazený obraz z kamery zaznamenáván, je také potřeba definovat šířku okna, ve kterém bude zpracovaný obraz zobrazen a také samotný enkodér výstupního videa.

```
cv2.namedWindow('kamera',0)
cv2.resizeWindow('kamera',1280,720)
fourcc = cv2.VideoWriter_fourcc(*'XVID')
out = cv2.VideoWriter(output_name, fourcc, 30.0, (1280, 720))
```

Hlavním cílem enkodéru videa je minimalizovat velikost souboru videa, aby se snížila náročnost na přenos dat a úložný prostor. Toho je dosaženo odstraňováním redundantních informací v obraze, jako jsou opakující se vzory nebo nepodstatné změny mezi snímky.

Následně je pak v hlavní smyčce získán snímek z kamery. Ten je pomocí knihovny Numpy podle hlavní osy rozdělen na polovinu– tedy levý a pravý snímek. Pro další zpracování se pracuje pouze se snímkem levým, který je otočen o 180°. Dále jsou na něm nastaveny zadané hodnoty parametrů alpha a beta, které upravují jas a kontrast. Na závěr je snímek zapsán do výstupního videa a cyklus pokračuje do té doby, dokud není zastaven uživatelem stisknutím klávesy q. Výstupní video je ve formátu HD720 (1280 x 720).

```
while(cap.isOpened()):
    ret, frame = cap.read()
    if ret:
        left_right_image = np.split(frame, 2, axis=1)
        left_frame = cv2.rotate(left_right_image[1], cv2.ROTATE_180)
        left_frame = cv2.convertScaleAbs(left_frame, alpha=alpha, beta=beta)

        vidout=cv2.resize(left_frame,(1280,720))
        out.write(vidout) #

        cv2.imshow('kamera', left_frame)

        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    else:
        break

cap.release()
out.release()
cv2.destroyAllWindows()
```

4.3.2 ALGORITMUS PRO DETEKCI DOPRAVNÍHO ZNAČENÍ

V úvodu algoritmu pro detekci dopravního značení jsou nejprve importovány všechny potřebné knihovny.

```
import torch
import os
import cv2
import random
import time
from torchvision.ops import nms
```

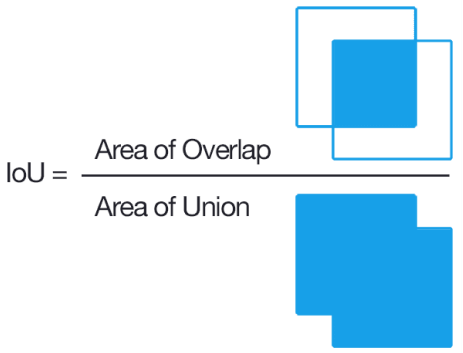
Dále jsou definovány cesty k souborům jako je vstupní video k detekci a předučený model YOLOv7. Pro výstupní video je nastaven název a adresář, do kterého má být uloženo. Hlavními parametry algoritmu jsou prahové hodnoty:

- Confidence threshold
- NMS threshold

```
confidence_threshold = 0.25
nms_threshold = 0.45
```

Hlavním úkolem parametru confidence threshold je odfiltrování detekcí s nízkou jistotou. Nastavením vyššího confidence threshold se zvyšuje přísnost detekce, což znamená, že jsou akceptovány pouze ty detekce s vysokou jistotou. To může vést k nižšímu počtu detekcí, ale zároveň k větší spolehlivosti detekovaných objektů. Naopak, snížení confidence threshold umožní zachytit více detekcí, ale může také zvýšit riziko falešných pozitivních detekcí.

NMS je algoritmus používaný v počítačovém vidění k odstranění překrývajících se detekcí. Ze všech detekcí v oblasti je vždy vybrána ta s největší pravděpodobností, podle které jsou následně posuzovány všechny okolní detekce. Prahová hodnota NMS threshold určuje dovolenou hodnotu překryvu mezi detekcemi. Pokud je tzv. IoU parametr vyšší než dovolená hodnota, posuzovaná detekce je potlačena. Tento parametr je vypočítán jako podíl oblasti překrytí a sdružené oblasti detekcí. [61]



Obr. 36 Výpočet parametru IoU [61]

Obr. 37 Detekce před a po použití NMS [62]

Po nastavení těchto parametrů je definován list, který obsahuje seřazené názvy všech tříd, tak jako tomu bylo v konfiguračním souboru pro trénování. Tento list slouží jako zdroj pro následné vykreslení ohraničujících rámečků s názvem třídy a přesností.

Dále je načten model a definována velikost, na kterou budou jednotlivé snímky z videa změněny před vstupem do modelu. Tato hodnota odpovídá rozlišení 640 x 640, na kterém byl samotný model trénován.

```
device = torch.device(0)
model = torch.hub.load('WongKinYiu/yolov7', 'custom', model_path,
                       force_reload=True, trust_repo=True)
model_input_size = (640, 640)
```

Podobně jako u algoritmu pro komunikaci s kamerou následuje inicializace zdroje, kterým je v tomto případě nahrané video.

```
video = cv2.VideoCapture(video_path)
```

Dále jsou získány parametry videa a definován enkodér pro výstupní video.

```
fps = video.get(cv2.CAP_PROP_FPS)
frame_width = int(video.get(cv2.CAP_PROP_FRAME_WIDTH))
frame_height = int(video.get(cv2.CAP_PROP_FRAME_HEIGHT))
frame_count = int(video.get(cv2.CAP_PROP_FRAME_COUNT))

fourcc = cv2.VideoWriter_fourcc(*'mp4v')
output_video_path = os.path.join(output_dir, output_file_name)
output_video = cv2.VideoWriter(output_video_path, fourcc, fps, (frame_width,
frame_height))
```

Po načtení videa a jeho inicializaci je spuštěna hlavní smyčka pro zpracování snímků. Během této smyčky se načte jeden snímek pomocí funkce *video.read()*. Pokud je načten, probíhá zpracování detekce objektů.

```
while video.isOpened():
    ret, frame = video.read()
    if not ret:
        break
```

Nejprve je snímek zmenšen na definovanou velikost. Poté je na snímku provedena detekce objektů pomocí načteného modelu a výsledky jsou uloženy. Z výsledků jsou extrahovány informace o detekcích, jako jsou souřadnice ohraničujících rámečků, název třídy a přesnost detekce. Tyto informace jsou uloženy do dataframe struktury z knihovny Pandas. Dochází také k přepočtu detekovaných souřadnic ohraničujících rámečků na původní velikost snímku.

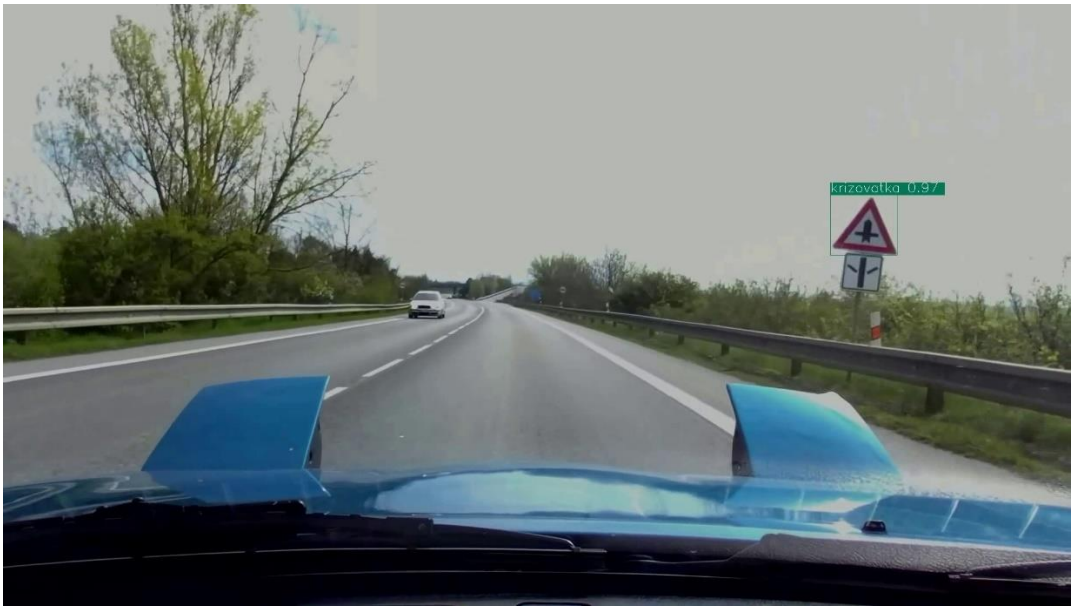
```
frame_resized = cv2.resize(frame, model_input_size)
results = model(frame_resized)
df = results.pandas().xyxy[0]
bbox_list = []

scale_x = frame_width / model_input_size[0]
scale_y = frame_height / model_input_size[1]
df['xmin'] *= scale_x
df['xmax'] *= scale_x
df['ymin'] *= scale_y
df['ymax'] *= scale_y
```

Následně je aplikován algoritmus NMS, který je nahrán z knihovny Torchvision. Po aplikaci NMS je seznam detekcí aktualizován.

```
if not df.empty:
    boxes = torch.from_numpy(df[['xmin', 'ymin', 'xmax',
    'ymax']].values.astype('float32'))
    scores = torch.from_numpy(df['confidence'].values.astype('float32'))
    keep = nms(boxes, scores, nms_threshold)
    df = df.iloc[keep]
for _, row in df.iterrows():
    if row['confidence'] > confidence_threshold:
        bbox_list.append([int(row['xmin']), int(row['ymin']),
                          int(row['xmax']), int(row['ymax']),
                          classes[int(row['class'])], row['confidence']])
```

Následuje vykreslení ohraničujících rámečků s názvem třídy a přesností detekce na snímek pro všechny detekované objekty. Název třídy a přesnost jsou podtrženy barvou dané třídy. Toto podtržení je dopočítáno na základě délky názvu. Ukázku vykreslení lze vidět na *Obr. 38*.



Obr. 38 Ukázka vykreslení detekce

```
    for bbox in bbox_list:
        cls = bbox[4]
        confidence = bbox[5]

        # nastaveni textu a barev zvyrazneni pro jednotlive tridy
        color = colors[cls]
        text_color = (255, 255, 255)
        text_bg_color = color

        # ziskani delky textu
        text = f'{bbox[4]} {bbox[5]:.2f}'
        text_size = cv2.getTextSize(text, cv2.FONT_HERSHEY_SIMPLEX, 0.8, 1)


        if len(text_size) > 0:
            text_width = text_size[0][0]
        else:
            text_width = 0

        # vypocet delky podtrzeni textu
        text_bg_width = text_width + 10

        cv2.rectangle(frame, (bbox[0], bbox[1]),
                      (bbox[2], bbox[3]), color, 1)
        if len(text_size) > 0:
            cv2.rectangle(frame, (bbox[0], bbox[1] - 22),
                          (bbox[0] + text_bg_width, bbox[1]),text_bg_color,-1)
            cv2.putText(frame, text, (bbox[0], bbox[1] - 5),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.8, text_color, 1)
```

Výsledný snímek s vykreslenou detekcí je následně zapsán do výstupního videa.

```
    output_video.write(frame)
```

Tento proces se opakuje pro každý snímek videa, dokud jsou snímky k dispozici. Po dokončení dochází k uzavření vstupního a výstupního videa.

```
video.release()
output_video.release()
```

5 TESTOVÁNÍ A VYHODNOCENÍ MODELU DETEKCE

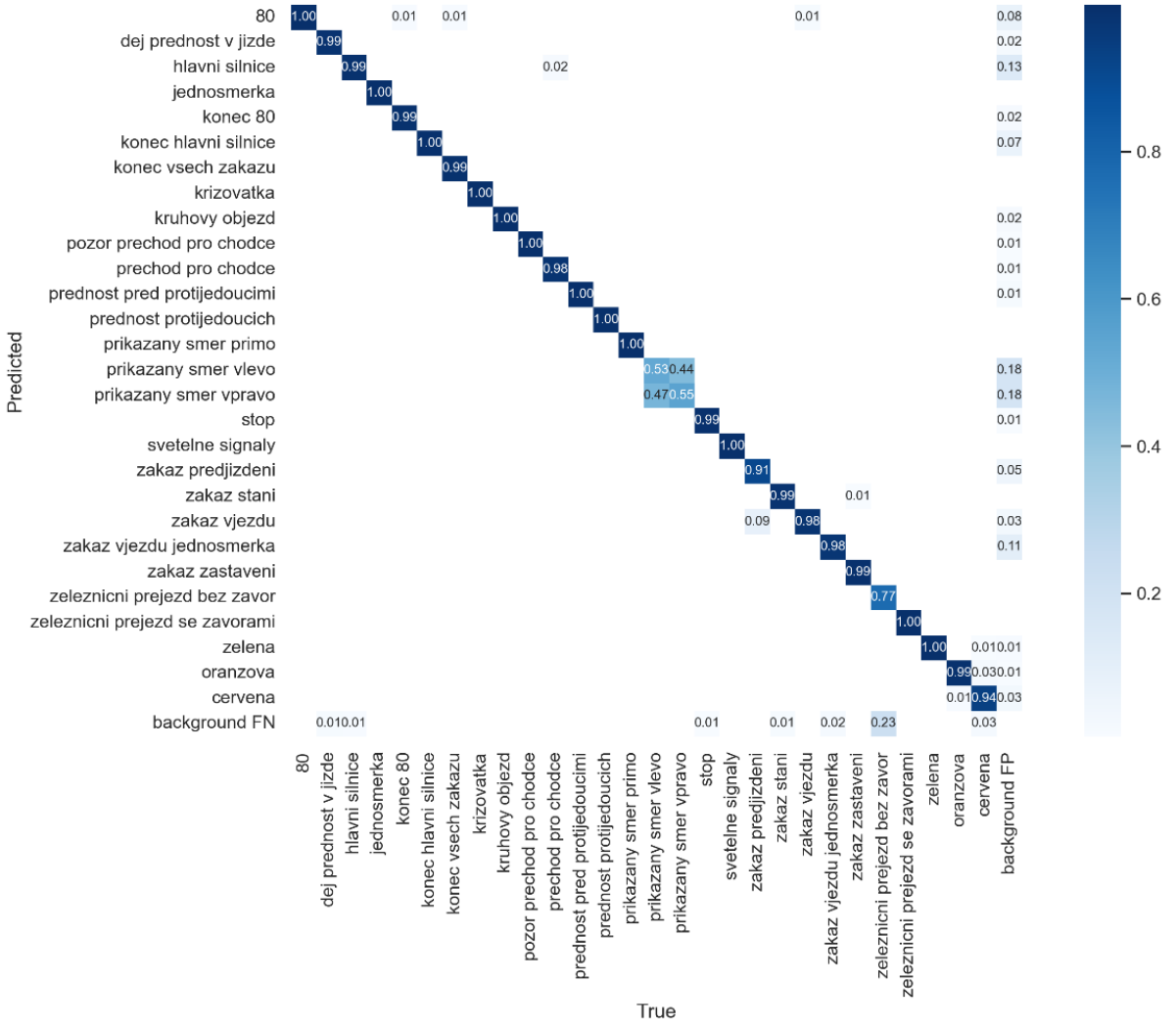
Během několika iterací trénování modelu byla po ukončení trénování použita k ověření správnosti detekce testovací datová sada i algoritmus detect.py z knihovny YOLOv7 na dalších datech nezávislých na původní datové sadě. Jelikož model dosahoval dobrých výsledků na testovací sadě i na testovacích videích, trénování bylo ukončeno. Poté byl naprogramován vlastní algoritmus pro detekci dopravního značení. Navzdory velkému množství iterací nebylo dosaženo podobné přesnosti detekce jako v případě algoritmu z knihovny YOLOv7. Porovnání detekcí obou algoritmů ze stejného snímku lze vidět na *Obr. 39* a *Obr. 40*. Při konečném zhodnocení modelu budou proto použity oba algoritmy, které mezi sebou budou následně porovnány.

Obr. 39 Detekce algoritmem z knihovny YOLOv7

Obr. 40 Detekce vlastním algoritmem

5.1 TESTOVÁNÍ MODELU NA TESTOVACÍ SADĚ

Vhodným způsobem, jak ověřit schopnost detekce daného modelu během trénování, je vykreslení grafu matice záměn, která je vytvořena z výsledků na testovací sadě. Z *Obr. 41* je patrné, že model bez problému zvládá většinu detekcí. Zároveň je ale také patrné, že dopravní značky „přikázaný směr vlevo“ a přikázaný směr vpravo“ jsou mezi sebou při detekcích zaměňovány.



Obr. 41 Matice záměn na testovací sadě

5.2 EXPERIMENTÁLNÍ OVĚŘENÍ DETEKCE

Jak již bylo zmíněno, pro experimentální ověření detekce bylo pomocí kamery ZED 2 zaznamenáno několik videí. Z důvodu zmenšení velikosti byly jednotlivé pasáže, obsahující dopravní značky z datové sady, sestříhány do videa o celkové délce 4 minuty a 29 sekund. Před sestříháním videí musela být provedena úprava videí, protože záznam nakonec nebyl pořízen přes algoritmus pro komunikaci s kamerou. To hlavně z toho důvodu, že Stereolabs vydalo ke kamerám ZED vlastní software ZED Explorer, který využívá GPU.

Díky využití GPU je možné zaznamenávat video v až 4K kvalitě, avšak při této rozlišovací schopnosti je omezena rychlost na 15 FPS. Proto byl záznam pořízen v HD kvalitě při 30 FPS. Nicméně, samotný algoritmus pro komunikaci s kamerou byl použit pro předzpracování zaznamenaných videí. Jelikož výstupní obraz měl rozlišení 3840 x 1080 (viz *Obr. 35*), bylo nutné převést stereo záznam na rozlišení 1920 x 1080 a obraz otočit o 180°.

V testovacím videu se nachází celkem 49 objektů, které musí být detekovány algoritmem. Seznam a počet zastoupení z jednotlivých tříd je v tabulce 3.

Tab. 3 Informace o výskytu dopravních značek v testovacím videu

název	výskyt	název	výskyt
nejvyšší dovolená rychlost 80	2	přikázaný směr vlevo	1
dej přednost v jízdě	3	přikázaný směr vpravo	1
hlavní silnice	4	stop	1
jednosměrný provoz	1	světelné signály	1
konec nejvyšší dovolené rychlosti 80	0	zákaz předjíždění	3
konec hlavní silnice	1	zákaz stání	1
konec všech zákazů	1	zákaz vjezdu	1
křižovatka	1	zákaz vjezdu (jednosměrný provoz)	1
kruhový objezd	1	zákaz zastavení	1
pozor přechod pro chodce	1	železniční přejezd bez závor	2
přechod pro chodce	2	železniční přejezd se závorami	2
přednost před protijedoucími vozidly	1	zelená	5
přednost protijedoucích vozidel	1	oranžová	5
přikázaný směr přímo	1	červená	4

Pro zjednodušení vyhodnocení modelu nebudou falešně pozitivní detekce zahrnuty. Příklady těchto detekcí jsou na *Obr. 42*.

Obr. 42 Falešně pozitivní detekce na dopravní značce a v odrazu na kapotě automobilu

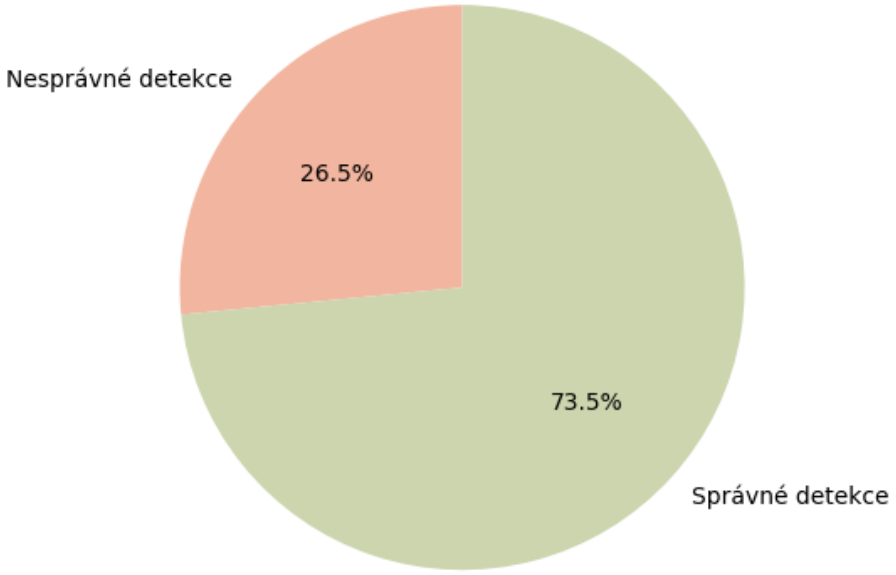
5.3 VYHODNOCENÍ MODELU NA POŘÍZENÉM ZÁZNAMU

Pro vyhodnocení modelu bude provedena srovnávací analýza mezi vlastním algoritmem a algoritmem z knihovny YOLO. Tím bude získán objektivní pohled na výkon (z hlediska rychlosti zpracování snímků v reálném čase) a přesnost vytvořeného detekčního modelu. Vyhodnocení proběhne prostřednictvím detailního zkoumání jednotlivých snímků videa (celkem 8087 snímků) a zaznamenáním správnosti a chybnosti detekce na každém snímku. Dále bude provedena analýza možných problémů a omezení vytvořeného modelu, pro lepší porozumění příčinám selhání detekce. Na základě těchto výsledků bude možno posoudit a vyhodnotit účinnost a přesnost modelu a navrhnout případná zlepšení pro budoucí vývoj.

5.3.1 NAVRŽENÝ ALGORITMUS DETEKCE

Z hlediska hodnocení výkonu byla průměrná rychlost zpracování jednoho snímku 26 ms. Uvážíme-li, že na videu je zaznamenáno 30 FPS, tak doba generování jednoho snímku je 33 ms. Jelikož 26 ms je průměrná hodnota, znamená to, že existují hodnoty, které přesahují dobu generování jednoho snímku a z tohoto důvodu by při zpracování v reálném čase mohlo docházet k opožděným nebo vynechaným detekcím.

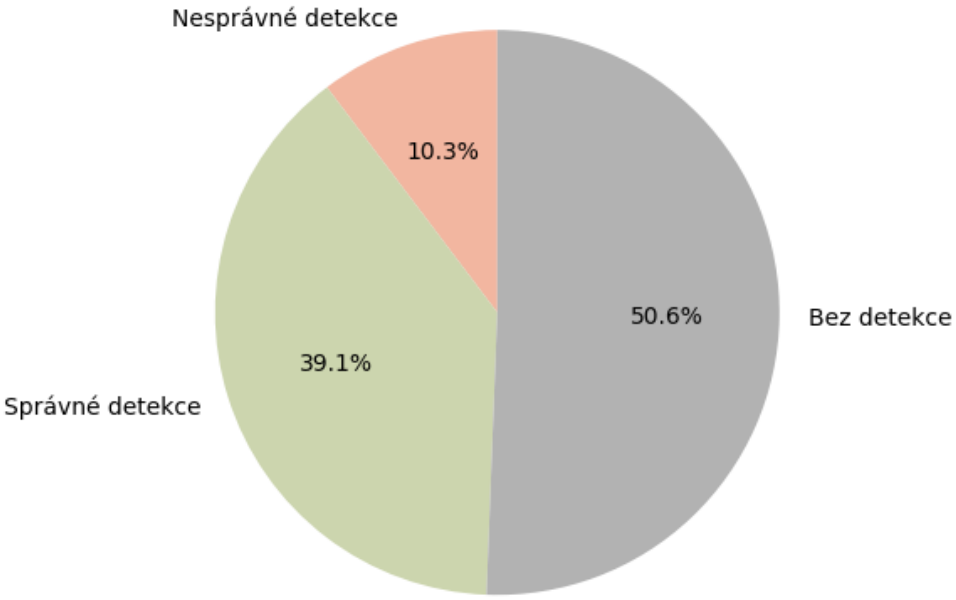
Podíváme-li se na hodnocení modelu z hlediska celkového počtu správných a nesprávných detekcí, z 49 objektů na videu bylo správně detekováno 36 a nesprávně 13 objektů. Tyto hodnoty jsou procentuálně znázorněny na grafu níže.



Obr. 43 Graf správných a chybných detekcí vlastního algoritmu

Díky zkoumání detekcí na jednotlivých snímcích odděleně bylo možné vyhodnotit, na kterém snímku se objekt k detekci objevil a poslední snímek, kdy byl ještě rozpoznatelný. Tyto údaje přinesly do vyhodnocení další proměnnou, a to snímky bez detekce, které se na základě těchto údajů daly dopočítat.

Na *Obr. 44* lze vidět, že z celkového počtu snímků, na kterých se objekty nacházely, bylo 53 % snímků bez detekce. V těchto nedetekovaných snímcích jsou nejvíce zahrnuty situace, kdy byl objekt příliš vzdálený a malý na to, aby ho model rozpoznal. Menší podíl pak tvoří situace, kdy detekce na některých snímcích neproběhly i přes to, že už předtím byly správně klasifikovány. Také zde patří situace, kdy detektor objekt během celé jeho přítomnosti na snímcích nerozpoznal.

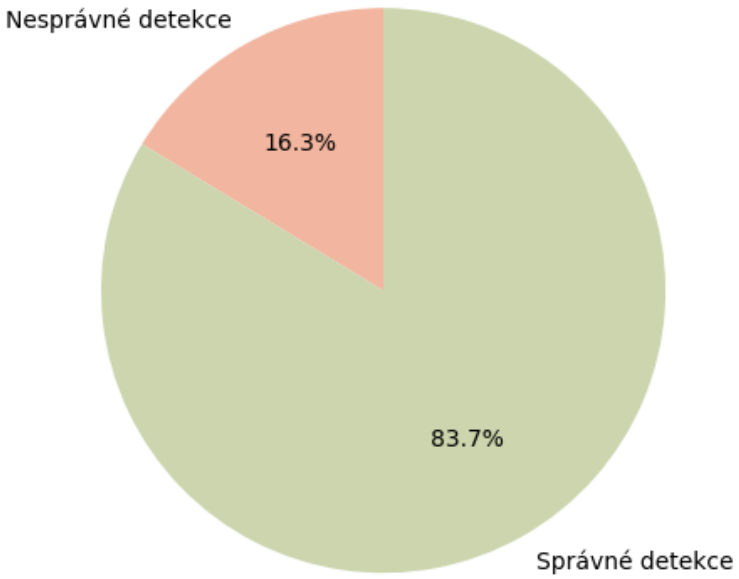


Obr. 44 Graf zahrnující snímky bez detekce v celkovém vyhodnocení vlastního algoritmu

5.3.2 ALGORITMUS DETEKCE Z KNIHOVNY YOLOV7

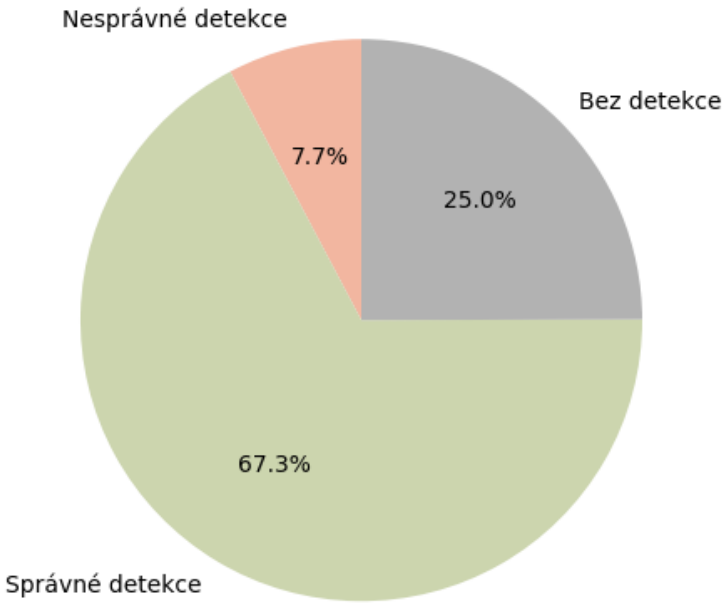
Průměrná rychlost zpracování jednoho snímku pomocí algoritmu z knihovny YOLOv7 byla 13 ms, což je o polovinu kratší čas oproti vlastnímu algoritmu detekce. Tento významný rozdíl v rychlosti naznačuje, že algoritmus YOLOv7 je optimalizovaný pro efektivní zpracování obrazových dat a je vhodný pro aplikace, které vyžadují rychlou odezvu v reálném čase, jako je autonomní řízení vozidel.

Výše zmíněné se také promítlo do samotných detekcí, kde algoritmus YOLOv7 dosáhl vyšší správnosti detekcí než vlastní algoritmus. Z 49 objektů na videu bylo správně detekováno 41 objektů, zbylých 8 bylo detekováno nesprávně. Tyto hodnoty jsou opět procentuálně znázorněny v grafu na *Obr. 45*.



Obr. 45 Graf správných a chybných detekcí algoritmu YOLOv7

Algoritmus YOLOv7 dosahuje lepších výsledků také v případě vyhodnocení po jednotlivých snímcích, které zahrnuje i počet snímků bez detekce objektu. Zatímco navržený algoritmus nedetekoval objekty v 51,3 % takovýchto snímků, algoritmus YOLOv7 nedetekoval pouze 25,4 % (viz *Obr. 46*). Z toho opět vyplývá, že detekce konkrétních objektů algoritmus zvládá z větší vzdálenosti a celkově jsou detekce stálejší.



Obr. 46 Graf zahrnující snímky bez detekce v celkovém vyhodnocení algoritmu YOLOv7

5.3.3 VYHODNOCENÍ OBOU ALGORITMŮ NA KONKRÉTNÍCH DETEKCÍCH

V následující části budou probrány a porovnány detekce na konkrétních případech z videa.

Z výsledků je patrné, že na prodlevu, která vzniká mezi zobrazením dopravní značky na snímku a její detekcí, má výrazný vliv kvalita pořízeného videa. Na *Obr. 47* lze vidět vzdálenost, kdy je dopravní značka rozlišitelná řidičem. Nicméně po jejím přiblížení, které je zobrazeno na *Obr. 48*, je patrné, že rozlišení není dostačující pro nalezení modelem naučeného vzoru.

Obr. 47 Vzdálenost od značky

Obr. 48 Rozlišení dopravní značky na snímku

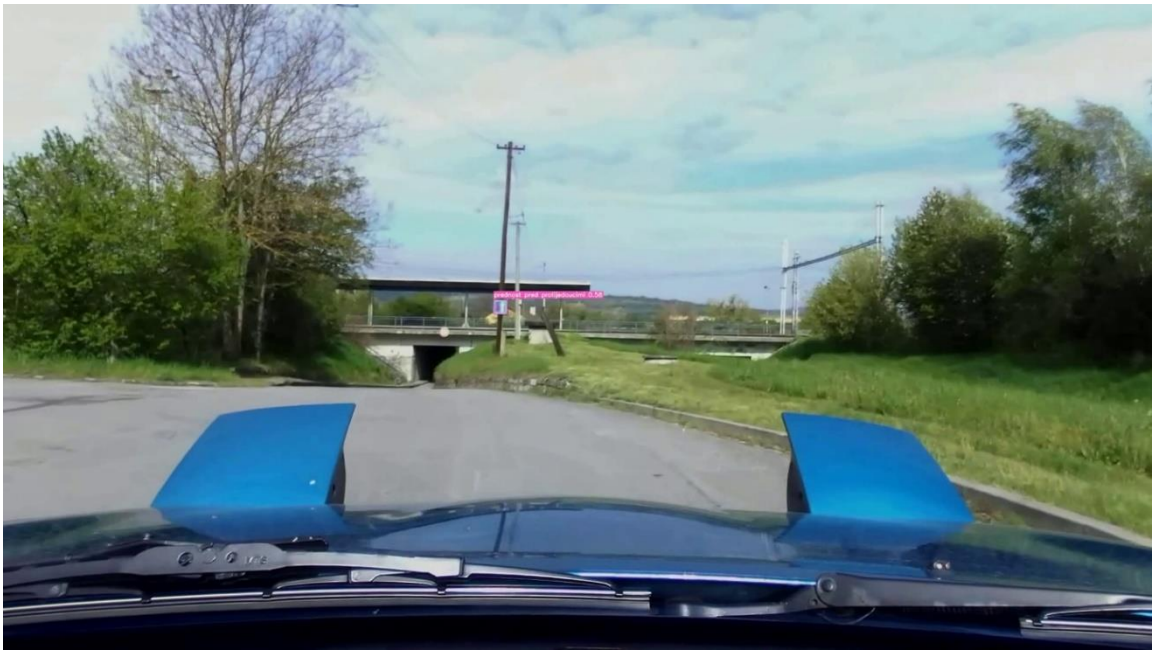
Dalším případem, kde nastal problém s detekcí z velké vzdálenosti, byla dopravní značka „konec hlavní silnice“. Zde se opět opakoval problém s rozlišením, neboť model byl z větší části trénován na kvalitnějších obrázcích. Na *Obr. 49* je ukázka rozlišení značky, která byla z důvodu chybějících detailů pruhů z větší vzdálenosti chybně klasifikována jako „hlavní silnice“. Na *Obr. 50* je pak zobrazen obrázek, na kterém byl pro tuto třídu model trénován.

Obr. 49 Detekovaná oblast z videa

Obr. 50 Obrázek z trénovací sady

Následující dva obrázky představují situaci, na které lze dobře objasnit vliv údajů o snímcích bez detekce na zjištění, z jaké vzdálenosti je model schopen detekovat objekty. Na *Obr. 51* je znázorněna první detekce dopravní značky, která byla detekována navrženým algoritmem na snímku číslo 1472. Na *Obr. 52* je zobrazen snímek 1418, na kterém proběhla první detekce této dopravní značky algoritmem YOLOv7. I bez znalostí těchto informací lze z obrázků na první pohled určit, že algoritmus YOLOv7 detekoval objekt z větší vzdálenosti. Nicméně, při zohlednění snímkové frekvence záznamu 30 FPS a rozdílu 54 snímků mezi detekcemi je možné odvodit, že čas první detekce se mezi algoritmy liší přibližně o 1,8 sekundy.

Obr. 51 První detekce dopravní značky navrženým algoritmem



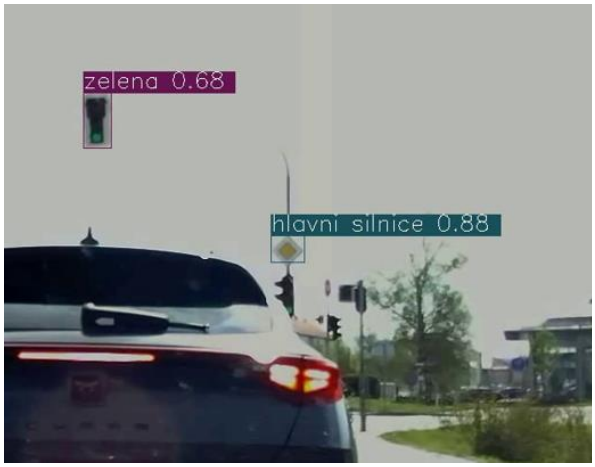
Obr. 52 První detekce dopravní značky algoritmem YOLOv7

Na následujících obrázcích je situace, kdy navržený algoritmus nejprve vůbec nedetekoval značku kruhového objezdu. Z celkových 142 snímků, na kterých se tato značka objevila, detekoval dopravní značku pouze na 35 snímcích, z toho pouze 13 bylo správně. Toto chování bylo vyhodnoceno jako nesprávná detekce. Oproti tomu druhý algoritmus nedetekoval dopravní značku pouze na 16 snímcích a i přesto, že 26 snímků bylo detekováno špatně, celková detekce byla vyhodnocena jako správná. Na zbylých 100 snímcích byl kruhový objekt klasifikován správně. Můžeme tedy říct, že i přes některé nesprávné detekce má druhý algoritmus výrazně vyšší úspěšnost a schopnost rozpoznat daný objekt.

Obr. 53 Neúplná detekce navrženým algoritmem

Obr. 54 Detekce algoritmem YOLOv7

Nyní bude nahlédnuto na situace, kdy oba algoritmy vykazují stejné chování, které je přímo ovlivněno modelem. V první situaci se jedná o případ, kdy část semaforu zakrývá velké vozidlo před kamerou. Semafor není vůbec detekován (viz *Obr. 55* a *Obr. 56*). Tento jev je způsoben tím, že v trénovací datové sadě nebyly případy, na kterých by části semaforů byly zakryté.



Obr. 55 Chybějící detekce překryté části – navržený algoritmus

Obr. 56 Chybějící detekce překryté části – algoritmus YOLOv7

Druhá situace potvrzuje výsledky testování modelu na testovací datové sadě, ve kterých docházelo k záměně klasifikací značek „přikázaný směr vlevo“ a „přikázaný směr vpravo“. Nesprávné výsledky detekcí na testovacím videu jsou zobrazeny na *Obr. 57* a *Obr. 58*.

Obr. 57 Nesprávná detekce navrženým algoritmem

Obr. 58 Nesprávná detekce algoritmem YOLOv7

Na schopnosti modelu správně detekovat naučené vzory dopravního značení, se také nepříznivě projevilo nízké zastoupení některých situací v datové sadě.

Na *Obr. 59* je dopravní značka „pozor přechod pro chodce“, která byla nesprávně klasifikována jako „přechod pro chodce“. Značka je na umístěna na reflexním pozadí, se kterým je v datové sadě obsažena pouze jednou. Oproti tomu přechod pro chodce na reflexním pozadí má větší zastoupení v datové sadě, a proto se model naučil tento vzor lépe.

Obr. 59 Chybná detekce značky „pozor přechod pro chodce“

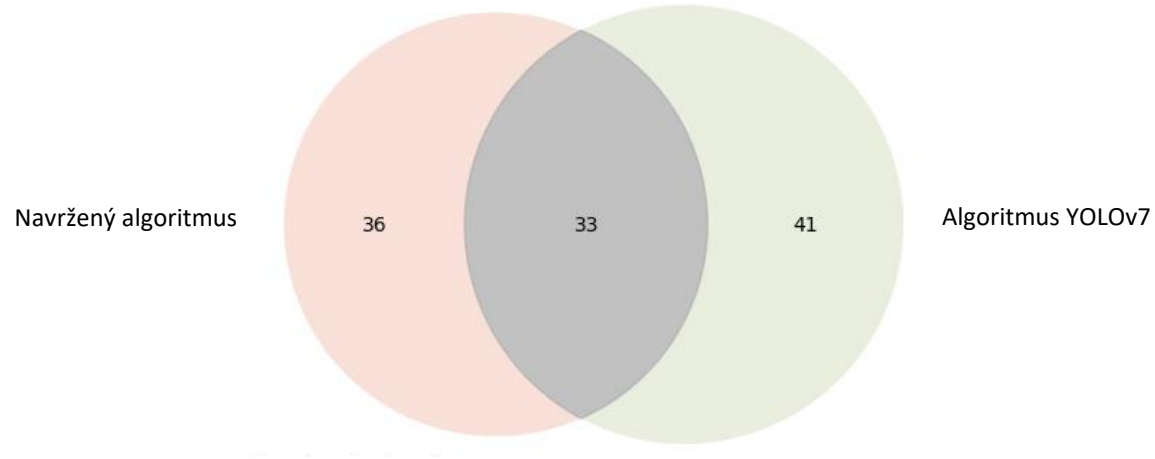
Dalším kritickým bodem pak byla světelná odrazivost od dopravních značek. Na *Obr. 60* je zobrazena situace, kdy dopravní značku ozářilo zapadající slunce. Detekce proběhla až na dalších snímcích.

Obr. 60 Ukázky detekce dopravní značky ozářené sluncem

5.4 SHRNUTÍ VYHODNOCENÍ

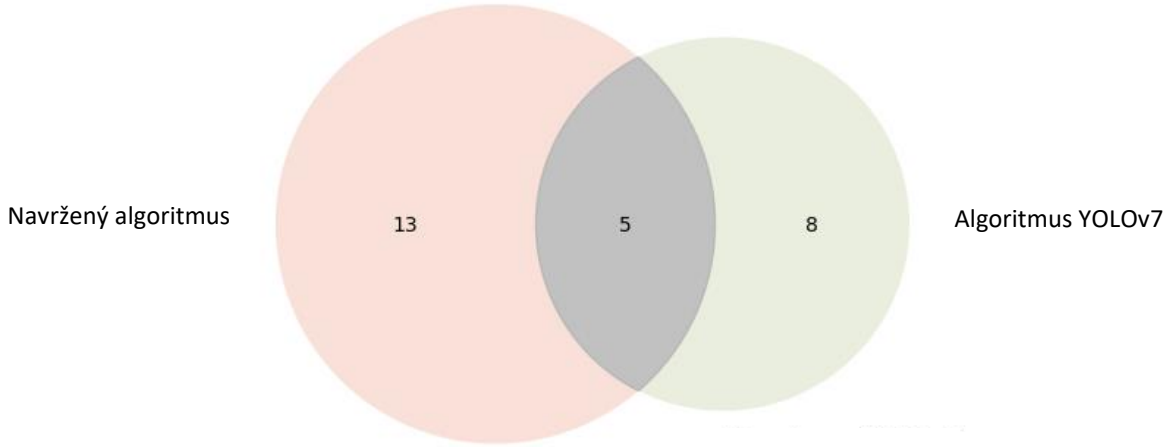
Celkové posouzení schopnosti modelu správné detekce proběhlo pomocí dvou detekčních algoritmů – vlastního navrženého a z knihovny YOLOv7. V *Příloha 1* je tabulka shrnující úspěšnosti detekce na jednotlivých třídách obou těchto algoritmů.

Pro objektivní posouzení natrénovaného modelu byl vytvořen průnik množin správně detekovaných objektů oběma algoritmy (viz *Obr. 61*). Výsledkem je počet objektů, které byly korektně detekovány jak navrženým algoritmem, tak algoritmem YOLOv7.



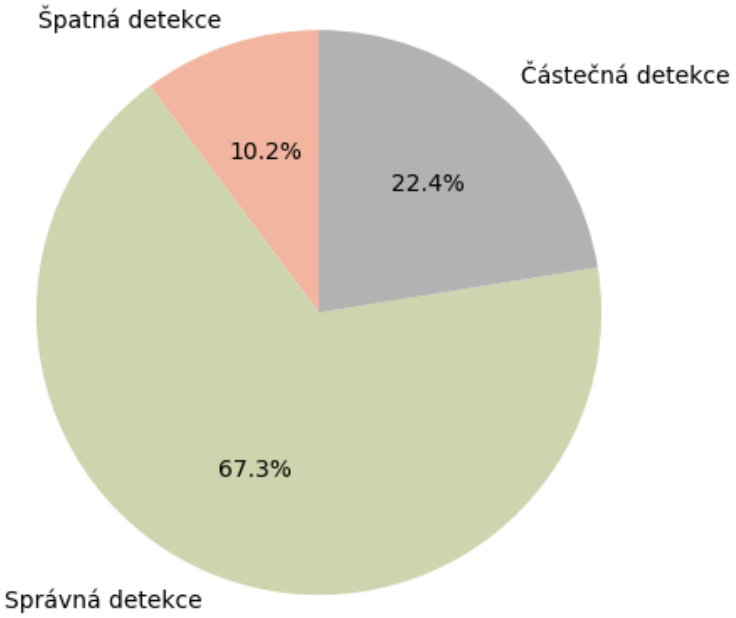
Obr. 61 Průnik množin správných detekcí obou algoritmů

Obdobně byl vytvořen průnik množin špatně detekovaných objektů, který je zobrazen na *Obr. 62* níže.



Obr. 62 Průnik množin nesprávných detekcí obou algoritmů

Celkově tedy bylo oběma algoritmy vyhodnoceno správně 33 objektů a špatně 5 objektů. Tyto hodnoty jsou vyjádřeny v procentech na Obr. 63. Tento graf zahrnuje všechny objekty, které měly být v testovacím videu detekovány. Pojem částečná detekce představuje detekce, které správně detekoval pouze jeden z testovaných algoritmů.



Obr. 63 Graf vyjadřující objektivní zhodnocení modelu na obou algoritmech

ZÁVĚR

Tato práce se zabývala tématem detekce dopravních značek pro autonomní vozidla. Cílem bylo vypracování rešerše o detekci dopravního značení, naprogramování algoritmů potřebných pro detekci a následné experimentální ověření správnosti detekce.

Z rešerše vyplývá, že nejčastěji využívanou metodou v oblasti detekce dopravního značení je v dnešní době metoda založená na hlubokém učení, která je často kombinována s metodou detekce na základě barev a tvaru. Celkově vývoj v této oblasti směřuje k detekci z dat vznikajících fúzí z více senzorů.

Následně byla popsána implementace modelu detekce dopravního značení. Celý tento proces zahrnoval sběr dat pro vytvoření vlastní datové sady, anotaci jednotlivých obrázků a trénování konvoluční neuronové sítě. Jako základ modelu byla zvolena předtrénovaná architektura sítě YOLOv7, na které byl následně model dotrénován pro aplikaci na dopravní značení. Celkově se model naučil rozpoznávat 28 tříd dopravního značení. Tyto třídy zahrnují nejčastější dopravní značky včetně značek železničních přejezdů a semaforů. Ačkoliv bylo do datové sady v malém množství zahrnuto i vodorovné dopravní značení, ověření správnosti detekce nebylo možné na tomto vzorku dat provést.

Pomocí kamery ZED 2 bylo nahráno video, které sloužilo k experimentálnímu ověření správnosti modelu detekce. Jelikož kamera snímala obraz stereoskopicky, bylo nutné použít algoritmus pro komunikaci s kamerou jako nástroj pro úpravu tohoto videa, které bylo následným vstupem pro detekční algoritmus.

Vyhodnocení modelu bylo provedeno na navrženém algoritmu detekce. Aby bylo závěrečné zhodnocení správnosti detekce objektivní k vytvořenému modelu, jeho výkon byl porovnán s algoritmem pro detekci z knihovny YOLOv7. Z celkového počtu dopravního značení ve videu byl model schopen správně detekovat 63,7 %. Nesprávně pak klasifikoval 10,2 %. Zbylých 22,4 % zahrnuje dopravní značení, které zvládl detekovat pouze jeden z algoritmů. V případě detekce pouze algoritmem YOLOv7 bylo správně detekováno 83,7 % ze všech dopravních značek. Schopnost správné detekce navrženého algoritmu byla 73,5 %.

Z experimentálního ověření správnosti detekce vyplývá, že navržený algoritmus je schopen detekovat dopravní značení. Nicméně pro aplikaci v reálném čase, která je u autonomních vozidel zásadní, by bylo potřeba tento algoritmus dále optimalizovat. Ke zlepšení přesnosti by také přispělo rozšíření datové sady, aby se model naučil i vzory, u kterých se objevily problémy s detekcí.

POUŽITÉ INFORMAČNÍ ZDROJE

[1] MAURER, Markus, J. Christian GERDES, Barbara LENZ a Hermann WINNER, ed. *Autonomous Driving* [online]. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016 [cit. 2022-11-19]. ISBN 978-3-662-48845-4. Dostupné z: doi:10.1007/978-3-662-48847-8

[2] YEONG, De, Gustavo VELASCO-HERNANDEZ, John BARRY a Joseph WALSH. Sensor and sensor fusion technology in autonomous vehicles: A review. *Sensors (Basel, Switzerland)* [online]. Switzerland: MDPI, 2021, 21(6), 1-37 [cit. 2022-11-20]. ISSN 1424-8220. Dostupné z: doi:10.3390/s21062140

[3] *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles* [online]. SAE International, 2021 [cit. 2023-04-19]. Dostupné z: https://doi.org/10.4271/J3016_202104

[4] IONITA, Silviu. Autonomous vehicles: from paradigms to technology. *IOP Conference Series: Materials Science and Engineering* [online]. Bristol: IOP Publishing, 2017, 252(1), 12098 [cit. 2023-04-19]. ISSN 1757-8981. Dostupné z: doi:10.1088/1757-899X/252/1/012098

[5] BILIK, Igal, Oren LONGMAN, Shahar VILLEVAL a Joseph TABRIKIAN. The Rise of Radar for Autonomous Vehicles: Signal Processing Solutions and Future Research Directions. *IEEE signal processing magazine* [online]. PISCATAWAY: IEEE, 2019, 36(5), 20-31 [cit. 2022-11-20]. ISSN 1053-5888. Dostupné z: doi:10.1109/MSP.2019.2926573

[6] FRENZEL, Louis E. Radio/Wireless. In: *Electronics Explained* [online]. Elsevier, 2018, s. 159-194 [cit. 2022-11-22]. ISBN 9780128116418. Dostupné z: doi:10.1016/B978-0-12-811641-8.00007-2

[7] *Radar sensors for assistance and automotive applications* [online]. In: . Bosch [cit. 2023-03-12]. Dostupné z: http://www.bosch.co.jp/tms2017/pdf/CC-ProductDataSheet-MRR-LRR-EN.pdf

[8] FANG, Jin, Dingfu ZHOU, Feilong YAN, Tongtong ZHAO, Feihu ZHANG, Yu MA, Liang WANG a Ruigang YANG. Augmented LiDAR Simulator for Autonomous Driving. *IEEE robotics and automation letters* [online]. Ithaca: IEEE, 2020, 5(2), 1930-1937 [cit. 2023-02-08]. ISSN 2377-3766. Dostupné z: doi:10.1109/LRA.2020.2969927

[9] LIDAR, Velodyne. A Guide to Lidar Wavelengths for Autonomous Vehicles and Driver Assistance. Velodyne Lidar [online]. Nov 06, 2018 [cit. 2023-05-25]. Dostupné z: *https://velodynelidar.com/blog/guide-to-lidar-wavelengths/*

[10] CHAI, Zhanxiang, Tianxin NIE a Jan BECKER. *Autonomous driving changes the future*. Singapore: Springer, 2021. ISBN 978-981-15-6727-8.

[11] ZHU, Li. Analyze the Advantages and Disadvantages of Different Sensors for Autonomous Vehicles. *Advances in Economics, Business and Management Research* [online]. Atlantis Press International B.V., 2022, 4 [cit. 2023-02-09]. ISSN 2352-5428. Dostupné z: https://www.atlantis-press.com/proceedings/icssed-22/125973944

[12] *Interfacing ultrasonic sensor with Raspberry PI 4 GPIO* [online]. In: . Robocraze, 2022 [cit. 2023-03-12]. Dostupné z: https://robocraze.com/blogs/post/interfacing-ultrasonic-sensor-with-raspberry-pi-4-gpio

[13] XU, Xianghua, Jiancheng JIN, Shanqing ZHANG, Lingjun ZHANG, Shiliang PU a Zongmao CHEN. Smart data driven traffic sign detection method based on adaptive color threshold and shape symmetry. *Future generation computer systems* [online]. AMSTERDAM: Elsevier B.V, 2019, 94, 381-391 [cit. 2023-02-12]. ISSN 0167-739X. Dostupné z: doi:10.1016/j.future.2018.11.027

[14] NANDI, Dip, A.F.M. SAIFUDDIN SAIF, Prottoy PAUL, Kazi MD. ZUBAIR a Seemanta AHMED SHUBHO. Traffic Sign Detection based on Color Segmentation of Obscure Image Candidates: A Comprehensive Study. *International journal of modern education and computer science* [online]. Hong Kong: Modern Education and Computer Science Press, 2018, 10(6), 35-46 [cit. 2023-02-10]. ISSN 2075-0161. Dostupné z: doi:10.5815/ijmecs.2018.06.05

[15] IPARRAGUIRRE, Olatz, Aiert AMUNDARAIN, Alfonso BRAZALEZ a Diego BORRO. Sensors on the move: Onboard camera-based real-time traffic alerts paving the way for cooperative roads. *Sensors (Basel, Switzerland)* [online]. Switzerland: MDPI, 2021, 21(4), 1-22 [cit. 2023-03-12]. ISSN 1424-8220. Dostupné z: doi:10.3390/s21041254

[16] HARSHAVARDHAN, Saka, Vadlamudi MADHAVI a Sajja TEJASWI. Road and Traffic Sign Detection Using Colour Segmentation. In: *Lecture Notes in Electrical Engineering*. 476. Singapore: Springer Singapore, 2018, , s. 189-200. ISBN 9811082332. ISSN 1876-1100. Dostupné z: doi:10.1007/978-981-10-8234-4_17

[17] HORÁK, Karel, Pavel ČÍP a Daniel DAVÍDEK. Automatic Traffic Sign Detection and Recognition Using Colour Segmentation and Shape Identification. *MATEC Web of Conferences* [online]. EDP Sciences, 68(1), 1-6 [cit. 2023-02-12]. ISBN 2261236X. ISSN 2261-236X. Dostupné z: doi:10.1051/matecconf/20166817002

[18] LU, Eric, Michal GOZDZIKIEWICZ, Kuei-hua CHANG a Jing-mei CIOU. A Hierarchical Approach for Traffic Sign Recognition Based on Shape Detection and Image Classification. *Sensors (Basel, Switzerland)* [online]. Basel: MDPI AG, 2022, **22**(13), 4768 [cit. 2023-02-18]. ISSN 1424-8220. Dostupné z: doi:10.3390/s22134768

[19] HECHRI, Ahmed a Abdellatif MTIBAA. Two-stage traffic sign detection and recognition based on SVM and convolutional neural networks. *IET image processing* [online]. The Institution of Engineering and Technology, 2020, 14(5), 939-946 [cit. 2023-02-18]. ISSN 1751-9659. Dostupné z: doi:10.1049/iet-ipr.2019.0634

[20] GOODFELLOW, Ian, Yoshua BENGIO a Aaron COURVILLE. *Deep learning*. United States of America: MIT Press, 2016. 801 s. ISBN 9780262035613.

[21] CHOLLET, François. *Deep learning with Python*. Second edition. Shelter Island: Manning, 2021, xxiii, 478 stran : barevné ilustrace ; 24 cm. ISBN 978-1-61729-686-4.

[22] AI vs Machine Learning vs Deep Learning: Know the Differences. In: *Simplilearn* [online]. Simplilearn, 2021 [cit. 2023-02-13]. Dostupné z: https://www.simplilearn.com/tutorials/artificial-intelligence-tutorial/ai-vs-machine-learning-vs-deep-learning

[23] ZHANG, Jianming, Zi YE, Xiaokang JIN, Jin WANG a Jin ZHANG. Real-time traffic sign detection based on multiscale attention and spatial information aggregator. *Journal of real-time image processing* [online]. Berlin/Heidelberg: Springer Berlin Heidelberg, 2022, 19(6), 1155-1167 [cit. 2023-02-14]. ISSN 1861-8200. Dostupné z: doi:10.1007/s11554-022-01252-w

[24] SZE, Vivienne, Yu-Hsin CHEN, Tien-Ju YANG a Joel S EMER. *Efficient processing of deep neural networks*. San Rafael: Morgan & Claypool Publishers, 2020, xxi, 319 stran : barevné ilustrace. ISBN 978-1-68173-831-4.

[25] GHOLAMALINEZHAD, Hossein a Hossein KHOSRAVI. *Pooling Methods in Deep Neural Networks, a Review* [online]. 2020 [cit. 2023-02-17]. Dostupné z: doi:10.48550/arxiv.2009.07485

[26] AYYADEVARA, V Kishore a Yeshwanth REDDY. *Modern Computer Vision with PyTorch*. Birmingham, UK: Packt Publishing, 2020. ISBN 978-1-83921-347-2.

[27] ZAFAR, Afia, Muhammad AAMIR, Nazri MOHD NAWI, Ali ARSHAD, Saman RIAZ, Abdulrahman ALRUBAN, Ashit DUTTA a Sultan ALMOTAIRI. A Comparison of Pooling Methods for Convolutional Neural Networks. *Applied sciences* [online]. Basel: MDPI AG, 2022, 12(17), 8643 [cit. 2023-02-17]. ISSN 2076-3417. Dostupné z: doi:10.3390/app12178643

[28] HUANG, Lixing, Jietao DIAO, Hongshan NIE, Wei WANG, Zhiwei LI, Qingjiang LI a Haijun LIU. Memristor Based Binary Convolutional Neural Network Architecture With Configurable Neurons. *Frontiers in neuroscience* [online]. Switzerland: Frontiers Research Foundation, 2021, 15, 639526-639526 [cit. 2023-02-17]. ISSN 1662-4548. Dostupné z: doi:10.3389/fnins.2021.639526

[29] ZHANG, Hang. Review on One-Stage Object Detection Based on Deep Learning. *EAI Endorsed Transactions on e-Learning* [online]. European Alliance for Innovation (EAI), 2022, **7**(23) [cit. 2023-02-18]. Dostupné z: doi:10.4108/eai.9-6-2022.174181

[30] YANG, Zhehui, Chenbo ZHAO, Hiroya MAEDA a Yoshihide SEKIMOTO. Development of a Large-Scale Roadside Facility Detection Model Based on the Mapillary Dataset. *Sensors (Basel, Switzerland)* [online]. Switzerland: MDPI AG, 2022, **22**(24), 9992 [cit. 2023-02-18]. ISSN 1424-8220. Dostupné z: doi:10.3390/s22249992

[31] REN, Shaoqing, Kaiming HE, Ross GIRSHICK a Jian SUN. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *ArXiv.org* [online]. Ithaca: Cornell University Library, arXiv.org, 2016 [cit. 2023-02-18].

[32] QIAN, Rongqiang, Qianyu LIU, Yong YUE, Frans COENEN a Bailing ZHANG. Road surface traffic sign detection with hybrid region proposal and fast R-CNN. In: *2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)* [online]. IEEE, 2016, s. 555-559 [cit. 2023-02-19]. ISBN 978-1-5090-4093-3. Dostupné z: doi:10.1109/FSKD.2016.7603233

[33] TABERNIK, Domen a Danijel SKOCAJ. Deep Learning for Large-Scale Traffic-Sign Detection and Recognition. *IEEE transactions on intelligent transportation systems* [online]. IEEE, 2020, 21(4), 1427-1440 [cit. 2023-02-18]. ISSN 1524-9050. Dostupné z: doi:10.1109/TITS.2019.2913588

[34] REDMON, Joseph, Santosh DIVVALA, Ross GIRSHICK a Ali FARHADI. You Only Look Once: Unified, Real-Time Object Detection. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* [online]. IEEE, 2016, , s. 779-788 [cit. 2023-02-18]. ISBN 9781467388511. ISSN 1063-6919. Dostupné z: doi:10.1109/CVPR.2016.91

[35] SOLAWETZ, Jacob. *What is YOLOv8? The Ultimate Guide.* [online]. In: . Roboflow, 2023 [cit. 2023-02-18]. Dostupné z: https://blog.roboflow.com/whats-new-in-yolov8/

[36] AZIZ, Lubna, Md. HAJI SALAM, Usman SHEIKH a Sara AYUB. Exploring Deep Learning-Based Architecture, Strategies, Applications and Current Trends in Generic Object Detection: A Comprehensive Review. *IEEE access* [online]. Piscataway: IEEE, 2020, 8, 170461-170495 [cit. 2023-02-18]. ISSN 2169-3536. Dostupné z: doi:10.1109/ACCESS.2020.3021508

[37] HU, Jun-ying, C.-J. SHI a Jiang-she ZHANG. Saliency-based YOLO for single target detection. *Knowledge and information systems* [online]. London: Springer London, 2021, 63(3), 717-732 [cit. 2023-02-21]. ISSN 0219-1377. Dostupné z: doi:10.1007/s10115-020-01538-0

[38] YANG, Zhonglai. Intelligent Recognition of Traffic Signs Based on Improved YOLO v3 Algorithm. *Mobile information systems* [online]. Amsterdam: Hindawi, 2022, 1-11 [cit. 2023-02-18]. ISSN 1574-017X. Dostupné z: doi:10.1155/2022/7877032

[39] WANG, Yongjie, Miaoyuan BAI, Mingzhi WANG, Fengfeng ZHAO a Jifeng GUO. Multiscale Traffic Sign Detection Method in Complex Environment Based on YOLOv4. *Computational intelligence and neuroscience* [online]. New York: Hindawi, 2022, 1-15 [cit. 2023-02-10]. ISSN 1687-5265. Dostupné z: doi:10.1155/2022/5297605

[40] SONG, Weizhen a Shahrel SUANDI. TSR-YOLO: A Chinese Traffic Sign Recognition Algorithm for Intelligent Vehicles in Complex Scenes. *Sensors (Basel, Switzerland)* [online]. Switzerland: MDPI AG, 2023, 23(2), 749 [cit. 2023-02-18]. ISSN 1424-8220. Dostupné z: doi:10.3390/s23020749

[41] WU, Jianjun a Shaowen LIAO. Traffic Sign Detection Based on SSD Combined with Receptive Field Module and Path Aggregation Network. *Computational intelligence and neuroscience* [online]. United States: Hindawi, 2022, 4285436-13 [cit. 2023-02-18]. ISSN 1687-5265. Dostupné z: doi:10.1155/2022/4285436

[42] LIU, Wei, Dragomir ANGUELOV, Dumitru ERHAN, Christian SZEGEDY, Scott REED, Cheng-yang FU a Alexander BERG. SSD: Single Shot MultiBox Detector. In: *Computer Vision – ECCV 2016* [online]. Cham: Springer International Publishing, 2016, , s. 21-37 [cit. 2023-02-24]. ISBN 3319464477. ISSN 0302-9743. Dostupné z: doi:10.1007/978-3-319-46448-0_2

[43] CUI, Yaodong, Ren CHEN, Wenbo CHU, Long CHEN, Daxin TIAN, Ying LI a Dongpu CAO. Deep Learning for Image and Point Cloud Fusion in Autonomous Driving: A Review. *IEEE transactions on intelligent transportation systems* [online]. Ithaca: IEEE, 2022, 23(2), 722-739 [cit. 2023-03-04]. ISSN 1524-9050. Dostupné z: doi:10.1109/TITS.2020.3023541

[44] CHOLLET, François a Rudolf PECINOVSKÝ. *Deep learning v jazyku Python: knihovny Keras, Tensorflow*. První vydání. Praha: Grada Publishing, 2019, 328 stran : ilustrace ; 24 cm. ISBN 978-80-247-3100-1.

[45] RAZA, Kazim a Song HONG. Fast and accurate fish detection design with improved yolo-v3 model and transfer learning. *International journal of advanced computer science & applications* [online]. West Yorkshire: Science and Information (SAI) Organization Limited, 2020, 11(2), 7-16 [cit. 2023-05-04]. ISSN 2158-107X. Dostupné z: doi:10.14569/ijacsa.2020.0110202

[46] KANDEL, Ibrahem a Mauro CASTELLI. The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset. *ICT express* [online]. Elsevier B.V, 2020, 6(4), 312-315 [cit. 2023-05-21]. ISSN 2405-9595. Dostupné z: doi:10.1016/j.icte.2020.04.010

[47] PRAMODITHA, Rukshan. *All You Need to Know about Batch Size, Epochs and Training Steps in a Neural Network* [online]. In: . Data Science 365, 2022 [cit. 2023-04-19]. Dostupné z: https://medium.com/data-science-365/all-you-need-to-know-about-batch-size-epochs-and-training-steps-in-a-neural-network-f592e12cdb0a

[48] VERDHAN, Vaibhav. *Computer Vision Using Deep Learning Neural Network Architectures with Python and Keras*. Apress, 2021. ISBN 978-1-4842-6615-1.

[49] BATARSEH, Feras A. a Tuixin YANG. *Data Democracy: At the Nexus of Artificial Intelligence, Software Development, and Knowledge Engineering*. Academic Press, 2020, 266 s. ISBN 978-0-12-818366-3.

[50] GAD, Ahmed Fawzy. Evaluating Object Detection Models Using Mean Average Precision (mAP). In: *Paperspace* [online]. 2020 [cit. 2023-04-24]. Dostupné z: https://blog.paperspace.com/mean-average-precision/

[51] *CUDA C++ Programming Gide* [online]. In: . NVIDIA Corporation & Affiliates, 2023 [cit. 2023-04-02]. Dostupné z: https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html#

[52] WANG, Chien-yao, Alexey BOCHKOVSKIY a Hong-Yuan LIAO. *YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors* [online]. 2022 [cit. 2023-05-03]. Dostupné z: doi:10.48550/arxiv.2207.02696

[53] BOESCH, Gaudenz. *YOLOv7: The Most Powerful Object Detection Algorithm Read more at: https://viso.ai/deep-learning/yolov7-guide/* [online]. In: . viso.ai [cit. 2023-04-02]. Dostupné z: https://viso.ai/deep-learning/yolov7-guide/

[54] ANKA, Andrej. *YOLO v4: Optimal Speed & Accuracy for object detection* [online]. In: . Medium, 2020 [cit. 2023-04-02]. Dostupné z: https://towardsdatascience.com/yolo-v4-optimal-speed-accuracy-for-object-detection-79896ed47b50

[55] CHEN, Jincheng, Shoujun BAI, Guoyang WAN a Yunfei LI. Research on YOLOv7-based defect detection method for automotive running lights. *Systems science & control engineering* [online]. Taylor & Francis, 2023, 11(1) [cit. 2023-04-04]. ISSN 2164-2583. Dostupné z: doi:10.1080/21642583.2023.2185916

[56] SKELTON, James. *Step-by-step instructions for training YOLOv7 on a Custom Dataset* [online]. In: . 2022 [cit. 2023-04-03]. Dostupné z: https://blog.paperspace.com/train-yolov7-custom-data/#:~:text=The%20YOLOv7%20Annotation%20Format&text=YOLOv7%20expects%20annotations%20for%20each,the%20bounding%20boxes%20shown%20above.

[57] SHAH, Deval. *COCO Dataset: All You Need to Know to Get Started* [online]. In: . v7, 2023 [cit. 2023-04-03]. Dostupné z: https://www.v7labs.com/blog/coco-dataset-guide

[58] *NVIDIA T4 70W Low Profile PCIe GPU Accelerator* [online]. NVIDIA, 2020 [cit. 2023-04-04]. Dostupné z: https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/tesla-t4/t4-tensor-core-product-brief.pdf

[59] HUSSAIN, Ambreen, Bidushi BARUA, Ahmed OSMAN, Raouf ABOZARIBA a A. Taufiq ASYHARI. Low Latency and Non-Intrusive Accurate Object Detection in Forests. *IEEE Symposium Series on Computational Intelligence (SSCI)* [online]. IEEE, 2021, 6 [cit. 2023-04-19]. Dostupné z: doi:10.1109/SSCI50451.2021.9660175

[60] *ZED 2* [online]. Stereolabs [cit. 2023-04-17]. Dostupné z: https://www.stereolabs.com/zed-2/

[61] ROSEBROCK, Adrian. Intersection over Union (IoU) for object detection. In: *PyImageSearch* [online]. 2016 [cit. 2023-04-17]. Dostupné z: https://pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/

[62] PRAKASH, Jatin. *Non Maximum Suppression: Theory and Implementation in PyTorch* [online]. In: . 2021 [cit. 2023-04-17]. Dostupné z: https://learnopencv.com/non-maximum-suppression-theory-and-implementation-in-pytorch/

[63] Vyhláška Ministerstva dopravy a spojů. In: *: Příloha č. 3 k vyhlášce č. 30/2001 Sb.* [online]. Ministerstvo dopravy a spojů, 2001 [cit. 2023-02-03]. Dostupné z: https://www.zakonyprolidi.cz/cs/2001-30#

SEZNAM POUŽITÝCH ZKRATEK A SYMBOLŮ

1D	jednorozměrný
2D	dvourozměrný
3D	trojrozměrný
ADAS	Advanced Driver Assistance Systems
AP	Average Precision
CNN	Convolutional Neural Network
CUDA	Compute Unifies Device Architecture
E-ELAN	Extended Efficient Layer Aggregation Network
FP	False positive
FPN	Feature Pyramid Network
GHz	gigahertz
GPU	Graphics Processing Unit
HOG	Histogram of Oriented Gradients
HSV	Hue Saturation Value
IoU	Intersection over Union
kHz	kilohertz
LiDAR	Light Detection and Ranging
mAP	Mean Average Precision
ms	milisekunda
OpenCV	Open Source Computer Vision Library
R-CNN	Region-based Convolutional Neural Network
ReLU	Rectified Linear Unit
RGB	Red, green, blue
RPN	Region Proposal Network
SAE	Society of Automobile Engineers
SiLU	Sigmoid Linear Unit
SSD	Single Shot Multibox Detector
SVM	Support Vector Machine
TP	True positive
tzv	takzvaný
YOLO	You Only Look Once

SEZNAM PŘÍLOH

Název	Výskyt	Počet správných detekcí – navržený algoritmus	Počet správných detekcí – algoritmus YOLOv7
nejvyšší dovolená rychlost 80	2	0	2
dej přednost v jízdě	3	3	3
hlavní silnice	4	4	4
jednosměrný provoz	1	1	1
konec hlavní silnice	1	1	0
konec všech zákazů	1	1	1
křižovatka	1	1	1
kruhový objezd	1	0	1
pozor přechod pro chodce	1	0	0
přechod pro chodce	2	2	2
přednost před protijedoucími vozidly	1	1	1
přednost protijedoucích vozidel	1	1	1
přikázaný směr přímo	1	1	1
přikázaný směr vpravo	1	0	0
přikázaný směr vlevo	1	1	0
stop	1	1	1
světelné signály	1	1	1
zákaz předjíždění	3	2	2
zákaz stání	1	1	1
zákaz vjezdu	1	0	1
zákaz vjezdu (jednosměrný provoz)	1	1	1
zákaz zastavení	1	1	1
železniční přejezd bez závor	2	2	2
železniční přejezd se závorami	2	1	2
zelená	5	3	3
oranžová	5	2	4
červená	4	4	4
Celkem	49	**37**	**41**

Příloha 1 Tabulka shrnující úspěšnosti detekce na jednotlivých třídách