Univerzita Pardubice

Fakulta elektrotechniky a informatiky

Model vozidla s automatickým sledováním trasy pomocí konvoluční neuronové sítě

Bakalářská práce

2024 Jakub Zahradník

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2023/2024

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Jakub Zahradník**
Osobní číslo:	**I21073**
Studijní program:	**B0714A150008 Automatizace**
Téma práce:	**Model vozidla s automatickým sledováním trasy pomocí konvoluční neuronové sítě**
Zadávající katedra:	**Katedra řízení procesů**

Zásady pro vypracování

Cílem práce je navrhnout a vytvořit autonomní vozítko řízené pomocí výstupů konvolučních neuronových sítí (CNN), které bude schopné autonomně sledovat vyznačenou trasu.
V teoretické části práce bude provedena rešerše problematiky autonomních vozidel, se zaměřením na základní technologie strojového vidění pro jejich autonomii používané v této oblasti. Dále se student zaměří na metody detekce a sledování trasy s využitím konvolučních neuronových sítí. Závěrem teoretické části bude návrh metodiky pro sběr a přípravu dat pro trénování neuronové sítě, včetně popisu anotace a rozdělení dat.
Praktická část se bude zabývat konstrukcí vozítka, výběrem vhodných komponentů a sestavením modelu vozítka s ohledem na integraci kamerového systému a řídící jednotky. Dále bude popsána implementace softwaru pro sběr dat během manuálního ovládání vozítka. Hlavním cílem praktické části bude spočívat v návrhu, implementaci a trénování CNN na základě předem nasbíraných dat. Po natrénování sítě bude model otestován na vozítku v reálných podmínkách na neznámé trase a vyhodnocen z hlediska přesnosti a spolehlivosti sledování trasy. V závěru praktické části student zhodnotí dosažené výsledky.

Rozsah pracovní zprávy: **cca 40 stran**
Rozsah grafických prací:
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

MAIXNER, Ladislav. *Mechatronika: učebnice*. Brno: Computer Press, [2006]. Učebnice (Computer Press). ISBN 80-251-1299-3.
GONZALEZ, Rafael C. a Richard E. WOODS, [2018]. Digital image processing. Fourth edition. New York: Pearson. ISBN 978-013-3356-724.

Vedoucí bakalářské práce: **Ing. Dominik Štursa**
Katedra řízení procesů

Datum zadání bakalářské práce: **15. prosince 2023**
Termín odevzdání bakalářské práce: **10. května 2024**

L.S.

Ing. Zdeněk Němec, Ph.D. v.r.
děkan

Ing. Daniel Honc, Ph.D. v.r.
vedoucí katedry

V Pardubicích dne 12. ledna 2024

Prohlašuji:

Práci s názvem Model vozidla s automatickým sledováním trasy pomocí konvoluční neuronové sítě jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 7/2019 Pravidla pro odevzdávání, zveřejňování a formální úpravu závěrečných prací, ve znění pozdějších dodatků, bude práce zveřejněna prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 17. 5. 2024

Jakub Zahradník v.r.

PODĚKOVÁNÍ

Rád bych poděkoval Ing. Dominikovi Štursovi za odborné vedení, za pomoc a podnětné rady při zpracování této práce a zároveň za zapůjčení komponent.

ANOTACE

Tato bakalářská práce se zaměřuje na návrh a implementaci autonomního vozidla s využitím konvolučních neuronových sítí pro automatické sledování trasy. Teoretická část zkoumá principy strojového vidění a senzoriky pro autonomní řízení. Dále jsou popsány principy konvolučních neuronových sítí a jejich aplikace pro metody detekce a sledování trasy. Je navržena metodika pro sběr dat a trénování sítě. Praktická část se zabývá konstrukcí vozidla, výběrem komponent a implementací softwaru pro sběr dat, trénování TensorFlow modelu a inferenci. Natrénovaný model je testován v reálných podmínkách a vyhodnocen z hlediska přesnosti a spolehlivosti sledování trasy.

KLÍČOVÁ SLOVA

autonomní vozidlo, konvoluční neuronové sítě, strojové vidění, sledování trasy, TensorFlow

TITLE

Model of a vehicle with automatic track following using a convolutional neural network

ANNOTATION

This bachelor thesis focuses on the design and implementation of an autonomous vehicle using convolutional neural networks for automatic track following. The theoretical part explores the principles of machine vision and sensors used for autonomous driving. The principles of convolutional neural networks and their application to track detection and following methods are also described. A methodology for data collection and training of the network is proposed. The practical part deals with vehicle design, selection of components and software implementation of data collection, training a TensorFlow model and inference. The trained model is tested in real conditions and evaluated in terms of accuracy and reliability of track following.

KEYWORDS

autonomous vehicle, convolutional neural networks, machine vision, track following, TensorFlow

OBSAH

SEZNAM ILUSTRACÍ A TABULEK

SEZNAM ZKRATEK A ZNAČEK

AD – Analog-to-digital

ADAS – Advanced Driver Assistance Systems

ALV – Autonomous land vehicle

ALVINN – An Autonomous Land Vehicle in a Neural Network

CCD – Charge-coupled device

CMOS – Complementary Metal–Oxide–Semiconductor

CMY – Cyan magenta yellow

CMYK – Cyan magenta yellow key

CNN – Convolutional neural network

DARPA – Defense Advanced Research Projects Agency

DCT – Dicrete cosine trasformation

ELU – Exponential linear unit

GPS – Global Positioning system

GPU – Graphical processing unit

HSI – Hue saturation intensity

IMU – Inertial measurement unit

LSTM – Long short term memory

MSE – Mean squared error

ReLU – Rectified linear unit

RGB – Red green blue

RL – Reinforcement learning

RNN – Recurrent neural network

TCN – Time convolutional network

ÚVOD

V posledních letech je výrazným trendem prosazování aplikací strojového učení, převážně založených na neuronových sítích, k řešení široké škály technických problémů. Zejména v oblasti strojového vidění dominují konvoluční neuronové sítě (CNN), které v mnoha ohledech nahradily předchozí metody pro extrakci atributů z obrazových dat, obvykle založených na ručně navržených filtrech. Tyto metody byly do jisté míry účinné, ale často selhávaly při zacházení s regularitami obrazových dat, jako jsou např. posuny, změny měřítka, změny osvětlení nebo rotace.

CNN přinesly v oblasti strojového učení revoluci, díky schopnosti automaticky se učit hierarchické vlastnosti přímo z hodnot pixelů obrazu, a to zejména díky objevu algoritmu zpětného šíření chyby. V mnoha úlohách překonaly tradiční metody a staly se stěžejním nástrojem v oblasti strojového vidění.

Úspěch této architektury vedl k rozsáhlému využití v aplikacích jako je zpracování obrazu, konkrétně např. pro rozpoznávání objektů či klasifikaci scén. S postupným rozvojem se CNN začaly uplatňovat i v oblasti autonomních vozidel, kde obvykle slouží k identifikaci okolních objektů pro následné rozhodování o výběrů akčních zásahů pro řízení vozidla. V některých případech je možné pomocí CNN konsolidovat celý proces řízení do jediného systému.

Cílem této práce je vytvořit end-to-end systém pro automatické sledování trasy pomocí CNN. Systém bude přijímat aktuální snímek trasy jako vstup a vracet hodnotu zatáčení jako výstup. V první řadě bude navržena metodika pro sběr dat, pomocí kterých poté bude natrénován CNN model. Dále bude vozidlo testováno na neznámé trase a vyhodnoceno z hlediska spolehlivosti sledování trasy. Kromě softwarové implementace bude navržen i fyzický model vozidla pro ověření funkčnosti systému.

1 REŠERŠE PROBLEMATIKY AUTONOMNÍCH VOZIDEL

Autonomní vozidlo lze obecně definovat jako vozidlo, které je schopné vnímat svoje okolí a operovat bez lidské účasti. Při dosažení plné autonomie lidský pasažér nemusí v žádném okamžiku převzít kontrolu nad vozidlem, ani nemusí být ve vozidle přítomen (What is an Autonomous Car, 2024). Autonomní vozidlo konečné úrovně SAE L5 by mělo dosahovat, nebo překračovat mobilní schopnosti lidských řidičů. Této úrovně zatím nebylo dosaženo. Úrovně SAE budou detailněji probrány v kapitole 1.2.

Tato rešerše se zaměřuje na historický vývoj autonomních vozidel, současný stav tohoto odvětví a klíčové principy senzoriky a strojového vidění, které umožňují autonomii.

1.1 HISTORIE A SOUČASNOST AUTONOMNÍCH VOZIDEL

1.1.1 Počátky autonomních vozidel (1950–1980)

Historie autonomních vozidel sahá až do padesátých let dvacátého století. Tehdy však byly počítače příliš velké a měly omezenou schopnost zpracovávat vizuální a senzorická data v reálném čase, což bránilo jejich přímé integraci do mobilních vozidel.

V roce 1953 byl v laboratoři společnosti Radio Corporation of America vyvinut prototyp autonomního vozidla, kterého bylo možné navigovat pomocí signálů vedených po podlaze laboratoře.

Tento systém byl později v roce 1957 rozšířen do podoby experimentální elektronické dálnice o délce 121 metrů. K řízení vozidel sloužila kombinace rádiové komunikace a signálů vedených po vozovce. Toto řešení však nebylo implementováno ve veřejné dopravě kvůli vysokým finančním nákladům a nedostatečné škálovatelnosti.

1.1.2 Rozvoj v 80. letech (1980–1989)

S rozvojem výpočetní síly počítačů v 80. letech vzniklo několik projektů zabývajících se autonomními vozidly, která využívala pokročilé technologie jako LiDAR, strojové vidění a umělé neuronové sítě. Jedním z těchto projektů bylo Autonomní pozemní vozidlo (ALV), financované Agenturou pro pokročilé obranné výzkumy (DARPA) ve Spojených státech, které úspěšně demonstrovalo autonomní sledování silnice. Toto vozidlo využívalo kombinaci strojového vidění, LiDARu a robotického řízení k navigaci a ovládání. Byla dosažena rychlost 3,1 km/h (Lowrie, 1985).

V roce 1989 bylo na Carnegie Mellon University představeno první úspěšné využití umělých neuronových sítí pro autonomní řízení vozidla. Projekt známý jako ALVINN zavedl inovativní přístup s třívrstvou neuronovou sítí, která byla trénována pomocí algoritmu zpětné propagace chyby. Proces se opíral o vstupní data získaná ze snímků kamery a laserového měřiče vzdálenosti. Výstupem této sítě byl úhel natočení volantu (Pomerleau, 1989).

1.1.3 Průlom v 90. letech (1990–1999)

V devadesátých letech minulého století se v USA uskutečnily snahy, jako byl zákon o autorizaci dopravy ISTEA a projekty jako Demo '97 v San Diegu (Bishop, 2005), které představily první

demonstrace automatizovaných systémů vozidel. Navzdory slibným pokrokům vedla rozpočtová omezení ke zrušení některých iniciativ, například projektu samořízeného vozidla v Jižní Koreji.

V počátcích druhého tisíciletí podnítily další výzkum a vývoj projekty financované armádou, jako byly výzvy agentury DARPA. Projekt Navlab Carnegie Mellon University a projekty VaMP a Vita-2 společnosti Daimler-Benz prokázaly významné úspěchy v oblasti technologie autonomního řízení.

1.1.4 Vývoj v 21. století (2000–dosud)

Vývoj hlubokého strojového učení od roku 2010 výrazně posílil zájem předních automobilek, včetně např. firem Google, Tesla nebo Audi, o pokroky v oblasti autonomních vozidel a jejich veřejné zkoušky (Ni, 2020). V roce 2015 vozidlo Audi A7 úspěšně absolvovalo autonomní jízdu z kalifornského Silicon Valley do Las Vegas, což představovalo přibližně 900 km dlouhou trasu (Yang, 2017). Technologie Drive Pilot, vyvinutá společností Mercedes a integrovaná do komerčně dostupných modelů od roku 2022, umožňuje řidičům delegovat řízení vozidla za určitých podmínek (Witman, 2022). Podobně společnost Tesla přináší službu Autopilot, která poskytuje určitou míru autonomie řízení (Mit, 2020).

Navzdory pokroku přetrvávají výzvy, včetně právních a bezpečnostních aspektů, což dokazují incidenty, jako byla první hlášená smrtelná nehoda chodce a samořízeného vozidla v roce 2018 (Death of Elaine Herzberg, 2018). Pokračující výzkum, technologický pokrok a regulační úsilí však nadále směřují vývoj autonomních vozidel k bezpečnější a efektivnější budoucnosti.

1.2 ÚROVNĚ AUTONOMIE

Autonomní vozidla jsou klasifikovány do různých úrovní autonomie, které odlišují míru zapojení lidského řidiče a úroveň zodpovědnosti vozidla. Podle standartu asociace Society Of Automotive Engineers (SAE) se autonomie vozidel dělí do šesti skupin (Shuttleworth, 2019). Od úrovně 0, kde je lidský řidič plně zodpovědný, až po úroveň 5, kde dokáže vozidlo samo řídit bez potřeby lidského zásahu. Tyto úrovně autonomie poskytují rámec pro pochopení schopností a omezení autonomních vozidel a usnadňují postupný přechod k plné autonomii na silnicích.

Úroveň 0 (L0) charakterizuje absenci autonomie, kde řízení zcela závisí na lidské interakci. Přechod na úroveň 1 (L1) zahrnuje implementaci systému Advanced Driver Assistance Systems

(ADAS). Tato fáze umožňuje vozidlu částečnou autonomii, kdy může systém ADAS samostatně ovládat buď zatáčení nebo akceleraci. Příkladem úrovně L1 je adaptivní tempomat.

Pokrok na úroveň 2 (L2) znamená, že vozidlo získává schopnost autonomního řízení ve dvou rovinách: zatáčení a akcelerace. Přesto je řidič povinen monitorovat chování vozidla a být připraven zasáhnout, pokud je to nutné.

Přechod na úroveň 3 (L3) označuje podmínečné automatické řízení, kde vozidlo je schopno samostatného řízení, ale může vyžádat zásah řidiče v určitých situacích. Úroveň 4 (L4) představuje vozidla, která jsou schopna plně autonomního řízení za předpokladu, že jsou splněny určité podmínky. V konečné úrovni 5 (L5) jsou vozidla schopna plné autonomie, což znamená, že nevyžadují žádnou lidskou intervenci a jsou schopna řídit bez lidského dohledu.

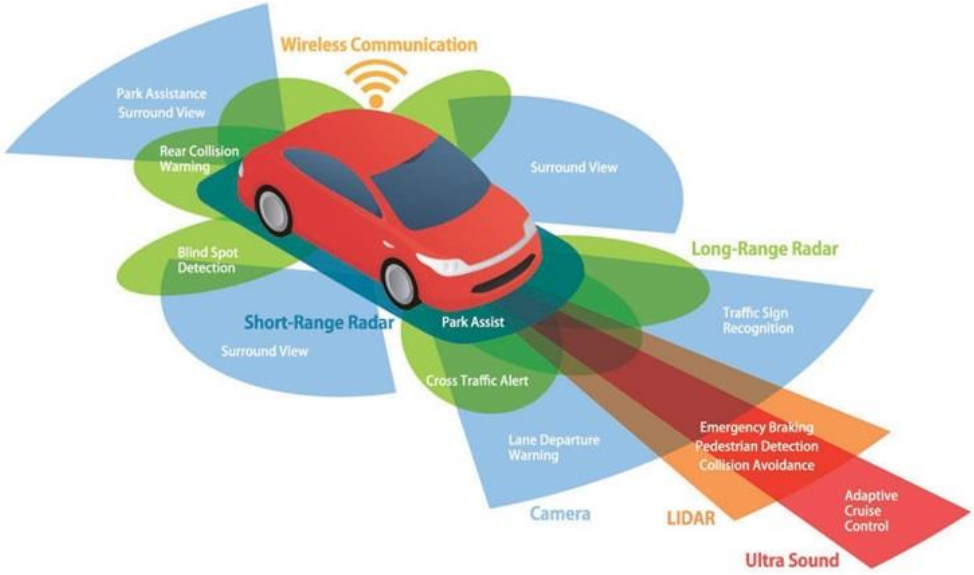
Implementace autonomních vozidel úrovně 5 představuje významnou výzvu v oblasti automobilového průmyslu a technologií. Tato vozidla mají za cíl dosáhnout mobilních schopností srovnatelných nebo přesahujících lidské řidiče. Musí zvládnout různorodé silniční, povětrnostní a dopravní podmínky, což vyžaduje schopnost adaptace na nejrůznější scénáře jízdy.

Lidské vnímání je schopné zpracovat a interpretovat rozsáhlé množství senzorických dat, které poté využívá k rozhodování v reálném čase, opírajíc se o minulé zkušenosti a predikce budoucího vývoje situace. Pro dosažení úrovně autonomie 5 je pravděpodobně nezbytné, aby autonomní vozidla byla schopna modelovat okolní svět s pomocí samoučících se algoritmů (LeCun).

1.3 SENZORY PRO AUTONOMNÍ VOZIDLA

Autonomní vozidla se spoléhají na řadu senzorů, které jim umožňují vnímat okolní prostředí. Mezi tyto senzory patří radar, vysokorozlišovací kamery, LiDAR, 2 D laserové skenery, GPS, IMU a ultrazvukové senzory, viz obrázek 1 (Fan, 2021). Ačkoli existují tvrzení, že senzory založené na obrazovém vnímání samy o sobě mohou být dostatečné, výrobci často preferují kombinaci různých senzorových technologií, aby maximalizovali přesnost a spolehlivost vnímání prostředí vozidla, což umožňuje dosáhnout vyšší úrovně autonomie (Schulte-Tigges, 2022). Senzory používané v autonomních vozidlech mají klíčový význam pro pokročilé techniky řízení a umožňují funkce, jako je detekce objektů, detekce vozovky nebo vyhýbání se překážkám (Häne, 2017).

Senzory lze rozdělit do dvou skupin, a to na aktivní a pasivní. Pasivní senzory detekují existující energii, jako je světlo nebo záření, které se odráží od objektů v okolí, zatímco aktivní senzory vysílají vlastní signál a zachytávají jeho odraz. Pasivní senzory jsou již obvykle využívány v automobilovém průmyslu na úrovni SAE L1 nebo L2, například pro asistenci při udržování vozidla v jízdním pruhu nebo pro adaptivní tempomat.



Obrázek 1 – Znázornění různých druhů senzorů pro autonomní vozidla a jejich funkcí (Autonomous Vehicle Sensors - Making Sense of The World, 2021).

1.3.1 Aktivní senzory

1.3.1.1 LiDAR

Jeden z nejvíce používaných aktivních senzorů pro autonomní vozidla je LiDAR, což je zkratka pro Light Detection and Ranging. LiDAR je technologie, která využívá laserové paprsky k získání velmi přesných výškových měření zemského povrchu (Webster, 2010). Princip fungování této technologie spočívá ve vysílání laserových impulzů a měření času, za který se světlo odrazí zpět k senzoru. Pomocí výpočtu doby návratu světla a znalosti rychlosti světla jsou systémy LiDAR schopny určit vzdálenost k objektům (Vierling, 2008). V kontextu autonomního řízení se LiDAR řadí mezi nejvíce využívané senzory vedle kamer a doplňuje radarové a ultrazvukové senzory (Heinzler, 2020).

Význam LiDARu v autonomních vozidlech dále podtrhuje jeho úloha při detekci objektů v reálném čase, kdy nabízí široké zorné pole a vysokou přesnost při rozpoznávání jízdního prostředí (Fan, 2021). Senzory LiDAR pomáhají autonomním vozidlům identifikovat různé situace v jejich okolí, což jim umožňuje činit informovaná rozhodnutí. Kromě toho jsou

nezbytné pro odometrii a mapování pro autonomní vozidla, které usnadňují současnou lokalizaci a mapování a umožňují vytvářet přesné mapy okolního prostředí s vysokým rozlišením.

1.3.1.2 Radar

Dalším typem aktivního senzoru, který může být využit pro autonomní vozidla je radar. Díky vysílání elektromagnetických pulzů a analýze odražených signálů umožňují radarové systémy vozidlům vnímat okolí, detekovat překážky a bezpečně navigovat (Ahmad, 2019). Radarové systémy hrají klíčovou roli při mapování terénu a umožňují autonomním vozidlům vytvářet podrobné 3D mapy svého okolí v reálném čase. Radarové systémy mohou sloužit k podobným účelům jako LiDAR, přičemž jejich výhodou jsou nižší finanční náklady, vykazují však menší přesnost.

1.3.1.3 Ultrazvukové senzory

Ultrazvukové senzory fungují tak, že vysílají vysokofrekvenční zvukové vlny a měří dobu, za kterou se tyto vlny po dopadu na objekt odrazí zpět. Tyto údaje se pak používají k určení vzdálenosti k objektu před snímačem. V kontextu autonomních vozidel se ultrazvukové senzory používají k detekci překážek v blízkosti vozidla. Poskytují nákladově efektivní řešení pro detekci překážek na krátkou vzdálenost a jsou užitečné zejména v nízkorychlostním prostředí, jako jsou parkoviště nebo městská doprava.

Jednou z výhod ultrazvukových senzorů je jejich schopnost detekovat objekty bez ohledu na jejich materiál nebo barvu, na rozdíl od jiných senzorů, které mohou mít s některými povrchy problémy. Ultrazvukové senzory jsou navíc ve srovnání se senzory LiDAR obecně cenově dostupnější, takže jsou praktickou volbou pro aplikace, kde jsou finanční náklady významným faktorem.

Ultrazvukové senzory však mají svá omezení. Jsou účinné pouze na krátkou a střední vzdálenost a na větší vzdálenosti mohou mít problémy s přesností. Jejich výkon navíc mohou ovlivňovat faktory prostředí, jako je teplota, vlhkost a akustické rušení, což může za určitých podmínek omezit jejich spolehlivost (Alonso, 2011).

1.3.2 Pasivní senzory

Pasivní senzory používané pro autonomní vozidla jsou obvykle založené na kamerové technologii. Digitální kamery nejčastěji využívají obrazové senzory typu CCD (*chrge-coupled device*) nebo CMOS (*complementary metal-oxide semiconductor*). Princip těchto senzorů je

založen na konvertování viditelného světla na elektrický signál (What is the difference between CCD and CMOS image sensors, 2010).

Povrch senzorů je obvykle rozdělen na matici pixelů, z nichž každý může samostatně snímat intenzitu přijatého světelného signálu na základě množství náboje nahromaděného v daném místě.

1.3.2.1 CCD senzor

Jak již bylo zmíněno, CCD senzor funguje na principu převodu světla na elektrické signály. Po dopadu fotonů na senzor se vytvoří páry elektronů a děr prostřednictvím fotoelektrického jevu (Janesick, 2001). Elektrony jsou poté shromažďovány do matice pixelů pomocí elektrického pole vytvořeného pod povrchem senzoru.

V dalším kroku jsou sloupce s elektrony postupně přesouvány horizontálními posuvnými registry do vertikálního posuvného registru, který pak zpracovává elektrický náboj jednotlivých pixelů. Následně je náboj zesílen a digitalizován pomocí AD převodníku.

CCD senzory nabízejí vysokou kvalitu obrazu a nízký šum jako své výhody. Jejich nevýhodou je výrazně vyšší spotřeba energie oproti CMOS senzorům, a to až stonásobně (What is the difference between CCD and CMOS image sensors, 2010). Tato vlastnost vyplývá z principu fungování CCD, kdy je náboj přesunován přes povrch senzoru.

1.3.2.2 CMOS senzor

CMOS senzory disponují zesilovacími tranzistory pro každý pixel, což umožňuje individuální čtení každého pixelu a poskytuje větší flexibilitu.

Mezi výhody CMOS senzorů patří nízká spotřeba energie a vysoká rychlost záznamu obrazu, což je ideální zejména pro mobilní zařízení (Bigas, 2006). Rozvoj této technologie přinesl inovace, jako je schopnost dosáhnout vysokých snímkových frekvencí, a to až 10 000 FPS při zachování nízké energetické spotřeby (Kleinfelder, 2001).

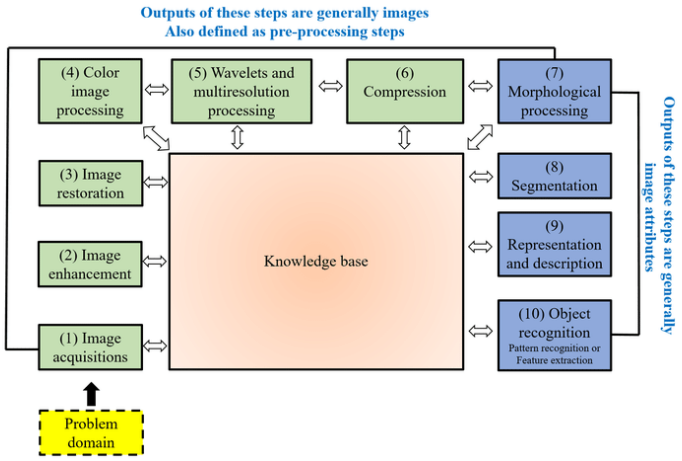
Mezi nevýhody CMOS senzorů patří nižší kvalita obrazu. Umístění několika tranzistorů vedle každé fotodiody způsobuje, že značné množství fotonů dopadá na tyto tranzistory místo na fotodiodu. To obvykle vede k nižší citlivosti na světlo v porovnání s technologií CCD (What is the difference between CCD and CMOS image sensors, 2010).

1.4 PRINCIPY STROJOVÉHO VIDĚNÍ

Podle sdružení Automated Imaging Association (AIA) zahrnuje strojové vidění všechny průmyslové i neprůmyslové aplikace, v nichž kombinace hardwaru a softwaru poskytuje zařízením provozní pokyny při provádění jejich funkcí na základě snímání a zpracování obrazu (What is machine vision, 2023).

1.4.1 Proces strojového vidění

Strojové vidění lze konceptuálně rozčlenit do dvou hlavních etap. První etapa se věnuje akvizici obrazových dat a jejich následnému zpracování. Druhá etapa se soustředí na extrakci atributů a vlastností získaných obrazových informací. Tato fáze strojového vidění je pevně propojena s pokrokem v oblasti algoritmů strojového učení a hlubokého učení, jež umožňují vozidlům inteligentně interpretovat okolní prostředí a adekvátně na něj reagovat. Proces strojového vidění lze vidět na obrázku 2.



Obrázek 2 – Blokové schéma procesu strojového vidění. První část, naznačená zelenými bloky, se věnuje zpracování obrazu. Druhá část procesu se věnuje extrakcí atributů obrazu. (ImageVI's: the software that collects the vegetation indices you need: user manual, 2020).

1.4.2 První etapa strojového vidění – zpracování obrazu

První etapa strojového vidění představuje komplexní soubor procesů, které mohu zahrnovat akvizici obrazu, jeho optimalizaci, obnovení, manipulaci s barvami, transformaci a kompresi. Souhrnně lze první etapu označit jako obrazové zpracování. I když neexistuje striktní hranice mezi obrazovým zpracováním a následujícími kroky strojového vidění, lze obrazové zpracování konceptuálně odlišit jako funkci, která přijímá obraz jako vstup a poskytuje upravený obraz jako výstup (Gonzales, 2018).

1.4.2.1 Akvizice a vylepšení obrazu

Akvizice obrazu označuje proces získání vizuálních dat v digitálním formátu, obecně také může zahrnovat škálování pro dosažení požadovaného formátu. Vylepšení obrazu představuje manipulaci s vizuálními daty, s cílem optimalizovat jejich vhodnost pro konkrétní aplikace. Neexistuje obecný postup pro vylepšení obrazu, protože se jedná o subjektivní proces, hodnocený lidským pozorovatelem, což implikuje individuální preference a účely aplikace (Gonzales, 2018).

1.4.2.2 Obnovení obrazu

Obnovení obrazu reprezentuje další aspekt vylepšení obrazu, avšak na rozdíl od předchozího procesu je založeno na objektivních metodách, které využívají pravděpodobnostní modely pro odhad degradace obrazu. Proces zpracování barevného obrazu zahrnuje zachycení obrazu, jeho digitalizaci do podoby pixelů, případný převod barevných prostorů, úpravy jako ořezání nebo zvětšení, aplikaci různých technik pro zlepšení kvality či extrakci informací, následovanou analýzou a interpretací, jako je identifikace objektů nebo rozpoznání vzorů (Gonzales, 2018).

1.4.2.3 Transformace obrazu

Transformace obrazu umožňuje zachycení a uchování obrazových dat v různých rozlišeních, což umožňuje jejich dekompozici na hierarchii detailů. Tato hierarchie umožňuje efektivní zobrazení a manipulaci s obrazem, jako je například možnost přiblížení detailů. Kromě toho jsou transformace klíčovým nástrojem pro kompresi obrazových dat, což znamená, že je možné ukládat obraz s menším využitím úložného prostoru (Song, 2006).

1.4.2.4 Komprese

Komprese obrazu je klíčová pro zredukování nároků na úložný prostor. Existují ztrátové metody, které odstraňují méně důležité informace z obrazu, jako je kvantizace a transformace (např. DCT), a bezztrátové metody, které zachovávají všechny informace a využívají redundanci dat. Kombinace těchto přístupů umožňuje nalézt optimální kompromis mezi velikostí datového souboru a zachováním kvality obrazu, což je klíčové pro mnoho aplikací včetně digitálního přenosu, ukládání a zpracování obrazových dat.

1.4.3 Druhá etapa strojového vidění – extrakce atributů obrazových dat

Druhá etapa strojového vidění představuje soubor procesů, které obvykle zahrnují morfologické zpracování obrazu, segmentaci, extrakci vlastností a klasifikaci (Gonzales, 2018). Morfologie a segmentace se zabývají rozdělením obrazu na konkrétní části.

Extrakce vlastností a klasifikace představují zásadní úkoly pro autonomní vozidla, jejichž řešení často spočívá v aplikaci umělých neuronových sítí, jež napodobují klíčové aspekty lidské inteligence. Pro tyto úkoly se často osvědčuje využití konvolučních neuronových sítí (CNN), jež excelují v extrakci relevantních vlastností z obrazových dat a následné klasifikaci, přičemž zajišťují robustní a efektivní zpracování dat pro účely autonomního řízení vozidel.

2 METODY DETEKCE A SLEDOVÁNÍ TRASY S VYUŽITÍM KONVOLUČNÍ NEURONOVÉ SÍTĚ

2.1 PRINCIP KONVOLUČNÍCH NEURONOVÝCH SÍTÍ (CNN)

CNN představují třídu modelů hlubokého strojového učení, jejichž historie sahá až do konce 80. let 20. století, kdy byly poprvé prezentovány LeCunem et al. s cílem klasifikovat ručně psané číslice (LeCun, 1989). Od té doby se CNN staly významnými díky své mimořádné účinnosti, zejména v oblasti strojového vidění. Této výkonnosti dosahují díky specifické architektuře, která vyhovuje konkrétním nárokům obrazových dat.

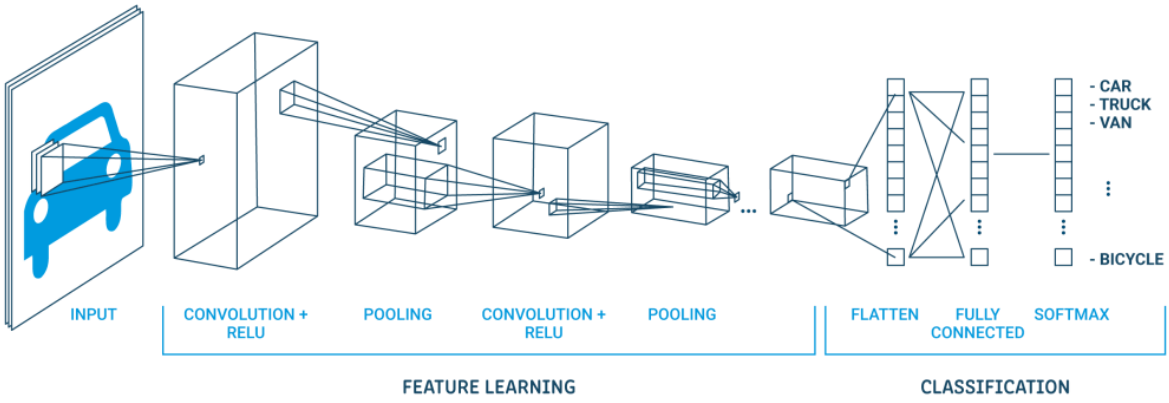
Prvním faktorem jsou vysoké rozměry obrazových dat. Například typický obrázek pro účely klasifikace z datasetu ImageNet obsahuje 224×224 RGB hodnot. Klasická neuronová síť by pro klasifikaci toho obrázku musela mít 150528 vstupů. I když by byla použita síť s pouze jednou skrytou vrstvou, síť by musela mít minimálně 150528^2 vah. Taková síť by byla v praxi obtížně implementovatelná s ohledem na výpočetní nároky (Prince, 2023).

Dalším faktorem je statistická souvislost sousedních pixelů v lokálních oblastech obrazu (Prince, 2023). Například na snímku trasy jsou určité sousední pixely v části okraje vozovky sémanticky podobné tím, že všechny jsou součástí právě okraje vozovky, existuje tedy mezi nimi určitý vztah. Klasické neuronové sítě však tuto souvislost nezohledňují.

CNN disponují předřadným extraktorem vlastností, který využívá zmíněných statistických souvislostí v obrazových datech. Tento extraktor vlastností je inspirován prací neurofyziologů Hubela a Wiesela z 50. let 20. století, kde ukázali, že vizuální kortex koček obsahuje neurony, které individuálně reagují na malé oblasti zorného pole (Hubel, 1959). Podobně jako vizuální kortex, CNN se tedy zaměřují na extrakci vlastností z lokálních oblastí vstupního obrazu.

Architektura CNN se obvykle skládá ze vstupní vrstvy, několika skrytých vrstev a výstupní vrstvy. Vstupní vrstva v kontextu strojového vidění obvykle představuje jednotlivé hodnoty pixelů obrazu. Pro vyjádření barvy jako číselné hodnoty se využívají různé druhy kódování,

obvykle černobílé, RGB (*red*, *green*, *blue*), CMY (*cyan*, *magenta*, *yellow*), CMYK (*cyan*, *magenta*, *yellow*, *black*) nebo HSI (*hue*, *saturation*, *intensity*) (Gonzales, 2018). Následuje řada konvolučních vrstev s aktivačními funkcemi, mezi kterými můžou, ale nemusí být pooling vrstvy. Pro účely klasifikace jsou za konvoluční a pooling vrstvy připojené plně propojené vrstvy. Typická architektura CNN je zachycena na obrázku 3.



Obrázek 3 – Architektura konvoluční neuronové sítě. Na levé straně je vstupní vrstva (*input*), dále vrstvy pro extrakci vlastností (*feature learning*) a nakonec vrstvy pro klasifikaci (A Comprehensive Guide to Convolutional Neural Networks, 2018).

2.1.1 Konvoluční vrstvy

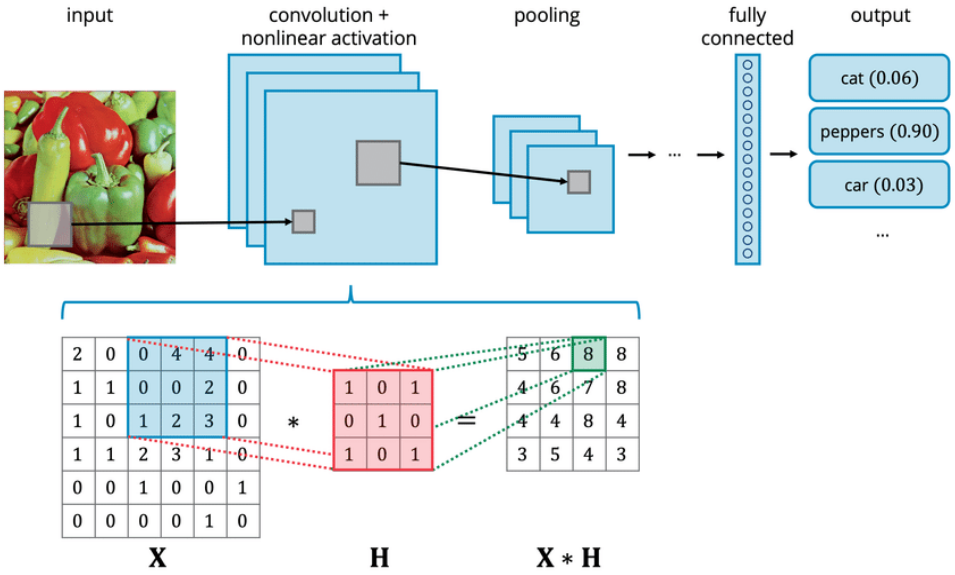
Konvoluční vrstvy jsou nezbytné pro extrakci vlastností (anglicky *feature learning*) ze vstupních dat. Tyto vrstvy se skládají z filtrů, které se posunují krokem daným parametrem *stride* nad vstupními daty a provádějí skalární součin mezi částí vstupní matice dat a maticí filtru, aby vytvořily mapu příznaků (anglicky *feature map*) (Krizhevsky, 2017).

Filtr, nebo jiným názvem kernel, je v kontextu CNN matice, obvykle velikostí výrazně menší než vstupní matice, která má za úkol najít vlastnosti dat v okolí částí vstupní matice dat. Extrakce vlastností je dosaženo pomocí matematické operace nazývané konvoluce, při níž se provede skalární součin prvků matice filtru a překrývající se části vstupní matice (viz obrázek 4). Filtr o velikosti 3×3 ($\mathbb{R}^{3\times3}$) aplikovaný na 2D vstup obsahující prvky $x_{i,j}$, vyprodukuje příznakovou mapu s prvky $h_{i,j}$:

$$h_{i,j} = f\left(\beta + \sum_{m=1}^{3}\sum_{n=1}^{3} \omega_{m,n} x_{i+m-2,j+n-2}\right) \tag{1}$$

kde $\omega_{m,n}$ jsou hodnoty filtru, β je práh a f je nelineární aktivační funkce. Parametry $\omega_{m,n}$ a β jsou optimalizovány během trénování sítě.

Konvolucí je možné získat významné vlastnosti z obrazu, včetně například hran a dalších charakteristik. CNN ovšem obvykle nejsou omezeny na pouze jednu konvoluční vrstvu. Využití série konvolučních vrstev, kdy následné vrstvy nachází vlastnosti z výstupů předchozích konvolučních vrstev, umožní extrakci nejen nízko úrovňových vlastností, jako jsou hrany, tak i vysoko úrovňové vlastnosti, například různé tvary nebo křivky v obraze.



Obrázek 4 – Znázornění operace konvoluce. Matice X představuje hodnoty pixelů vstupního obrazu. Matice H je konvoluční filtr. Matice X*H je výsledná matice po posouvání filtru H přes matici X a provádění skalárního součinu X*H. Je znázorněn jeden krok konvoluce překryté části matice X (vyznačeno modrou barvou) a filtru H, výsledek této operace je skalár, v tomto případě číslo 8 (Basic CNN architecture and kernel, 2020).

2.1.1.1 Kanály

Aplikace konvolučního filtru na vstupní obraz vyprodukuje příznakovou mapu, také označovanou jako kanál. Při konvoluci s použitím pouze jednoho filtru však dojde ke ztrátě informace (Prince, 2023). Například nastaví-li se filtr na extrakci vertikálních hran, dojde k potlačení všech ostatních charakteristik vstupních dat. Z tohoto důvodu se obvykle používá více filtrů, z nichž každý filtr vytvoří vlastní příznakovou mapu nebo kanál. V každém kanálu pak mohou být zachyceny různé vlastnosti. Využití kanálů je zobrazeno na obrázku 3, kde vstupní vrstva má 3 kanály (reprezentující hodnoty RGB) a první konvoluční vrstva již disponuje více než třemi kanály.

2.1.1.2 Padding

Rovnice (1) demonstruje, že operace konvoluce spočívá ve váhovaném součtu předchozích, aktuálních a následujících prvků vstupních dat (za předpokladu rozměru filtru 3×3). Problém nastává například u okrajových hodnot vstupu, kde neexistují předchozí vstupy.

Nabízí se dva možné přístupy k řešení tohoto problému. První přístup, nazývaný *zero padding*, vyplní neexistující vstupy nulami. Druhý přístup, nazývaný *valid padding*, pracuje se vstupními daty jako s cyklickým polem, kde např. pokud chybí předchozí vstupní hodnoty na levé straně, jsou použity okrajové hodnoty z pravé strany.

2.1.2 Aktivační vrstvy

Aktivační vrstvy do sítě zavádí nelinearitu, čímž umožňují modelování složitých vzorů. Jsou umístěny mezi konvolučními a sdružovacími vrstvami. Typické aktivační funkce zahrnují ReLU, Leaky ReLU, ELU, sigmoid, tangens hyperbolický a logistickou funkci.

ReLU je stále jednou z nejpoužívanějších funkcí, která díky své jednoduchosti umožňuje rychlejší učení s více vrstvami (LeCun, 2015). Matematický zápis ReLU je:

$$f(x) = max\,(x, 0) \tag{2}$$

Leaky ReLU je modifikovaná verze ReLU, která ponechá malou část výstupu pro záporné vstupy. Jednou z klíčových výhod je schopnost lépe řešit problém mizejícího gradientu, který se může vyskytnout u tradičních funkcí ReLU (Klein, 2022). Matematický předpis Leaky ReLU je:

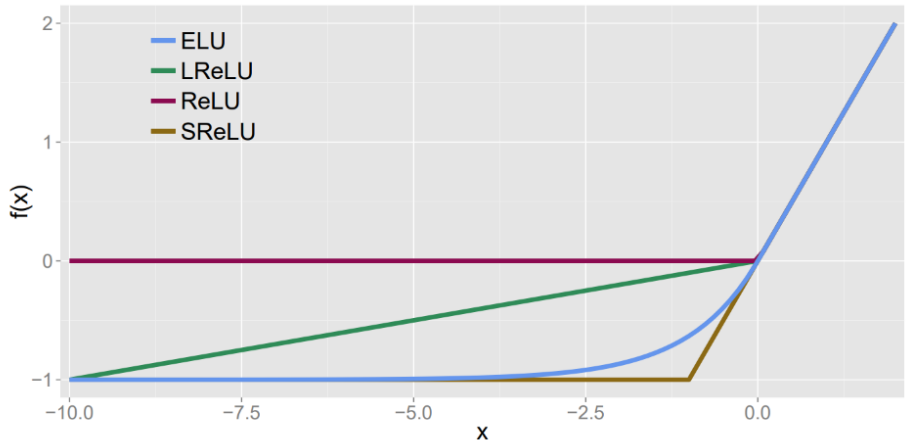
$$f(x) = \begin{cases} \alpha x, & x \leq 0 \\ x, & x > 0 \end{cases} \tag{3}$$

kde hyperparametr α udává sklon křivky, který určuje míru propuštění pro záporné vstupy, obvykle se jedná o velice malou hodnotu.

ELU, podobně jako Leaky ReLU, zachovává malou část záporných vstupů a zároveň také řeší problém mizejícího gradientu. V experimentech bylo prokázáno, že použití ELU má několik výhod oproti ReLU či Leaky ReLU. ELU umožňuje nejen rychlejší učení, ale také výrazně zlepšuje schopnost neuronové sítě generalizovat, zejména u sítí s více než 5 vrstvami (Clevert, 2015). Matematický předpis ELU je:

$$f(x) = \begin{cases} \alpha(\exp(x) - 1), & x \leq 0 \\ x, & x > 0 \end{cases} \tag{4}$$

kde hyperparametr α stejně jako u Leaky ReLU určuje míru propuštění záporných vstupů. Provnání aktivačních funkcí lze vidět na obrázku 5.



Obrázek 5 – Porovnání průběhu vybraných aktivačních funkcí. Červená křivka je ReLU, zelená je Leaky ReLU (LReLU, α = 0,1), hnědá je Shifted ReLU (SReLU) a modrá je Exponential Linear Unit (ELU, α = 1,0) (Clevert, 2015).

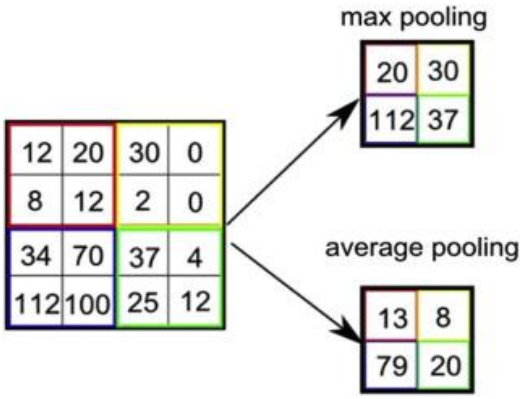
2.1.3 Sdružovací vrstvy

Sdružovací vrstvy v architektuře CNN redukují rozměry příznakových map, při zachování klíčových vlastnosti. V architektuře CNN mohou následovat po konvolučních vrstvách.

Existují dva obvyklé přístupy ke sdružování. První přístup je *max pooling*, který vybírá maximální hodnotu z malé oblasti příznakové mapy, zachycuje tak nejvýraznější rysy v datech. *Average pooling* vypočítává aritmetický průměr, poskytuje tedy obecnější reprezentaci vlastností. Porovnání obou přístupů lze vidět na obrázku 6.

Redukce rozměrů snižuje počet trénovatelných parametrů a umožňuje výběr nejreprezentativnějších vlastností. Sdružovací vrstvy zároveň umožňují síti tolerovat malé posuny v příznakových mapách, což je klíčové pro úlohy jako obrazová klasifikace, kde je

důležité rozpoznat objekt bez ohledu na jeho umístění v obraze (Schmidhuber, 2015; LeCun, 2015).



Obrázek 6 – Znázornění operace sdružovacích vrstev a rozdílu mezi operacemi max pooling a average pooling (Types of pooling layers, 2022).

2.1.4 Plně propojené vrstvy

Plně propojené vrstvy (anglicky *fully conected layers*) jsou nezbytné pro zpracování vysoko úrovňových vlastností extrahovaných konvolučními a sdružovacími vrstvami. Tyto vrstvy představují poslední fázi v architektuře CNN a jejich úkolem je klasifikace nebo regrese. Strukturou jsou obdobné klasickým umělým neuronovým sítím, kde jsou všechny neurony z předchozích vrstev propojeny se všemi neurony v následující vrstvě.

2.1.4.1 Umělý neuron

Pojem *neuron* v kontextu umělých neuronových sítí označuje matematický model, který v roce 1943 představili McCulloch a Pitts za účelem pochopit, jak lidský mozek dokáže vykazovat složité vzorce chování při použití velkého počtu jednoduchých, vzájemně propojených výpočetních prvků (McCulloch, 1943).

Konkrétně se jedná o matematickou funkci, obvykle nazývanou *umělý neuron*, která přijímá jeden nebo více vstupů. Každý z těchto vstupů je násoben reálným číslem, obvykle nazývaným *váha*, které odráží jeho důležitost či příspěvek k výstupu neuronu. Poté jsou vážené vstupy sečteny a k tomuto součtu je přičten tzv. *práh* neuronu, také označovaný jako *bias*. Nakonec je na tento výsledek aplikována nelineární aktivační funkce, která určuje, zda a jak silně bude neuron aktivován a jaký výstup vygeneruje. Rovnice umělého neuronu lze vyjádřit jako:

$$a = f\left(\sum_i w_i x_i + b\right) \tag{5}$$

kde f je nelineární aktivační funkce, x jsou vstupy do umělého neuronu, i je index vstupů, w jsou váhy, b je práh a a je výstup umělého neuronu (nebo jiným výrazem aktivace). V původním modelu byla jako aktivační funkce použita skoková funkce, dnes se nejčastěji používá ReLU a její modifikace.

2.1.4.2 Plně propojená síť

Vzájemným propojením umělých neuronů do více vrstev vznikne plně propojená síť. Aby tato síť mohla být napojena na konvoluční a sdružovací vrstvy, využívá se tzv. zplošťovací vrstva (anglicky *flatten layer*), která transformuje více dimenzionální matici příznakové mapy (v případě obrazu 2D) na 1D vektor. Tento vektor je použit jako vstupní vrstva pro danou síť. Zobecněním rovnice (4) pro plně propojenou síť s více vrstvami a neurony vznikne tvar:

$$a_j^l = f\left(\sum_k w_{j,k}^l a_k^{l-1} + b_j\right) \tag{6}$$

kde l je index vrstvy, j je index umělého neuronu ve vrstvě l, k je index vstupů, a_j^l je výstup neuronu a a_k^{l-1} jsou vstupy do neuronu (a zároveň výstupy neuronů z předchozí vrstvy). Rovnice (6) lze přepsat do více kompaktní vektorové formy:

$$a^l = f(W^l a^{l-1} + b^l) \tag{7}$$

kde a^l je vektor všech výstupů vrstvy l, a^{l-1} je vektor všech výstupů předchozí vrstvy, W je matice vah a b je vektor prahů. Výstup poslední vrstvy je obvykle denotován symbolem $\hat{y}$, reprezentující predikci sítě:

$$\hat{y} = f(W^l a^{l-1} + b^l) \tag{8}$$

2.1.4.3 Aproximační vlastnost plně propojených sítí

Je dokázáno, že umělé neuronové sítě s pouze jednou skrytou vrstvou obsahující konečný počet neuronů dokáží aproximovat libovolnou spojitou funkci s libovolnou přesností aproximace ϵ, za předpokladu, že síť disponuje dostatečným počtem umělých neuronů (Cybenko, 1989; Hornik, 1989). Pro libovolnou spojitou funkci $f: \mathbb{R}^n \to \mathbb{R}$ existují parametry sítě $\{W^1, b^1, W^2, b^2\}$, tak že pro $\epsilon > 0$ existuje neuronová síť pro kterou platí:

$$|g(x) - f(x)| < \epsilon \tag{7}$$

kde $g(x)$ je aproximace $f(x)$ nalezená neuronovou sítí.

Vlastnost plně propojených neuronových sítí aproximovat funkce se osvědčuje pro účely klasifikace nebo regrese, kde je nutné, aby síť identifikovala neznámou mapovací funkci mezi zploštěným vektorem kanálů konvolučních vrstev a požadovaným výstupem u trénovacích dat. Síť by měla v ideálním případě být schopna nalezení tohoto vztahu a zobecnění na neznámá data, která nejsou zahrnuty v trénovací sadě.

2.1.5 Trénování CNN

Trénování CNN a neuronových sítí obecně zahrnuje optimalizaci parametrů sítě s cílem minimalizovat chybovou funkci. Při metodě tzv. „učení s učitelem“ (anglicky *supervised learning*) je síť učena prostřednictvím trénovacích dat. Trénovací data jsou soubor párů vstupů a požadovaných výstupů, které jsou obvykle manuálně vytvořeny člověkem, který má znalosti o vztahu mezi těmito daty. V kontextu této práce může být vstup například matice hodnot RGB jednotlivých pixelů snímku trasy a požadovaný výstup může být míra zatáčení na základě aktuálního snímku trasy.

Cílem trénování je najít takový soubor parametrů sítě, aby bylo dosaženo toho, že pro každý vstup síť vyprodukuje výstup, který je co nejblíže požadovanému výstupu (Rumelhart, 1986). Cílem je tedy minimalizovat rozdíl mezi výstupy sítě a požadovanými výstupy. Obvykle používaná chybová funkce pro účely regrese je střední kvadratická chyba (jiným názvem *MSE*, anglicky *mean squared error*), definovaná jako:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{8}$$

kde N je počet párů trénovacích dat, $\hat{y}$ je predikce sítě a y je požadovaný výstup.

K minimalizaci chybové funkce se obvykle využívá technika gradientního sestupu, kdy je vypočten záporný gradient MSE, pomocí kterého se optimalizují parametry sítě. K vypočtení gradientu je třeba obdržet parciální derivace MSE vůči všem trénovatelným parametrům sítě. Pro vypočtení parciálních derivací gradientu se osvědčil algoritmus zpětného šíření chyby.

2.1.5.1 Zpětné šíření chyby

Algoritmus zpětného šíření chyby je základním mechanismem učení neuronových sítí, který spočívá v postupném předávání chybových signálů z výstupních vrstev sítě zpět ke vstupním vrstvám. Nejprve se provede dopředná fáze, kdy se vstupní signál přenese přes síť a získají se predikce. Poté se vypočítá chyba predikce pomocí porovnání s očekávanými výstupy. Tato

chyba se poté propaguje zpět skrze síť, přičemž se pomocí gradientního sestupu upravují váhy jednotlivých spojení tak, aby se minimalizovala chyba. Tento proces se opakuje pro každý trénovací vzorek a probíhá iterativně, dokud není dosaženo dostatečné úrovně přesnosti nebo nejsou splněna jiná ukončovací kritéria.

2.1.5.2 Trénovací množina

Trénovací množina je soubor dat, který se používá přímo k trénování neuronové sítě. Obvykle představuje největší podíl dostupných dat. Data obsažená v trénovací množině jsou v průběhu trénování využívána k výpočtu gradientu a následné optimalizaci parametrů sítě.

2.1.5.3 Validační množina

Validační množina je typicky podmnožina trénovací množiny, která se používá k ladění hyperparametrů sítě. Data z validační množiny nejsou zahrnuta do výpočtu gradientu a slouží primárně k poskytování zpětné vazby během trénování. Tato zpětná vazba je klíčová pro prevenci nadměrného přizpůsobení na trénovací data.

2.1.5.4 Testovací množina

Testovací množina je oddělená sada dat, která slouží k ověření výkonnosti natrénovaného modelu. Poskytuje objektivní měřítko schopnosti sítě generalizace na neznámá data. Je důležité, aby všechny množiny byly disjunktní, to znamená, že žádná data z jedné sady by neměla být obsažena v jiné sadě. Tento postup zajišťuje spolehlivé zhodnocení výkonu sítě.

2.1.5.5 Dávky a epochy

Během každé iterace při trénování se náhodně vybere podmnožina trénovacích dat nazývaná „dávka“ (anglicky *batch*). Gradient je poté vypočten pomocí této podmnožiny. Velikost dávky se může pohybovat od jednoho vzorku až po celou trénovací množinu.

Při trénování sítě je iterováno přes všechny dávky, dokud nejsou vyčerpána všechna trénovací data. Následně se celý proces opakuje. Jeden takový průchod přes všechna trénovací data se označuje jako „epocha“.

2.2 METODY DETEKCE A SLEDOVÁNÍ TRASY

2.2.1 Sémantická segmentace

CNN lze efektivně využít pro sémantickou segmentaci v oblasti počítačového vidění. Tato technika zahrnuje klasifikaci každého pixelu v obraze, což umožňuje detailní identifikaci obsahu obrazu. Pomocí sémantické segmentace lze snímek trasy rozdělit do různých částí a extrahovat pixely odpovídající trase. Tyto informace mohou být následně předány dalším algoritmům, které provádějí vyhodnocení a navrhují akční zásahy pro vozidlo.

Pro zvýšení přesnosti a efektivity sémantické segmentace byly vyvinuty různé techniky hlubokého učení, jako je například projekt DeepLab. Tento přístup využívá hluboké konvoluční sítě a plně propojená podmíněná náhodná pole k dosažení přesné sémantické segmentace obrazu. Metoda využívá sílu konvolučních neuronových sítí k efektivní extrakci sémantických rysů (Chen, 2018).

2.2.2 End-to-End systémy

End-to-End systémy autonomních vozidel na bázi CNN využívají hluboké neuronové sítě trénované obvykle pomocí algoritmů učení s učitelem k predikci řídicích pokynů, například úhlů zatočení, přímo ze vstupních dat, jako jsou snímky nebo údaje ze senzorů (Bojarski, 2016). Tato metodika konsoliduje celý proces řízení do jediného systému, což umožňuje neuronové síti uchopit kompletní procesní řetězec nezbytný pro řízení vozidla.

Existují přístupy, které kombinují CNN s dalšími architekturami hlubokého učení, jako jsou sítě s dlouhou krátkodobou pamětí (LSTM), aby vytvořily end-to-end sítě, které využívají současné i budoucí obrazy s využitím V2V (*vehicle to vehicle*) komunikace pro řízení vozidla (Valiente, 2019).

Výhodou end-to-end přístupu je schopnost zefektivnit proces učení přímým mapováním senzorických vstupů na řídící výstupy, čímž se eliminuje potřeba ručně vytvářených algoritmů nebo zprostředkujících rozhodovacích modulů (Chen, 2015). Na druhou stranu výraznou nevýhodou těchto systémů je potenciální nedostatečná interpretovatelnost a transparentnost rozhodovacího procesu, protože tyto modely často fungují jako černé skříňky, takže je obtížné pochopit, jak a proč uskutečňují konkrétní rozhodnutí.

2.2.3 Fúze zpětnovazebního učení a CNN

V tomto přístupu jsou zkombinovány metodiky CNN a zpětnovazebního učení (*reinforcement learning*). CNN jsou využity pro extrakci vlastností a rysů vstupních senzorických dat. Prostřednictvím řady konvolučních a sdružovacích vrstev zpracovává CNN vstupní obrazy a hierarchickým způsobem extrahuje významné rysy. Tato hierarchická reprezentace je klíčová, protože zachycuje podstatné vizuální informace nezbytné pro rozhodování agenta RL.

Jakmile CNN zpracuje vstupní obrazy a extrahuje relevantní rysy, výstup je předán agentovi RL. Agent RL pak na základě pozorovaného stavu rozhoduje o akcích, jako je zatáčení, akcelerace nebo brzdění. Tyto akce spouštějí smyčku zpětné vazby v podobě signálu odměny, který indikuje účinnost rozhodnutí agenta. Signál odměny slouží k posílení žádoucího chování, například dodržování značení jízdních pruhů, a zároveň k penalizaci odchylek nebo kolizí.

Pro usnadnění učení a rozhodování se agent RL snaží naučit politiku, která v průběhu času maximalizuje kumulativní odměny. Tento proces často zahrnuje techniky, jako je Q-learning (Mnih, 2015), kdy agent iterativně upravuje své parametry na základě získaných odměn. Současně probíhá učení CNN pod dohledem, aby se zlepšily schopnosti extrakce vlastností, což vede k symbiotickému tréninkovému procesu, který zvyšuje celkový výkon v úloze sledování trasy.

Jakmile je model CNN-RL natrénován, projde vyhodnocením na samostatné testovací sadě, aby se zjistila jeho účinnost a schopnost zobecnění. Úspěšný výkon ve fázi hodnocení opravňuje k nasazení v reálných scénářích, jako jsou autonomní vozidla. V tomto kontextu model umožňuje vozidlům autonomně se učit pohybovat po silnicích, komunikovat s okolím a získávat zpětnou vazbu bez potřeby explicitního dohledu nebo označených tréninkových dat.

2.2.4 CNN s využitím temporálních informací

CNN lze kombinovat s rekurentními neuronovými sítěmi (RNN) nebo časovými konvolučními sítěmi (TCN), aby se do nich začlenily časové informace z videosekvencí. CNN vynikají v extrakci prostorových rysů ze snímků videa, zatímco RNN a TCN jsou schopné zachytit časové závislosti v rámci sekvencí. To pomáhá při přijímání informovanějších rozhodnutí o sledování trasy díky zohlednění dynamiky scény v čase.

Spojení CNN s RNN nebo TCN poskytuje ucelené řešení pro autonomní vozidla, které využívá prostorové i časové informace z videosekvencí. Tato integrace umožňuje vozidlům činit lépe

informovaná rozhodnutí na základě komplexního pochopení vizuálních dat, se kterými se setkávají.

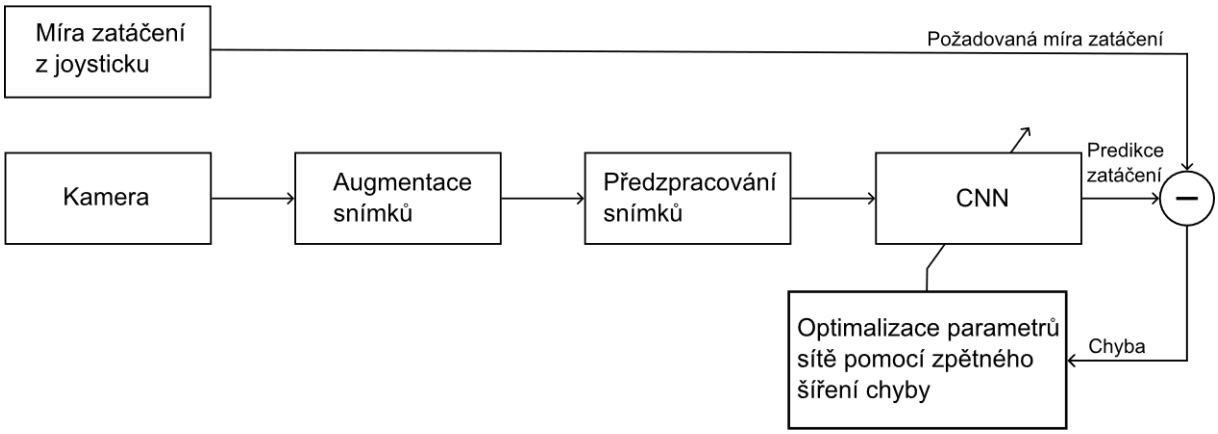
3 NÁVRH METODIKY PRO SBĚR A PŘÍPRAVU DAT A TRÉNOVÁNÍ NEURONOVÉ SÍTĚ

3.1 PŘEHLED NAVRŽENÉHO SYSTÉMU PRO TRÉNOVÁNÍ SÍTĚ

Systém pro trénování CNN je inspirovaný článkem *End to End Learning for Self-Driving Cars* od Bojarskiho et al. (2016) a je modifikován pro aplikaci pro počítač NVIDIA Jetson Nano. CNN je trénována metodou učení s učitelem pomocí manuálně anotovaných dat, obsahující snímky trasy a odpovídající hodnoty zatáčení v rozsahu -1 až 1, kde -1 znamená úplné zatáčení doleva a 1 úplné zatáčení doprava. Jelikož je výstupem sítě pouze jedna hodnota, jedná se o regresní úlohu. Proces trénování je zachycen na obrázku 7.

Trénování probíhá iterativně přes tréninková data, kde se porovnává požadovaná a predikovaná hodnota zatáčení a tento rozdíl je vyhodnocen pomocí funkce střední kvadratické chyby (MSE). Tato chyba se pak používá k optimalizaci parametrů sítě algoritmem zpětné propagace chyby.

Trénování na základě údajů od lidského řidiče není dostatečné, neboť se síť musí naučit, jak se zotavit z chyb (Bojarski, 2016). V opačném případě by docházelo ke kumulaci chyb a vozidlo by se postupně odchylovalo od vyznačené trasy. Z tohoto důvodu jsou trénovací data augmentována tak, že náhodně vybrané snímky trasy jsou posunuty ve svislém i horizontálním směru.



Obrázek 7 – Blokové schéma systému pro trénování sítě.

3.1.1 Augmentace snímků

Na snímky jsou pro zlepšení schopnosti generalizace a prevenci kumulace odchylek od trasy náhodně aplikovány tyto augmentační techniky:

- Afinní transformace obrazu, vertikální a horizontální posun.
- Afinní transformace obrazu, přiblížení.
- Zvýšení nebo snížení jasu.
- Převrácení obrazu podle svislé osy, v tomto případě je nutné současně vynásobit hodnotu zatáčení -1.

3.1.2 Předzpracování snímků

Cílem předzpracování obrazu (*preprocessing*) je zlepšit kvalitu vstupních dat a učinit je vhodnějšími pro trénování CNN, což může vést k lepšímu výkonu a schopnosti generalizace modelu. Jsou využity tyto konkrétní techniky:

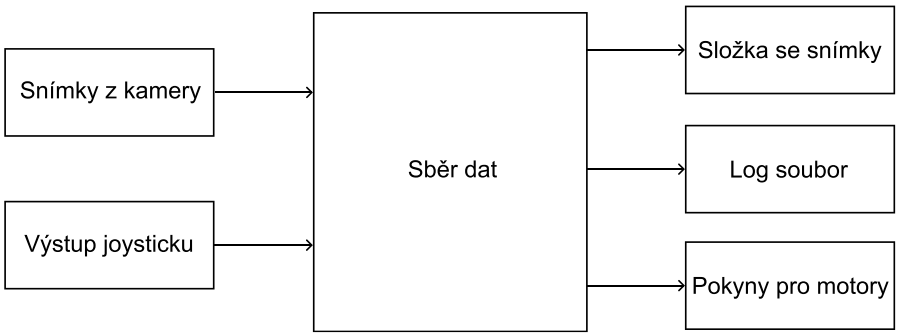
- Oříznutí snímku: oříznutí pomáhá odstranit nepotřebné části, které neobsahují relevantní informace pro daný úkol.
- Převod do barevného prostoru YUV: YUV odděluje jasovou složku (Y) od chrominančních složek (U a V). Použití YUV může zvýšit schopnost modelu rozlišit vlastnosti související s jasem od vlastností souvisejících s barvou.
- Gaussovské rozostření: použití Gaussova rozostření může pomoci vyhladit obraz, redukovat šum a nepodstatné detaily. To může zlepšit robustnost modelu tím, že je méně citlivý na malé odchylky a šum ve vstupních datech (Vogelsang, 2018).
- Normalizace: normalizace hodnot pixelů ze standartního rozsahu <0; 255> na rozsah <0; 1> pomáhá stabilizovat a urychlit proces trénování. Zajišťuje, že vstupní hodnoty jsou ve normalizovaném rozsahu, který neuronová síť může efektivně zpracovat.

3.2 SBĚR DAT, ANOTACE A ROZDĚLENÍ

Průběh sběru dat je následující: vozidlo je ručně řízeno po trénovací trase pomocí joystickových povelů, zatímco kamera nepřetržitě pořizuje snímky této trasy. Algoritmus sběru dat má za úkol uložit všechny zachycené snímky do specifické složky. Paralelně jsou motorům vozidla předávány instrukce o jejich rychlosti založené na výstupech joysticku, aby vozidlo dokázalo bezchybně po trase manévrovat.

Po dokončení procesu sběru je důležité správně anotovat a rozdělit nasbíraná data. K tomu účelu slouží log soubor, který je vytvořen ve formátu CSV (*Comma-Separated Values*), typického pro ukládání tabulkových dat. Formát CSV umožňuje ukládat data tak, že každý řádek obsahuje sadu hodnot oddělených čárkami, přičemž každá hodnota může být textový řetězec nebo číslo. Vygenerovaný CSV soubor je rozdělen na dva sloupce. První sloupec obsahuje kompletní systémovou cestu k danému snímku trasy. Druhý sloupec pak obsahuje hodnotu zatáčení, která odpovídá danému snímku, zaznamenaného během sběru dat.

Během trénování sítě jsou postupně čteny všechny řádky vygenerovaného log souboru, přičemž konkrétní snímek odpovídající aktuálnímu řádku je načten pomocí kompletní systémové cesty obsažené v prvním sloupci CSV souboru. Síti je zároveň prezentována hodnota zatáčení z druhého sloupce. Blokové schéma procesu sběru dat lze vidět na obrázku 8.



Obrázek 8 – Blokové schéma metodiky sběru dat.

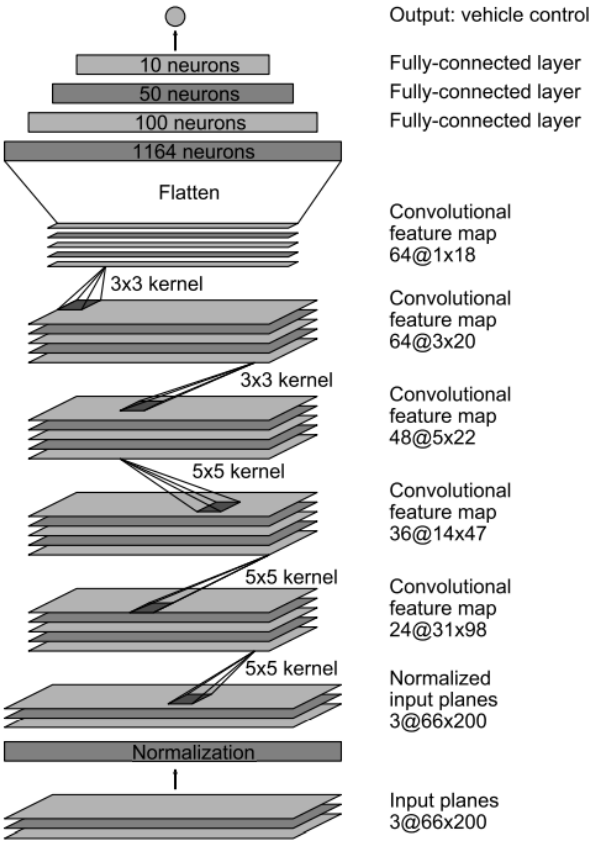
3.3 ARCHITEKTURA SÍTĚ

Architektura sítě vychází z návrhu uvedeného v článku *End to End Learning for Self-Driving Cars* od Bojarskiho et al. (2016). Síť se skládá z devíti vrstev, včetně normalizační vrstvy, pět konvolučních vrstev a třech plně propojených vrstev, viz obrázek 9.

Vstupní vrstva má tři kanály pro YUV složky. Normalizační vrstva převede hodnoty pixelů obrazu na rozsah <0; 1>. Filtry 5×5 a stride 2×2 jsou použity v prvních třech konvolučních vrstvách pro zachycení rozsáhlejších rysů a redukci rozměrů dat.

Poslední dvě konvoluční vrstvy mají filtry 3×3 bez stride, což umožňuje zachytit komplexnější vlastnosti s menším počtem parametrů. Tři plně propojené vrstvy vedou k výstupu, reprezentujícímu míru zatáčení. Tyto vrstvy jsou navrženy jako řídící systém pro vyhodnocení

zatáčení, autoři článku ovšem podotýkají, že není možné čistě oddělit, které části sítě fungují primárně jako extraktor vlastností a které slouží jako řídicí jednotka (Bojarski, 2016)

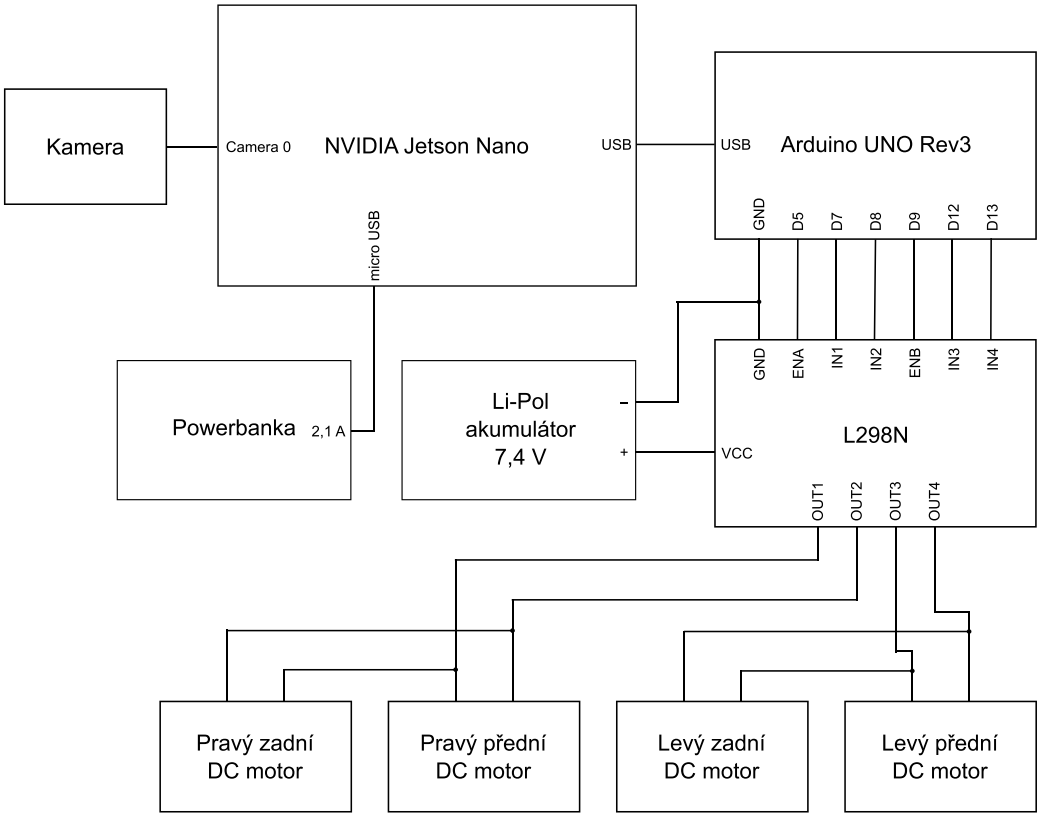


Obrázek 9 – Architektura použité konvoluční neuronové sítě (Bojarski, 2016).

4 HARDWAROVÁ IMPLEMENTACE

4.1 PŘEHLED NAVRŽENÉHO AUTONOMNÍHO VOZIDLA

Navržené vozidlo je osazeno celkem čtyřmi koly, z nichž dvě jsou na levé straně a dvě na pravé. Každé kolo je poháněno samostatným stejnosměrným motorem. Vozidlo je řízené prostřednictvím diferenciálního zatáčení, což znamená, že je určeno rozdílem rychlostí motorů levé a pravé strany. Hlavním počítačem pro vyhodnocování obrazových dat pomocí CNN a provádění algoritmů umožňujících autonomní řízení je NVIDIA Jetson Nano. Pro zachycení obrazových informací o okolním prostředí je využívána kamera IMX219-77. Ovládání pohybu vozidla a řízení zatáčení je realizováno pomocí kombinace mikrokontroléru Arduino UNO Rev3 a driveru L298N, které operují ve spojení s Jetsonem jako periferní zařízení. Blokové schéma zapojení jednotlivých komponent vozidla lze vidět na obrázku 10. Reálné sestavené vozidlo je zachyceno na obrázku 11.



Obrázek 10 – Blokové schéma autonomního vozidla.

4.2 VÝBĚR KOMPONENT

4.2.1 NVIDIA Jetson Nano

Pro vyhodnocení míry zatáčení na základě snímků trasy s využitím výstupů CNN byl na základě několika faktorů zvolen počítač NVIDIA Jetson Nano. Prvním faktorem je výpočetní výkon. Díky GPU s architekturou NVIDIA CUDA dokáže Jetson Nano efektivně provádět paralelní zpracování a složité výpočty potřebné pro konvoluční sítě. Tato schopnost je stěžejní pro rychlé a spolehlivé rozhodování autonomního vozidla v reálném čase.

Dalším faktorem je nízká spotřeba energie, což je zásadní pro zařízení poháněné bateriemi nebo jinými omezenými zdroji energie. Konkrétně, spotřeba Jetsonu je 10 W v režimu MAXN. Další výhodou jsou také kompaktní rozměry počítače, umožňující snadnou integraci do omezených prostorů vozítka, optimalizaci rozložení komponent a minimalizaci hmotnosti.

Konečně, Jetson Nano je plně kompatibilní s vývojovými nástroji a knihovnami pro práci s algoritmy strojového učení, což usnadňuje vývoj a ladění aplikací pro autonomní vozítka. Celkově lze konstatovat, že Jetson Nano poskytuje optimální kombinaci výkonu, efektivity a přizpůsobivosti. Podobu počítače Jetson Nano lze vidět na obrázku 12.

Obrázek 11 – Počítač NVIDIA Jetson Nano (Jetson Nano Developer Kit, 2024).

4.2.2 Arduino Uno Rev3

Arduino Uno Rev3 představuje open-source platformu založenou na mikrokontroléru ATMega328P, disponující 14 digitálními piny, z nichž 6 podporuje PWM výstupy, 6 analogovými vstupy a USB připojením. Pro generování logických signálů pro driver L298N bylo Arduino vybráno na základě několika faktorů. Prvním je rozsah logických signálů digitálních pinů Arduina, který se pohybuje v rozmezí 0–5 V, na rozdíl od 0–3,3 V u GPIO pinů Jetsonu. Tato vlastnost lépe koresponduje s požadavky L298N, jenž přijímá signály v rozmezí 0–5 V.

Dalším důležitým faktorem je nižší maximální proudové zatížení GPIO pinů Jetsonu. Arduino Uno Rev3 umožňuje maximální proudové zatížení digitálních pinů až 40 mA, což umožňuje přímé připojení k L298N bez nutnosti dalších externích spínacích prvků.

Přenesení výpočtu zatáčení na Jetson Nano a následné ovládání motorů pomocí Arduina nabízí několik výhod, umožňuje distribuované zpracování, optimalizuje využití zdrojů a zlepšuje odezvu v reálném čase díky využití vyhrazených hardwarových akcelerátorů Jetsonu.

Tento přístup zároveň zvyšuje modularitu a flexibilitu systému, což usnadňuje modernizaci nebo výměnu jednotlivých komponent bez nutnosti přepracovávat celý systém. Současně zjednodušuje izolaci poruch, což usnadňuje odstraňování problémů oddělením jednotlivých úloh a identifikaci potenciálních problémů. Celkově tento přístup zvyšuje výkon, odezvu a flexibilitu v aplikacích autonomních vozidel.

4.2.3 L298N

Driver L298N je integrovaný obvod vhodný k řízení stejnosměrných nebo krokových motorů. Obvod obsahuje dva integrované H-můstky, které umožňují nezávislé řízení rychlosti a polarity dvou DC motorů. Pro řízení vozítka jsou využity celkem čtyři DC motory, které operují v párech, dva motory jsou tedy zapojeny jako jeden motor. Tento driver byl vybrán díky jednoduchému rozhraní, umožňující snadnou integraci do navrženého systému. Zároveň, nabízí robustní a kompaktní konstrukci a je široce dostupný.

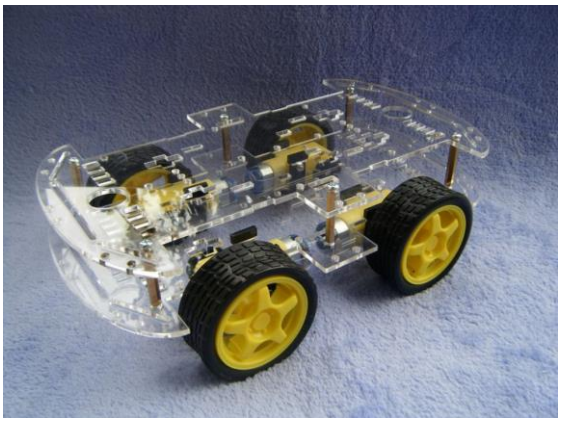
L298N je vybaven dvěma výstupními kanály A (piny OUT1 a OUT2) a B (piny OUT3 a OUT4) pro připojení motorů, z nichž každý je schopen dodávat až 2 A proudu až do 35 V. Tato vlastnost umožňuje snadné řízení DC motorů běžně používaných v robotických aplikacích. Dále je ovladač vybaven ochranou proti zkratu a možností využití externích diod pro ochranu před zpětným napětím z motorů.

K řízení rychlosti a polarity DC motorů slouží vstupní piny IN 1-4, EN A a EN B, které jsou přímo spojeny s digitálními piny Arduina. Směr otáčení motorů je řízen prostřednictvím logických úrovní IN 1-2 pro kanál A a IN 2-4 pro kanál B. Rychlost motorů pak určují signály PWM generované Arduinem, jež jsou aplikovány na piny EN A a EN B.

4.2.4 Robotická platforma s DC motory

Jako základní konstrukce vozidla je využita univerzální robotická platforma. Jedná se o široce dostupnou a jednoduchou konstrukci, která zahrnuje dvě plexiskla, čtyři DC motory a čtyři kola s pneumatikami (viz obrázek 13). Plexiskla jsou propojeny distančními sloupky. Specifikace DC motorů je následující:

- Napájení DC motorů: 3–7 V
- Rozměry kola: 26 × 65 mm
- Převodní poměr: 48:1
- Rozměry podvozku: 260 × 150 mm
- Proudový odběr při 6 V: 120 mA



Obrázek 12 – Robotická platforma (Robotické vozítko 4WD, 2020).

4.2.5 ZIPPY 6000 mAh 35 C Hardcase Pack

Akumulátor Zippy Hardcase 6000 mAh 35 C je široce používaný v RC modelářství pro svou vysokou kapacitu 6000 mAh a výkon 35 C. Hardcase design zajišťuje odolnost a ochranu proti poškození a deformaci. Je ideální volbou pro napájení DC motorů díky své vysoké kapacitě, což umožňuje dlouhé provozní doby, a výkonu 35 C, který zajišťuje dostatečný proud pro zrychlení a rozjezdy. Navíc, přestože má vysokou kapacitu, má poměrně nízkou hmotnost, což zlepšuje manévrovatelnost vozidla.

4.2.6 Powerbanka Aligator PB1000

Pro napájení Jetsonu je použita powerbanka značky Aligator o kapacitě 10000 mAh. Tato powerbanka poskytuje výstupní napětí 5 V o maximálním proudu 2,1 A. Díky poměrně vysoké kapacitě poskytuje dostatek energie pro provoz Jetsonu Nano po delší dobu. To je důležité zejména pro rozsáhlejší experimenty nebo testy s vozítkem.

Kompaktní rozměry a nízká hmotnost powerbanky umožňují snadnou integraci do omezených prostor vozítka.

4.2.7 Kamera IMX219-77

Pro pořízení snímků trasy byla zvolena kamera IMX219-77. Jedná se o CSI (*Camera Serial Interface*) kameru kompatibilní s počítačem Jetson Nano. IMX219-77 nabízí rozlišení 8 megapixelů, což poskytuje dostatečně detailní obrazová dat pro účely detekce a rozpoznávání objektů. Kameru lze vidět na obrázku 14.

Tato kamera je běžně dostupná a má dobrou podporu ve vývojářské komunitě. To usnadňuje integraci kamery do existujících systémů. Specifikace kamery jsou následující:

- Rozlišení: 3280 × 2464 pixelů
- Konektor: CSI
- Rozměry: 25 × 24 mm
- Zorný úhel FOV: 77°
- Typ matice: CMOS
- Senzor: Sony IMX219
- Zkreslení: <1 %

Obrázek 13 – Kamera IMX219-77 (Kamerový modul Sony IMX219-77, 2018).

4.3 KONSTRUKCE VOZIDLA

Konstrukce je založená na stavebnicové robotické platformě (viz obrázek 13), jelikož se jedná spíše o experimentální aplikaci pro ověření funkce sledování trasy. Kamera je umístěna na nastavitelném nástavci a je přichycena k plexisklu základní konstrukce pomocí dvojice šroubů, jak lze vidět na obrázku 14. Ostatní komponenty jsou buď uchyceny šrouby nebo stahovacími pásky. Na pravé straně obrázku 14 je zachycen podvozek vozidla s ukázkou zapojení motorů.

Obrázek 14 – Sestavený model autonomního vozidla.

5 SOFTWAROVÁ IMPLEMENTACE

5.1 PŘEHLED NAVRŽENÉ SOFTWAROVÉ IMPLEMENTACE

Softwarové řešení je systematicky rozděleno do tří fází. První fází je sběr dat. Je využito celkem šesti skriptů, z toho pět je naprogramováno v jazyce Python 3.6.9 a jeden v jazyce C++ pro Arduino. Python skripty zajišťují pořizování snímků, čtení výstupů bezdrátového ovladače DualSense pro uložení hodnot zatáčení, přenos řídících signálů přes USB do Arduina a konečně ukládání dat do formátu CSV. Všechny python skripty jsou spouštěny lokálně na počítači Jetson Nano.

Další fází je trénování CNN. Pro tyto účely jsou použity dva skripty naprogramované v jazyce Python 3.7. První skript zajišťuje podpůrné metody jako např. předzpracování nebo augmentaci dat pro druhý skript, který pak obstarává samotné trénování CNN. Oba skripty jsou spouštěny na desktopovém počítači kvůli potřebě vyššího výpočetního výkonu pro efektivní trénování neuronové sítě.

Poslední fází je testování autonomního řízení pomocí výstupů CNN v reálném čase. Opět jsou využity Python skripty pro pořizování snímků a sériovou komunikaci s Arduinem, podobně jako při sběru dat. Zároveň je využito předzpracování a inference CNN pro predikce míry zatáčení. Tyto skripty jsou implementovány v kontejneru Docker kvůli snadné správě závislostí knihoven.

5.2 POUŽITÉ TECHNOLOGIE

5.2.1 TensorFlow

TensorFlow představuje open-source platformu pro strojové učení, charakterizovanou svou flexibilitou a širokým ekosystémem nástrojů, knihoven a komunitních zdrojů. Tato rozmanitost umožňuje snadnou tvorbu a nasazení aplikací využívajících strojového učení (Why TensorFlow, 2015).

Původně vyvinut výzkumníky a inženýry z týmu Google Brain, TensorFlow sloužil primárně k účelům výzkumu v oblasti strojového učení a neuronových sítí. Nicméně, jeho univerzální povaha umožňuje využití i v jiných odvětvích (Tensorflow, 2015).

TensorFlow poskytuje stabilní rozhraní API pro Python a C++, a také rozhraní API, které, ač nezaručeně zpětnou kompatibilitu, umožňuje integraci s dalšími programovacími jazyky (Tensorflow, 2015).

5.2.2 Keras

Keras je uživatelsky přívětivá knihovna pro hluboké učení, která umožňuje vytvářet, trénovat a testovat neuronové sítě s minimálním úsilím. Poskytuje jednoduché a intuitivní aplikační rozhraní pro vytváření modelů na platformě TensorFlow. Keras nabízí širokou škálu vrstev pro vytváření modelů, včetně konvolučních, rekurentních a plně propojených vrstev, a umožňuje jednoduché experimentování s různými architekturami sítí a optimalizačními algoritmy (Get started, 2015).

5.2.3 Matplotlib

Matplotlib je knihovna naprogramovaná v jazyce Python určená pro vizualizaci dat. Nabízí širokou škálu možností pro tvorbu statických, interaktivních a animovaných grafů. Své využití nachází především v oblasti analýzy dat, vědeckého výzkumu a prezentace výsledků (Matplotlib: Visualization with Python, 2012).

5.2.4 Pandas

Knihovna Pandas je nástroj pro analýzu a manipulaci s daty v jazyce Python. Poskytuje efektivní a flexibilní nástroje pro práci s tabulkovými daty, umožňuje načítání, filtrování, transformaci a agregaci dat pomocí DataFrame objektů, což jsou dvourozměrné datové struktury. Pandas je široce používaný pro práci s daty ve vědeckém výzkumu, průmyslové analýze a finančním modelování díky své snadné použitelnosti a rozsáhlé sadě funkcí pro manipulaci s daty (About pandas, 2020).

5.2.5 NumPy

NumPy je knihovna určená pro vědecké výpočty v Pythonu, která poskytuje množství efektivních nástrojů pro práci s multidimenzionálními poli, lineární algebrou, Fourierovou transformací a generováním pseudonáhodných čísel. NumPy je široce používanou knihovnou v oblastech jako jsou strojové učení, analýza dat nebo vědecké simulace (Numpy – About us, 2020).

5.2.6 OpenCV

OpenCV je open-source knihovna primárně určená pro aplikace počítačové vidění a zpracování obrazu. Poskytuje širokou škálu funkcí pro práci s obrazovými daty nebo videi, včetně načítání a ukládání souborů, manipulace s obrazy, detekci a rozpoznávání objektů, sledování pohybu a mnoho dalších. OpenCV je často používána v akademických výzkumech i průmyslových

aplikacích pro svou flexibilitu, efektivitu a podporu mnoha programovacích jazyků, včetně C++, Pythonu a Javy (About OpenCV, 2019).

5.2.7 Imgaug

Knihovna imgaug je nástroj pro augmentaci obrazů. Nabízí širokou škálu transformací, jako je rotace, posunutí, zvětšení nebo změna jasu. Pomocí této knihovny lze vytvářet různorodá trénovací data pro strojové učení a zlepšit výkonnost a schopnost generalizace modelů v oblasti počítačového vidění (Docs – imgaug, 2017).

5.2.8 Evdev

Knihovna evdev je nástroj pro práci se vstupními zařízeními v Linuxu, jako jsou klávesnice, myši nebo joysticky. Poskytuje jednoduché rozhraní pro předávání událostí generovaných v kernelu přímo do uživatelského prostoru od zařízení typicky umístěných v adresáři */dev/input/* (Introduction – Python-evdev, 2016).

S využitím této knihovny lze snadno implementovat různé funkce, jako je detekce stisknutých kláves, pohyb kurzoru myši nebo zachytávání událostí joysticku, což z ní činí užitečný nástroj pro vývoj v oblasti vstupních periferií v Linuxovém prostředí.

Tato knihovna je využita pro čtení událostí z ovladače DualSense, jako stisky tlačítek levé klávesnice nebo polohy joysticku.

5.2.9 PySerial

Knihovna PySerial je nástroj pro práci se sériovými porty v jazyce Python. Díky zapouzdření umožňuje snadnou komunikaci se zařízeními připojenými pomocí USB, jako jsou mikrokontroléry Arduino, senzory nebo jiná zařízení. PySerial poskytuje jednoduché a intuitivní rozhraní pro odesílání a přijímání dat po sériovém portu, což umožňuje propojení a komunikaci mezi různými zařízeními a softwarovými aplikacemi v prostředí Python (PySerial – Overview, 2016).

5.2.10 VSCode

Visual Studio Code (VSCode) je integrované vývojové prostředí (IDE) vyvinuté společností Microsoft, které nabízí širokou škálu funkcí pro efektivní programování na platformě Jetson Nano. Konkrétně byla využita open-source varianta VSCode Code OSS, kterou vyvíjí Microsoft společně s vývojářskou komunitou. To znamená, že existuje mnoho zdrojů a příspěvků na fórech týkajících se používání IDE. Code OSS také poskytuje intuitivní

uživatelské rozhraní, různé možnosti konfigurace a bohatou nabídku doplňků, což umožňuje efektivně vytvářet, ladit a spravovat projekty (Visual Studio Code – Open Source ("Code – OSS"), 2015).

5.2.11 PyCharm

PyCharm je IDE používané primárně pro Python, které nabízí velké množství funkcí pro efektivní vývoj softwaru, jako např. inteligentní doplňování kódu, vestavěný debugger nebo automatické instalování knihoven. Dále je PyCharm snadno rozšiřitelný pomocí pluginů, které umožňují přizpůsobit IDE potřebám konkrétního projektu. Uživatelsky přívětivé rozhraní a pokročilé nástroje pro ladění a správu projektů dělají z PyCharm vhodnou volbu pro řešení rozsáhlejších projektů v Pythonu (PyCharm – The Python IDE for data science and web development, 2010).

5.2.12 Docker

Docker je open-source platforma pro izolaci aplikací a souvisejících závislostí a do kontejnerů. Kontejnery poskytují izolaci vývojové úlohy a její oddělení od konkrétního prostředí počítače (Use These! Jetson Docker Containers Tutorial, 2023).

Společnost NVIDIA poskytuje repozitáře s image dockeru l4t-ml, který obsahuje TensorFlow, PyTorch a další populární knihovny pro strojové učení, jako jsou scikit-learn, scipy nebo Pandas, předinstalované pro prostředí Python 3 (NVIDIA L4T ML, 2022).

Docker zjednodušuje vývoj, nasazení a správu aplikací pomocí standardizovaných kontejnerů, což vede k větší flexibilitě a efektivitě ve vývoji softwaru.

5.2.13 JupyterLab

JupyterLab je open-source IDE navržené pro interaktivní výpočty, vizualizace dat a sdílení vědeckých projektů. S jeho pomocí lze psát kód v různých programovacích jazycích, včetně Pythonu, R a Julia, a ihned zobrazovat výsledky pomocí bohatých nástrojů pro vizualizaci a editaci dat, což umožňuje efektivní práci na vědeckých a analytických projektech.

Instance serveru JupyterLab je automaticky spouštěna s Docker kontejnerem na počítači Jetson Nano (NVIDIA L4T ML, 2022). Jedná se o server na lokální síti, výhodou tedy je, že se k Jetsonu lze dálkově připojit pomocí desktopového počítače. To je vhodné např. při testování autonomie vozidla.

5.3 SBĚR DAT

Data pro trénování sítě byla sbírána na trénovací trase, kterou lze vidět na obrázku 15. Sběr dat probíhal tak, že vozítko bylo manuálně řízeno pomocí bezdrátového ovladače DualSense, přičemž byly nepřetržitě ukládány snímky trasy a korespondující hodnoty zatáčení. Frekvence snímání kamery byla stanovena na 10 FPS. Použití vyšší snímkové frekvence by vedlo k zahrnutí snímků, které jsou si velmi podobné, a tudíž by neposkytovaly další relevantní informace (Bojarski, 2016).

Trasa byla projeta celkem 29krát, a to v obou směrech. Pro zajištění vyšší rozmanitosti trénovacích dat byly snímky pořizovány za různých podmínek osvětlení. Celkem bylo nasbíráno 5836 snímků.

Obrázek 15 – Trénovací trasa.

5.3.1 Skript *data_collection_main.py*

Jedná se o hlavní Python skript pro spouštění systému pro sběr dat. Vzhledem k experimentální povaze tohoto systému nenabízí grafické uživatelské rozhraní a spouští se z konzole prostřednictvím příkazu *python data_collection_main.py*.

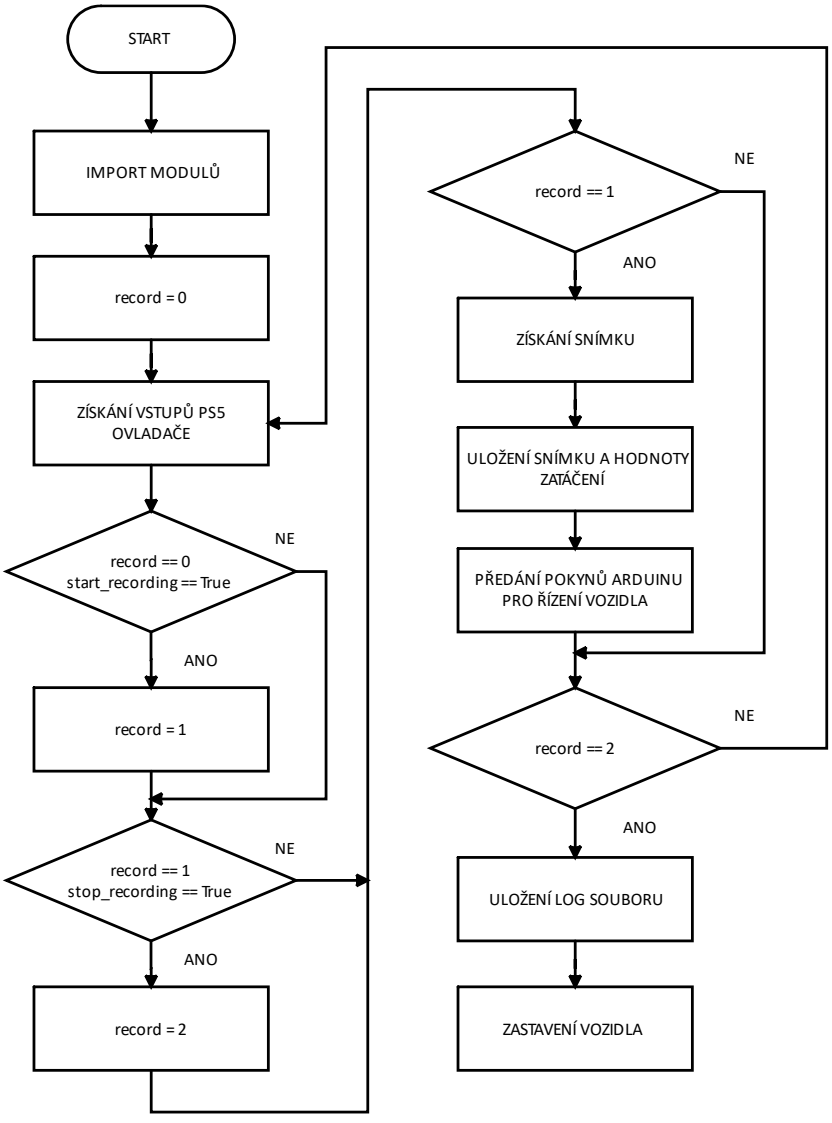
Ke správnému fungování sběru dat jsou využity tyto podpůrné Python moduly:

- *camera_module*: Skript určený k pořizování snímků trasy.
- *ps5_controller_module*: Skript pro monitorování stavu DualSense ovladače.
- *data_collection*: Skript pro zajištění ukládání snímků a log souboru.
- *arduino_module*: Odeslání dat o rychlosti a zatáčení do Arduina.

Sběr dat se zahájí stisknutím tlačítka levé šipky na ovladači DualSense. Poté program vstoupí do nekonečné smyčky (viz obrázek 16), kde neustále monitoruje stav levých tlačítek a polohu

pravého joysticku ovladače (viz obrázek 17). Paralelně se pořizují a ukládají snímky trasy. Pro unikátní identifikaci jednotlivých snímků je využito časové značky. Zároveň jsou do Arduina předávány údaje o rychlosti a hodnotě zatáčení na základě horizontální polohy pravého joysticku ovladače pro řízení pohybu vozítka (viz obrázek 18).

Sběr dat lze zastavit stiskem tlačítka pravé šipky nebo klávesovou kombinací *ctrl+c*. Po ukončení sběru dat se automaticky uloží log soubor a Arduinu je předána informace o nulové rychlosti a zatáčení, což způsobí zastavení pohybu vozítka (viz obrázek 19).



Obrázek 16 – Vývojový diagram skriptu *data_collection_main.py*.

```
speed, steering, start_coll, stop_coll = get_controller_data()
```

Obrázek 17 – Ukázka čtení stavů vybraných vstupů ovladače pomocí metody ze skriptu *ps5_controller_module.py*.

```
image = camera.getFrame()            # Získání snímku
save_data(image, steering)            # Uložení snímku
send_data_to_arduino(0.7, steering)   # Předání dat Arduinu
```

Obrázek 18 – Ukázka akcí po zahájení sběru dat. Metodu *save_data* poskytuje skript *data_collection.py* a metodu *send_data_to_arduino* skript *arduino_module.py*.

```
save_log()  # Uložení CSV log souboru
camera.close()
break
```

Obrázek 19 – Ukázka kódu při ukončení sběru dat.

5.3.2 Skript *camera_module.py*

Tento skript implementuje funkcionalitu pro práci s kamerou v Pythonu od GitHub uživatele Jetson Hacks (JetsonHacksNano / CSI-Camera, 2019). Pro efektivní sběr dat je nutné provádět několik úloh současně, z toho důvodu je využit threading. Zde je stručný přehled struktury tohoto skriptu:

- Konfigurace GStreameru: V první části je konfigurována kamera pro zachytávání obrazových dat, ukázku konfigurace lze vidět na obrázku 26.
- Třída *FrameReader*: Tato třída čte snímky z kamery a ukládá je do fronty. Má metodu pro přidání fronty a metodu pro získání snímku.
- Třída *Previewer*: Třída, která zobrazuje náhled obrazu z kamery v reálném čase pomocí OpenCV
- Třída *Camera*: Třída, která propojuje *FrameReader* a *Previewer* a poskytuje rozhraní pro získání snímku pomocí metody *getFrame()*.

5.3.3 Skript *ps5_controller_module.py*

Tento skript je navržen k interakci s herním ovladačem DualSense připojeným k počítači Jetson Nano pomocí rozhraní */dev/input/*. Skript čte vstupy z ovladače a na základě těchto vstupů aktualizuje různé proměnné, důležité pro řízení sběru dat. Pro tento skript je opět využit threading.

Pro čtení vstupů z ovladače byla implementována metoda *get_controller_data()*. Tato metoda vrací aktuální výstupy ovladače ve formě vektoru obsahujícího čtyři hodnoty. První z nich, označená jako *speed*, je nastavena na hodnotu 1 po stisknutí tlačítka horní šipky na ovladači a na 0 po jeho uvolnění.

Druhá hodnota, *steering*, reprezentuje spojitou hodnotu zatáčení a odpovídá horizontální poloze pravého joysticku. Tato hodnota se pohybuje v rozmezí od -1 do 1.

Třetí prvek vektoru, *start_coll*, je logická hodnota nastavená na *True* po stisknutí levé šipky. Poslední položka, *stop_coll*, je také logická hodnota, která se nastaví na *True* po stisku pravé šipky.

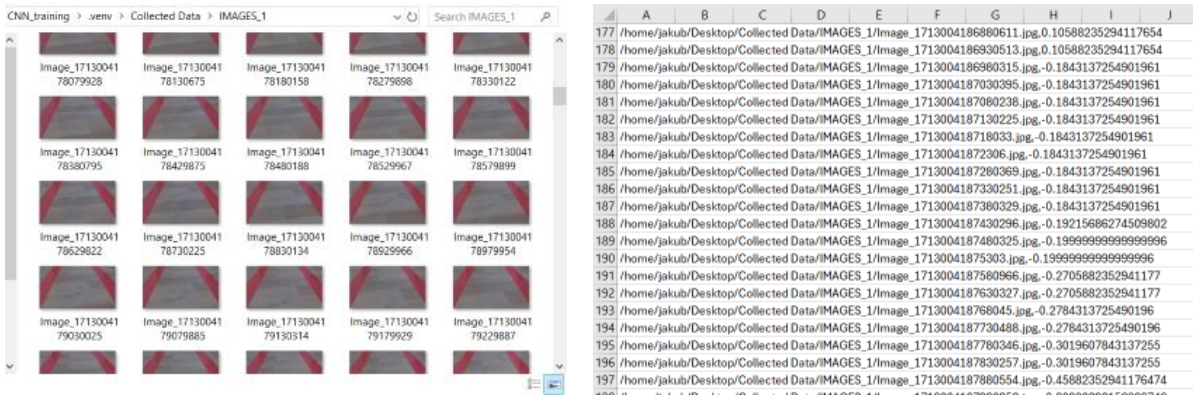
5.3.4 Skript *data_collection.py*

Tento skript je navržen pro systematické ukládání snímků během procesu sběru dat a pro následné vygenerování log souboru. Ukázku výstupu skriptu lze vidět na obrázku 20.

Jednou z funkcí skriptu je vytvoření adresářové struktury pro ukládání snímků. V případě existence předchozích adresářů se snímky je vytvořen nový adresář s číslem vyšším, než je nejvyšší číslo v názvu existujících adresářů.

Pro účely ukládání snímků a generování log souboru jsou v tomto skriptu implementovány následující metody:

- *save_data(image, steering)*: Tato metoda je definována k uložení snímku. Zároveň je vložena systémová cesta ke snímku a korespondující hodnota zatáčení do určených listů pro následné vytvoření log souboru. Každý snímek je uložen s unikátním názvem zahrnujícím časovou známku.
- *save_log()*: Tato metoda vytváří pandas dataframe pomocí listu systémových cest obrázků a listu hodnot zatáčení a ukládá je do log souboru ve formátu CSV. Zároveň uživatele informuje o počtu uložených snímků a úspěšném vytvoření log souboru.



Obrázek 20 – Ukázka výstupu sběru dat. Na levé straně je ukázka složky se snímky trasy. Na pravé straně je vygenerovaný log soubor ve formátu CSV.

5.3.5 Skript *arduino_module.py*

Tento skript implementuje sériovou komunikaci mezi počítačem Jetson Nano a Arduinem s cílem řídit vozidlo při sběru dat. Hlavním úkolem je přenášet informace o rychlosti a zatáčení.

Pro odesílání dat je k dispozici metoda *send_data_to_arduino*, která přijímá dva argumenty: *speed* a *steering*. Tyto hodnoty jsou reprezentovány čísly s pohyblivou řádovou čárkou a jsou odesílány ve formátu řetězce s třemi desetinnými místy, oddělenými novým řádkem. Ukázku kódu pro sériovou komunikaci lze vidět na obrázku 21.

```
ser = serial.Serial('/dev/ttyACM0', 115200)
time.sleep(1)

def send_data_to_arduino(speed, steering):
    data_string = "{:.3f}\n{:.3f}\n".format(speed, steering)
    ser.write(data_string.encode())
```

Obrázek 21 – Ukázka kódu pro sériovou komunikaci s Arduinem.

5.3.6 Skript *DC_motors_control.ino*

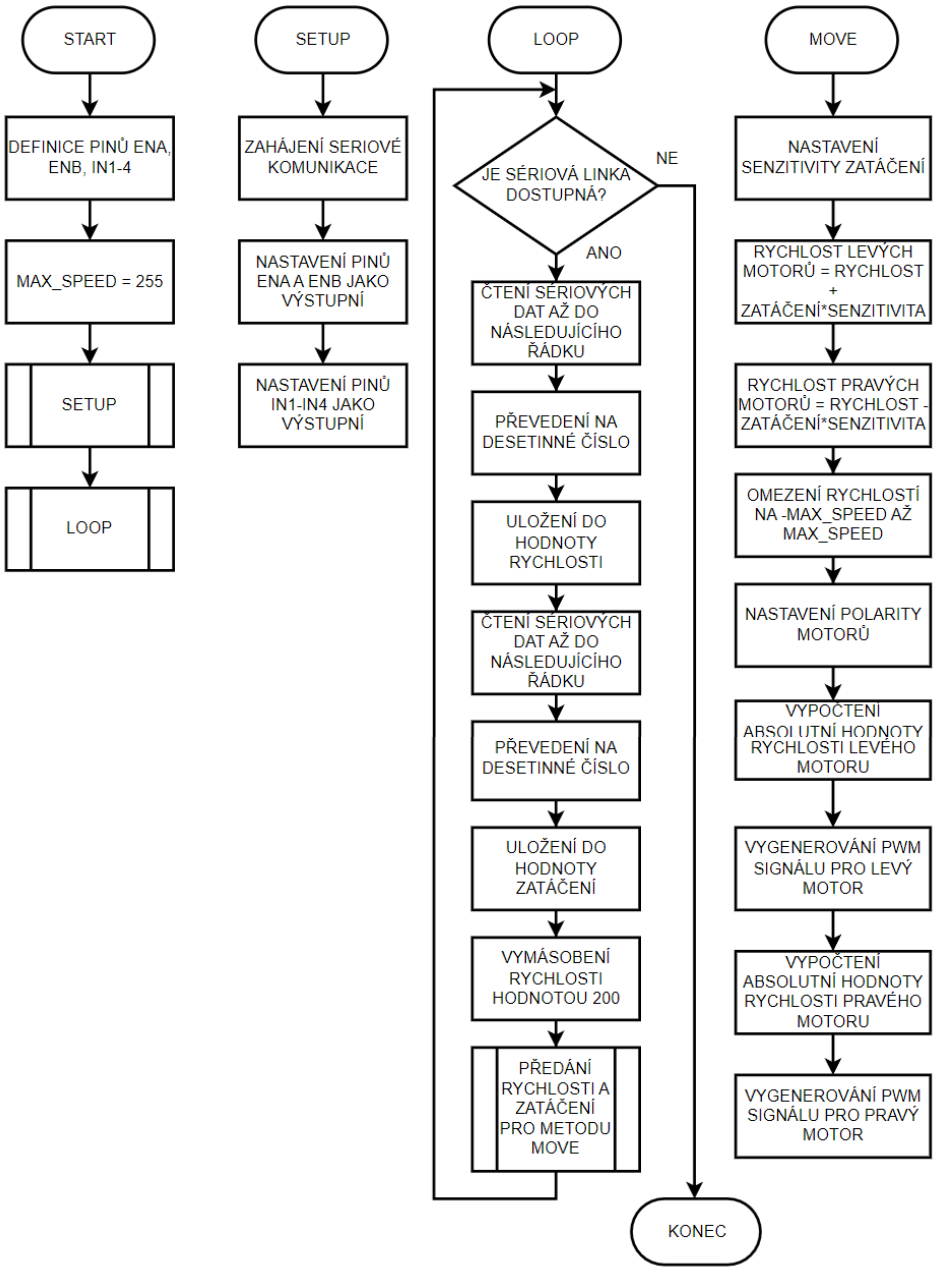
Tento C++ skript je nahrán na Arduinu a je určen pro řízení rychlosti a polarity DC motorů pro ovládání vozidla. Pro řízení motorů jsou přes sériovou linku přijímány údaje o rychlosti a hodnoty zatáčení od nadřazeného Python skriptu. Výstupem skriptu jsou PWM signály pro řízení rychlosti a řídící signály pro řízení polarity. Vývojový diagram tohoto skriptu lze vidět na obrázku 24. Nekonečnou smyčku programu lze vidět na obrázku 22. Výpočet rychlosti motorů je zachycen na obrázku 23.

```
void loop() {
  while (Serial.available()) {
    String input = Serial.readStringUntil('\n');
    float motors_speed = input.toFloat();

    input = Serial.readStringUntil('\n');
    float steering_val = input.toFloat();

    motors_speed *= 200;

    move(motors_speed, steering_val);
  }
}
```

Obrázek 22 – Ukázka metody loop v Arduino skriptu.

```
float steering_sensitivity = 1.5;
steering_val *= float(MAX_SPEED);
int leftMotorSpeed = int(motors_speed +
(steering_val*steering_sensitivity));
int rightMotorSpeed = int(motors_speed -
(steering_val*steering_sensitivity));

leftMotorSpeed = constrain(leftMotorSpeed, -MAX_SPEED, MAX_SPEED);
rightMotorSpeed = constrain(rightMotorSpeed, -MAX_SPEED, MAX_SPEED);
```

Obrázek 23 – Ukázka kódu pro vypočtení rychlosti motorů.



Obrázek 24 – Vývojový diagram pro Arduino program.

5.4 TRÉNOVÁNÍ SÍTĚ

Pro trénování modelu byly použity dva skripty. Prvním z nich je *training_main.py*, který představuje hlavní spustitelný skript pro trénování. Druhým skriptem je *training_utils.py*, který obsahuje podpůrné metody pro proces trénování.

5.4.1 Volba hyperparametrů

V rámci trénování sítě byly testovány různé konfigurace hyperparametrů, jako je velikost dávky (*batch size*), velikost kroku učení (*learning rate*), změny architektury sítě a přidání dropout

vrstev. Proběhly také experimenty se změnami velikostí filtrů a využití různých aktivačních funkcí. Původní architektura s aktivační funkcí ELU obvykle dosahovala nižších hodnot ztráty, ve všech testovaných modelech tedy byla využita ELU.

Vzhledem k omezené velikosti datasetu se ukázalo, že aplikace dropout vrstev měla pozitivní vliv na snížení příznaků nadměrného přizpůsobení modelu na trénovací data.

Pro trénování byly také zkoušeny různé optimalizační algoritmy, jmenovitě Adam (*Adaptive Moment Estimation*), RMSprop (*Root Mean Squared propagation*) a SGD (*Stochastic Gradient Descend*). Obecně algoritmus Adam vedl k nejrychlejší konvergenci, byl tedy zvolen pro optimalizaci všech uvedených modelů.

Celkem bylo vytvořeny čtyři modely s hyperparametry, které jsou uvedené v tabulce 1:

- Model 1: Původní, nezměněná architektura sítě.
- Model 2: Odstranění poslední konvoluční vrstvy, přidání dropout vrstev za první a druhou plně propojenou vrstvu.
- Model 3: Původní architektura, přidání jedné dropout vrstvy za první plně propojenou vrstvu.
- Model 4: Původní architektura, přidání dropout vrstev za první a druhou plně propojenou vrstvu.

Tabulka 1 – Konfigurace hyperparametrů sítě.

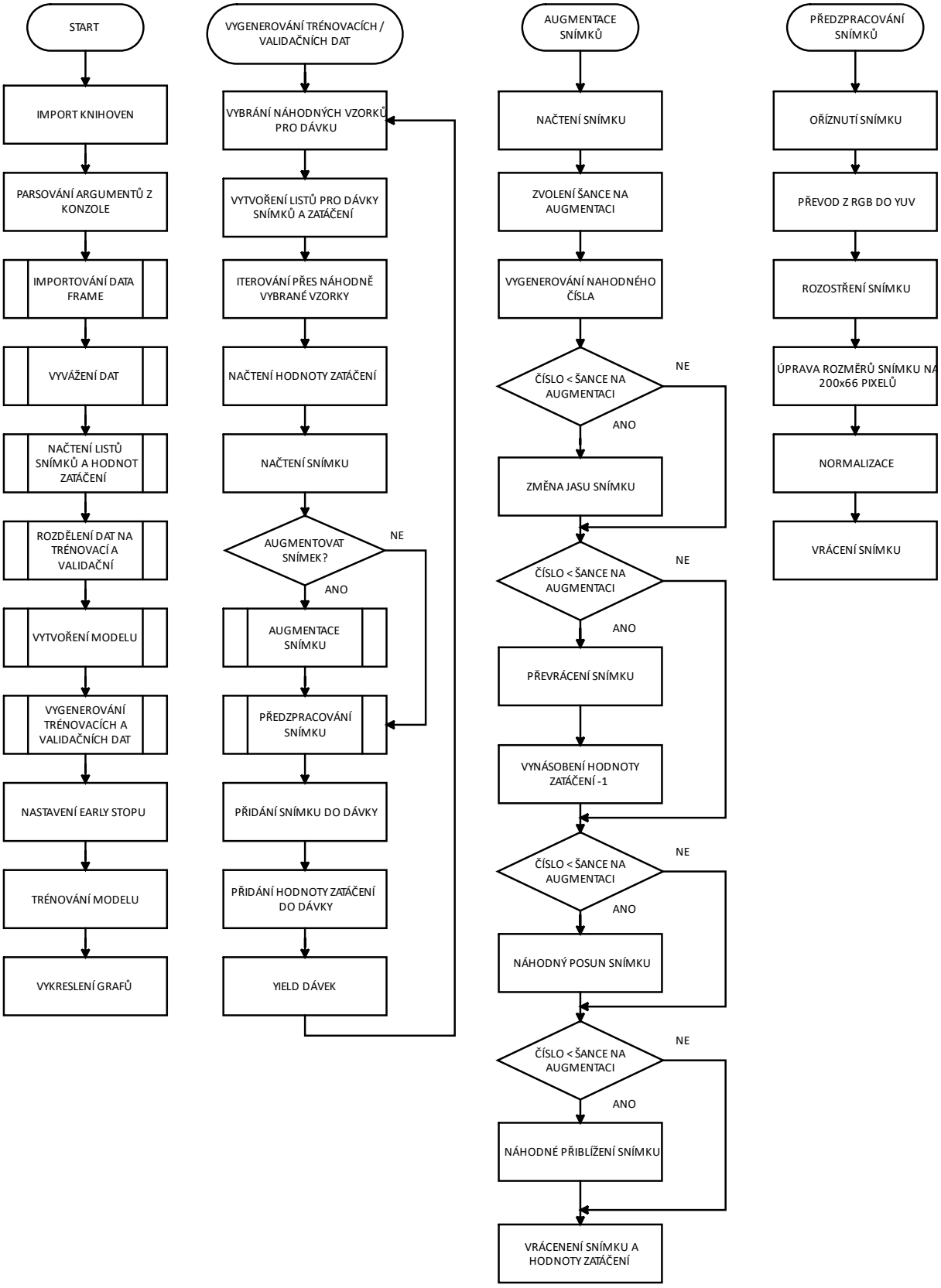
Hyperparametr	**Model 1**	**Model 2**	**Model 3**	**Model 4**
Velikost dávky	32	32	32	25
Veikost kroku učení	0,0001	0,0001	0,0001	0,0005
Dropout	-	0,2	0,2	0,2
Aktivační funkce	ELU	ELU	ELU	ELU
Optimizér	ADAM	ADAM	ADAM	ADAM

5.4.2 Skript *training_main.py*

Jedná se o hlavní Python skript pro spouštění systému pro trénování sítě. Vývojový diagram lze vidět na obrázku 25. Stejně jako u systému pro sběr dat je využito konzolové prostředí. Skript se spouští příkazem *python training_main.py* a přijímá tyto volitelné argumenty:

- --*epochs*: Počet epoch pro trénování modelu, výchozí hodnota je 200.

- --*batch_size*: Velikost dávky, výchozí hodnota je 32.
- --*learning_rate*: Velikost kroku pro učení sítě, výchozí hodnota je 0,0001.
- --*early_stop*: Počet epoch, po kterých dojde k ukončení trénování sítě, pokud se nesníží validační ztráta, výchozí hodnota je 25.
- --*data_path*: Systémová cesta pro trénovací data, výchozí hodnota je „Collected Data“.



Obrázek 25 – Vývojový diagram procesu trénování sítě.

5.4.3 Skript *training_utils.py*

Jedná se o podpůrný skript pro trénování sítě. Obsahuje tyto důležité metody:

- *import_data_frame(path)*: Tato metoda vrací Pandas data frame, který je vytvořen pomocí CSV log souborů umístěných na *path*. Vytvořený data frame obsahuje dva sloupce, první obsahuje systémové cesty na snímky a druhý hodnoty zatáčení.
- *balance_data(data_frame, max_samples)*: Tato metoda je určená pro vyvážení dat. Argument *data_frame* je Pandas data frame a *max_samples* je maximální počet snímků pro každou hodnotu zatáčení. Metoda vrátí vyvážený data frame.
- *load_data_lists(path, data_frame)*: Metoda, která vrací listy obsahující systémové cesty na obrázky a hodnoty zatáčení, vytvořené pomocí data framu.
- *augment_image(image_path, steering)*: Metoda pro augmentaci snímku, viz obrázek 26.
- *preprocess(image)*: Metoda pro předzpracování snímku, viz obrázek 27.
- *generate_train_data(images_path, steering_list, batch_size, augment_data, increase_factor)*: Metoda pro vygenerování dat pro trénování sítě. Augmentace se povolí argumentem *augment_data*. Argument *increase_factor* slouží pro zvětšení datasetu pomocí augmentace.
- *create_model(learning_rate)*: Metoda pro vytvoření CNN Tensorflow modelu.

```
def augment_image(image_path, steering):
    image = mpimg.imread(image_path)
    aug_chance = 0.5
    if np.random.rand() < aug_chance:
        brightness = iaa.Multiply((0.5, 1.3))
        image = brightness.augment_image(image)
    if np.random.rand() < aug_chance:
        image = cv2.flip(image, 1)
        steering = -steering
    if np.random.rand() < aug_chance:
        pan = iaa.Affine(translate_percent={"x": (-0.3, 0.3),
                                            "y": (-0.1, 0.1)})
        image = pan.augment_image(image)
    if np.random.rand() < aug_chance:
        zoom = iaa.Affine(scale=(1, 1.3))
        image = zoom.augment_image(image)
    return image, steering
```

Obrázek 26 – Ukázka kódu pro augmentaci snímku.

```
def preprocess(image):
    image = image[20:360, :, :]                    # oříznutí
    image = cv2.cvtColor(image, cv2.COLOR_RGB2YUV)  # převod z RGB do YUV
    image = cv2.GaussianBlur(image, (3, 3), 0)      # rozostření
    image = cv2.resize(image, (200, 66))            # úprava rozměrů
    image = image / 255                             # normalizace
    return image
```

Obrázek 27 – Ukázka kódu pro předzpracování snímku.

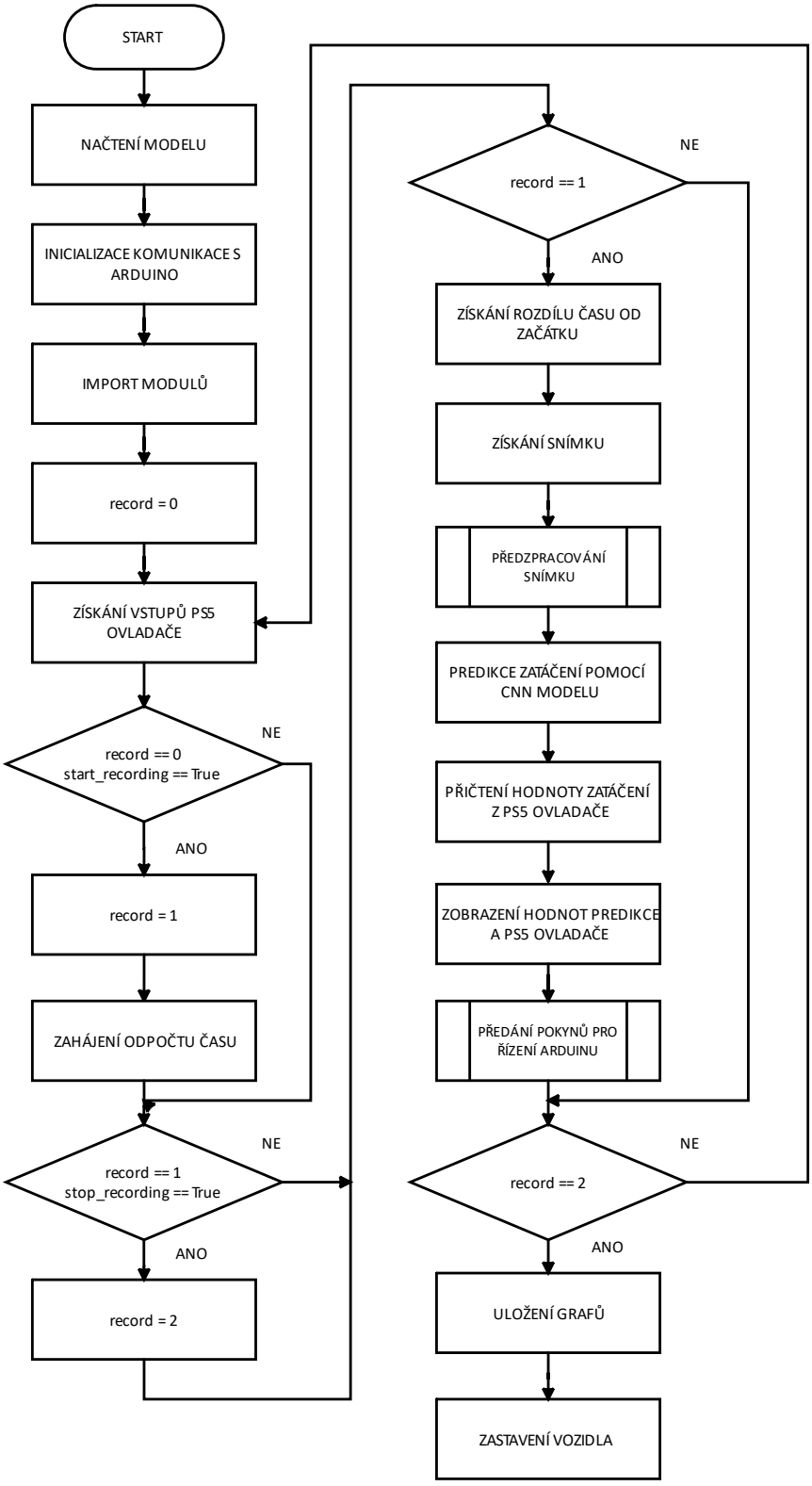
5.5 TESTOVÁNÍ AUTONOMIE MODELU

Model byl testován na testovací trase, jejíž podoba je znázorněna na obrázku 28. Pro implementaci programu byla využita technologie Docker v prostředí Jupyter Lab. Pro spuštění testů autonomie slouží notebook skript *autonomy_test.ipynb*, jehož vývojový diagram lze rovněž nalézt na obrázku 29.

Testování autonomie se stejně jako systém pro sběr dat spouští stiskem levé šipky na připojeném PS5 ovladači. Po zahájení testování je vozidlo automaticky řízeno pomocí predikcí CNN modelu, s možností manuálního zásahu pravým joystickem ovladače. Během testování jsou jak predikce, tak manuální zásahy zaznamenávány pro pozdější vyhodnocení a zobrazení v grafické podobě.

Testování lze kdykoli přerušit stiskem pravé šipky. V takovém případě je vozidlo zastaveno a jsou vykresleny odpovídající grafy.

Obrázek 28 – Testovací trasa.



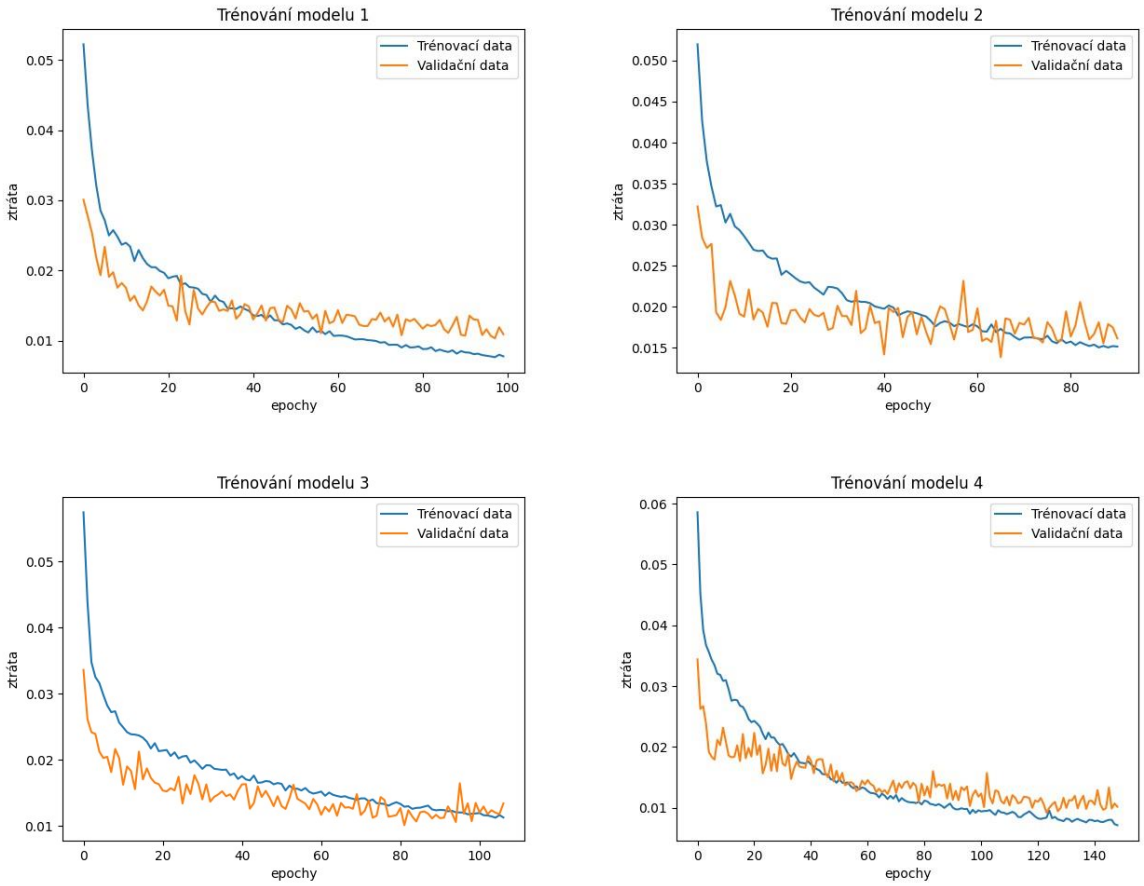
Obrázek 29 – Vývojový diagram skriptu pro testování autonomie.

6 VÝSLEDKY A TESTOVÁNÍ

6.1 TRÉNOVÁNÍ A VYHODNOCENÍ MODELŮ

Nastavení předčasného ukončení trénování bylo aplikováno na všechny modely s podmínkou, že trénování bude ukončeno, pokud nedojde ke snížení validační ztráty během posledních 25 epoch. Výsledky trénování jsou prezentovány na obrázku 30.

Modely byly zároveň vyhodnoceny na neznámém datasetu obsahujícím celkem 1198 snímků a příslušných údajů o zatáčení, sbíraných na testovací trase. Výsledky vyhodnocení jednotlivých modelů jsou shrnuty v tabulce 2. Pro následné testování spolehlivosti sledování trasy byl vybrán model 4 na základě nejnižší testovací ztráty.



Obrázek 30 – Trénování modelů.

Tabulka 2 – Vyhodnocení modelů.

Ztráta	Model 1	Model 2	Model 3	Model 4
Trénovací ztráta	0,0085	0,0151	0,0109	0,0089
Validační ztráta	0,0109	0,0165	0,0136	0,0101
Testovací ztráta	0,0388	0,0357	0,0321	0,0295

6.2 VYHODNOCENÍ SPOLEHLIVOSTI SLEDOVÁNÍ TRASY

Míra autonomnosti, tedy spolehlivost automatického sledování testovací trasy, je odhadnuta procentem času, kdy síť dokáže samostatně řídit vozidlo. Metrikou je počítání manuálních zásahů do řízení. Tyto zásahy jsou vyvolány v okamžiku, kdy vozidlo celou kontaktní plochou jednoho z kol najede na pásku, která vymezuje okraj testovací trasy. Pro kvantifikaci míry autonomie je definován následující vztah:

$$A = \left(1 - \frac{N \cdot T_Z}{T}\right) \cdot 100 \tag{9}$$

kde A je procento autonomie, N je počet zásahů do řízení, T_Z je průměrný čas potřebný pro zásah a T je celková doba testování.

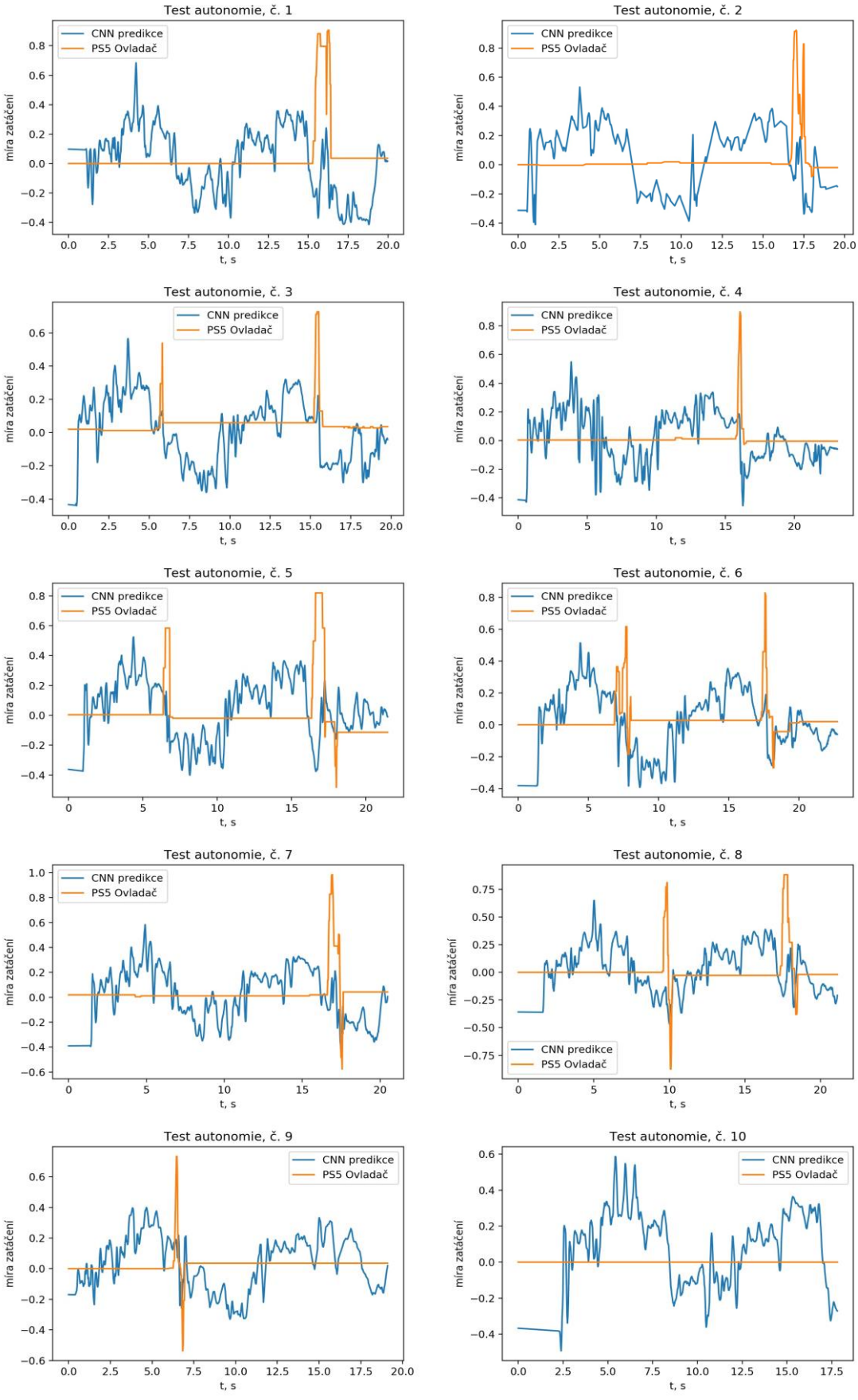
Testování proběhlo na testovací trase za různých podmínek osvětlení. Průměrný čas potřebný pro zásah byl stanoven na 0,9 s. Výsledky testování jsou obsaženy v tabulce 3 a statistické vyhodnocení výsledků je uvedeno v tabulce 4. Naměřené grafy jsou zachyceny na obrázku 31.

Tabulka 3 – Výsledky měření autonomie.

Test	**Naměřené hodnoty, Tz = 0,9 s**		**Autonomie (%)**
	Počet zásahů	**Doba testování (s)**	
1	1	19,96	95,49
2	1	19,55	95,40
3	2	20,21	91,09
4	1	23,94	96,24
5	3	21,45	87,41
6	2	22,73	92,08
7	1	20,48	95,61
8	2	21,13	91,48
9	1	19,11	95,29
10	0	17,81	100

Tabulka 4 – Statistické vyhodnocení výsledků.

Statistická míra	**Autonomie (%)**
Aritmetický průměr	94,01
Medián	95,35
Nejlepší případ	100
Nejhorší případ	87,41



Obrázek 31 – Naměřené grafy pro testování autonomie.

6.3 INTERPRETACE VÝSLEDKŮ

Z výsledků statistické analýzy testování sledování trasy vyplývá, že vozidlo bylo schopné po většinu doby testování poměrně úspěšně automaticky sledovat trasu. Nicméně, v určitých úsecích testovací trasy se vyskytl problém s najížděním na okraj trasy, což není v souladu s definovanými kritérii. Tento problém se projevoval zejména při průjezdu poslední ostré zatáčky, což je patrné z grafů naměřených dat, kde bylo často zapotřebí manuálního zásahu přibližně mezi patnáctou až sedmnáctou sekundou testování.

Tento jev lze částečně vysvětlit omezeným trénovacím datasetem. Přesto vozidlo dokázalo s určitými nepřesnostmi v naprosté většině případů automaticky projet testovací trasu od začátku do konce bez výrazného odchýlení.

V poslední řadě je možné predikce CNN programově ladit pomocí zesílení výstupu modelu. Zvýšením zesílení lze snížit problém nedostatečného zatáčení v ostrých zatáčkách. Nicméně, nevýhodou mohou být zesílené oscilace modelu nebo neadekvátní reakce v určitých situacích.

ZÁVĚR

V této práci byla v teoretické části představena rešerše autonomních vozidel, zabývající se principy strojového vidění a senzoriky pro autonomní vozidla. Dále byly popsány principy konvolučních neuronových sítí a jejich uplatnění pro metody detekce a sledování trasy. V závěru teoretické části byla navržena metodika pro sběr dat pro následné trénování sítě prostřednictvím metody „učení s učitelem“, včetně popisu procesu rozdělení a anotace dat s využitím CSV formátu. Také byla představena architektura konvoluční neuronové sítě pro automatické řízení vozidla.

V praktické části byl navržen model vozidla pro ověření funkce automatického sledování trasy v reálných podmínkách. Pro centrální řízení vozidla byl zvolen počítač NVIDIA Jetson Nano, který je vhodný pro aplikace vyžadující paralelní výpočty. Pro řízení DC motorů, které pohánějí vozidlo, byla použita kombinace mikrokontroléru Arduino Uno Rev 3 a H můstku L928N. K pořizování obrazových dat byla vybrána kamera IMX219-77, která díky technologii CMOS nabízí nízkou energetickou náročnost a kompaktní rozměry pro snadnou integraci do omezeného prostoru vozidla. Pro napájení Jetsonu a Arduina byla využita konvenční powerbanka, zatímco pro motorový pohon byl zvolen Li-Pol akumulátor Zippy 6000.

Software pro sběr dat byl naprogramován v jazyce Python, který je vhodnou volbou pro práci s knihovnami strojového učení. Současně byl implementován program pro Arduino, určený k řízení motorů vozidla. Výsledkem byl funkční systém pro sběr dat, který bylo možné ovládat prostřednictvím PS5 ovladače. Výstupem systému byl trénovací dataset obsahující snímky trasy anotované hodnotami zatáčení zaznamenaných s pomocí ovladače.

Dále byly navrženy Python skripty pro trénování TensorFlow CNN modelu na základě nasbíraných dat. Trénování modelů proběhlo pro urychlení procesu na desktopovém počítači. Celkem byly testovány čtyři modely s různými variacemi hyperparametrů. Pro následné testování byl vybrán model s nejnižší testovací ztrátou na testovacím datasetu.

V závěru praktické části byl v prostředí Jupyter Lab navržen Python notebook skript určený pro testování autonomie vozidla. Tento skript sloužil k provedení celkem deseti testů, během kterých byla podle stanovených kritérií hodnocena spolehlivost automatického sledování trasy. Celková kvantifikace spolehlivosti sledování trasy byla provedena na základě výsledků získaných z testů.

Výsledkem práce je vozidlo, které je schopné automaticky sledovat libovolně definovanou trasu v závislosti na kvalitě trénovacího datasetu. V práci byly trénovací i testovací trasy vyznačeny barevnou páskou. V tomto případě by pro sledování trasy teoreticky bylo možné využít jednoduší systém, jako např. „line follower“ na bázi detektoru barvy. Výhoda využití CNN spočívá v automatické extrakci vlastností ze vstupních dat, jako např. okraje nebo zakřivení trasy. To v praxi znamená, že systém by měl být schopen rozpoznat a sledovat jakkoliv definovanou trasu, buď explicitně či implicitně. Nicméně je důležité upozornit na potenciální nevýhody využití CNN pro celý rozhodovací proces řízení, jako je nedostatečná interpretovatelnost rozhodovacího procesu nebo potenciální neadekvátní reakce na šum ve vstupních obrazových datech.

Navržený model vozidla není uzpůsobený pro konkrétní praktické využití, neboť je určen spíše k ověření funkce automatického sledování trasy. Nicméně softwarovou část je možné integrovat do jiného systému s odlišnými hardwarovými prvky.

POUŽITÁ LITERATURA

A Comprehensive Guide to Convolutional Neural Networks, 2018. In: *Towards Data Science* [online]. [cit. 2024-03-08]. Dostupné z: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

About OpenCV, 2019. *OpenCV* [online]. [cit. 2024-04-24]. Dostupné z: https://opencv.org/about/

About pandas, 2020. *Pandas* [online]. [cit. 2024-04-24]. Dostupné z: https://pandas.pydata.org/about/

AHMAD, Husnain, Asra TARIQ, Amir SHEHZAD, et al., 2019. Stealth technology: Methods and composite materials—A review. *Polymer Composites* [online]. **40**(12), 4457-4472 [cit. 2024-03-06]. ISSN 0272-8397. Dostupné z: doi:10.1002/pc.25311

ALONSO, Luciano, Vicente MILANÉS, Carlos TORRE-FERRERO, Jorge GODOY, Juan P. ORIA a Teresa DE PEDRO, 2011. Ultrasonic Sensors in Urban Traffic Driving-Aid Systems. *Sensors* [online]. **11**(1), 661-673 [cit. 2024-03-06]. ISSN 1424-8220. Dostupné z: doi:10.3390/s110100661

Autonomous Vehicle Sensors – Making Sense of The World, 2021. In: *Wards Intelligence* [online]. [cit. 2024-03-07]. Dostupné z: https://wardsintelligence.informa.com/wi965823/autonomous-vehicle-sensors---making-sense-of-the-world

Basic CNN architecture and kernel, 2020. In: *ResearchGate* [online]. [cit. 2024-03-12]. Dostupné z: https://www.researchgate.net/figure/Basic-CNN-architecture-and-kernel-A-typical-CNN-consists-of-several-component-types_fig2_343441194

BIGAS, M., E. CABRUJA, J. FOREST a J. SALVI, 2006. Review of CMOS image sensors. *Microelectronics Journal* [online]. **37**(5), 433-451 [cit. 2024-05-11]. ISSN 00262692. Dostupné z: doi:10.1016/j.mejo.2005.07.002

BISHOP, Richard, 2005. *Intelligent Vehicle Technology and Trends*. Artech House. ISBN 978-1580539111.

BOJARSKI, Mariusz, Davide Del TESTA, Daniel DWORAKOWSKI, et al., 2016. *End to End Learning for Self-Driving Cars* [online]. 1-9 [cit. 2024-04-16]. Dostupné z: doi: arXiv:1604.07316v1

CLEVERT, D., T. UNTERTHINER a S. HOCHREITER, 2015. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). *Under Review of ICLR2016* [online]. 1-14 [cit. 2024-03-14]. Dostupné z: doi:10.48550/arxiv.1511.07289

CYBENKO, G., 1989. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems* [online]. **2**(4), 303-314 [cit. 2024-03-18]. ISSN 0932-4194. Dostupné z: doi:10.1007/BF02551274

Death of Elaine Herzberg, 2018. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation [cit. 2024-05-14]. Dostupné z: https://en.wikipedia.org/wiki/Death_of_Elaine_Herzberg

Docs - imgaug, 2017. *Imgaug* [online]. [cit. 2024-04-24]. Dostupné z: https://imgaug.readthedocs.io/en/latest/

EVERINGHAM, Mark, Luc VAN GOOL, Christopher K. I. WILLIAMS, John WINN a Andrew ZISSERMAN, 2010. The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision* [online]. **88**(2), 303-338 [cit. 2024-03-07]. ISSN 0920-5691. Dostupné z: doi:10.1007/s11263-009-0275-4

FAN, Yu-Cheng, Chitra Meghala YELAMANDALA, Ting-Wei CHEN, Chun-Ju HUANG a Ismail BUTUN, 2021. Real-Time Object Detection for LiDAR Based on LS-R-YOLOv4 Neural Network. *Journal of Sensors* [online]. 2021-5-26, **2021**, 1-11 [cit. 2024-03-06]. ISSN 1687-7268. Dostupné z: doi:10.1155/2021/5576262

Get started, 2015. *Keras* [online]. [cit. 2024-04-24]. Dostupné z: https://keras.io/

GONZALES, Rafael C. a Richard E. WOODS, 2018. *Digital image processing*. 4. New York: Pearson. ISBN 978-1-292-22304-9.

HÄNE, Christian, Lionel HENG, Gim Hee LEE, Friedrich FRAUNDORFER, Paul FURGALE, Torsten SATTLER a Marc POLLEFEYS, 2017. 3D visual perception for self-driving cars using a multi-camera system: Calibration, mapping, localization, and obstacle detection. *Image and Vision Computing* [online]. **68**, 14-27 [cit. 2024-03-06]. ISSN 02628856. Dostupné z: doi:10.1016/j.imavis.2017.07.003

HE, Kaiming, Xiangyu ZHANG, Shaoqing REN a Jian SUN, 2016. Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* [online]. IEEE, 770-778 [cit. 2024-03-13]. ISBN 978-1-4673-8851-1. Dostupné z: doi:10.1109/CVPR.2016.90

HEINZLER, Robin, Florian PIEWAK, Philipp SCHINDLER a Wilhelm STORK, 2020. CNN-Based Lidar Point Cloud De-Noising in Adverse Weather. *IEEE Robotics and Automation Letters* [online]. **5**(2), 2514-2521 [cit. 2024-03-06]. ISSN 2377-3766. Dostupné z: doi:10.1109/LRA.2020.2972865

HORNIK, Kurt, Maxwell STINCHCOMBE a Halbert WHITE, 1989. Multilayer feedforward networks are universal approximators. *Neural Networks* [online]. **2**(5), 359-366 [cit. 2024-03-18]. ISSN 08936080. Dostupné z: doi:10.1016/0893-6080(89)90020-8

HUBEL, D. H. a T. N. WIESEL, 1959. Receptive fields of single neurones in the cat's striate cortex. *The Journal of Physiology* [online]. **148**(3), 574-591 [cit. 2024-03-11]. ISSN 0022-3751. Dostupné z: doi:10.1113/jphysiol.1959.sp006308

CHEN, Chenyi, Ari SEFF, Alain KORNHAUSER a Jianxiong XIAO, 2015. DeepDriving: Learning Affordance for Direct Perception in Autonomous Driving. *2015 IEEE International*

Conference on Computer Vision (ICCV) [online]. IEEE, 2722-2730 [cit. 2024-04-16]. ISBN 978-1-4673-8391-2. Dostupné z: doi:10.1109/ICCV.2015.312

CHEN, Liang-Chieh, George PAPANDREOU, Iasonas KOKKINOS, Kevin MURPHY a Alan L. YUILLE, 2018. DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. *IEEE Transactions on Pattern Analysis and Machine Intelligence* [online]. 2018-4-1, **40**(4), 834-848 [cit. 2024-04-16]. ISSN 0162-8828. Dostupné z: doi:10.1109/TPAMI.2017.2699184

ImageVI's: the software that collects the vegetation indices you need: user manual, 2020. In: *ResearchGate* [online]. [cit. 2024-03-07]. Dostupné z: https://www.researchgate.net/figure/Fundamental-steps-on-digital-image-processing_fig3_341394746/actions#reference

Introduction — Python-evdev, 2016. *Python-evdev* [online]. [cit. 2024-04-24]. Dostupné z: https://python-evdev.readthedocs.io/en/latest/

JANESICK, James R., 2001. *Scientific Charge-Coupled Devices* [online]. [cit. 2024-05-11]. Dostupné z: doi:10.1117/3.374903

Jetson Nano Developer Kit, 2024. In: *NVIDIA Developer* [online]. [cit. 2024-04-19]. Dostupné z: https://developer.nvidia.com/embedded/jetson-nano-developer-kit

JetsonHacksNano / CSI-Camera, 2019. *GitHub* [online]. [cit. 2024-05-14]. Dostupné z: https://github.com/JetsonHacksNano/CSI-Camera

Kamerový modul Sony IMX219-77, 2018. In: *Botland* [online]. [cit. 2024-04-22]. Dostupné z: https://botland.cz/prislusenstvi-pro-nvidia/14784-kamerovy-modul-sony-imx219-77-8mpx-kompatibilni-s-jetson-nano-waveshare-16507-5904422322564.html

KLEIN, Itzik, 2022. A Hybrid Model and Learning-Based Adaptive Navigation Filter. *IEEE Transactions on Instrumentation and Measurement* [online]. **71**, 1-11 [cit. 2024-03-12]. ISSN 0018-9456. Dostupné z: doi:10.1109/TIM.2022.3197775

KLEINFELDER, S., L. SUKHWAN, L. XINQIAO a A. EL GAMAL, 2001. A 10000 frames/s CMOS digital pixel sensor. *IEEE Journal of Solid-State Circuits* [online]. **36**(12), 2049-2059 [cit. 2024-05-11]. ISSN 00189200. Dostupné z: doi:10.1109/4.972156

KRIZHEVSKY, Alex, Ilya SUTSKEVER a Geoffrey E. HINTON, 2017. ImageNet classification with deep convolutional neural networks. *Communications of the ACM* [online]. 2017-05-24, **60**(6), 84-90 [cit. 2024-03-12]. ISSN 0001-0782. Dostupné z: doi:10.1145/3065386

LECUN, Y., B. BOSER, J. S. DENKER, D. HENDERSON, R. E. HOWARD, W. HUBBARD a L. D. JACKEL, 1989. Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation* [online]. **1**(4), 541-551 [cit. 2024-03-11]. ISSN 0899-7667. Dostupné z: doi:10.1162/neco.1989.1.4.541

LECUN, Yann, 2021. [online]. 2021, 24.3.2021 [cit. 2024-05-03]. Dostupné z: https://twitter.com/ylecun/status/1374699540545159178?lang=cs

LECUN, Yann, Yoshua BENGIO a Geoffrey HINTON, 2015. Deep learning. *Nature* [online]. 2015-05-28, **521**(7553), 436-444 [cit. 2024-03-13]. ISSN 0028-0836. Dostupné z: doi:10.1038/nature14539

LOWRIE, James W., Mark THOMAS, Keith GREMBAN a Matthew TURK, 1985. The Autonomous Land Vehicle (ALV) Preliminary Road-Following Demonstration. *Proceedings of the SPIE*. **579**, 336-350.

Matplotlib: Visualization with Python, 2012. *Matplotlib* [online]. [cit. 2024-04-24]. Dostupné z: https://matplotlib.org/

MCCULLOCH, Warren S. a Walter PITTS, 1943. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics* [online]. **5**(4), 115-133 [cit. 2024-03-17]. ISSN 0007-4985. Dostupné z: doi:10.1007/BF02478259

MIT, Roi, Yoav ZANGVIL a Dror KATALAN, 2020. *Analyzing Tesla's Level 2 Autonomous Driving System Under Different GNSS Spoofing Scenarios and Implementing Connected Services for Authentication and Reliability of GNSS Data* [online]. 621-646 [cit. 2024-03-07]. Dostupné z: doi:10.33012/2020.17687

MNIH, Volodymyr, Koray KAVUKCUOGLU, David SILVER, et al., 2015. Human-level control through deep reinforcement learning. *Nature* [online]. 2015-02-26, **518**(7540), 529-533 [cit. 2024-04-17]. ISSN 0028-0836. Dostupné z: doi:10.1038/nature14236

NI, Jianjun, Yinan CHEN, Yan CHEN, Jinxiu ZHU, Deena ALI a Weidong CAO, 2020. A Survey on Theories and Applications for Self-Driving Cars Based on Deep Learning Methods. *Applied Sciences* [online]. **10**(8) [cit. 2024-03-06]. ISSN 2076-3417. Dostupné z: doi:10.3390/app10082749

Numpy - About us, 2020. *Numpy* [online]. [cit. 2024-04-24]. Dostupné z: https://numpy.org/about/

NVIDIA L4T ML, 2022. *NVIDIA NGC | CATALOG* [online]. [cit. 2024-04-24]. Dostupné z: https://catalog.ngc.nvidia.com/orgs/nvidia/containers/l4t-ml

POMERLEAU, Dean, 1989. ALVINN: an autonomous land vehicle in a neural network. *Proceedings of (NeurIPS) Neural Information Processing Systems*. Morgan Kaufmann, 305-313

PRINCE, Simon J.D., 2023. *Understanding Deep Learning* [online]. The MIT Press [cit. 2024-05-14]. ISBN 978-0262048644. Dostupné z: http://udlbook.com

PyCharm - The Python IDE for data science and web development, 2010. *JetBrains* [online]. [cit. 2024-04-24]. Dostupné z: https://www.jetbrains.com/pycharm/

PySerial - Overview, 2016. *GitHub* [online]. [cit. 2024-04-24]. Dostupné z: https://github.com/pyserial/pyserial

Robotické vozítko 4WD, 2020. In: *XDUINO* [online]. [cit. 2024-04-22]. Dostupné z: https://xduino.cz/index.php?id_product=139&id_product_attribute=0&rewrite=roboticke-vozitko-4wd-26cm&controller=product

RUMELHART, David E., Geoffrey E. HINTON a Ronald J. WILLIAMS, 1986. Learning representations by back-propagating errors. *Nature* [online]. **323**(6088), 533-536 [cit. 2024-03-19]. ISSN 0028-0836. Dostupné z: doi:10.1038/323533a0

SHUTTLEWORTH, Jennifer, 2019. SAE Standards News: J3016 automated-driving graphic update. *SAE International* [online]. [cit. 2024-02-28]. Dostupné z: https://www.sae.org/news/2019/01/sae-updates-j3016-automated-driving-graphic

SCHMIDHUBER, Jürgen, 2015. Deep learning in neural networks: An overview. *Neural Networks* [online]. **61**, 85-117 [cit. 2024-03-15]. ISSN 08936080. Dostupné z: doi:10.1016/j.neunet.2014.09.003

SCHULTE-TIGGES, Joschua, Marco FÖRSTER, Gjorgji NIKOLOVSKI, Michael REKE, Alexander FERREIN, Daniel KASZNER, Dominik MATHEIS a Thomas WALTER, 2022. Benchmarking of Various LiDAR Sensors for Use in Self-Driving Vehicles in Real-World Environments. *Sensors* [online]. **22**(19) [cit. 2024-03-06]. ISSN 1424-8220. Dostupné z: doi:10.3390/s22197146

SONG, Myung-Sin, 2006. Wavelet image compression. *Contemporary Mathematics*. 2.

Tensorflow, 2015. *GitHub* [online]. [cit. 2024-05-01]. Dostupné z: https://github.com/tensorflow/tensorflow

Types of pooling layers, 2022. In: *ScienceDirect* [online]. [cit. 2024-03-15]. Dostupné z: https://www.sciencedirect.com/topics/mathematics/pooling-layer

Use These! Jetson Docker Containers Tutorial, 2023. *JetsonHacks* [online]. [cit. 2024-04-24]. Dostupné z: https://jetsonhacks.com/2023/09/04/use-these-jetson-docker-containers-tutorial/

VALIENTE, Rodolfo, Mahdi ZAMAN, Sedat OZER a Yaser P. FALLAH, 2019. Controlling Steering Angle for Cooperative Self-driving Vehicles utilizing CNN and LSTM-based Deep Networks. *2019 IEEE Intelligent Vehicles Symposium (IV)* [online]. IEEE, 2423-2428 [cit. 2024-04-16]. ISBN 978-1-7281-0560-4. Dostupné z: doi:10.1109/IVS.2019.8814260

VIERLING, Kerri T, Lee A VIERLING, William A GOULD, Sebastian MARTINUZZI a Rick M CLAWGES, 2008. Lidar: shedding new light on habitat characterization and modeling. *Frontiers in Ecology and the Environment* [online]. **6**(2), 90-98 [cit. 2024-03-06]. ISSN 1540-9295. Dostupné z: doi:10.1890/070001

Visual Studio Code - Open Source ("Code - OSS"), 2015. *GitHub* [online]. [cit. 2024-04-24]. Dostupné z: https://github.com/microsoft/vscode

VOGELSANG, Lukas, Sharon GILAD-GUTNICK, Evan EHRENBERG, Albert YONAS, Sidney DIAMOND, Richard HELD a Pawan SINHA, 2018. Potential downside of high initial visual acuity. *Proceedings of the National Academy of Sciences* [online]. 2018-10-

30, **115**(44), 11333-11338 [cit. 2024-05-01]. ISSN 0027-8424. Dostupné z: doi:10.1073/pnas.1800901115

WEBSTER, Tim L., 2010. Flood Risk Mapping Using LiDAR for Annapolis Royal, Nova Scotia, Canada. *Remote Sensing* [online]. **2**(9), 2060-2082 [cit. 2024-03-06]. ISSN 2072-4292. Dostupné z: doi:10.3390/rs2092060

What is an Autonomous Car?, 2024. *Synopsys* [online]. [cit. 2024-03-19]. Dostupné z: https://www.synopsys.com/automotive/what-is-autonomous-car.html

What is machine vision, 2024. *Cognex | Machine Vision and Barcode Readers* [online]. [cit. 2024-02-23]. Dostupné z: https://www.cognex.com/what-is/machine-vision/what-is-machine-vision

What is the difference between CCD and CMOS image sensors, 2010. *HowStuffWorks* [online]. [cit. 2024-05-10]. Dostupné z: https://electronics.howstuffworks.com/cameras-photography/digital/question362.htm

Why TensorFlow [online], 2015. [cit. 2024-04-23]. Dostupné z: https://www.tensorflow.org/about

WITMAN, Paul D., Jim PRIOR, Scott MACKELPRANG, 2022. Teaching Case Risk assignments and tradeoffs on the road to driverless vehicles: Yours, mine and ours. *Proceedings of the EDSIG Conference*. p. 4901.

YANG, Ji Hyun, Jimin HAN a Jung-Mi PARK, 2017. Toward Defining Driving Automation from a Human-Centered Perspective. *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems* [online]. New York, NY, USA: ACM, 2017-05-06, 2248-2254 [cit. 2024-03-06]. ISBN 9781450346566. Dostupné z: doi:10.1145/3027063.3053101

ZHANG, Yongbing, Tao SHEN, Xiangyang JI, Yun ZHANG, Ruiqin XIONG a Qionghai DAI, 2018. Residual Highway Convolutional Neural Networks for in-loop Filtering in HEVC. *IEEE Transactions on Image Processing* [online]. **27**(8), 3827-3841 [cit. 2024-03-12]. ISSN 1057-7149. Dostupné z: doi:10.1109/TIP.2018.2815841

Zippy 6000mAh 2s 7.4v 35c 45c Hardcase. In: *Ubuy* [online]. [cit. 2024-04-22]. Dostupné z: https://www.algeria.ubuy.com/en/product/2P441UDI-zippy-6000mah-2s-7-4v-35c-45c-hardcase-lipo-traxxas-hpi-deans-5000mah-turnigy

PŘÍLOHY

Příloha A – On-line

Příloha A

Příloha k bakalářské práci

Model vozidla s automatickým sledováním trasy pomocí konvoluční neuronové sítě

Jakub Zahradník

On-line

Obsah

1 – Text bakalářské práce ve formátu PDF

2 – Úplné zdrojové kódy pro sběr dat, trénování CNN a testování

3 – Natrénované CNN modely

4 – Odkaz na video zachycující ukázku funkce systému