UNIVERZITA PARDUBICE

Fakulta elektrotechniky a informatiky

Automatické sledování objektu v prostoru

Bc. Petr Stibor

Diplomová práce

2023

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2022/2023

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Petr Stibor**
Osobní číslo:	**I21310**
Studijní program:	**N0714A150005 Automatické řízení**
Téma práce:	**Automatické sledování objektu v prostoru**
Zadávající katedra:	**Katedra řízení procesů**

Zásady pro vypracování

Cílem práce je návrh a implementace systému pro automatické sledování objektu v prostoru založeného na technologii konvolučních sítí. Student v rámci práce vytvoří zařízení pro polohování kamerového systému s dvěma stupni volnosti k umožnění snímání a sledování objektu v jeho okolí. Polohovací zařízení bude možné ovládat nadřazeným systémem přes standardní komunikační sběrnici. Ovládání bude umožněno v manuálním i automatickém režimu. V automatickém režimu bude natočení kamerového systému řízené posunem definovaného objektu detekovaného v několika po sobě jdoucích snímcích. Detekční systém bude založen na technologii konvolučních sítí.

Teoretická část: Rešerše zadaného tématu, popis technického řešení konstrukce polohovací jednotky. Stručná rešerše existujících nástrojů pro detekci a lokalizaci objektů v obrazových datech založených na metodách umělé inteligence. Popis senzorové techniky pro sběr dat pro učení neuronové sítě.

Praktická část: Návrh a realizace systému pro automatické polohování. Sběr a analýza dat. Návrh a implementace softwarového nástroje pro řízení polohovací jednotky užívající vybrané paradigma umělých neuronových sítí. Testování a zhodnocení systému pomocí běžných metrik používaných pro hodnocení neuronových sítí. Technická dokumentace polohovací jednotky. Popis softwaru včetně testovacího scénáře demonstrujícího použití kompletního systému.

Rozsah pracovní zprávy: **cca 50 stran**
Rozsah grafických prací:
Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam doporučené literatury:

GOODFELLOW, Ian, Yoshua BENGIO a Aaron COURVILLE. *Deep learning*. Cambridge, Massachusetts: The MIT Press, [2016]. ISBN 978-02-620-3561-3.
GONZALEZ, Rafael C. a Richard E. WOODS. *Digital image processing*. Fourth edition. New York: Pearson, [2018]. ISBN 978-013-3356-724.
MAIXNER, Ladislav. *Mechatronika: učebnice*. Brno: Computer Press, [2006]. Učebnice (Computer Press). ISBN 80-251-1299-3.

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

Datum zadání diplomové práce: **8. listopadu 2022**
Termín odevzdání diplomové práce: **19. května 2023**

L.S.

Ing. Zdeněk Němec, Ph.D. v.r.
děkan

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

V Pardubicích dne 15. listopadu 2022

Prohlášení

Prohlašuji:

Tuto práci 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., autorský zákon, 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. 9/2012, bude práce zveřejněna v Univerzitní knihovně a prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 19. 5. 2023

Bc. Petr Stibor

Poděkování

Rád bych poděkoval panu Ing. Dominiku Štursovi za odborné vedení diplomové práce. Dále bych rád poděkoval všem ostatním pedagogům, kteří se podělili o drahocenné zkušenosti. Děkuji také své rodině a přátelům, kteří mě podporovali a drželi mi palce po celý průběh studia.

V Pardubicích dne 19. 5. 2023

Bc. Petr Stibor

ANOTACE

Cílem této diplomové práce je využít technologii konvolučních neuronových sítí pro vývoj sytému, který dokáže automaticky sledovat objekt v prostoru. Projekt zahrnuje návrh a implementaci kamerového polohovacího zařízení se dvěma stupni volnosti, které jsou zajišťovány servo motory. Zařízení lze ovládat ve dvou režimech, a to ručně nebo automaticky. Komunikace probíhá přes standartní komunikační sběrnici Modbus, což umožňuje bezproblémovou integraci s nadřazeným systémem. V automatickém režimu je rotace kamerového systému řízena pomocí detekčního algoritmu YOLO ve verzi YOLOv5, který v několika po sobě jsoucích snímcích detekuje pohyb předem definovaného objektu.

KLÍČOVÁ SLOVA

Konvoluční neuronové sítě, CCN, průmyslová kamera, polohovací zařízení, počítačové vidění, Modbus, detekce objektů, YOLOv5.

TITLE

AUTOMATIC OBJECT TRANKING AND DETECTION IN SPACE

ANNOTATION

The goal of this thesis is to utilize convolutional neural networks to develop a system capable of automatically tracking objects in space. The project encompasses the design and implementation of a camera positioning device equipped with two servo motors, providing two degrees of freedom. The device can be controlled manually or automatically. Communication occurs via a standard communication bus Modbus, facilitating seamless integration with the superior system. In automatic mode, the rotation of the camera system is governed by the YOLOv5 version of the YOLO detection algorithm. This algorithm detects the movement of a predetermined object across several consecutive frames.

KEYWORDS

Convolutional Neural Networks, CCN, Industrial Camera, Positioning Device, Computer Vision, Modbus, Object Detection, YOLOv5.

OBSAH

SEZNAM ZKRATEK A ZNAČEK

AP	average precision
API	Application Programming Interface
CCN	Convolutional neural netvork
CSP	Cross Stage Partial
FAIR	Facebook AI Research
FLOPs	floating point operations
FPN	Feature Pyramid Network
FPS	frames per second
GPIO	General-purpose input/output
IoU	Intersection over Union
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
mAP	mean average precision
MS COCO	Microsoft Common Objects in Context
OpenCV	Open Source Computer Vision Library
R-CNN	Region-based Convolutional Neural Network
RPN	region proposal network
ReLU	Rectified Linear Unit
ResNet	residual network
SSD	Single Shot Multibox Detector
Tanh	Hyperbolický tangens
TCP/IP	Transmission Control Protocol / Internet protocol
TPU	Tensor Processing Unit
TP	True positive
YOLO	You Only Look Once

SEZNAM ILUSTRACÍ

ÚVOD

Umělá inteligence a její používání je poslední dobou téma, které hýbe lidskou společností. Jednou z velkých oblastí využití je detekce objektů a lokalizace v obrazových datech. Schopnost automaticky identifikovat a lokalizovat objekty na snímcích má četné aplikace, včetně autonomních vozidel, sledovacích systémů, robotiky a rozšířené reality. Tato práce se zaměřuje na teoretické a praktické aspekty vývoje systému pro automatické polohování jednotky s dvěma stupni volnosti. Využívá se techniky detekce a lokalizace objektů na bázi umělé inteligence.

Teoretická část této práce začíná komplexní výzkumnou studií daného tématu, jejímž cílem je pochopit základní principy a výzvy spojené s automatickým polohováním. Dále obsahuje podrobný popis detekčních a klasifikačních algoritmů. Součástí je samozřejmě technické řešení konstrukce polohovací jednotky. To zahrnuje diskusi o nezbytných komponentách pro hardware a komunikační rozhraní, aby bylo možné přesně určit polohu na základě detekovaných objektů.

Praktická část této práce zahrnuje návrh a implementaci automatického polohovacího systému. To zahrnuje sběr dat pomocí vhodných senzorových technik pro usnadnění trénování neuronových sítí. Shromážděná data budou použita k trénování a optimalizaci modelů, což umožní přesnou detekci a lokalizaci objektů v reálném čase.

Součástí práce je také podrobný popis softwarového nástroje pro ovládání polohovací jednotky pomocí vybraného detekčního algoritmu YOLO. Tento nástroj používá model trénovaný na principu učení s učitelem neuronových sítí. Systém je dále testován a výsledky metrik popisující přesnost jsou součástí práce.

Pro komplexní pochopení celého systému je součástí práce rozebráno jednotlivé propojení částí funkčního celku ve formě technické dokumentace. Tato dokumentace nastiňuje specifikace návrhu a podrobnosti implementace hardwarových a softwarových komponent. Kromě toho bude popsán testovací scénář demonstrující praktickou aplikaci a užitečnost celého systému v reálném prostředí.

V závěru je hodnocení celé práce a jeho možné přispění do oblasti učení umělé inteligence. Jedná se o kombinaci teoretického výzkumu, analýzy dat, vývoje softwaru a praktického použití. Práce obsahuje inženýrský přístup k danému tématu, které by mohlo najít uplatnění v různých oblastech.

1 NEURONOVÁ SÍŤ

Neuronové sítě jsou součástí toho, co chápeme jako umělou inteligenci, která se nabývá na významu v různých oblastech. Aplikace neuronových sítí sahá od rozpoznání obrazu, řeči nebo řešení autonomního vozidla či celých výrobních linek. Neuronové sítě jsou modelovány podle lidského mozku. Mají schopnost analyzovat složitá data, rozpoznávat vzory. Neuronové sítě jsou navrženy tak, aby se časem mohli učit, zlepšovat a i předpovídat. Jednou z největších výhod u neuronových sítí je možnost klasifikace, segmentace nebo detekce vlastností bez asistence člověka. (Zelinka, 2005)

1.1 Architektura neuronové sítě

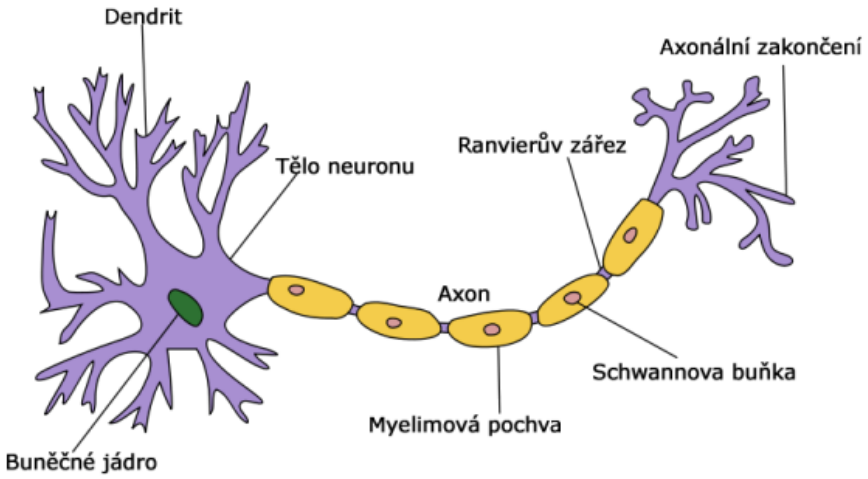
Základním kamenem neuronové sítě je jednoduchý perceptron nebo také lze říct neuron. Je no nejmenší jednotka a spojením několika neuronů vznikne vrstva. Model neuronu vychází z biologického neuronu. Mozek člověka je složen z velkého množství propojených buněk a mezi nejdůležitější patří právě neurony. Biologické neurony zpracovávají, přenášení a uchovávají informace. Každý neuron v mozku je navzájem propojen s dalšími až 5000 neurony a celkem jich v mozkové kůře najdeme okolo 15 miliard. (HABRNÁL, 2014)

Celá architektura se skládá z několika vrstev záleží na typu a složitosti řešeného problému. Přičemž první vrstvou, která nijak data netransformuje, je vstupní vrstva. Dále jsou dle typu sítě další.

1.2 Biologický neuron

Biologický neuron se skládá z těla neuronu *(somatu)*, které má rozměry jen několik mikrometrů. Tělo neuronu se dále dělí na buněčné jádro a dendrit. Dendrity jsou vlastně vstupy do neuronu z jiných neuronů, pomocí nichž je přivedena vstupní informace v podobě elektrického impulsu. Na opačné straně těla neuronu je axon. Axon je výstup neuronu a každý neuron má pouze jeden. Axon může vést k jiným neuronům nebo do svalů či žláz. Koncové větve axonu mají malé knoflíky zvané synaptické terminály, které uvolňují chemické neurotransmitery pro komunikaci s jinými neurony. Synaptické terminály jednoho neuronu se spojují s dendrity nebo buněčným tělem jiného neuronu ve spojení zvaném synapse. (Novák, 1993)

Chování neuronu lze zjednodušeně popsat takto. Na dendrit je přivedena informace (výboj), pokud se těchto výbojů sejde v těle neuronu více, může neuron reagovat. Každý neuron má vlastnost, která se nazývá práh neuronu. Tato hranice určuje, při jaké velikosti součtu vstupních impulsů neuron reaguje. Jinak řečeno, jak je neuron citlivý. Do doby, dokud není překročen práh citlivosti, neuron nereaguje. Pokud je práh překonán, neuron vyšle signál skrz axon do dalších neuronů. V tuto chvíli je neuron na určitý čas necitlivý na jiné vstupní podměty. Tento proces se znovu opakuje a pokud je práh stále překročen dochází k dalšímu vyslání impulsu. (Novák, 1993)

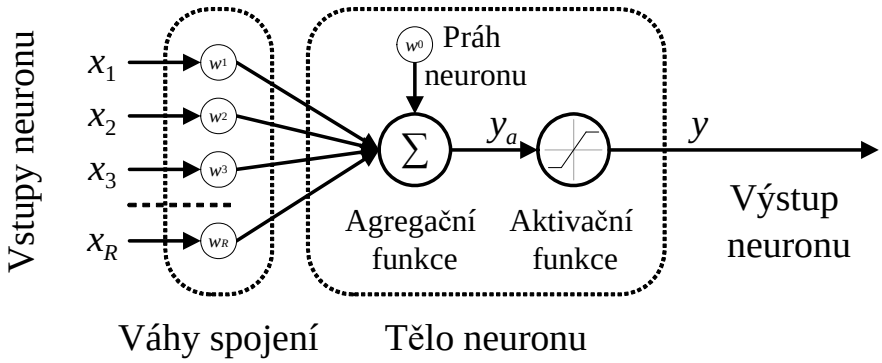


Obr. 1.1 – Biologický neuron (Neuron, 2009)

1.3 Umělý neuron

Pro umělý neuron zůstává hlavní myšlenka stejná jako u biologického neuronu, nicméně jeho struktura je velmi zjednodušena. Umělý neuron přijme pomocí vstupů signály ve formě čísel, které nějaký způsobem reprezentují vnější okolí nebo vlastnosti řešeného problému. Tyto čísla zpracuje, a pokud je celkový výsledný potenciál velký, vyšle vlastní signál. (Mařík, 1993)

Existuje několik typů modelů umělého neuronu, které se liší topologií nebo některými vlastnostmi. Nejčastěji používán je tzv. formální model, nebo podle autorů se mu také říká McCulloch-Pittsuv neuron. Vyobrazení struktury tohoto typu je na obrázku 1.2. (Doležel, 2016)



Obr. 1.2 – McCullochuv-Pittsuv model neuronu (Doležel, 2016)

1.3.1 Vstupy neuronu

Vstupy neuronu nahrazují dendrity, které i zde mohou být buď výstupy z jiných presynaptických neuronů, nebo představují vstupy z vnějšího okolí. Vstupy jsou tedy charakterizovány jedním vektorem X, který reprezentuje soubor konkrétních hodnot. Hodnoty vstupů dělíme na kvalitativní nebo kvantitativní. Kvalitativní jsou ty hodnoty, které zpravidla nabývají pouze dvou hodnot a zastupují zde booleovské schéma rozhodování. Například byl detekován pohybující se objekt, ano nebo ne. Druhým možným vstupem je kvantitativní hodnota, ta je vyjádřena reálným číslem. Příkladem může být množství kapaliny v nádrži nebo rychlost automobilu v daném úseku. Do kvantitativních vstupů patří také výstupy fuzzy logiky. (Doležel, 2016)

1.3.2 Váhy spojení

Váhy mezi jednotlivými neurony v neuronové síti ovlivňují celou síť. Každý vstup je ohodnocen určitou hodnotou, která reprezentuje citlivost. Tato vlastnost je také přebrána z biologického neuronu. Váhy jsou reálná čísla, které svou hodnotou určují vzájemnou důležitost a průchodnost spojení. Změnou vah u jednotlivých spojení je možné dosáhnout ideální shody mezi výstupem neuronové sítě a zkoumaným procesem. Určení velikostí váhy mezi neurony se nazývá konfluence. Výpočet konfluence je popsán následující rovnicí, kde operátor konfluence je možné nahradit prostým součinem. (Doležel, 2016)

$$z_i = x_i \oplus w_i \tag{1}$$

1.3.3 Práh neuronu

Stejně jako u biologického neuronu je u umělého také prahová hodnota. Tuto hodnotu musí neuron překonat, aby se mohl šířit neuronovou sítí, jedná se tedy o bariérovou hodnotu. Pokud je součet všech vstupních signálů vetší, než prahová hodnota neuron je aktivní v opačném případě odpovídá pasivní hodnotě. Obecně se zde ale prahová hodnota používá pro jakési odhlučnění okolního světa, a ne hodnot z předešlých neuronů. Princip určení prahu hodnot je tedy podobný jako u určení vah spojení. (Doležel, 2016)

1.3.4 Agregační funkce

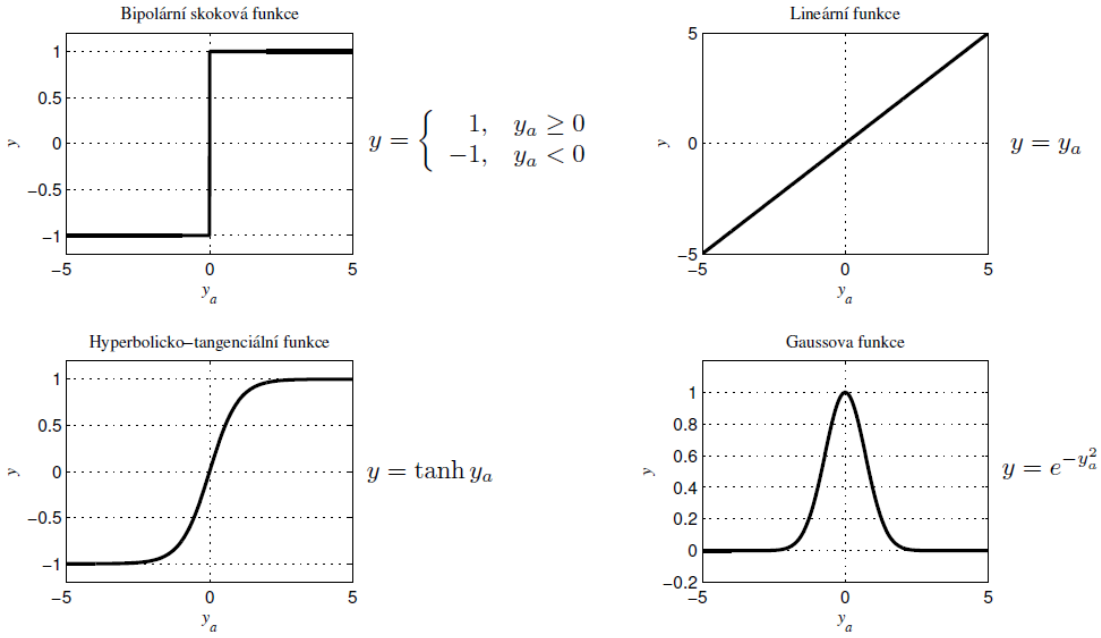
Někdy také popisována jako sumační funkce má za úkol sloučit vstupní signály neuronu. To tedy znamená transformovat vektor $x_i(i=0, 1,\ldots,R)$ na skalární signál y_a a ten je předán dál do aktivační funkce. Agregace je tedy zároveň seskupení a prahování pomocí obecného operátoru G. Obecnou agregaci operátorem G můžeme podle základního modelu nahradit operací sumace. Tím seskupíme všechny vstupy neuronu do jedné hodnoty. K tomu přičteme i prahové hodnoty w_0. (Doležel, 2016)

$$y_a = \sum_{i=1}^{n} x_i * w_i + w_0 \tag{2}$$

1.3.5 Aktivační funkce

Základní funkcí aktivační funkce je převést hodnotu potenciálu, tedy agregační funkce, na výstupní hodnotu neuronu. V počátečním tvaru formálního modelu byla jako aktivační funkce používána skoková změna. Nicméně dnes je tato funkce různorodá a aktivační funkce se dá rozdělit na lineární, nebo nelineární, případně spojitá a nespojitá.

Výběr správné podoby aktivační funkce záleží na typu řešeného problému a ovlivňuje technickou náročnost. Nejčastěji používané funkce jsou na obrázku 1.3, přičemž nejvíce používaná je hyperbolická tangenta. (Doležel, 2016)



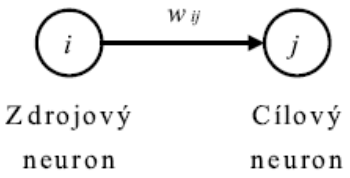
Obr. 1.3 – Vybrané aktivační funkce (Doležel, 2016)

1.3.6 Výstup neuronu

Do každého neuronu je tedy přivedeno několik vstupů, které jsou dle popisu modelu zpracovány a výsledkem je výstupní signál. Ten může být nadále vstupem do dalších neuronů. Stejně tak tomu je i u biologického neuronu. Je zvykem, že výstupní funkce je stejná jako výsledek aktivační funkce. Nicméně existují sítě, ve kterých se výstup neuronu určuje i na základě sousedících neuronů. Jedná se o tzv. soutěživé sítě. (HABRNÁL, 2014)

1.4 Obecné schéma neuronové sítě

Umělá neuronová síť může být popsána jednoduchým grafem s vrcholy a orientovaným hranami. Vrcholy představují jednotlivé neurony a hrany zase jejich vzájemné propojení. Do každého neuronu může být přivedeno několik hran. To je základní vlastnost, která vyplívá z modelu biologického neuronu. Spojení dvou neuronu je na obrázku 1.4. (Doležel, 2016)



Obr. 1.4 – Spojení neuronů (Doležel, 2016)

Podle umístění vzájemné vazby dělíme neurony na presynaptické, tedy zdrojové, a postsynaptické, nazývané cílové. Váha jejích spojení je označena symbolem w_{ij}, kde indexy označují nejprve zdrojový neuron a následně cílový neuron. (Doležel, 2016)

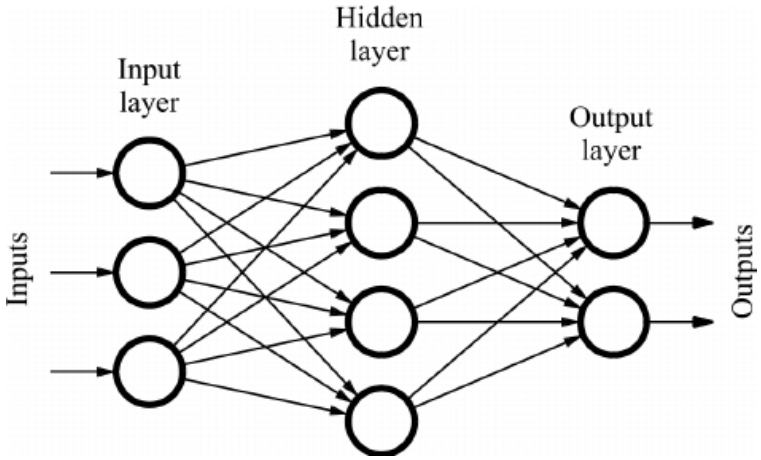
Význam neuronových sítí spočívá v jejich schopnosti provádět složité úkoly, jako je rozpoznávání obrazu a řeči, zpracování přirozeného jazyka a rozhodování. Důležitý pojmem je architektura sítě, která udává, jak jsou neurony v síti propojeny a uspořádány. Společným rysem je většinou vrstevnatá struktura. Samozřejmě mohou být i další například mřížková struktura. Jednotlivé neurony jsou tedy škálovány do určitých vrstev a podle polohy rozeznáváme:

a) Vstupní vrstvu, která zajišťuje vstup signálu z okolí nebo jiných neuronů;
b) Skrytou nebo normální vrstvu, těchto vrstev může být velmi mnoho;
c) Výstupní vrstvu, která předává signál do okolí.

Podle počtu vrstev a neuronů v každé z nich určujeme jak kvalitní a složitá bude výsledná neuronová síť. Neuronové sítě jsou rozděleny do tří kategorií podle toku signálů, a to dopředné neuronové sítě a rekurentní neuronové sítě. (Zelinka, 2005)

1.5 Dopředná neuronová síť

Dopředná neuronová síť je typem neuronové sítě, která je navržena tak aby zpracovávala vstupní signály do každé vrstvy a ty následně předala vrstvě další. V tomto typu jsou tedy připojeny výstupy neuronu k jedné nebo více vrstvám. Avšak každá vrstva má určité hodnotové ohodnocení a fungují tedy na principu kaskády, tedy jedním směrem.



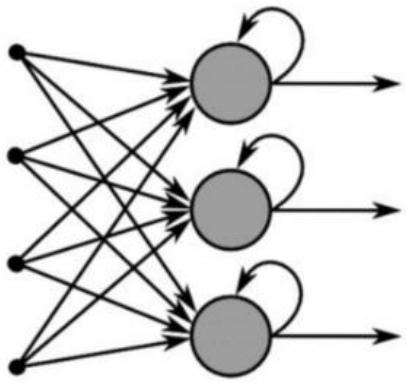
Obr. 1.5 – Princip dopředné neuronové sítě (Quiza, 2011)

Dopředná neuronová síť je zvláště vhodná pro problémy s rozpoznáváním vzorů a klasifikací, kde jsou data strukturovaná a dobře definovaná. Některé aplikace dopředných

neuronových sítí zahrnují rozpoznávání obrazu, rozpoznávání řeči a prediktivní modelování ve financích a marketingu. Jejich účinnost je však závislá na kvalitě a množství trénovacích dat a také na návrhu a konfiguraci konkrétní síťové architektury. (Quiza, 2011)

1.6 Rekurentní neuronová síť

Rekurentní sítě mají jedinečnou vlastnost oproti dopředným sítím, a to zpětnovazební charakter. Umožňuje tedy použít výstup z předchozího neuronu nebo celé vrstvy do dalšího kroku. Díky této vlastnosti je ale těžké někdy definovat vstupní a výstupní vrstvy. Tento mechanismus zpětné vazby umožňuje RNN zachytit časové závislosti a vztahy mezi vstupy v průběhu času. RNN jsou široce používány při zpracování přirozeného jazyka, rozpoznávání řeči, rozpoznávání rukopisu a skládání hudby. Jednou z oblíbených aplikací RNN je jazykové modelování, kde je síť trénována tak, aby předpovídala další slovo ve větě na základě předchozích slov. (Donges, 2022)



Obr. 1.6 – Princip rekurentní neuronové sítě (Donges, 2022)

2 UČENÍ NEURONONÉ SÍTĚ

Jedním z hlavních důvodů, proč využíváme umělé neutronové sítě je jejich vlastnost učení. Při učení dochází k vhodným modifikacím vazeb neuronů. Proces učení zahrnuje úpravu vah a vychýlení uzlů sítě tak, aby síť produkovala požadovaný výstup pro daný vstup. Toto je nejčastější proces, nicméně je možné se také setkat s nastavením aktivační funkce nebo postupnou změnou struktury neuronové sítě. Proces učení se také občas označuje jako algoritmus učení. (Doležel, 2016)

Přístup k učení dělíme na systematický a analytický. Systematické učení neboli také induktivní metoda učení, je přístup na základě pozorování množiny jevů, pro které platí obecně platné závěry. Analytický přístup se oproti tomu soustředí pouze na jeden jev a ten analyzuje, někdy je analytický přístup nazýván jako deduktivní metoda. (Doležel, 2016)

Nejčastěji se používá učení systematické, které se nějakým způsobem spoléhá na lidský zásah. Pokud požadovaný výsledek učení, tedy výstup ze sítě hodnotí člověk, jedná se o učení s učitelem. V opačném případě se jedná o učení bez učitele, kde výsledek neurčuje uživatel, ale sama neuronová síť si ji sama určuje ze svého výstupu na základě zpětné vazby. (Doležel, 2016)

To, jak jsou vazby a váhy mezi neurony počítány charakterizuje Hebbův zákon učení, který bude popsán v následující podkapitole. Dále bude přiblíženo učení s učitelem a bez učitele.

2.1 Hebbův zákon učení

Hebbův postulát poprvé navrhl kanadský psycholog Donald Hebb v roce 1949 a stal se principem neurovědy. Zákon říká, že „buňky, které spolu střílejí, spojují dohromady“. Jinými slovy, když jsou aktivovány dva neurony současně, spojení mezi nimi se posílí, takže je pravděpodobnější, že v budoucnu vystřelí společně. Naopak, pokud jsou dva neurony spolu aktivovány jen zřídka, jejich spojení časem slábne. Tento proces je známý jako synaptická plasticita a má se za to, že je základem učení a paměti v mozku. (Novák, 1993)

Hebbovo pravidlo tedy vychází z popisu biologických neuronových sítí, Nicméně je možné toto aplikovat i u umělých sítí, zpravidla pak u sítí typu jednoduchý perceptron a Hopfieldova síť. (Doležel, 2016)

2.2 Učení s učitelem

Učení s učiteli je typ přístupu strojového učení, ve kterém učitel poskytuje zpětnou vazbu a vedení systému učení. Tento přístup se často používá při učení, kde je výukový systém (neuronová síť) trénován pomocí předem označených dat. Učitel poskytuje správný výstup pro každý vstup a systém učení upravuje své vnitřní parametry tak, aby minimalizoval rozdíl mezi předpokládaným výstupem a správným výstupem. Neurony jsou organizovány do vrstev a spojení mezi neurony mají váhy, které určují sílu přenosu signálu. (Mařík, 1993; Doležel, 2016)

Při učení s učitelem lze trénovat pomocí algoritmu zpětného šíření, což je typ učení s přístupem uživatele. Algoritmus zpětného šíření zahrnuje výpočet chyby mezi předpokládaným výstupem a správným výstupem. A také úpravu vah spojení mezi neurony, aby se chyba minimalizovala. (Mařík, 1993; Doležel, 2016)

Je tedy nutné již na začátku mít rozřazená data s nějakými identifikátory a pro ně chtěný výstup. Porovnávané výstupní hodnoty se během učení porovnávají s výsledkem učení neuronové sítě y_j. Při této iteraci dochází k přepočtení váhy spojení viz rovnice 1, aby došlo k co největší shodě s chtěným výsledkem. Tedy hledá se minimum chybové funkce. (Doležel, 2016)

$$w_{ij} = w_{ij} + \Delta w_{ij} \tag{3}$$

Kde

w_{ij} … váha spojení mezi uzly i a j

Δw_{ij} … změna váhy vypočtená algoritmem učení.

Příklad chybové funkce může být dát vztahem 2. (Doležel, 2016)

$$E(s) = \sum_{k=1}^{M} \sum_{j=1}^{Q} [\, t_j(k) - y_j(s,k)]^2 \tag{4}$$

Kde

s … je pořadové číslo epochy;

M … počet vzoru trénovací množiny;

R … počet vstupů do umělé neuronové sítě;

Q … $počet\ vstup$ů $z\ neuronov$é $sít$ě;

k … je pořadové číslo vzoru trénovací množiny;

t_j … váha spojení mezi uzly i a j;

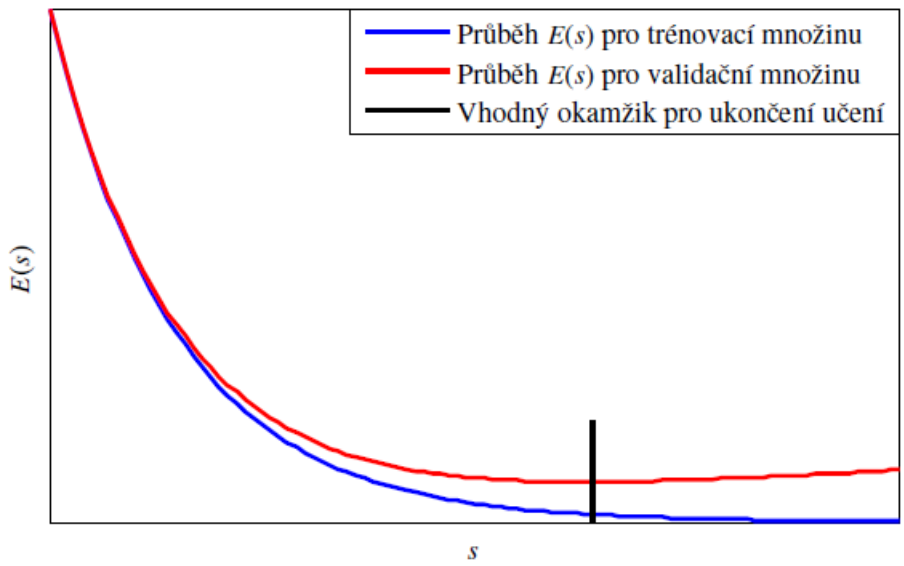
y_j … skutečné hodnoty na výstupu z neuronové sítě.

Často také data získaná od uživatele dělíme na trénovací a testovací, případně ještě validační množiny. Pro určení vah a prahů bývají použita data z trénovací množiny.

2.2.1 Trénovací množina

Pokud je v trénovací množině *M* dvojic vstup-cíl, pak jedno opakování učícího algoritmu je zpravidla provedeno *M* krát v jedné epoše trénování. Po provedení výpočtů jedné epochy se testuje podmínka ukončení algoritmu. Podmínka ukončení může být různá, ale odpovídá jednomu ze těchto možností. (Doležel, 2016)

a) Nejčastěji dojde k ukončení po překročení maximální hodnoty předem stanoveného počtu epoch.
b) Učení může být ukončeno, pokud hodnota chybové funkce pro trénovací množinu ve dvou po sobě jdoucích epochách je menší než nastavená mez.
c) Je velmi časté, že hodnoty vah a prahů nejsou ideální až na konci vypočtení poslední iterace epochy, ale někde v průběhu trénování. Příklad je na obrázku 2.1.



Obr. 2.1 – Příklad průběhu chybové funkce (Doležel, 2016)

2.2.2 Testovací množina

Testovací sada je samostatná datová sada, která se během tréninkového procesu nepoužívá. Jakmile je neuronová síť natrénována pomocí trénovacích a validačních sad, je testována na testovací sadě, aby se vyhodnotil její výkon na nových, neviditelných nebo ještě neznámých datech. Výkon na testovací sadě je dobrým ukazatelem toho, jak dobře bude neuronová síť fungovat v reálném světě. Tento výsledek lze použít k porovnání výkonu různých architektur neuronových sítí. (Zelinka, 2005)

2.2.3 Validační množina

Validační sada, známá také jako ověřovací sada, je podmnožinou trénovacích dat. Používá se k vyhodnocení výkonu neuronové sítě během trénovacího procesu. Neuronová síť je trénována pomocí trénovací sady a po každé iteraci trénování je výkon sítě vyhodnocen na validační sadě. Ověřovací sada pomáhá předcházet nadměrnému přizpůsobení, což je případ, kdy neuronová síť funguje dobře na trénovací sadě, ale špatně na nových, neviditelných datech. Vyhodnocením výkonu sítě na samostatné ověřovací sadě lze upravit tréninkový proces tak, aby se zlepšil výkon sítě na obecných datech. (Zelinka, 2005)

Souhrnně lze říci, že během tréninkového procesu se používá ověřovací sada, aby se předešlo nadměrnému přizpůsobení a zlepšil se výkon zobecnění neuronové sítě, zatímco testovací sada se používá k vyhodnocení výkonu neuronové sítě na nových, neviditelných datech a porovnání výkonu různých modelů.

2.2.4 Online a off-line učení s učitelem

Učení s učitelem se dá dále rozdělit podle toho, jak bude probíhat učení. Může být buď spojité nebo vsádkové. Při spojitém, označovaném také online učení probíhá určování vah po každém vstupu hodnoty do neuronové sítě. U off-line učení, tzv. vsádkovém se přepočet vah kumuluje do zvláštní proměnné a ke změně dojde až poté, co neuronová síť projde celou trénovací množinu. (Doležel, 2016)

2.2.5 Postup učení s učitelem

Postup algoritmu učení s učitelem lze jednoduše popsat ve třech hlavních bodech.

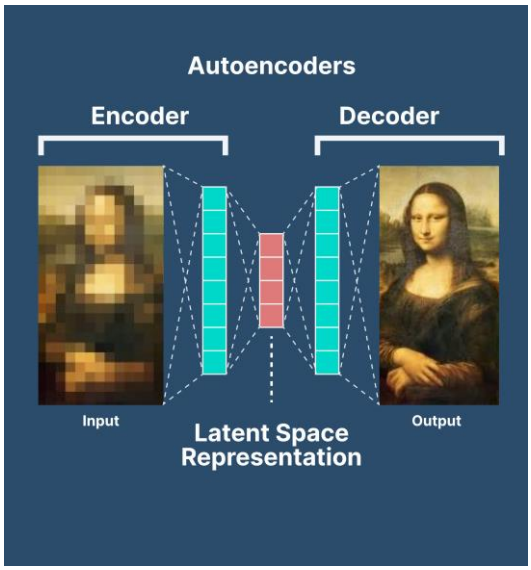
1. Stanov (změř, vypočti) trénovací, testovací a validační množinu.
2. Nastav podle nastavené počáteční (aktivační) funkce váhy a prahy sítě.
3. Při každé epoše trénování (s = 1, 2, 3, 4, …) proveď:
    a. Pro dvojici vstup – výstup
        i. Vypočti podle algoritmu hodnotu prahu a spojení sítě.
        ii. Pro online verzi, změň všechny prahy a váhy v síti.
        iii. U off-line verze, ulož přírůstek do dané proměnné.
    b. Pokud se jedná o off-line učení, přenastav pomocí daných proměnných všechny prahy a váhy v síti.
    c. Otestuj podmínky ukončení algoritmu.

Tímto způsobem je možné chápat principiální učení neuronové sítě pro učením s učitelem. (Doležel, 2016)

2.3 Učení bez učitele

Učení bez učitele je typ primárně strojového učení, ve kterém se neuronová síť učí vzorce a vztahy v datech, aniž by potřebovala označené příklady nebo typ toho co má hledat. Tento přístup se často používá při učení bez dozoru, kde jsou učícímu neuronovému systému poskytnuta neoznačená data a jeho úkolem je odhalit základní strukturu a vzory v datech. Často používanou metodou hledání je tzv. shlukování. Při shlukování jsou neurony v neuronové síti organizovány do shluků a každý shluk představuje skupinu podobných vstupů. Neuronová síť se učí seskupovat vstupy na základě jejich podobností, aniž by dostávala nějaké předchozí informace. (Bandyopadhyay, 2023; Doležel, 2016)

Další metoda pro učení bez učitele se nazývá autoencodér. V autokodéru je neuronová síť trénována k rekonstrukci vstupních dat z části komprimované reprezentace dat. Neuronová síť se učí kódovat důležité vlastnosti dat v komprimované reprezentaci a poté dekódovat komprimovaná data, aby rekonstruovala původní soubor dat. (Bandyopadhyay, 2023)



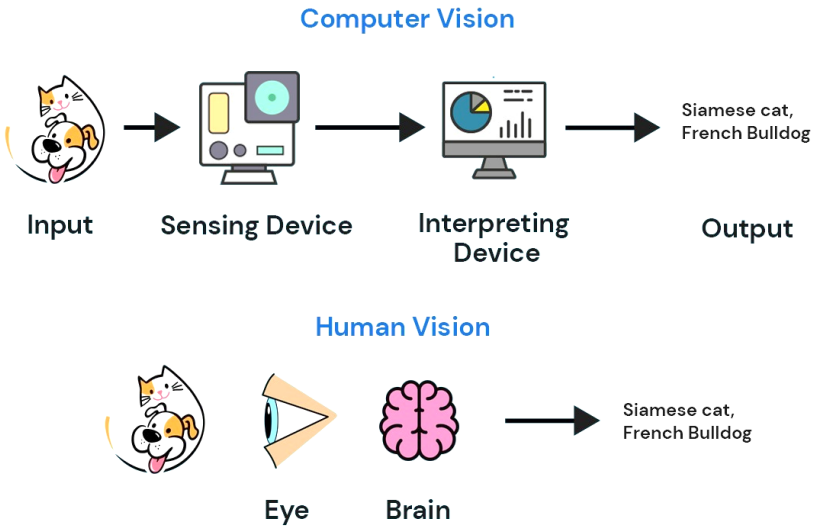
Obr. 2.2 – Autoenkodér (Bandyopadhyay, 2023)

Učení bez učitele je užitečné k objevování vzorců a vztahů ve složitých datech, aniž by bylo nutné předem data nějak třídit. Může však být náročné vyhodnotit výkon algoritmů učení bez dozoru, protože neexistuje žádná zpětná vazba, se kterou by bylo možné výsledky porovnat. (Bandyopadhyay, 2023)

3 POČÍTAČOVÉ VIDĚNÍ

Počítačové vidění je obor umělé inteligence a informatiky, který se zaměřuje na to, aby počítače mohly interpretovat a porozumět vizuálním informacím z okolního světa. Zahrnuje vývoj algoritmů a technik, které mohou počítačům umožnit analyzovat a porozumět nejen digitálním obrázkům a videím v reálném čase.

Počítačové vidění zahrnuje několik fází, včetně získávání obrazu, předběžného zpracování, extrakce rysů, rozpoznávání objektů a analýzy obrazu. Některé z klíčových aplikací počítačového vidění zahrnují rozpoznávání obrazu a videa, autonomní vozidla, rozpoznávání obličeje, lékařské zobrazování a robotiku. Jedná se o velmi silný nástroj, a proto nachází uplatnění v mnoha oborech i mimo elektrotechniku. Hlavní princip funkce je na obrázku 3.1 níže.



Obr. 3.1 – Počítačové vidění (Computer Vision, 2021)

Jednou z klíčových výzev v počítačovém vidění je vývoj algoritmů, které dokážou přesně a robustně interpretovat vizuální informace, i když jsou obrázky nebo videa zašuměné, neúplné nebo obsahují jiné typy rušení. Výzkumníci v této oblasti neustále vyvíjejí nové techniky a přístupy ke zlepšení výkonu a přesnosti systémů počítačového vidění. (Jirkovský, 2018; Computer Vision, 2021)

Mezi základní úlohy počítačového vidění patří klasifikace nebo klasifikace s lokací, dále detekce objektů, sémantická a instalační segmentace.

3.1 Klasifikace

Klasifikace objektů je nejjednodušší část strojového vidění, ve které dochází hlavně k zařazení obrázku do určité třídy podle rozpoznaného objektu. Výstupem tedy pro jednoduchou klasifikaci je třída. Pro klasifikaci s lokalizací je výstupem nejen třída, ale také vektor umístění objektu v daném obrázku. Existují různé přístupy ke klasifikaci objektů v počítačovém vidění, včetně tradičních metod strojového učení nebo technik hlubokého učení. (Rais, 2022; Gallagher, 2023)

Některé běžné metody klasifikace objektů zahrnují:

1. Shoda šablon *(Template matching)*: Tato metoda zahrnuje porovnání snímků se sadou předdefinovaných šablon k identifikaci objektu.
2. Klasifikace založená na prvcích *(Feature-based classification)*: Tato metoda zahrnuje extrahování prvků z obrázku, jako jsou hrany, rohy nebo textury, a použití těchto prvků k trénování modelu strojového učení pro klasifikaci objektů.
3. Klasifikace založená na hlubokém učení: Tento přístup zahrnuje použití konvolučních neuronových sítí (CNN) k učení funkcí přímo z obrázků nebo videa a následné použití těchto funkcí ke klasifikaci objektů. Přístupy založené na hlubokém učení se staly nejmodernějším pro klasifikaci objektů díky jejich schopnosti učit se složité vlastnosti a vzorce v datech. (Kubínek, 2009)

Obr. 3.2 – Klasifikace obrázku (Gallagher, 2023)

3.2 Detekce

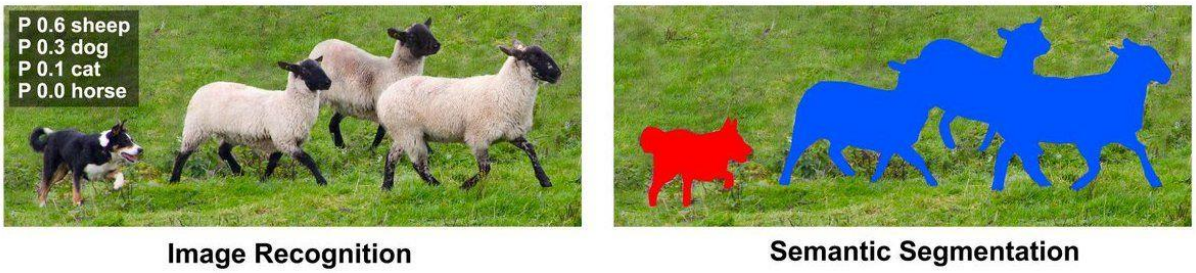
Detekce objektů zahrnuje identifikaci a lokalizaci objektů ve snímku nebo ve videu. Na rozdíl od klasifikace objektů detekce objektů nejen identifikuje kategorii objektu, ale také lokalizuje objekt v rámci snímku nebo videa. Hlavním rozdílem oproti jednoduché klasifikaci je fakt, že na obrázku se může vyskytovat více objektů stejných nebo různých tříd. U detekce

fungují podobné metody jako u klasifikace. Navíc je zde poměrně častý princip tzv. hybridního přístupu kdy se kombinuje tradiční přístup se systémem konvolučních sítí. (Rais, 2022; Kubínek, 2009)

3.3 Segmentace

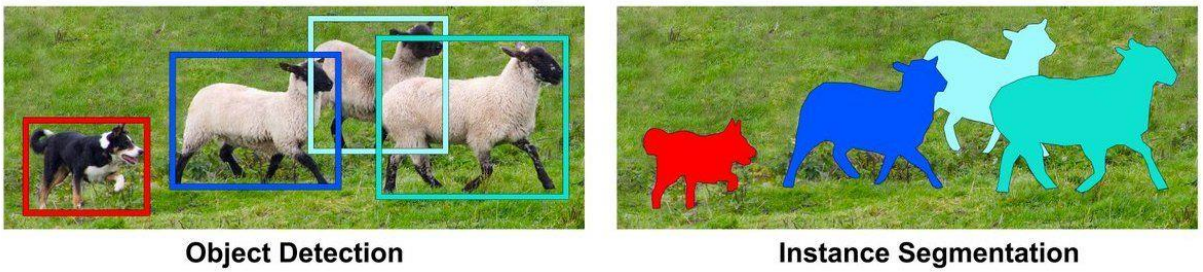
Sémantická segmentace rozděluje obraz na sémanticky smysluplné části nebo oblasti. To tedy znamená, že každý pixel obrázku zařadí do příslušné třídy, ke které patří. Toto označení umožňuje jemnější analýzu obrazu.

Rozdíl oproti klasifikaci obrazu je ten, že u sémantické segmentace přiřazujeme pouze pixely, a ne celé objekty.



Obr. 3.3 – Sémantická segmentace (Walia, 2022)

Instalační segmentace se liší tím, že oproti sémantické segmentaci, jde více do hloubky obrazu. Tedy u sémantické segmentace nemůžeme rozlišit mezi různými objekty stejné kategorie, například všechny ovce jsou ovce. Instalační segmentace může rozlišovat mezi objekty stejných kategorií, tj. každá ovce má vlastní barvu.

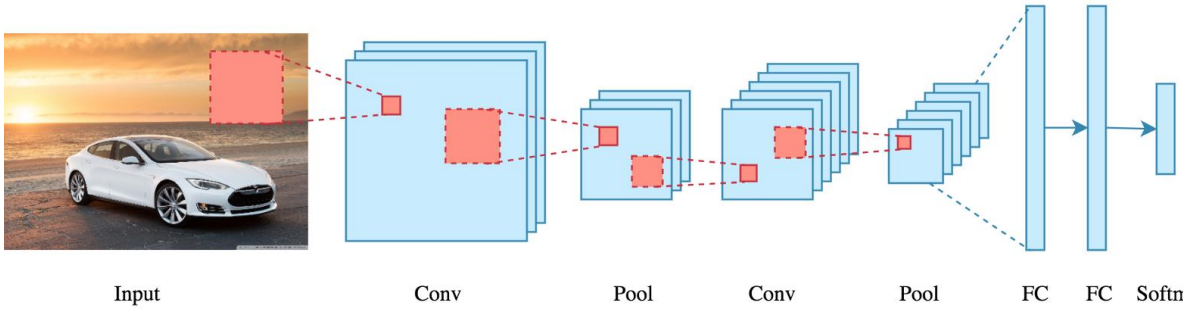


Obr. 3.4 – Instalační segmentace (Walia, 2022)

4 KONVOLUČNÍ NEURONOVÉ SÍTĚ (CCN)

Konvoluční neuronová síť (*CNN*) je typ hluboké neuronové sítě, která se běžně používá pro rozpoznávání, klasifikaci a zpracování obrazu. Je speciálně navržena tak, aby rozpoznávala a analyzovala obrázky. Zpracování probíhá prostřednictvím více vrstev se specializovanými filtry, které identifikují a extrahují vlastnosti obrázku.

Základním stavebním kamenem CNN je konvoluční vrstva, která na vstupní obrázek aplikuje řadu filtrů nebo jader, aby vytvořila mapu prvků. Tyto filtry se posouvají po vstupním obrázku a provádějí matematickou operaci známou jako konvoluce, která z obrázku extrahuje informace, jako jsou hrany, tvary a vzory. (Potrimba, 2023)



Obr. 4.1 – Obecná architektura konvoluční neuronové sítě (Potrimba, 2023)

Každý výstup z konvoluční vrstvy prochází typicky nelineární aktivační funkcí, aby se zavedla nelinearita a zlepšila se schopnost modelu učit se složité funkce.

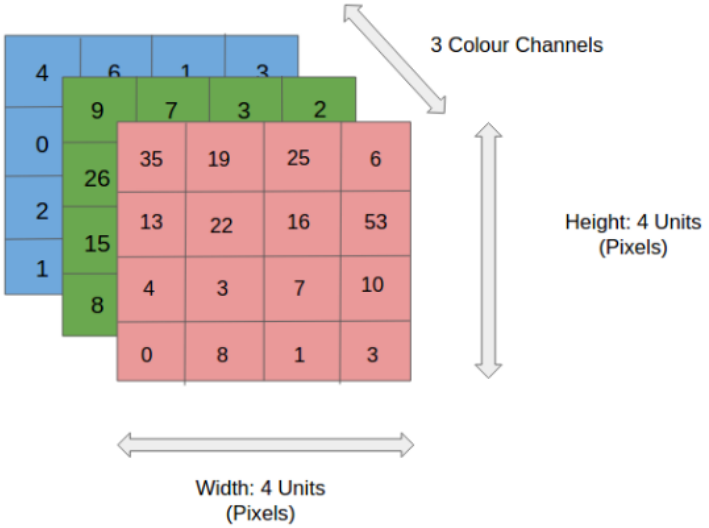
Další vrstvy v CNN mohou zahrnovat sdružovací vrstvy, které pře vzorkují mapu prvků tím, že v každém okně vezmou maximální nebo průměrnou hodnotu. Plně propojené vrstvy, které kombinují prvky extrahované z předchozích vrstev, a hodnota maximální nebo průměrná vytvářejí konečnou předpověď.

Ideálně nastavená konvoluční síť je taková, že kvalitně pořízený obraz převede na reprezentaci, které neuronová síť dokáže porozumět, a přitom zachovala důležité funkce, které vedou k přesné předpovědi.

CNN prokázaly působivý výkon v mnoha úlohách počítačového vidění, včetně rozpoznávání objektů, klasifikace obrazu, segmentace a detekce. Byly také použity v jiných oblastech, jako je zpracování přirozeného jazyka, rozpoznávání řeči a objevování léků. (Potrimba, 2023)

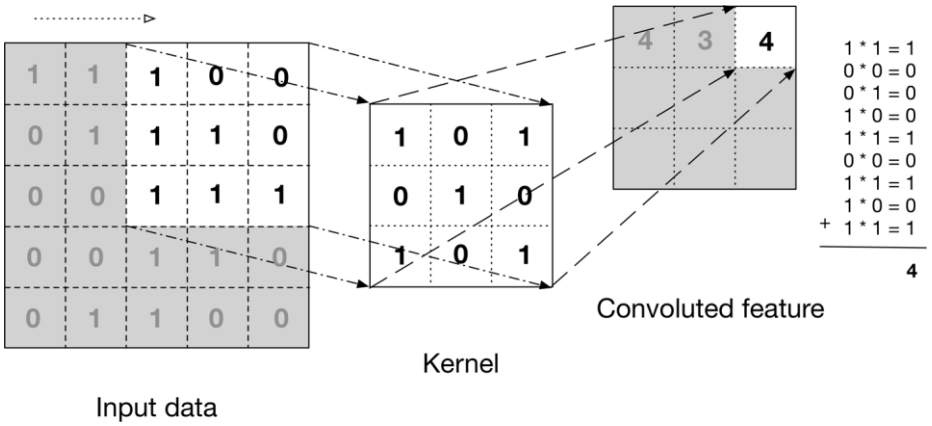
4.1 Princip funkce

Nejprve je důležité, jak je reprezentován obrázek. Obrázek je matice hodnot pixelů, přičemž každý pixel má specifickou hodnotu barvy. Obrazy RGB, které jsou nejběžnějším používaným typem. Mají tři roviny představující červený, zelený a modrý barevný kanál. Další možností může být například obrázek ve stupních šedi, ale zde máme jen jednu hodnotovou rovinu.



Obr. 4.2 – Reprezentace RGB obrazu (Potrimba, 2023)

Počet pixelů v obrázku určuje rozlišení, čím větší číslo tím větší rozlišení, a i podrobnější informace. Pro fungování konvolučních sítí je nezbytná funkce konvoluce. Zjednodušený postup pro obrázek ve stupních šedi na obrázku 4.3.



Obr. 4.3 – Výpočet kroku konvoluce

Při konvoluci je pro vstupní obraz aplikována malá matice, označována jako filtr nebo jádro, která má za funkci extrahovat důležité vlastnosti. Filtr se posouvá po obrázku a pro každou pozici se vynásobí odpovídající hodnoty a výsledek se sečte. Tomuto výsledku se říká

„konvolovaný prvek“, který může být předán další vrstvě s novým filtrem. Tento proces se může i několikrát opakovat.

První takto získanou vrstvou je *ConvNet* vrstva. ConvNet má funkci identifikovat základní prvky jako hrany, rohy a další. Každou další vrstvou je neuronová síť schopna detekovat složitější objekty.

Poslední vrstva konvoluční sítě vytváří „aktivační mapu“. Pokud se jedná o problém klasifikace je tako vrstva použita jako mapa k určení pravděpodobnosti, že vstupní obraz patří do určité třídy. Hodnota pravděpodobnosti je nabývá hodnot od 0 do 1 podle toho, jak podrobná a kvalitní data byla předána k učení. Například ConvNet je navržena pro rozpoznání ovce, koně a psa. Výstupem poslední vrstvy bude pravděpodobnost odpovídající možnému výskytu těchto zvířat. (Potrimba, 2023)

4.2 Výstup CCN pro detekci objektů

Detekce zahrnuje zároveň identifikaci a lokalizaci objektů v obraze. Výstupem by tedy měla být sada ohraničujících rámečků kolem objektů právě v obraze, který je zpracováván. Nejlépe spolu s popisky tříd a skóre spolehlivosti, které udávají, co je každý objekt zač a jaká je jistota tohoto obrazu k dané třídě. Nejběžněji je používán přístup dvoufázového rámce, jako jsou modely R-CNN *(Regions with Convolutional Neural Networks)*. V první fázi se používá síť k navrhování tzv. kandidátských oblastí objektů na obrázku. Tyto návrhy jsou pak ve druhé fázi použity pro plně propojenou síť. Tato síť blíže upřesní klasifikaci a ohraničení rámečkem.

Nejnovější přístupy, mezi které patří YOLO (*You Only Look Once)* a RetinaNet, používají pouze jednostupňový přístup. Zde síť předpovídá ohraničující rámečky, označení tříd a skóre spolehlivosti přímo ze vstupního obrazu. To má za následek vyšší rychlost, ale často o něco menší přesnost. Samozřejmě také velmi záleží na druhu a tvaru rozpoznávaného obrazu nebo na homogenitě prostředí. (Potrimba, 2023)

4.3 Vrstvy v konvolučních neuronových sítích

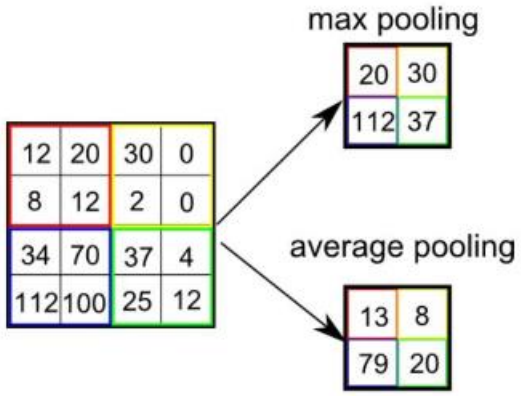
V této kapitole je popsán přehled a základní rozdělení vrstev, které se dále běžně používají v architektuře konvolučních sítí. Název vrstvy je častěji známý v anglickém jazyce, a proto jsem tyto názvy přiložil za název vrstvy.

4.3.1 Sdružovací vrstva *(Pooling Layer)*

Pooling vrstva je v principu velmi podobná konvoluční vrstvě, snižuje prostorovou velikost konvolučních prvků v konvoluční neuronové síti. Snižování je prováděno tzv. *downsamplingem* a to má dvojí účel. Nejprve se sníží výpočetní výkon, který je zapotřebí ke zpracování dat, a také zvyšuje úrovně abstrakce ve funkcích. V konvolučních sítích se používají dva typy sdružování, a to průměrné sdružování nebo maximální sdružování.

Maximální sdružování využívá *downsampling* tak, že vybere maximální hodnotu pixelu z části obrazu vybrané filtrem nebo jádrem. Tímto se sníží potřebný výpočetní výkon, ale také se odstraňuje šum a sníží rozměry.

Průměrné sdružování funguje naopak na principu průměrování všech hodnot vybraných jádrem. Také při průměrném sdružování dojde ke snížení rozměrů, nicméně postrádá schopnost potlačení šumu. Výsledkem je, že v praxi častěji používáné maximální sdružování je efektivnější a robustnější. Poskytuje tady lepší výkon oproti průměrnému sdružování. Samozřejmě i tady závisí na typu dat, které jsou síti předloženy. (Potrimba, 2023)



Obr. 4.4 – Výstupy výpočtů maximálního a průměrného sdružování (Potrimba, 2023)

4.3.2 Aktivační vrstva *(Activation Function Layer)*

Aktivační vrstva pomáhá konvoluční neuronové síti naučit se nelineární vztahy mezi vstupem a výstupem. Díky tomu umožňuje modelovat složité vzory a vztahy v datech bez předním známých vzorců. Tato vrstva je typicky aplikována na výstup každého neuronu v síti.

Nejčastěji používané aktivační funkce u konvolučních sítí jsou Rektifikovaná lineární jednotka (ReLU), Sigmoid nebo Hyperbolická tečna (tanh). Pro správné určení typu aktivační funkce je třeba znát typ propojení mezi vrstvami.

4.3.3 Dávková normalizační vrstva *(Batch Normalization Layer)*

Dávková normalizace se používá u neuronových sítí na vstupu každého neuronu tak, aby měl nulový průměr a jednotkový rozptyl. Pomáhá to ke stabilizaci procesu učení a také během tréninkového procesu, kde se často mění rozložení vstupů do vrstvy.

Je to tedy výkonná technika, která se široce používá v konvolučních sítích ke stabilizaci procesu učení a zlepšení výkonu modelu. Normalizuje vstupy do každého neuronu, snižuje vnitřní posun vzájemných vazeb a reguluje model, což vede k lepšímu výkonu zobecnění. (Potrimba, 2023)

4.3.4 Vrstva vyřazení *(Dropout Layer)*

Tato vrstva je hlavně proto aby se určitým způsobem regulovalo překrytí. Funguje to tak, že během každé iterace tréninku náhodně nastaví poměr vstupních jednotek na nulu, čímž tyto jednotky efektivně „vypustí“ a zabrání jim, aby se podílely na průchodu vpřed. Dělá se to hlavně proto, aby se síť naučila více nezávislých reprezentací vstupů. Díky tomu je pak odolnější vůči změnám ve vstupních datech.

Jednoduchý příklad je pro detekování obličeje. Obličej tvoří oči, ústa, nos atd. Nicméně síť mohu trénovat tak aby byl obličej detekován i v případě, že jsou například ústa zakryta. Je tedy výhodné nemít všechny funkce zahrnuty pro učení, ale náhodně učit podmnožiny těchto funkcí. (Potrimba, 2023)

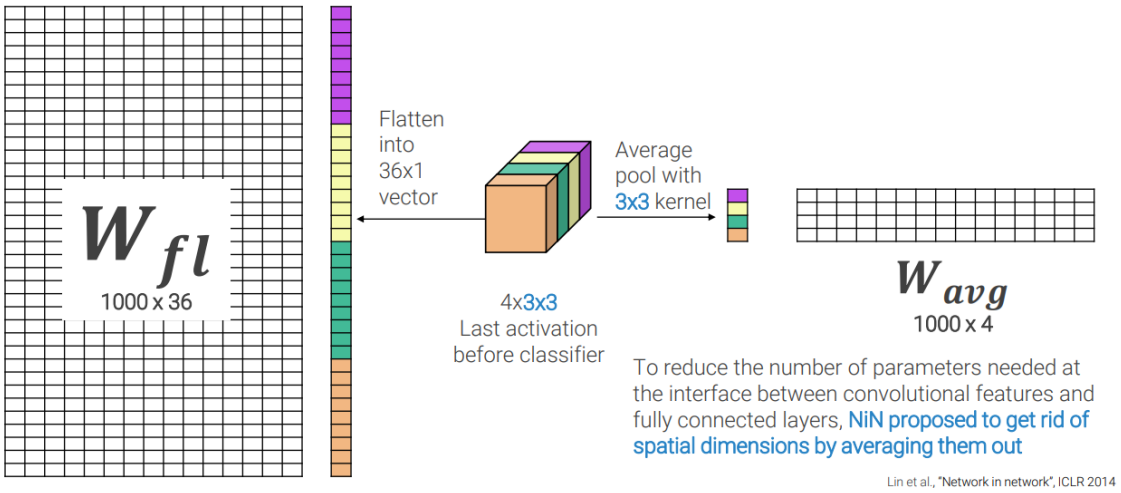
4.3.5 Klasifikační vrstva *(Classification Layer)*

V neuronové síti je klasifikační vrstva poslední vrstvou, která produkuje výstup modelu. U konvolučních sítí se používají dva typy, a to plně propojené vrstvy a globální průměrné sdružovací vrstvy.

Plně propojená vrstva spojuje všechny neurony v přechozí vrstvě se všemi v aktuální vrstvě. Výstupem je pak výpočet z aplikací matice váhy a vektoru zkreslení na vstupu, po které následuje aktivační funkce. Dále je plně propojená vrstva připojena k aktivaci *softmax* za účelem vytvoření skóre třídy. Výhodou je že síť se může naučit složité rozhodovací hranice mezi třídami. Na druhou stranu toto může vést k velké složitosti a je potřeba velké množství parametrů.

Globální průměrná sdružovací vrstva je uzpůsobena právě pro řešení těchto problémů s parametry. Zmenšuje prostorové rozměry map prvků tek, že vezme průměrnou hodnotu každé mapy prvků. Výstupem je vektor o velikosti počtu map prvků. Dále je podobné zpracování jako u plně propojené vrstvy. Výhodou je, že jsou tyto vrstvy méně náročné a potřebují méně parametrů. Nicméně neumožňují učení složitých rozhodovacích funkcí. (Potrimba, 2023)

Princip je na obrázku níže, kde vlevo je plně propojená síť a vpravo globální průměrná sdružovací síť.



Obr. 4.5 – Plně propojená vrstva a globální průměrná sdružovací vrstva (Potrimba, 2023)

5 ALGORITMY PRO KLASIFIKACI OBJEKTŮ

V této kapitole si představíme nejznámější a nejvíce rozšířené algoritmy pro klasifikaci objektů. Algoritmy jsou založeny na principu konvolučních neuronových sítí. Existuje několik typů konvolučních sítí, které lze použít pro klasifikaci objektů, z nichž každý má svou vlastní architekturu a detaily implementace.

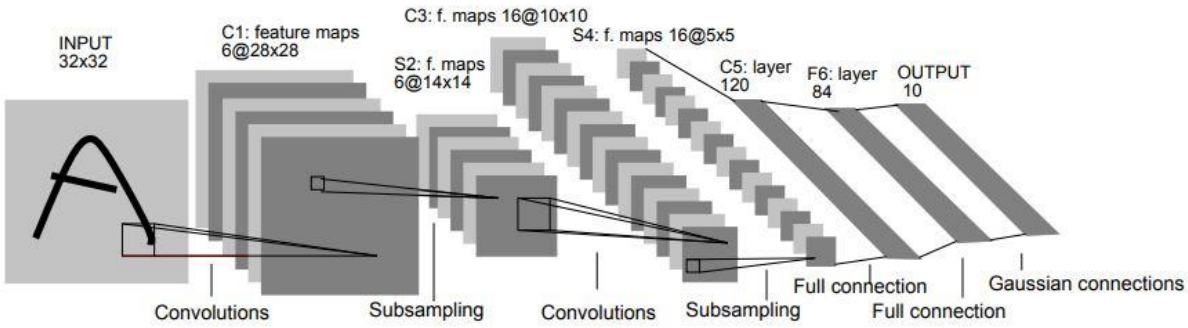
Mezi metriky hodnocení přesnosti konvolučních sítí patří hlavně přesnost klasifikace. Výstupem klasifikace je seznam tříd a pravděpodobnost, do které obrázek spadá. Nejčastěji se udává jako tzv. *top-1* nebo *top-5* ve vyhodnocovací datové sadě. Pro *top-1* je musí shodovat skutečná a nejpravděpodobnější předpovězená třída, u *top-5* stačí pokud je předpovězená třída mezi pěti nejpravděpodobnějšími variantami. (Rais, 2022)

Dalším důležitým parametrem je velikost sítě. Nejvíce se udává v reprezentaci počtu parametrů, které je možné v rámci sítě trénovat. Může se také udávat jako počet operací s plovoucí řádkovou čárkou, označované jako *FLOPs*. Vztahuje se k počtu operací s plovoucí desetinnou čárkou, jako jsou sčítání, odečítání, násobení a dělení, které jsou nutné k provedení dopředného průchodu modelu na daném vstupu. FLOP se často používají k odhadu výpočetních požadavků a rychlosti modelu a k porovnání účinnosti různých modelů. Rychlost tedy je také velmi důležitá pro charakterizaci sítě a udává se nejčastěji v *ms,* za kterou je provedena klasifikace obrázku.

Dále jsou uvedeny nejběžnější typy konvolučních neuronových sítí pro klasifikaci.

5.1 LeNet

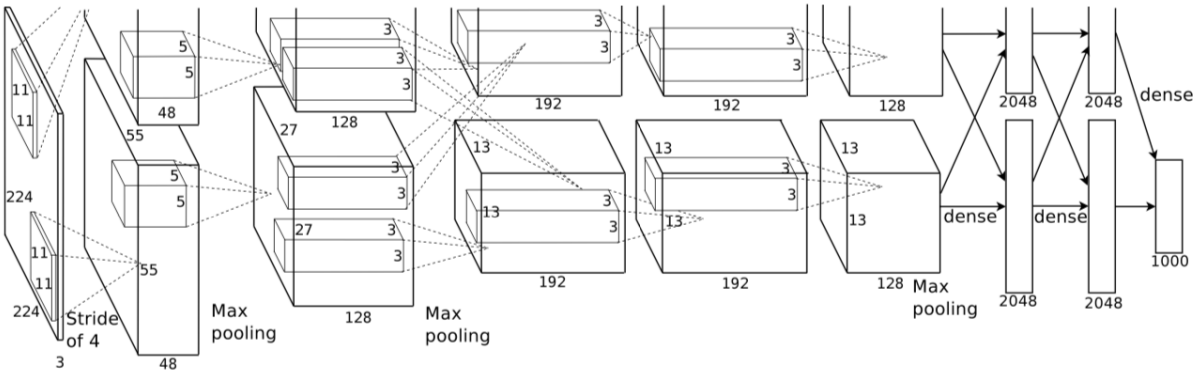
LeNet je jednoduchá architektura CNN, kterou představil Yann LeCun a kol. v roce 1998. Skládá se ze dvou konvolučních vrstev následovaných sdružovacími vrstvami, které se používají pro zpracování vstupního obrazu. Dále jsou poslední dvě plně propojené vrstvy pro vytvoření klasifikačního výstupu. LeNet byl původně navržen pro rozpoznávání ručně psaných číslic, ale od té doby se používá pro jiné úkoly klasifikace obrázků. Jedná se většinou jako výchozí bod pro klasifikaci. (Potrimba, 2023)



Obr. 5.1 – Architektura LeNet (Potrimba, 2023)

5.2 AlexNet

AlexNet je hluboká architektura CNN, kterou představil Alex Krizhevsky et al. v roce 2012. Skládá se z pěti konvolučních vrstev, následovaných sdružovacími (pooling) vrstvami a třemi plně propojenými vrstvami. AlexNet byla první CNN, která vyhrála *ImageNet Large Scale Visual Recognition Challenge (ILSVRC)*, což prokázalo sílu hlubokého učení v úkolech klasifikace obrázků. Jedním z hlavních změn byla zvolena funkce *ReLU*, u které byly dosaženy mnohem lepší výsledky než u *tanh*. (Potrimba, 2023)

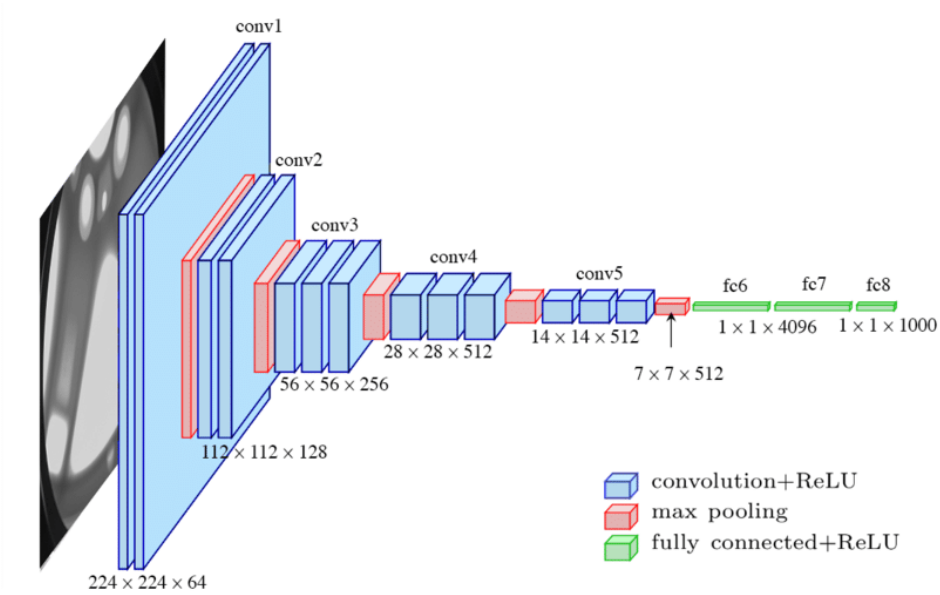


Obr. 5.2 – Architektura AlexNet

5.3 VGGNet

VGGNet je architektura CNN, kterou představili Karen Simonyan a Andrew Zisserman v roce 2014. Skládá se ze série konvolučních vrstev s malými filtry 3x3, po nichž následují pooling vrstvy a poslední plně propojená vrstva. Vyznačuje se velmi hlubokou architekturou až

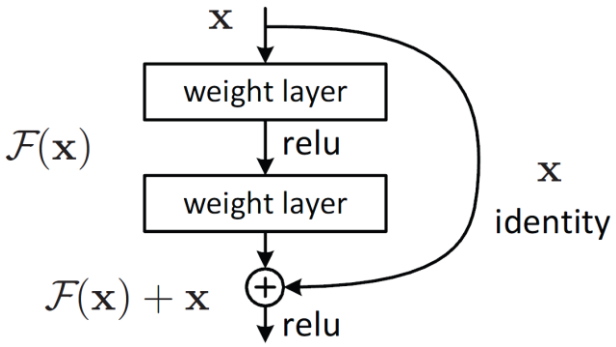
devatenácti vrstev. VGGNet byl navržen tak, aby se z obrázků naučil složitější funkce. Používá se tedy v upravené formě na detekci a segmentaci obrázků. (Potrimba, 2023)



Obr. 5.3 – Architektura VGG (Potrimba, 2023)

5.4 ResNet

ResNet je architektura CNN, kterou představil Kaiming He et al. v roce 2015. Skládá se ze „zbytkových bloků“, které umožňují síti učit se a skládat velmi hluboké vrstvy, až stovky vrstev, bez snížení výkonu. V principu zbytkové bloky obcházejí jednu nebo více vrstev a umožňují přechod přímo do dřívějších vrstev. ResNet byl navržen tak, aby řešil problém mizejícího gradientu, který může nastat ve velmi hlubokých sítích.

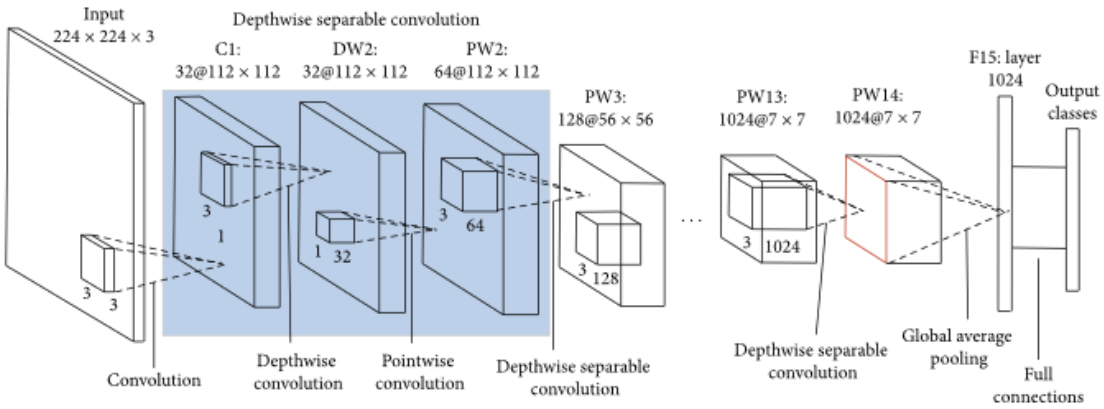


Obr. 5.4 – Princip architektura ResNet (Potrimba, 2023)

5.5 MobileNet

Byl vyvinutý společností Google v roce 2017 speciálně pro použití na mobilních a vestavěných zařízeních s omezenými výpočetními zdroji. Architektura MobileNet je založena na hloubkově oddělitelných konvolucích, které rozkládají standardní konvoluci na dvě samostatné vrstvy: hloubkové konvoluce a bodové konvoluce. Hloubkové konvoluce aplikují samostatný konvoluční filtr na každou část vstupu, zatímco bodové konvoluce kombinují výstupy hloubkových konvolucí pomocí konvoluce 1x1. Tento přístup snižuje počet parametrů a FLOP požadovaných pro konvoluční vrstvy při zachování vyjadřovací schopnosti modelu.

MobileNet je široce používán v reálných mobilních a vestavěných aplikacích, jako je detekce objektů, rozpoznávání obličeje a segmentace obrazu. (Potrimba, 2023)



Obr. 5.5 – Architektura MobileNet (Potrimba, 2023)

6 ALGORITMY PRO DETEKCI OBJEKTŮ

V této kapitole jsou uvedeny nevýznamnější algoritmy pro detekci objektů. Algoritmy detekce objektů založené na neuronových sítích se obvykle skládají ze dvou hlavních fází: sítě pro návrh regionu *(RPN)* a klasifikační sítě. Často jsou tyto sítě podobné nebo dokonce vycházejí z architektur pro klasifikaci. (Gandhi, 2018)

Fáze RPN využívá konvoluční neuronovou síť ke generování sady návrhů objektů, které jsou kandidátskými oblastmi ve vstupním obrazu. RPN pracuje s mapou prvků vstupního obrázku a posouvá malé okno, které nazýváme kotva, přes mapu prvků v různých měřítcích a poměrech stran. Pro každou kotvu RPN předpovídá dvě hodnoty: pravděpodobnost, že kotva obsahuje objekt, a souřadnice ohraničujícího rámečku kolem objektu. RPN pak na základě těchto předpovědí vybere sadu návrhů objektů s vysokým skóre. (Gandhi, 2018)

Stupeň klasifikační sítě přebírá každý návrh objektu generovaný RPN a klasifikuje jej do jedné z několika předem definovaných tříd, jako je osoba, auto nebo pes. Klasifikační síť se obvykle skládá ze série konvolučních a plně propojených vrstev, které zpracovávají obraz v zájmové oblasti a poskytují pravděpodobnosti v různých třídách.

Nejdůležitější rozdíly hledáme ve vlastnostech přesnost a rychlost algoritmů. Ty se porovnávají na známých volně dostupných sadách jako Pascal VOC nebo MS COCO. Rychlost je měřena jako počet zpracovaných snímků za vteřinu, tedy *Frames Per Second (FPS)*. Pro měření přesnosti je potřeba znalosti více pojmů, které jsou vysvětleny dále. Výsledná přesnost je udávána ve formě *Avarange precision (AP)*, nebo její střední průměrná hodnota *Mean averange precision (mAP)*. (Rais, 2022)

6.1 Důležité pojmy pro detekci

Přesnost *(Precision)* je zlomek správně detekovaných objektů mezi všemi detekovanými objekty. Vyvolání *(Recall)* je zlomek správně detekovaných objektů mezi všemi skutečnými objekty. Hodnoty přesnosti a vyvolání jsou vypočítány pro různé prahové hodnoty spolehlivosti nebo skóre spolehlivosti přiřazené každému detekovanému objektu.

Pro výpočet AP se nejprve vypočítají hodnoty přesnosti a vyvolání pro každou třídu zvlášť. Křivka *Precisin-Recall* se pak sestrojí vynesením hodnot přesnosti proti hodnotám vybavování pro různé prahy spolehlivosti. AP se pak vypočítá jako plocha pod křivkou.

Pomocí *Intersection over Union (IoU) se* měří překrytí mezi základním pravdivým ohraničujícím rámečkem a předpokládaným hraničním rámečkem pro detekovaný objekt. IoU se běžně používá k definování prahové hodnoty pro přijetí detekovaného objektu jako skutečně pozitivního.

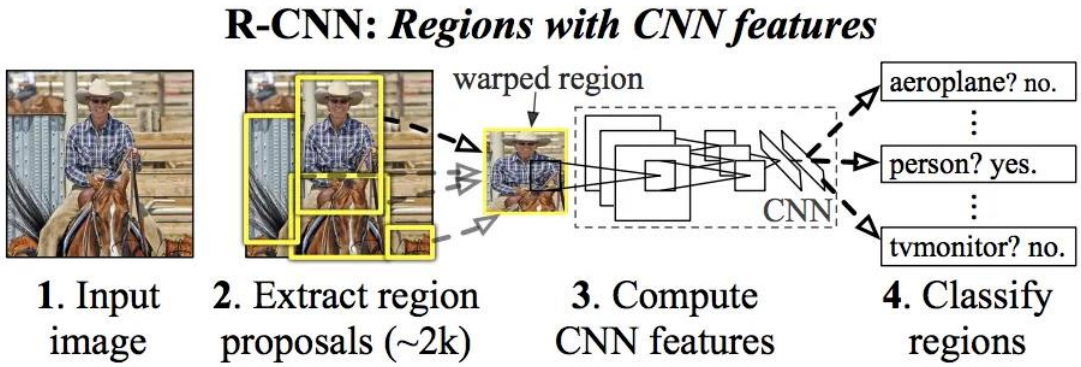
Pravdivá detekce (TP) je správná detekce. Tato detekce je vyhodnocena jako pravdivá, pokud skóre přesnosti je vetší než nastavená prahová hodnota a zároveň je vetší než IoU.

Falešně pozitivní frekvence (FPR) zobrazuje podíl nesprávných detekcí mezi všemi oblastmi, které nejsou předmětem. Nízká FPR je žádoucí pro algoritmy detekce objektů, aby se minimalizovaly falešné poplachy.

Falešně negativní frekvence (FNR) je zlomek ztracených objektů mezi všemi skutečně pravdivými objekty. Objekt tedy není detekován i když se ve skutečnosti vyskytuje. (Rais, 2022; Gandhi, 2018)

6.2 R-CNN

Region-based Convolutional Neural Network navrhl Ross Girshick v roce 2014 a byl jedním z prvních algoritmů využívajících neuronové sítě pro detekci objektů. R-CNN používá selektivní vyhledávání ke generování návrhů objektů, kterých je něco přes 2 000, a poté aplikuje CNN na každý návrh, aby extrahoval vlastnosti a klasifikoval objekt. (Gandhi, 2018)

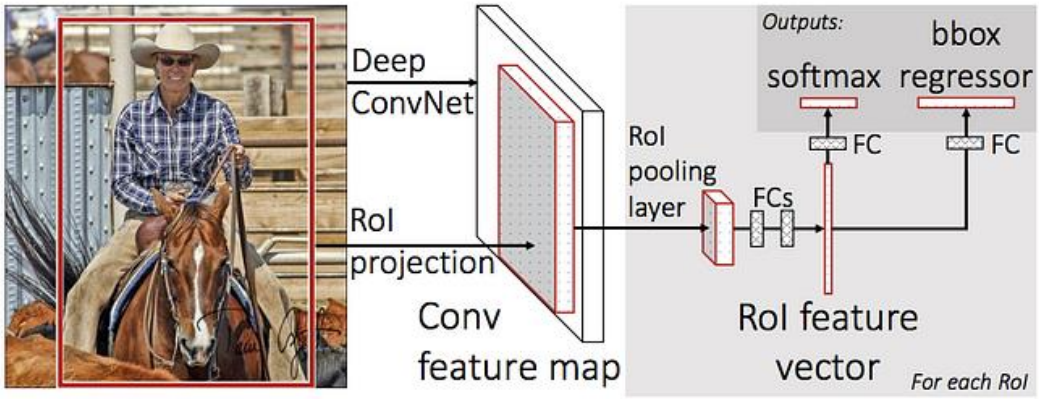


Obr. 6.1 – Princip funkce R-CNN (Gandhi, 2018)

Mezi největší problémy R-CNN patří obrovské množství času pro zpracování dat. Nelze jej tedy implementovat na aplikace v reálném čase.

6.3 Fast R-CNN

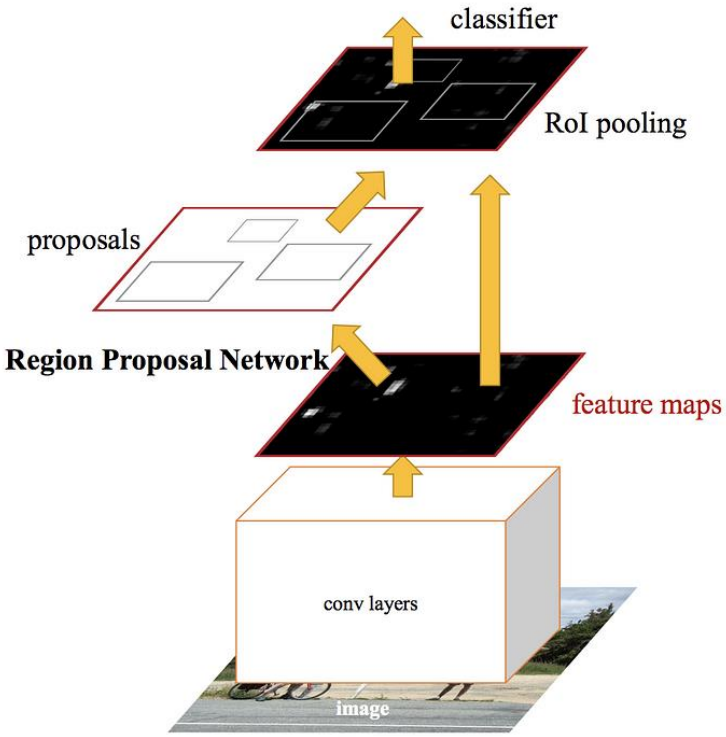
Ve Fast R-CNN je vstupní obraz předán konvoluční neuronovou sítí, aby se vytvořila mapa konvolučních prvků. Vrstva sdružování oblasti zájmu (RoI) je pak aplikována na mapu prvků, která se bere jako vstup navrhované oblasti zájmu a vydává mapy prvků s pevnou velikostí. Tyto mapy jsou pak dodávány do plně propojené sítě pro klasifikaci a regresi hraničního rámečku. Kombinací fází návrhu objektu a klasifikace do jediné sítě snižuje Fast R-CNN ve srovnání s R-CNN trénovací čas a požadavky na paměť. Rozdíl oproti R-CNN je tedy primárně v tom, že konvoluce neprobíhá pro zpracování každého snímku. (Gandhi, 2018)



Obr. 6.2 – Fast R-CNN (Gandhi, 2018)

6.4 Faster R-CNN

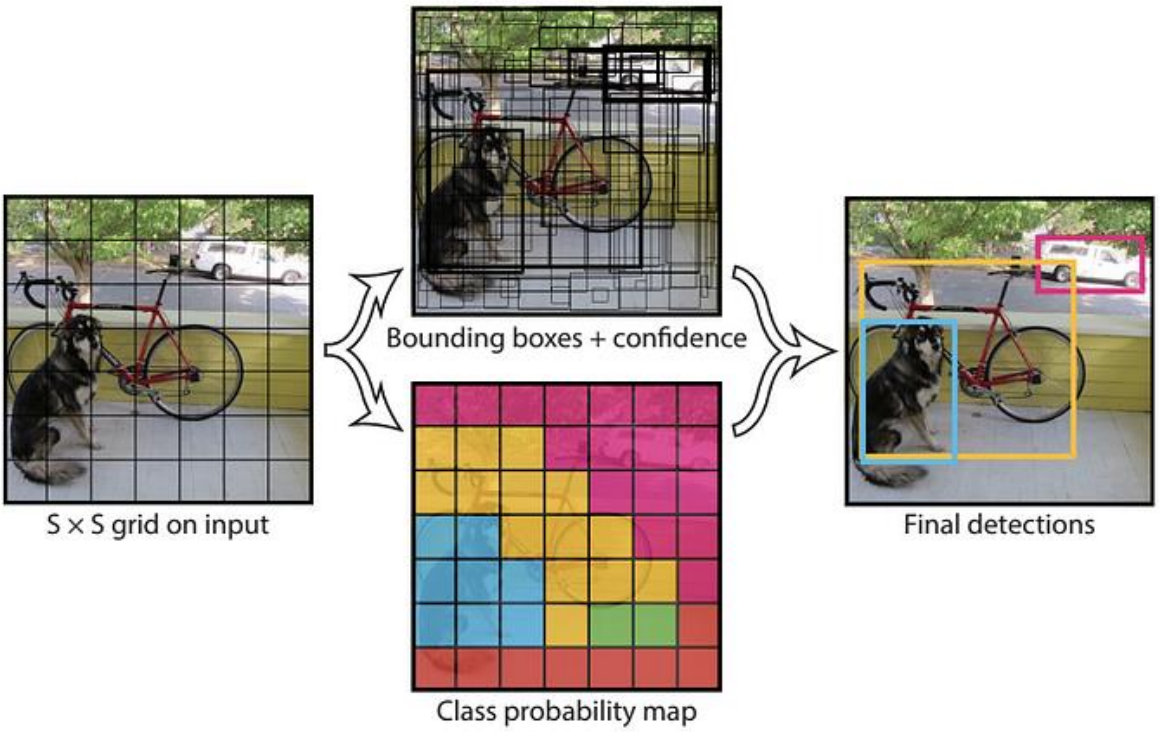
Rychlejší R-CNN navrhl Shaoqing Ren a kol. v roce 2015 a zlepšil architekturu R-CNN nahrazením selektivního vyhledávání fází RPN. To umožňuje algoritmu generovat návrhy objektů efektivněji a přesněji. Důležitou výhodou je tedy zvýšení rychlosti, protože konvoluční síti již není předávána pokaždé sada návrhů. Návrhy jsou poté zpracovány pooling vrstvou RIO, která dokončí klasifikaci obrazu a předpovídá ohraničující hodnoty pro rámečky. Tento systém je mnohem rychlejší než jeho předchůdce a může být použit i pro detekci v reálném čase. (Gandhi, 2018)



Obr. 6.3 – Faster R-CNN (Gandhi, 2018)

6.5 YOLO

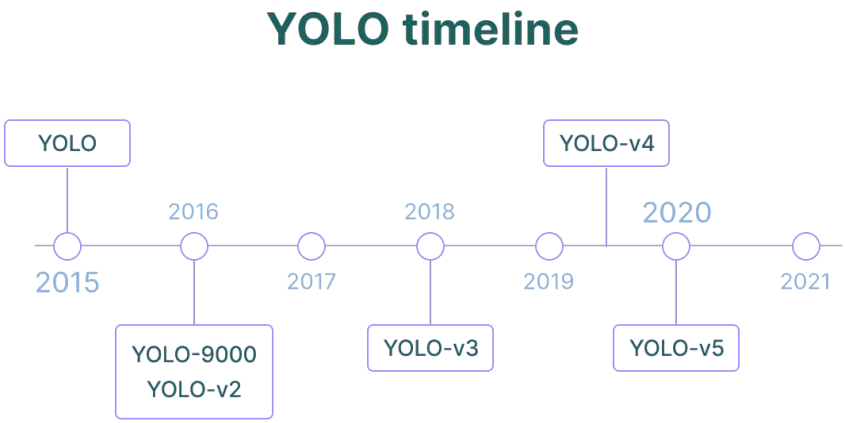
YOLO (You Only Look Once) navrhl Joseph Redmon, Santosh Divvala, Ross Girshick a Ali Farhadi v roce 2015. Tento detekční algoritmus a je známý svým výkonem v reálném čase. YOLO se velmi liší od předchozích algoritmů. Používá pouze jedinou neuronovou konvoluční síť současně k předpovídání pravděpodobnosti tříd, a i k návrhu hraničních rámečků pro každý návrh objektu, což je rychlejší a efektivnější než vícestupňové metody detekce. Funguje to tak, že vstupní obrázek si rozdělí na mřížku a každé mřížce přidá několik ohraničujících rámečků. Pro každý rámeček vygeneruje třídu pravděpodobnosti a hodnoty pro ohraničení. Jako výsledek se vyberou ty rámečky, které jsou nad prahovou hodnotou pravděpodobnosti. (Kundu, 2023)



Obr. 6.4 – Princip funkce YOLO (Gandhi, 2018)

YOLO je řádově rychlejší než jiné algoritmy. Dosahuje rychlosti až 45 FPS. Omezení dochází u detekce malých objektů, například hejna ptáků. To je způsobeno prostorovým omezením obrázku. (Gandhi, 2018)

6.5.1 Verze YOLO



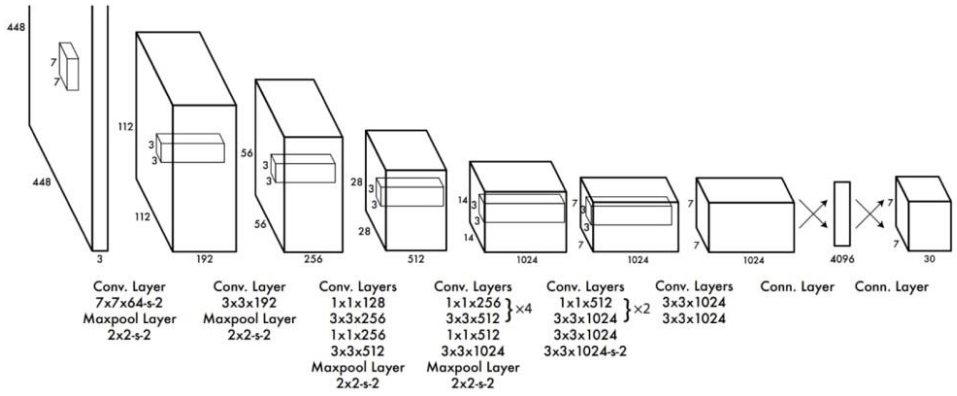
Obr. 6.5 – Časová posloupnost YOLO (Kundu, 2023)

Existuje několik typů modelů YOLO, které byly vyvinuty v průběhu let, z nichž každý má své vlastní jedinečné vlastnosti a vylepšení. Zde jsou některé z nejpozoruhodnějších:

- **YOLOv1:** Původní verze YOLO, která byla vydána v roce 2016. K vytváření předpovědí používá jedinou neuronovou síť a je známá svou rychlostí a efektivitou.
- **YOLOv2:** YOLOv2, vydané v roce 2017, zavedlo několik vylepšení oproti YOLOv1, včetně použití kotevních rámečků ke zlepšení předpovědí ohraničovacích rámečků, normalizace dávek pro zlepšení rychlosti tréninku a předpovědí ve více měřítcích pro zlepšení přesnosti.
- **YOLOv3:** Vydáno v roce 2018, YOLOv3 zavedlo několik dalších vylepšení oproti předchozí verzi, včetně použití pyramidové sítě funkcí *(FPN)* pro zlepšení extrakce funkcí, nové funkce ztráty pro zlepšení stability tréninku a vylepšeného ukotvení.
- **YOLOv4:** Vydáno v roce 2020, YOLOv4 zavedlo několik významných vylepšení oproti YOLOv3, včetně použití páteřní sítě *CSP (Cross Stage Partial)* ke zlepšení rychlosti a efektivity tréninku, využití prostorového pyramidového sdružování *(SPP)* ke zlepšení extrakce funkcí a zavedení nové architektury s názvem *YOLOv4-csp*, která kombinuje výhody YOLOv4 i *CSP*.
- **YOLOv5:** Vydáno v roce 2020, YOLOv5 je nejnovější verze YOLO a zavádí několik nových funkcí, včetně nové architektury s krkem a hlavou, použití *AutoML* k optimalizaci architektury neuronové sítě a použití nového tréninkového přístupu zvaného *self-labelling*, abyste se poučili z neoznačených dat.
- **YOLOv6:** Vydán v roce 2022 a novinkou je použitá architektura CNN. Pro YOLOv6 se používá EfficientNet-L2, která i s méně parametry dokáže vyšší výpočetní výkon.
- **YOLOv7:** Jedním z hlavních vylepšení je použití kotevních boxů. To umožňuje detekovat širší škálu tarů a velikostí objektů a pomáhá snížit počet falešných poplachů. Rychlost zpracovávat snímky s YOLOv7 je až 155 FPS a průměrná přesnost nabývá hodnotám okolo 37,2 % při prahové hodnotě IoU 0,5 na datové sadě COCO.
- **YOLOv8:** V současné době, tedy v roce 2023, vzniká nové YOLOv8. Slibuje nové funkce a vylepšený výkon. Má představit nové *API*, které výrazně usnadní trénování a vybavovaní. (Kundu, 2023)

6.5.2 Architektura YOLO

Architektura YOLO se skládá z jediné neuronové sítě, která zpracovává celý obraz v jediném dopředném průchodu. Síť se skládá z řady konvolučních vrstev následovaných dvěma plně propojenými vrstvami. Poslední vrstva sítě předpovídá ohraničující rámečky a pravděpodobnosti tříd pro každou buňku v mřížce. (Kundu, 2023)



The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224 input image) and then double the resolution for detection.

Obr. 6.6 – Architektura YOLO (Kundu, 2023)

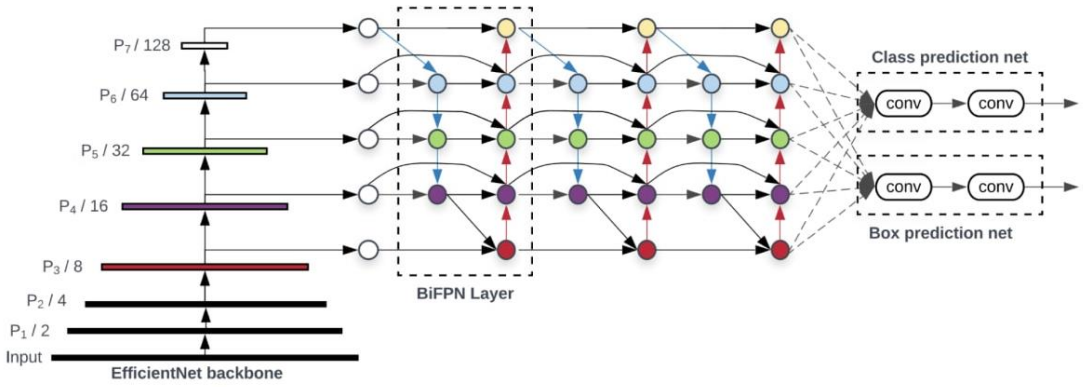
Zde je stručný popis hlavních kroků zahrnuté v architektuře YOLO:

- **Vstupní obrázek:** Vstupní obrázek je rozdělen do mřížky buněk. Velikost mřížky je dána výstupním rozlišením neuronové sítě.
- **Extrakce prvků:** Obraz prochází řadou konvolučních vrstev, aby se extrahovaly prvky ve více měřítcích. To je podobné procesu extrakce rysů používaného v jiných konvolučních neuronových sítích (CNN).
- **Predikce objektu:** Pro každou buňku v mřížce síť předpovídá, zda je v buňce přítomen objekt nebo ne. To se provádí předpovídáním jediného skóre pravděpodobnosti pro každou buňku, které představuje pravděpodobnost, že je v buňce přítomen objekt.
- **Predikce ohraničujícího rámečku:** Pro každou buňku v mřížce, která obsahuje objekt, síť předpovídá souřadnice ohraničujícího rámečku, který objekt těsně obklopuje. Souřadnice ohraničujícího rámečku jsou předpovězeny vzhledem k buňce a jsou transformovány na souřadnice celého obrázku.
- **Predikce třídy:** Pro každou buňku, která obsahuje objekt, síť předpovídá pravděpodobnost každé třídy objektu, na které byla trénována. To se provádí pomocí funkce *softmax*.

- **Nemaximální potlačení:** Ohraničující rámečky předpovězené sítí jsou filtrovány pomocí nemaximálního potlačení, aby se odstranily duplicitní předpovědi a zachovala se pouze ta nejpravděpodobnější. (Kundu, 2023)

6.5.3 YOLOv5

YOLOv5 je vylepšením oproti předchozím verzím YOLO v několika ohledech. Využívá pokročilejší architekturu, která se skládá z páteřní sítě následované krkem a hlavou. Páteřní síť je zodpovědná za extrahování prvků ze vstupního obrazu, zatímco krk a hlava jsou zodpovědné za predikci ohraničujících rámečků a pravděpodobností tříd.



Obr. 6.7 – Architektura sítě YOLOv5 (Kundu, 2023)

YOLOv5 také používá techniku nazvanou *Dynamic Convolution*, která upravuje velikost konvolučních filtrů použitých v modelu za účelem zlepšení efektivity a snížení využití paměti. YOLOv5 navíc používá nový tréninkový přístup zvaný *self-labelling*, který umožňuje modelu učit se z neoznačených dat předpovídáním štítků pro samotná data.

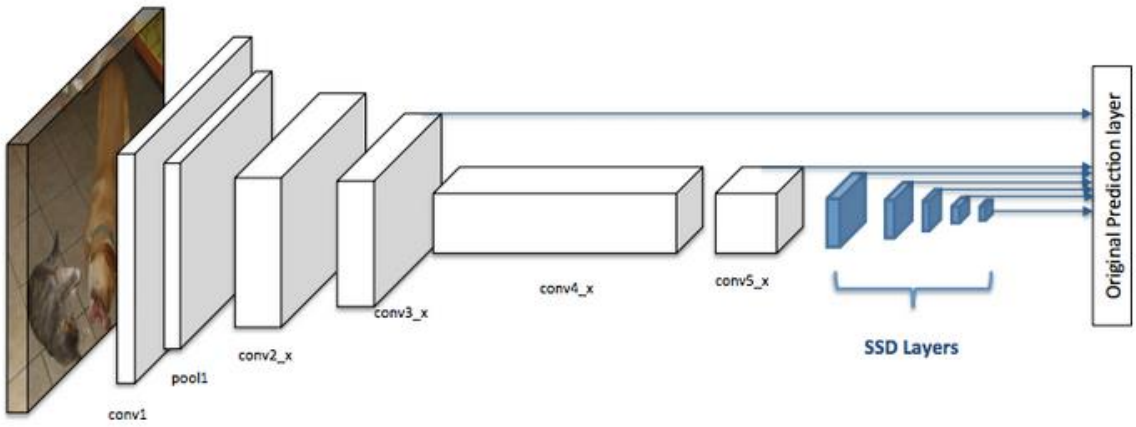
A konečně, YOLOv5 má k dispozici ke stažení několik před trénovaných modelů, které jsou optimalizovány pro různé případy použití, jako je detekce objektů na obrázcích nebo videích, a lze je doladit pro konkrétní aplikace.

Celkově jsou YOLO a YOLOv5 oblíbené systémy detekce objektů, které jsou známé svou rychlostí, efektivitou a přesností. Jsou široce používány v různých aplikacích, včetně robotiky, autonomních vozidel a video monitorovacích systémů. (Kundu, 2023)

6.6 SSD

Single Shot Detector (SSD) je algoritmus detekce objektů, který poprvé představil Wei Liu et al. v roce 2016. SSD je navržen jako rychlý a účinný algoritmus detekce objektů, který může pracovat v reálném čase na zařízeních s nízkou spotřebou. (How single-shot detector (SSD) works?, 2023)

Klíčovou inovací v SSD je použití jediné hluboké neuronové sítě pro provádění detekce a klasifikace objektů v jediném průchodu přes obraz. Díky tomu je SSD výrazně rychlejší než jiné algoritmy detekce objektů, jako je Faster R-CNN, které vyžadují vícenásobné průchody přes obraz pro generování návrhů objektů a provádění klasifikace. Predikce tedy není pouze z příznakové mapy poslední konvoluční vrstvy, ale ze všech příznakových map uvnitř celé sítě.



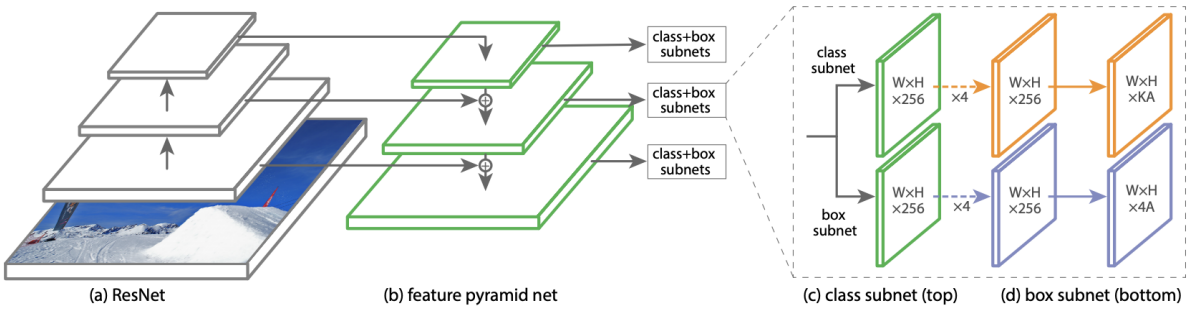
Obr. 6.8 – Architektura CNN s SSD (How single-shot detector (SSD) works?, 2023)

SSD využívá techniku nazvanou „kotevní boxy“, což jsou předem definované boxy různých velikostí a poměrů stran, které se používají k detekci objektů v různých měřítcích. Každý kotevní box je spojen se sadou tříd a offsetů pro ohraničující rámeček. Pro vyhodnocení jsou vypočteny polohy jednotlivých boxů a zároveň je pro každou třídu vypočteno skóre určující definující pozici v daném boxu. (Rais, 2022; How single-shot detector (SSD) works?, 2023)

6.7 RetinaNet

RetinaNet používá novou ztrátovou funkci nazvanou „Focal Loss“, která snižuje váhu příspěvku jednoduchých příkladů během tréninku a zaměřuje se na těžké příklady, které jsou s vysokou spolehlivostí špatně klasifikovány. Tato ztrátová funkce pomáhá algoritmu lépe zvládnout nerovnováhu tříd a zlepšit přesnost detekce objektů.

Architektura RetinaNet se skládá z páteřní sítě, jako je ResNet nebo DenseNet, která se používá k získání map prvků ze vstupního obrazu. Tyto mapy rysů pak procházejí sítí pyramid prvků (FPN), které generují sadu map prvků v různých měřítcích. Tyto mapy vlastností jsou pak přiváděny do detekční hlavy, což je sada konvolučních vrstev, které předpovídají ohraničující rámečky objektů a pravděpodobnosti tříd. (Lin, 2018)



Obr. 6.9 – Architektura RetinaNet (Lin, 2018)

7 KAMERY S AUTOMATICKÝM SLEDOVÁNÍM

Kamery, které umožňují automatické sledování, dokážou, jak už název napovídá, sledovat a zachycovat pohyby objektu v reálném čase. Tato technologie se začala běžně používat například na videokonferencích, živých akcích, u sportovních zápasů a velmi důležitý segment je i bezpečnostní monitorování.

Kamery s automatickým sledováním využívají pokročilé algoritmy a umělou inteligenci k detekci a sledování pohybu osob nebo objektů v zorném poli kamery. Funkce otáčení, náklonu a případně zoomu fotoaparátu jsou následně nastavovány tak, aby objekt zůstal v záběru a nejlépe pokud je to možné zaostřený.

Některé kamery s automatickým sledováním používají senzory, jako jsou infračervené nebo ultrazvukové senzory, k detekci pohybu objektu. Zatímco jiné, pokročilejší kamery, používají technologii rozpoznávání obrazu ke sledování objektu. Některé modely také přicházejí s dalšími funkcemi, jako je rozpoznávání obličeje, gest nebo hlasové ovládání.

Je tedy zřejmé, že tento způsob natáčení nebo fotografování lze uplatnit v různých prostředích, jako posluchárny či konferenční místnosti. Díky této technologii navíc můžeme nerušeně a automaticky zlepšit zážitek ze sledování pro nejen účastníky na místě, ale i ty vzdálené. Automatické kamery se také velmi často používají pro bezpečnostní účely. Zde například pro zajištění určitého perimetru nemusí být zapotřebí několik kamer, ale jedna z možností automatického pohybu a sledování narušitelů nebo podezřelých osob. (Valenta, 2021)

7.1 Základní rozdělení kamer

Kamery, které využívají automatické sledování, lez dále rozdělit do několika podskupin, přičemž, každý typ kamery má své jedinečné vlastnosti a výhody. Jednotlivé skupiny kamer se tedy navzájem propojují a kombinují.

7.1.1 Kamery s umělou inteligencí

Tyto kamery využívají umělou inteligenci a algoritmy strojového učení k analýze obrázků nebo videí zpravidla v reálném čase. Jsou schopny provádět různé úkoly, jako je detekce objektů, rozpoznávání tváří a předpovídání událostí na základě vzorů a dat. Běžně

používají pro video dohled a bezpečnostní aplikace, protože dokážou rychle analyzovat obrovské množství záběrů a poskytovat upozornění v reálném čase. (Mlčoch, 2012)

7.1.2 Stropní kamery

Stropní kamery jsou obvykle ve vnitřních prostorech, jako jsou kanceláře, školy nebo prodejny. Tyto kamery jsou navrženy tak, aby poskytovaly pohled na místnost nebo oblast z ptačí perspektivy, díky čemuž jsou užitečné pro monitorování velkých prostor nebo davů. Stropní kamery mohou být buď pevné, nebo kamery PTZ. (Mlčoch, 2012)

7.1.3 Bezdrátové kamery

Bezdrátové kamery se mohou připojit k síti nebo internetu bez potřeby kabelů nebo drátů. Obvykle se používají v situacích, kdy je vedení kabelů nepraktické nebo nemožné, například ve vzdálených místech nebo ve venkovním prostředí. Bezdrátové kamery mohou být napájeny z baterie nebo z externího zdroje energie. (Mlčoch, 2012)

7.1.4 Bezpečnostní kamery

Mohou být buď drátové nebo bezdrátové. Použít je lze podle typu uvnitř nebo venku. Bezpečnostní kamery se běžně používají v domácnostech, podnicích a na veřejných prostranstvích k odvrácení zločinu, sledování aktivity a poskytování důkazů v případě incidentu. (Mlčoch, 2012)

7.1.5 Robotické kamery

Robotické kamery jsou namontovány na robotických platformách, což jim umožňuje pohybovat se a upravovat jejich zorné pole ve více směrech. Tyto kamery se obvykle používají v situacích, kde je důležitá mobilita. Hlavní využití nalezli v průmyslovém prostředí, výrobních závodech, ale používají se i ve venkovním prostředí. Robotické kamery lze ovládat na dálku nebo naprogramovat tak, aby sledovaly předem stanovenou dráhu.

7.2 PTZ kamery

PTZ kamera je typ kamery, která dokáže posouvat, naklánět i přibližovat obraz a ten také zachycovat. Je vhodná širokou škálu použití v obrazové technice. PTZ je zkratka pro pan, tilt a zoom. Blíže *pan* označuje horizontální pohyb, tedy do stran. *Tilt* označuje vertikální pohyb kamery, který jí umožňuje pohybovat se nahoru a dolů. *Zoom* označuje schopnost fotoaparátu upravit čočku pro zvětšení obrazu nebo pro zobrazení širšího zorného pole.

PTZ kamery se běžně používají pro bezpečnostní aplikace, protože mohou pokrýt velkou oblast jedinou kamerou a lze je dálkově ovládat pro sledování objektů nebo osob. Používají se také při videokonferencích, živém vysílání a tam, kde je potřeba zachytit různé úhly scény nebo ke sledování přednášejícího při pohybu.

Tyto kamery se zpravidla dodávají i s ručním ovládáním, takže obsluha má k dispozici dálkové ovládání nebo joystick. Samozřejmě primární funkce je automatická detekce a sledování. U pokročilejších modelů se lze setkat se sledováním obličeje, stabilizaci obrazu nebo i možnost zobrazení tepelného spektra.

Existuje několik druhů a typů těchto kamer, od malých kompaktních modelů pro výhradně vnitřní použití až po robustní větší modely, které dokáží odolat povětrnostním vlivům. (Kunar, 2009; Valenta, 2021)

Obr. 7.1 – PTZ kamera AREC CI-T21H (Valenta, 2021)

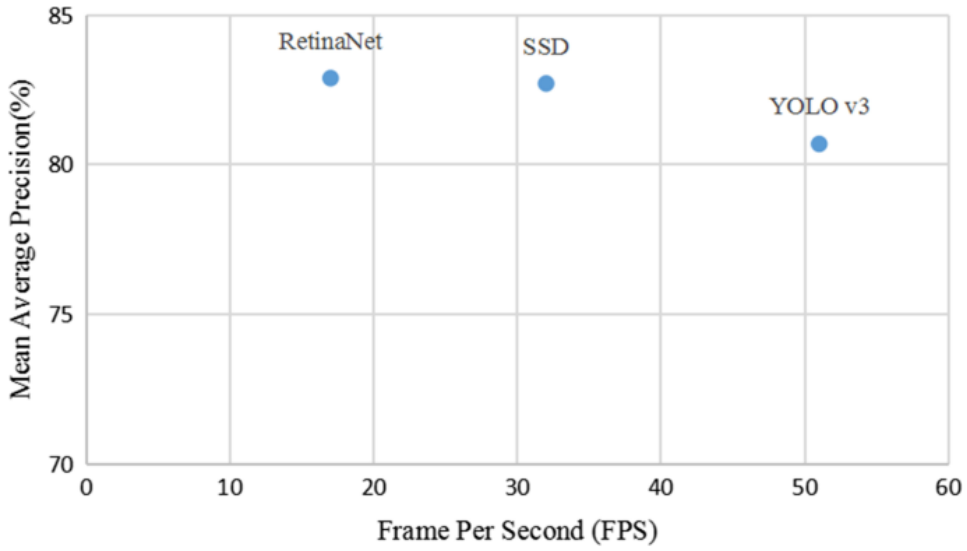
8 POUŽITÁ TECHNIKA A ALGORITMY

V podkapitolách jsou popsány jednotlivé algoritmy, technika i programy, které byly nutné k vypracování jak hardwarové části, tak i softwarové části prototypu kamery pro sledování objektu v prostoru.

8.1 Výběr detekčního algoritmu

Srdcem celého programu je detekční algoritmus, na který byl velký nárok co se tyká rychlosti zpracování obrazu. Rychlost detekce byla tedy klíčová, protože se jedná o zařízení, které musí pracovat v reálném čase. Výběr se tedy provedl z již výše uvedených detekčních algoritmů tedy R-CNN a jeho rychlejší verze, dále SSD, RetinaNet a samozřejmě YOLO.

Porovnáním vlastností se Faster R-CNN, RetinaNet a SSD příliš nelišili nicméně rychlost těchto modelů dosahuje maximálně 40 FPS. Srovnání výkonu detekčních sítí je na obrázku níže, kde je i zřejmé, že už YOLOv3 dosahuje vyšších rychlostí FPS než ostatní modely. (Tan, 2021)



Obr. 8.1 – Porovnání výkonu detekčních algoritmů (Tan, 2021)

Neustálím vývojem architektur a vlastností je již k dispozici YOLO ve verzi YOLOv7. Nicméně čím vyšší řada tím je vyšší i výpočetní náročnost. Dostatečný a stále velmi používané starší modely mají často vetší variabilitu a více použitelných, dohledatelných příkladů použití. Na testovací datové sadě jsou novější modely i přesnější, ale často jejich trénování je zdlouhavé a velmi náročné. Pro všechny tyto aspekty se jeví YOLOv5 jako nejideálnější varianta. Navíc další vlastností, která byla z počátku výběru velmi důležitá byla aplikace detekčního modelu na

platformu Raspberry. Takže i výpočetní náročnost hrála důležitou roli. V současné době má být představen právě nový YOLOv8, který by měl slibovat opět lepší vlastnosti. Nicméně v současné době zatím nebyl k dispozici, a tak výsledným vybraným algoritmem je YOLOv5s, který je navíc druhý nejmenší a jeho vlastnosti naprosto splňují požadavky.

8.2 Kamera Basler

Německá firma Basler AG je předním světovým výrobcem digitálních kamer a kamerových systémů pro průmyslové aplikace. Založena byla již v roce 1988. Specializuje se na vývoj a výrobu vysoce kvalitních kamer, které se používají v různých průmyslových odvětvích, jako je automatizace továren, lékařská technika, dopravní systémy a další.

Obr. 8.2 – Logo firmy Basler AG (Basler AG. Company Profile, 2023a)

Fotoaparáty Basler jsou známé svým vysokým výkonem, spolehlivostí a vynikající kvalitou obrazu. Jsou navrženy tak, aby splňovaly požadavky průmyslových aplikací, kde je přesnost a rychlost rozhodující. Basler nabízí širokou škálu modelů fotoaparátů s různými rozlišeními, technologiemi senzorů a rozhraními, aby vyhověl všem požadavkům. Basler kamery jsou vybaveny dle typu různými standardními průmyslovými rozhraními, jako je USB, Gigabit Ethernet a Camera Link, což umožňuje snadnou integraci do stávajících systémů. (Basler AG. Company Profile, 2023a)

Pro tuto práci byla vybrána kamera ze série *ace Classic*, přímo tedy *acA2500-14uc*. Tato kamera je pro aplikaci vhodná díky malým rozměrům (29 x 29 mm), a na to, že se jedná o průmyslovou kameru tak má i nízkou pořizovací cenu. Velikost rozlišení je velmi vysoké a to 2590 x 1942 pixelů. Připojení s počítačem je možné díky rozhraní USB nebo Camera link. (Basler AG. Company Profile, 2023b)

Obr. 8.3 – Kamera Basler acA2500-14uc (Basler AG. Company Profile, 2023a)

Další důležitou součástí je samozřejmě kompatibilní objektiv. Důležité vlastnosti jsou hlavně ohnisková vzdálenost, samotné rozměry objektivu, zorný úhel atd. Jako vhodný se ukázal objektiv typu 16L od společnosti Computar s ohniskovou vzdáleností 5 mm.

8.3 Pylon Viewer

Jedná se o oficiální software od společnosti Basler AG pro nastavená parametrů kamer. Díky této aplikaci je možné nastavit různé parametry a vyzkoušet, zda je správně připojená a nastavená. Program umožňuje i výběr z řady nástrojů jako indikátor ostrosti obrazu nebo histogram kamery. Software je k dispozici zcela zdarma na oficiálních stránkách, ale nejedná se o opensource. (Basler AG. Company Profile, 2023c)

8.4 PyPylon

PyPylon slouží jako most mezi Pythonem a Pylonem a umožňuje snadno integrovat kamery Basler do aplikací založených na Pythonu. Poskytuje pohodlný a intuitivní způsob interakce s fotoaparátem, pořizování snímků nebo videí a přístup k různým parametrům fotoaparátu. Jako právě důležité zpracování snímků, ovládání kamery, nebo podpora více kamer. Navíc je jedná o otevřený software. (Basler AG., 2023)

Obr. 8.4 – PyPylon API (Basler AG., 2023)

8.5 Python

Python je interpretovaný programovací jazyk na vysoké úrovni, který se široce používá pro vývoj různých typů softwarových aplikací. Byl vytvořen na konci 80. let *Guido van Rossum* a nyní je spravován *Python Software Foundation*. Python je známý svou jednoduchostí, čitelností a všestranností, díky čemuž je oblíbenou volbou mezi vývojáři pro širokou škálu úkolů, od vývoje webu přes vědecké výpočty až po umělou inteligenci.

Mezi klíčové funkce patří snadné učení, má velkou podporu knihoven a rozšíření a je objektově orientovaný. Dále je také multiplatformní, a to je v této práci velmi důležité, protože se propojuje několik přístrojů a programů. Součástí hlavní instalace je i debugger, což umožňuje rychlé a snadné ladění kódu. (What is Python? Executive Summary, 2023)

Obr. 8.5 – Logo Python (What is Python? Executive Summary, 2023)

8.6 Torch a PyTorch

Torch a PyTorch jsou příbuzné, ale odlišné pojmy. Oba ale se pohybují v kontextu neuronových sítí a strojového učení.

Torch, také známý jako Torch7, je open-source vědecký výpočetní nástroj, který poskytuje širokou škálu matematických operací a algoritmů. Původně jej vyvinuli Ronan Collobert, Koray Kavukcuoglu a Clement Farabet a je napsán v programovacím jazyce Lua. Torch se zaměřuje na tenzorové operace a zahrnuje moduly pro lineární algebru, zpracování signálu, zpracování obrazu a strojové učení. Ve výzkumné komunitě je široce používán pro své snadné použití.

PyTorch je nástroj pro hluboké učení založené na Pythonu Staví na základních funkcích Torche. Byl vyvinut laboratoří *Facebook AI Research (FAIR)* a představen jako open-source projekt. PyTorch si zachovává možnosti výpočtu tenzorů Torche, ale poskytuje i více uživatelsky přívětivé rozhraní, což vývojářům usnadňuje programování, iteraci a vytváření

modelů hlubokého učení. PyTorch je v poslední době velmi oblíbený díky flexibilitě a jedním z předních nástrojů pro výzkum a vývoj hlubokého učení. (Yasar, 2022)

Obr. 8.6 – Logo PyTorch (Yasar, 2022)

8.7 OpenCV

OpenCV *(Open Source Computer Vision)* je open-source knihovna pro počítačové vidění a strojové učení. OpenCV původně vyvinula společnost Intel v roce 1999 a od té doby se stala široce používanou a velmi oblíbenou knihovnou v oblasti počítačového vidění.

OpenCV implementuje širokou škálu algoritmů počítačového vidění, jako je detekce hran, detekce rohů, segmentace obrazu, rozpoznávání objektů a kalibrace kamery. Tyto algoritmy umožňují úkoly, jako je detekce objektů, rozpoznávání obličeje či optické rozpoznávání znaků. Dále také poskytuje implementaci platforem jako TensorFlow a PyTorch. Poskytuje také optimalizované funkce pro analýzu obrazu a videa v reálném čase, díky čemuž je vhodný pro aplikace, jako jsou sledovací systémy, autonomní vozidla a robotika.

OpenCV je podporována napříč platformami a komunita okolo tohoto nástroje je velmi aktivní a dobře zdokumentovaná. (About OpenCV, 2023)

Obr. 8.7 – Logo OpenCV (About OpenCV, 2023)

8.8 Modbus TCP/IP

Modbus je komunikační protokol, který se běžně používá v průmyslových automatizačních systémech pro výměnu dat mezi zařízeními, jako jsou programovatelné logické automaty (PLC), senzory a další zařízení. Byl vyvinut na konci 70. let a od té doby se stal široce přijatým standardem v průmyslovém sektoru. Protokol sám o sobě podporuje sériovou komunikaci (RS232/RS485) tak právě i komunikaci přes Ethernet (TCP/IP).

Řídí se architekturou master-slave, kde hlavní zařízení zahajuje komunikaci a odesílá požadavky jednomu nebo více podřízeným zařízením, která odpovídají požadovanými daty. Protokol definuje různé funkční kódy pro různé operace, včetně čtení a zápisu diskrétních vstupů, vstupních registrů a přídržných registrů. Podporuje také další funkce, jako je detekce a zpracování chyb. V této práci je použita knihovna PyModbusTCP v Pythonu. (What is the Modbus Protocol & How Does It Work?, 2023)

PyModbusTCP je knihovna Pythonu, která poskytuje implementaci protokolu Modbus TCP/IP. Umožňuje vývojářům snadno komunikovat se zařízeními Modbus pomocí kódu Python. PyModbusTCP zjednodušuje proces navazování připojení Modbus, odesílání požadavků a přijímání odpovědí. S touto knihovnou lze vytvářet klientské aplikace jako zápis a čtení registrů. PyModbusTCP je knihovna s otevřeným zdrojovým kódem a je aktivně udržována a podporována komunitou. Je kompatibilní s různými platformami a operačními systémy, takže je vhodná pro širokou škálu aplikací. (Lefebvre, 2023)

8.9 Raspberry Pi 4B

Raspberry Pi 4 Model B je jednodeskový počítač vyvinutý nadací *Raspberry Pi Foundation*. Jedná se o čtvrtou generaci modelu v řadě Raspberry Pi a ve srovnání se svými předchůdci nabízí výrazné zlepšení výkonu, konektivity a možnostmi aplikace.

Raspberry Pi 4B, které je k dispozici pro tuto práci poskytuje:

- Čtyřjádrovým procesor Broadcom BCM2711 Cortex-A71 s taktem až 1,5 GHz. Výsledkem je znatelné zvýšení celkového výkonu, díky čemuž je schopen zvládat náročnější úkoly.
- RAM paměť o velikosti 8 GB LPDDR4 SDRAM.

- Dále obsahuje jednotku grafického zpracování VideoCore VI (GPU) s podporou OpenGL ES 3.0 a přehrávání videa 4K rychlostí 60 snímků za sekundu. Má také dva porty HDMI, které mohou ovládat dva displeje současně.
- Je vybaveno dvěma porty USB 3.0, dvěma porty USB 2.0, portem Gigabit Ethernet, dvoupásmovou 2,4 GHz a 5 GHz bezdrátovou sítí LAN 802.11ac, Bluetooth 5.0 a podporou napájení přes Ethernet prostřednictvím samostatného HAT.
- Stejně jako předchozí modely Raspberry Pi má Raspberry Pi 4B 40pinový GPIO *(General Purpose Input/Output)* header, který umožňuje snadné propojení s celou řadou externích zařízení a senzorů.
- Podporuje celou řadu operačních systémů. Nicméně nejpoužívanější, a i zde je použit operační systém na základě Ubuntu.

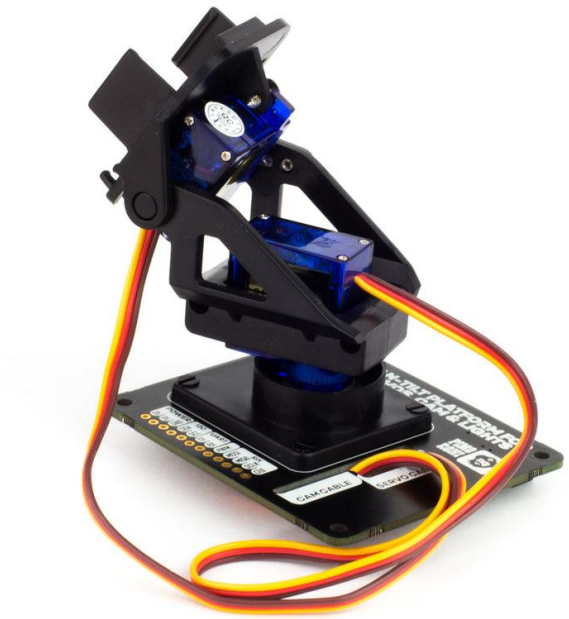
Raspberry vyžaduje samostatný napájecí zdroj, microSD kartu pro ukládání. Dále periferie jako klávesnice, myš a display. Nicméně k programování je možné využít například program VNC Viewer, který umožňuje propojení s počítačem, a tak nutnost vlastních připojených periférii odpadá. (Upton, 2016; Norris, 2015)

Obr. 8.8 – Mikropočítač Raspberry Pi 4B (Botland, 2023)

8.10 Pan-Tilt HAT

Pan-Tilt HAT je hardwarová přídavná deska pro Raspberry Pi. Je vyvinuta britskou společností Pimoroni, která vyrábí řadu elektroniky a příslušenství pro Raspberry Pi a další mikrokontroléry. Pan-Tilt HAT umožňuje ovládat dva servomotory, které mohou pohybovat kamerou nebo jiným zařízením horizontálně i vertikálně. Zároveň jsou tyto pohyby nezávislé.

Pan-Tilt HAT obsahuje ovladač PCA9685 PWM, který může ovládat až 16 kanálů signálů PWM, takže je ideální pro ovládání servomotorů. Je také vybaven rozhraním I2C, které umožňuje připojení k GPIO pinům Raspberry Pi. Pimoroni poskytuje knihovnu Python a příklady pro ovládání Pan-Tilt HAT, což usnadňuje integraci do projektů. (Pan-Tilt HAT, 2023)



Obr. 8.9 – Pan-Tilt HAT od Pimoroni (Pan-Tilt HAT, 2023)

8.11 Google Colaboratory

Google Colaboratory, také známá jako Colab, je cloudová platforma vyvinutá společností Google, která poskytuje bezplatné a interaktivní prostředí pro psaní, spouštění a sdílení kódu. Je postaven na platformě *Jupyter* Notebook a je primárně určen pro datovou vědu, strojové učení a úkoly umělé inteligence.

Díky tomu, že se jedná o cloudovou platformu tak není třeba instalovat žádný software, protože je přístup k virtuálnímu počítači s již instalovanými knihovnami. Tím že je prostředí

postaveno z Jupiter Notebook, je propojeno jak vytváření kódu, tak s živými poznámkami, které mohou obsahovat text, obrázky i videa. Dále poskytuje velkou škálu knihoven, které se využívají ve strojovém učení. Nejznámější je například *NumPy, Pandas, TensorFlow, Pytorch* a mnoho dalších.

Colab poskytuje přístup k bezplatným zdrojům GPU (*Graphical Processing Unit*) a TPU (*Tensor Processing Unit*), které jsou zvláště užitečné pro urychlení hloubkového učení a výpočetně náročných úloh. Tyto prostředky lze přidělit také na vlastní notebook pro rychlejší práci s učením.

Dále podporuje společné úpravy, což umožňuje více uživatelům spolupracovat na stejném projektu, tedy programu, v reálném čase. Své poznámkové bloky je možné také sdílet s ostatními vygenerováním odkazů. Díky tomu je vhodný pro spolupráci a sdílení znalostí.

I když Colab je volně k dispozici, tak běží na virtuálních počítačích s omezenými prostředky. Omezení se týká hlavně doby běhu a to na 12 hodin. Dále je zde limitace paměti a místa na disku. Nicméně je možné pro náročnější uživatele využít placenou verzi Colab Pro+, které má výrazně menší omezení. Colab umožňuje snadno experimentovat, vytvářet prototypy a sdílet kód bez nutnosti místního nastavení, což z něj činí cenný nástroj pro analýzu dat. (Ming Chng, 2022)

V této diplomové práci byla využita bezplatná verze Colab k trénování modelu na základě vytvořené datové sady.

Obr. 8.10 – Logo Google Colaboratory (Ming Chng, 2022)

8.12 Thonny IDE

Thonny je uživatelsky přívětivé integrované vývojové prostředí (IDE) speciálně navržené pro začátečníky, kteří se učí programovat v Pythonu, ale svou jednoduchostí je oblíbený všemi programátory. Má čisté a minimalistické rozvržení, které zabraňuje zahlcení

uživatelů nadbytečnými funkcemi a možnostmi. Rozhraní se snadno orientuje a poskytuje užitečné tipy a návrhy, které uživatelům pomohou pochopit syntaxi a koncepty Pythonu. Thonny klade důraz na jednoduchost a srozumitelnost.

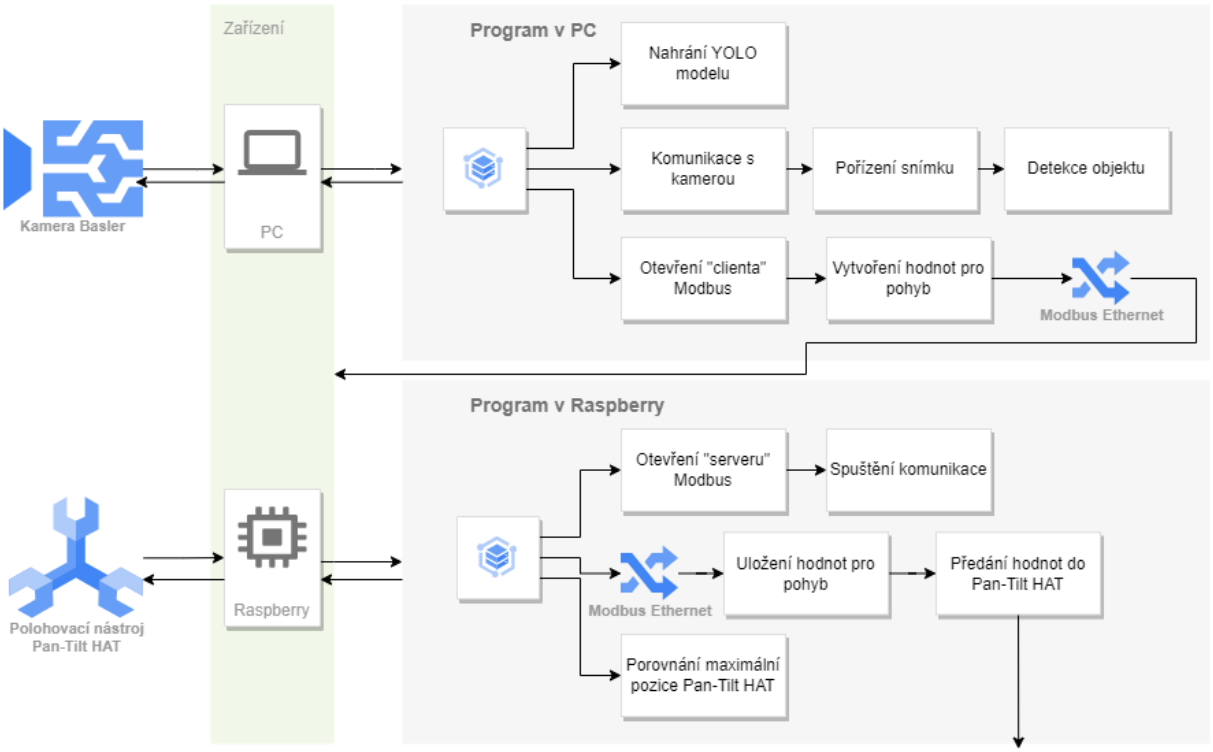
Spouštění a ladění je možné krok za krokem, což je výhodné nejen pro začátečníky, ale také je to důležité pro pochopení částí programu, jednotlivých prvků a identifikovat jednotlivé chyby. Prostředí také poskytuje průzkumník proměnných, který umožňuje prohlížet a kontrolovat hodnoty proměnných v různých fázích provádění programu.

Thonny je kompatibilní s různými operačními systémy, včetně Windows, macOS a Linuxu, takže je přístupný širokému spektru uživatelů. V současné době existuje poslední verze Thonny 4.0.2. (Thonny Python IDE for beginners, 2023)

Obr. 8.11 – Logo Thonny IDE (Thonny Python IDE for beginners, 2023)

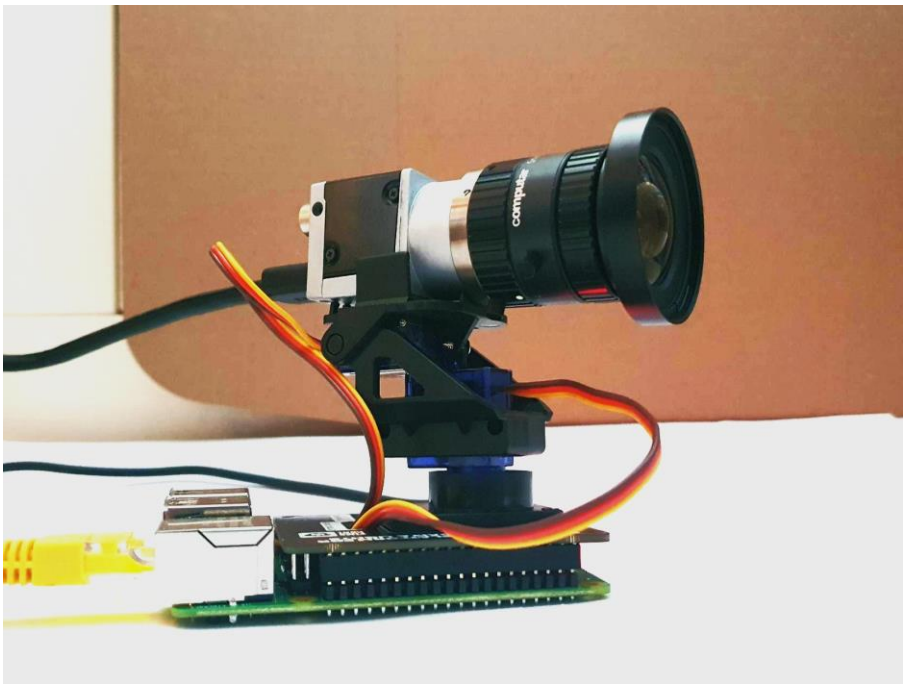
9 NÁVRH A IMPLEMENTACE PROTOTYPU

Cílem funkce prototypu zařízení je detekovat a následně sledovat daný objekt. Nejprve je tedy nutné přesně lokalizovat objekt v prostoru zorného pole kamery. Tuto informaci je třeba vyhodnotit a podle nastavení a porovnáním s mezemi, které definují střed, předat do systému zajišťující pohyb. Vyhodnocení probíhá pomocí před trénovaného modelu YOLOv5s na vytvořené datové sadě. Datová sada byla vytvořena a anotována pomocí aplikace *Roboflow*. Model YOLOv5s byl vytvořen a trénován pomocí nástroje Google Colaboratory.



Obr. 9.1 – Funkční schéma prototypu

Pohyb je realizován pomocí Raspberry a pro ni speciální periferie Pan-Tilt HAT. Periferie je určena pro lehčí kamery, nebo jiná zařízení a její výstup udává přímo pozici servomotorů zajištující jak vertikální, tak horizontální pohyb. Kamera je tedy umístěna v držáku této periférie a pro vetší stabilitu je kamera připevněna k podstavci. Pro lepší manipulaci a vyhodnocovací perimetr je celé zařízení připevněno do chytu stativu, který není nijak speciálně upravený.



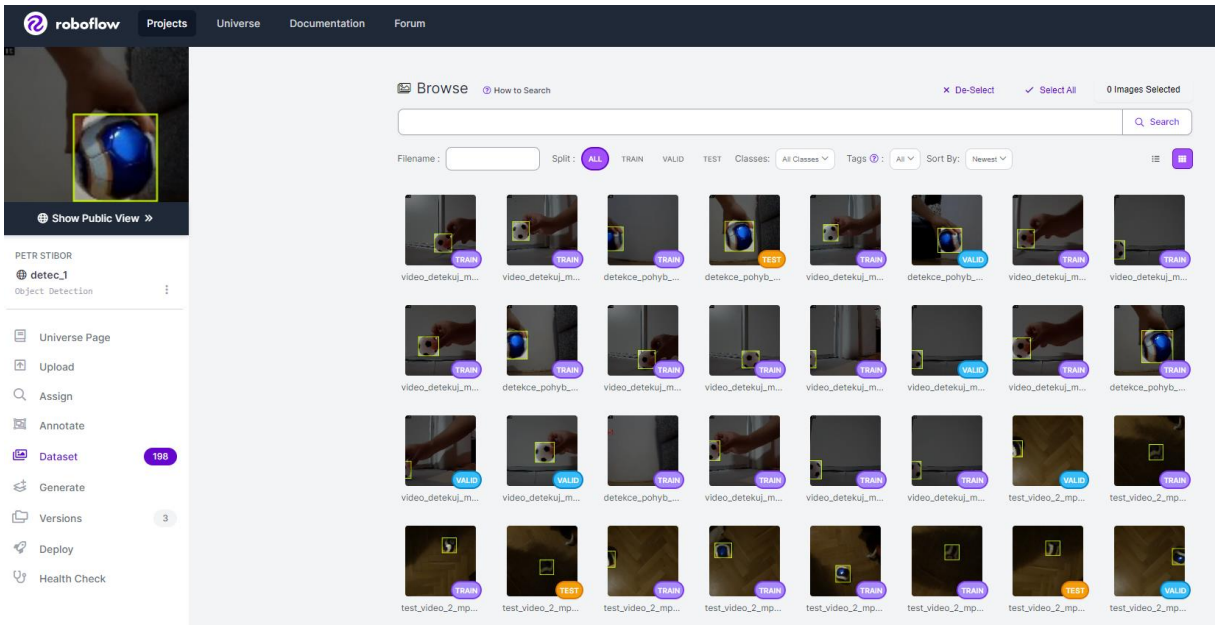
Obr. 9.2 – Podoba prototypu zařízení bez stativu

Kamera je připojena do notebooku pomocí USB kabelu. V notebooku tedy probíhá první program pro zpracování obrazu z kamery, vytvoření klienta pro komunikaci pomocí Modbus a předání potřebných informací pro pohyb.

V počítači Raspberry je spuštěn druhý program, který zajišťuje otevření serveru pro komunikaci po sběrnici Modbus. Dále zajištění daných registrů a jejich programové zpracování a předání hodnot pro připojené dvoupolohové zařízení Pan-Tilt HAT. To zajišťuje natočení a vrací hodnoty polohy obou servomotorů.

9.1 Sběr dat a tvorba datové sady

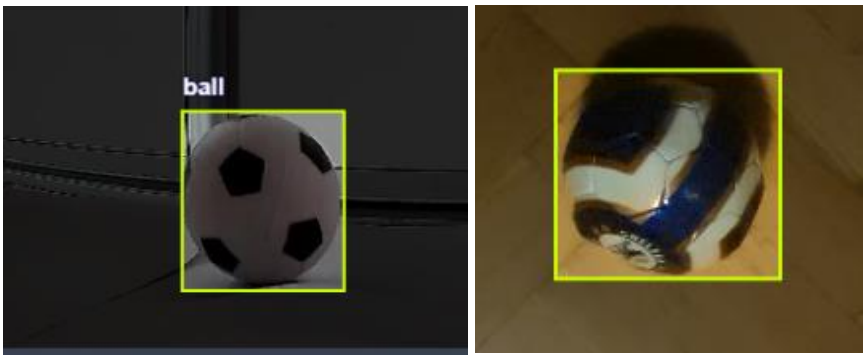
Data byla pořizována jak na kameru Basler, tak i na jiná zařízení kvůli různorodosti vlastností a víceúčelovosti tvořeného modelu. Dále bylo důležité obrázky pořizovat jak při takřka ideálních světelných podmínkách, tak i za šera. Světlo dopadající na objekt se ukázalo jako velký problém pro detekci. Datová sada byla tedy zpracovávána v aplikaci *Roboflow*. Tato aplikace je velmi oblíbená právě pro vytváření datových sad. Díky tomu, že je velmi jednoduchá na práci a její výstup lze jednoduše přes odkaz využít pro trénování v *Colabu*. Nicméně trénování modelu lze uskutečnit i přímo v aplikaci Roboflow, ale tento model se ukázal jako méně kvalitní. Náhled do aplikace je na obrázku níže.



Obr. 9.3 – Náhled do aplikace Roboflow

9.1.1 Anotace sledovaného objektu

Detekční model byl tedy použit pouze pro detekci objektu v prostoru zorného pole kamery. Jako detekovaný objekt byl vybrán fotbalový míč v různých velikostech a barvách. Pro tento objekt byla vytvořena datová sada v aplikaci *Roboflow*. V tomto prostředí je možné nahrát požadovaná data, která lze následně celkem jednoduše anotovat. Anotace je výběr objektu v obraze a jeho přiřazení do určité třídy. V tomto případě se tedy jednalo o jednu třídu nazvanou „ball“.



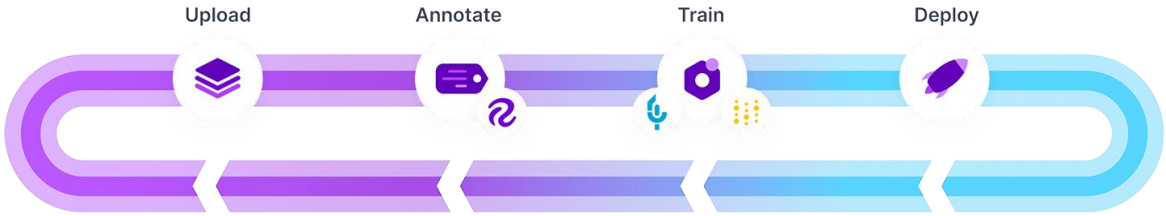
Obr. 9.4 – Anotace objektu

Celková velikost datové sady tvořila přibližně 300 obrázků, přičemž pro trénování byl každý zrcadlově převrácen a vložen jako nový obrázek. Tudíž tím se celkový data set ještě zvětšil.

9.2 Trénování modelu YOLOv5

Pro rychlejší proces trénování byl proces spuštěn na platformě Google Colaboratory, který využívá GPU akcelerátor výpočtů.

Dále je postup tedy rozdělen do čtyřech kroků, které budou popsány v dalších podkapitolách. (Custom Training with YOLOv5, 2023)



Obr. 9.5 – Postup vytváření modelu (Custom Training with YOLOv5, 2023)

9.2.1 Příprava datové sady

Pro trénování vlastního modelu YOLOv5 byla využita připravená vlastní datová sada, která se skládá z označených obrázků přidělených do konkrétní třídy. Datový soubor obsahuje poznámky ohraničujících rámečků kolem detekovaného objektu a štítek. Datovou sadu z Roboflow je možné skrz odkaz s dalšími parametry vložit přímo do Colabu, nebo ji lze stáhnout jako zazipovaný soubor a vložit do pracovní plochy Colabu nazvaného jako Soubory v levé vyskakovací liště.

9.2.2 Konfigurace modelu

YOLOv5 poskytuje různé předdefinované konfigurace, jako jsou YOLOv5s, YOLOv5m, YOLOv5l a YOLOv5x, které se liší velikostí modelu a složitostí. Lze tedy vybrat vhodnou konfiguraci na základě výpočetních zdrojů a požadované přesnosti. Pro tuto práci byla vybrána nejmenší velikost výsledného modelu a to YOLOv5s, protože svou rychlostí k poměru velikosti splňuje možnost implementace do Raspberry Pi 4B. Bohužel k tomu nakonec nemohlo dojít protože Raspberry Pi, které bylo k dispozici nesplňovalo všechny potřebné parametry.

9.2.3 Tréninkový proces

Jakmile je datová sada a konfigurace modelu připravena, je třeba definovat nastavení tréninku. To zahrnuje nastavení parametrů školení, jako je rychlost učení, velikost dávky, počet epoch a techniky rozšíření. Rozšiřující techniky, jako je náhodné oříznutí, převrácení a změna měřítka, mohou pomoci zvýšit rozmanitost trénovacích dat a zlepšit zobecnění modelu.

Trénovací skript *train.py* má celou řadu parametrů, mezi kterými je třeba najít ideální rovnováhu. Parametry, které je třeba nastavit pro trénování:

- --*img* je pro velikost vstupního obrázku,
- --*batch* určuje velikost dávky,
- --*epochs* nastavuje počet epoch,
- --*data* odkazuje na cestu, kde se nachází konfigurační soubor datové sady,
- --*weights* složí pro nastavení výchozích vah.

Podoba příkazu pro trénování je na obrázku níže.

```
!python train.py --img 416 --batch 16 --epochs 150 --data {dataset.location}/data.yaml --weights yolov5s.pt
```

Obr. 9.6 – Příkaz pro trénování modelu YOLOv5s

9.2.4 Vyhodnocení a nasazení

Výsledky trénování jsou samostatně popsány v kapitole viz 9.3.

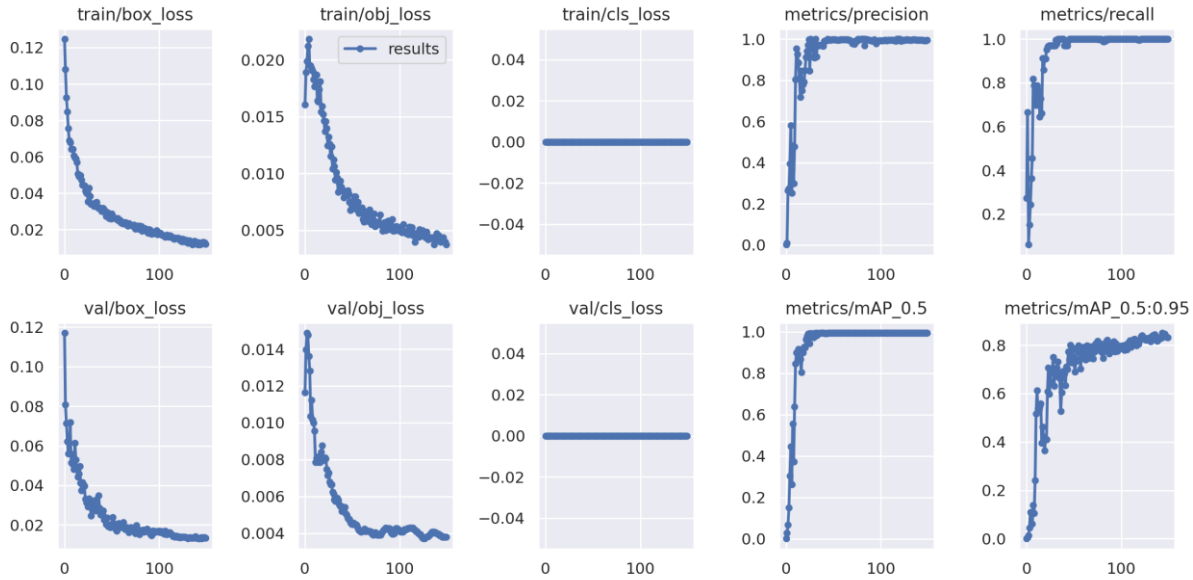
Jakmile je model trénován a vyhodnocen, může být použit pro detekci objektů na nových, neviditelných datech. Výsledný model je v Colabu uložen opět v příslušném Google disku v levé části okna. Poslední trénovaný modelu je uložena pod nevyšším číslem *exp* a pokud se jedná o jediné spuštění, tak k modelu označeného jako *best.pt* vede cesta *'./runs/train/exp/weights/best.pt'*. Tento model lze jednoduše zazipovat a z Colabu stáhnout jako jednoduchý soubor zip. Tento soubor ale pro další použití je třeba znovu otevřít a uložit jako nezazipovaný adresář.

9.3 Výsledky trénování

Výsledky trénování se udávají v metrice jako střední průměrná přesnost *(mAP)* a posuzují přesnost a kvalitu modelu. *mAP* se měří na specifické prahové hodnotě, tedy detekce je považována za správnou, pokud je překrytí mezi předpovězeným a definovaným rámečkem

nad procentuální hodnotou prahové hodnoty. Pro metriku *mAP_0.5:0.95* vyhodnocujeme výkon modelu v rozsahu prahových hodnot v krocích po 0,05. Tato metrika poskytuje komplexnější hodnocení výkonu modelu, protože bere v úvahu různé úrovně překrytí předpovězených a základních rámečků. Čím vyšší číslo, tím lepší výkon detekce objektů. Přičemž volba prahu velmi ovlivňuje výslednou detekci, například pro práh nad úroveň 0,9 má za následek velmi přísného kritéria a pro detekci u reálného zařízení byla takřka nepoužitelná. Hodnota pro *mAP_0.5* se často používá jako standartní měřítko pro porovnání různých modelů detekce. (Rais, 2022)

Konečné výsledky vytvořeného modelu pro *mAP_0.5* se na validační množině na úrovni 0,995 a pro výsledek *mAP_0.5:0.95* vyšla hodnota 0,83. Všechny dostupné metriky trénovacího cyklu jsou na obrázku 9.7.

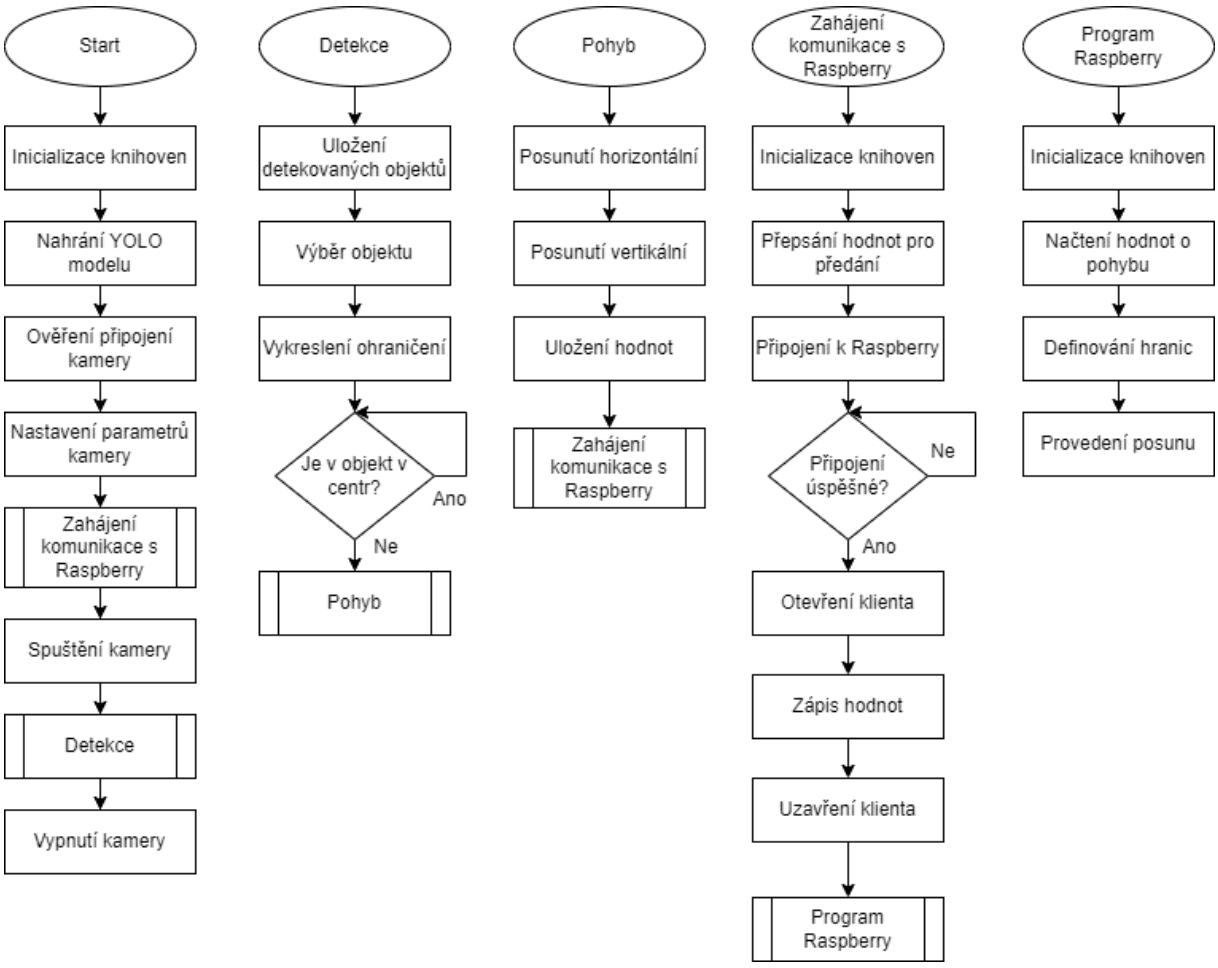


Obr. 9.7 – Výsledky důležitých metrik při trénování modelu YOLOv5s

Pro testování přesnosti detekce lze využít script v Colabu *val.py*, který po obdobném nastavení parametrů jako při trénování, otestuje model a výstupem jsou hodnoty metrik *mAP*. Z výsledků, které jsou viditelné v obrázku 9.7 je patrné, že už takto natrénovaný model je dostatečný pro reálné zařízení.

9.4 Popis funkčního programu

Program je rozdělen do několika fází. Obsahuje několik smyček, které na sebe navzájem navazují. Celkově je softwarové řešení rozděleno na 2 části. První část, která zahrnuje získání snímku i detekci a je spuštěna v programovém prostředí Thonny IDE, což je jazyk Python ve verzi 3.10.9. Druhá část je spuštěna a běží na Raspberry Pi 4B a obsahuje smyčky pro pohyb, zahájení komunikace přes sběrnici Modbus. Jedná se pouze o prototyp zařízení, takže spuštění obou programů je nezávislé a musí ho obsloužit uživatel.



Obr. 9.8 – Funkční diagram programu

9.4.1 Program pro PC

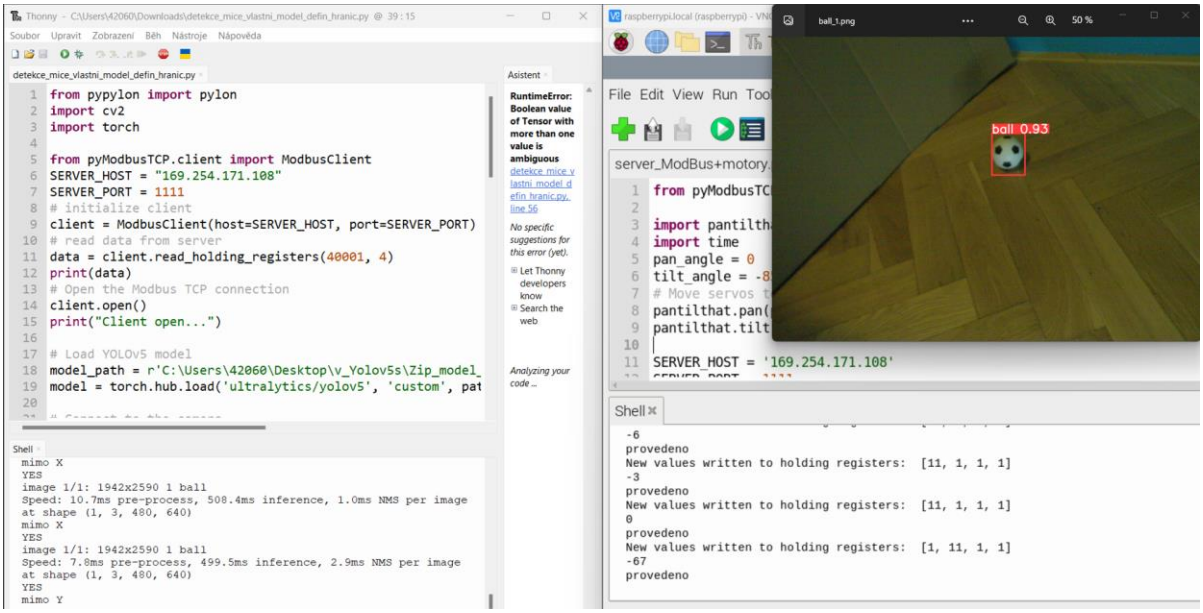
Program běžící na externím počítači funguje primárně pro obsloužení kamery a pro detekční činnost. Po inicializaci všech potřebných knihoven jako *PyPylon, OpenCV, PyTorch* a *PyModbusTCP* dojde i k otevření komunikace a nastavení s kamerou Basler. Dále je skrz cestu nahrán do pracovní plochy vytvořený model YOLOv5s a pokud vše proběhne úspěšně,

tak program skočí do smyčky *Detekce*. V této smyčce, pokud nedojde k odpojení kamery program setrvává a každý snímek vyhodnocuje na přítomnost objektu. Pro každý snímek jsou vypsány parametry jako velikost, počet detekovaných objektů, rychlost zpracování a pozice rámečku. Vybraný detekovaný objekt neboli jeho ohraničující rámeček je porovnávám s definovanými hranicemi středu zorného pole kamery. Pokud dojde k vyosení detekovaného objektu mimo střed, jsou v terminálu vypsány chybové hlášky a změněny registry pro Modbus.

9.4.2 Program pro Raspberry

V Raspberry nejprve samozřejmě musí také dojít k inicializaci používaných knihoven, a to *PyModbusTCP* a *Pimoroni*. Skrze knihovnu *PyModbusTCP* je vytvořen server se čtyřmi hodnotami registru, které slouží pro předávání informací o pohybu. Server je tedy vytvořen a spolu s tím je zahájena komunikace s klientem používaným v programu v PC. Tímto provázáním dochází k výměně informací. Pokud dojde ke změně alespoň jedné hodnoty registru, je v terminálu Raspberry vypsána hláška o změně. Dále je porovnávána nová hodnota změny a proveden patřičný posun. Aby nedošlo k zacyklení, jsou hodnoty pro pohyb přepsány do výchozí pozice a čeká se na nový snímek. Jelikož nový snímek je zpracován do 5 ms je nastaven i čas malého pozdržení, aby motory zvládly pohyb provést. Dále je vypisována horizontální i vertikální pozice, která je porovnávána s maximální hodnotou, aby nedošlo k poškození zařízení. Po spuštění je možnost volby mezi automatickým a manuálním řízením.

Příklad průběhu zpracování jednoho snímku je na obrázku níže.



Obr. 9.9 – Zpracování snímku z kamery

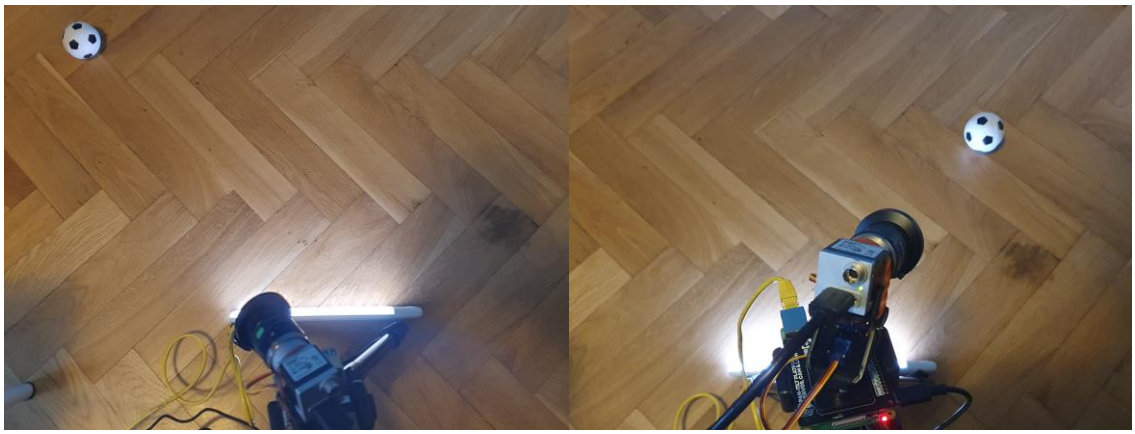
10 TESTOVÁNÍ FUNKCE SYSTÉMU

Ověření funkce systému probíhalo prakticky po celou dobu, neboť závislost jednotlivých funkcí by se projevila jako nefunkčnost celku. V první řadě model YOLO byl otestován jak pro detekci na statických datech, tedy na natočeném videu, tak i na jiné kameře, a to na webkameře notebooku. Při detekci na webkameru měl model malé problémy s rozlišením a občas chybně detekoval objekt. Nicméně na kameru Basler při normálních světelných podmínkách tento problém nebyl zpozorován. Pokud ale světelné podmínky byly opravdu špatné tak, model nedokázal objekt detekovat. To by mohlo vyřešit zvětšení datového setu o větší množství „tmavších“ fotek, anebo přidáním externího světelného zdroje.

Rychlost detekce na živých záběrech i díky tomu, že probíhala na jiném zařízení, než které obsluhovalo pohyb byla ovlivněna rychlostí pořízení fotografie, jejím zpracováním a předáním parametrů přes Modbus. Při této činnosti, pokud není potřeba uživateli fotografii s detekovaným objektem ukázat, byla rychlost více než dostatečná. Drobné zpomalení probíhá při pohybu servopohonů, nicméně vetší problém byl s rozmazanými objekty. Pokud se objekt pohyboval velmi rychle v blízkosti kamery, nejen že opustil perimetr zabíraný kamerou, ale také velmi často byl velmi rozmazán a nebyla možnost detekce. Zlepšení přineslo zvednutí zařízení a uložení jej do držáku stativu, viz obrázek 10.1. Do vzdálenosti dvou metrů od kamery byl objekt takřka vždy spolehlivě detekován a sledován.

Obr. 10.1 – Zařízení umístěné ve stativu

Pro pohyb do stran, tedy horizontální, je maximální úhel otočení 0 až 180 °. Počáteční hodnota je tedy nastavena na polovinu rozsahu, a tak na každou stranu je možné otočit kameru o 90 °. Pohyb je znázorněn na obrázku níže.



Obr. 10.2 – Horizontální ukázka pohybu

Pro vertikální pohyb je maximální úhel natočení 0 až 90 °, protože pokud byla kamera více nakloněna docházelo k velkému namáhání servopohonu a mohlo by dojít k jeho zničení. Ukázka pohybu je opět na obrázku níže.

Obr. 10.3 – Vertikální ukázka pohybu

Rychlost ustálení na statickou pozici ze středu na okraj zorného pole proběhne po první detekci do 3 sekund, ale občas bylo patrné drobné překmitnutí. Bohužel po nastavení drobnějšího kroku pro servomotory docházelo zase k delšímu běhu do ustálení.

11 ZÁVĚR

Tato diplomová práce se zaobírala vytvořením modelu pro detekci a sledování objektu v prostoru natáčením kamery. Nejprve byl představen základní princip neuronových sítí, blíže pak konvoluční neuronové sítě. Právě konvoluční neuronové sítě v posledních letech stojí za velkým vývojem umělé inteligence, potažmo strojového vidění. Dále byly popsány nejpoužívanější architektury pro klasifikaci a detekci objektů v obrazových datech. Spolu s nimi byly popsány také používané metriky pro určení přesnosti těchto modelů. V poslední části teoretické sekce byly popsány různé druhy a typy kamer s jejich využití v průmyslu.

V praktické části byly představeny jednotlivé komponenty, ze kterých se skládá systém pro automatické polohování. Poté bylo nutné vytvořit a analyzovat datovou sadu na které byl natrénován model určený k detekci objektu. Pro této účel byl nejvhodnější algoritmus pro detekci YOLOv5s. Pro trénování modelu byla využita platforma Google Colaboratory. Colab poskytuje výborný výkon vzhledem k jednoduchosti použití. Vytvořený model byl implementován do programového jazyka Python na standartní notebook a spolu s průmyslovou kamerou bylo možné vytvořit strojové vidění. Pro snímání obrazových dat byla pro tuto práci využita průmyslová kamera od společnosti Basler.

Dále bylo nutné vytvořit systém s dvěma stupni volnosti. To je zajišťováno počítačem Raspberry Pi 4 a zvolenou periférii se servopohony. Každý stupeň volnosti je tedy realizován jedním servomotorem. Použití standartní periférie má několik pozitiv i negativ. Pozitivum je určitě nízká pořizovací hodnota a jednoduchá programovatelnost. Jako největší negativum je asi malý kroutící moment servopohonů, a pokud by se jednalo o těžší kameru nemusely by servomotory stačit.

K propojením jednotlivých větších celků bylo nutné vytvořit datový tok informací. Předávání informací probíhá pomocí standartní sběrnice Modbus, pomocí ethernetového kabelu. Nastavení komunikace po sběrnici Modbus mezi počítači byla vcelku jednoduše realizovatelná.

Při vytváření prototypu byl kladen důraz na jednoduchost a možnost dalšího využití. Proto při výběru samotných součástek bylo vybíráno ze standartně dostupných komponent. Programování jak na straně notebooku, tak Raspberry bylo provedeno v Pythonu, který je také volně k dostání a knihovny obsažené v programech mají otevřený zdrojový kód. Takže dalšímu využití nic nebrání.

Výsledkem práce je zařízení, který zvládá pracovat s reálnými daty. Jedná se ale pouze o experiment určený k dalšímu vývoji. Proto nebyla vytvořena nějaká kompaktnější varianta. A co se týče například rychlosti pohybu by bylo nutné aplikovat silnější motory. Tento prototyp ale ukazuje, že je možné sestrojit zařízení využívající vlastnosti průmyslových kamer do praktického procesu.

POUŽITÁ LITERATURA

ALZUBAIDI, Laith, et al. Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. In: J Big Data [online] 2021, 8, 53 [cit. 2. 8. 2022]. ISSN 2196-1115. Dostupné z: https://doi.org/10.1186/s40537-021- 00444-8

About OpenCV, 2023. In*: OpenCV team [online]. [cit. 2023-05-12]*. Dostupné z: https://opencv.org/about/

BANDYOPADHYAY, Hmrishav, 2023. Autoencoders in Deep Learning: Tutorial & Use Cases. In: V7labs [online]. [cit. 2023-03-29]. Dostupné z: https://www.v7labs.com/blog/autoencoders-guide

BASLER AG., 2023. PyPylon [online]. In: . *[cit. 2023-05-12]. Dostupné* z: https://www.baslerweb.com/en/products/basler-pylon-camera-software-suite/pylon-open-source-projects/

BASLER AG. COMPANY PROFILE, 2023a. *Basler* [online]. [cit. 2023-05-12]. Dostupné z: https://www.baslerweb.com/en/company/about-us/

BASLER AG. COMPANY PROFILE, *2023b. Basler ace Series [online]. In:* . [cit. 2023-05-12]. Dostupné z: https://www.baslerweb.com/en/products/cameras/area-scan-cameras/ace/#productline=aceclassic

BASLER AG. COMPANY PROFILE, 2023c. Pylon Viewer [*online]. In: . [cit.* 2023-05-12]. Dostupné z: https://www.baslerweb.com/en/products/basler-pylon-camera-software-suite/pylon-viewer/

BOTLAND, 2023. *Raspberry Pi 4 model B WiFi DualBand Bluetooth 8 GB* [online]. [cit. 2023-05-12]. Dostupné z: https://botland.cz/moduly-a-sady-raspberry-pi-4b/16579-raspberry-pi-4-model-b-wifi-dualband-bluetooth-8-gb-ram-15-ghz-765756931199.html

Computer Vision: What this Fancy Term is All About *for Businesses, 2021. In: ITech* [online]. [cit. 2023-04-01]. Dostupné z: https://itechindia.co/blog/computer-vision-what-this-fancy-term-is-all-about-for-businesses/

Custom Training with YOLOv5, 2023. In: Colab [online]. Google Colaboratory [cit. 2023-05-15]. Dostupné z: https://colab.research.google.com/github/roboflow-ai/yolov5-custom-training-tutorial/blob/main/yolov5-custom-

training.ipynb?fbclid=IwAR1wKLHFFDxXrmze4o1VMblozYzidzjFf3XnN1vPGoqOEkwpAQV9h1NhWUM#scrollTo=eaFNnxLJbq4J

DOLEŽEL, Petr, 2016. Úvod do *umělých neuronových sítí pro studenty technických vysokých škol*. 1. Pardubice: Univerzita Pardubice. ISBN 978-80-7560-022-6.

DONGES, Niklas, 2022. A Guide to Recurrent Neural Networks: Understanding RNN and LSTM *Networks* [online]. Buildin [cit. 2023-03-23]. Dostupné z: https://builtin.com/data-science/recurrent-neural-networks-and-lstm

GALLAGHER, James, 2023. How to Use the Roboflow Bird Detection API. In: Roboflow [online]. [cit. 2023-04-01]. Dostupné z: https://blog.roboflow.com/bird-detection-api/

GANDHI, Rohith, 2018. R-CNN, Fast R-CNN, Faster R-CNN, YOLO — Object Detection Algorithms: *Understanding object detection algorithms* [online]. Towards Data Science [cit. 2023-04-15]. Dostupné z: https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e

HABRNÁL, Matěj, 2014. *HLUBOKÉ NEURONOVÉ SÍTĚ*. Brno. Dostupné také z: https://www.fit.vut.cz/study/thesis-file/14535/14535.pdf#page=58&zoom=100,130,442. Diplomová práce. VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ. Vedoucí práce Doc. Ing. FRANTIŠEK V. ZBOŘIL, CSc.

How single-shot detector (SSD) works? [online], 2023. Esri [cit. 2023-04-15]. Dostupné z: https://developers.arcgis.com/python/guide/how-ssd-works/#references

JIRKOVSKÝ, Jaroslav, 2018. Počítačové vidění: od hledání vzoru po Deep Learning. In: *Automa* [online]. Automa [cit. 2023-04-01]. Dostupné z: https://automa.cz/cz/casopis-clanky/pocitacove-videni-od-hledani-vzoru-po-deep-learning-2018_04_0_11425/

KUBÍNEK, Jiří, 2009. DETEKCE OBJEKTŮ V OBRAZE. Brno. *Dostupné* také z: https://www.vut.cz/www_base/zav_prace_soubor_verejne.php?file_id=116946. Diplomová práce. VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ. Vedoucí práce Ing. Michal Hradiš.

KUNAR, Pankaj, 2009. Real Time Target Tracking with *Pan Tilt Zoom Camera. In: DICK, Anthony* a Tan SHENG. Real Time Target Tracking with Pan Tilt Zoom Camera. Digital Image Computing: Techniques and Applications. ISBN 978-1-4244-5297-2. Dostupné také z: https://media.adelaide.edu.au/acvt/Publications/2009/2009-Real%20Time%20Target%20Tracking%20with%20Pan%20Tilt%20Zoom%20Camera.pdf

KUNDU, Rohit, 2023. *YOLO: Algorithm for Object Detection Explained* [online]. [cit. 2023-04-15]. Dostupné z: https://www.v7labs.com/blog/yolo-object-detection

LEFEBVRE, Loïc, 2023. PyModbusTCP Documentation [online]. [cit. 2023-05-18]. Dostupné z: https://pymodbustcp.readthedocs.io/en/latest/

LIN, Tsung-Yi, 2018. Focal Loss for Dense Object Detection [online]. [cit. 2023-04-15]. Dostupné z: https://arxiv.org/abs/1708.02002v2

MAŘÍK, Vladimír, Olga ŠTĚPÁNKOVÁ a Jiří LAŽANSKÝ, 1993. *Umělá inteligence*. Praha: Academia. ISBN 978-80-200-2276-9.

MING CHNG, Zhe, 2022. Google Colab for Machine Learning Projects. In: Machine Learning Mastery [online]. [cit. 2023-05-12]. Dostupné z: https://machinelearningmastery.com/google-colab-for-machine-learning-projects/

MLČOCH, Vladimír, 2012. *BEZPEČNOSTNÍ KAMEROVÝ SYSTÉM CCTV*. Brno. Bakalářská práce. VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ.

Neuron [online], 2009. Wikipedia [cit. 2023-03-21]. Dostupné z: https://cs.wikipedia.org/wiki/Neuron

NORRIS, Donald, 2015. Raspberry Pi: projekty. 1. vyd. Brno: Computer Press. ISBN 978-80-251-4346-9.

NOVÁK, Mirko, Josef FABER a Olga KUFUDAKI, 1993. *Neuronové sítě a informační systémy živých organismů*. Praha: Grada. ISBN 80-854-2495-9.

Pan-Tilt HAT, 2023. In: PIMORONI [online]. [cit. 2023-05-12]. Dostupné z: https://shop.pimoroni.com/products/pan-tilt-hat

POTRIMBA, Petru, 2023. What is a Convolutional Neural Network?. In: *Roboflow* [online]. [cit. 2023-04-01]. Dostupné z: https://blog.roboflow.com/what-is-a-convolutional-neural-network/

QUIZA, Ramon, 2011. Sample of a feed-forward neural network . In: ResearchGate [online]. [cit. 2023-03-23]. Dostupné z: https://www.researchgate.net/figure/Sample-of-a-feed-forward-neural-network_fig1_234055177

RAIS, Vítek, 2022. *Systém pro detekci* objektů v obrazových datech. Pardubice. Dostupné také z: https://dk.upce.cz/bitstream/handle/10195/80440/RaisV_SystemDetekci_PD_2022.pdf?seque

nce=1&isAllowed=y. Diplomová práce. Univerzita Pardubice. Vedoucí práce Doc. Ing. Petr Doležel, Ph. D.

TAN, Lu, Tianran HUANGFU, Liyao WU a Wenying CHEN, 2021. Comparison of RetinaNet, SSD, and YOLO v3 for real-time pill identification. In: *BMC* [online]. [cit. 2023-05-12]. Dostupné z: https://bmcmedinformdecismak.biomedcentral.com/articles/10.1186/s12911-021-01691-8

Thonny Python IDE for beginners, 2023. In: Thonny [online]. [cit. 2023-05-12]. Dostupné z: https://thonny.org/

UPTON, Eben a Gareth HALFACREE, 2016. Raspberry Pi: uživatelská příručka. *2.*, aktualizované vydání. Přeložil Jakub GONER. Brno: Computer Press. ISBN 978-80-251-4819-8.

VALENTA, Vít, 2021. *Kamery se sledováním pohybu* [online]. [cit. 2023-04-18]. Dostupné z: https://magazin.disk.cz/cs/kamery-se-sledovanim-pohybu

WALIA, Mrinal, 2022. Semantic Segmentation vs. Instance Segmentation. In: Roboflow [online]. [cit. 2023-04-01]. Dostupné z: https://blog.roboflow.com/difference-semantic-segmentation-instance-segmentation/

What is Python? Executive Summary, 2023. In: *Python* [online]. Python Software Foundation [cit. 2023-05-12]. Dostupné z: https://www.python.org/doc/essays/blurb/

What is the Modbus Protocol & How Does It Work?, 2023. In: NATIONAL INSTRUMENTS CORP. [online]. [cit. 2023-05-12]. Dostupné z: https://www.ni.com/cs-cz/shop/seamlessly-connect-to-third-party-devices-and-supervisory-system/the-modbus-protocol-in-depth.html

YASAR, Kinza, 2022. PyTorch: What is PyTorch?. In: TechTarget [online]. [cit. 2023-05-12]. Dostupné z: https://www.techtarget.com/searchenterpriseai/definition/PyTorch

ZELINKA, Ivan, 2005. Umělá inteligence, aneb, *Úvod do neuronových sítí, evolučních algoritmů*--. Vyd. 2. Zlín: Univerzita Tomáše Bati ve Zlíně. ISBN 80-731-8277-7.

PŘÍLOHY

A – CD

Příloha A

Příloha k diplomové práci

Automatické sledování objektu v prostoru

Petr Stibor

CD

Obsah

1 Text diplomové práce ve formátu PDF.

2 Úplný zdrojový kód pro PC.

3 Úplný zdrojový kód pro Raspberry Pi 4B.

4 Natrénovaný model YOLOv5s.