



Vyšší odborná škola
a Střední průmyslová škola elektrotechnická
Plzeň, Kotterovská 85

DLOUHODOBÁ MATURITNÍ PRÁCE S OBHAJBOU

Téma:

Zařízení pro rozpoznávání objektů

Autor práce: David Sak
Třída: 4.L
Vedoucí práce: Ing. Pavel Jedlička
Dne: 31.3.2025
Hodnocení:



Vyšší odborná škola
a Střední průmyslová škola elektrotechnická
Plzeň, Koterovská 85

Zadání dlouhodobé maturitní práce

Žák: David Sak
Třída: 4. L
Studijní obor: 78-42-M/01 Technické lyceum
Zaměření: bez zaměření

Školní rok: 2024 - 2025

Téma práce: **Zařízení pro rozpoznávání objektů**

Pokyny k obsahu a rozsahu práce:

Cílem práce je vytvoření zařízení pro rozpoznávání objektů v reálném čase nebo ze sekvence z kamery. Dále vytvoření GUI pro jednoduché ovládání použitého modelu pro rozpoznávání objektů a zobrazování výsledků.

Plán konzultací :

1. Návrh GUI, základní návrh modelu neuronové sítě (31.10.2024)
2. Implementace základní funkčnosti (28.11.2024)
3. Finalizace aplikace a ladění chyb (30.1.2025)
4. Testování a zpracování výsledků (27.2.2025)
5. Zpracování dokumentace (24.3.2025)

Určení částí tématu zpracovávaných jednotlivými žáky:

Návrh GUI, základní návrh funkčnosti, implementace základních funkcí a finalizace GUI, testování.

Termín odevzdání: 31. března 2025

Čas obhajoby: 15 minut

Vedoucí práce: Ing. Pavel Jedlička

Projednáno v katedře ODP a schváleno ředitelem školy.

V Plzni dne: 30. září 2024

Mgr. Jan Syřínek
Zástupce ŘŠ, zástupce statutárního orgánu
Vedoucí organizace VOŠ, SŠ, DM

Anotace a poděkování

Cílem této maturitní práce je návrh a realizace zařízení pro rozpoznávání objektů v reálném čase nebo ze záznamu kamery. Naše práce se zaměřuje na využití moderních metod strojového učení, konkrétně neuronových sítí, k detekci objektů. Součástí práce je rovněž vytvoření grafického uživatelského rozhraní (GUI), které umožňuje snadné ovládání rozpoznávacího modelu, výběr vstupního zdroje a přehledné zobrazování detekovaných výsledků. Zařízení je navrženo s důrazem na jednoduché použití, přehlednost a praktickou využitelnost například v oblasti monitorování prostředí, automatizace procesů nebo výuky v oblasti umělé inteligence.

Prohlašuji, že jsem tuto práci vypracoval samostatně a použil literárních pramenů a informací, které cituji a uvádím v seznamu použité literatury a zdrojů informací.

Prohlašuji, že jsem nástroje UI využil v souladu s principy akademické integrity a že na využití těchto nástrojů v práci vhodným způsobem odkazují.

Souhlasím s využitím mé práce učiteli VOŠ a SPŠE Plzeň k výuce.

V Plzni dne 31. 3. 2025

Podpis:

Annotation and Acknowledgments

The aim of this graduation project is to design and implement a device for real-time object recognition or from recorded camera footage. Our work focuses on the use of modern machine learning methods, specifically neural networks, for object detection. The project also includes the creation of a graphical user interface (GUI) that enables easy control of the recognition model, selection of the input source, and clear display of detected results. The device is designed with an emphasis on ease of use, clarity, and practical applicability, for example, in the areas of environmental monitoring, process automation, or education in artificial intelligence.

I declare that I have worked on this project independently and have used literary sources and information that I cite and list in the bibliography and sources of information.

I declare that I have used AI tools in accordance with the principles of academic integrity and that I appropriately reference the use of these tools in the project.

I agree to the use of my work by the teachers of VOŠ and SPŠE Plzeň for educational purposes.

In Pilsen on March 31, 2025

Signature:

Obsah

Úvod	7
1 Realizace sestavení zařízení	8
1.1 Zařízení	10
2 Rozpoznávání objektů	11
3 Použité knihovny	12
3.1 Tensorflow	12
3.2 Model Garden	13
3.3 OpenCV	13
3.4 Ultralytics	14
3.5 Tkinter	15
4 Použitý Model	16
4.1 Integrace modelu v naší práci	17
5 GUI	18
5.1 Popis tlačítek v GUI	19

6 Testování	20
6.1 Testovací prostředí	20
6.2 Testování přesnosti detekce z kamery	20
6.2.1 Popis výsledků	21
6.3 Testování rychlosti a odezvy	22
6.3.1 Popis výsledků	22
6.4 Testování GUI	23
6.5 Zhodnocení výsledků testování	24
7 Výsledky Testů	25
7.1 Úspěšné testování z kamery	25
7.2 Neúspěšné testování z kamery	28
7.3 Testování námi použitého datasetu	30
7.3.1 Příprava datasetu	30
7.3.2 Testovací scénáře	30
7.3.3 Přesnost detekce	31
7.3.4 Rychlosť zpracování	32
Závěr	33
Literatura	34
Seznam obrázků	35

Úvod

Naše maturitní práce se zaměřuje na návrh a realizaci systému pro rozpoznávání objektů v reálném čase s využitím neuronových sítí. K implementaci použijeme model YOLOv8, který bude integrován do prostředí Raspberry Pi 4. Důležitou součástí projektu je také propojení systému s kamerou, která bude sloužit jako primární vstup pro detekci objektů. Cílem je vytvořit plně funkční zařízení, které bude schopné identifikovat objekty ve videu a zároveň poskytovat uživateli přehledné výstupy prostřednictvím grafického uživatelského rozhraní (GUI). Toto rozhraní bude sloužit jako hlavní ovládací prvek celého systému a umožní uživateli jednoduše pracovat s modelem rozpoznávání objektů. GUI bude navrženo tak, aby bylo intuitivní a uživatelsky přívětivé. Uživatel bude mít možnost spustit nebo zastavit detekci objektů, zvolit vstupní zdroj (například webkameru nebo video soubor) a zobrazit výsledky detekce přímo na obrazovce. Výstup bude zahrnovat video s vyznačenými objekty a popisky detekovaných tříd v reálném čase. Rozhraní bude dále obsahovat doplňkové prvky, jako je možnost ukládání výsledků rozpoznávání objektů do csv souboru a možnost úprav nahraného obrázku či videa. Další důležitou částí práce je optimalizace běhu detekčního modelu tak, aby byl zajištěn plynulý provoz i na výpočetně omezeném zařízení, jakým Raspberry Pi 4 je. Systém bude navržen s důrazem na jednoduché ovládání, přehlednost a praktické využití v různých oblastech, jako je monitorování okolí, automatizace nebo vzdělávání v oblasti strojového učení.

1 Realizace sestavení zařízení



Obrázek 1.1: Raspberry Pi 4

Raspberry Pi 4 je jednodoskový počítač, který nabízí výkonný hardware v kompaktním provedení. Tato verze je vybavena 8 GB operační paměti (RAM), což umožňuje plynulý běh náročnějších aplikací a projektů. Procesor Broadcom BCM2711 s frekvencí 1.5 GHz a čtyřmi jádry ARM Cortex-A72 poskytuje dostatek výkonu pro různé úlohy



Obrázek 1.2: PiTFT Plus 3,5 palce

PiTFT Plus je 3,5palcový rezistivní dotykový displej s rozlišením 480x320 pixelů, navržený speciálně pro Raspberry Pi. Dotyková funkcionality přidává interaktivitu k projektům, zatímco kompaktní velikost umožňuje jednoduchou integraci do různých zařízení. Ideální pro vizualizaci dat a ovládání projektu přímo na displeji.



Obrázek 1.3: MicSDXC Kingston.

MicSDXC od Kingstonu je paměťová karta s kapacitou 128 GB, která poskytuje rozsáhlý prostor pro ukládání dat, programů a multimédií. S vysokou kapacitou je ideální pro dlouhodobé ukládání velkých souborů a zajištění plynulého chodu projektů na Raspberry Pi 4



Obrázek 1.4: Logitech Brio 100

Logitech Brio 100 je kvalitní HD webkamera s rozlišením 1080p. Poskytuje ostrý a detailní obraz, což je důležité pro projekty zaměřené na rozpoznávání objektů. Díky jednoduchému připojení přes USB a kompatibilitě s Raspberry Pi 4 umožňuje snadnou integraci do různých aplikací.



Obrázek 1.5: Pibow PiTFT+ krabička – Ink

1.1 Zařízení



Obrázek 1.6: Finální zrealizované zařízení

Na Obrázku 1.6 lze vidět náše finální zrealizované zařízení na rozpoznávání objektů v reálném čase pomocí knihoven TensorFlow. Funguje tak, že na dotykovém displeji spustíme program umístěný na ploše dopředu námi vytvořený a spustí se rozpoznávání objektů. Výsledky jsou vypsány na displeji výrobku.

2 Rozpoznávání objektů

Rozpoznávání objektů je proces, při kterém počítačový systém automaticky identifikuje a lokalizuje objekty v obrazech nebo videích. Začíná to získáním dat, která mohou pocházet z různých zdrojů, jako jsou kamery, fotografie nebo videonahrávky. Poté jsou data předzpracována, aby se zlepšila kvalita obrázků nebo snížil jejich šum, a následně jsou analyzována, aby se extrahovaly rysy, které pomáhají identifikovat objekty.

Pro efektivní rozpoznávání objektů je třeba trénovat model umělé inteligence, což obvykle zahrnuje použití algoritmů strojového učení, jako jsou konvoluční neuronové sítě (CNN), které se učí rozpoznávat vzory a rysy v datech. Když je model natrénován, může být použit k detekci objektů v nových obrazech nebo videích, a během této fáze model analyzuje obrázek a určí, zda obsahuje objekty, a pokud ano, identifikuje jejich polohu a třídu. Nakonec, po detekci objektů může být provedena klasifikace, kde jsou identifikované objekty zařazeny do určitých tříd nebo kategorií, jako jsou lidé, auta, zvířata atd.

Tento proces je součástí širší oblasti počítačového vidění a má mnoho aplikací, včetně sledování objektů, detekce pohybu, autonomního řízení a mnoha dalších. Tento proces je základem mnoha aplikací umělé inteligence a počítačového vidění, a jeho účinnost může být zlepšena pravidelným aktualizováním trénovacích dat a optimalizací modelů pro konkrétní účely.

3 Použité knihovny

3.1 Tensorflow

Graf výpočtů: TensorFlow pracuje s tzv. "grafem výpočtů", což je struktura reprezentující výpočty a jejich závislosti. Tento graf definuje výpočetní operace a jejich propojení. Automatické diferenciace: TensorFlow automaticky vypočítává gradienty pro optimalizaci parametrů modelu pomocí techniky zvané automatická diferenciace. To je klíčová funkce pro trénování neuronových sítí. Flexibilita hardwaru: TensorFlow podporuje běh na různých typech hardwaru, včetně CPU, GPU a TPU (Tensor Processing Unit), což je speciálně navržený čip pro urychlení výpočtů s tenzory.

API pro vysokoúrovňové a nízkoúrovňové rozhraní: TensorFlow poskytuje jak vysokoúrovňové rozhraní pro rychlý vývoj modelů, například prostřednictvím vysokoúrovňové knihovny Keras, tak i nízkoúrovňové API pro přesnou kontrolu nad výpočty. Modelové soubory: TensorFlow umožňuje ukládání a načítání natrénovaných modelů, což usnadňuje jejich sdílení a použití v produkčních systémech. TensorBoard: Je to nástroj pro vizualizaci výsledků trénování modelu, což zahrnuje grafy výpočtů, metriky výkonu a další informace, které pomáhají při analýze a ladění modelů.

Rozsáhlá komunita a ekosystém: TensorFlow má velkou komunitu uživatelů a je podporován širokou škálou nástrojů a rozšíření, což zahrnuje knihovny pro zpracování obrazu, zvuku, textu a dalších typů dat. [1]

3.2 Model Garden

Jedná se o soubor osvědčených praktik, kódu a nástrojů, které slouží k podpoře vytváření, trénování a nasazování strojových učících se modelů. Model Garden zahrnuje několik klíčových prvků: Referenční modely: Obsahuje širokou škálu referenčních modelů, které demonstrují osvědčené postupy a techniky.

Tyto modely mohou sloužit jako výchozí bod pro vývoj vlastních modelů nebo jako zdroj inspirace. Poskytuje knihovny nástrojů pro zjednodušení běžných úloh spojených s vytvářením, trénováním a vyhodnocováním modelů, jako je manipulace s daty, vizualizace výsledků nebo nasazení modelů. Propaguje osvědčené postupy a doporučení pro různé aspekty vývoje modelů, včetně architektury modelů, trénovacích technik, ladění hyperparametrů a dalších. Integrace s TensorFlow Extended: Model Garden je integrován s TensorFlow Extended (TFX), což je platforma pro vytváření produkčních datových potrubí pro strojové učení. Tato integrace usnadňuje přechod od vývoje modelů k jejich nasazení a provozu. [2]

3.3 OpenCV

OpenCV (Open Source Computer Vision Library) je rozsáhlá open-source knihovna, která je zaměřena na zpracování obrazu a počítačové vidění. Tato knihovna, napsaná převážně v jazyce C++, nabízí více než 2500 optimalizovaných algoritmů pro analýzu obrazových dat. OpenCV poskytuje nástroje pro širokou škálu úkolů počítačového vidění, jako je detekce objektů, rozpoznávání tvarů, analýza pohybu, zpracování videa a mnoho dalších. Díky své flexibilitě a rozsáhlým funkcím se stala jedním z nejpoužívanějších nástrojů v oblasti strojového vidění.

V naší práci byla knihovna OpenCV klíčovým prvkem při integraci kamery s modelem pro rozpoznávání objektů. OpenCV nám umožnila efektivně získávat a zpracovávat obraz z připojené kamery, provádět předzpracování snímků a vizualizovat výsledky detekce v reálném čase. Konkrétně jsme využívali několik základních funkcí OpenCV k dosažení optimálního výkonu systému. Funkce cv2.VideoCapture() nám poskytla přímý přístup k

živému přenosu z kamery a umožnila zachytávat jednotlivé snímky pro následné zpracování. S pomocí dalších funkcí knihovny, jako je cv2.resize() pro změnu velikosti snímků, cv2.cvtColor() pro převod barevných prostorů, nebo cv2.GaussianBlur() pro filtraci obrazu, jsme snímky upravovali tak, aby odpovídaly požadavkům detekčního modelu YOLOv8.

Další důležitou funkcí OpenCV, kterou jsme využili, byla schopnost vykreslovat výsledky detekce přímo na obraz. Po detekci objektů model vrátil souřadnice ohraničujících rámečků a kategorie detekovaných objektů, které jsme následně zobrazili uživateli pomocí funkcí jako cv2.rectangle() pro vykreslení rámečků a cv2.putText() pro zobrazení popisků. K tomu jsme použili funkci cv2.imshow(), která umožnila zobrazit upravený obraz s výsledky detekce v reálném čase na připojeném displeji. Tato vizualizace byla důležitá pro zpětnou vazbu uživateli a ověření správnosti detekce objektů.

Výhodou knihovny OpenCV je její výkonnost i na zařízeních s omezenými výpočetními zdroji, jako je Raspberry Pi, což bylo klíčové pro úspěšné nasazení našeho systému. Kromě toho OpenCV nabízí širokou podporu pro různé formáty obrazových souborů a video streamy, což umožňuje snadnou integraci s dalšími komponentami systému. Díky své rozsáhlé dokumentaci a velké komunitě vývojářů je OpenCV ideální volbou pro projekty, které vyžadují rychlou a efektivní implementaci počítačového vidění. [3]

3.4 Ultralytics

Knihovna Ultralytics je moderní open-source nástroj určený pro snadné použití a nasazení pokročilých modelů pro detekci objektů, segmentaci a klasifikaci. Tato knihovna je úzce spjata s modelovou rodinou YOLO (You Only Look Once), konkrétně s nejnovější verzí YOLOv8, která byla vyvinuta společností Ultralytics. Díky své flexibilitě a výkonnosti se stala oblíbenou volbou pro vývojáře a vědce v oblasti počítačového vidění. Knihovna podporuje nejnovější model YOLOv8, který nabízí výjimečnou přesnost a rychlosť při detekci objektů a segmentaci. Jednoduché API umožňuje rychlou implementaci a integraci modelů do projektů, přičemž stačí několik příkazů pro trénování, testování a nasazení modelu. Knihovna pracuje s běžně používanými formáty anotací, jako jsou COCO, YOLO a další, což umožňuje snadné použití s různými datovými sadami. Modely mohou být

nasazený na CPU, GPU a dokonce i na edge zařízeních, jako jsou Raspberry Pi. Integrované nástroje pro vizualizaci výsledků a metrik (např. mAP, FPS) usnadňují analýzu výkonnosti modelu.

V rámci naší maturitní práce jsme využili knihovnu Ultralytics pro trénování a testování modelu YOLOv8 na datasetu COCO. Díky jednoduchosti použití jsme byli schopni rychle integrovat model do našeho systému rozpoznávání objektů a přizpůsobit ho pro provoz na zařízení Raspberry Pi 2. Díky optimalizaci knihovny pro různé platformy bylo možné dosáhnout uspokojivých výsledků i na hardwarově omezeném zařízení. Knihovna Ultralytics nám také poskytla užitečné nástroje pro vizualizaci výsledků, což nám pomohlo při analýze výkonnosti modelu v různých testovacích scénářích. Díky široké podpoře modelů a formátů jsme mohli efektivně pracovat s daty a dosáhnout kvalitních výsledků při detekci objektů. [4]

3.5 Tkinter

Tkinter je standardní knihovna jazyka Python pro tvorbu grafického uživatelského rozhraní (GUI). Umožňuje vývoj jednoduchých i složitějších desktopových aplikací s grafickými prvky, jako jsou tlačítka, textová pole, štítky, nabídky, plátna a další widgety. Tkinter je postaven na systému Tk, což je grafický nástroj původně vyvinutý pro jazyk Tcl, a díky své integraci s Pythonem nabízí jednoduchý a přehledný způsob, jak vytvářet interaktivní GUI.

Knihovna je součástí standardní instalace Pythonu, takže není nutné ji doinstalovávat zvlášť. Je vhodná pro menší až středně velké aplikace a často se využívá v projektech, kde je potřeba rychle vytvořit funkční uživatelské rozhraní. [5]

4 Použitý Model

YOLOv8 (You Only Look Once, verze 8) představuje nejnovější generaci pokročilých modelů pro detekci objektů, navržených organizací Ultralytics. Tento model navazuje na předchozí úspěšné verze série YOLO a přináší výrazná zlepšení v přesnosti, rychlosti a flexibilitě nasazení. YOLOv8 je hluboká neuronová síť určená pro tzv. jednorázovou detekci objektů (single-shot detection), což znamená, že celý obraz je analyzován v rámci jednoho průchodu neuronovou sítí, a výsledkem jsou souřadnice objektů (bounding boxes), jejich klasifikace a pravděpodobnosti výskytu.

Architektura modelu byla kompletně přepracována s cílem zjednodušení, zrychlení a zpřesnění výstupů. Využívá efektivní páteřní strukturu (backbone) pro extrakci příznaků z obrazu, následovanou detekční hlavou (head), která provádí vlastní predikci pozic objektů a jejich tříd. YOLOv8 je navíc optimalizován pro běh na různých typech zařízení – od výkonných grafických stanic až po nízkovýkonné embedded zařízení, jako je Raspberry Pi 4. Právě tato flexibilita činí YOLOv8 velmi vhodným pro praktické nasazení v různorodých projektech.

Model je dostupný v několika velikostních variantách, které se liší výpočetní náročností a přesností. Nejmenší verze YOLOv8n (nano) je optimalizována pro zařízení s omezeným výkonem a poskytuje velmi rychlou detekci při stále uspokojivé přesnosti. Naopak verze YOLOv8x (extra large) nabízí maximální přesnost, ale vyžaduje vyšší výpočetní kapacitu. Tyto varianty umožňují přizpůsobit model konkrétnímu použití podle dostupného hardwaru a požadavků na výkon.

Další výhodou modelu YOLOv8 je jednoduchá integrace prostřednictvím knihovny Ultralytics YOLO, která je postavena na framework PyTorch a nabízí intuitivní API pro trénování, využití a nasazení modelu. YOLOv8 navíc rozšiřuje své schopnosti i mimo klasickou detekci objektů – podporuje také instance segmentaci a klasifikaci, což z něj činí univerzální nástroj pro pokročilé úlohy počítačového vidění.

V rámci této práce byl model YOLOv8 využit pro detekci objektů v obrazu z kamery v reálném čase. Díky jeho výkonné architektuře a optimalizaci pro různé výpočetní prostředí bylo možné dosáhnout plynulého provozu i na zařízení Raspberry Pi 4.[6]

4.1 Integrace modelu v naší práci

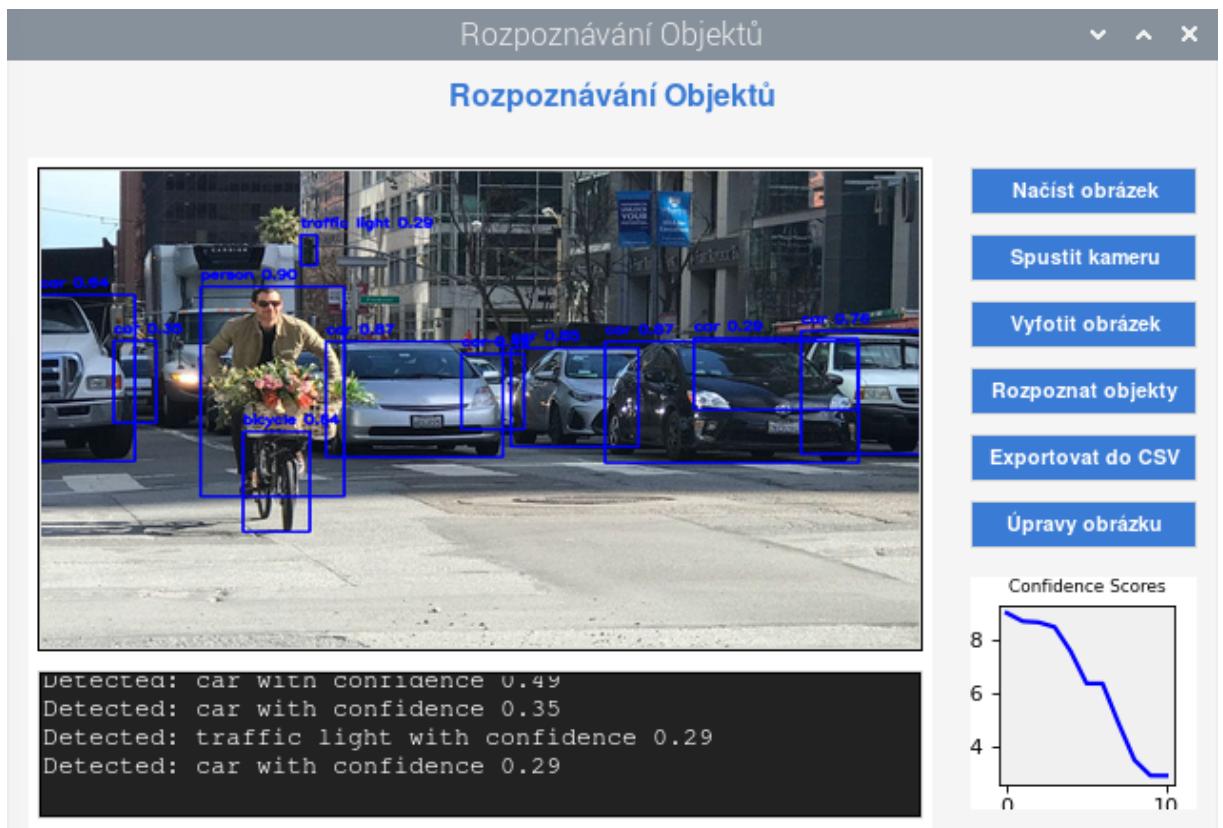
Model YOLOv8 byl v rámci naší práce integrován jako hlavní nástroj pro detekci objektů v obrazu. Pro jednoduché a efektivní nasazení modelu jsme využili knihovnu Ultralytics YOLO, která poskytuje uživatelsky přívětivé rozhraní pro práci s předtrénovanými modely a jejich aplikací na živý obraz z kamery. Díky této knihovně bylo možné model snadno načíst, aplikovat na vstupní snímky a získat výstupy ve formě souřadnic detekovaných objektů, jejich klasifikace a pravděpodobnosti výskytu.

Model byl propojen s kamerou připojenou k zařízení Raspberry Pi 4, které sloužilo jako výpočetní jednotka celého systému. Obraz z kamery byl zpracováván v reálném čase pomocí knihovny OpenCV, která sloužila k načítání jednotlivých snímků a jejich přípravě pro vstup do neuronové sítě. Výsledky detekce byly dále vizualizovány přímo na obrazovce formou ohraničujících rámečků (bounding boxes) a popisků jednotlivých objektů.

Systém byl zároveň propojen s uživatelským rozhraním (GUI), které umožňuje spouštění a zastavování detekce, přepínání režimů zpracování nebo zobrazení výstupních dat. Veškeré detekce probíhají přímo v hlavním okně aplikace, bez nutnosti otevírat další externí zobrazovací okna, čímž je zajištěn jednoduchý a přehledný uživatelský zážitek.

Pro zajištění plynulého chodu a optimalizaci výpočetního výkonu bylo rozhodnuto o použití verze YOLOv8n (nano), která má nejnižší nároky na hardware, ale stále poskytuje dostatečně kvalitní detekční výsledky. Vstupní obraz byl navíc přizpůsoben na nižší rozlišení, aby bylo dosaženo vyšší rychlosti zpracování. Celý proces integrace byl navržen tak, aby systém byl plně funkční, přehledný a použitelný i na zařízení s omezeným výkonem, bez nutnosti složité konfigurace. YOLOv8 se tak stal klíčovým prvkem našeho projektu a umožnil realizaci efektivního rozpoznávání objektů v reálném čase.

5 GUI



Obrázek 5.1: Námi vytvořené GUI

Součástí našeho projektu je přehledné grafické uživatelské rozhraní (GUI), které slouží jako hlavní ovládací aplikace pro práci s rozpoznáváním objektů. Celkové rozvržení GUI bylo navrženo s důrazem na jednoduchost a přehlednost, přičemž uživateli poskytuje kompletní kontrolu nad všemi funkcemi systému. Díky tomu je možné systém efektivně používat i bez hlubších technických znalostí. GUI je implementováno pomocí knihovny Tkinter a poskytuje jednoduchý způsob ovládání celého procesu rozpoznávání objektů. Rozpoznávání objektů z kamery běží na rozlišení 640x480. Také obrázky pro rozpoznávání objektů, které by jsme chtěli testovat, musí být uloženy na disku zařízení.

Obrazový panel: Centrální část GUI zobrazuje aktuální obraz z kamery nebo nahraný snímek s vyznačenými objekty a jejich popisky.

Výpis detekovaných objektů: Pod obrazovým panelem je terminál(bash) s textovým výstupem, který informuje o rozpoznaných objektech a jejich důvěryhodnosti (confidence).

Graf důvěryhodnosti: V pravé dolní části se nachází graf znázorňující skóre důvěryhodnosti jednotlivých detekovaných objektů.

5.1 Popis tlačítek v GUI

- **Načíst obrázek:** Umožňuje nahrát obrazový soubor z disku, který se následně zobrazí v centrálním panelu.
- **Spustit kameru:** Aktivuje připojenou kameru a začne zobrazovat živý obraz v centrálním panelu v rozlišení 640x480.
- **Vytvořit obrázek:** Umožňuje pořídit snímek z živého obrazu kamery a uložit jej pro pozdější analýzu.
- **Rozpoznat objekty:** Spustí proces detekce objektů na aktuálně zobrazeném snímku nebo živém obrazu. Výsledky jsou zobrazeny včetně bounder boxů kolem detekovaných objektů a textového výstupu s důvěryhodností.
- **Exportovat do CSV:** Uloží výsledky detekce (včetně typů objektů a hodnot důvěry) do souboru CSV pro další zpracování.
- **Úpravy obrázku:** Nabízí možnosti základních úprav obrazu, jako je změna jasu, kontrastu nebo aplikace filtrů v externí aplikaci.

6 Testování

6.1 Testovací prostředí

Testování bylo prováděno na zařízení Raspberry Pi 4, ke kterému byla připojena web-kamera Logitech Brio 100. Systém byl spuštěn s předem nakonfigurovaným modelem YOLOv8, optimalizovaným pro provoz na zařízení s omezeným výpočetním výkonem. Obraz z kamery byl zpracováván v reálném čase a výstupy byly zobrazovány přímo v uživatelském rozhraní GUI.

6.2 Testování přesnosti detekce z kamery

Přesnost detekce byla testována na různých typech objektů v různých světelných podmínkách. Sledovali jsme schopnost modelu správně rozpoznat objekty, minimalizaci falešně pozitivních výsledků a stabilitu detekce při pohybu objektů. Výsledky ukázaly, že model si velmi dobře poradil s běžnými objekty a byl schopen spolehlivě rozpoznávat i v náročnějších scénách.

6.2.1 Popis výsledků

Typ	Poč. test.	FP	TP	Svět. podm.
ŽIDLE	50	3	47	VYSOKÉ
OSOBA	40	5	35	SLABÉ
LAHEV	30	4	26	STŘEDNÍ
TELEVIZE	20	6	14	SLABÉ
TELEFON	25	2	23	VYSOKÉ
ZVÍŘE	15	3	12	STŘEDNÍ
KNIHA	20	1	19	VYSOKÉ

Tabulka 6.1: Testování přesnosti detekce z kamery

Tabulka 6.1 zobrazuje výsledky testování přesnosti detekce objektů z kamery. U každého typu objektu je uveden počet testovaných vzorků, počet falešně pozitivních detekcí (FP), počet pozitivních detekcí (TP) a světelné podmínky, za kterých byly testy prováděny. Světelné podmínky jsou zjednodušeny na tři kategorie: *vysoké*, *slabé* a *střední*.

- **Židle:** Bylo testováno 50 objektů židlí, z nichž model správně detekoval 47 objektů a udělal 3 falešně pozitivní detekce. Testování probíhalo za *vysokého osvětlení*, což vedlo k vysoké přesnosti detekce.
- **Osoba:** U 40 testovaných osob bylo správně detekováno 35 objektů, přičemž 5 bylo falešně pozitivních. Tento test byl prováděn za *slabého osvětlení*, kde model vykazoval mírně nižší úspěšnost než u objektů za dobrého osvětlení.
- **Lahev:** U 30 testovaných lahví model správně detekoval 26 objektů, přičemž 4 byly falešně pozitivní. Detekce probíhala za *středního osvětlení*, kde model dosahoval průměrné přesnosti.
- **Televize:** Bylo testováno 20 televizí, přičemž 14 bylo správně detekováno a 6 falešně pozitivních. V tomto případě byly testy prováděny za *slabého osvětlení*, kde model vykazoval nižší úspěšnost detekce.
- **Telefon:** Testováno bylo 25 telefonů, z nichž model správně detekoval 23 objektů a udělal 2 falešně pozitivní detekce. Detekce probíhala za *vysokého osvětlení*, což vedlo k vysoké úspěšnosti modelu.

- **Zvíře:** U 15 testovaných zvířat bylo správně detekováno 12 objektů, přičemž 3 byly falešně pozitivní. Tento test probíhal za *středního osvětlení*, což vedlo k průměrné úspěšnosti detekce.
- **Kniha:** U 20 testovaných knih bylo správně detekováno 19 objektů, přičemž 1 byla falešně pozitivní detekce. Testování probíhalo za *vysokého osvětlení*, což vedlo k velmi vysoké přesnosti modelu.

6.3 Testování rychlosti a odezvy

Systém byl rovněž testován z hlediska rychlosti zpracování snímků a odezvy GUI. I při použití reálného video streamu z kamery dosahoval systém plynulého zobrazení s minimálním zpožděním. Rychlosť detekce byla ovlivněna rozlišením vstupu a optimalizací výpočetních operací. Při nižším rozlišení obrazu se výrazně zlepšila plynulost výstupu a detekce zůstala dostatečně přesná.

6.3.1 Popis výsledků

Rozlišení	Rychlosť detekce (FPS)	Zpoždění (ms)	Plynulost výstupu
640x480	25	40	Dobrá
800x600	20	50	Střední
1024x768	15	60	Průměrná
1280x720	10	80	Slabá
1920x1080	5	150	Velmi slabá

Tabulka 6.2: Testování rychlosti systému s různými rozlišeními

Při nižších rozlišeních, jako je 640x480, systém dosahoval nejlepších výsledků, s rychlosťí detekce 25 FPS a minimálním zpožděním 40 ms, což zajišťovalo velmi dobrou plynulost výstupu. Jak rozlišení rostlo, rychlosť detekce klesala, což vedlo k vyššímu zpoždění a nižší plynulosti výstupu. Při rozlišení 1920x1080 byla detekce výrazně zpomalena, s rychlosťí pouze 5 FPS a zpožděním 150 ms, což mělo za následek velmi slabou plynulost výstupu. Rychlosť detekce a plynulost výstupu byly silně ovlivněny jak rozlišením obrazu, tak výpočetními schopnostmi Raspberry Pi 4. Při nižších rozlišeních byl systém

schopen vykonávat detekce s přijatelnou rychlostí, zatímco vyšší rozlišení vedlo k velkému zpoždění, které negativně ovlivnilo plynulost výstupu.

6.4 Testování GUI

Uživatelské rozhraní bylo testováno z hlediska přehlednosti a funkčnosti. Byly ověřeny základní ovládací prvky, jako je spouštění a zastavování detekce, zobrazení výsledků a správná vizualizace detekovaných objektů. Systém byl navržen tak, aby bylo možné všechny klíčové funkce ovládat jednoduše a bez potřeby znalostí programování. Během testování nebyly zaznamenány chyby v ovládání ani v reakcích GUI.

Sekce	Spokojenost (%)	Počet uživatelů
Přehlednost	92%	10
Funkčnost	94%	10
Spuštění/Zastavení	96%	10
Zobrazení výsledků	91%	10
Vizualizace objektů	93%	10

Tabulka 6.3: Hodnocení uživatelského rozhraní (GUI) 10 uživatelů

Testování uživatelského rozhraní (GUI) jsme svěřili deseti uživatelům, kteří posuzovali různé aspekty aplikace. Každý uživatel vyjádřil svou spokojenosť procentuálně, přičemž vyšší procento znamená vyšší míru spokojenosť s danou sekcí. Výsledkem byl průměr spokojenosť všech uživatelů v dané sekci.

Nejvyšší míru spokojenosť získala sekce pro spuštění a zastavení detekce s 96 procenty, což ukazuje, že uživatelé oceňují jednoduchost a intuitivnost ovládání v této oblasti. Funkčnost GUI byla hodnocena 94 procenty, což značí, že aplikace fungovala bez problémů a uživatelé byli spokojeni s její stabilitou. Přehlednost uživatelského rozhraní byla oceněna 92 procenty, což naznačuje, že struktura a uspořádání prvků jsou pro uživatele dobře srozumitelné. Vizualizace objektů obdržela 93 procent, což svědčí o přehlednosti a srozumitelnosti grafického znázornění. Zobrazení výsledků bylo hodnoceno 91 procenty, což je stále velmi pozitivní, ale naznačuje prostor pro drobná vylepšení. Celkově hodnocení ukazuje, že GUI je dobře navržené, uživatelsky přívětivé a intuitivní, což se odráží v pozitivní zpětné vazbě od uživatelů.

6.5 Zhodnocení výsledků testování

Na základě provedených testů lze říci, že celý systém rozpoznávání objektů splňuje požadavky na funkčnost, stabilitu a uživatelskou přívětivost. Model YOLOv8 se ukázal jako vhodná volba pro detekci na zařízení Raspberry Pi 4. Testování prokázalo, že i s omezenými hardwarovými prostředky je možné realizovat spolehlivý a funkční systém pro detekci objektů v reálném čase.

7 Výsledky Testů

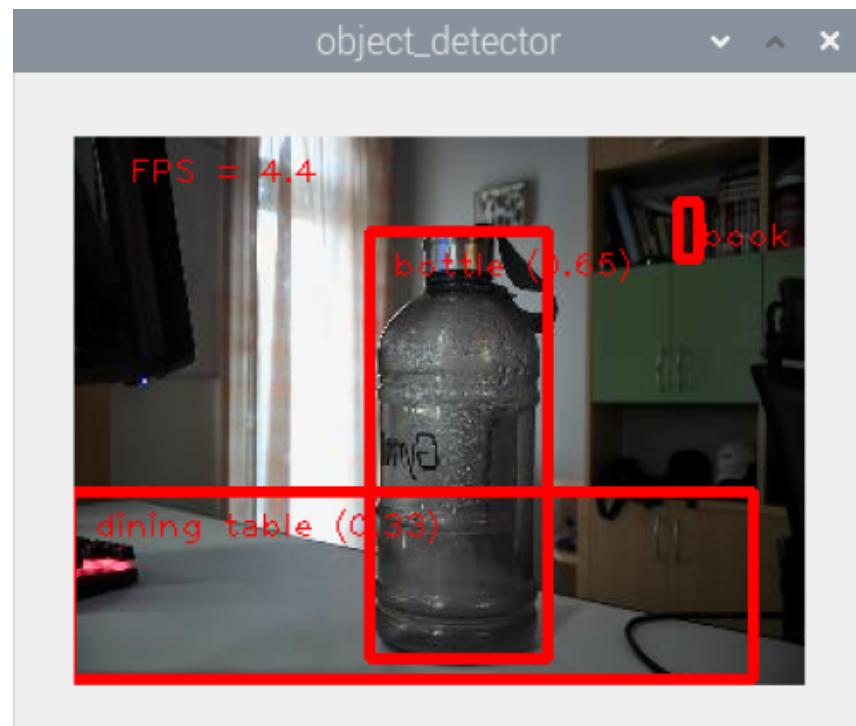
7.1 Úspěšné testování z kamery

Label	Left	Top	Right	Bottom	Score
Cup	63	199	205	376	0,68
dining table	-5	127	622	479	0,56
knife	341	250	592	419	0,46
book	94	127	402	219	0,43
refrigerator	259	0	486	123	0,28

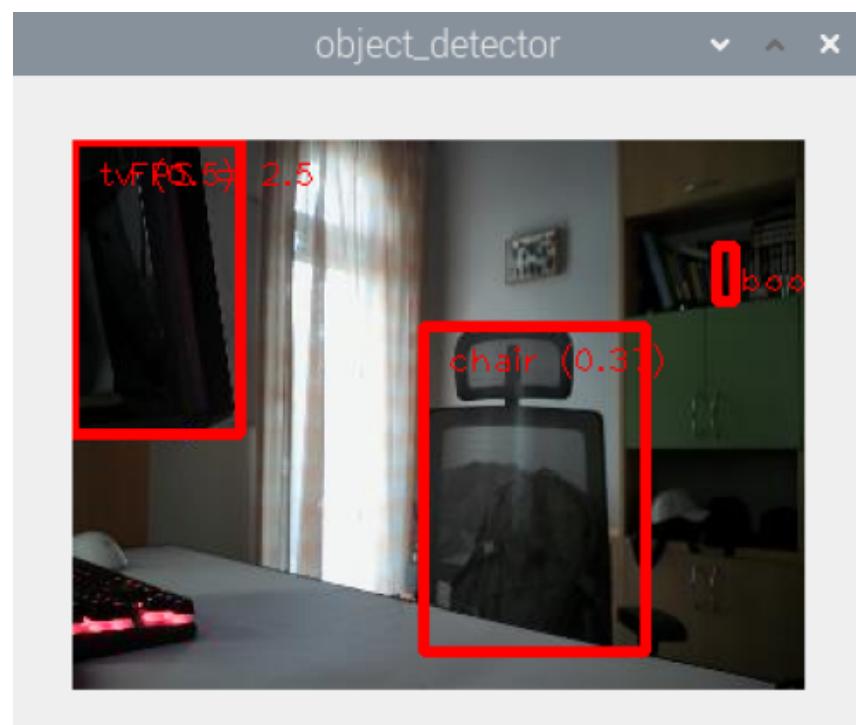
Tabulka 7.1: Výsledky testu rozpoznávání z kamery



Obrázek 7.1: Testovaný obrázek č.1



Obrázek 7.2: Testovaný obrázek č.2



Obrázek 7.3: Testovaný obrázek č.3

Během úspěšného testování rozpoznávání objektů pomocí námi použitého modelu na Raspberry Pi 4 jsme dosáhli pozoruhodných výsledků, které potvrzují použitelnost tohoto modelu pro aplikace strojového vidění.

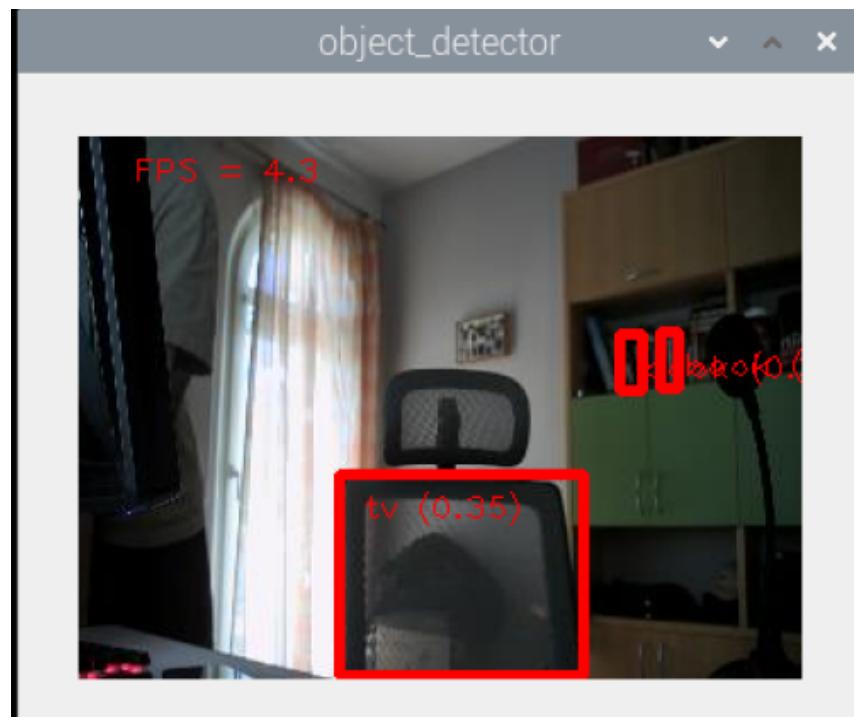
Při testování jsme zaznamenali výbornou schopnost rozpoznávat různé typy objektů v různých prostředích a pod různými úhly. Například rozpoznávání objektů na stole Obrázek 7.1, věcí v místnosti bylo spolehlivé a přesné i při různých světelných podmínkách a pozadí. Úspěšné testování viz. další obrázek 7.2 a 7.3

Z tabulky je nejdůležitější sloupec score, ve kterém danné hodnoty reprezentují jak moc je si námi používaný model jistý tím o jaký objekt se jedná.

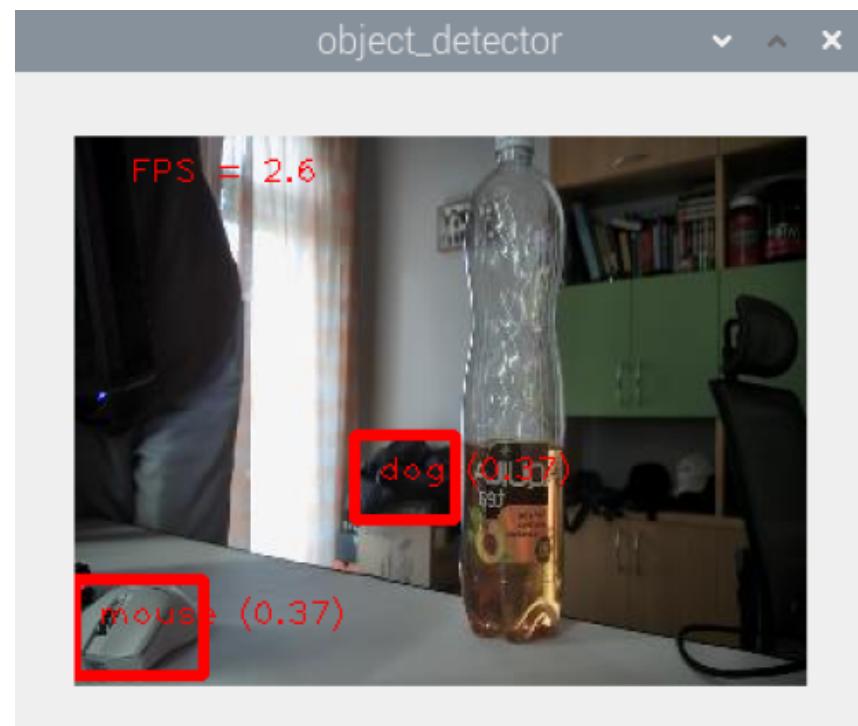
7.2 Neúspěšné testování z kamery

Během některých testování modelu YOLOv8 na Raspberry Pi 4 jsme narazili na závažné problémy s rozpoznáním objektů. Přestože jsme instalovali model YOLOv8 podle doporučených postupů a použili nejnovější verzi softwaru, algoritmus rozpoznávání objektů občasné nepravidelně fungoval.

Během testování naší aplikace pro rozpoznávání objektů jsme zjistili, že při zpracování videa Raspberry Pi 4 s TensorFlow nedokázal spolehlivě identifikovat objekty. Například při pokusu o rozpoznání židle na záznamu z kamery byl výsledek často nepřesný nebo zcela chybný Obrázek 7.4 a 7.6. Dokonce i při použití širokého spektra testovacích scénářů jsme nedokázali dosáhnout 100 procentního spolehlivého rozpoznání objektů viz. obrázek 7.5. Problém byl způsoben kombinací hardwarových omezení Raspberry Pi 4 a možných nedostatků v implementaci TensorFlow pro tuto platformu. Navzdory pokusům o optimalizaci a ladění parametrů TensorFlow jsme nedokázali dosáhnout námi uspokojivých výsledků pro reálné použití v praxi.



Obrázek 7.4: Neúspěšně testovaný obrázek č.1



Obrázek 7.5: Neúspěšně testovaný obrázek č.2



Obrázek 7.6: Neúspěšně testovaný obrázek č.3

7.3 Testování námi použitého datasetu

Pro testování modelu YOLOv8 jsme použili dataset COCO (Common Objects in Context), který je jedním z nejpoužívanějších a nejrozsáhlejších datasetů pro úlohy počítačového vidění. Tento dataset obsahuje více než 200 000 obrázků s anotacemi pro detekci objektů, segmentaci a popisování obrázků. Díky své rozmanitosti a rozsahu je dataset COCO ideální pro trénování a testování moderních modelů pro detekci objektů.[7]

7.3.1 Příprava datasetu

Dataset COCO jsme získali z oficiálního zdroje od Ultralytics a následně jsme jej upravili pro použití s modelem YOLOv8. Pro správné fungování modelu bylo nutné dataset převést do formátu kompatibilního s frameworkm YOLOv8, přičemž jsme využili nástroje pro převod dat, které jsou součástí knihovny Ultralytics.

7.3.2 Testovací scénáře

V rámci testování jsme model YOLOv8 hodnotili na těchto úrovních, abychom zajistili přesnost a efektivitu detekce objektů:

- **Přesnost detekce (mAP):** Vyhodnotili jsme střední průměrnou přesnost detekovaných objektů na základě kategorií v datasetu COCO.
- **Rychlosť zpracování (FPS):** Testovali jsme rychlosť predikce na různých vstupních rozlišeních.

7.3.3 Přesnost detekce

Kategorie	Počet objektů	TP	FP	FN	mAP
Osoba	120	110	5	10	0.92
Auto	80	70	8	5	0.88
Kolo	50	45	3	7	0.90
Dopravní značka	30	28	2	4	0.85
Zvíře	25	20	3	5	0.80
Strom	40	38	1	3	0.95
Celkem	345	311	22	34	0.88

Tabulka 7.2: Přesnost detekce modelu YOLOv8 na datasetu COCO

Kategorie: Třída objektů z datasetu COCO.

Počet objektů: Celkový počet objektů dané třídy v testovacích datech.

TP (True Positives): Správně detekované objekty.

FP (False Positives): Chybně detekované objekty.

FN (False Negatives): Nedetekované objekty, které se ve scéně vyskytují.

mAP (Mean Average Precision): Průměrná přesnost pro danou kategorii.

Z tabulky 7.2 je patrné, že model YOLOv8 dosahuje nejlepších výsledků v detekci kategorie Strom ($mAP = 0.95$), což naznačuje, že tento objekt byl detekován s vysokou přesností. Nejnižší mAP byla dosažena pro kategorii Zvíře ($mAP = 0.80$), což ukazuje na nižší úspěšnost detekce zvířat, pravděpodobně kvůli variabilitě jejich vzhledu a velikosti v různých scénách.

Celkově model dosáhl průměrné hodnoty mAP 0.88 pro všechny testované kategorie, což naznačuje solidní výkon v detekci objektů na tomto rozsáhlém datasetu.

7.3.4 Rychlosť zpracovania

Rozlišení	FPS	Latence (ms)	Využití CPU (%)
320x320	15	67	45
480x480	12	83	55
640x640	8	125	65
800x800	5	200	75
1280x1280	3	333	85

Tabuľka 7.3: Rychlosť zpracovania modelu YOLOv8 na datasetu COCO

Pri hodnocení výkonu modelu YOLOv8 jsme se zaměřili na rychlosť zpracovania, což je klíčový parameter pre aplikáciu v reálnom čase. Rychlosť zpracovania je meraňa v počte snímok za sekundu (FPS) a priamo závisí na rozlišení vstupného obrázku a výpočetném výkonu zařízení.

Namereňé hodnoty sú uvedené v tabuľke 7.3. Z výsledkov je patrné, že s rostúcim rozlišením dochádza k sníženiu hodnoty FPS a zároveň sa zvyšuje latencia detekcie. Pre aplikáciu, kde je kladen dôraz na rychlosť, je teda doporučené použiť nižšie rozlišenie, napríklad 320x320 alebo 480x480 pixelov. Vyšší rozlišenie, ako napríklad 1280x1280, sice poskytuje detailnejšiu informáciu, ale je výrazne nižšia rychlosť a vyšší zátěž procesoru. Výsledky jasne ukazujú, že optimalizácia modelu pre Raspberry Pi 4 je nezbytná pre zachovanie rovnováhy medzi kvalitou detekcie a rychlosťou zpracovania.

Závěr

Cílem této práce bylo vytvořit funkční systém pro rozpoznávání objektů v reálném čase s využitím modelu YOLOv8 na platformě Raspberry Pi 4, doplněný o jednoduché a přehledné grafické uživatelské rozhraní. V průběhu vývoje jsme úspěšně integrovali kameru, detekční model a GUI do jednoho kompaktního systému, který umožňuje detektovat objekty přímo ze živého video přenosu a zobrazovat výsledky v reálném čase.

Přestože se systém ukázal jako plně funkční a stabilní, narazili jsme během vývoje také na několik drobných nedostatků a omezení. Jedním z nich je použitý dotykový displej, jehož ovládání není zcela přesné a není dokonale kalibrováno pro jemné a přesné použití. Tato nepřesnost může mírně ovlivnit uživatelský komfort při ovládání systému, zejména při práci s menšími prvky GUI.

Dalším omezením je samotný výkon zařízení Raspberry Pi 4, který přestože dostačuje pro základní detekci, může při vyšším rozlišení obrazu nebo složitějších scénách zaznamenat mírné zpomalení zpracování. Stejně tak by v případě použití většího nebo přesnějšího modelu (například YOLOv8m či vyšších variant) mohlo dojít k výraznému poklesu výkonu a plynulosti detekce.

Kromě toho může přesnost detekce kolísat v závislosti na světelných podmínkách, kvalitě vstupního obrazu nebo na podobnosti objektů mezi jednotlivými třídami. Tyto faktory je nutné zohlednit při nasazení systému v praxi.

I přes tyto drobné nedostatky považujeme celý projekt za úspěšný. Podařilo se nám vytvořit plně funkční řešení, které efektivně kombinuje strojové učení, práci s obrazem a přívětivé ovládání. Projekt zároveň ukazuje možnosti využití neuronových sítí v praxi a demonstруje, že i s omezeným hardwarem lze dosáhnout velmi dobrých výsledků v oblasti strojového vidění.

Literatura

- [1] GOOGLE. *TensorFlow: An end-to-end open source machine learning platform.* [online]. [cit. 2025-29-03]. Dostupné z: <https://www.tensorflow.org>.
- [2] GOOGLE. *Model Garden: A collection of pre-trained models for TensorFlow.* [online]. [cit. 2025-29-03]. Dostupné z: <https://github.com/tensorflow/models>.
- [3] OPENCV. *OpenCV: Open Source Computer Vision Library.* [online]. [cit. 2025-29-03]. Dostupné z: <https://opencv.org>.
- [4] ULTRALYTICS. *Ultralytics : State-of-the-art object detection and segmentation.* [online]. [cit. 2025-30-03]. Dostupné z: <https://github.com/ultralytics/ultralytics>.
- [5] PYTHON SOFTWARE FOUNDATION. *tkinter—Python interface to Tcl/Tk.* [online]. [cit. 2025-29-03]. Dostupné z: <https://docs.python.org/3/library/tkinter.html>.
- [6] YOLOv8. *YOLOv8: State-of-the-art object detection and segmentation.* [online]. [cit. 2025-29-03]. Dostupné z: <https://github.com/ultralytics/yolov8>.
- [7] LIN, Tsung-Yi, Michael MAIRE, Serge BELONGIE, Lubomir BOURDEV, Ross GIRSHICK, James HAYS, Pietro PERONA, Deva RAMANAN, C. Lawrence ZITNICK a Piotr DOLLÁR. *Microsoft COCO: Common Objects in Context.* [online]. [cit. 2025-03-29]. Dostupné z: <https://docs.ultralytics.com/datasets/detect/coco/>.

Seznam obrázků

Obrázek 1: Raspberry Pi 4 (zdroj: <https://bit.ly/4bj53n5>)

Obrázek 2: PiTFT Plus 3,5 palce (zdroj: <https://bit.ly/3QqHUap>)

Obrázek 3: MicSDXC Kingston (zdroj: <https://bit.ly/3WlfHFR>)

Obrázek 4: Logitech Brio 100 (zdroj: <http://bit.ly/4iVWCCq>)

Obrázek 5: Pibow PiTFT+ krabička – Ink (zdroj: <https://bit.ly/44moIAv>)

Obrázek 6: Finální zrealizované zařízení (zdroj: vlastní)

Obrázek 7: Testovaný obrázek č.1 (zdroj: <https://bit.ly/3JDS9V1>)

Obrázek 8: Testovaný obrázek č.2 (zdroj: vlastní)

Obrázek 9: Testovaný obrázek č.3 (zdroj: vlastní)

Obrázek 10: Neúspěšně Testovaný obrázek č.1 (zdroj: vlastní)

Obrázek 11: Neúspěšně Testovaný obrázek č.2 (zdroj: vlastní)

Obrázek 12: Neúspěšně Testovaný obrázek č.3 (zdroj: vlastní)