STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST

Obor č. 18: informatika

RATT - Platforma pro analýzu chování hlodavců poháněnou umělou inteligencí

Diego Alejandro Portillo Abad
Pardubický kraj

Pardubice, 2025

STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST

Obor č. 18: informatika

RATT - Platforma pro analýzu chování hlodavců poháněnou umělou inteligencí
RATT - Platform for AI-powered rodent behavior analysis

jméno: Diego Alejandro Portillo Abad
škola: DELTA - střední škola informatiky a ekonomie, s.r.o.
kraj: Pardubický kraj
konzultant: Ing. Richard Cimler, Ph.D.

Pardubice, 2025

prohlášení

prohlašuji, že jsem svou práci soč vypracoval/a samostatně a použil/a jsem pouze prameny a literaturu uvedené v seznamu bibliografických záznamů.

prohlašuji, že tištěná verze a elektronická verze soutěžní práce soč jsou shodné.

nemám závažný důvod proti zpřístupňování této práce v souladu se zákonem č. 121/2000 sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších předpisů.

v Pardubicích dne 26. března 2025 .....

Diego Alejandro Portillo Abad

Poděkování

Rád bych poděkoval Ing. Richardu Cimlerovi, Ph.D. a Přírodovědecké fakultě Univerzity Hradec Králové za podporu a pomoc při rozvoji tohoto projektu.

Abstrakt

Tato prace představuje nový nástroj řízený umělou inteligencí pro analýzu a sledování pohybu myší ve video sekvenci a také klec navrženou pro neustálé sledování myší. Využití konvoluční neuronové sítě Yolov11 trénované na vlastní datové sadě přizpůsobené speciálně pro tento úkol. Diskutujeme o koordinaci analytických úloh prostřednictvím fronty zpráv NATS a orchestraci mnoha služeb, které tvoří projekt. Výsledná platforma umožňuje uživatelům přístup k sofistikovaným analytickým nástrojům, které by pro ně byly za normálních okolností nedostupné, data získaná z analyzovaných videí mohou vědci využít k posílení svého výzkumu.

Klíčová slova

hluboké učení, architektura mikroslužeb, kontejnerizace, Raspberry Pi

Abstract

This report presents a novel AI-driven tool for analysing and tracking the movement of mice in a video sequence, as well as a cage designed for constant monitoring of mice. Leveraging a Yolov11 convolutional neural network trained on a custom dataset tailored specifically for this task. We discuss the coordination of analysis jobs through a NATS messaging queue, and the orchestration of the many services that make up the project. The resulting platform allows users to access sophisticated analysis tools that would normally be inaccessible to them, the data resulting from the analysed videos can be used by scientists to boost their research.

Keywords

deep learning, microserivice architecture, conteinarization, Raspberry Pi

Seznam zkratek a nazvy

Zkratka	Význam
AI	Umělá inteligence
API	Aplikační programové rozhraní
AWS	Amazon Web Services
CNN	Konvoluční neuronová síť
DHT22	Senzor teploty a vlhkosti
FFMPEG	Nástroj pro zpracování multimédií
FFMPROBE	Nástroj pro analýzu multimediálních souborů
FVTP	Festival vědy a techniky pro děti a mládež v Pardubickém kraji
GEM	Geneticky modifikované modely
Go	Programovací jazyk Go
HC-SR04	Ultrazvukový senzor vzdálenosti
HTTP	Hypertext Transfer Protocol
IDE	Integrované vývojové prostředí
LCD	Displej z tekutých krystalů
LSTM	Dlouhá krátkodobá paměť
NATS	Fronta zpráv NATS
NoIR	Bez infračerveného filtru
ORM	Objektově relační mapování
Pi	Raspberry Pi
ReLU	Rectified Linear Unit
S3	Simple Storage Service
SDK	Software Development Kit
SQL	Structured Query Language
SSH	Secure Shell
TPU	Tensor Processing Unit

Obsah

1 Úvod

Hlodavci jsou nepostradatelní v biomedicínském výzkumu, slouží jako primární model pro studium lidské fyziologie, mechanismů onemocnění a terapeutických zásahů. Jejich genetická podobnost s lidmi, krátké reprodukční cykly a přizpůsobivost kontrolovaným prostředím je činí obzvláště cennými v preklinickém výzkumu, přičemž více než 95% studií na zvířatech se provádí na myších. Schopnost vytvářet geneticky modifikované modely (GEM) transformovala modelování nemocí, umožňující přesnou manipulaci s geny zapojenými do rakoviny, metabolických poruch a neurodegenerativních onemocnění. Tyto modely, v kombinaci s pokročilými zobrazovacími a molekulárními nástroji, poskytují poznatky, které překlenují základní výzkum a klinický překlad, a vedou objevování léků a regulační schvalovací procesy.

Behaviorální výzkum zůstává základním kamenem neurovědy a psychologie, protože chování je nejintegrovanějším a funkčně relevantním výstupem nervové aktivity. Etologie, zakořeněná v Darwinových principech a později zdokonalená Lorenzem, Tinbergenem a von Frischem, formovala naše chápání instinktů, učení a sociálních interakcí. Behaviorální testy jsou klíčové v neuropsychiatrickém výzkumu, pomáhají dekódovat kognitivní procesy, afektivní stavy a mechanismy rozhodování. Kromě klinických aplikací má chování hluboké implikace v oblastech tak rozmanitých, jako je interakce člověka s počítačem, vojenská strategie a umělá inteligence, kde jsou rozsáhle studovány modely detekce hrozeb, spolupráce a adaptace. Propojení etologie a aplikovaných věd zdůrazňuje význam behaviorálních studií i mimo tradiční laboratorní prostředí.

Ačkoli je analýza chování s pomocí umělé inteligence stále v rané fázi, má potenciál tuto oblast revolučně změnit, poskytuje přesnější, vysoce propustná a objektivní data v podstatně kratším čase. Objevující se algoritmy strojového učení začínají mapovat komplexní behaviorální fenotypy, detekovat mikroexprese a odvozovat motivační stavy s úrovní přesnosti nedosažitelnou tradičními metodami. Jak se tyto technologie budou rozvíjet, nejenže zlepší reprodukovatelnost a škálovatelnost behaviorálních studií, ale také usnadní nové objevy v neurovědě, psychologii a biomedicínském výzkumu. Integrace umělé inteligence do behaviorální vědy představuje posun k efektivnějším, datově řízeným výzkumným metodikám, které zpřesní naše chápání kognice, emocí a sociálních interakcí napříč druhy.

Tradiční metody studia chování hlodavců jsou často časově náročné, náchylné k zkreslení a postrádají reprodukovatelnost. Kromě toho jsou behaviorální data často nedostatečně využívána kvůli nevhodným nástrojům pro zpracování, což zdůrazňuje potřebu integrované platformy, která umožňuje monitorování a analýzu v reálném čase. Mnohá současná softwarová řešení jsou neúměrně drahá, což je činí nedostupnými pro malé výzkumné týmy a majitele domácích zvířat. Tato finanční bariéra omezuje široké přijetí nástrojů pro analýzu chování, což zdůrazňuje poptávku po cenově dostupném a uživa-

telsky přívětivém řešení. EthoVision XT[2] je široce uznáván pro svůj robustní výkon a rozsáhlé funkce, jeho vysoká cena, neintuitivní rozhraní a náročný proces nastavení mohou omezit jeho dostupnost – zejména pro menší laboratoře nebo nezávislé výzkumníky. Na druhou stranu open source alternativy jako ToxTrac[1] poskytují nákladově efektivní řešení s působivými sledovacími schopnostmi; často však vyžadují solidní znalosti IT a technické know-how pro efektivní instalaci, kalibraci a odstraňování problémů.

Tento projekt řeší tuto potřebu prostřednictvím webového rozhraní pro analýzu chování hlodavců, kde si uživatelé mohou vytvořit účty a nahrát vlastní videa k analýze. Analýza probíhá na pozadí a uživatelé budou mít přístup k analyzovaným výsledkům na vyžádání, jakmile budou k dispozici. Projekt také uživatelům poskytuje speciálně navrženou klec pro nepřetržité monitorování a analýzu, kterou mohou připojit ke svému účtu. Klec bude neustále odesílat video útržky z klece a nahrávat je na platformu. Analýza se opět provádí na pozadí a uživatelé mají přístup k analyzovaným výsledkům a údajům ze senzorů z klece, jako je teplota a světlo. Výsledkem je dostupné a nákladově efektivní řešení, které zajišťuje neustálý a spolehlivý dohled nad hlodavci pro výzkumné i chovatelské účely.

2 Aktuální řešení

Oblast analýzy chování hlodavců je v současnosti obsluhována kombinací komerčních a open-source řešení, z nichž každé má své silné a slabé stránky. Komerční software, jako je EthoVision XT, nabízí komplexní sadu nástrojů pro sledování a analýzu chování zvířat, poskytuje robustní funkce, jako je detailní sledování pohybu, sofistikovaná detekce chování a rozsáhlé generování datových výstupů. Významná finanční investice, kterou vyžaduje, spolu se složitostí její konfigurace však vytváří překážku pro vstup menších výzkumných týmů a jednotlivých výzkumníků. Kromě toho se specializovaná komerční řešení zaměřují na specifické aspekty analýzy chování, jako je sociální interakce nebo spánkové vzorce, ale ta jsou často vysoce specializovaná a mají vysokou cenu. Na druhou stranu open-source alternativy, jako je ToxTrac, poskytují nákladově efektivní řešení se základními možnostmi sledování pohybu, nabízejí výhodu bezplatné dostupnosti a přizpůsobení. Tyto nástroje však často vyžadují určitou úroveň technické zdatnosti pro instalaci a odstraňování problémů, což omezuje jejich dostupnost pro ty, kteří nemají silné IT dovednosti.

Navzdory dostupnosti těchto nástrojů zůstávají na trhu významné mezery. Projekt RATT si klade za cíl řešit tyto nedostatky tím, že poskytuje cenově dostupnou a uživatelsky přívětivou platformu, která integruje hardware i software. Primárním problémem stávajících řešení jsou neúměrné náklady na komerční možnosti, které vylučují mnoho menších výzkumných skupin a nezávislých výzkumníků. RATT se snaží demokratizovat přístup k těmto technologiím tím, že nabízí cenově výhodnější open-source alternativu. Mnoho současných platforem je navíc zatíženo složitými konfiguracemi, které vyžadují značné technické znalosti. RATT upřednostňuje snadné použití a navrhuje intuitivní rozhraní, které zjednodušuje nastavení a provoz. Klíčovou inovací projektu RATT je jeho holistický přístup, který kombinuje videoanalýzu s klecí navrženou na míru pro nepřetržité monitorování a sběr dat, což je funkce, která ve stávajících řešeních do značné míry chybí. Tato integrace umožňuje komplexnější pochopení chování hlodavců. A konečně, RATT si klade za cíl překlenout mezeru v dostupných nástrojích pro analýzu dat tím, že poskytuje sofistikované analýzy řízené umělou inteligencí, které jsou obecně vyhrazeny pro větší, dobře financované výzkumné instituce.

V podstatě projekt RATT přispívá demokratizací přístupu k pokročilým analytickým nástrojům, usnadňuje výzkum prostřednictvím bezproblémové integrace hardwaru a softwaru, zajišťuje cenovou dostupnost pro širší publikum a upřednostňuje uživatelskou přívětivost. Tím, že RATT řeší tyto kritické mezery, umožňuje výzkumníkům i majitelům domácích zvířat provádět efektivnější a poučnější studie, což v konečném důsledku posouvá naše chápání chování hlodavců jak ve vědeckém, tak v domácím kontextu.

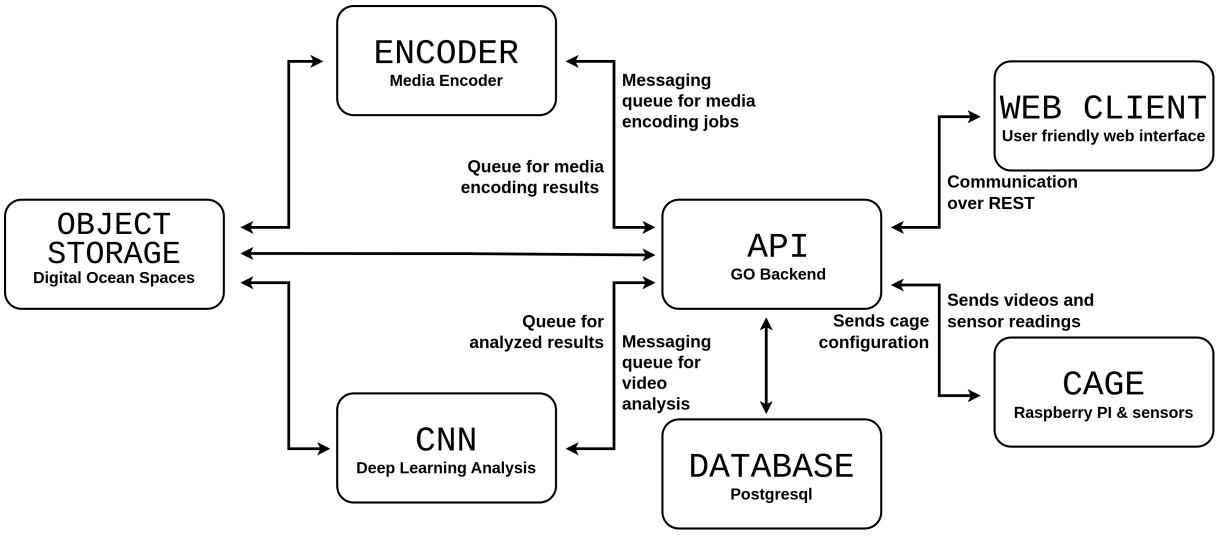
3 Architektura

Tento projekt implementuje komplexní systém pro automatizované monitorování zvířat, využívající architekturu mikroslužeb k zajištění škálovatelnosti a udržovatelnosti. Systém se skládá z několika klíčových komponent:

- **Hlavní API:** fungující jako centrální orchestrátor, zpracovává autentizaci uživatelů, správu dat a delegování úkolů;
- **Služba Hlubokého Učení:** poháněná Pythonem a frameworky hlubokého učení, zpracovává video data pro analýzu chování zvířat;
- **Služba Kódování:** napsaná v Go, optimalizuje video kodeky pro konzistentní přehrávání;
- **Klecový klient:** založený na Raspberry Pi, shromažďuje data ze senzorů a zaznamenává video v prostoru pro zvířata;
- **Webový klient:** poskytující uživatelské rozhraní pro vizualizaci dat a interakci se systémem.

Tyto komponenty komunikují prostřednictvím fronty zpráv NATS a využívají databázi PostgreSQL pro strukturované ukládání dat, přičemž mediální soubory jsou uloženy v DigitalOcean Spaces. Tato architektura umožňuje bezproblémovou integraci hardwarových a softwarových komponent, což umožňuje robustní a efektivní monitorování zvířat.

Architektura mikroslužeb byla zvolena jako požadovaná praxe, kterou je třeba dodržovat. Rozdělením projektu na menší, samostatné služby můžeme izolovat chyby, rozdělit kódové základny, libovolně kombinovat technologie a izolovat data [3]. Tato volba byla motivována především možností horizontálního rozšiřování služeb na vyžádání, zejména pokud by v systému existovala nějaká úzká hrdla. To se ukázalo jako užitečné při začleňování fronty zpráv do platformy, protože to umožnilo vyvážit zatížení náročných úkolů do více služeb, aby mohly být zpracovávány paralelně. V následujících částech bude každá služba podrobně vysvětlena. Diagram architektury pro služby je znázorněn na obrázku 3.1.



Obrázek 3.1: Schéma architektury.

3.1 Hlavní API

Hlavní API by se dalo nazvat kapitánem celé operace, je zodpovědné za komunikaci a delegování úkolů na ostatní služby. Bylo napsáno v Golangu[9] pro jeho vysokou rychlost, širokou škálu balíčků, snadné multithreading[10] a osobní preference. API muselo zvládnout více úkolů, včetně zpracování HTTP požadavků, provádění databázových dotazů, správy ověřování uživatelů, nahrávání mediálních objektů do S3 a odesílání zpráv ve frontě zpráv.

API je postaveno podle filozofie návrhu hexagonální architektury a dodržuje osvědčené postupy pro zapouzdření, můžeme oddělit externí porty (jako je databáze, fronta zpráv, S3) od základní obchodní logiky, což umožňuje snadnou výměnu portů (Pokud databáze A způsobuje problémy, můžeme ji nahradit databází B úpravou pouze logiky externího portu, aniž bychom změnili základní obchodní logiku), odděluje také uživatelskou stranu API tím, že odděluje HTTP handlery od obchodní logiky, získáváme zapouzdření potřebné k výměně komponent bez rizika úplného refaktoringu naší obchodní logiky. Hexagonální architektura podporuje systém, který je flexibilní i robustní, a pro projekt takového rozsahu, jako je tento, se technologie mohou časem měnit, což vytváří potřebu budoucích opatření a chrání interní jádro obchodní logiky před změnami v externích technologiích. [4]

API potřebovalo životní cyklus aplikace, aby spustilo potřebné komponenty, četlo konfigurace a spouštělo různé klienty. Pro tento účel byl vybrán open source Go balíček, který řeší právě tento problém, GoNextService[5] umožňuje modularizaci různých běhových komponent, pro účely tohoto projektu se zaměříme na komponenty fronty a HTTP, protože to jsou hlavní komponenty, které služba využívá, balíček také umožňuje komponenty pro čištění, Healthz a Metricsz, které jsou nepostradatelné pro monitorování a kontroly stavu a hodily by se při nasazování služby. Při spuštění hlavního API načte konfigurace ze souborů YAML a proměnných prostředí, poté spustí router gorilla mux[6] a potřebné klienty fronty, S3 a databáze. Poté je vytvořena komponenta fronty s klientem

fronty a obchodní logikou pro operace fronty, HTTP router gorilla mux a komponenta fronty jsou poté načteny do nové aplikace GoNextService a spuštěny, což je životní cyklus aplikace API, který zajišťuje elegantní vypnutí a modularitu.

API má také za úkol provést počáteční operaci při prvním spuštění, musí také spustit migrace databáze a migrace fronty zpráv, což se provádí jako init kontejner oddělený od běhového API. Migrace jsou spravovány pomocí Goose [7], správce migrací napsaného v Go, který umožňuje soubory migrací Go a SQL, v tomto projektu byly pomocí Goose spravovány pouze migrace SQL, migrace fronty zpráv byly spravovány pomocí konfigurace YAML a spuštěny jako součást binárního souboru migrací.

Při komunikaci s databází bylo použito ORM Jet[8], na rozdíl od jiných ORM go-jet pracuje s definovanými tabulkami v relační databázi a generuje typy, které se používají v API, tyto typy umožňují typově bezpečné databázové dotazy a eliminují riziko možných útoků SQL injection.

3.1.1 Fronta zpráv

Účelem fronty zpráv je umožnit službám komunikovat mezi sebou na vyžádání, zejména odesílat úlohy nebo výsledky úloh. Například API může odeslat zprávu úlohy do služby Deep learning a ta po splnění úkolu odešle zprávu s výsledkem úlohy zpět do API. Pro frontu zpráv byl vybrán NATS, protože je poměrně nenáročný, má Docker image a Helm chart a existují pro něj SDK jak v Go, tak v Pythonu, což ho činí ideálním pro náš projekt.

3.1.2 Databáze

Databází zvolenou pro tento projekt byla PostgreSQL, relační databáze, která splňuje potřeby projektu, jako je tento. Má Docker image i Helm chart, což usnadňuje integraci do současného systému, a má také Go SDK, což usnadňuje práci s ní.

3.1.3 Úložiště objektů

Úložiště objektů je místo, kde budou uloženy všechny mediální soubory generované uživateli nebo aplikací. Na začátku bylo zvoleno MinIO díky faktorům, jako je snadnost použití, jeho Helm chart a kompatibilita s AWS S3 SDK. Během vývoje se objevil neočekávaný problém s MinIO, protože image z Helm chartu měla problémy se zpracováním větších datových objemů videa. Nakonec muselo být nahrazeno cloudovým řešením. Bylo vybráno DigitalOcean Spaces kvůli snadnosti nastavení, většina jeho API je kompatibilní s S3 SDK, které projekt používal, takže bylo potřeba změnit jen málo logiky. DigitalOcean Spaces dokázalo zpracovat datové objemy videa odeslané z aplikace, a tak se stalo řešením pro úložiště objektů.

3.2 Služba hlubokého učení

Služba hlubokého učení má za úkol poslouchat nové úlohy analýzy a plnit je. Spotřebovává zprávy z NATS streamu, ve zprávách najde URL adresu videa uloženého v úložišti objektů, stáhne video a spustí ho pomocí trénovaného modelu CNN, aby našla, kde se myši ve videu nacházejí. Poté nahraje analyzovanou verzi videa s ohraničujícími rámečky kolem předpokládané polohy myší ve videu a heatmapu pozic myší během videa[1], cenná data, ze kterých mohou výzkumníci vytvářet nové objevy. Tato služba je napsána v Pythonu, protože je to nejpoužívanější programovací jazyk pro vývoj neuronových sítí, s knihovnami PyThorch nebo MatPlotLib, které se ukázaly jako důležité během vývoje projektu. Služba je založena na Docker image ultralytics, aby nejdůležitější SDK pro práci s modely YOLO byly již nainstalovány a připraveny k použití, což minimalizuje počet případů, kdy je třeba závislosti po sestavení znovu nainstalovat.

3.3 Služba kódování

Během vývoje se objevil problém s kodeky analyzovaných videí ze služby hlubokého učení, videa se nikde nezobrazovala. Po překódování pomocí FFMPEG se opravila a dala se zobrazit v prohlížeči. První pokus o opravu byl spustit překódování přímo ve službě hlubokého učení, ale to se nepodařilo. Nějaký problém s image ultralytics zabránil Pythonu správně rozpoznat kodek X264. Druhý pokus o vyřešení tohoto problému byl vytvořit službu, jejímž jediným účelem bylo zpracování kódování a zkoumání různých formátů médií. Pro tyto úkoly jsme potřebovali rychlost, takže Go bylo opět zvoleno jako jazyk, ve kterém měla být služba napsána.

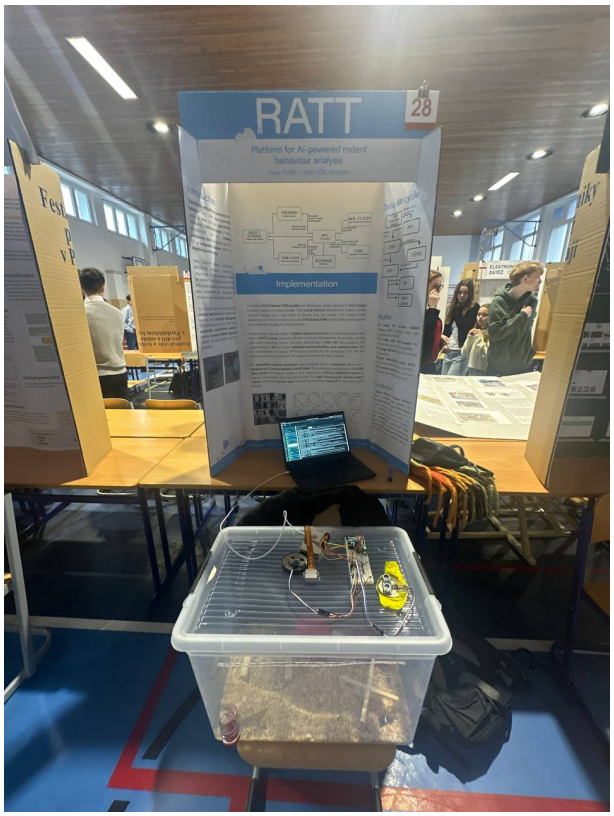
Služba spotřebovává zprávy z NATS streamu, ve zprávách jsou úlohy kódování odeslané z API, podobně jako úlohy analýzy obsahují URL adresu videa v úložišti objektů. Videa jsou stažena a na nich je spuštěn FFMPROBE, aby se shromáždila všechna důležitá data (data, rozměry, délka, kódování...). Poté se na nich spustí FFMPEG, aby se překódovala na X264, výsledný soubor mp4 se nahraje do úložiště objektů a zpráva s výsledkem úlohy se odešle zpět do hlavního API.

Rozhodnutí vybudovat a vyvinout vyhrazenou službu kódování je dlouhodobá investice. Služba je navržena tak, aby se dala snadno rozšířit pro zpracování jakéhokoli jiného typu média, které lze analyzovat pomocí FFMPEG. Lze ji použít ke generování miniatur pro analyzovaná videa nebo k překódování videí do jiných formátů než X264. Vzhledem k tomu, že spotřebovává úlohy z NATS streamu, horizontální rozšíření služby není problém, protože NATS by vyvážil zatížení úloh mezi různými replikami služby, což zaručuje méně úzkých míst a vyšší propustnost.

3.4 Klecový klient

[1]Heatmapa je vizuální reprezentace dat, kde jsou hodnoty zobrazeny jako barvy. V tomto případě heatmapa ukazuje, kde se myši ve videu nejčastěji pohybovaly, přičemž intenzivnější barvy označují častější výskyt myší na daném místě.

Klecový klient je hardwarová strana projektu, skládající se z Raspberry Pi 5 s různými senzory pro neustálé monitorování myší v kleci. Raspberry Pi bylo konfigurováno přes SSH, po počátečním nastavení a stažení potřebných nástrojů a závislostí bylo připraveno k zahájení vývoje. Psaní Python kódu pro klecový klient přes SSH se ukázalo jako nepraktické kvůli lagům a nedostatku správného IDE. Pro vyřešení tohoto problému by se kód psal jako obvykle v jiném počítači, kde tyto problémy neexistují, a nahrával by se do git repozitáře projektu v novém adresáři. Poté bylo Raspberry Pi nakonfigurováno tak, aby stahovalo pouze z tohoto konkrétního adresáře a spouštělo v něm Python skript. Během vývoje klece byly provedeny dvě iterace. V první iteraci by se ultrazvukové senzory používaly k měření hladiny jídla a vody, které zvíře nechalo. K dosažení tohoto cíle byly odříznuty horní části dvou nádob na vodu a připevněn uzávěr obsahující ultrazvukový senzor. U nádoby na jídlo tento přístup fungoval dobře, totéž se nedá říci o nádobě na vodu. Po odstranění horní části začala voda unikat, což nakonec zaplavilo klec. Po vysušení všeho bylo rozhodnuto, že nádoba na vodu zůstane bez měřicího senzoru. Na konci se první iterace klece skládala z optického senzoru TSL2591 pro měření světla, LCD, Raspberry Pi Camera Module 3 NoIR Wide, ultrazvukového senzoru HC-SR04 pro měření objemu jídla a senzoru teploty a vlhkosti DHT22. Kód pro klec se skládá z inicializačního kroku a hlavního cyklu. V inicializačním stavu klec požádá API o nové identifikační údaje. V rámci identifikačních údajů klec najde token pro autorizaci vůči API a párovací kód. Párovací kód se zobrazí na OLED LCD a uživatelé mohou klec spárovat se svým profilem pomocí tohoto kódu. Hlavní cyklus se spustí, jakmile uživatel spáruje klec se svým účtem. Klec automaticky zahájí nahrávání video útržku z kamery. Po nahrání klec uloží aktuální hodnoty ze senzorů a odešle je s videem do API. To se provádí asynchronně, aby klec mohla zahájit další cyklus, aniž by musela čekat na dokončení nahrávání předchozí zprávy. Tento cyklus bude probíhat neomezeně, dokud se program nezastaví nebo se klec nevypne. Pro druhou variantu klece byla zvolena větší krabice, aby bylo zaručeno plné pokrytí kamery. Nevýhodou této volby byla ztráta možnosti implementovat válec na jídlo použitý v první variantě. Jinak je klec funkčně stejná a poskytovala lepší a stabilnější video výstup. Druhý prototyp klece, který byl vystaven na soutěži FVTP, je vidět na obrázku 3.2.



Obrázek 3.2: Klec vystavená na soutěži FVTP.

3.5 Webový klient

Webový klient nebyl v původním návrhu systému, jako důkaz konceptu nebylo jisté, zda bude čas ho implementovat. Slouží pro základní vizualizaci dat a existuje mnoho funkcí, které je třeba implementovat a vylepšit, aby se dal nazvat stabilním produktem, nicméně se mu zhruba daří plnit svůj úkol, takže bude popsán i zde. Slouží jako rozhraní pro uživatele k vizualizaci jejich vytvořených klecí a videí, podporuje také vytváření a autorizaci uživatelů. Uživatelé mají přístup ke svým analyzovaným videím, aby viděli výstupní data, a také ke svým klecím, aby viděli nejnovější hodnoty ze senzorů a analyzovaná videa z klece. Služba byla také napsána v Go, přičemž využívala stejné postupy životního cyklu a kontejnerizace jako dříve zmíněné služby založené na Go. Nicméně, protože tato služba měla sloužit jako webové rozhraní, byl jako šablonovací jazyk pro soubory HTML vybrán Templ[11]. Templ umožňuje vývojářům psát šablony HTML s logikou, jako by psali program v Go, což umožňuje použití paradigmat objektově orientovaného programování, jako je dědičnost a zapouzdření. Byl vybrán z osobní preference, aby se pokud možno pokračovalo v psaní v Go a aby se zabránilo použití jazyků jako JavaScript nebo TypeScript pro logiku na straně serveru. Templ poskytl funkce potřebné pro webového klienta, a proto bylo ideální s ním psát.

4 Data

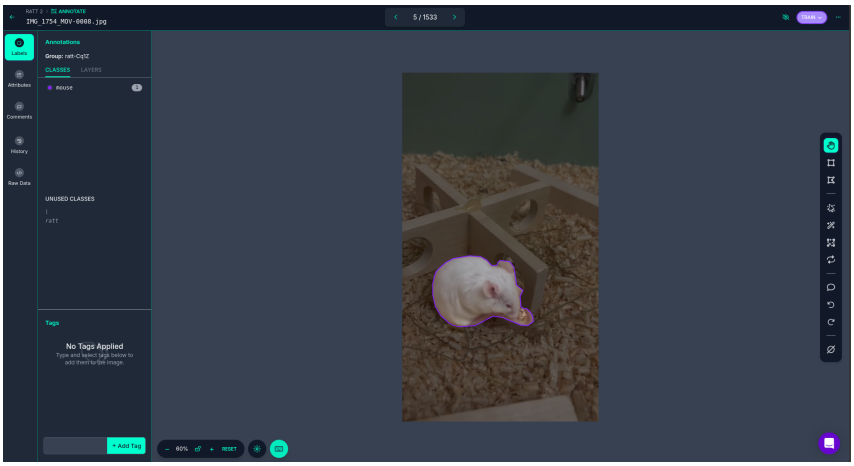
Hlavním zaměřením tohoto projektu jsou data, jak byla data shromažďována, jak byla využívána a jak shromáždit co nejvíce dat a zajistit, aby žádná z nich nebyla nevyužita. Již jsme poskytli přehledové vysvětlení toho, jak jsou data neustále shromažďována klecí, a v této kapitole budeme diskutovat o mnoha způsobech, jakými tato data využíváme.

4.1 Sběr dat

Sběr dat začíná kamerou, může to být kamera klece nebo jakákoli jiná kamera schopná poskytnout dostatečně kvalitní obraz/video, abychom s ním mohli pracovat. Data o myších byla vyhledána na webu a v raných fázích vývoje byla použita slušně velká datová sada s více než 6000 obrázky, nicméně plně nepokrývala naše potřeby a byla vyžadována specifičtější datová sada. Jakmile byl první prototyp klece připraven k zahájení nahrávání, byla zakoupena myš z místního zverimexu, umístěna do klece a klec začala poprvé nahrávat klipy chování myši. Neupravená videa byla ukládána do úložiště objektů a fáze sběru dat byla plně automatizována.

4.2 Označování dat

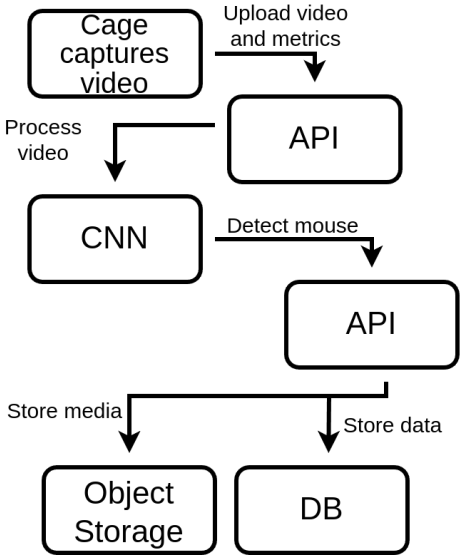
Data bylo nutné označit pro trénování a vývoj přesného a precizního agenta hlubokého učení. Označování dat je manuální proces, při kterém člověk připravuje data přiřazováním tříd k datům v datové sadě. V tomto případě je cílem agent hlubokého učení pro detekci objektů, takže označování dat spočívalo v manuálním obkreslování oblasti, ve které se myš nacházela. Pro tento účel byl jako platforma pro označování dat použit Roboflow. Platforma obsahovala kontextově řízený detektor objektů, který pomáhal během procesu označování. Označená data jsou shromažďována v datové sadě, kterou lze poté exportovat ve správném formátu pro trénování detekční konvoluční neuronové sítě. Rozhraní Roboflow pro označování dat je znázorněno na tomto obrázku 4.1



Obrázek 4.1: Označování dat v Roboflow.

4.3 Životní cyklus dat

Data hrají v tomto projektu hlavní roli. Celý životní cyklus dat lze považovat za hlavní výstup generovaný tímto projektem a o data je pečlivě postaráno od samého začátku až do konce analytického zpracování. Data začínají v kleci nebo jsou nahrána uživatelem, poté jsou odeslána do API, kde jsou uložena a je řízeno jejich zpracování. Jsou odeslána do služby hlubokého učení, kde je analyzuje model umělé inteligence, a výsledné predikce jsou poté odeslány zpět do API a výsledky jsou uloženy v databázi a úložišti objektů. Tento projekt nabízí kompletní pokrytí dat o myších od jejich sběru až po celé jejich zpracování. Tento diagram slouží jako znázornění celého životního cyklu dat4.2.

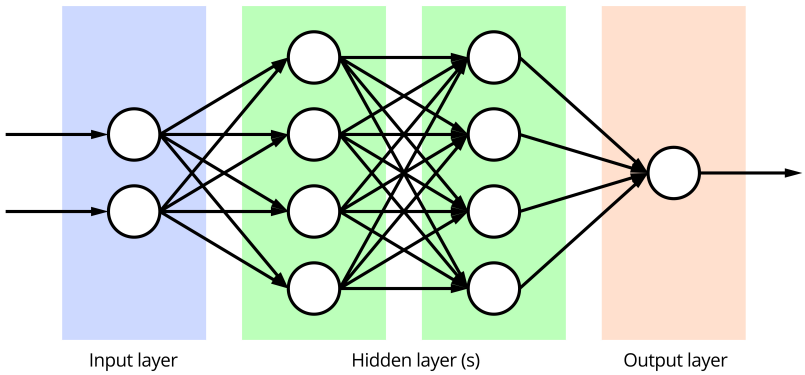


Obrázek 4.2: Diagram životního cyklu dat.

5 Vývoj modelu

Tento projekt byl hlubokým ponorem do neuronových sítí a umělé inteligence. Od samého začátku bylo zřejmé, že neuronové sítě budou nejefektivnějším přístupem pro zpracování a analýzu daných dat. Na rozdíl od tradičních metod založených na algoritmech, které vyžadují rozsáhlou kalibraci a spoléhají se na předdefinovaná pravidla, nabízejí neuronové sítě flexibilnější a adaptivnější řešení. Jejich schopnost extrahovat smysluplné vzory přímo z dat je činí zvláště vhodnými pro úkoly, kde se podmínky mohou lišit a ruční definování pravidel by bylo nepraktické nebo nedostatečné.

Bylo provedeno důkladné zkoumání různých aplikací neuronových sítí, aby se určily nejvhodnější techniky pro tento projekt. Neuronové sítě, inspirované strukturou a funkcí lidského mozku, se skládají z vrstev propojených uzlů, které zpracovávají vstupní data předáváním vážených signálů mezi nimi. Prostřednictvím trénovacího procesu, který upravuje tyto váhy pomocí optimalizačních algoritmů, jako je backpropagation[12], síť zdokonaluje svou schopnost rozpoznávat vzory, provádět predikce a zobecňovat na nová data[13]. Tato schopnost samoučení umožňuje neuronovým sítím efektivně zpracovávat komplexní, vysoce dimenzionální data, což je činí ideálními pro analýzu biologických a behaviorálních vzorů v reálných podmínkách[14]. Vizuální znázornění této struktury, kde je každý neuron propojen s další vrstvou, je znázorněno na obrázku 5.1, který ilustruje, jak informace proudí tradiční neuronovou sítí.



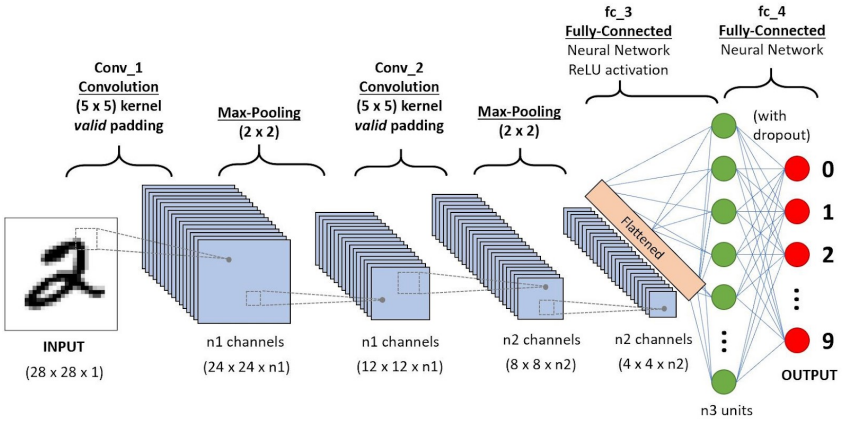
Obrázek 5.1: Vizuální znázornění tradiční neuronové sítě[15].

Využitím těchto vlastností si tento projekt klade za cíl využít techniky hlubokého učení k dosažení spolehlivé, automatizované analýzy. V následujících částech prozkoumáme specifické architektury neuronových sítí a použité metodiky, přičemž podrobně popíšeme jejich roli ve vývoji našeho systému.

5.1 CNN

Konvoluční neuronové sítě (CNN) jsou specializovaná třída modelů hlubokého učení navržená pro zpracování dat s mřížkovou topologií, jako jsou obrázky. Staly se de facto standardem v analýze obrázků a úlohách počítačového vidění díky své schopnosti automaticky a adaptivně se učit prostorové hierarchie prvků prostřednictvím backpropagation.

Architektura typické CNN se skládá z několika klíčových komponent, včetně konvolučních vrstev, které aplikují sadu učitelných filtrů na vstupní obrázek, provádějí konvoluční operace k extrahování prostorových prvků, jako jsou hrany, textury a vzory. Každý filtr aktivuje určité prvky ve vstupu, což síti umožňuje učit se různorodé a komplexní reprezentace. Aktivační funkce, jako je Rectified Linear Unit (ReLU), zavádějí nelinearitu do modelu, což mu umožňuje učit se složitější vzory. Poolingové vrstvy, známé také jako subsampling nebo downsampling vrstvy, snižují dimenzionalitu map prvků při zachování základních informací. Toto snížení snižuje výpočetní zátěž a pomáhá při detekci prvků invariantních vůči změnám měřítka a orientace. Plně propojené vrstvy, které následují po několika konvolučních a poolingových vrstvách, provádějí finální klasifikaci a rozhodovací procesy[17, 16]. Vizualizace vrstev, které tvoří konvoluční neuronovou síť, je k dispozici na tomto obrázku5.2



Obrázek 5.2: Vizualizace konvoluční neuronové sítě[17].

5.2 Detekce objektů

Detekce objektů je pokročilý úkol v počítačovém vidění, který přesahuje jednoduchou klasifikaci obrázků tím, že nejen identifikuje objekty v obrázku, ale také je lokalizuje pomocí ohraničujících rámečků. Tato schopnost identifikovat a lokalizovat objekty je nezbytná pro sledování pohybu a interakcí hlodavců v experimentálních prostředích, kde je vyžadováno přesné sledování a analýza. Tradiční metody detekce objektů, jako jsou přístupy s posuvným oknem a regionální konvoluční neuronové sítě (R-CNN), byly historicky výpočetně náročné a pomalé. Tyto metody fungují tak, že sekvenčně skenují celý obrázek nebo extrahují potenciální oblasti objektů před aplikací klasifikátoru, což může vést k vysokým výpočetním nákladům a snížení rychlosti zpracování.

V posledních letech se objevily modely založené na hlubokém učení, které řeší tato omezení. Tyto modely, zejména jednostupňové a dvoustupňové detektory, výrazně zlepšily rychlost i přesnost úloh detekce objektů. Jednostupňové detektory, jako je YOLO (You Only Look Once), zpracovávají obrázek v jednom průchodu a nabízejí rovnováhu mezi rychlostí a přesností. Tyto modely jsou zvláště výhodné, když je kritický výkon v reálném čase, protože nabízejí nižší latenci ve srovnání s jejich dvoustupňovými protějšky. Na druhou stranu dvoustupňové detektory, jako je Faster R-CNN, upřednostňují přesnost tím, že nejprve generují návrhy oblastí a poté je klasifikují, což je činí výpočetně náročnějšími, ale ideálními pro aplikace, kde je prvořadá přesnost.

Integrace transformátorů do modelů detekce objektů je dalším nedávným pokrokem. Tyto modely používají mechanismy pozornosti k zaměření na relevantní části obrázku, což zlepšuje jejich schopnost detekovat a klasifikovat objekty ve složitých scénách. Navzdory jejich slibnosti se modely založené na transformátorech stále vyvíjejí a obvykle vyžadují značné výpočetní zdroje.

Pro tento projekt bylo zaměření na použití jednostupňového detekčního modelu kvůli nutnosti sledování v reálném čase a potřebě efektivního výkonu v dynamických prostředích. YOLOv11 bylo vybráno pro svou rovnováhu rychlosti a přesnosti, díky čemuž je vhodné pro sledování pohybu hlodavců v reálném čase bez obětování kvality detekce.

5.3 YOLO

YOLO (You Only Look Once) je široce uznávaný model hlubokého učení pro detekci objektů v reálném čase, známý svou efektivitou a rychlostí. Na rozdíl od tradičních metod detekce objektů, které zpracovávají obrázek vícekrát v různých měřítkách nebo oblastech, YOLO zpracovává celý obrázek v jednom jediném dopředném průchodu. To výrazně snižuje výpočetní režii a umožňuje rychlejší detekci objektů, což je ideální pro aplikace v reálném čase. Architektura YOLO je navržena tak, aby predikovala jak třídní štítky, tak souřadnice ohraničujících rámečků pro více objektů současně, což přispívá k jejímu vysokorychlostnímu výkonu.

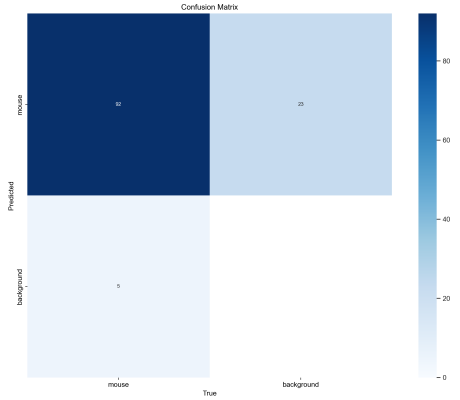
Nejnovější iterace, YOLOv11, zavádí několik klíčových vylepšení oproti svým předchůdcům, optimalizuje jak přesnost, tak výkon v reálném čase. Jedním z pozoruhodných vylepšení je začlenění extraktoru prvků založeného na transformátorech do své páteřní architektury, což zlepšuje schopnost modelu detekovat složité vzory a objekty s vyšší přesností. Tato nová páteř umožňuje YOLOv11 dosahovat lepších výsledků v úlohách, které vyžadují jemnozrnnou extrakci prvků. Dalším pokrokem je dynamický modul hlavy, který se přizpůsobuje objektům různých velikostí. Tato funkce zlepšuje přesnost detekce, zejména v náročných podmínkách, kde se objekty objevují v různých měřítkách nebo orientacích. Kromě toho je YOLOv11 optimalizováno pro nízkou latenci, což zajišťuje, že monitorování v reálném čase může probíhat bez významných výpočetních zpoždění, což je klíčový aspekt pro dynamické a nepřetržité sledování.

Pro tento projekt bylo YOLOv11 trénováno na vlastní datové sadě označených dat o pohybu hlodavců, což modelu umožňuje detekovat a klasifikovat chování hlodavců s

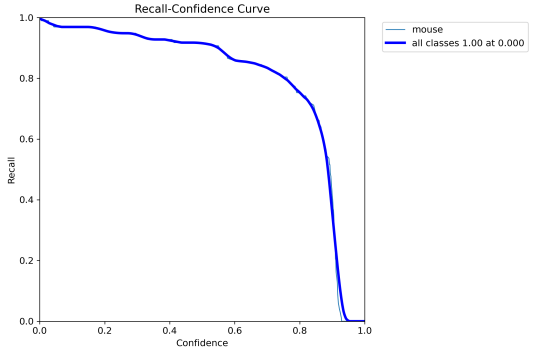
vysokou přesností. Trénovací proces zahrnoval několik fází, počínaje předzpracováním dat, včetně technik augmentace pro zvýšení rozmanitosti trénovacích dat. Model byl poté trénován na datové sadě, následovalo vyhodnocení jeho výkonu pomocí metrik precision-recall. Tento proces umožnil optimalizaci hyperparametrů pro zlepšení detekčních schopností modelu.

Integrace YOLOv11 do systému zajišťuje, že chování hlodavců lze přesně detekovat v reálném čase, což tvoří základ pro následnou analýzu a rozhodování. Výkon trénovaného modelu byl hodnocen pomocí dvou klíčových metrik: matice záměn a křivka recall-confidence. Matice záměn vizualizuje výkon klasifikace modelu porovnáním skutečně pozitivních, falešně pozitivních, skutečně negativních a falešně negativních detekcí, což nabízí vhled do jeho celkové účinnosti a oblastí pro zlepšení. Křivka recall-confidence na druhé straně ilustruje, jak se míra recall mění v různých prahových hodnotách confidence, což pomáhá určit optimální rovnováhu mezi recall a confidence detekce ve výkonu modelu.

Obrázky 5.3a a 5.3b představují metriky výkonu pro model YOLOv11, přičemž matice záměn ukazuje přesnost klasifikace a běžné chybné klasifikace, zatímco křivka recall-confidence poskytuje hlubší pochopení kompromisu mezi recall a confidence detekce ve výkonu modelu.



(a) Matice záměn modelu YOLO.



(b) Křivka recall-confidence pro model YOLO.

Obrázek 5.3: Metriky výkonu modelu YOLO pro detekci myší.

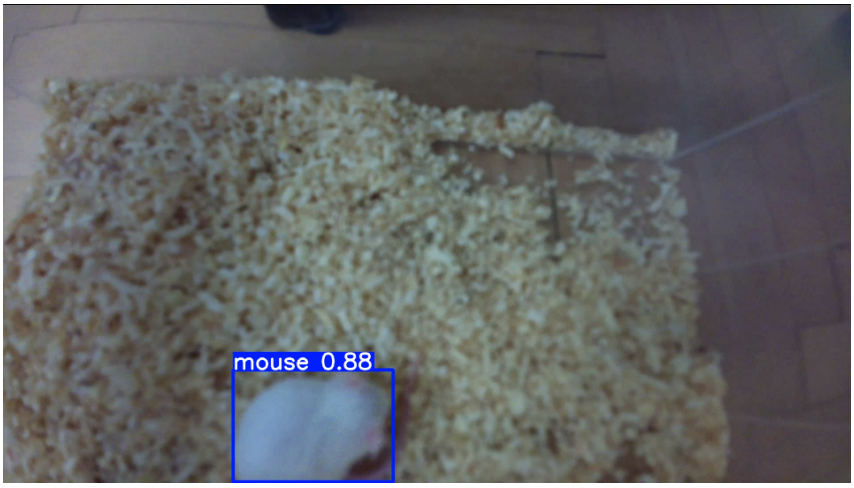
6 Výsledky

Tento projekt přinesl několik významných a hmatatelných výsledků, které demonstrují úspěšný vývoj komplexní platformy pro analýzu chování hlodavců. Za prvé jsme vytvořili plně funkční, modulární kódovou základnu pro webově přístupnou platformu. Tato platforma umožňuje výzkumníkům a majitelům domácích zvířat nahrávat, analyzovat a vizualizovat data o chování hlodavců prostřednictvím intuitivního rozhraní. Architektura, postavená na mikroslužbách a kontejnerizaci, zajišťuje škálovatelnost a udržovatelnost, což usnadňuje budoucí vylepšení a nasazení.

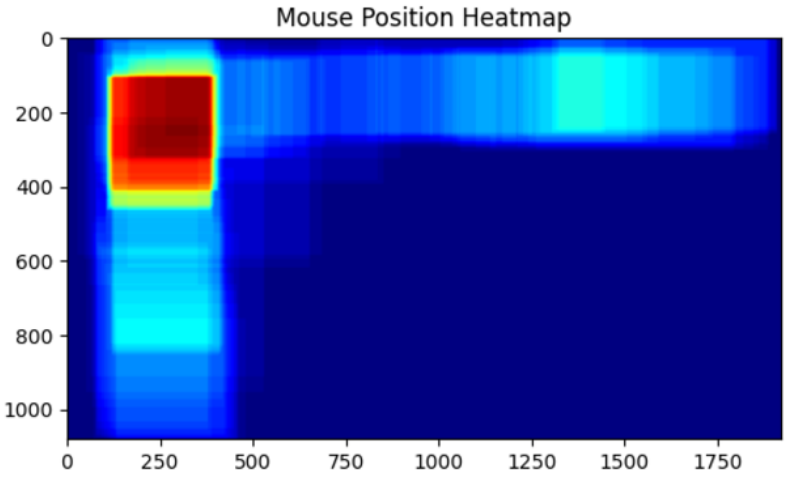
Za třetí byla vytvořena vlastní datová sada pečlivě označených obrázků myší. Tato datová sada, nezbytná pro trénování modelu hlubokého učení YOLOv11, představuje cenný zdroj pro budoucí výzkum a vývoj. Úspěšné trénování modelu pomocí této vlastní datové sady vedlo k přesné detekci objektů a sledování hlodavců ve video sekvencích. Konkrétně byl model YOLOv11 schopen identifikovat a lokalizovat myši v jednotlivých snímcích, jak je znázorněno na obrázku 6.1, který ukazuje příklad obrázku s ohraničujícími rámečky přesně zvýrazňujícími detekovanou myš.

Kromě toho platforma generuje heatmapy vizualizující prostorové rozložení pohybu hlodavce v kleci. Tyto heatmapy, jak je znázorněno na obrázcích 6.2 a 6.3, poskytují jasné a intuitivní znázornění vzorců aktivity hlodavce v průběhu času. Tato schopnost umožňuje rychlou identifikaci oblastí zájmu a úrovní aktivity a nabízí cenné poznatky o chování hlodavce.

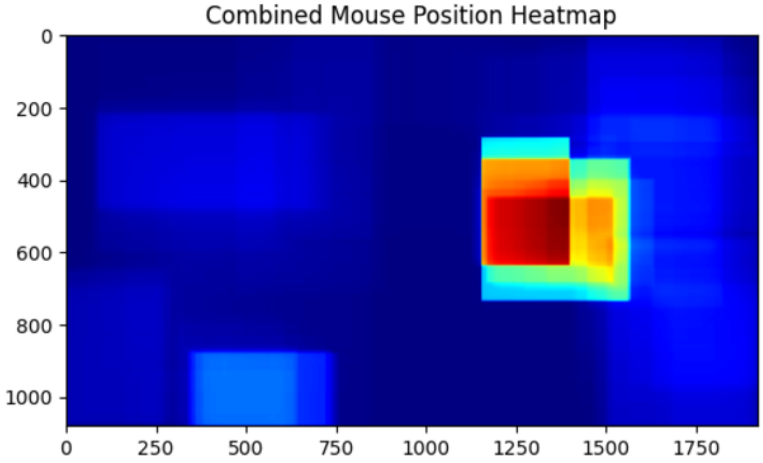
Stručně řečeno, projekt dodal robustní platformu, funkční prototyp hardwaru a cennou datovou sadu, čímž dosáhl hlavních cílů poskytnutí dostupného a efektivního nástroje pro analýzu chování hlodavců.



Obrázek 6.1: Detekce myši v akci.



Obrázek 6.2: Heatmapa polohy myši po dobu deseti sekund.



Obrázek 6.3: Heatmapa polohy myši po dobu tří minut.

7 Budoucí práce

Projekt RATT je připraven na významný budoucí vývoj, zaměřený na pokroky, které zpřesní jeho analytické schopnosti a rozšíří jeho užitečnost ve výzkumné komunitě. Primární zaměření zahrnuje vylepšení modelu umělé inteligence pro podrobnou behaviorální analýzu. V původních plánech projektu byl cíl implementovat vlastní model Long Short-Term Memory (LSTM) pro klasifikaci akcí hlodavců. Tento model byl vyvinut a prokázal počáteční potenciál v rozpoznávání časových vzorů v chování hlodavců. Vzhledem k omezením objemu dat a následnému nedostatku komplexního tréninku však jeho výsledky nebyly dostatečně spolehlivé pro zahrnutí do této práce. Budoucí úsilí se zaměří na rozšíření a zpřesnění tohoto modelu LSTM, aby byla zajištěna jeho přesnost při klasifikaci a interpretaci složitého chování hlodavců. To poskytne výzkumníkům jemnější data a odliší RATT od stávajících analytických nástrojů tím, že nabídne pokročilé schopnosti časové analýzy.

Kromě toho jsou plánována podstatná vylepšení webového klienta. Současné rozhraní, navržené primárně pro vizualizaci interního vývoje, vyžaduje komplexní redesign pro veřejnou přístupnost. Budoucí vývoj bude upřednostňovat intuitivní navigaci, robustní nástroje pro vizualizaci dat a zjednodušené pracovní postupy, aby se usnadnila interakce uživatelů a interpretace dat.

Klec navíc projde redesignem zaměřeným na výrobu zařízení připraveného k výrobě. To bude zahrnovat vývoj vlastních komponent a vyhrazené desky s plošnými spoji spolu s integrací jednotek Coral Tensor Processing Units (TPU) pro analýzu na zařízení. Tato integrace minimalizuje režii serveru a urychlí zpracování dat, což umožní analýzu chování v reálném čase. Esteticky bude klec přepracována s vlastním šasi, které bude odrážet sofistikovanou technologii, kterou obsahuje. Tyto pokroky vyvrcholí produktem nabízejícím vylepšené funkce a vynikající uživatelské prostředí.

8 Závěr

Tento projekt úspěšně vyvinul novou platformu řízenou umělou inteligencí pro analýzu a monitorování chování hlodavců, která řeší potřebu dostupných a efektivních nástrojů jak ve výzkumu, tak v péči o domácí zvířata. Využitím architektury mikroslužeb, kontejnerizace a vlastního trénovaného modelu YOLOv11 jsme vytvořili robustní systém schopný analýzy video dat v reálném čase. Integrace účelové monitorovací klece dále rozšiřuje možnosti platformy a zajišťuje nepřetržitý sběr a analýzu dat.

Návrh platformy, využívající frontu zpráv NATS a modulární architekturu, zajišťuje škálovatelnost a udržovatelnost, což umožňuje budoucí rozšíření a adaptaci. Úspěšná implementace označování dat, trénování a nasazení modelu umělé inteligence demonstruje proveditelnost použití pokročilých technik hlubokého učení pro behaviorální analýzu. Webový klient, i když je ve svých raných fázích, poskytuje funkční rozhraní pro uživatele pro přístup a vizualizaci jejich dat.

Zatímco projekt dosáhl svých primárních cílů, existuje několik cest pro budoucí vývoj. Vylepšení modelu umělé inteligence pro predikci specifického chování hlodavců, zpřesnění webového klienta pro lepší použitelnost a vývoj vyleštěnější finální verze monitorovací klece s možnostmi analýzy na zařízení jsou klíčové oblasti pro budoucí práci. Tato vylepšení dále upevní pozici platformy jako cenného nástroje pro výzkumníky i majitele domácích zvířat a v konečném důsledku přispějí k hlubšímu porozumění chování a pohodě hlodavců.

Literatura

[1] Rodriguez, A., Zhang, H., Klaminder, J., Brodin, T., Andersson, P. L., & Andersson, M. (2018). ToxTrac: A fast and robust software for tracking organisms. Methods in Ecology and Evolution, 9(3), 460–464. https://doi.org/10.1111/2041-210X.12874

[2] Noldus Information Technology. (n.d.). EthoVision® XT [Computer software]. Retrieved March 20, 2025, from https://www.noldus.com/ethovision-xt

[3] Di Francesco, P., Lago, P., & Malavolta, I. (2019). Architecting with microservices: A systematic mapping study. Journal of Systems and Software, 150, 77-97. https://doi.org/10.1016/j.jss.2019.01.001

[4] Filho, Nagib. (2024). Comparison between Ports in Hexagonal Architecture and Interfaces in Clean Architecture: A Conceptual and Practical Analysis. 1. 1-10. 10.5281/zenodo.13380201. https://www.researchgate.net/publication/383456440_Comparison_between_Ports_in_Hexagonal_Architecture_and_Interfaces_in_Clean_Architecture_A_Conceptual_and_Practical_Analysis

[5] Nextap Solutions (2024) GoNextService [Source Code] https://github.com/nextap-solutions/goNextService

[6] Gorilla (2024) Mux [Source Code] https://github.com/gorilla/mux

[7] Pressly (2024) Goose [Source Code] https://github.com/pressly/goose

[8] Go-jet (2024) Jet [Source Code] https://github.com/go-jet/jet

[9] Go Developers. (n.d.). The Go programming language. Retrieved [2025], from https://go.dev

[10] Binet, S. (2012). Can Go address the multicore issues of today and the manycore problems of tomorrow? Journal of Physics: Conference Series, 368(1), 012017. https://doi.org/10.1088/1742-6596/368/1/012017

[11] Templ Developers. (n.d.). Templ: The Go HTML templating engine. Retrieved [date], from https://templ.guide

[12] Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. Nature, 323(6088), 533–536. https://doi.org/10.1038/323533a0

[13] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436–444. https://doi.org/10.1038/nature14539

[14] Zitnik, M., Agrawal, M., & Leskovec, J. (2018). Modeling polypharmacy side effects with graph convolutional networks. Bioinformatics, 34(13), i457–i466. https://doi.org/10.1093/bioinformatics/bty294

[15] Payroll Schedule. (n.d.). Visual representation of a traditional neural network. Retrieved [23.04.2023], from https://blog.payrollschedule.net/

[16] Zhao, Xia & Wang, Limin & Zhang, Yufei & Han, Xuming & Deveci, Muhammet & Parmar, Milan. (2024). A review of convolutional neural networks in computer vision. Artificial Intelligence Review. 57. 57-99. https://www.researchgate.net/publication/379189553_A_review_of_convolutional_neural_networks_in_computer_vision

[17] Darwish, D. (2024). Improving Techniques for Convolutional Neural Networks Performance. European Journal of Electrical Engineering and Computer Science, 8(1), 1–16. https://doi.org/10.24018/ejece.2024.8.1.596

[18] Khanam, R., & Hussain, M. (2024). YOLOv11: An overview of the key architectural enhancements. arXiv. https://arxiv.org/abs/2410.17725

Seznam obrázků