Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačních technologií

Bakalářská práce

Metody computer vision pro rozpoznávání prvků grafického uživatelského rozhraní

Robert Sládek

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Robert Sládek

Informatika

Název práce

Metody computer vision pro rozpoznávání prvků grafického uživatelského rozhraní

Název anglicky

Computer vision methods to detect graphical user interface elements

---

Cíle práce

Bakalářská práce je tematicky zaměřena na computer vision grafické uživatelské rozhraní (GUI)

Hlavní cíl práce je nalezení optimálního AI modelu pro rozpoznávání prvků GUI.

Dílčí cíle práce jsou:
– Analýza současného stavu v oblasti computer vision se zaměřením na analýzu GUI
– Experimentální implementace vybraných modelu/modelů pro rozpoznávání prvků GUI
– Výběr vhodných metrik pro porovnání

Metodika

Metodika řešené problematiky bakalářské práce je založena na studiu a analýze odborných informačních zdrojů.

Praktická část je založena na implementaci vybraného modelu/modelů pro rozpoznávání prvků GUI a jejich následné testování a optimalizaci. Budou provedeny experimenty s různými daty a hyperparametry modelů, aby se získala co nejlepší přesnost a úspěšnost pro rozpoznávání GUI. Pro vyhodnocení a výběr optimální varianty budou použity vhodné metriky.

Na základě syntézy teoretický poznatků a výsledků praktické části budou formulovány závěry práce

Doporučený rozsah práce

40 – 50 stran (od Úvodu po Závěr)

Klíčová slova

GUI, grafické uživatelské rozhraní, modely, computer vision, AI

Doporučené zdroje informací

BISHOP, Christopher M. *Pattern recognition and machine learning.* [New York]: Springer, 2006. ISBN 0-387-31073-8.

GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. *Deep learning.* Cambridge, Massachusetts: The MIT Press, 2016. ISBN 978-0-262-03561-3.

PRINCE, S. J. D. Computer Vision: Models, Learning, and Inference. Cambridge, Massachusetts: The MIT Press, 2012. ISBN 978-1-107-01179-3

SZELISKI, R. Computer Vision: Algoritms and Applications. 2022. ISBN 978-3-030-34372-9

Předběžný termín obhajoby

2023/24 LS – PEF

Vedoucí práce

Ing. Jan Masner, Ph.D.

Garantující pracoviště

Katedra informačních technologií

Elektronicky schváleno dne 4. 7. 2023

doc. Ing. Jiří Vaněk, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 3. 11. 2023

doc. Ing. Tomáš Šubrt, Ph.D.

Děkan

V Praze dne 26. 02. 2024

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci "Metody computer vision pro rozpoznávání prvků grafického uživatelského rozhraní" jsem vypracoval(a) samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor(ka) uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 15 .03. 2024 _____

Poděkování

Rád(a) bych touto cestou poděkoval(a) Ing. Janu Masnerovi, Ph. D za cenné rady, podporu a čas strávený nad touto bakalářskou prací.

Metody computer vision pro rozpoznávání prvků grafického uživatelského rozhraní

Abstrakt

Tato studie se zabývá metodami computer vision pro rozpoznávání prvků grafického uživatelského rozhraní (GUI). Přiblíží aktuální stav modelů, jenž jsou schopny detekovat prvku GUI a představí jednotlivé metody computer vision pro detekci prvků. Hlavním cílem této práce je vytvoření modelu, jenž bude schopen identifikovat a klasifikovat jednotlivé prvky GUI. Bude provedena analýza současného stavu computer vision se zaměřením na analýzu GUI. V této práci se také vyberou vhodné metriky pro porovnání různých modelů a vyhodnocení jejich úspěšnosti při detekci prvků GUI.

Klíčová slova: GUI, grafické uživatelské rozhraní, modely, computer vision, AI

Computer vision methods to detect graphical user interface elements

Abstract

This study investigates computer vision methods for graphical user interface (GUI) feature recognition. It reviews the current state of the art of models that can detect GUI elements and introduces different computer vision methods for element detection. The main objective of this work is to develop a model that can identify and classifying GUI elements. An analysis of the current state of computer vision will be performed with a focus on GUI analysis. This work will also select appropriate metrics to compare different models and evaluate their success in detecting GUI elements.

Keywords: GUI, graphical user interface, models, computer vision, AI

Obsah

1 Úvod

V dnešní době, kdy se všechny informace přenášejí na digitální formu, tak se do popředí dostává téma, které spojuje reálný svět s digitálním prostředím, jde o computer vision, což je jedno z nejmodernějších odvětví IT a vývoje software. Tato disciplína umožňuje vnímat a líčit své okolí, což má velký vliv na vzájemnou komunikaci reálným světem s digitálním prostředím.

Díky tomuto stále rozvíjejícímu se odvětví vzniká téma této bakalářské práce, která se zaměřuje na metody computer vision a jejich využití pro rozpoznávání grafického uživatelského rozhraní (GUI). S rostoucí obtížností a funkcionalitou uživatelských grafických rozhraní se stává důležitým bodem porozumět jednotlivým prvkům GUI pro lepší uživatelský zážitek a díky tomu následně rozvíjet jednotlivé aplikace.

Tato bakalářská práce umožní nalézt inovativní řešení pomocí metod computer vision, která umožní efektivní rozpoznávání a analýzu prvků grafického uživatelského rozhraní.

2 Cíl práce a metodika

2.1 Cíl práce

Jak bylo zmíněno v Úvod, tato bakalářská práce se věnuje metodám computer vision pro rozpoznávání GUI. Hlavním cílem této bakalářské práce je vytvoření optimálního AI modelu pro rozpoznávání grafického uživatelského rozhraní (GUI). Jelikož se jedná o rozsáhlou práci, tak práce má, mimo hlavního cíle práce, i další tři dílčí cíle.

Prvním dílčím cílem je: Analýza současného stavu v oblasti computer vision se zaměřením na analýzu prvků GUI. Proběhne detailnější popis stávajícího postupu v oblasti computer vision a následně zhodnocení metod, které jsou v současné době používané pro rozpoznávání GUI.

Druhým dílčím cílem je: Experimentální implementace vybraných modelu/modelů pro rozpoznávání GUI. Pro splnění tohoto dílčího cíle se aplikují teoretické poznatky v praxi.

Třetím dílčím cílem je: Výběr vhodných metrik pro porovnávání. Tento dílčí cíl se bude zaměřovat na vyhodnocení výsledků experimentů a pomocí vhodně vybraných metrik, dojde k objektivnímu vyhodnocení.

2.2 Metodika

Metodika řešené problematiky bakalářské práce je založena na implementaci vybraného modelu/modelů pro rozpoznávání prvků GUI a jejich následné testování a optimalizaci. Budou provedeny experimenty s různými daty a hyperparametry modelů, aby se získala co nejlepší přesnost a úspěšnost pro rozpoznávání GUI. Pro vyhodnocení a výběr optimální varianty budou použity vhodné metriky.

Na základě syntézy teoretický a výsledků praktické části budou formulovány závěry práce.

3 Teoretická východiska

3.1 Graphical user interface (GUI)

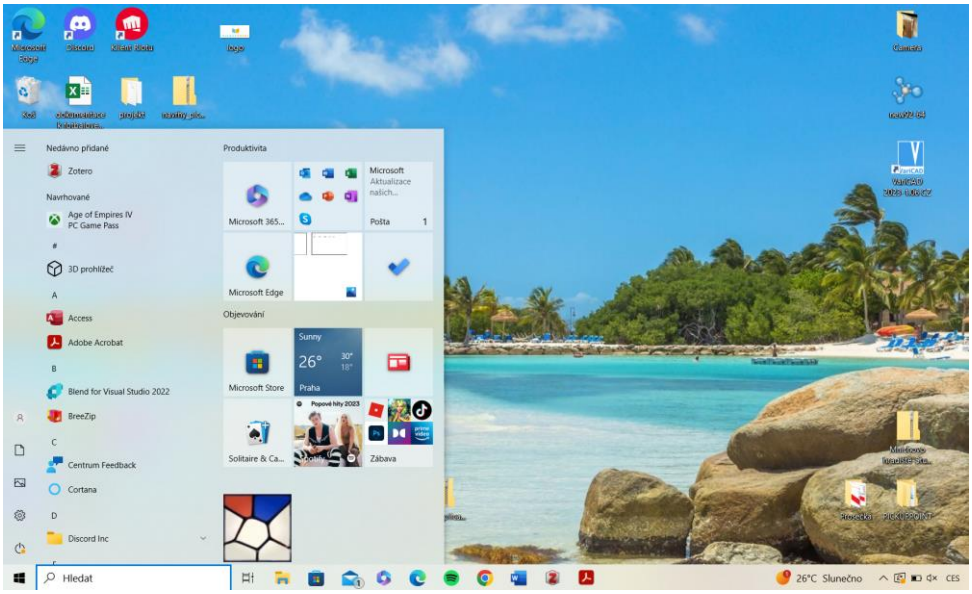
GUI lze vysvětlit vzorem model-pohled-ovladač, který nám odděluje vnitřní informace od způsobu, jakým jsou informace prezentovány uživateli (1). Jedná se tedy o digitální rozhraní, kde uživatel komunikuje pomocí různých tlačítek, nabídek, ikon.
GUI je systém pro interakci člověka s mobilem, tabletem, počítačem. Funkce jsou vázány na grafické ikony (2). Existuje spoustu prvků, které dohromady utvářejí jednotlivá grafického uživatelského rozhraní a ty nejčetnější z nich jsou vyobrazeny na Obrázek 1.
Obrázek 1 Charakteristické prvky GUI

Zdroj: (3)

3.1.1 Význam GUI

Uživatel prostřednictvím GUI komunikuje s elektronickými zařízeními (mobily, tablety, počítači). Grafické rozhraní funguje na principu grafického zobrazení informace, zatímco textové rozhraní jsou pouze data a příkazy jen v textové podobě. Uživatel s GUI komunikuje prostřednictvím myši u počítače nebo dotykem u mobilu/tabletu (4). Příklad GUI počítače je vyobrazen na Obrázek 2.

Obrázek 2 Příklad GUI

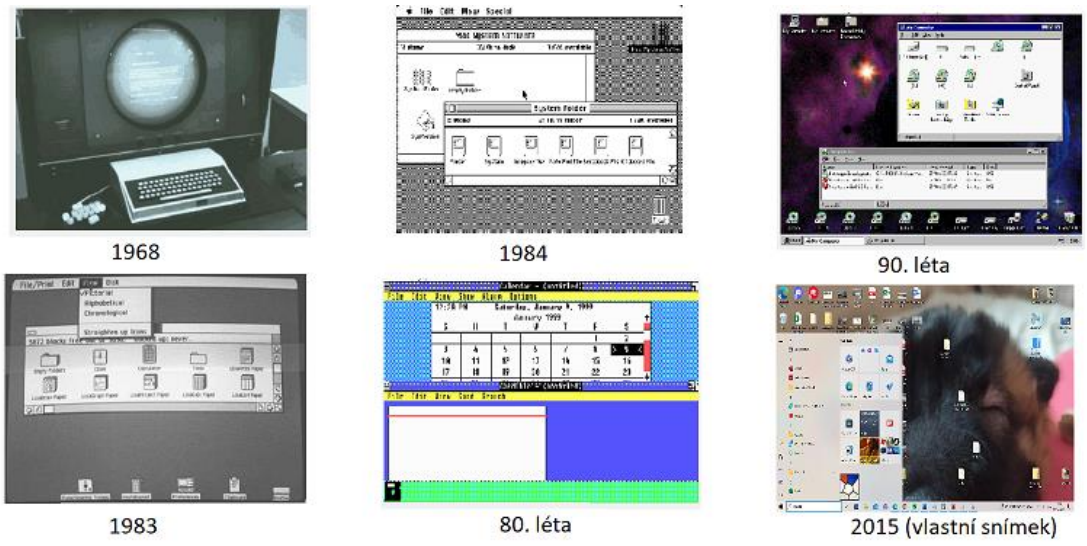


Zdroj: (Vlastní zdroj)

3.1.2 Historie GUI

GUI se začaly objevovat v 70. letech minulého století, kdy se začaly vyskytovat v jednoduché podobě a postupem času se přidávaly nové funkce, až se vyvinuly do podoby, jak je známé dnes. Za praotce grafického uživatelského rozhraní je považován Douglas Engelbart (5). Vývoj grafických uživatelských zobrazení je vyobrazen na Obrázek 3.

Obrázek 3 Historický vývoj GUI

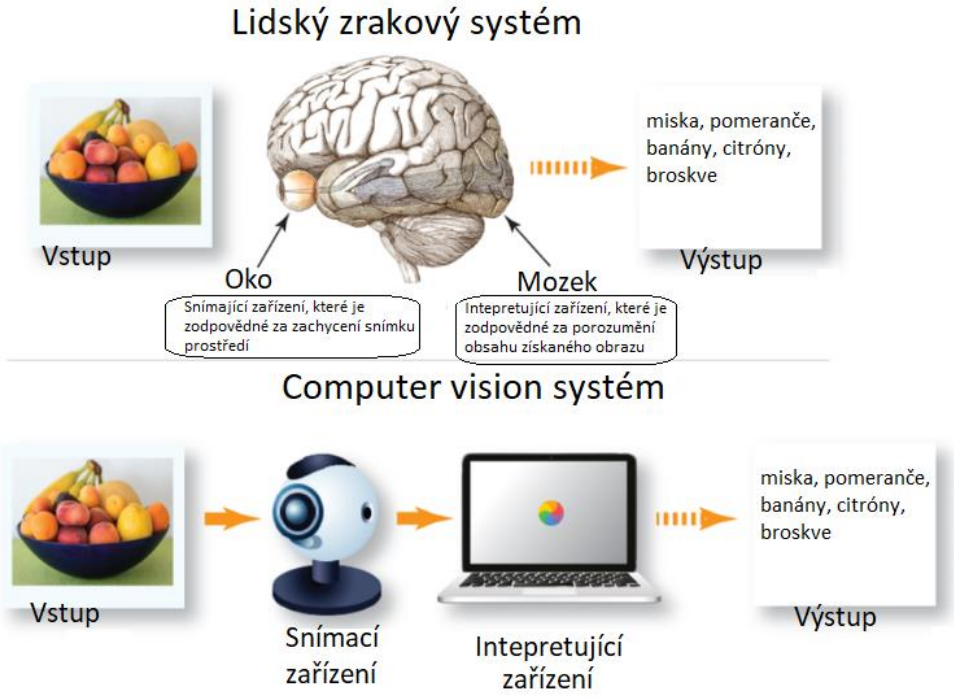


Zdroj: (5)

3.2 Úvod do computer vision

Computer vision je jedno z nejaktivnějších oblastí výzkumu pro hluboké učení, jedná se o obor umělé inteligence, který umožňuje zařízením získávat informace ze všech vizuálních vstupů, na základě zpracovaných informací umožňuje provádět akce nebo vydávat doporučení. Jinými slovy computer vision umožňuje zařízením vidět, pozorovat a chápat (6, 7). Rozdíl, jakým zařízení oproti člověku získává informace pomocí metody computer vision, je vyobrazen na Obrázek *4*.

Obrázek 4 Systém computer vision vs lidský zrakový systém



Zdroj: (8)

3.2.1 Výhody computer vision

Computer vision se uplatňuje v celé řadě průmyslových odvětví. Mezi největší benefity využívání computer vision patří:

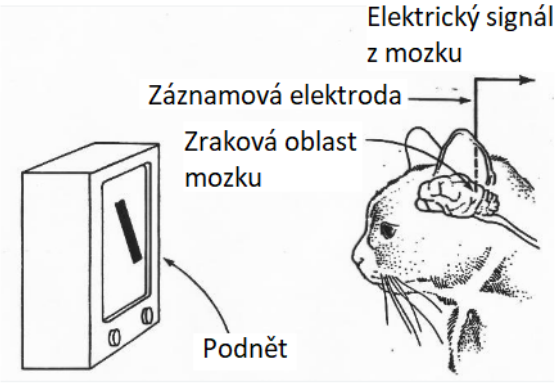
- Zlepšení efektivity práce;
- Automatizace – opakovaných se úloh;
- Zlepšení přesnosti – algoritmy analyzují data s velkou přesností;

- Úspora nákladů – díky automatizaci a vyšší efektivitě práce dochází ke snížení nákladů;
- Bezpečnost – využití k identifikaci hrozeb, prevenci nehod, účely dohledu práce;
- Rozhodování – poskytuje informace v reálném čase a analýzu dat, pro lepší výběr stávajících možností (9).

3.2.2 Historie computer vision

První zmínky o computer vision se objevují již v 50. letech minulého století, když se začalo na univerzitách objevovat analogové computer vision. Následně, pár let na to, se uskutečnilo testování reakce kočky s zapojenými elektrodami do její zrakové kůry, kde se zjistilo, že její mozek přednostně reaguje na jednoduché tvary, jako jsou čáry a křivky (8). Způsob testování kočky je vyobrazen na Obrázek *5*.

Obrázek 5 Testování reakce kočky

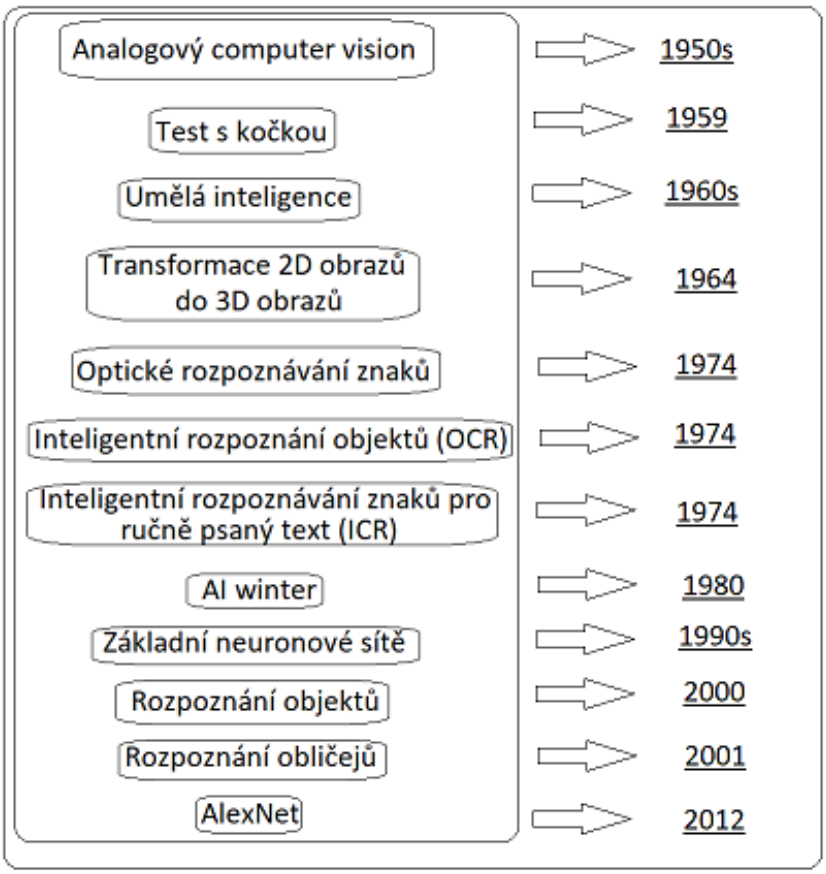


Zdroj: (8)

V 60. letech vzniká nový obor umělá inteligence, která má za cíl řešit problémy lidského vidění. Jeden z hlavních pokroků bylo zavedení optického rozpoznávání znaků, a následně také inteligentního rozpoznávání znaků pro ručně psaný text (6).

V roce 2000 se začalo s rozpoznáním objektů a rok na to se objevují aplikace pro rozpoznání obličejů v reálném čase (6). Jednotlivé milníky v pokroku v computer vision jsou zachyceny v časovém vývoji, který je vyobrazen na Obrázek *6*.

Obrázek 6 Stručný popis historie computer vision



Zdroj: (8)

3.2.3 Směřování computer vision

Dochází k zdokonalovaní technologie computer vision, která umožní plnit více funkcí. Možnost detekování více obrazů. Computer vision se bude integrovat s dalšími technologiemi, které navazují na AI. Možnost zpracování stále více informací. Velkým cílem computer vision je spolupracovat s roboty ve fyzickém světě. Pokouší se vytvořit spolupráci inteligentních robotů a lidí tak, aby roboti pomáhali plnit konkrétní cíle (10).

3.3 Přehled computer vision

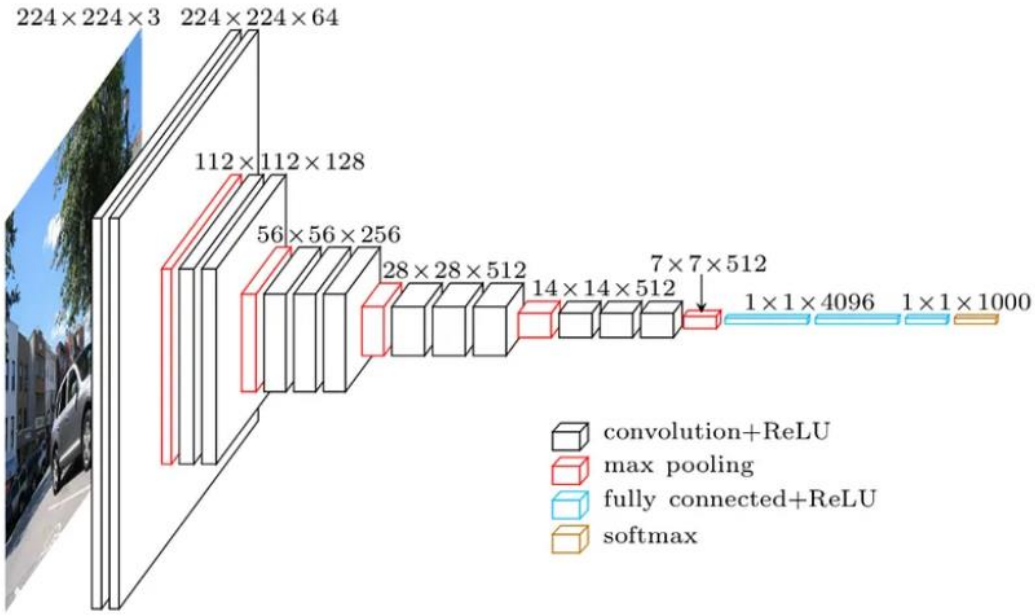
3.3.1 Convolutional neural network (CNN) (Konvoluční neuronové sítě)

Konvoluční neuronové sítě je jedno z témat strojového učení. Jedná se o jednu ze skupiny metod z umělých neuronových sítí, které se používají pro různé aplikace a typy

dat. Jedná se o druh síťové architektury, který se používá zejména pro rozpoznávání obrazu a úlohy, který obsahují zpracování jednotlivých pixelů (11).

CNN je typ modelu hlubokého učení, který je navržen tak, aby se automaticky adaptivně učil prostorové hierarchii rysů od vzorů nízké úrovně až po vzory vysoké úrovně (12). CNN nám umožňuje extrahovat rys obrazu a převést tento rys obrazu do nižší dimenze, aniž by ztratila jakékoli vlastnosti. Na Obrázek *7* je zobrazeno, jak probíhá proces zjednodušení, je zde původní obrázek 224x224x3, což je 100 353 neuronů, který se nachází u vstupního obrazu, ale po aplikování konvoluce se rozměr postupně zmenší až na 1x1x1000, což je pouze 1000 neuronů (13).

Obrázek 7 Zjednodušení obrazu CCN



Zdroj: (13)

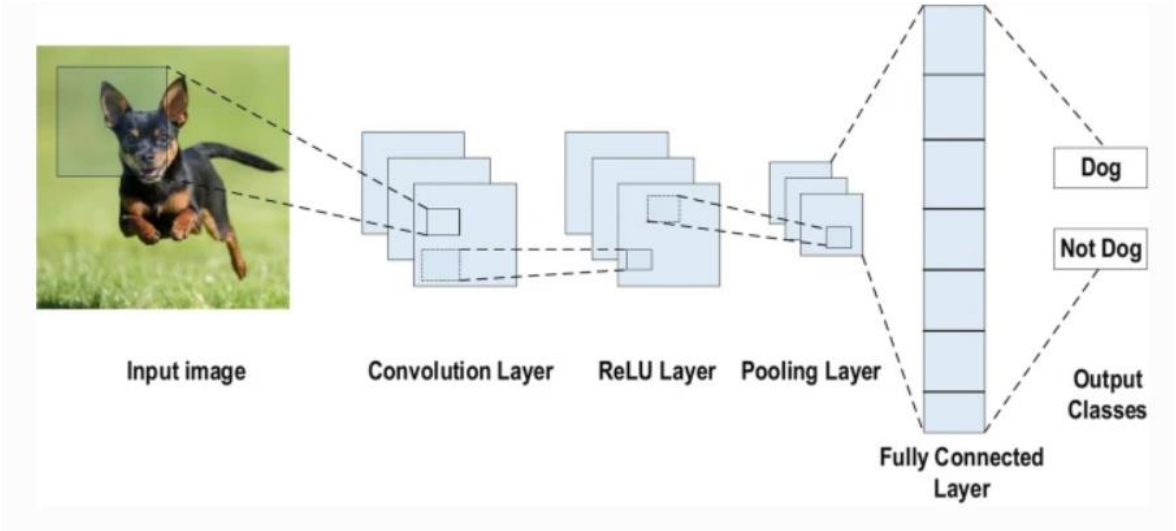
Architektura CNN se skládá z řady vrstev a ty jsou:

1. Convolutional layer (Konvoluční vrstva): Jedná se o nejvýznamnější složku CNN, která se skládá z kolekce konvolučních filtrů (kernelů). Vstupní obraz je vyjádřený N-rozměrnými metriky a je konvolvován konvolučními filtry, za účelem vytvoření výstupní mapy prvků;
2. Pooling layer (Sdružovací vrstva): jejím hlavním úkolem je zmenšovat velké mapy prvků a vytvářet menší mapy prvků a současně udržovat většinu dominantních informací v každé fázi sdružování. Mapy se zmenšují pro snížení výpočetního výkonu;

3. Activation function (Aktivační funkce): jedná se o mapování vstupu na výstup, pomocí základních aktivačních funkcí ve všech typech neuronových sítí. Výpočtem váženého součtu vstupu neuronu spolu s jeho zkreslením s vypočítá vstupní hodnota;
4. Fully connected layer (Plně připojená vrstva): jedná se většinou o poslední vrstvu v architektury CNN. Spojuje neurony v jedné vrstvě s neurony v jiné vrstvě. Používá se ke klasifikaci obrázku do různých kategorií (12, 14, 15).

Na Obrázek 8 je vyobrazen příklad architektury CNN, kde jsou názorně zobrazeny jednotlivé vrstvy této architektury.

Obrázek 8 Příklad architektury CNN pro klasifikaci obrázků



Zdroj: (15)

3.3.2 Edge detection (Detekce hran)

Detekce hran pracuje na způsob, že vyhledává body, u kterých se radikálně mění intenzita (jas) obrazu, což naznačuje přítomnost přechodu mezi různými objekty či strukturami v obraze. Detekce hran se využívá například pří segmentace obrazu, rozpoznávání objektů (16).

3.3.3 Computer vision detection using automatization tools

Metody detekce pomocí automatizovaných nástrojů umožňují sběr dat o struktuře a prvcích GUI, které nepotřebují zásah člověka. Využití těchto metod je při testovaní a optimalizaci aplikací.

3.3.3.1 Screenshot Analysis (Testování snímků obrazovky)

Testování snímků obrazovky, funguje na principu pořízení snímku obrazovky uživatelského rozhraní, který se následně testuje. Hlavním úkolem testování snímků obrazovky, je zkontrolovat, jak dané uživatelské rozhraní vypadá na jednotlivých zařízení (17).

Postup při této metodě je následující, nejdříve dojde k výběru nástroje pro testování snímku obrazovky. Následně se definují scénáře, kde jde o určení stránek, popřípadě obrazovek dané aplikace, která má být testována. Zde lze také definovat jednotlivé kroky, které uživatel bude na dané stránce provádět. Po tomto kroku následuje pořízení snímků obrazovky, tuto část lze provést jak ručně, tak i automaticky. Takto pořízené snímky obrazovky jsou stavebním kamenem pro jednotlivé testy snímků obrazovky (18).

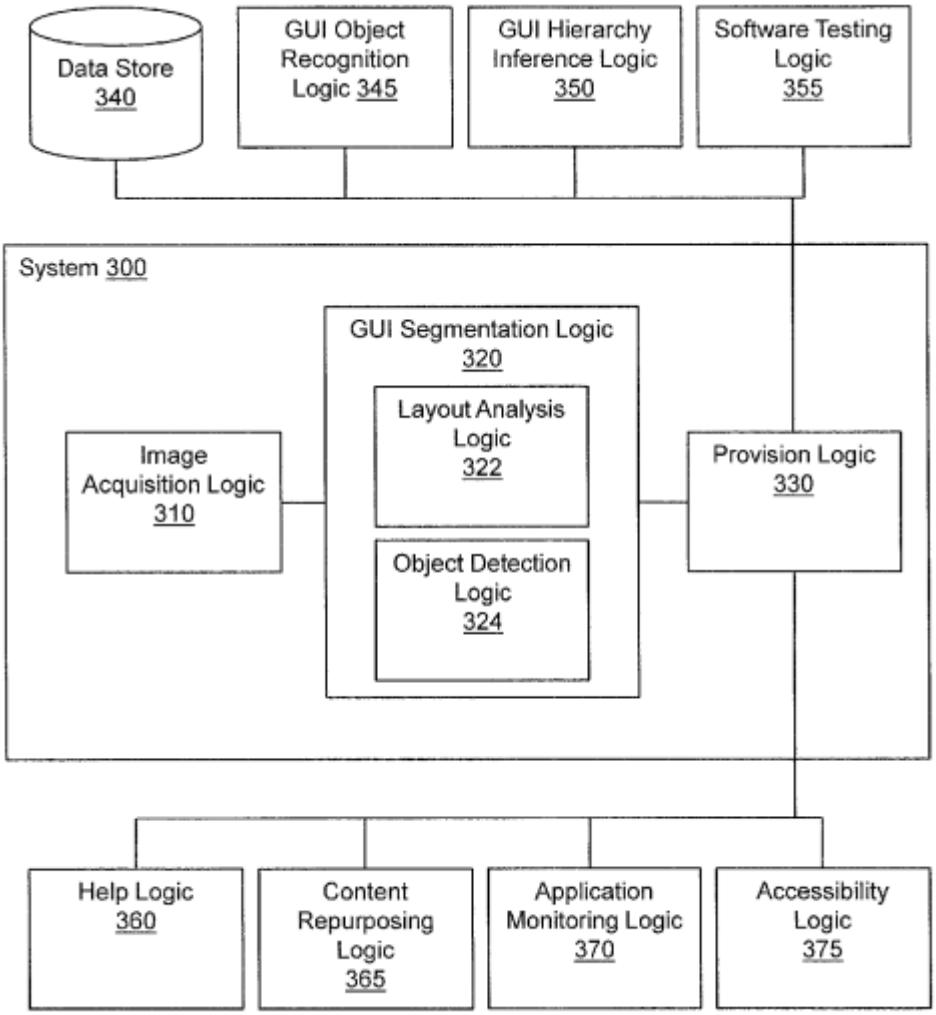
Po pořízení snímků následuje jednotlivé porovnání snímků obrazovky, kde se zjistí, zda testované GUI odpovídá očekáváním, neboť se zjišťuje, kde se vyskytují jednotlivé nesrovnalosti či vizuální chyby. Na závěr dojde k řešení problému, pokud byl objeven, tak v tomto bodu dochází ke změně kódu nebo designu stránky, tak aby byly jednotlivé chyby odstraněny. Po provedení úprav je doporučeno znovu provést celé testovaní snímku obrazovky, aby bylo zajištěno, že všechny problémy byly vyřešeny (18).

3.3.3.2 Document Object Model analysis (DOM)

Document object model je v základu reprezentace všech prvků uživatelského rozhraní na webové stránce, která se vykresluje ve webovém prohlížeči klienta nebo může jít i o dokument XML. V DOM jde tedy o reprezentaci, která není závislá na platformách a jazycích a umožňuje interakci s objekty v dokumentech XML, HTML, XML (19).

Příklad modelu, který umožní identifikovat jednotlivé prvky GUI je vyobrazen na Obrázek 9, popisy jednotlivých části jsou uvedeny v Tabulka *1*.

Obrázek 9 Model pro detekci objektů



Zdroj: (20)

Tabulka 1 Popis modelu pro detekci GUI

Číslo označení	Popis
300	Prvky spojené s identifikací komponentů GUI
310	Získávání obrazu, která generuje obraz GUI
320	Rozdělení GUI na sadu operačních segmentů a logiku
322	Logika analýzy rozložení – rekurzivně rozdělit části obrazu GUI
324	Logika detekce objektů
330	Poskytnutí dat identifikující sadu operačního systému
340	Datové uložiště
345	Logika rozpoznávání objektů GUI
350	Odvozování hierarchie GUI
355	Logika testování softwaru
360	Logika nápovědy
365	Logika přetváření obsahu
370	Logika sledování aplikace
375	Logika přístupnosti

Zdroj: (20)

3.3.4 Image segmentation

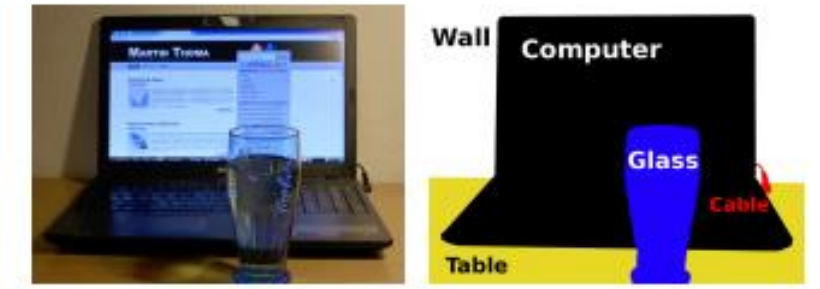
Jedná se o metodu v oblasti computer vision, která pracuje na bázi rozdělení obrazu na jednotlivé segmenty (21). Tyto segmenty lze identifikovat jako jednotlivé objekty, pozadí, různé hrany. Jedná se o způsob rozpoznávání objektů, porozumění a segmentace hranic. Základním přístupem k současnému rozpoznávání a segmentaci spočívá v formulaci problému označení každého pixelu v oblasti příslušností ke třídě objektu (22, 23).

Existují dva způsoby segmentace. Jeden způsob je označování pixelů podle objektu a následně je přiřazovat k sobě. Každý pixel v obraze je bod v trojrozměrném prostoru, jenž se skládá z intenzity červeného, modrého a zeleného kanálu, jednoduše řečeno každý pixel v obraze se považuje za samostatný datový bod (21). Druhý způsob segmentace je odvozením polohy uzavřeného obrysu, jenž ohraničuje dva objekty (24).

Na Obrázek 10 je vyobrazen příklad fungování segmentace v praxi. Na Obrázek 10 (a) máme zkoumanou scénu, ze které pomocí použití segmentace získáme vyobrazení, které

je vidět na Obrázek *10*(b). Zde je zkoumaný obraz rozdělený na jednotlivé segmenty(objekty) (22). Díky tomu lze po určité úpravě použít segmentaci k detekci prvků GUI.

Obrázek 10 Příklad použití segmentace



(a) Example Scene (b) Visualization of a found segmentation

Zdroj: (22)

3.3.5 Object detection (Detekce objektů)

Object detection je technika z oboru computer vision, která je schopná identifikovat a lokalizovat jednotlivé objekty na snímcích. V případě, že na snímcích je více objektů jiných tříd, tak je nutné nejen lokalizovat tyto objekty, ale také je klasifikovat do příslušných tříd (25). Hlavním cílem detekce objektů je simulovat počítačem schopnost člověka, který je schopen při pohledu na obrázek nebo video okamžitě identifikovat a lokalizovat jednotlivé objekty (26).

3.3.6 Optical character recognition (OCR)

Hlavním úkolem OCR je rozeznat s co největší možnou přesností všechny znaky, který se nachází na obraze (16). Existují dva základní typy OCR: tradiční a ručně psané. Tyto typy se liší ve způsobu extrakce informací, ale oba způsoby vedou ke stejnému výsledku (27).

Systém OCR se dělí na hardware a software. Práce OCR se dá členit na tři části, kde jedna část se věnuje předzpracování obrazu, kde se převede skutečný tvar dokumentu na obrázek, následně se převede na černobílý formát, kde se posuzují světlé a tmavé znaky, poté pomocí metody OCR je obraz rozdělen na jednotlivé části (tabulky, text, grafika). Další část se věnuje rozpoznávání znaku AI, kde se k tomu používají dvě metody: rozpoznávání vzorů

a rozpoznávání funkcí. Poslední část se zabývá Post-Preprocessing, kde se pomocí AI opraví chyby v konečném souboru (28).

V dnešní době se technologie OCR hojně používá pro pomoc při digitalizaci dokumentů, v bankovnictví a finančním sektoru, převodu textu na řeč, v oboru lékařství pro přepis lékařských štítků, pro rozpoznávání SPZ u vozidel, pro detekci ulic, silnic a klasifikaci značek (27). Metodu OCR lze využít pro rozpoznávání a detekci textu v GUI.

3.3.7 Region-based methods

Jelikož CNN nejsou schopny se vypořádat s větší frekvencí výskytu objektů, tak jedním způsobem, jak s tímto problémem pracovat, je vyhledávání pomocí posuvného okna. Tento přístup vymezí oblast, kde se následně aplikuje CNN. Bohužel je tento způsob velmi neefektivní, jelikož stejný objekt, může být reprezentován na obrázku s různými velikostmi, či jinými poměry stran. Pokud by se každý tento objekt zvlášť zkoumal, tak by to bylo velmi výpočetně nákladné (29).

V roce 2014 objevuje metoda R-CNN, která je představena vědci z kalifornské univerzity v Berkeley. Na rozdíl od klasických konvolučních neuronových sítí funguje R-CNN tak, že používá algoritmus selektivního vyhledávání, který rozdělí obrázek do regionů (30).

Regiony jsou menší oblasti obrázku, které mohou obsahovat objekty, které ve vstupním obrázku hledáme. K snížení počtu návrhů regionů se používá algoritmus selektivního vyhledávání. Tento algoritmus bere obrázek jako vstup a výstupem je generování návrhů regionů. Tento algoritmus vygeneruje takto přibližně 2000 návrhů. Postup algoritmu selektivního vyhledávání lze rozdělit do tří bodů:

- Vygeneruje se počáteční dílčí segmentaci vstupního obrázku, například pomocí metody Felzenswalben;
- Rekurzivně se kombinují menší podobné regiony do větších, kde lze použít Greddyho algoritmus;
- Pomocí návrhů segmentovaných regionů se generují kandidátní umístění objektů (29).

3.4 Modely detekce objektů

Modely detekce objektů jsou trénovány na obrázcích pro detekci, identifikaci objektů s přesným umístěním a ohraničením pomocí *bounding* boxu. Tyto modely jsou založený na hlubokém učení a provádí detekci pomocí konvolučních neuronových sítí. Existují dva druhy modelů, nimiž jsou jednostupňové modely a dvoustupňové modely (31).

3.4.1 One-stage object detectors (jednostupňový modely)

Jednostupňové modely jsou navrženy tak, aby prováděly detekci objektů v jednom jediném kroku. Obrázky prochází neuronovou sítí pouze jednou a rovnou se vymezí všechny *bounding* boxy. Jednostupňové modely jsou rychlejší než dvoustupňové modely, které detekci rozdělují na dva kroky. Nejznámější jednostupňové modely jsou: YOLO, Single Shot Detector (SSD), RetinaNet (32).

3.4.1.1 RetinaNet

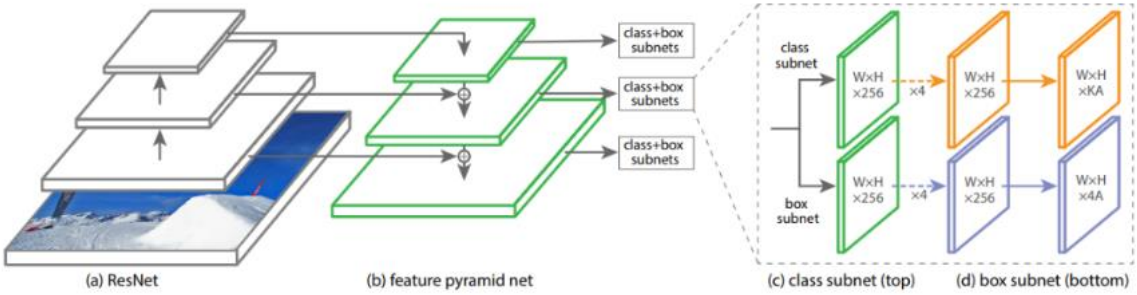
RetinaNet je jedním z nejlepších jednostupňových modelů detekce objektů, který je efektivní i při zachycování malých objektů. RetinaNet byl vyvinut jako vylepšení stávajících jednostupňových modelů detekce objektů Feature Pyramid Networsk a Focal Loss (33).

Tento model se skládá z páteřní sítě a dvou dílčích, z nichž každá je určena pro konkrétní úkol. Páteřní síť má za úkol vytvoření příznakových map na základě zpracování vstupního obrázku. První dílčí síť se stará o konvoluční klasifikaci objektů na výstupu páteřní sítě. Druhá dílčí síť provádí konvoluční regresi ohraničujících polí (34).

Architektura Retina modelu je zobrazena na Obrázek 11 a funguje na principu pyramidy, na základě čtyř stěžejných bodů:

- Cesta zdola nahoru: Vypočítává mapy prvků v různých měřítkách, přičemž nezáleží na velikosti vstupního obrázku;
- Cesta shora dolů a Laterální spojení: Převzorkuje nahoru prostorově hrubší mapy prvků z vyšších úrovní pyramidy;
- Klasifikační dílčí sít: Má za úkol předpovídat pravděpodobnost přítomnosti objektu v každém prostorovém umístění;
- Regresní dílčí sít: Má za úkol regresovat offset pro ohraničující rámečky od kotevních rámečků pro každý skutečný objekt (33).

Obrázek 11 Architektura modelu RetinaNet



Zdroj: (33)

3.4.1.2 YOLO (You Only Look Once)

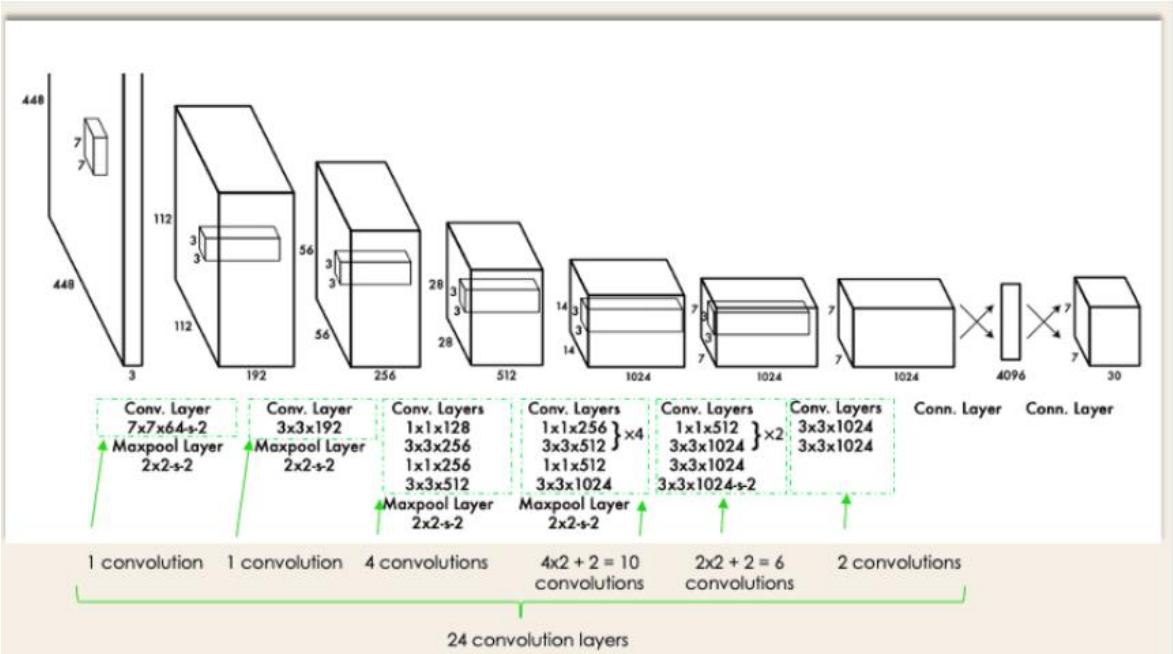
Jedná se o velmi populární jednostupňový model, který se používá pro detekci objektů. Je známý především díky své přesnosti a rychlosti práce. Model YOLO byl představen v roce 2016 Josephem Redmonem. YOLO používá *end-to-end* neuronové sítě, které dělají předpovědi hraničních rámečků a pravděpodobnosti tříd současně (35).

Výhody modelu YOLO jsou následující:

- Rychlost: YOLO model pracuje velmi rychle, díky tomu dokáže detekovat jednotlivé objekty v reálném čase;
- Vysoká přesnost: YOLO poskytuje přesné výsledky s minimálními chybami na pozadí;
- Schopnost učení se: Algoritmus YOLO se velmi dobře učí, což mu umožnuje zpracovávat reprezentace objektů a aplikovat tuto znalost při jejich detekci (36).

Architektura YOLO se skládá z 24 konvolučních vrstev, zahrnující čtyři maximální sdružovací vrstvy a dvě plně propojené vrstvy. Prvním krokem je změna velikosti vstupního obrazu na 448x448, než projde konvoluční sítí. Poté následuje 1x1 konvoluce pro snížení počtu kanálů, po které následuje konvoluce 3x3 pro generování kvádrového výstupu. Aktivační funkce je ve správě ReLU, mimo finální vrstvy, která využívá lineární aktivační funkci. Objevují se zde další techniky, které jsou například dávková normalizace a *dropout*, tyto techniky zabraňují *overfittingu* (37). Architektura YOLO je vyobrazena na Obrázek 12.

Obrázek 12 Architektura YOLO u původní verze



Zdroj: (37)

Model YOLO se neustále vylepšuje a přináší nové možnosti, pro lepší detekci objektů, v Tabulka *2* jsou uvedené jednotlivé verze, které byly představeny.

Tabulka 2 Verze modelu YOLO

Verze	**Rok**	**Informace**
YOLO	2016	Jedná se o původní verzi.
YOLOv2	2016	Nová technika vícestupňové učení, která umožňuje síti předpovídat různé velikosti vstupních dat, a tak dosahovat kompromisu mezi rychlostí a přesností.
YOLOv3	2018	Přidán odhad objektivity do předpovědí *bounding* boxu a zlepšení výkonu na malých objektech přidáním detailů na třech samostatných úrovních.
YOLOv4	2020	Přidány nové funkce pro zlepšení přesnosti a rychlosti konvoluční neuronové sítě.
YOLOv5	2020	Nabízí různé architektury detekce objektů předem vycvičených na datové sadě MS COCO. Tento model je jedním z modernějších s velkou podporou a jednodušším použitím.

Zdroj: (38)

Tabulka 2 Verze modelu YOLO - pokračování

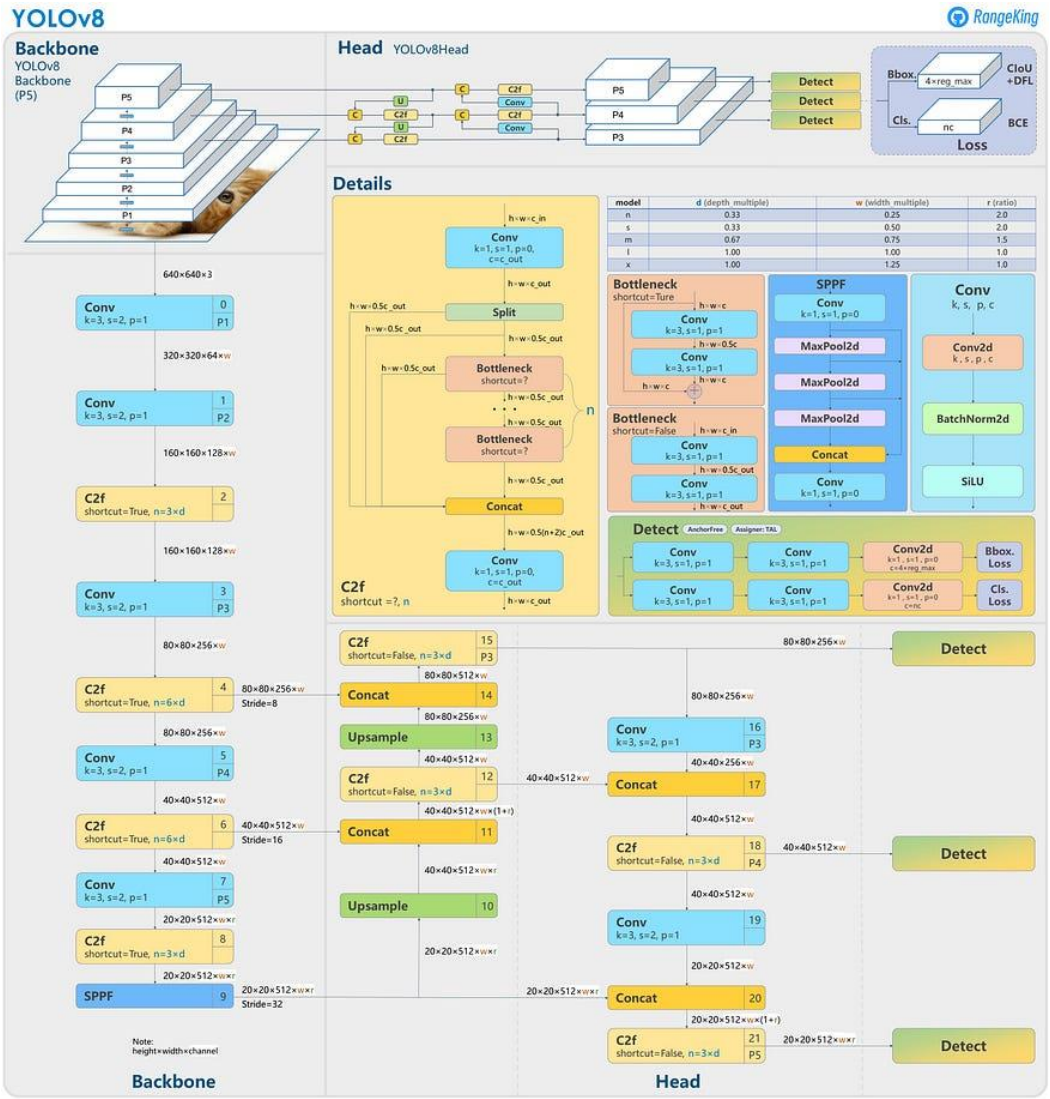
Verze	Rok	Informace
PP-YOLO	2020	Využívá regularizace DropBlock, Matrix NMS.
PP-YOLOv2	2021	Drobné úpravy za cílem zlepšení výkonu, přidání aktivace mish a sítě agregace cest.
YOLOv5-v6.0	2021	V této verzi se objevuje spoustu nových funkcí a oprav. Je možné s touto verzí pracovat i na mobilním telefonu.
YOLOv6	2022	Oproti všem předchozím verzím jsou v této verzi oddělené hlavy, jinými slovy má další vrstvy oddělující vlastnosti od finální hlavy, což empiricky prokázalo zvýšení výkonu.
YOLOv7	2022	V této verzi se bere v úvahu množství paměti potřebné k uložení vrstev v paměti a vzdálenost, kterou potřebuje, aby se gradient rozšířil zpět vrstvami, čím je gradient kratší, tím se síť efektivněji učí.
YOLOv8	2023	Nejnovější model, který podporuje všechny předchozí verze YOLO, umožňuje používat nejnovější technologii YOLO, a přitom stále využívat starší modely YOLO. V této verzi se objevuje nová páteřní síť, detekční hlava bez kotev a ztrátová funkce. Jedná se tedy o vysoce efektivní model, který může běžet i na různém hardwaru, od CPU až po GPU.

Zdroj: (38)

3.4.1.3 YOLOv8

Jedná se o nejnovější model z rodiny YOLO, je vytvořen společností Ultralytics. Architektura YOLOv8 je vyobrazena na Obrázek 13. YOLOv8 disponuje několika výkonnými funkcemi. Dává možnost využít předtrénované modely na datasetu COCO, tyto modely se naučily identifikovat a klasifikovat velký počet objektů. YOLOv8 umožňuje vytvářet vlastní modely, které lze přizpůsobit dle specifickým potřebám pro detekci (39).

Obrázek 13 Model architektury YOLOv8



Zdroj: (40)

3.4.1.4 Single Shot Multibox Detector (SSD)

SSD model je založen na konvoluční síti, která generuje pevnou velikost souboru ohraničujících boxů a skóre přítomnost třídy objektů v těchto boxech. Po tomto kroku následuje potlačení bez maximalizace, což vytváří finální výstup detekce. Prvotní vrstvy SSD si zakládají na standartní architektuře používané pro vysoce kvalitní klasifikaci obrazu (41).

SSD nepoužívá sítě pro návrh regionů, což mu umožnuje zrychlení procesu, díky tomu může být použit v reálném čase. Detekce objektů pomocí SSD lze rozdělit na dvě části:

extrahování mapy objektů a aplikaci konvolučních filtrů pro detekci objektů (42). K extrahování map se používá VGG16, jedná se o typ CNN (43).

3.4.2 Two-stage object detection (Dvoustupňové modely)

Dvoustupňové modely jsou rozděleny do dvou kroků. V první části se extrahuje oblast, ve které se nachází objekt. Poté následuje druhá část, kde dochází ke klasifikaci a dalšímu zpřesnění lokalizace objektů. Jedná se o výkonné modely, ale jsou považovány za pomalé. U dvojstupňových modelů došlo k pokroku se sdílením funkcí, díky tomu mají dvoustupňové modely podobné výpočetní náklady jako jednostupňové modely. Nejznámější dvoustupňové modely jsou: R-CNN, Fast R-CNN, Faster R-CNN, Mask R-CNN, Cascade R-CNN (44).

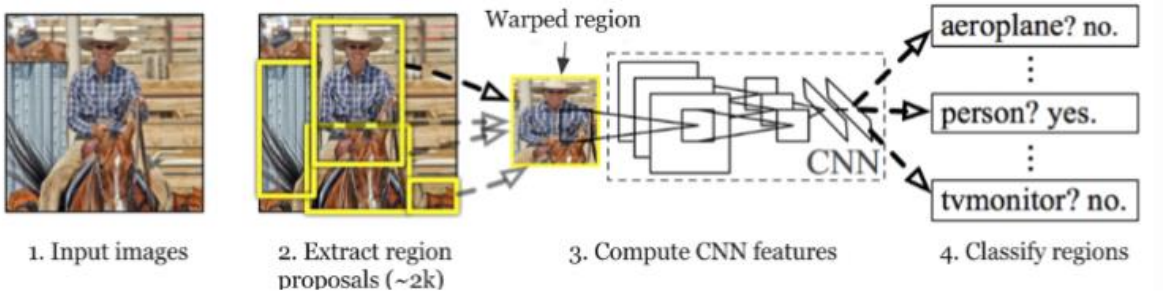
3.4.2.1 R-CNN

R-CNN je metoda hlubokého učení, která využívá konvolučních neuronových sítí ve spojení s generováním regionů. Jedná se o dvoufázový detekční algoritmus, kde v první fázi se identifikuje podmnožina oblastí ve snímku, který mohou obsahovat objekt. V druhé fázi se klasifikuje objekt v každém regionu (45). Na Obrázek 14 je vyobrazený tento postup.

Nevýhody R-CNN jsou následující:

- Dlouhá doba pro natrénování sítě, jelikož se musí klasifikovat 2000 návrhů regionů;
- R-CNN nelze použít v reálném čase, protože testovaní každého snímku trvá okolo 47 sekund;
- Algoritmus selektivního vyhledávání je pevně daný algoritmus, tudíž neprobíhá žádné učení (46).

Obrázek 14 Příklad funkce R-CNN



Zdroj: (46)

3.4.2.2 Fast R-CNN

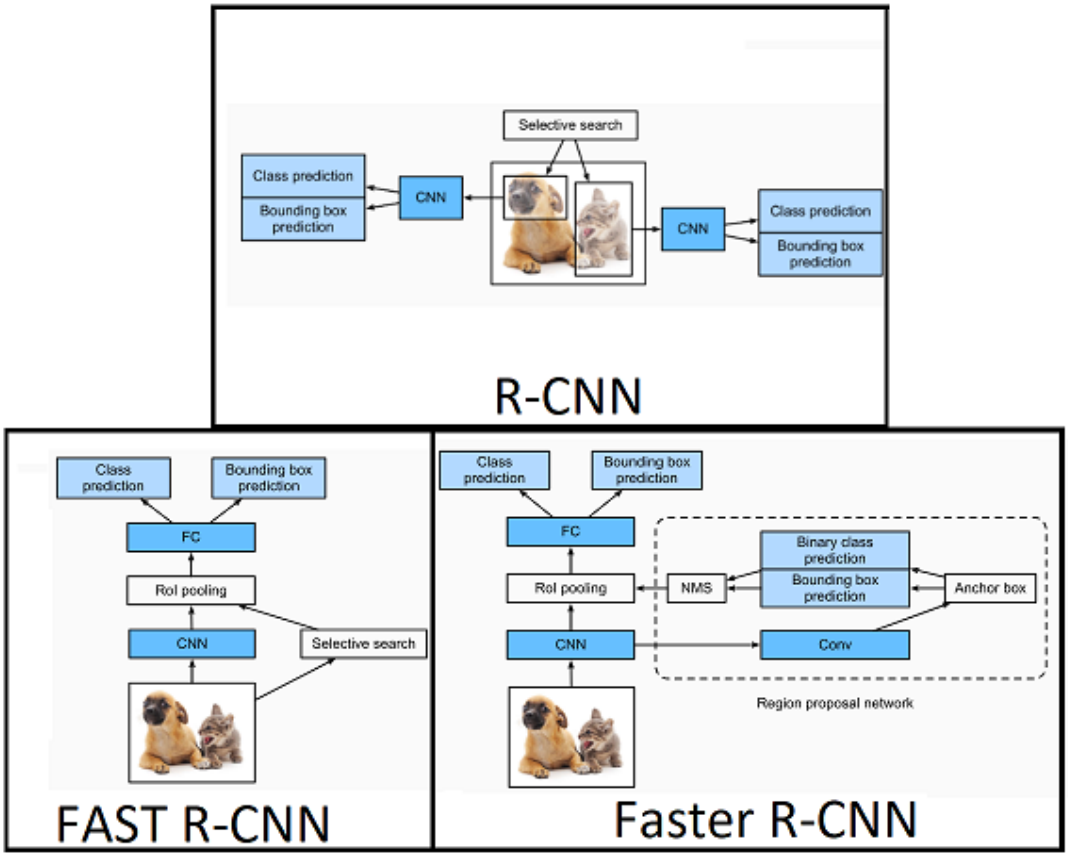
Stejně jak u R-CNN, tak i Fast R-CNN používá ke generování regionů algoritmus. Na rozdíl od R-CNN, který ořezává a mění velikost návrhů oblasti, tak Fast R-CNN zpracovává celý obrázek. Dává dohromady vlastnosti CNN odpovídající každému návrhu regionu. Hlavní výhodou Fast R-CNN je to, že pracuje efektivněji než R-CNN, jelikož prováděné výpočty pro překrývající se oblasti jsou sdílené, kdežto v R-CNN každý region provádí výpočty bez sdílení (45).

3.4.2.3 Faster R-CNN

Fast R-CNN pro zvýšení přesnosti potřebuje generovat mnoho návrhů regionů při selektivním vyhledáním. Pro snížení počtu návrhu regionů bez vzniku ztráty přesnosti, tak Faster R-CNN navrhuje selektivní vyhledávání nahradit generováním návrhů regionů v síti. Generování návrhů v síti je rychlejší a lépe se přizpůsobuje datům (45, 47).

Na Obrázek 15 jsou vyobrazeny architektury jednotlivých R-CNN modelů. Oproti klasickému modelu R-CNN, Fast R-CNN pro objekty s podobnými vlastnostmi využívá stejný výpočetní algoritmus. Faster R-CNN nahrazuje selektivní vyhledávání generování návrhů regionů přímo v síti (47).

Obrázek 15 Architektury jednotlivých R-CNN modelů



Zdroj: (47)

3.4.2.4 Cascade R-CNN

Jedná se o metodu detekce objektů, jenž se pokouší řešit problémy s nižším výkonem při zvyšovaní hodnot IoU. Cascade R-CNN používá místo jednoho klasifikátoru pro detekci objektů tři klasifikátory(kaskády), které jsou navzájem propojeny, díky tomu Cascade R-CNN dosahuje lepších výsledků než původní R-CNN modely (44, 48).

3.4.2.5 Mask R-CNN

Pokud se v trénovací sadě dat na snímcích vyskytují označené pozice objektu na úrovni pixelů, tak může maska R-CNN efektivně využít těchto podrobných značek k dalšímu zvýšení přesnosti detekce objektů (47).

3.5 Nástroje

3.5.1 Detectron

Detectron je AI model pro detekci objektů vyvinutý společností Meta. Jedná se o flexibilní model, který podporuje rychlou implementaci a hodnocení nového vyvíjeného modelu. Detectron umožňuje rychlé nasazení a možné upravení pro trénování na uživatelských vytvořených datasetech (49, 50).

V detectronu jsou implementovány jednotlivé algoritmy pro detekci objektů, mezi nimi jsou:

- Mask R-CNN;
- RetinaNet;
- RPN;
- Faster R-CNN;
- RPN;
- Fast R-CNN;
- R-FCN.

3.6 Metriky pro zhodnocení výsledků detekce objektů

Jedná se o kvantitativní měření, které slouží k vyjádření stavu určitého systému. Mohou být využity k posouzení kvality, efektivity a výkonnosti určitého systému. Interpretace metrik může probíhat v číslech nebo v grafické podobě pomocí jednotlivých grafů (51).

3.6.1.1 Precision (P)

Jedná se o klasifikační metriku, která je známá jako precision a vyjadřuje poměr mezi počtem správně identifikovaných objektů a celkovým počtem objektů. Tato metrika slouží k posouzení schopnosti modelu odlišit skutečně hledané objekty od nesouvisejících objektů či pozadí (52). Precision lze určit na základě vztahu:

$$P = \frac{TP}{\mathrm{TP} + \mathrm{FP}} \tag{1}$$

Kde

P je precision;

TP je true positives;

FP je false positives (52).

3.6.1.2 Accuracy (A)

Metrika, která poskytuje informaci o tom, jak často model předpověděl správný výsledek. Tato metrika vyjadřuje poměr mezi počtem správně identifikovaných objektů ke všem predikcím. Metriku accuracy lze využít k hodnocení celkové kvality výkonu modelu (52). Accuracy lze určit na základě vztahu:

$$A = \frac{\mathrm{CP}}{\mathrm{AP}} \tag{2}$$

kde

A je accuracy;

CP je correct predictons;

AP je all prediction (52).

3.6.1.3 Recall (R)

Metrika, která vyjadřuje poměr počtu skutečně pozitivních výsledků k celkovému počtu skutečných objektů. Metrika recall měří schopnost modelu zachytit všechny skutečné objekty na obrázku (53, 54). Recall lze určit na základě vztahu:

$$Recall = \frac{\text{True Positive}}{True\ Positive + False\ Negative} \tag{3}$$

kde

R je recall;

TP je true positive;

FN je false negative (53, 54).

3.6.1.4 F1 score

Jedná se o kombinaci precision a recall, obě hodnoty mají stejnou váhu. Používá se k posouzení vyváženostmi mezi schopností modelu správně identifikovat pozitivní případy a minimalizovat špatné pozitivy a falešné negativy. Rozmezí výsledné hodnoty je 0-1, kde 1 je maximální a 0 minimální (55). F1 score lze určit na základě vztahu:

$$F1 = \frac{2}{\frac{1}{P} + \frac{1}{R}} \tag{4}$$

kde

F1 je F1 score;

P je precision;

R je recall (55).

3.6.1.5 Mean Average Precision (mAP)

Metrika mAP se používá k analýze výkonu systémů detekce objektů a segmentace. Vzorec mAP je založen na těchto metrikách: Confusion Matrix, IoU (Intersection over Union), Recall, Precision (56). Jedná se o kombinaci AP (average precision) přes různé třídy a prahy spolehlivosti. Pro každou třídu a práh spolehlivosti je vypočtena AP, jenž nám dává informaci, jak dobře model detekuje objekty dané třídy. Následně jsou tyto AP hodnoty pro jednotlivé třídy, spočteny pomocí aritmetického průměru (54). mAp lze určit na základě vztahu:

$$mAP = \frac{1}{N}\sum_{i=1}^{N} AP_i \tag{5}$$

kde

mAP je mean Average Precision;

N je počet tříd;

AP je průměrná přesnost pro jednu třídu (54).

3.7 Obdobné práce s modely pro detekci GUI

3.7.1 Datové sady

3.7.1.1 Dataset 1

Rico dataset obsahuje 66 261 grafických uživatelských rozhraní. Dataset byl redukován o 15 737 na finální počet 50 524 snímků. Tyto snímky pocházejí z 8 018 mobilních aplikací pro Android. Tato GUI obsahuje 923 404 prvků GUI, z nichž jsou 426 404 netextové prvky a 497 000 textových prvků (3).

Dataset byl rozdělen do trénovací/ validační/testovací sady s poměrem rozdělení 8:1:1 (40 K: 5 K: 5 K). Všechna grafická uživatelská rozhraní jsou pouze v jednom rozdělení, aby nedošlo ke zkreslení (3).

Celkem je v datasetu definováno 15 tříd objektů: *Button, Spinner, Chonometer, ImageView, RatingBar, ProgressBar, VideoView, ToggleButton, RadioButton, Switch, Checkbox, ImageButton, TextView, EditText, SeekBar* (3).

3.7.1.2 Dataset 2

VINS dataset obsahuje 4800 snímků grafických uživatelských rozhraní. Byl redukován o 257 snímků na celkový počet 4543 snímků. Následně byl dataset rozdělen do tří sekcí trénovací/validační/testovací v procentovém poměru 80 %, 10 %, 10 % (57).

Celkem je v datasetu definováno 12 tříd: *Backround image, Checked View, Icon, Input Field, Image, Text, Text Button, Slidding Menu, Page Indicator, Pop-Up Window, Switch, Upper Task Bar* (57).

3.7.2 Použité metody

V obdobných studií se zkoumají metody detekce objektů GUI, mezi nimiž jsou Staromódní metody a metody založené na hlubokém učení. Mezi staromódní metody patří detekce hran a metody založené na šablonách. Mezi metody založené na hlubokém učení patří jednostupňové a dvoustupňové modely (3).

V rámci analýzy detekce GUI byla implementována a porovnána škála metod. Tyto metody zahrnují kombinaci OCR a detekce hran. Metody REMAUI a XIANYU využívají

detekci hran k identifikaci prvků GUI. U metod FASTER RCNN, YOLOv3 a CenterNet se využívá CNN k detekci a klasifikaci prvků GUI (3, 57).

3.7.2.1 REMAUI

Pro textové prvky používá nástroj OCR. U netextových prvků detekuje strukturální hrany prvků GUI pomocí nástroje Cannyho detekce hran, pomocí Gaussova filtru k vyhlazení a snížení šumu následuje vícestupňová filtrace k identifikaci skutečných hran. Na závěr REMAUI provede sloučení hran, získá obrysy a získá ohraňující rámeček (3).

3.7.2.2 XIANYU

Binarizuje obrázek a provede horizontální/vertikální řezání. Používá detekci hran pomocí Laplaceova filtru a získá obrys v binarizovaném obraze. Pomocí algoritmu food fill identifikuje propojené body oblastí a odfiltruje šum z pozadí (3).

3.7.2.3 FASTER RCNN

Princip této metody byl již popsán v kapitole Faster R-CNN.

3.7.2.4 YOLOv3

Princip této metody byl již popsán v kapitole YOLO (You Only Look Once).

3.7.2.5 CenterNet

Jedná se o one-stage model, který předpovídá polohu levého horního, pravého dolního rohu a středu objektu a následně z toho sestaví ohraňující rámeček (3).

3.7.2.6 Tesseract

Jedná se o nástroj OCR pro texty dokumentů, jenž se skládá z detekce textových řádků a rozpoznávání textu. Tesseract převede obrázek na binární mapu a následně provede analýzu spojitých komponentů, aby našel obrysy prvků. V posledním kroku Tesseract model sloučí řádky textu, které se horizontálně alespoň z poloviny překrývají (3).

3.7.2.7 East

Funguje na principu hlubokého učení pro detekci prvků. Vstupní obrázek je zpracován v pyramidové síti, což umožňuje zpracování obrazu v různých měřítkách. Následně se pro každý bod vypočítává šest hodnot na základě vlastností a informací získaných z vstupního obrázku (3).

3.7.2.8 Yolov5

Princip této metody byl již popsán v kapitole YOLO (You Only Look Once).

3.7.3 Výsledky

Výsledky 5 metod pro detekci oblastí netextových prvků GUI jsou vyobrazeny v Tabulka *3*, kde jsou zobrazeny výsledky jednotlivých metrik při prahové hodnotě IoU>0,9. Modely jsou trénovány na Dataset 1 (3).

Tabulka 3 Tabulka výsledných metrik pro jednotlivé modely z obdobné studie při IoU>0,9

Method	#bbox	Precision	Recall	F1
REMAUI	54 903	0,175	0,238	0,201
XIANYU	47 666	0,142	0,168	0,154
Faster-RCNN	39 995	0,440	0,437	0,438
YOLOv3	36 191	0,405	0,363	0,383
CenterNet	36 096	0,424	0,380	0,401

Zdroj: (3)

Při dalším pokusu byly tři modely (Faster-RCNN, YOLOv3, CenterNet) trénovány na datech z Dataset 1, při třech velikostech trénovacích sad, konkrétně 2000, 10 000 a 40 000 obrázků v sadě. Validace pokaždé probíhala na validačním datasetu, který obsahuje 5 000 obrázků. V Tabulka 4 jsou vyobrazeny výsledky jednotlivých modelů při trénování na různých velikostech trénovací sady. Z výsledků je zřejmé, že s velikostí trénovací sady se zlepšují výsledky jednotlivých metrik (3).

Tabulka 4 Výsledky při různé velikosti trénovací sady

Method	Size	Precision	Recall	F1
Faster-RCNN	2 000	0,361	0,305	0,331
	1 000	0,403	0,393	0,398
	40 000	**0,404**	**0,437**	**0,438**
YOLOv3	2 000	0,303	0,235	0,265
	10 000	0,337	0,293	0,313
	40 000	**0,405**	**0,363**	**0,383**
CenterNet	2 000	0,319	0,313	0,316
	10 000	0,328	0,329	0,329
	40 000	**0,424**	**0,380**	**0,401**

Zdroj: (3)

Pro výběr vhodného modelu pro detekci textu v GUI byly porovnány modely Tesseract, EAST, REMAUI, Xianiyu. Všechny zkoumané modely byly trénovány na Dataset 1. Všechny zkoumané modely používají nástroj OCR, jenž je popsán v Optical character recognition (OCR). Výsledky jednotlivých metrik pro dané modely jsou vyobrazeny v Tabulka *5* zdaleka nejlepších výsledků dosáhl model East (3).

Tabulka 5 Vyhodnocení modelu pro detekci textu v GUI

Method	Precision	Recall	F1
Tesseract	0,291	0,518	0,372
EAST	**0,402**	**0,720**	**0,516**
REMAUI	0,297	0,489	0,369
Xianyu	0,272	0,481	0,348

Zdroj: (3)

Porovnání dvou modelů, Bunian, YOLOv5, bylo provedeno na základě tréninku obou modelů na Dataset 2. Oba modely byly trénovány pro detekci 12 tříd. Výsledky porovnání jsou zobrazeny v Tabulka *6*. Zjistilo se, že ve 10 z 12 případů dosáhl model YOLOv5 lepších model. Pro všech 12 tříd dosáhl s celkovou přesnosti pro mAP50 92,1 %, což je o 15,7 % více než Bunianův model. (57)

Tabulka 6 Porovnání modelů při mAP50

Class	Bunian	YOLOv5
All 12 classes	76,4	92,1
Background image	89,3	84,6
Checked view	44,5	67,6
Icon	50,5	89,6
Input field	78,2	94,6
Image	79,2	83,9
Text	64,0	96,3
Text button	87,4	98,8
Sliding menu	100	99,5
Page indicator	59,4	98,0
Pop-Up window	93,8	95,5
Switch	80,0	97,2
UpperTaskBar	90,4	99,5

Zdroj: (57)

4 Vlastní práce

Klíčovým rozdílem mezi zkoumanými modely a již stávajícími modely uvedenými v teoretické části je použití nejnovější a aktuálně nejmodernější architektury pro detekci prvků, kterou je YOLOv8 3.4.1.2. Na rozdíl od předchozích modelů, které jsou zmíněny v teoretické části, novější architektura YOLOv8 nabízí pokročilejší funkce a vylepšení.

Vlastní práce se zaměřovala na konkrétní implementaci a následné experimentováním s navrženými modely YOLOv8 pro detekci GUI. Při nastavení jednotlivých hyperparametrů se studie inspirovala z obdobných studií. Postupně se snažila optimalizovat stávající modely pomocí změn trénovacích hyperparametrů za účelem získání lepších výsledků. Získaná výsledná data byla vyhodnocena pomocí jednotlivých metrik a následně srovnána s výsledky dosaženými v obdobných pracích. Na závěr proběhlo zhodnocení vybraného modelu a porovnání s výsledky z obdobných prací. Pro práci s modelem byl použit programovací jazyk Python a framework Pytorch.

4.1 Příprava dat

4.1.1 Datový soubor

Pro tuto práci byl vybrán dataset VINS, který byl následně upraven na stránce roboflow. Dataset byl rozdělen do tří částí: trénovací, validační a testovací. Pro trénování bylo vyhrazeno 3627 obrázků (80 % z celkového datasetu). Pro validaci bylo vyhrazeno 462 obrázku (10 % z celkového datasetu) a pro testování 453 obrázků (10 % z celkového datasetu)

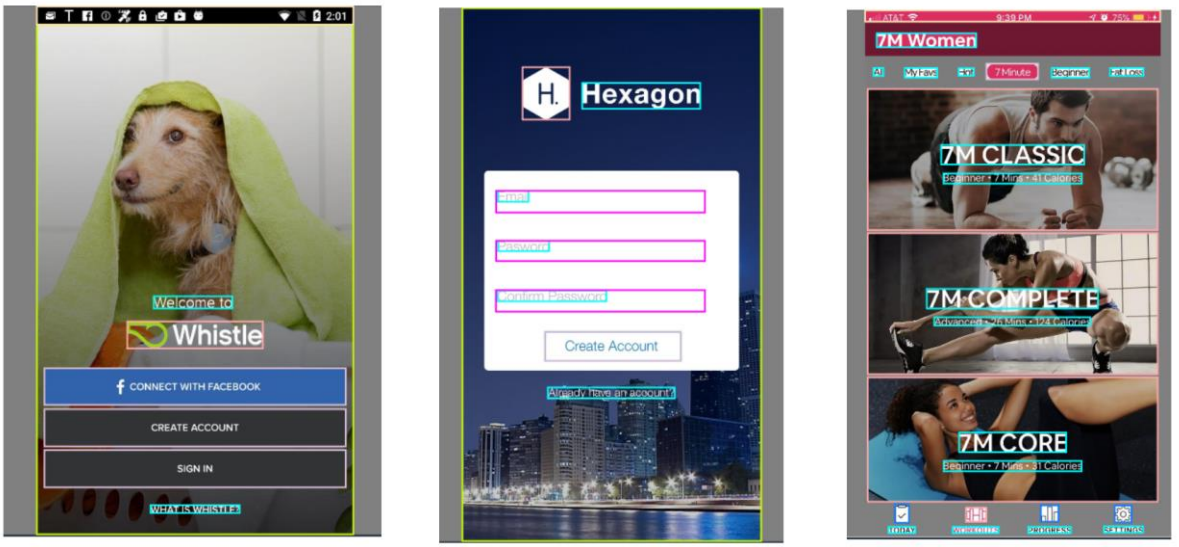
Tento dataset Vins byl vybrán a takto upraven z 4800 obrázku na 4542 a rozdělen v poměru (80 % / 10 % / 10 %), což odpovídá úpravám provedeným v obdobné studii (v kapitole Dataset 2). Díky tomu je možné výsledné modely porovnávat s modely z této obdobné studie.

4.1.2 Anotace a označení dat

Data jsou anotována ve formátu YOLOv8 a obsahují informace o 12 třídách. Třídy jsou pojmenovány následovně: *Text, Icon, Image, TextButton, UpperTaskBar, EditText, PageIndicator, CheckedTextView, BackgroundImage, Modal, Toolbar, Switch.*

Každý obrázek v datasetu má příslušný anotační soubor v textovém formátu. V tomto souboru je každý *bounding* box popsán na samostatném řádku, kde první hodnota označuje třídu a následující hodnoty definují okraje *bounding* boxu. Obrázky z datasetu spolu s jejich *bounding* boxy jsou zobrazeny na Obrázek *16*.

Obrázek 16 Obrázky z datasetu i s *bounding* boxy



Zdroj: (58)

4.1.3 Předzpracování dat

Anotace je sestavena ve formátu YOLOv8 pro PyTorch. Byla použita metoda Preprocessing, kde byl aplikován Auto-Orient. Na datasetu není provedená žádná augmentace, což znamená, že počet obrázků v datasetu nebyl zvýšen pomocí dalších uměle vytvořených obrázků z již existujících.

4.2 Návrh a implementace modelu

4.2.1 Výběr architektury

Během trénování pracováno se třemi varianty architektury: YOLOv8n, YOLOv8s, YOLOv8m. Každá z těchto architektur má odlišný počet vrstev a velikost parametrů. Větší modely jsou časově náročnější na trénování.

4.3 Trénink modelu

Model YOLOv8 není původně předtrénován k detekci prvků GUI, proto bylo potřeba provést trénování modelu na daném datasetu obsahujícím prvky GUI, aby byl schopen detekovat jednotlivé prvky grafického uživatelského rozhraní.

4.3.1 Definice trénovaných hyperparametrů

Definování trénovacích hyperparametrů je klíčové pro dosažení požadovaných výsledků, jelikož hyperparametry stanovují výchozí bod pro začátek trénování. Prvotní nastavení hyperparametrů bylo inspirováno nastavením z obdobných prací, toto nastavení je zobrazeno v Tabulka *7*.

Tabulka 7 Definice parametrů

Název	Popis	Nastavení pro první trénování
Epochs	Uvádí kolikrát se celý trénovací dataset projde během trénování. Vyšší počet může vést k lepší konvergenci, ale může se stát, že dojde k přetrénování modelu.	epochs=160
Batch	Určuje počet obrázků, které jsou zpracovány současně v jedné iteraci trénování (určuje velikost dávky).	Batch=8
Data	Obsahuje konfiguraci pro dataset, jenž je používaný při trénování.	data=trainGUI.yaml
Optimizer	Jedná se o algoritmus, který je použit při aktualizaci parametrů během tréninku.	Optimazer=Adam
Model	Specifikuje cestu k předtrénovaným vahám. Předtrénované váhy jsou použity jako výchozí bod před začátkem trénování, mohou urychlit proces trénování a zlepšit výsledky.	model=yolov8n.pt
Patience	Hyperparametr, který slouží k předčasnému ukončení trénování, pomáhá zamezit přeučení modelu. Patience je známý také pod pojmem „Early stopping“. K předčasnému ukončení dojde, pokud se model přestane po určitý počet epoch zlepšovat.	Patience=10

Zdroj: (59)

4.3.2 Proces tréninku

Před spuštěním procesu trénování byla inicializována služba Comet pro sledování a záznam experimentu. Comet se používá pro zaznamenávání a vizualizaci různých metrik a poskytne informaci o průběhu trénování a vyhodnocování modelu. API klíč se získá na přihlášeném účtu služby Comet.

Prvotní nastavení hyperparametrů byla provedeno s ohledem na předchozí studie a bylo nastaveno následujícím způsobem: na 160 epoch s velikostí dávkování (cache) 8 obrázku, pro vstupní váhy byl použit yolov8n.pt (model=yolov8n.pt), byl nastaven „early stopping“ (patience) na 10, byl nastaven optimizer Adam.

4.4 Optimalizace a zlepšení modelu

Zlepšení modelu bylo posuzováno na základě získaných výsledných metrik P, R, mAP50, mAP50-95, které po dokončení daného trénování byly získány z výsledků z testovacího datasetu. Celkem bylo provedeno 9 trénování s rozdílnými hyperparametry. Nastavení jednotlivých hyperparametrů pro jednotlivé trénování modelu je vyobrazeno v Tabulka *8*.

Tabulka 8 Nastavení hyperparametrů pro jednotlivé trénování modelu

model	Epoch	Batch	model	patience	Imgsz
1	160	8	yolov8n.pt	10	1280
2	160	8	yolov8s.pt	10	1280
3	160	8	yolov8m.pt	10	1280
4	160	16	yolov8n.pt	10	1280
5	160	16	yolov8s.pt	10	1280
6	160	16	yolov8m.pt	10	1280
7	190	16	yolov8s.pt	20	1280
8	190	16	yolov8m.pt	20	1280
9	190	32	yolov8s.pt	20	1280

Zdroj: (Vlastní zdroj)

Postup byl následující: nejdříve se provedl první trénink modelu s hyperparametry: epoch=160, batch=8, model= yolov8n.pt, patience=10. Pro následující dva tréninky (modely 2 a 3) s velikostí modelů yolov8s a yolov8m. Poté následovaly další tři tréninky (modely 4,

5 a 6), kde se změnily vstupní hyperparametry: batch na 16 a bylo provedeny pro tři velikosti modelů (yolov8n, yolov8s, yolov8m), ostatní hyperparametry zůstaly nezměněny. Pro následující dva tréninky (modely 7 a 8) byl počet epoch zvednut z 160 na 190 a patience byl zvednut z 10 na 20 a byly vybrány velikosti modelu yolov8s a yolov8m, zbylé hyperparametry zůstaly nezměněny. Pro poslední trénink (model 9) byl rozšířen batch z 16 na 32, velikost modelu byla vybrána yolov8s a ostatní hyperparametry zůstaly zachovány.

Všechny tréninky modelů, které měli nastaven batch na 8, skončili předčasně díky „early stopingu“ (po určitý počet epoch [10] nedošlo ke zlepšení modelu). U 1. trénování bylo provedeno celkem 64 epoch. 2. trénink byl zastaven po 53 epoch a 3. trénink byl ukončen po provedení 70 epoch. Jediný trénink (model 6), který také skončil předčasně a měl jinou hodnotu batch než 8, byl ukončen po 23 epochách. U tohoto tréninku byl počet epoch 160, batch byl nastaven na 16, byl vybrán model yolov8n.pt a patience byla nastavena na 10.

5 Výsledky a diskuse

5.1 Výsledky

Výsledky jednotlivých metrik pro trénované modely jsou vyobrazeny v tab. 6. Na základě dosažených výsledků byly vybrány modely s nejlepšími výsledky pro další porovnání. Nejlepší dosažené jednotlivé metriky jsou v Tabulka *9* zvýrazněny žlutě. Nejlépe si vedly tři modely: model 5, model 8 a model 9, přičemž model 8 dosáhl nejlepších výsledků ve dvou ze čtyř metrik ve srovnání s ostatními modely.

Tabulka 9 Výsledné metriky pro jednotlivé modely

Model	P [%]	R [%]	mAP50 [%]	mAP50-95 [%]
1	81,2	85,2	87,5	72,8
2	76,9	81,7	83,7	70,8
3	83,3	83,6	87,4	73,8
4	88,4	89,1	92,7	82,4
5	89,6	89,7	93,1	83,2
6	64,7	73,1	73,5	59,1
7	88,8	89	92,8	82,7
8	89,3	89,6	93,2	83,7
9	88,5	91,6	92,7	82,9

Zdroj: (Vlastní zdroj)

Z trénovaných devíti modelu byly vybrány tři nejlepší, mezi tyto modely se dostaly modely 5, 8 a 9, tyto modely měli následující nastavený vstupních hyperparametrů:

- Model 5: epoch=160, batch=16, model=yolov8s.pt, patience = 10, imgsz=1280;
- Model 8: epoch=190, batch=16, model=yolov8m.pt, patience=20, imgsz=1280;
- Model 9: epoch=190, batch=32, model=yolov8s.pt, patience=20, imgsz=1280.

Tyto tři modely byly vybrány, protože každý z nich měl alespoň jednu výslednou metriku nejlepší v porovnání se všemi ostatními trénovanými modely, jak je uvedeno v Tabulka *9*.

5.1.1 Precision (Přesnost)

U metriky precision dosáhl nejlepších výsledků model 5 s 89,6 %. Porovnání této metriky pro všechny tři modely je vyobrazeno v Tabulka *10*. Všechny tři modely jsou

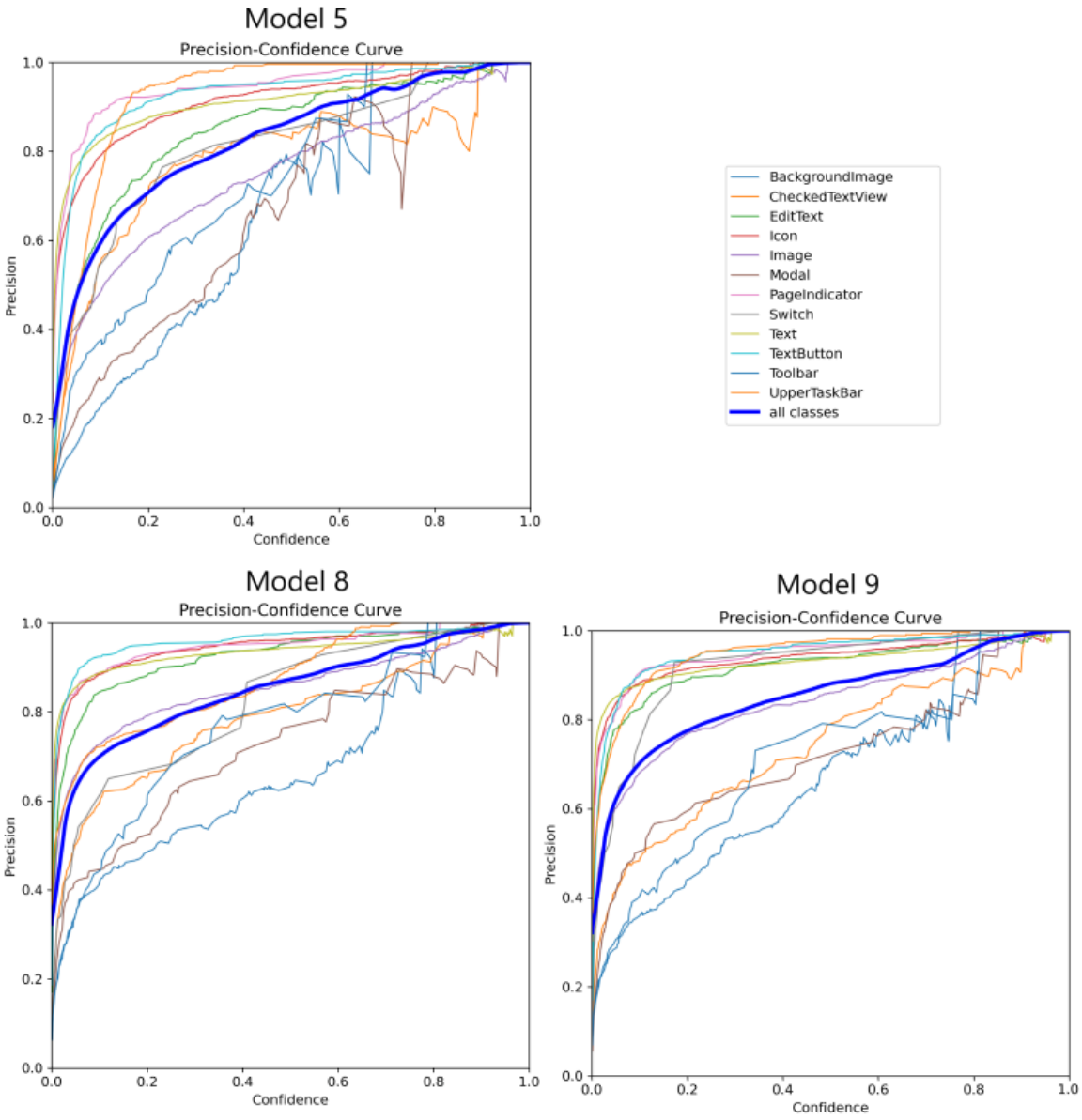
vyobrazeny v Graf *1* kde, na ose x je confidence(Důvěra) a na ose y je precision(Přesnost). Z tohoto grafu vyplývá, že nejlépe si vedl model 9, následuje model 8, nejhůře se skončil model 5.

Tabulka 10 Precision (Přesnost)

Model	Precision [%]
5	89,6
8	89,3
9	88,5

Zdroj: (Vlastní zdroj)

Graf 1 Precision-Confidence



Zdroj: (Vlastní zdroj)

5.1.2 Recall (Vratnost)

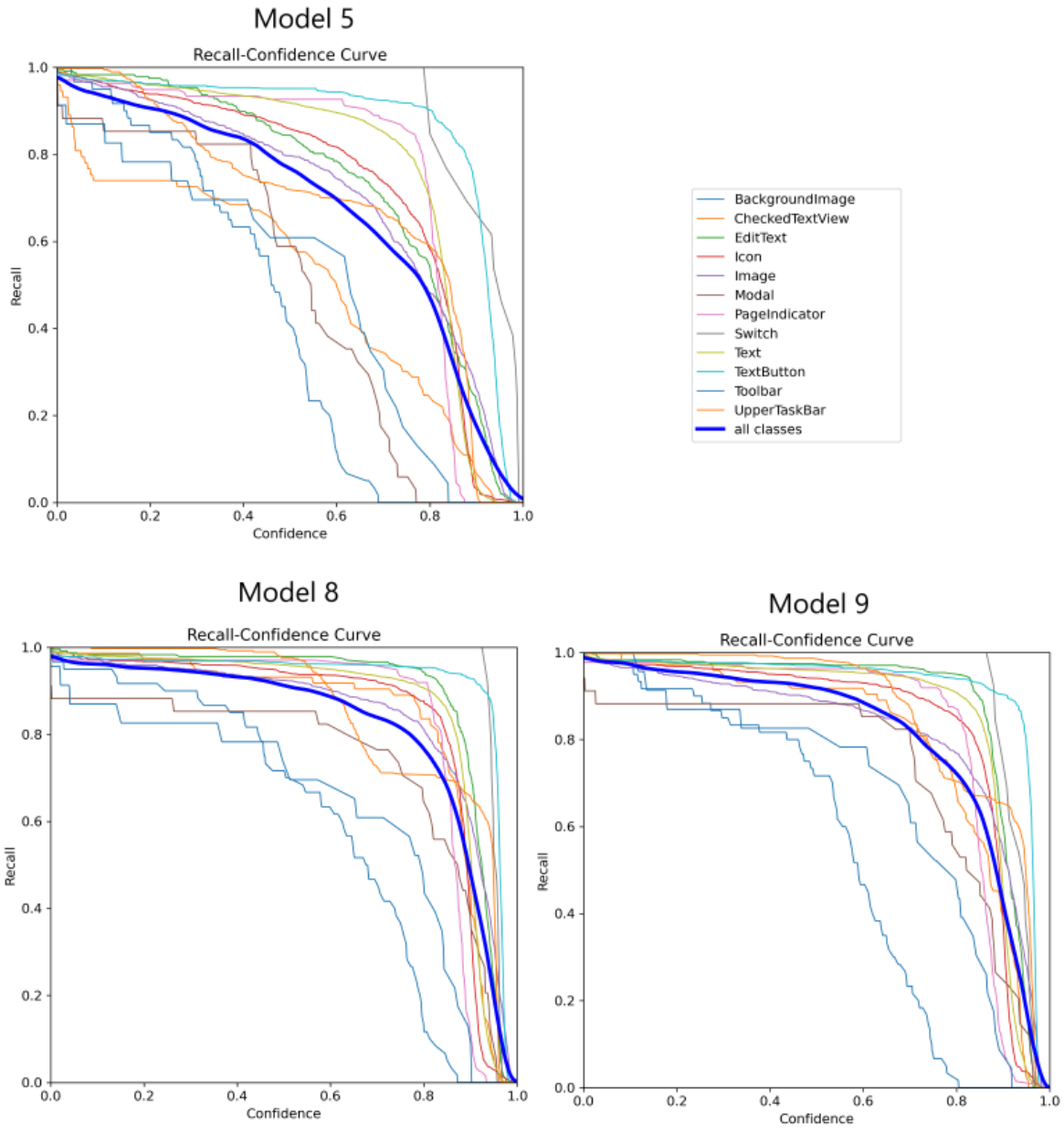
U metriky recall dosáhl nejlepších výsledků model 9 s hodnotou 91,6 %. Tabulka pro porovnání třech nejlepších modelů je vyobrazena v Tabulka *11*. Všechny tři modely jsou vyobrazeny v Graf 2, kde na ose x je Confidence(Důvěra) a na ose y je Recall(Vratnost). Z tohoto grafu vyplývá, že si nejlépe vedl model 8, následovaný modelem 9 a nejhůře model 5.

Tabulka 11 Recall (Vratnost)

Model	Recall [%]
5	89,7
8	89,6
9	91,6

Zdroj: (Vlastní zdroj)

Graf 2 Recall-Confidence



Zdroj: (Vlastní zdroj)

5.1.3 mAP (mean average precision)

Model 8 v této metrice vychází nejlépe v obou možnostech. Pro mAP s IoU=0,5 dosáhl přesností 93,2 % a pro mAP s IoU=0,5-0,95 byla výsledná přesnost 83,7 %. Tabulka pro porovnání metriky mAP mezi jednotlivými modely je vyobrazena v Tabulka *12*.

Tabulka 12 Mean average precision (mAP)

Model	mAP50 [%]	mAP50-95 [%]
5	93,1	83,2
8	93,2	83,7
9	92,7	82,7

Zdroj: (Vlastní zdroj)

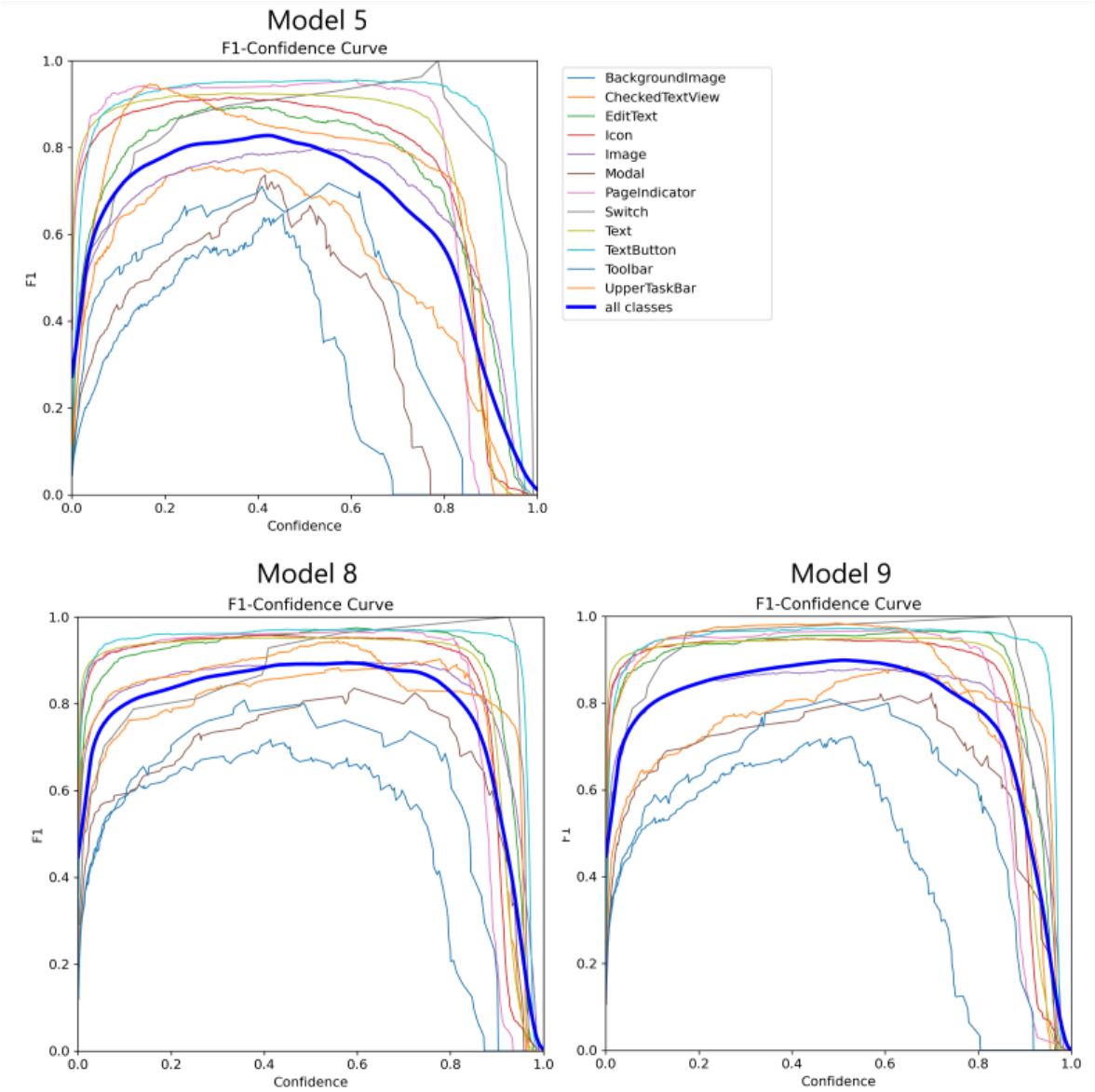
5.1.4 F1 – score

V Graf 3

Zdroj: (Vlastní zdroj)

jsou zobrazeny hodnoty F1 pro jednotlivé třídy a také celkové hodnoty F1 pro všechny třídy dohromady. Z grafu je patrné, že model 5 dosáhl nejhorších výsledků, zatímco model 9 dosáhl nejlepších výsledků, model 8 se v tomto grafu velmi přibližuje k modelu 9.

Graf 3 F1-Confidence



Zdroj: (Vlastní zdroj)

5.1.5 Výsledky jednotlivých tříd

Dle výsledných tabulek a grafů vychází celkově nejlépe model 8, pro který jsou v Tabulka *13* uvedeny všechny výsledky jednotlivých metrik pro všechny třídy. Model 8 dosáhl celkové přesnosti 93,2 % pro mAP50. Vyskytují se zde dvě třídy: *BackgroudImage,*

Toolbar, které dosáhly nižší přesnosti, jedním z důvodů může být nízké zastoupení těchto tříd v trénovacím datasetu.

Tabulka 13 Výsledky jednotlivých metrik pro všechny třídy u modelu 8

Class	Images	Instances	Precision [%]	Recall [%]	mAP50 [%]	mAP50-95 [%]
All	453	6195	83,9	89,6	93,2	83,7
BackgroundImage	453	60	66,6	66,7	75,2	41,3
CheckedTextView	453	73	83,9	91,8	95,3	85,1
EditText	453	237	96,6	97,9	98,4	94,9
Icon	453	1089	96,8	93,8	98,2	87,8
Image	453	778	88,6	90,9	95,3	90,8
Modal	453	34	77,9	82,8	81,6	67,9
PageIndicator	453	136	96,3	97	97,8	81,6
Switch	453	13	93	100	99,5	96,4
Text	453	2901	94,9	95,2	97,3	90,2
TextButton	453	473	98,1	96,2	98,5	98,2
Toolbar	453	23	84,2	69,5	82,9	80,8
UpperTaskBar	453	378	94,4	93,7	98,5	89

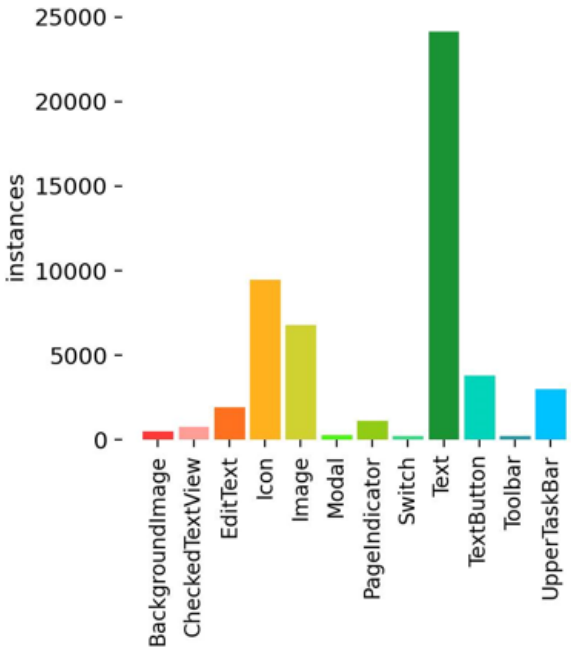
Zdroj: (Vlastní zdroj)

5.1.6 Identifikace slabých stránek modelu

Některé třídy v datasetu byly nedostatečně zastoupené, což mohlo vést ke snížení přesnosti správné identifikace daného prvku, jelikož se reprezentace těchto tříd v trénovacím datasetu objevila v mnohem menším počtu. Zastoupení jednotlivých tříd v trénovací sadě je vyobrazeno v Graf 4.

Jak již bylo zmíněno v sekci Výsledky jednotlivých tříd, nižší přesnost byla dosažena u třídy *BackgroudImage* a *Toolbar*, tyto obě třídy jsou v datasetu méně zastoupené. Pro zvýšení přesnosti detekce těchto tříd by bylo vhodné zvýšit počet anotací daných tříd v trénovacím datasetu.

Graf 4 Počet anotací jednotlivých tříd pouze v trénovací sadě



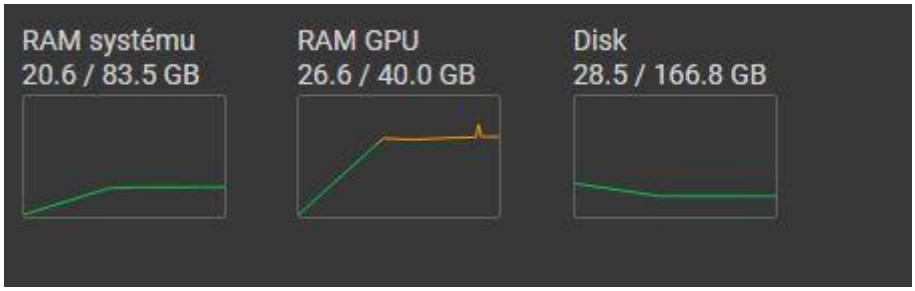
Zdroj: (Vlastní zdroj)

5.2 Diskuse

5.2.1 Omezení

Trénování bylo prováděno na osobním počítači a v Colab. Nevýhoda osobního počítače je limitace u jednotlivých součástek počítače. Počítač, který byl využit má pouze 16 GB RAM a GPU GTX 1650, což není optimální pro trénování. Už jen pokud se trénovalo s batch = 16 (dávkování) a modelem YOLOv8s nebo YOLOv8m, tak byla vyžadována RAM alespoň 17 GB, což bohužel osobní počítač nedokázal splnit. Obrázek *17* zobrazuje, kolik paměti bylo potřeba pro trénování s modelem YOLOv8m a s batch=16. Toto trénování bylo prováděno již v prostředí Colabu.

Obrázek 17 Velikost paměti použit pro trénování



Zdroj: (Vlastní zdroj)

Omezení v Colabu byly dvě. Prvním omezením byla časová limitace pro běh procesu, která platila jak pro verzi zdarma (maximálně 12 hodin), tak i pro placenou verzi (maximálně 24 hodin). I když placená verze poskytovala delší dobu běhu, občas se stávalo, že došlo k neočekávanému odpojení běhu, což vedlo ke ztrátě veškerého průběhu daného tréninku. Při ztrátě průběhu se musel celý trénink opakovat od začátku.

5.2.2 Možnosti zlepšení a budoucím výzkum

Odstranění slabých stránek modelu (jak je popsáno v 5.1.6) představuje možný krok ke zlepšení stávajícího modelu. Možnost přidání anotací tříd, které jsou nedostatečně zastoupené, do trénovacího datasetu.

V průběhu práce na této studii se vyvíjí nový model YOLOv9, který představuje vylepšenou verzi stávajícího YOLOv8. Práce s modelem YOLOv9 může poskytnout lepší výsledky a přesnost detekce prvků GUI. Využitím nového modelu YOLOv9 by mohla studie pokračovat a přispět k rozvoji stávajícího výzkumu.

5.2.3 Porovnání s obdobnými pracemi

V této studii bylo zkoumáno 10 tříd, které jsou stejné jako ty, které se objevují v obdobných pracích. Výsledné porovnání je zobrazeno v Tabulka 14. Při Celkovém vyhodnocení dosáhl model YOLOv8 (vlastní řešení, konkrétně model 8) nejlepšího výsledku s celkovou přesností pro porovnávané třídy 95,3 % (pro mAP50). Za ním následuje

model YOLOv5 s přesností 91 %. Nejhůře vyšel z porovnání model Bunian s hodnocením 72,3 %. Model YOLOv8 (vlastní řešení, konkrétně model 8) vyšel 7 z 11 porovnání nejlépe.

Tabulka 14 Porovnání jednotlivých modelů pro mAP50

Class	YOLOv8 (vlastní řešení, model 8)	Bunian (obdobná studia)	YOLOv5 (obdobná studie)
All 10 classes	95,3	72,3	91
Background Image	75,2	89,3	84,6
CheckedTextView	95,3	44,5	67,6
EditText	98,4	78,2	94,6
Icon	98,2	50,5	89,6
Image	95,3	79,2	83,9
PageIndicator	97	59,4	98
Switch	99,5	80	97,2
Text	97,3	64	96,3
TextButton	98,5	87,4	98,8
UpperTaskBar	98,5	90,4	99,5

Zdroj: (Vlastní zdroj)

6 Závěr

První dílčí cíl se zabýval analýzou současného stavu v oblasti computer vision zaměřením na analýzu GUI. V praktické části, konkrétně v kapitole 3.3, byly představeny jednotlivé metody computer vision používané v této oblasti. Kapitola 3.4 se pak věnovala přehledu modelů pro detekci objektů a v závěru praktické části, v kapitole 3.7 byly popsány modely, jež byly použity obdobných prací, které pomocí computer vision detekují prvky GUI.

Druhý dílčí cíl se soustředil na experimentální implementaci vybraných modelu/modelů pro rozpoznávání prvků GUI. Tato implementace byla prováděná v kapitole 4.4, kde byla provedena optimalizace existujících modelů pomocí změn jednotlivých hodnot vstupních hyperparametrů.

Třetí dílčí cíl se zabýval výběrem vhodných metrik pro porovnání. Vhodné metriky se probíraly v praktické části v kapitole 3.6 a byly následně použity pro porovnání modelů v obdobných prací (3.7.3). Tyto metriky byly důležité při vyhodnocení všech modelů během optimalizace v kapitole 4.4 a následně proběhlo pomocí těchto metrik vyhodnocení třech nejlepších modelů 5.1.

Hlavním cílem této práce bylo nalezení optimálního AI modelu pro rozpoznávání prvků GUI. Nalezení optimální modelu se věnovala celá vlastní práce (4). Pro práci byla vybrána architektura YOLOv8, která v době vypracování studie byla nejnovější. Pro trénování modelu na rozpoznávání prvků GUI byl vybrán dataset VINS a bylo rozpoznáváno 12 jednotlivých tříd. Bylo prováděno trénování modelů se vstupními hyperparametry, které se v průběhu trénování měnily. Celkově proběhlo 9 trénování s různými hyperparametry (viz Tabulka 8). Těchto 9 modelů bylo mezi sebou porovnáno pomocí jednotlivých metrik (Precision, Recall, mAP) a byly vybrány 3 nejlepší. Tyto 3 modely byly mezi sebou opět následně porovnány a byl vybrán nejlepší z nich (model 8). Na závěr byl model 8 porovnán s modely s obdobných práci, kde dosáhl lepších výsledků. Jako nejlepší model byl v této studii vybrán model 8, nastavení vstupních hyperparametrů pro trénování tohoto modelu je zobrazeno v Tabulka 8. Výsledky tohoto modelu pro jednotlivé metriky jsou zobrazeny v Tabulka 13.

7 Seznam použitých zdrojů

1. HEAVI.AI. *What is a Graphical User Interface? Definition and FAQs | HEAVY.AI* [online]. [vid. 2023-08-16]. Dostupné z: https://www.heavy.ai/technical-glossary/graphical-user-interface
2. JUVILER, Jamie. What Is GUI? Graphical User Interfaces, Explained. *What Is GUI? Graphical User Interfaces, Explained* [online]. 25. srpen 2022 [vid. 2023-08-16]. Dostupné z: https://blog.hubspot.com/website/what-is-gui
3. CHEN, Jieshan, Mulong XIE, Zhenchang XING, Chunyang CHEN, Xiwei XU, Liming ZHU a Guoqiang LI. Object Detection for Graphical User Interface: Old Fashioned or Deep Learning or a Combination? In: *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* [online]. 2020, s. 1202–1214 [vid. 2023-08-15]. Dostupné z: doi:10.1145/3368089.3409691
4. MARGARET, Rouse. Graphical User Interface. *Techopedia* [online]. 28. květen 2021 [vid. 2023-08-16]. Dostupné z: https://www.techopedia.com/definition/5435/graphical-user-interface-gui
5. REIMER, Jeremy. A History of the GUI. *Ars Technica* [online]. 5. květen 2005 [vid. 2024-03-15]. Dostupné z: https://arstechnica.com/features/2005/05/gui/
6. IBM. *What is Computer Vision? | IBM* [online]. [vid. 2023-08-15]. Dostupné z: https://www.ibm.com/topics/computer-vision
7. IAN GOODFELLOW, YOSHUA BENGIO, a AARON COURVILLE. *Deep Learning*. 2016.-11.–18. vyd. Cambridge, Massachusetts: MIT Press Ltd, nedatováno. Adaptive Computation and Machine Learning series. ISBN 978-0-262-03561-3.
8. BANDYOPADHYAY, Hmrishav. *What Is Computer Vision? [Basic Tasks & Techniques]* [online]. 9. červen 2022 [vid. 2023-08-15]. Dostupné z: https://www.v7labs.com/blog/what-is-computer-vision, https://www.v7labs.com/blog/what-is-computer-vision
9. RISHABH SOFTWARE. What is Computer Vision and its Benefits. *Rishabh Software* [online]. 12. duben 2023 [vid. 2023-08-15]. Dostupné z: https://www.rishabhsoft.com/blog/computer-vision
10. ADDEPTO. Addepto. *Addepto* [online]. 11. duben 2022 [vid. 2023-08-15]. Dostupné z: https://addepto.com
11. AWATI, Rahul. What are Convolutional Neural Networks? | Definition from TechTarget. *Enterprise AI* [online]. [vid. 2023-08-30]. Dostupné z: https://www.techtarget.com/searchenterpriseai/definition/convolutional-neural-network
12. YAMASHITA, Rikiya, Mizuho NISHIO, Richard Kinh Gian DO a Kaori TOGASHI. Convolutional neural networks: an overview and application in radiology. *Insights into Imaging* [online]. 2018, **9**(4), 611–629. ISSN 1869-4101. Dostupné z: doi:10.1007/s13244-018-0639-9
13. DSHAHID380. Convolutional Neural Network. *Medium* [online]. 26. únor 2019 [vid. 2023-08-30]. Dostupné z: https://towardsdatascience.com/covolutional-neural-network-cb0883dd6529
14. MANDAL, Manav. Introduction to Convolutional Neural Networks (CNN). *Analytics Vidhya* [online]. 1. květen 2021 [vid. 2023-09-04]. Dostupné z: https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-cnn/
15. ALZUBAIDI, Laith, Jinglan ZHANG, Amjad J. HUMAIDI, Ayad AL-DUJAILI, Ye DUAN, Omran AL-SHAMMA, J. SANTAMARÍA, Mohammed A. FADHEL,

Muthana AL-AMIDIE a Laith FARHAN. Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. *Journal of Big Data* [online]. 2021, **8**(1), 53. ISSN 2196-1115. Dostupné z: doi:10.1186/s40537-021-00444-8
16. POKORNÝ, Pavel. Optické rozpoznávání znaků [online]. 2010 [vid. 2023-08-30]. Dostupné z: http://dspace.vutbr.cz/handle/11012/56031
17. SRIVASTAVA, Abhishek. Android Screenshot testing for UI. *Medium* [online]. 26. únor 2023 [vid. 2023-08-27]. Dostupné z: https://abhiappmobiledeveloper.medium.com/android-screenshot-testing-for-ui-2b9774dd7eed
18. KIRUTHIKA, Devaraj. Screenshot Testing: A Detailed Guide! *Testsigma Blog* [online]. 3. červenec 2023 [vid. 2023-08-27]. Dostupné z: https://testsigma.com/blog/screenshot-testing/
19. P, Srinivas. What is DOM - Document object model. *Testing Tools* [online]. 27. březen 2015 [vid. 2023-08-28]. Dostupné z: https://testingtools.co/general/what-is-dom-document-object-model
20. PNUELI, Ayelet, Omer BARKOL, Ruth BERGMAN, Michael POGREBISKY a Sagi SCHEIN. Graphical user interface component identification [online]. US20110099499A1. 28. duben 2011. [vid. 2023-08-28]. Dostupné z: https://patents.google.com/patent/US20110099499A1/en
21. CHRISTOPHER M. BISHOP. *Pattern Recognition and Machine Learning (Information Science and Statistics)* [online]. B.m.: Springer International Publishing, 2006. ISBN 0-387-31073-8. Dostupné z: https://www.amazon.com/Pattern-Recognition-Learning-Information-Statistics/dp/0387310738
22. THOMA, Martin. *A Survey of Semantic Segmentation* [online]. B.m.: arXiv. 11. květen 2016 [vid. 2023-08-29]. Dostupné z: http://arxiv.org/abs/1602.06541. arXiv:1602.06541 [cs]
23. SZELISKI, Richard. *Computer Vision: Algorithms and Applications* [online]. Cham: Springer International Publishing, 2022 [vid. 2023-08-29]. Texts in Computer Science. ISBN 978-3-030-34371-2. Dostupné z: doi:10.1007/978-3-030-34372-9
24. PRINCE, Simon J. D. Computer Vision: Models, Learning, and Inference. *Higher Education from Cambridge University Press* [online]. 18. červen 2012 [vid. 2023-08-30]. Dostupné z: doi:10.1017/CBO9780511996504
25. KUMARI, Kajal. A Basic Introduction to Object Detection. *Analytics Vidhya* [online]. 1. březen 2022 [vid. 2023-08-29]. Dostupné z: https://www.analyticsvidhya.com/blog/2022/03/a-basic-introduction-to-object-detection/
26. MATHWORKS. *What Is Object Detection?* [online]. [vid. 2023-08-29]. Dostupné z: https://www.mathworks.com/discovery/object-detection.html
27. SHAIP. Co je optické rozpoznávání znaků (OCR): Přehled a jeho aplikace | Shaip. *shaip.com* [online]. 10. květen 2022 [vid. 2023-08-30]. Dostupné z: https://cs.shaip.com/blog/ocr-overview-and-applications/
28. JAY. Úvod do optického rozpoznávání znaků (OCR). *HashDork* [online]. 5. květen 2022 [vid. 2023-08-30]. Dostupné z: https://hashdork.com/cs/optick%C3%A9-rozpozn%C3%A1v%C3%A1n%C3%AD-znak%C5%AF/
29. GEEKSFORGEEKS. R-CNN | Region Based CNNs. *GeeksforGeeks* [online]. 29. únor 2020 [vid. 2023-09-03]. Dostupné z: https://www.geeksforgeeks.org/r-cnn-region-based-cnns/
30. GIRSHICK, Ross, Jeff DONAHUE, Trevor DARRELL a Jitendra MALIK. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In: *2014*

IEEE Conference on Computer Vision and Pattern Recognition: *2014 IEEE Conference on Computer Vision and Pattern Recognition* [online]. 2014, s. 580–587. ISSN 1063-6919. Dostupné z: doi:10.1109/CVPR.2014.81
31. LOHIA, Aditya, Kalyani Dhananjay KADAM, Rahul Raghvendra JOSHI a Dr Anupkumar M BONGALE. Bibliometric Analysis of One-stage and Two-stage Object Detection. 2018.
32. MATTHIJS HOLLEMANS. *One-stage object detection* [online]. 9. červen 2018 [vid. 2024-03-03]. Dostupné z: https://machinethink.net/blog/object-detection/
33. ARCGIS. How RetinaNet works? *ArcGIS API for Python* [online]. [vid. 2023-09-05]. Dostupné z: https://developers.arcgis.com/python/guide/how-retinanet-works/
34. TAN, Lu, Tianran HUANGFU, Liyao WU a Wenying CHEN. Comparison of RetinaNet, SSD, and YOLO v3 for real-time pill identification. *BMC Medical Informatics and Decision Making* [online]. 2021, **21**(1), 324. ISSN 1472-6947. Dostupné z: doi:10.1186/s12911-021-01691-8
35. KUNDU, Rohit. *YOLO Algorithm for Object Detection Explained [+Examples]* [online]. 17. leden 2023 [vid. 2023-09-06]. Dostupné z: https://www.v7labs.com/blog/yolo-object-detection, https://www.v7labs.com/blog/yolo-object-detection
36. KARIMI, Grace. Introduction to YOLO Algorithm for Object Detection. *Engineering Education (EngEd) Program | Section* [online]. zima 2021 [vid. 2023-09-06]. Dostupné z: https://www.section.io/engineering-education/introduction-to-yolo-algorithm-for-object-detection/
37. KEITA, Zoumana. *YOLO Object Detection Explained: A Beginner's Guide* [online]. září 2022 [vid. 2023-09-06]. Dostupné z: https://www.datacamp.com/blog/yolo-object-detection-explained
38. SPODAREC, Dmitrij, Soham SHARMA a Maryna MARCHUK. A Guide to the YOLO Family of Computer Vision Models. *Data Phoenix* [online]. 12. únor 2023 [vid. 2023-09-06]. Dostupné z: https://dataphoenix.info/a-guide-to-the-yolo-family-of-computer-vision-models/
39. KEYLABS. Under the Hood: YOLOv8 Architecture Explained. *Keylabs: latest news and updates* [online]. 20. prosinec 2023 [vid. 2024-03-12]. Dostupné z: https://keylabs.ai/blog/under-the-hood-yolov8-architecture-explained/
40. MUKILAN KRISHNAKUMAR. YOLOv8 Introducion. *W&B* [online]. 7 2023 [vid. 2023-12-15]. Dostupné z: https://wandb.ai/mukilan/wildlife-yolov8/reports/A-Gentle-Introduction-to-YOLOv8--Vmlldzo0MDU5NDA2
41. LIU, Wei, Dragomir ANGUELOV, Dumitru ERHAN, Christian SZEGEDY, Scott REED, Cheng-Yang FU a Alexander C. BERG. SSD: Single Shot MultiBox Detector. In: [online]. 2016 [vid. 2023-09-07], s. 21–37. Dostupné z: doi:10.1007/978-3-319-46448-0_2
42. HUI, Jonathan. SSD object detection: Single Shot MultiBox Detector for real-time processing. *Medium* [online]. 15. prosinec 2020 [vid. 2023-09-07]. Dostupné z: https://jonathan-hui.medium.com/ssd-object-detection-single-shot-multibox-detector-for-real-time-processing-9bd8deac0e06
43. LEARNING, Great. Everything you need to know about VGG16. *Medium* [online]. 23. září 2021 [vid. 2023-09-07]. Dostupné z: https://medium.com/@mygreatlearning/everything-you-need-to-know-about-vgg16-7315defb5918

44. PARK, Sieun. A guide to Two-stage Object Detection: R-CNN, FPN, Mask R-CNN and more. *CodeX* [online]. 29. červenec 2021 [vid. 2024-03-03]. Dostupné z: https://medium.com/codex/a-guide-to-two-stage-object-detection-r-cnn-fpn-mask-r-cnn-and-more-54c2e168438c
45. MATHWORKS. *Začínáme s R-CNN, Fast R-CNN a Faster R-CNN - MATLAB & Simulink* [online]. [vid. 2023-09-03]. Dostupné z: https://www.mathworks.com/help/vision/ug/getting-started-with-r-cnn-fast-r-cnn-and-faster-r-cnn.html
46. GANDHI, Rohith. R-CNN, Fast R-CNN, Faster R-CNN, YOLO — Object Detection Algorithms. *Medium* [online]. 9. červenec 2018 [vid. 2023-09-03]. Dostupné z: https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e
47. DIVE INTO DEEP LEARNING. *14.8. Region-based CNNs (R-CNNs) — Dive into Deep Learning 1.0.3 documentation* [online]. [vid. 2023-09-04]. Dostupné z: https://d2l.ai/chapter_computer-vision/rcnn.html
48. TSANG, Sik-Ho. Reading: Cascade R-CNN — Delving into High Quality Object Detection (Object Detection). *Medium* [online]. 13. říjen 2021 [vid. 2024-03-05]. Dostupné z: https://sh-tsang.medium.com/reading-cascade-r-cnn-delving-into-high-quality-object-detection-object-detection-8c7901cc7864
49. META. Detectron. *ai.meta.com* [online]. [vid. 2023-12-13]. Dostupné z: https://ai.meta.com/tools/detectron
50. *Detektron* [online]. Python. B.m.: Meta Research. 13. prosinec 2023 [vid. 2023-12-13]. Dostupné z: https://github.com/facebookresearch/Detectron
51. SKORKOVSKÝ, Jaromír. Metriky v informatice. nedatováno.
52. EVIDENTLY AI. *Accuracy vs. precision vs. recall in machine learning: what's the difference?* [online]. [vid. 2023-12-15]. Dostupné z: https://www.evidentlyai.com/classification-metrics/accuracy-precision-recall
53. ARCGIS PRO. *Jak funguje výpočetní přesnost pro detekci objektů – ArcGIS Pro | Dokumentace* [online]. [vid. 2024-03-12]. Dostupné z: https://pro.arcgis.com/en/pro-app/latest/tool-reference/image-analyst/how-compute-accuracy-for-object-detection-works.htm
54. VEDOVELI, Henrique. Metrics Matter: A Deep Dive into Object Detection Evaluation. *Medium* [online]. 15. září 2023 [vid. 2024-03-12]. Dostupné z: https://medium.com/@henriquevedoveli/metrics-matter-a-deep-dive-into-object-detection-evaluation-ef01385ec62
55. ROHIT KUNDU. *F1 Score in Machine Learning: Intro & Calculation* [online]. 12 2023 [vid. 2023-12-15]. Dostupné z: https://www.v7labs.com/blog/f1-score-guide, https://www.v7labs.com/blog/f1-score-guide
56. DEVAL SHAH. *Mean Average Precision (mAP) Explained: Everything You Need to Know* [online]. 7. březen 2022 [vid. 2024-01-15]. Dostupné z: https://www.v7labs.com/blog/mean-average-precision, https://www.v7labs.com/blog/mean-average-precision
57. ALTINBAS, Mehmet Dogan a Tacha SERIF. GUI Element Detection from Mobile UI Images Using YOLOv5. In: Irfan AWAN, Muhammad YOUNAS a Aneta PONISZEWSKA-MARAŃDA, ed. *Mobile Web and Intelligent Information Systems* [online]. Cham: Springer International Publishing, 2022, s. 32–45. Lecture Notes in Computer Science. ISBN 978-3-031-14391-5. Dostupné z: doi:10.1007/978-3-031-14391-5_3

58. SBUNIAN. *sbunian/VINS* [online]. 14. březen 2024 [vid. 2024-03-15]. Dostupné z: https://github.com/sbunian/VINS
59. ULTRALYTICS. *Configuration* [online]. [vid. 2024-03-15]. Dostupné z: https://docs.ultralytics.com/usage/cfg

8 Seznam obrázků, tabulek, grafů a zkratek

8.1 Seznam obrázků

8.2 Seznam tabulek

8.3 Seznam grafů

8.4 Seznam použitých zkratek

GUI – Graphical User Interface
CNN – Convolution Neural Network
DOM – Document Object Model
OCR – Optical Character Recognition
AI – Artificial Intelligence
SSD – Single Shot multibox Detector
P – Precision
R – Recall
mAP – mean Average Precision

Přílohy

Odkazovaný seznam příloh