Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačních technologií

Bakalářská práce

Výběr vhodných metod z oblasti Computer Vision pro klasifikaci a identifikaci prvků uživatelského rozhraní

Jan Bosák

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Jan Bosák

Informatika

Název práce

Výběr vhodných metod z oblasti Computer Vision pro klasifikaci a identifikaci prvků uživatelského rozhraní

Název anglicky

Selection of appropriate methods of Computer Vision for classification and identification of user interface elements

---

Cíle práce

Hlavním cílem práce je vybrat a porovnat vhodné metody počítačového vidění pro rozpoznávání prvků a jejich detekci v uživatelském prostředí.

Dílčí cíle jsou:
- charakteristika jednotlivých metod a jejich vhodné využití v praxi,
- výběr a implementace vhodných metod a modelů pro porovnání,
- nalezení a případně vytvoření a anotace experimentální datové sady.

Metodika

Metodika řešené problematiky bakalářské práce je založena na studiu a analýze odborných informačních zdrojů. Nejprve bude nalezena, nebo vytvořena datová sada formou screenshotů různých uživatelských prostředí. Následně bude provedena jejich anotace. Vybrané metody budou dále natrénovány s použitím anotovaných dat. Porovnání zvolených modelů bude provedeno zhodnocením vhodných metrik nad validační množinou dat. Na základě syntézy teoretických poznatků a výsledků praktické části budou formulovány závěry práce.

Doporučený rozsah práce

40 – 50 stran

Klíčová slova

Computer vision, Machine learning, UI, neuronové sítě

Doporučené zdroje informací

CHOLLET, François. Deep learning v jazyku Python (přeloženo). Praha: Grada Publishing, 2019. ISBN 978-80-271-2751-1

JIRKOVSKÝ, Jaroslav. Metody Deep Learning k segmentaci obrazu. Automa. Děčín: Automa – ČAT, 2017. ISSN 1210-9592

Klette, R. Concise Computer Vision An Introduction into Theory and Algorithms. London : Springer London, 2014. ISBN 1-4471-6320-6

REDMON, Joseph. S. D. R. G. A. F. You Only Look Once: Unified, Real-Time Object Detection [online]. Dostupné z: https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Redmon_You_Only_Look_CVPR_2016_paper.pdf.

SOLEM, Jan Erik. Programming Computer Vision with Python. Sebastopol, CA, USA: OŔeilly Media Inc., 2021. ISBN 978-1-449-31654-9

ŠONKA, Milan, HLAVÁČ Václav a Roger BOYLE. Image processing, analysis, and machine vision. Fourth edition. Stamford, CT, USA: Cengage Learning, 2015. ISBN 978-113-3593-607

Předběžný termín obhajoby

2022/23 LS – PEF

Vedoucí práce

Ing. Jan Masner, Ph.D.

Garantující pracoviště

Katedra informačních technologií

Elektronicky schváleno dne 14. 7. 2022

doc. Ing. Jiří Vaněk, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 27. 10. 2022

doc. Ing. Tomáš Šubrt, Ph.D.

Děkan

V Praze dne 14. 03. 2024

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci "Výběr vhodných metod z oblasti Computer Vision pro klasifikaci a identifikaci prvků 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 15.3. _____

Poděkování

Rád(a) bych touto cestou poděkoval(a) panu Ing. Janu Masnerovi, Ph. D za cenné rady a odborné vedení při zpracování této práce.

Výběr vhodných metod z oblasti Computer Vision pro klasifikaci a identifikaci prvků uživatelského rozhraní

Abstrakt

Tato bakalářská práce je členěna na část teoretickou a praktickou. Hlavním tématem je Computer Vision, podrobněji představený v teoretické části spolu s dalšími pojmy a metodami z tohoto oboru. Teoretická část charakterizuje jednotlivé metody, popisuje jejich vývoj a přístupy k detekci. Zároveň se zaměřuje na současný stav technologií v oblasti Computer Vision. Praktická část aplikuje teoretické poznatky a metody na konkrétní modely. Pomocí praktických sestavené datové sady, experimentů, trénování modelů a následné aplikaci se snažilo demonstrovat funkčnost a využitelnost metod DETR, Faster R-CNN, RetinaNet a YOLOv8 v reálných situacích s využitím reálného uživatelského rozhraní s obsahem různých prvků. Cílem práce je poskytnout ucelený přehled o metodách a technologiích Computer Vision a jejich aplikaci a porovnání, s důrazem na aktuální vývoj a možnosti budoucího rozvoje.

Klíčová slova: computer vision, objektová detekce, neuronové sítě, deep learning, uživatelské rozhraní, klasifikace, segmentace, identifikace,

Selection of appropriate methods of Computer Vision for classification and identification of user interface elements

Abstract

This bachelor thesis is divided into theoretical and practical parts. The main topic is Computer Vision, elaborated in detail in the theoretical section along with other concepts and methods from this field. The theoretical part characterizes individual methods, describes their development, and approaches to detection. It also focuses on the current state of technology in the field of Computer Vision. The practical part applies theoretical knowledge and methods to specific models. Through practical assembly of datasets, experiments, model training, and subsequent application, it aims to demonstrate the functionality and usability of the DETR, Faster R-CNN, RetinaNet and YOLOv8 methods in real-life situations using a real user interface with various elements. The aim of the thesis is to provide a comprehensive overview of Computer Vision methods and technologies and their application and comparison, with an emphasis on current development and future possibilities.

Keywords: Computer vision, object detection, neural networks, deep learning, user interface, classification, segmentation, identification,

Obsah

1 Úvod ... 10

2 Cíl práce a metodika ... 11

2.1 Cíl práce ... 11

2.2 Metodika ... 11

3 Teoretická východiska ... 12

3.1 Umělá inteligence ... 12

3.1.1 Historie Umělé inteligence ... 13

3.1.2 Turingův test ... 13

3.2 Strojové učení ... 14

3.2.1 Dělení strojového učení ... 15

3.2.2 Modely vnímání ve strojovém učení ... 15

3.3 Neuronové sítě ... 16

3.3.1 Hluboké učení ... 18

3.3.2 Vícevrstvé neuronové sítě ... 18

3.3.3 Konvoluční neuronové sítě ... 23

3.4 Computer Vision ... 25

3.4.1 Historie Computer visionu ... 26

3.4.2 Hluboké učení v Computer vision ... 27

3.5 Architektury CNN ... 27

3.5.1 R-CNN (Region Based Convolitonal Neural Network) ... 27

3.5.2 Fast R-CNN (Fast Region Based Convolitonal Neural Network) ... 29

3.5.3 Faster R-CNN (Faster Region Based Convolitonal Neural Network) ... 29

3.5.4 SSD ... 30

3.5.5 YOLO ... 31

3.5.6 DETR ... 35

3.6 Analýza současných přístupů k detekci UI objektů ... 36

3.6.1 UIED (UI Element Detection) ... 36

3.6.2 Imgcook ... 37

3.6.3 Srovnání tradičních a hlubokých metod detekce ... 38

4 Praktická část práce ... 39

4.1 Příprava vhodného datasetu ... 39

4.1.1 Sestavení testového setu snímků pro nezávislý test modelů ... 40

4.2 Prostředí Google Colab ... 42

4.3 Stanovení metrik pro analýzu výsledků ... 43

4.4 Sestavení modelů ... 45

4.4.1 Model DETR ... 46

4.4.2 Model Faster R-CNN ... 48
4.4.3 Model RetinaNet ... 50
4.4.4 Model YOLOv8 ... 52

5 Zhodnocení výsledků ... 55
5.1 Vyhodnocení výsledků na nezávislém testu ... 55
5.2 Prezentace vizuálních výsledků ... 56
5.3 Srovnání celkových výsledků a jejich analýza ... 57
5.4 Doporučení ... 60

Závěr ... 61

Seznam použitých zdrojů ... 62

Seznam obrázků, tabulek, grafů a zkratek ... 67
5.5 Seznam obrázků ... 67
5.6 Seznam scriptů ... 68
5.7 Seznam tabulek ... 68
5.8 Seznam grafů ... 68
5.9 Seznam použitých zkratek ... 69

Přílohy ... 70
Příloha A ... 70

1 Úvod

Počítačové vidění neboli Computer vision využívá obrazu a videa pro automatizované sledování objektů, detekci a klasifikaci pro jejich pochopení. Jde o oblast spojenou s umělou inteligencí, fyzikou a neurobiologií. Umělá inteligence je brána jako schopnost, která umožňuje počítačům myslet, kdežto Computer vision se soustředí, aby počítač viděl, pozoroval a zároveň správně chápal, co je mu předkládáno. Existuje mnoho způsobů, jak lze toto řešit. Dnes se nejčastěji používají techniky z Deep learningu. Deep learning bere z obrazu potřebné informace, které poté „přetvoří“ do 2D, 3D modelů nebo do různých formátů dat. Snaží se napodobit lidské vidění, často pomocí neuronových sítí, kdy i geometrické řešení může být řešení problému. Computer vision je relativně málo prozkoumané odvětví, protože jeho kořeny lze nalézt teprve na konci 70.let.

Computer vision v podstatě funguje stejně jako lidský zrak s výjimkou některých jasně patrných omezení v jeho schopnostech. Lidský zrak má výhodu v kontextu celoživotního tréninku, jak od sebe rozlišit předměty, jak a jestli se pohybují nebo jejich vzdálenost a polohu. Computer vision je na tyto úkony trénovaný v mnohem kratším časovém období pomocí kamer, různých algoritmů a dat. Trénovaný systém dokáže analyzovat tisíce produktů nebo procesů za minutu a zaznamenává nepostřehnutelné vady nebo problémy, čímž by mohl v budoucnu překonat lidský zrak. Je využíván v několika odvětvích, od energetiky a veřejných služeb až po výrobu a průmysl. Jeho využití na trhu neustále roste, což lze vidět na událostech posledních let.

2 Cíl práce a metodika

2.1 Cíl práce

Hlavním cílem práce je vybrat a porovnat vhodné metody počítačového vidění pro rozpoznávání prvků a jejich detekci v uživatelském prostředí.

Dílčí cíle jsou:
- charakteristika jednotlivých metod a jejich vhodné využití v praxi,
- výběr a implementace vhodných metod a modelů pro porovnání,
- nalezení a případně vytvoření a anotace experimentální datové sady.

2.2 Metodika

Metodika řešené problematiky bakalářské práce bude založena na studiu a analýze odborných informačních zdrojů. Nejprve bude nalezena, nebo vytvořena datová sada formou snímků obrazovky různých uživatelských prostředí. Následně bude provedena jejich anotace. Vybrané metody budou dále natrénovány s použitím anotovaných dat. Porovnání zvolených modelů bude provedeno zhodnocením vhodných metrik nad validační množinou dat. Na základě syntézy teoretických poznatků a výsledků praktické části budou formulovány závěry práce.

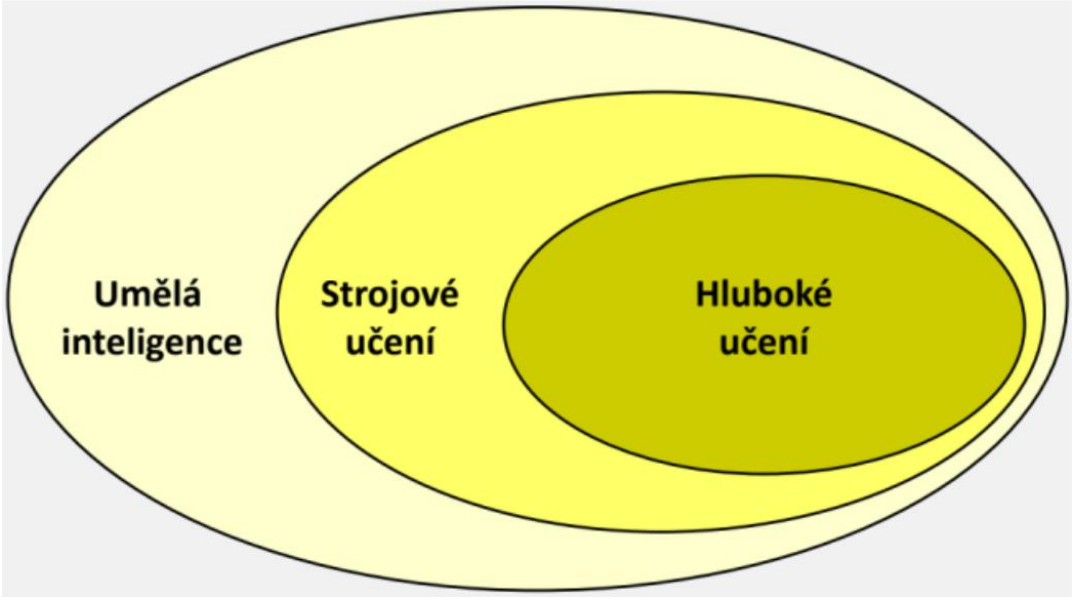
3 Teoretická východiska

3.1 Umělá inteligence

Stručná definice cíle umělé inteligence podle (Chollet, 2019) zní: *„pokusit se automatizovat intelektuální úlohy, které normálně provádějí lidé.“*

Umělou inteligenci si lze představit jako strukturu, která je schopná automaticky plnit úkoly, kde není potřeba člověka s inteligencí pro jejich splnění. Umělá inteligence se dnes často spojuje s pojmy jako strojové nebo hluboké učení, přeloženo z angličtiny – machine learning a deep learning a mimo jiné i computer vision (počítačové vidění). Dříve umělá inteligence měla všechny hlavní a dílčí úkoly splňovat na základě vepsaných pravidel, které měla napsané pomocí kódu od jejího vývojáře a neobsahovalo nějaké učení. Tento přístup se dříve označovalo jako „symbolic AI“ (Sikka, 2021).

Obrázek 1 – Schéma Umělé inteligence, strojového učení a hlubokého učení



Zdroj: Chollet (2019)

Symbolic AI lze považovat za inteligentní, ale jen tehdy kdy vykonává dobře definované logické problémy (např. stolní hry, matematické úlohy), ale není schopna plnit úkoly spojené s identifikací a klasifikací obrazu, řízení aut, rozpoznávání řeči, rozpoznávání prvků uživatele v obrazu na sociálních sítích, překlad jazyků a vybírání cílených reklam, videí nebo hudby v aplikacích, co uživatel zrovna hledá nebo v poslední době hledal (Chollet, 2019).

3.1.1 Historie Umělé inteligence

V posledních letech se Umělá inteligence, v anglickém jazyce Artificial intelligence neboli AI, stalo velkým a velmi často probíraným tématem. Budoucí vývoj umělé inteligence slibuje zjednodušení či plné nahrazení člověka u rutinních úloh, ale také vznik různých virtuálních asistentů a chatbotů. Dnes již také mnoho automobilových firem využívá umělou inteligenci i pro vývoj a výrobu autonomních systémů (Sikka, 2021).

Počátkem umělé inteligence je padesátých letech 20.století. V tomto období již probíhal výzkum mnohými laboratořemi po celém světě. Za první produkt s prvky umělé inteligence se považuje počítač Arthura Samuela z roku 1952, který dokázal simulovat deskovou hru Dáma. Poté v roce 1954 umělá inteligence IBM zvládla splnit prvotní úspěchy při překladu jazyků. Roku 1959 přichází Arthur Samuel s pojmem *„strojové učení“,* která definuje AI s dovedností pochopení dat a vytváření samostatně učících se algoritmů. Významnými roky jsou také 1966, kdy byl vytvořen první virtuální asistent ELIZA a 1970 první humanoidní robot „Wabot-1“ z Japonska. Robot dokázal samostatně komunikovat s lidmi a manipulovat s předměty (Mikelsten, Teigens, Skalfist, 2020).

Od počátku padesátých let byly dominantní symbolické umělé inteligence, které jsou již popsané výše. Teprve v osmdesátých letech byly symbolické umělé inteligence nahrazeny pojmem strojové učení. Postupně vývoj pomalu utichal z důvodu nedostatku technologií a i kapitálu (Chollet, 2018).

3.1.2 Turingův test

Spolu se vznikem umělých inteligencí také vzniknul hypotetický test tzv. Turingův test od Alana Turinga (britský matematik, logik, kryptoanalytik a zakladatel moderní informatiky), který pojednával o tom, zda je zkoumaný stroj, program nebo umělá inteligence skutečně inteligentní (Kaďousková, 2022).

Turingův test si lze představit jako konverzaci mezi dvěma místnostmi, kdy v jedné z nich se nachází počítač s člověkem oddělených od sebe a v druhé místnosti testující člověk, jehož úkolem je rozeznat, zda probíhající konverzaci vede s člověkem nebo se strojem. Testující pomocí otázek zjišťuje, zda probíhající konverzaci vede s osobou nebo počítačem. Otázky jsou pokládány náhodně člověku i počítači v druhé místnosti. Pokud osoba, která provádí test, nerozpozná rozdíl mezi výslednými odpověďmi člověka a počítače, je počítač,

software nebo stroj (záleží, co testující osoba testuje), označena za inteligentní (Damassion, 2020).

Existuje tzv. „argument čínského pokoje“, který pojednává o tom, že test nepočítá, s tím, že by počítač mohl porozumět obsahu a smyslu otázky, ale neodpoví. Test pouze posuzuje správnost odpovědí. Proto existuje rozdílné označení „slabé“ a „silné“ umělé inteligence. Turingův test by v rámci tohoto rozdělení testoval jen slabé AI.

Problémy napodobení lidské inteligence:

- Obecná inteligence;
- Sociální inteligence;
- Emoční inteligence;
- Řešení problémů;
- Plánování;
- Reprezentace znalostí (ontologie);
- Učení (machine learning);
- Pohyb a manipulace;
- Zpracování přirozeného jazyka (Waltlová, 2019).

3.2 Strojové učení

Strojové učení je koncept, kde se stroje nebo systémy učí pochopit ze zkušeností, jak se lépe rozhodovat. Pracují s algoritmy využívající DM (Data mining), které pomocí rozpoznávání dokážou předpovídat obsah dat z předchozích dat. „Učení“ v strojovém učení znamená schopnost zachytávání chyb při předpovídání. S menší chybovostí poté můžeme dosahovat přesnější předpovídaní výskytu informací v datech (Sikka, 2021).

Dobrým příkladem tohoto učení by mohlo být označování fotografií na základě toho, co na nich je. Například bude na několika fotografiích fotky z přírody a budeme chtít jen ty, které obsahují nějakou vodní plochu. Ukážeme systému tedy pár fotografií, které byly označeny jako ty s vodní plochou. Systém se učí statistická pravidla pro rozdělování obrázků podle požadavků. Statistické metody strojového učení se liší od těch matematických, a to tím, že má schopnost pracovat s rozsáhlými datovými množinami jako např. několik milionů obrázků kde každý jeden obrázek obsahuje tisíce pixelů (Chollet, 2019).

3.2.1 Dělení strojového učení

Strojové učení lze rozdělit následovně:

Podle typu učení:

- Statistické učení – vyhodnocení na základě nezpracovaných dat;
- Neuronové sítě – složení z umělých neuronů odpovídající těm biologickým;
- Posílené učení – použití podobných technik jako u zvířat (stav, akce, odměna a politika). Použití metod a technik jako jsou například Bellmanova rovnice, Markovův rozhodovací proces, Q.learning.

Podle metod učení ze souborů dat:

- Učení pod dohledem (Supervised learning) – datasety obsahují data s pomocnými prvky, který systém musí chápat, aby danému vstupu dat porozuměl, např. program na kávovaru, databáze filmů;
- Učení bez dohledu (Unsupervised learning) – datasety neobsahují data s pomocnými prvky. Je nutné z některých dat podchytit informace podle kterých jsou prvky systémem pochopeny, např. seznam transakcí zákazníka z účtu banky;
- Učení pod částečným dohledem (Semi-supervised learning) – datasety kombinují učení obou předešlých metod (Alpaydin, 2014).

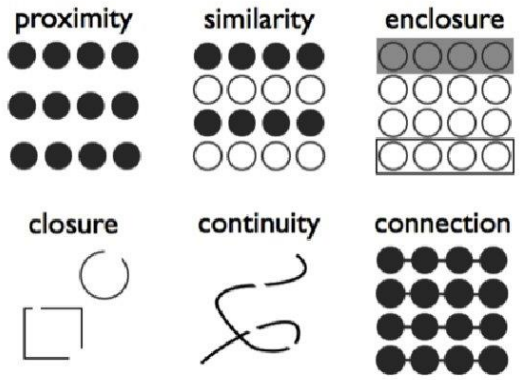
3.2.2 Modely vnímání ve strojovém učení

Algoritmy strojového učení jsou schopny z dat rozpoznat vzory pomocí několika metod, které jsou postupně popsány níže. Vychází z Gestaltových zákonů vizuálního vnímání (dobrého tvaru, neuzavřenost, kontinuita, blízkost, podobnost a spojení). Tyto zákony nám popisují, jak jsou vzory vnímány a nejsou skutečně podobné algoritmům strojového učení:

1. Zákon uzavřenosti (closure) – vzory jsou vnímány systémem jako úplné nebo pravidelné, i když se nám na první pohled může zdát, že jsou nedokončené. Člověk má tendenci je doplňovat;
2. Zákon neuzavřenosti (enclosure) – vnímaní hranice, která spojuje dohromady neslučitelné vzory;
3. Zákon kontinuity (continuity) – vzory jsou vnímány jako celek, pokud se zdá, že na sebe navazují;
4. Zákon blízkosti (proximity) – vnímání série nebo skupinu vzorů, které se chápou jako podobné;

5. Zákon podobnosti (similarity) – vnímání skupinu vzorů podle tvaru, velikosti a barvy;
6. Zákon spojení (connection) – vnímání vzorů patřící k sobě, pokud jsou propojeny.

Obrázek 2 – Gestaltovy zákony – Principy percepčního pole



Zdroj: https://slidetodoc.com/pednka-pro-ppravn-kurz-ke-studiu-psychologie-djiny/

Algoritmy tedy poté co se naučí, jak data zpracovávat, ať už pomocí učení s dohledem nebo bez a správně vnímat jejich strukturu, měly by být schopny co nejmenší chybovosti, což je tedy hlavním cílem těchto algoritmů. Pro jejich učení je zapotřebí určit jaký typ dat bude na vstupu, to znamená, jestli se budou používat zvukové nahrávky např. překládání řeči člověka anebo obrázkové soubory např. pro detekci, klasifikaci obsahu obrázků. Určit příklady očekávaných vstupů hodnot v rozpoznávání obrázků, zda je na obrázku strom, dům nebo voda. A dále pak měření, zda algoritmus svoji úlohu plní správně (Sikka, 2021).

Modely strojového učení se snaží přeměnit vstupy na smysluplné výstupy pomocí toho, co se naučily. Strojové, a i hloubkové učení mají problém se naučit užitečně používat příklady z dat tak aby výstup byl co nejpřesnější očekávání. Proto je důležitá správná prezentace naučených schopností (Chollet, 2019).

3.3 Neuronové sítě

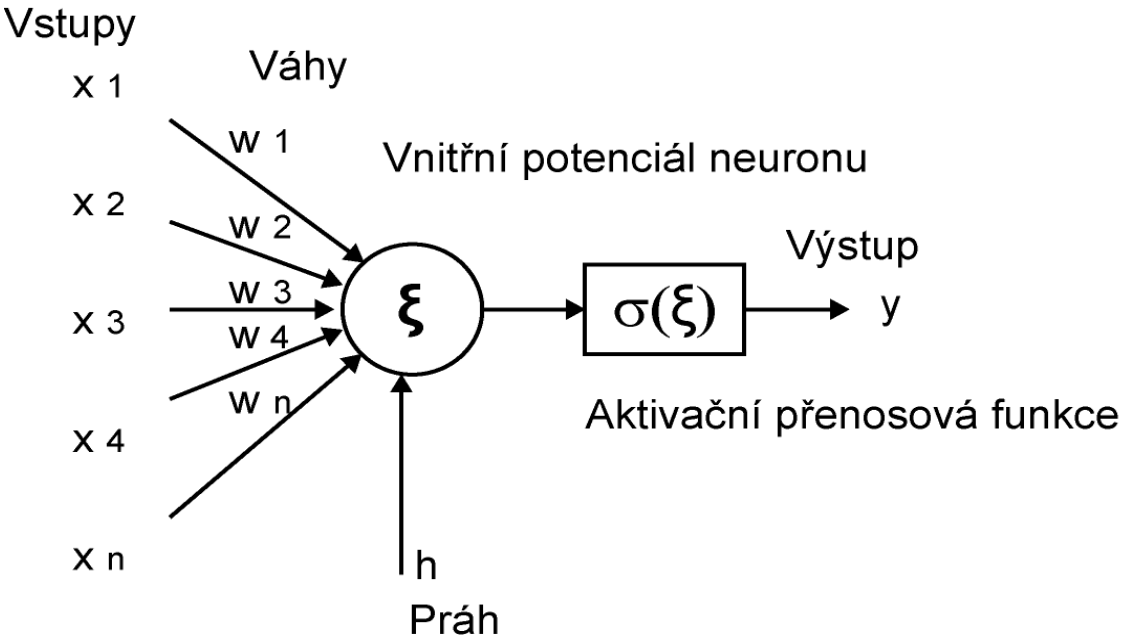
Jsou to sítě s datovou strukturou s cílem napodobit ty biologické a jejich základní funkce, zejména lidský mozek. Rozdílem v tvorbě běžných uživatelských programů, kde je zapotřebí vytvářet algoritmy, které transformují vstupní data na výstupní, je ve schopnosti

učení se. Neuronové sítě se dost často spojují s oborem CV, protože využívá jejich metody. Schopnost „učení se“ transformuje vstupní data na výstupní založené na struktuře vzorků popisující řešený aktuální problém. Tréninkové vzory umělé neuronové sítě a toto učení tak dokonale nahrazuje algoritmizaci dané úlohy.

Umělý neuron je matematický model neuronu a je základním kamenem neuronových sítí. Umělé neurony se třídí na základě, jakou matematickou funkci používají a jejich složitosti. Nejpoužívanější neuron je tzv. formální neuron. Neurony se skládají ze dvou částí:

- Obvodová funkce – určuje způsob, jak budou kombinovány parametry vstupů uvnitř neuronu.
- Aktivační funkce – určuje přenosovou funkci, jak budou vstupní parametry transformovány na výstup neuronu (Vondrák, 2009).

Obrázek 3 – Model neuronu



Zdroj:https://portal.matematickabiologie.cz/index.php?pg=analyza-a-hodnoceni-biologickych-dat--umela-inteligence--neuronove-site-jednotlivy-neuron--jednotlivy-neuron--matematicky-model-a-aktivni-dynamika-neuronu

Formální neuron má n obecně reálných vstupů x, modelující dendrity. Každý vstup je ohodnocen odpovídající synaptickou váhou w, která určuje jeho propustnost. Změnou synaptických vah během učení je formální neuron schopen adaptovat nově nabyté zkušenosti. h značí práh n-tého neuronu (bias). Neuron je aktivní pouze tehdy, pokud je vážená sumu vstupů větší než práh. – Vondrák, 2009.

3.3.1 Hluboké učení

Computer vision lze řešit různými metodami mezi ty spolehlivé a přesné patří hluboké učení – z anglického překladu deep learning. Hluboké učení je rozsáhlá větev neuronových sítí, což je technika učení ve strojovém myšlení, která obsahuje víc vzájemně propojených neuronových sítí. Inspirace jednoznačně plyne z neurobiologických sítí, přesto nejde o stejnou síť. Neurony v těchto sítích nesoucí parametry jsou neurony provádějící výpočty. Sítě takových to neuronů vzájemně propojených a přidáním další vrstvy je znakem hlubokého učení. Hluboké učení znamená čím více těchto vrstev bude přidáno, tím je hlubší. Neuronové sítě se dvěma nebo třemi vrstvami neuronů se nazývají „shallow networks“ tzv. mělké sítě. S těmito sítěmi se většinou pracuje ve strojovém učení. Slovo „hluboké“ tedy v hlubokém učení neznamená hlubší význam tohoto učení, ale hlubší dosah neuronových sítí (Sikka, 2021).

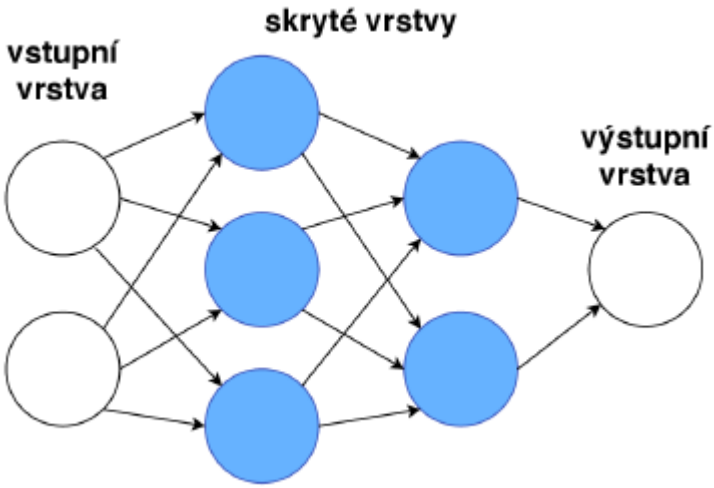
Strojové učení využívá provádí mapování vstupů pomocí pozorování příkladů, které v sobě nesou vstupní data. Hluboké neuronové sítě naopak toto mapování provádí hloubkovou transformací dat a tyto transformace jsou naučeny na pochopení vyřešených příkladů. To, co vrstva dělá se vstupnými daty můžeme chápat jako ukládání spousty čísel do hmotnosti dané vrstvy. Parametry pro transformování vrstvou pak záleží na hmotnosti té vrstvy, kterou transformujeme (Chollet, 2019).

3.3.2 Vícevrstvé neuronové sítě

Neuronové sítě mohou být rozličné počtem neuronů, typem učení a svojí stavbou modelu. Neurony v těchto sítích tvoří společnou propojenou vrstvu. Podle tohoto propojení lze dělit na sítě s dopředným šířením informace – „feedforward“ a rekurentní sítě – „recurrent networks“. Vícevrství neuronové sítě vždy tvoří minimálně tři vrstvy a ty se nazývají:

- Vstupní vrstva – vrstva do které vstupují data a nejsou nijak měněny;
- Skrytá vrstva – data přijímá z vrstvy vstupní, transformuje je a pošle dál do další vrstvy v pořadí;
- Výstupní vrstva – obsahuje výstupní data po celém průchodu neuronovou sítí (Mikulský, 2021).

Obrázek 4 – Jednoduchá architektura vícevrstvé neuronové sítě

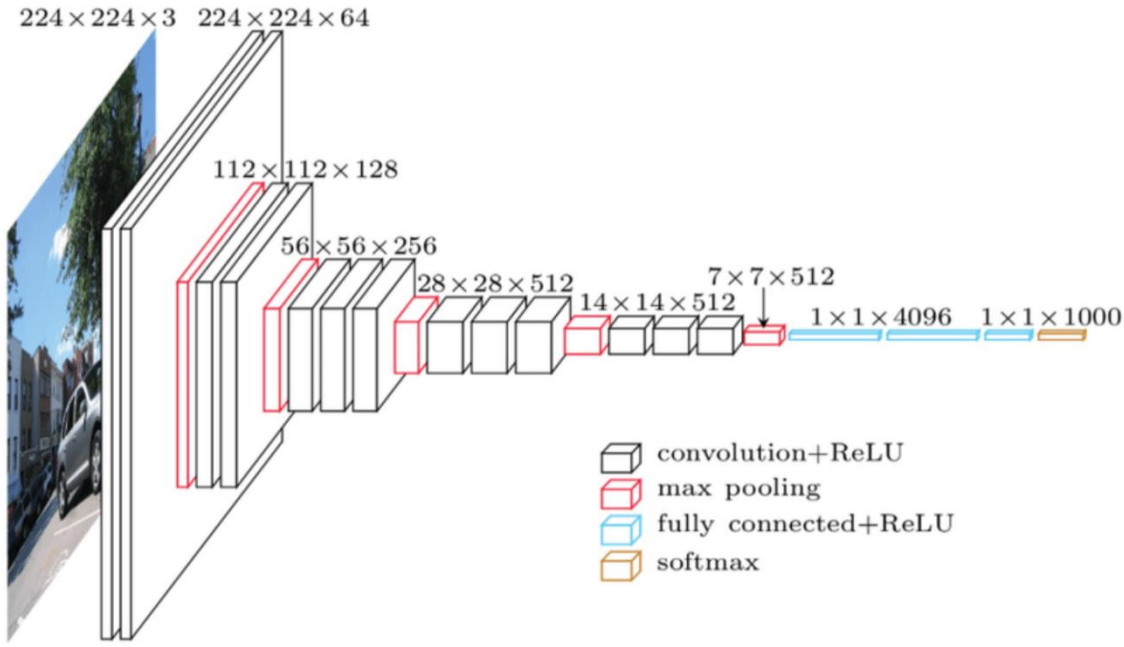


Zdroj: Mikulovský (2021)

VGG-16 a VGG-19

VGG-16 a VGG-19 jsou jedny z typů VGG sítě. VGG-16 obsahuje 16 vrstev hloubky a VGG-19 jich má 19. Liší se hlavně v rozdílnosti výsledků testů top-5 v ImageNet, kdy VGG-19 vychází větší procentuální přesnost než VGG-16. Není překvapením, že od toho, kolik obsahují vrstev tak se také tak jmenují. Tuto síť navrhnuli vědci z Oxfordské univerzity Andrew Zisserman a Karen Simonyan. Jedná se o jednu z nejmodernějších neuronových sítí. Je trénována tak, aby klasifikovala 100 různých tříd s více než 14 miliony obrázků s přibližně 144 miliony parametry. Tento soubor tříd, obrázků a parametrů se nazývá ImageNet. Procentuální přesnost sítí VGG-16 a VGG-19 se pohybují někde kolem 90 v testech top-5 ImageNet. Existuje několik výsledků těchto sítí, ale VGG-19 je považována za přesnější (Sikka, 2021).

Obrázek 5 – Architektura VGGnet



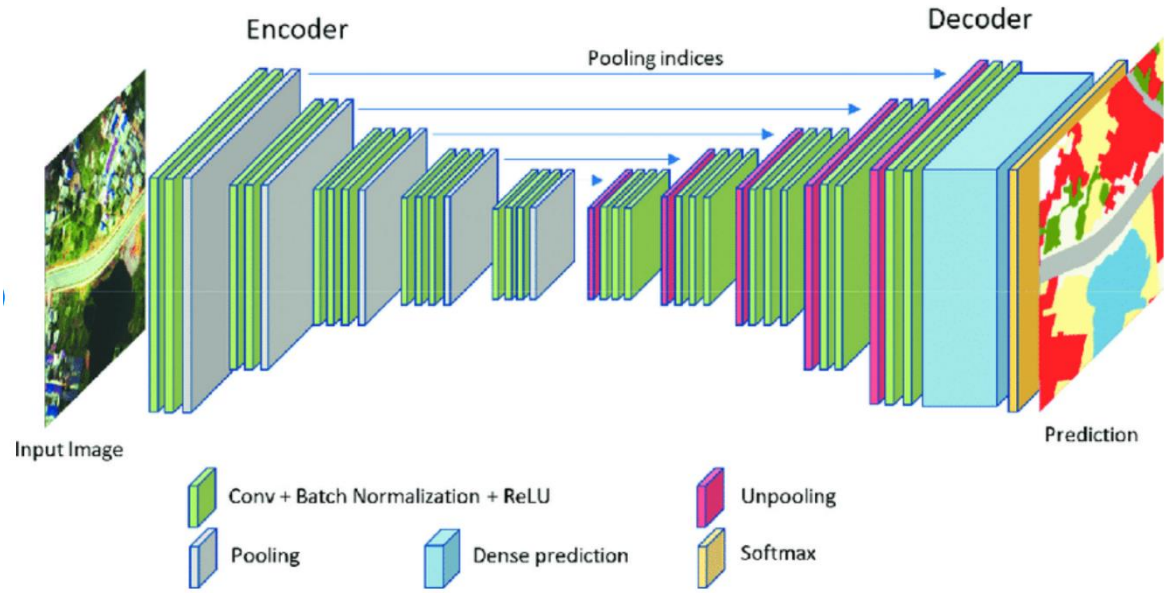
Zdroj: https://www.researchgate.net/figure/VGGNet-architecture-19_fig2_333242381

SegNet

Neuronová síť je složená z kodérů a dekodérů po které následuje klasifikační vrsta pixelů. Architektura sítě kodérů je topologicky shodná s 13 konvolučními vrstvami v síti VGG16. Úlohou sítě dekodérů je mapovat mapy funkcí kodéru s nízkým rozlišením na mapy funkcí s plným rozlišením pro klasifikaci podle pixelů. SegNet spočívá v postupu jakým dekodér „převzorkuje“ své vstupní mapy funkcí s nižším rozlišením. Dekodér používá slučovací indexy vypočítané v kroku maximálního sdružování odpovídajícího kodéru k provádění nelineárního převzorkování. To eliminuje učit se převzorkování. Převzorkované mapy jsou řídké a jsou pak spojeny s trénovatelnými filtry, aby se vytvořily husté mapy prvků. Segnet je navržen tak, aby byl efektivní z hlediska paměti. Je také výrazně menší v počtu trénovatelných parametrů. (Badrinarayanan a kol. 2015).

Obrázek v síti SegNet, který je segmentován na výstupu má stejnou velikost jako při jejím vstupu. Počet kodérů a dekodérů je nastavitelný. SegNet je typem sítě DAG, což znamená, že je sítí s uspořádanými vrstvami jako acyklicky graf. Může mít tedy složitější strukturu ve které vrstvy mají více vstupů z jiných vrstev a výstupů do více vrstev. (Jirkovský, 2017).

Obrázek 6 – Architektura sítě SegNet



Zdroj: https://www.researchgate.net/figure/SegNet-architecture_fig5_343566178

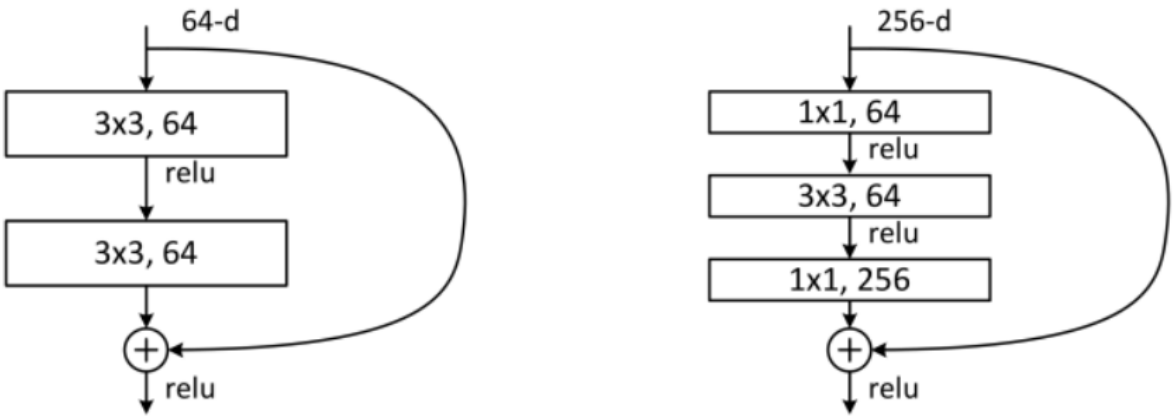
ResNet

Zkratka pro Residual Networks, je klasická neuronová síť používána jako páteřní síť pro mnoho úloh CV. Zásadní průlomem této sítě, že umožňuje trénovat extrémně hluboké neuronové sítě s více než 150 vrstvami. Než přišel ResNet bylo trénování velmi hlubokých sítí složité kvůli problému mizících gradientů (Dwivedi, 2019).

Zvýšení hloubky sítí však nespočívá pouhým skládáním vrstev dohromady. Hluboké sítě je těžké trénovat kvůli problému mizejícího gradientu. Gradient se zpětně šíří do předešlých vrstev. Opakovaným násobením vrstev může způsobit, že bude extrémně malý. Jak se síť prohlubuje, její výkon klesá nebo dokonce začne rychle klesat. ResNet poprvé představil koncept přeskočení spojení.

Je-li aktivována vrstva a rychle přesměrována do hlubší neuronové sítě jedná se o tzv. *„zbytkový blok“*. Ve zbytkovém bloku dochází k aktivaci hlubší vrstvy v síti jak z vrstvy předešlé, tak z mělké horní vrstvy. Tímto způsobem se dá jednoduše trénovat hlubší vrstvy sítě. Teoreticky by se trénovací chyba postupně měla snižovat po přidávání vrstev. V praxi však tradiční neuronové sítě trénovací chyba postupným snižováním dosáhne bodu, kde se začne zvyšovat. ResNet tímto problémem netrpí. Chyba trénování bude stále klesat (Hitchhiker, 2019).

Obrázek 7 – Standardní zbytkový blok a zbytkový blok s úzkým hrdlem



Zdroj: https://www.researchgate.net/figure/Standard-Residual-vs-Bottleneck-Residual-Block-He-et-al-2015_fig4_337486420

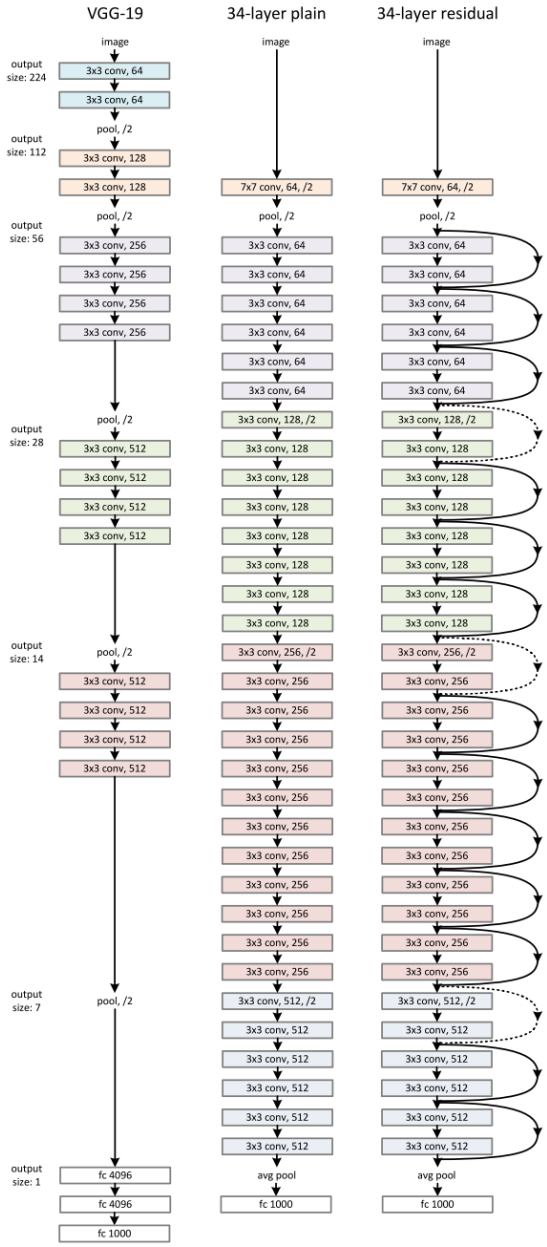
Zbytek se dá chápat v ResNetu jako výsledek hodnoty odhadu odečteného od skutečné hodnoty. Pokud si představíme košík jablek, který váží 3 kilogramy a my odhadneme, že váží 4 kilogramy, tak po odečtení máme hodnotu -1. Tato výsledná hodnota je zbytkový blok (Maladkar, 2018).

Prosté a zbytkové neuronové sítě

Konvoluční vrstvy mají 3x3 filtry a řídí se dvěma pravidly. Výstupní mapy prvků, které mají stejnou velikost mají vrstvy stejný počet filtrů. Pokud je velikost mapy objektů poloviční počet filtrů se zdvojnásobí, aby zachovala časovou komplexitu na jednu vrstvu. Převzorkování se provádí přímo na konvolučních vrstvách, které mají 2 kroky od sebe. Síť končí globální průměrnou sdružovací vrstvou a „tísícecestnou“ plně propojenou vrstvou.

Zbytkové sítě vrstvy lze použít přímo, když vstup a výstup mají stejné rozměry. Když se rozměry zvětšují má také dvě možnosti. Stále v síti probíhá mapování doplněné o nulové prvky navíc vyplněnými rozměry. Tato možnost nezavádí parametr navíc. Možnost druhé zkratky promítání používá ke shodě rozměrů 1x1 filtry. U obou možností, že zkratky jdou přes mapy objektů dvou velikostí, se provádějí dvěma kroky (Maladkar, 2018).

Obrázek 8 – Příklad síťových architektur pro ImageNet



Zdroj: https://arxiv.org/pdf/1512.03385.pdf

3.3.3 Konvoluční neuronové sítě

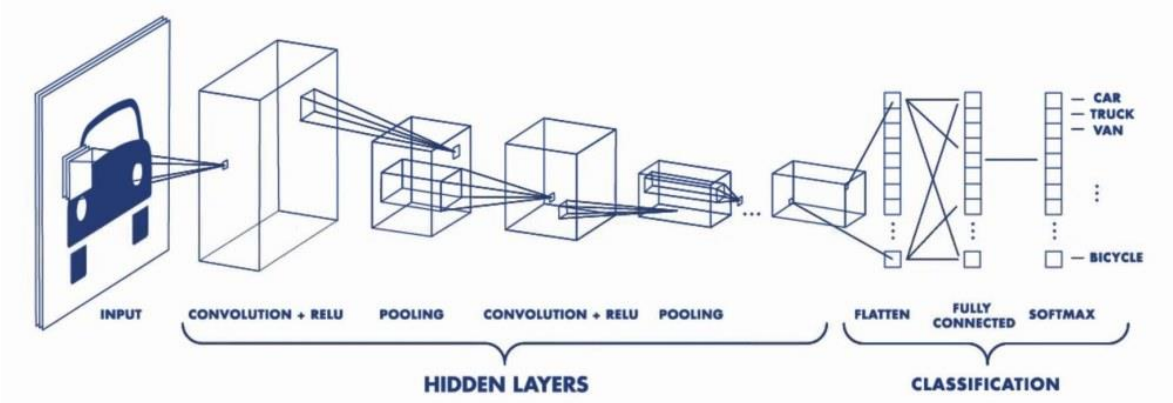
Pokud chceme zvolit vhodný typ sítě pro rozpoznávání a zpracování dat z velkých rozměrů jako jsou obrázky a videa, tak je vhodné využít funkcí z oblasti konvolučních neuronových sítí. Také vhodné pro rozpoznávání (UI) uživatelského rozhraní prvků. CNN má každá tři základní vrstvy:

- Konvoluční vrstvu;
- Pooling vrstvu;
- Plně propojenou vrstvu.

CNN fungují jako ty standardní neuronové sítě. Hlavním rozdílem je v používaná jednotek v konvoluční vrstvě zvaný filtr (kernel), který je aplikován na vstupu této vrstvy pomocí konvoluce.

Každý obrázek je promítán v trojrozměrné matici pixelů. Jde o šířku, výšku a barvu. Filtry mají za úkol na vstupu detekovat jednoduché vzory např. hrany. Konvoluční sítě snižují počet parametrů, která se síť musí naučit (Mikulovský, 2021).

Obrázek 9 – Konvoluční síť v Deep Learningu



Zdroj: https://www.technickytydenik.cz/rubriky/ict/deep-learning-pro-segmentaci-obrazu_42430.html

Konvoluční vrstva

Vrstva, jejíž výskyt v síti je nejdůležitější ze všech ostatních vrstev. Obsahuje sadu filtrů s definovanou velikostí, které jsou nastaveny během tréninku sítě. Často používané rozměry vrstvy jsou 3×3, 5×5, a 7×7. Hloubka těchto filtrů se označuje h a závisí na hloubce vstupních dat. Tyto filtry se často používají u zpracování obrazu.

CNN přináší dvě vlastnosti pro zefektivnění. Počet parametrů pro učení sítě je výrazně snížen použitím malých filtrů. Malé filtry se také učí charakteristické vzory z částí obrazu. Výstupem konvoluční vrstvy je příznaková mapa (Khan, 2018).

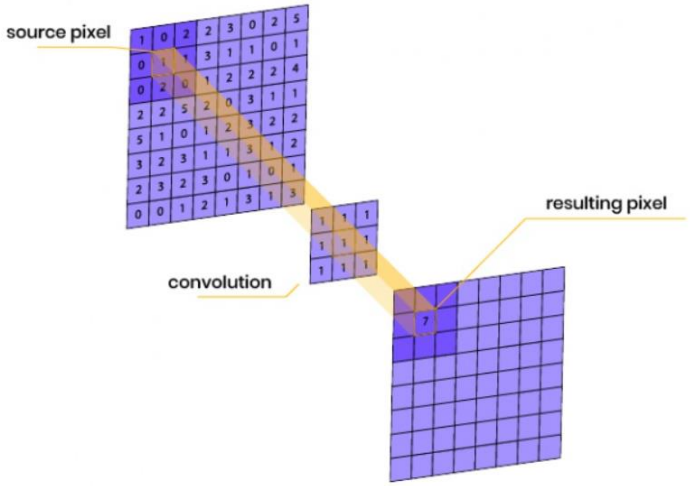
Pooling vrstva

Je následující vrstvou po vrstvě konvoluční v síti. Redukuje rozměry dat. Pooling efektivně vzorkuje vstupní příznakovou mapu. Jak už bylo zmíněno konvoluční sítě mají vlastnost snížení potřebných parametrů k učení. Tato vrstva tyto parametry snižuje. Stejně jako tomu je konvoluční vrstvy je potřeba definovat rozměr a délku kroku posunu filtru. To se provádí funkcí „max pooling“ a „average pooling“. Max pooling pracuje s nejvyšší hodnotou pro zpracování a average pooling s průměrem hodnot (Khan, 2018).

Plně propojená vrstva

Všechny neurony ve vrstvě jsou propojeny mezi sebou navzájem s ostatními předchozími vrstvami sítě, aby identifikovala větší vzory. Tato vrstva je kombinací všech vzorů naučené předchozími vrstvami. Vrstvy plně propojené se v síti nachází na konci. Existují, ale architektury, kdy se nachází někde v prostřední části sítě (Khan, 2018).

Obrázek 10 – Architektura konvoluční vrstvy



Zdroj: https://www.optixs.cz/slovnik-17/dekonvoluce-a-konvoluce-68s

3.4 Computer Vision

Computer Vision, česky Počítačové vidění, je vědecký obor, který se zabývá tím, jak vizuální obrazy (resp. videa, fotky, obrázky) mohou počítačům pomoci dosáhnout porozumění reálného světa na vysoké úrovni. Porozumění implikuje transformací vizuálních obrazů na vysvětlení světa (sítnice), které tvoří smysl pro myšlenkové procesy a dokáže vyvolat účinnou reakci. Tato interpretace může být viděno jako uvolnění symbolických znalostí od obrazových dat pomocí vyvinutých modelů s pomocí geometrie, fyziky, statistiky a teorie učení. Disciplína konstrukce systémů Computer Vision si klade za cíl aplikovat své teorie a modely. Rekonstrukce místa, identifikace událostí, sledování videa a rozpoznávání objektů, 3D odhad pozice, učení, indexování, odhad pohybu, modelování 3D scény a obnova obrazu jsou subdomény Computer vision.

CV je velice závislý na aplikaci systémů. Přesná implementace systémů často závisí na tom, zda je jeho funkčnost předem specifikovaná, zda se nějaký jeho aspekt může naučit

nebo změnit během služby. V mnoha počítačových systémech však existují standardní funkce Computer vision:

- Pořízení snímku – digitální snímek je vytvořen jedním nebo více obrazovými snímači, které zahrnují senzory vzdálenosti, radar a ultrazvukové kamery. Výstupní obrazová data jsou potom obyčejný 2D, 3D obraz nebo série snímků podle toho jakého typu je snímač;
- Předzpracování – za účelem získání konkrétní informace před počítačem. Je důležité data zpracovat, aby bylo zajištěno, že se bude řídit některými předpoklady implikovaným systémem;
- Extrakce rysů – z obrazových dat jsou rysy obrazu na různých úrovních složitosti extrahovány. Textura, forma nebo pohyb mohou souviset se složitějšími funkcemi.
- Detekce/segmentace obrazu – rozhodnutí, které oblasti nebo objekty jsou na snímku důležité k dalšímu zpracování;
- Zpracování na vysoké úrovni – vstupem je obvykle malý soubor dat například soubor bodů nebo oblast obrazu, o které se předpokládá, že obsahuje určitou entitu;
- Rozhodování – učinit potřebné rozhodnutí. Rozhodnutí o úspěšném nebo neúspěšném kroku u systémů s automatickou kontrolou. Rozpoznaná shoda a neshoda obrazu (Shahdadpuri, 2020).

3.4.1 Historie Computer visionu

V dnešní době je zvýšený zájem o používání mobilních kamer, a to znamená jenom jediné a to, že roste neustálý tok snímků a videí. Technologie Computer vision se stalo snadno dostupným. Za méně, než deset let se míra přesnosti pro rozpoznávání objektů zvedla z 50 % na 99 % a dnešní systémy jsou ještě přesnější než lidé. V 50.letech, kdy rané neuronové sítě začaly detekovat hrany objektů a organizovat je podle jejich typů podnikl Computer vision první kroky k velikosti. První komerční systémy Computer vision byly používány již v 70.letech, využívající rozpoznávání znaků ke čtení psaného text pro nevidomé. Sbírky obrázků se staly dostupnými online ke kontrole jako internet se vyvinula v 90.letech. To vedlo k rozvoji programů rozpoznávání obličejů a později k rozpoznávání i objektů. (Shahdadpuri, 2020).

3.4.2 Hluboké učení v Computer vision

Metody zpracování obrazu lze rozdělit do dvou kategorií. Úprava obrazu a CV. V kategorii úprava obrazu jde spíše o úpravu za použití různých filtrů, redukcí šumu, stylizačních nástrojů, transformaci obrazu, úpravy kontrastu, jasu a barevné škály. Také kromě vzhledových úprav je schopen detekovat objekty na základě informací barev, morfologických operacích, prahování a tvorbě masky. Computer vision vylepšuje detekci o možnost klasifikovat a sledovat objekty na základě jejich parametrů. Existuje mnoho softwarů, které obsahují velmi mnoho inženýrských nástrojů a disponují velice profesionálním vzhledem pro vědecké i technické výpočty pro použití specializovaných algoritmů použitelných k efektivním řešením z oboru CV. (Jirkovský, 2017).

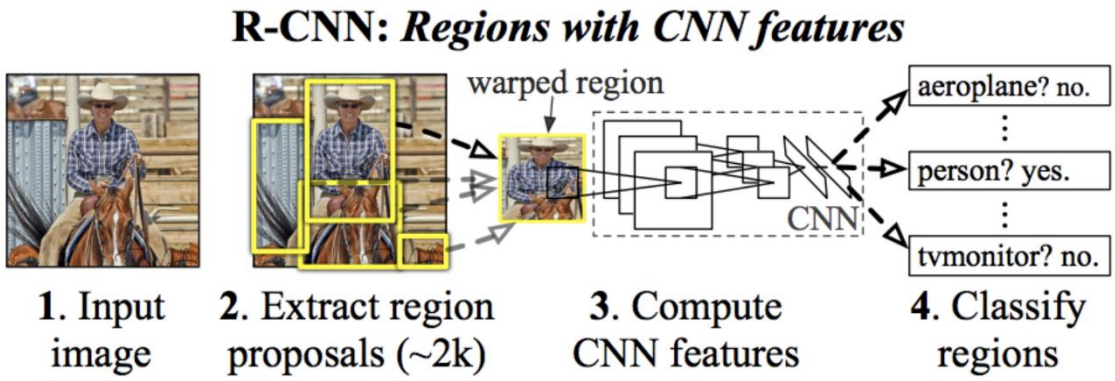
3.5 Architektury CNN

3.5.1 R-CNN (Region Based Convolitonal Neural Network)

Tento druh neuronové sítě se využívá hlavně pro detekci objektů v obraze posuvným oknem. Při použití této metody stačí projít celý obrázek s různě velkými obdélníky a podívat na tyto menší obrázky metodou hrubé síly. Problém je, že bude obrovské množství malých obrázků (Elfouly, 2019).

Proto Ross Girshick navrhl metodu selektivního vyhledávání k extrahování 2000 oblastí z obrázku a nazval ji „region proposals“. Je možné tedy pracovat při klasifikaci s velkým počtem regionů, které jsou generovány pomocí selektivního algoritmu. Těchto 2000 regionů je shlukováno do čtverce a přiváděno do konvoluční neuronové sítě, která vytváří výstup 4096rozměrný příznakový vektor. CNN extrahuje rysy a prvky z obrazu do klasifikátoru SVM. Kromě predikce přítomnosti objektu v návrzích oblasti algoritmu také předpovídá čtyři hodnoty, které jsou offsetovými hodnotami, aby se zvýšila přesnost ohraničeného rámečku. R-CNN se ale nezbavila problému s trváním procesu učení. Nelze implementovat v reálném čase. Testovací snímek trvá přibližně až 47 sekund. Dalším problémem by mohl být algoritmus selektivního vyhledávání, který je pevným algoritmem, tím neprobíhá žádné učení v této fázi. To by mohlo vést ke generování špatných návrhů regionů (Gandhi, 2018).

Obrázek 11 – Architektura R-CNN

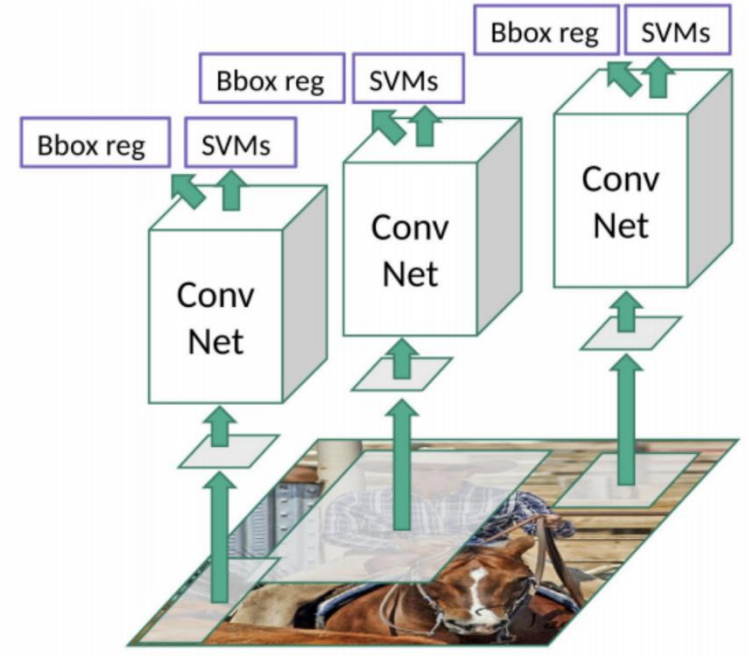


Zdroj: https://medium.com/dair-ai/papers-explained-14-rcnn-ede4db2de0ab

SVM klasifikátor

Klasifikátor „Support vector machines“ neboli metoda podpůrných vektorů je metoda z oblasti strojového učení sloužící pro klasifikaci. Základem je lineární klasifikátor do dvou tříd. Pracuje z tzv. nadrovinami, které prostory příznaků optimálně rozdělují tak, že trénovací data náležející odlišným třídám leží v opačných poloprostorech. Na popis nadroviny stačí body ležící na okraji pásma necitlivosti. Toto pásmo se nachází okolo nadroviny na obě strany (Elfouly, 2019).

Obrázek 12 – SVM metoda za použití konvoluční vrstev pro klasifikaci obrazu



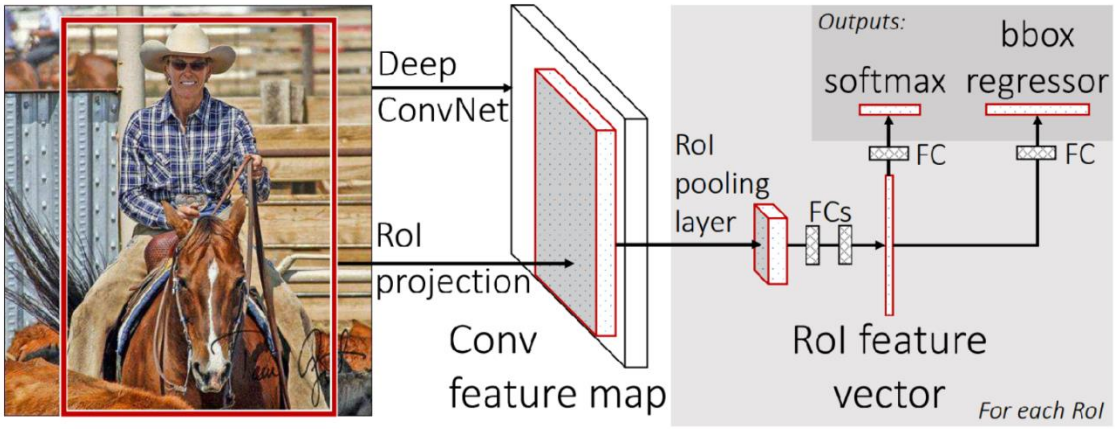
Zdroj: https://www.datasciencecentral.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms/

3.5.2 Fast R-CNN (Fast Region Based Convolitonal Neural Network)

Tato architektura je podobná R-CNN, která řeší jeho nedostatky. Princip modelu spočívá v tom, že místo posílání předvídaných regionů na vstup CNN, přivádíme vstupní obraz (Girshick, 2015).

Z mapy konvolučních prvků identifikujeme předvídaných regionů a zformuje je do čtverců a pomocí sdružovací vrstvy RoI je přetváříme na pevnou velikost, aby je bylo možné vložit do plně propojené vrstvy. Z vektoru vlastností RoI používáme vrstvu softmax k předpovědi třídy navrhovaného regionu a také hodnot offsetu pro ohraničující rámeček. Důvodem, proč je rychlejší Fast R-CNN než R-CNN je ten, že nemusíme pokaždé dodávat 2000 návrhů regionů. Místo toho se operace konvoluce provádí pouze jednou na snímek a je z ní generována mapa prvků (Gandhi, 2018).

Obrázek 13 – Architektura Fast R-CNN



Zdroj: https://arxiv.org/pdf/1504.08083.pdf

3.5.3 Faster R-CNN (Faster Region Based Convolitonal Neural Network)

Architektury R-CNN a Fast R-CNN používají selektivní algoritmy. Selektivní vyhledávání je pomalý a časově náročný proces ovlivňující výkon sítě. Proto se přišlo s algoritmem, který eliminuje algoritmus selektivního vyhledávání a umožňuje síti naučit se návrhy regionů. Namísto použití selektivního algoritmu na mapě prvků je použita samostatná síť. Návrhy předvídaných regionů jsou poté přetvořeny pomocí sdružovací vrstvy ROI, která se pak používá ke klasifikaci obrazu v rámci navrhovaného regionu a předpovídání hodnot posunu pro ohraničující rámečky (Gandhi, 2018).

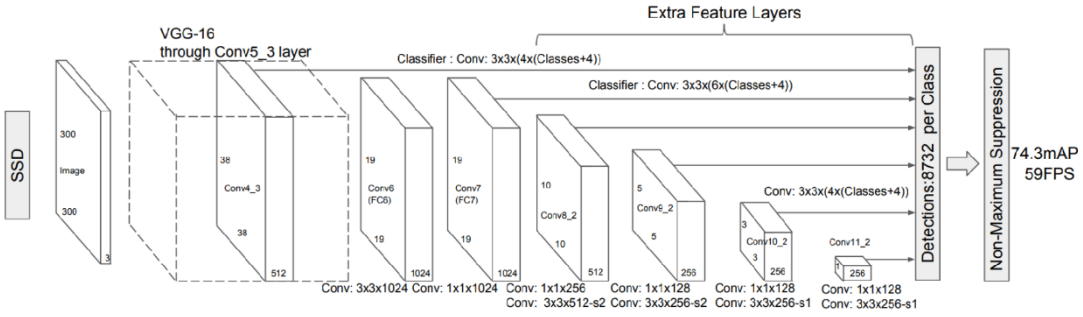
Detekční systém Faster R-CNN architektury je složen ze dvou modulů (RPN – Region Proposal Network) a druhý je detektor Fast R-CNN. První modul je hlubokou konvoluční sítí, která navrhuje regiony. Druhý modul, detektor, tyto navrhované regiony

využívá. RPN dává pokyny detektoru, kde dané objekty, co je nutno hledat, najde. Systém tvoří jednotnou síť (Mikulský, 2021).

3.5.4 SSD

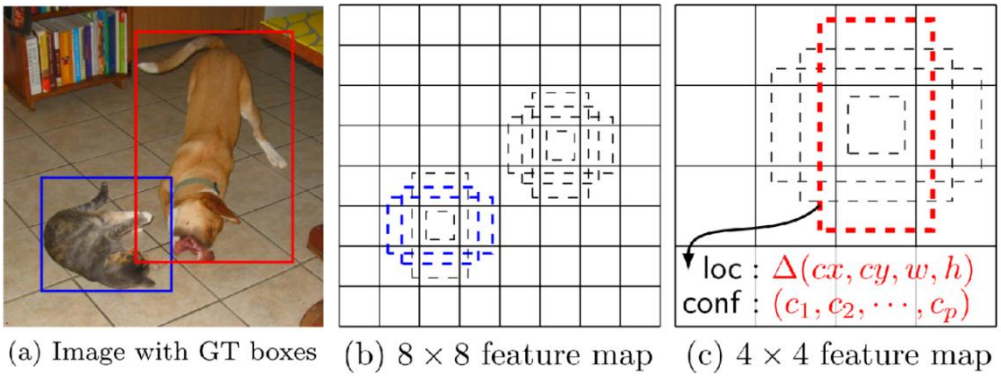
Metoda (Single Shot Multibox Detector) vytvořena v roce 2016. Je založena na jednoduchosti v dopředné konvoluční neuronové sítě. Vytváří skupinu „bounding boxů“ a skóre přítomnosti třídy objektu v boxech. Je postavena na architektuře VGG-16, ale neobsahuje plně propojené vrstvy. Architektura VGG-16 byla použita z důvodu její schopnosti vysoce kvalitní kvalifikaci obrazu. Následující vrstvy, po VGG-16 části, jsou konvoluční, které umožňují detekci pro různá měřítka a poměry. Nepoužívá síť RPN pro návrh regionů jako je tomu u R-CNN. SSD jí nahrazuje malými konvolučními filtry, které i mimo jiné zajišťují klasifikaci. Detekční část pracuje s odlišnými příznakovými mapami. Počet predikcí se tedy zvyšuje. Příznakové mapy jsou děleny na buňky a počet buněk určuje mřížka (Liu, 2016).

Obrázek 14 – Architektura SSD



Zdroj: https://medium.com/@aiclubvast/road-damage-detection-system-895d3a36613a

Obrázek 15 – SSD příklad postupu



Zdroj: https://www.researchgate.net/figure/SSD-default-boxes-GT-stands-for-ground-truth-Figure-taken-from-1_fig2_329747508

3.5.5 YOLO

Originální verze YOLO „You Only Look Once“ byla představena v roce 2012, přestavuje algoritmus využívající neuronové sítě k co nejrychlejší a nejpřesnější detekci objektů v reálnem čase, díky čemuž je tak populární.

Je úspěšně využíván v různých aplikacích pro rozpoznávání dopravních signálů, identifikaci osob, správu parkovacích automatů a sledování zvířat. V YOLO je detekce objektů realizována jako regresní úloha, přičemž poskytuje pravděpodobností příslušnost k jednotlivým třídám detekovaných obrázků. Využívá konvolučních neuronových sítí (CNN), ale pracuje s pouhým jedním průchodem dopředným šířením neuronových sítí k detekci objektů. Z toho vyplývá, že v celém obrázku se provádí predikce v jediném běhu algoritmu (Karmini, 2021).

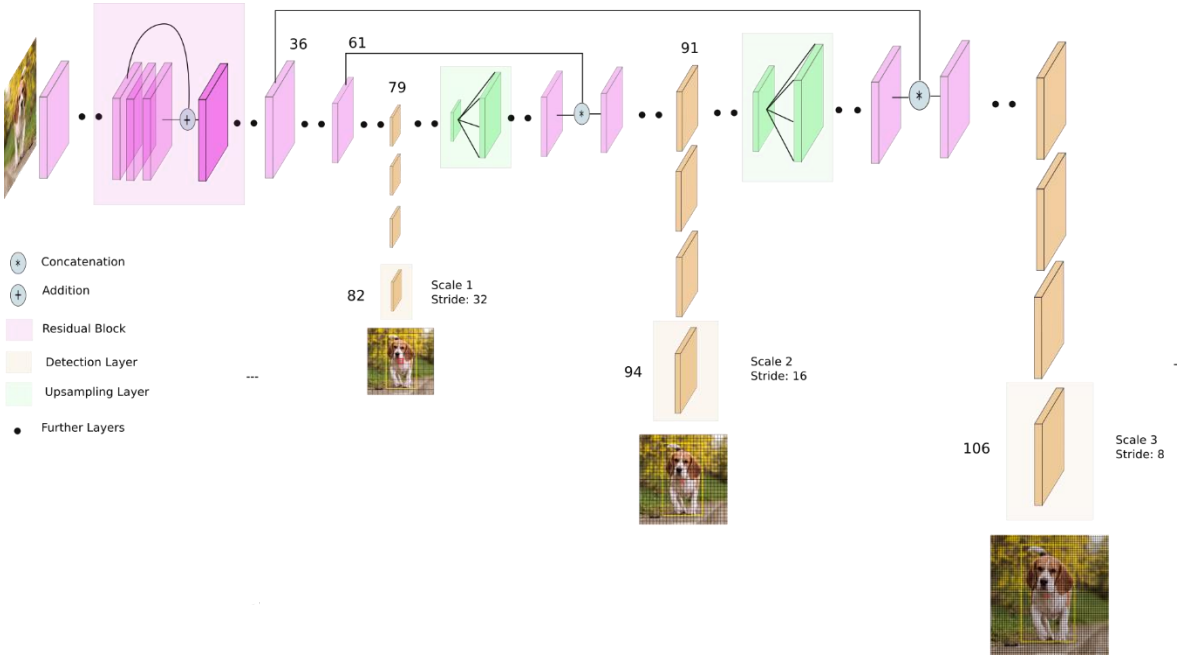
Vývoj YOLO

Algoritmus má několik verzí (YOLOv1, YOLOv2, YOLOv3, YOLOv4 a verze TinyYOLO). Mezi ty naprosto běžné patří YOLOv1 a YOLOv3.

První verze pracuje s jedinou hlubokou konvoluční neuronovou sítí (CNN), která využívá 2 plně propojené vrstvy a 24 konvolučních vrstvev od čehož se odvíjí velikost, jejíž hodnota dosáhla 1 GB. Vysoké požadavky jsou kladeny jak na úložný prostor, tak vysoký výkon používané platformy. YOLOv1 rozděluje snímek do mřížky a provede předpovědi v každé buňce mřížky. Každá buňka je zodpovědná za predikci pevného počtu ohraničujících rámečků a jejich odpovídajících tříd pravděpodobnosti. Takto dosáhla detekci objektů v reálnem čase působivou rychlostí, ale měla určité omezení v detekci malých objektů a přesné lokalizaci překrývajících se objektů. Verze druhá je rychlejší a robustnější verzí první. YOLOv2 odstraňuje plně propojené vrstvy a zavádí kotevní rámečky k lepšímu předvídání ohraničujících rámečků různých velikostí a poměrů stran. Kombinace kotevních rámečků a víceúrovňového trénování pomocí datových sad PASCAL VOC a COCO pomohla zlepšit detekci malých objektů a zvýšit počet detekovaných tříd. V YOLOv3 je využito konceptu pyramidových sítí s více vrstvami (53 vrstvami, nazvanou Darknet-53) detekce pro další prohloubení síťové vrstvy. Umožňuje modelu detekovat objekty v různých měřítkách a rozlišeních. Při běhu na výkonném GPU zařízení dosáhly YOLO a její vylepšení vysoké úrovně přesnosti a rychlosti. Avšak velikost modelů těchto algoritmů pro detekci objektů je příliš velká pro prostředí omezená svým úložným prostorem a pamětí. To znemožňuje provozovat tyto algoritmy v omezených prostředích v reálném čase. YOLOv3

využívá pro detekci tři různá měřítka mřížky: 13x13, 2626 a 52x52. Existuje, ale alternativní menší verze YOLOv3, Tiny-YOLOv3 a Tinier-YOLOv3 (Rustamy, 2023) (Karmini, 2021).

Obrázek 16 – Architektura YOLOv3



*Zdroj: https://miro.medium.com/v2/resize:fit:2000/format:webp/1*d4Eg17IVJ0L41e7CTWLLSg.png*

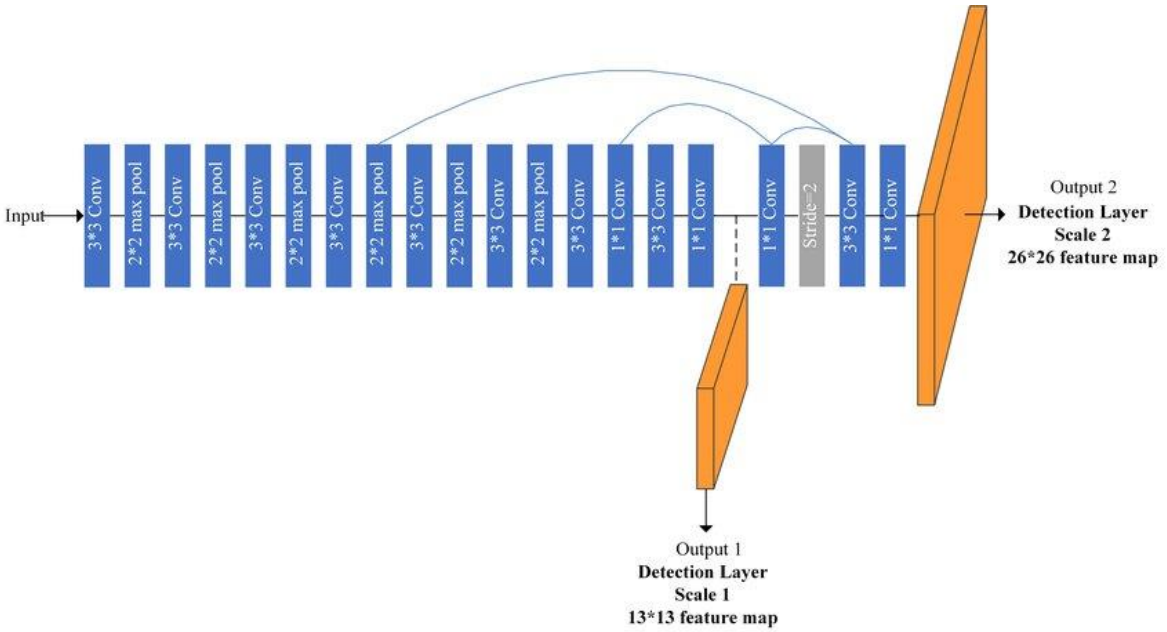
YOLOv4 bylo představeno jako vylepšená verze předchozích verzí YOLO, například zavedením několika vylepšených sítí. CSPDarknet53 anebo CSPResNeXt50 zde hrají roli páteřních sítí k lepšímu extrahování funkcí ze vstupního obrazu. PANet s FPN (Feature Pyramid Network) spojuje funkce v různých měřítkách, aby se zlepšila výkonnost detekce objektů. FPN využívá sílu pyramidních funkcí, které zlepšují přesnost pro detekci objektů různých velikostí. Propojuje úrovně pyramidy horizontálně i vertikálně. To znamená, že informace mohou cirkulovat mezi úrovněmi vrstev a umožňují lepší propojení informací o detailu v kontextu s informováním. Vyšší úrovně pyramidy jsou schopny zachytit širší kontext, zatímco nižší úrovně jsou schopny zachytit detaily. Za účelem zvýšení obecnosti modelu implementuje YOLOv4 různé metody augmentace dat, včetně techniky mozaiky, náhodných tvarů a mixování (Kushwah, 2023).

3.5.5.1.1 Tiny-YOLOv3

Jak už název napovídá, Tiny-YOLOv3 jde o zmenšenou verzi YOLOv3, která je vhodná pro omezené prostředí a je zároveň nejnovějším vylepšením YOLO. Přesnost kvůli jeho velikosti však není moc vysoká a výkon detekování v reálnem čase není uspokojivý

jako je tomu u jeho větší verze. Struktura sítě se skládá ze sedmi konvolučních vrstev a šesti maxpool vrstev pro extrakci prvků snímku a dvě měřítka detekčních vrstev. Konvoluční vrstvy využívají konvoluční filtry 512 a 1024, což způsobuje hromadění velkého množství parametrů a je nutno většího uložiště, kdy poté pak nastává zpomalení detekce na zařízeních omezených svými možnostmi. Další nevýhodou je také nízká přesnost detekce i může být nižší z důvodu neoptimálních metod pro kompresi dat v síti. Na řešení těchto nedostatků se zaměřuje model Tinier-YOLO (Sánchez Leal a kol., 2023).

Obrázek 17 – Struktura vrstev Tiny-YOLOv3



Zdroj: https://www.researchgate.net/publication/338162578/figure/fig1/AS:839998031032320@1577282545408/The-network-structure-of-Tiny-YOLO-V3.jpg

3.5.5.1.2 Tinier-YOLO

Je navržen ke snížení velikosti modelů YOLO společně s dosažením zlepšené přesnosti detekce v reálném čase. Inspiroval se „fire“ modulem ze SquuezeNet, aby se snížily parametry modelu, což pomáhá zmenšit velikost, oproti tradičním plně propojeným vrstvám. Fire modul je základní stavební blok, který kombinuje konvoluční vrstvy s 1x1 a 3x3 filtry. Struktura „fire“ modulu obvykle zahrnuje:

1. Squeeze layer (stlačovací vrstva): Skládá se z konvolučních bloků s 1x1 filtry, které slouží k redukci dimenze dat a tím snižují počet kanálů;

2. Expand layer (rozšiřovací vrstva): Tato vrstva obsahuje jak konvoluční bloky s 1x1 filtry (pro zvýšení dimenze), tak i bloky s 3x3 filtry, které zachycují vzory a informace v datech.

Fire modul tedy umožňuje stlačit informace do nižší dimenze (redukce počtu kanálů) a následně je rozšířit do vyšší dimenze, což zlepšuje efektivitu a umožňuje lépe pracovat s výpočetně náročnými úlohami při zachování výkonu s menším počtem parametrů.

Další důležitou výzvou je definovat styl připojení mezi fire moduly, aby se dále zvýšila přesnost detekce a výkon v reálném čase. Proto se v Tinier-YOLO používá průchozí vrstva k vyřešení tohoto problému, kterým můžeme sloučit mapy prvků z přední vrstvy pro získání jemnozrnných rysů (Fang, Wang, Ren, 2020).

3.5.5.1.3 YOLOv5

Jde o nezávislý projekt od Ultralytics vytvořený Glennem Jocherem, představený v roce 2020, který přináší jednoduchost použití, rychlost a možnost přizpůsobení. YOLOv5 bylo navrženo s ohledem na snadné použití a efektivitu. Původně neobsahoval žádný modelový kód. Později však přidal implementaci YOLOv3 s potvrzením „Pozdravy YOLOv5“. Jocherova implementace YOLOv5 se od předchozích verzí liší v několika významných aspektech. Prvním z nich je absenci veřejně dostupné dokumentace doprovázejícího jeho vydání. Za druhé, Jocher implementoval YOLOv5 nativně v PyTorch, což je odlišnost od předchozích modelů YOLO, které využívaly framework Darknet. Jocher rovněž přispěl k vytvoření rozšíření mozaikových dat, které byly zahrnuty do jeho úložiště YOLOv3, a to je jedno z několika nových rozšíření dat využívaných v YOLOv4. Jocher obdržel uznání ve formě potvrzení v dokumentaci YOLOv4 (Nelson a Solawetz, 2020).

3.5.5.1.4 YOLOv8

Ultralytics, která také vytvořila model YOLOv5 vyvinula tuto nejnovější verzi modelu YOLO a je postavena na špičkovém vývoji v hlubokém učení a Computer vision. YOLOv6 byl open source společností Meituan v roce 2022 a používá se v mnoha autonomních doručovacích robotech společnosti. YOLOv7 přidal další úkoly, jako je odhad pozice v datové sadě klíčových bodů COCO. Model YOLOv8 staví na úspěchu předchozích verzí a přináší nové funkce a vylepšení pro vyšší výkon, flexibilitu a efektivitu. YOLOv8 podporuje celou řadu úkolů vize AI, včetně detekce, segmentace, odhadu pozice, sledování a klasifikace. Tato všestrannost umožňuje uživatelům využívat schopnosti YOLOv8 napříč různými aplikacemi a doménami (Jocher a Chaurasia 2023).

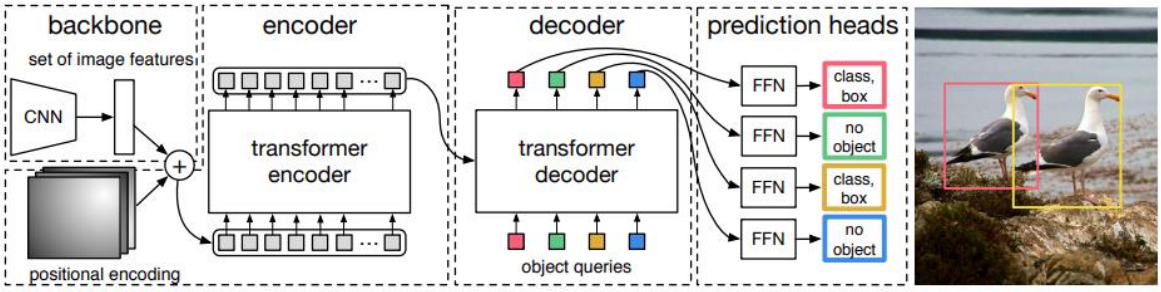
Spolu s vylepšeními samotné architektury modelu představuje YOLOv8 vývojářům nové přátelské rozhraní prostřednictvím balíčku PIP pro použití modelu YOLO. YOLOv8 obsahuje četné změny a vylepšení architektonických a vývojářských zkušeností oproti YOLOv5. YOLOv8 je v neustálém vývoji, jelikož vývojáři Ultralytics vzájemně spolupracují s komunitou, která jim dává neustálou zpětnou vazbu, se kterou poté vývojáři pracují a snaží se o co největší zdokonalení. (Solawetz a Francesco, 2023).

3.5.6 DETR

Představuje inovativní přístup k detekci objektů. Predikuje všechny objekty najednou a je trénován end-to-end to znamená, že celý proces detekce objektů, včetně extrakce příznaků, reprezentace objektů, klasifikace a lokace objektů, je řešen jako jediný end-to-end model, oproti tradičním metodám, které využívají konvoluční sítě s post-processingem pro zpracování bounding boxů. Místo používání klasických anotací bounding boxů DETR pracuje s tzv. "většinovým výběrem" (set-based selection), kde jsou výsledky detekce reprezentovány množinami objektů, nikoli individuálními bounding boxy. Přístup "end-to-end" může přinést výhody v jednoduchosti implementace, optimalizaci a škálovatelnosti.

Ve srovnání s většinou předchozích architektur jsou hlavními výhodami DETR spojení bipartitního (dvoudílního) přizpůsobení ztráty a transformátorů s (neautoregresivním) paralelním dekódováním. Dosahuje srovnatelných výkonů jako model R-CNN. (Karion a kol., 2020).

Obrázek 18 – Model DETR



Zdroj:https://lh5.googleusercontent.com/rPXh8HHA-KnqdFuTjXsqBLS3A1y1QGZ5eQnqt3RFV-ZcogCZlku7VpnFFeIcIBmIYN4nevzDlWOEq69nS2d_ZDWM5fO_RgZ7OJrAh6X5h3OvLLXlqBDc3vGvTvfQRTG2KAu7Mv3tH4QJ4Nl_OtSHxdo

DETR začíná se vstupním obrazem, který projde konvolučním blokem (backbone). Tento blok extrahuje různé úrovně příznaků z obrazu. Obraz se zploští a doplní o poziční kódování před vstupem do transformátoru dekodéru. DETR používá transformační enkodér-dekodér (Transformer), který převádí vstupní embeddingy objektů na výstupní embeddingy

s přiřazením tříd a polohou. Transformátorový dekodér přijímá jako vstup malý pevný počet naučených polohových vložení, kterým říkáme objektové dotazy, a navíc se stará o výstup kodéru.

Objektové dotazy jsou počet n vstupních vložení, které se liší aby model dosahoval různých výsledků. Jsou naučená pozičním kódováním a přidávána na vstup každé „attention“ vrstvy. Tyto vrstvy jsou klíčovým prvkem, který umožňují modelu efektivně zachytit vzájemné závislosti mezi různými částmi vstupních dat, zejména při zpracování sekvencí. Jejich použití je v různých částech modelu, zejména v enkodéru pro zpracování vstupních dat a v dekodéru pro generování výstupních dat.

Počet *n* objektových dotazů je transformováno do výstupního vložení dekodérem a poté jsou nezávisle dekódovány do souřadnic krabic a štítků tříd pomocí dopředné sítě, což vede k *n* konečným předpovědím. Konečná předpověď je vypočítána pomocí FFN. Obvykle obsahuje dvě plně propojené vrstvy s funckí ReLU, následovaná druhou plně propojenou vrstvou. FFN má svojí roli jak v post-processingu a dalším zpracovávání příznaků z předchozích vrstev, kde pomáhá modelu adaptovat se na specifika úloh detekce objektů a vytvářet vhodné reprezentace pro následné kroky predikce, tak v enkodéru i dekodéru. (Karion a kol. 2020).

3.6 Analýza současných přístupů k detekci UI objektů

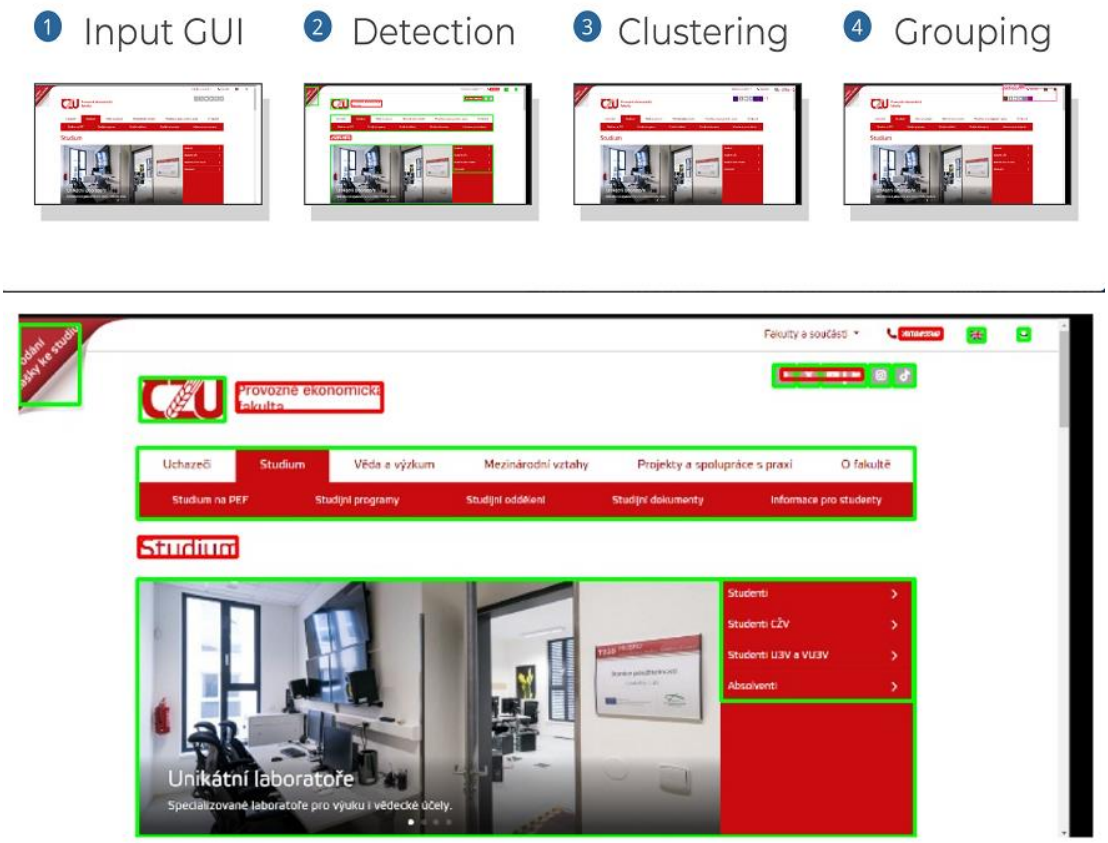
Momentálně je oblast detekce prvků v uživatelském rozhraní (UI) ve fázi intenzivního vývoje a inovací. S rostoucím důrazem na uživatelskou zkušenost a efektivitu aplikací stoupá i potřeba přesného a spolehlivého rozpoznávání a detekce prvků UI. Nástroje pro detekci prvků UI, využívající technologie jako CV a strojového učení směřují k minimalizaci chyb a optimalizaci výkonu těchto systémů. Snaží se adaptovat na různé platformy, včetně mobilních zařízení, webových stránek a desktopových aplikací. Celkově lze vidět rostoucí význam detekce prvků v UI a její vliv na automatizaci a vývoj aplikací.

3.6.1 UIED (UI Element Detection)

Jde o staromódní přístup k detekci UI prvků založeném na CV. Tento projekt od uživatele MulongXie ze stránky github.com, je v neustálém vývoji a má i svojí webovou aplikaci (http://uied.online/#carousel-app). Vstupem pro UIED můžou být obrázky uživatelského rozhraní (příklad obrázek č.19), což může být jak snímek z obrazovky počítače, tak i telefonu. Zajímavým vstupem může být také návrh stránky UI vytvořený

v aplikacích jako je Photoshop nebo Sketch, dokonce i ručně kreslený návrh. Model pak detekuje a klasifikuje textové a grafické prvky v UI a exportuje výsledek detekce jako soubor JSON pro budoucí aplikaci. UIED se skládá ze dvou částí pro detekci textu a grafických prvků uživatelského rozhraní, jako je button, obrázek a input box. U textu využívá Google OCR k detekci a pro grafické prvky používá přístupy CV (OpenCV a Pandas) k umístění prvků a klasifikátor CNN k dosažení klasifikace. Výsledky detekce se dají přímo v aplikaci využít pro generování kódu HTML a CSS (MulongXie, 2022).

Obrázek 19 – Příklad vstupního snímku pro UIED



Zdroj: UIED (2024)

3.6.2 Imgcook

Podobně jako u UIED platforma Imgcook pro generování kódu společnosti Alibaba, je navržena tak, aby umožnila vývojářům generovat uživatelská rozhraní automaticky na základě návrhů UI z Photoshopu, PSD, Skica, statických obrázků nebo dalších vizuálních vstupů. To může usnadnit synchronizaci mezi návrháři a vývojáři. Z návrhů rozděluje div prvky, obrázky a rozpětí. Bere vizuální návrhy jako vstupy a generuje udržovatelný frontend kód jediným kliknutím pomocí inteligentních technologií. Generování kódu návrhů Sketch

nebo Photoshop vyžaduje instalaci zásuvných modulů. Pomocí modulu plug-in imgcook můžete exportovat popis objektové notace JavaScript (JSON) vizuálních konceptů a vložit jej do vizuálního editoru imgcook a poté jej v editovacím prostředí upravovat pohledy a logiku a měnit popis JSON (Suchuan, 2021).

3.6.3 Srovnání tradičních a hlubokých metod detekce

V tomto výzkumu bylo autory provedeno, rozsáhlou empirickou studií sedmi reprezentativních metod detekce prvků GUI na více než 50 000 obrazech GUI, porovnání vyspělých metod z oboru Computer vision (CV), včetně zastaralých metod, které se spoléhají na tradiční funkce zpracování obrazu (např. nenápadné okraje, obrysy) a modely hlubokého učení, proto aby byly pochopeny možnosti, omezení a efektivní návrhy těchto metod. Tato studie nejen vrhá světlo na technické výzvy, které je třeba řešit, ale také informuje o návrhu nových metod detekce prvků GUI. V rámci tohoto průzkumu byl navrhnut nový přístup pro detekci netextových prvků uživatelského rozhraní (UI), který je specifický pro grafické uživatelské rozhraní (GUI). Tato metoda využívá novou strategii, která postupuje od hrubého k jemnému vyhledávání shora dolů. Navržený přístup je integrován do pokročilého modelu hlubokého učení určeného pro detekci textu v GUI. Výsledky z analýzy 25 000 obrázků GUI ukazují, že navržená metoda výrazně zlepšuje úspěšnost detekce prvků v grafickém uživatelském rozhraní. V experimentu s novým přístupem byly porovnávány modely jako je Faster R-CNN, YOLOv3, CenterNet, Xianyu a REMAUI (Chen, Xie, Xing, Chen, Xu, Zhu, 2020).

4 Praktická část práce

4.1 Příprava vhodného datasetu

Správná detekce objektů založených na hlubokém učení je zpravidla závislá na důkladném trénování určitým datasetem. Dataset je rozdělen do 3 sad (train, valid, test). Sad train obsahuje obrázky a odpovídající anotace, které jsou používány k trénování modelu strojového učení. Model se na těchto datech učí a snaží se získat znalosti potřebné k efektivnímu rozpoznání nebo klasifikaci objektů ve snímcích. Sada validation obsahuje nezávislý soubor obrázků a anotací, který se používá k pravidelné validaci modelu během trénování. A nakonec složka test obsahuje další nezávislý soubor obrázků a anotací, který se používá ke konečnému vyhodnocení výkonnosti modelu po jeho dokončení tréninku. Testovací data by měla být pečlivě vybrána a nesmí být použita při trénování ani validaci, aby poskytovala objektivní hodnocení modelu na nových datech. Model je pravidelně ověřován na těchto datech, aby bylo možné sledovat jeho výkonnost na nových neviděných datech a zabránit přeučení (overfittingu). Takto rozřazený dataset lze exportovat ve formátech, které jsou kompatibilní s různými frameworky pro strojové učení, například TensorFlow, PyTorch, YOLO a další.

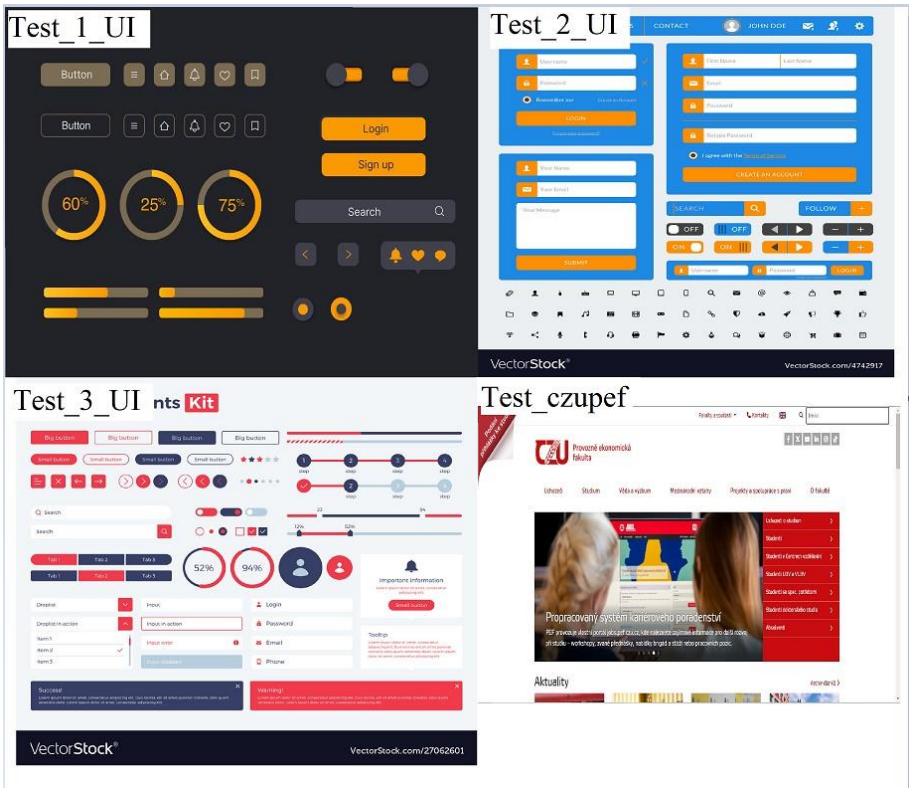
Proto je nutné si nějaký takový dataset připravit nebo použít již vytvořený dataset. Je mnoho online nástrojů, které nabízí již vytvořené datasety nebo se zde dají vytvořit podle potřeb uživatele. Mezi nejpoužívanější patří Kaggle, UCI Machine Learning Repository, Google Dataset search nebo Roboflow. Vybráno bylo pro bakalářskou práci prostředí Roboflow z důvodu snadné dostupnosti a organizace dat, různorodost dostupných datasetů a integrace s populárními platformami nebo nástroji a mimo jiné i s modelem YOLOv8. Byl použit již předvytvořený dataset UI_elements (zdroj: https://universe.roboflow.com/cha-ebebh/cingoz_/dataset/2).

Dataset je rozdělen na tři části: Train set s 3420 snímky, Valid set s 427 snímky, Test set s taktéž 427 snímky s celkovým počtem 4274 snímků a obsahuje 8 tříd (button, checkbox, dropdown, icon, input, label, radio, switch). Snímky obsahují vzhledy různých uživatelských prostředí.

4.1.1 Sestavení testového setu snímků pro nezávislý test modelů

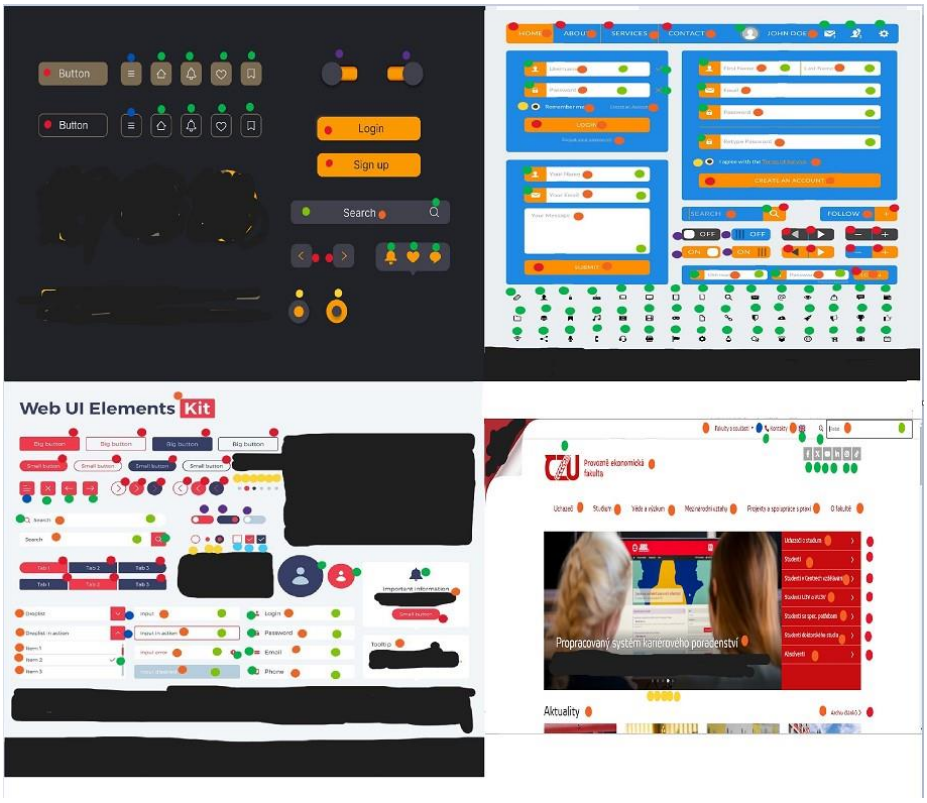
Vytvoření nezávislého testového setu pro testování modelů v praktické části bakalářské práce představuje klíčový krok v hodnocení výkonnosti a generalizace použitých modelů. Jeden ze snímků obsahuje úvodní stránku webových stránek pef.čzu.cz ostatní tři pak náhodné obrázky s nejčastěji používanými UI prvky. Proces tvorby vyžadoval pečlivý výběr jednotlivých obrázků, které by reprezentovaly různé aspekty uživatelského rozhraní, jako jsou tlačítka, labely, ikony, switche, radio, dropdown nebo inputy. Nejdříve byly rozřazeno, jaké prvky daných obrázků jsou pro testování relevantní, aby bylo reprezentativní pro reálné scénáře použití. Po získání dat byla provedena řada úprav a označení, které měly za cíl zvýšit kvalitu výsledků. To zahrnovalo odstranění chybějících nebo neplatných hodnot a normalizaci dat. Konečný testovací set byl následně rozdělen na ukázkovou část a testovací část, která byla předkládána modelům pro detekci a zhodnocení výkonnosti modelů na nezávislých datech. Obrázky s vyznačenými typy instancí tak sloužily jako důležitý nástroj pro interpretaci a vizualizaci výsledků analýzy a hodnocení modelů. V praktické části bakalářské práce byl tento nezávislý testovací set použit pro interpretaci výsledků a provádění analýzy kritérií, což umožnilo objektivní porovnání jejich metrik a identifikaci potenciálních silných i slabých stránek. Celkově tento proces vytváření nezávislého test setu poskytl vhodný základ pro testování a vyhodnocování výkonnosti modelů, která představuje různorodost, realističnost a komplexnost reálného uživatelského rozhraní, což je klíčové pro objektivní a spolehlivé testování modelů.

Obrázek 20 – Ukázka obrázků nezávislého test setu



Zdroj: Vlastní zpracování

Obrázek 21 – Ukázka obrázků označených instancí nezávislého test setu



Zdroj: Vlastní zpracování

Tato koláž viz. Obrázek č.21 ukazuje označení obsahu jednotlivých obrázků z test setu, co by mělo být detekováno, co bylo z detekcí vyřazeno jako neplnohodnotný výstup pro analýzu výkonu modelů a jaké druhy instancí obsahují. Druhy instancí, které mají být detekované jsou barevně označeny, kde třídy prvků UI je:

- Červená – button;
- Světle modrá – checkbox;
- Tmavě modrá – dropdown;
- Světle zelená – input;
- Tmavě zelená – icon;
- Oranžová – label;
- Žlutá – radio;
- Fialová – switch.

Následně bylo vyhodnocení podle počtů prvků jednotlivých instancí převedeno do tabulky celkových detekcí v jednotlivých obrázcích nezávislého testového setu:

Tabulka 1 – Celkový počet instancí k detekci

	Button	Checkbox	dropdown	icon	input	label	radio	switch
TEST_1_UI	6	0	2	14	1	1	2	2
TEST_2_UI	18	0	0	62	10	27	2	4
TEST_3_UI	21	3	3	15	10	18	9	3
TEST_czupef	8	0	1	10	1	20	5	0
CLK_PČ_DET	53	3	6	101	22	66	18	9

Zdroj: Vlastní zpracování

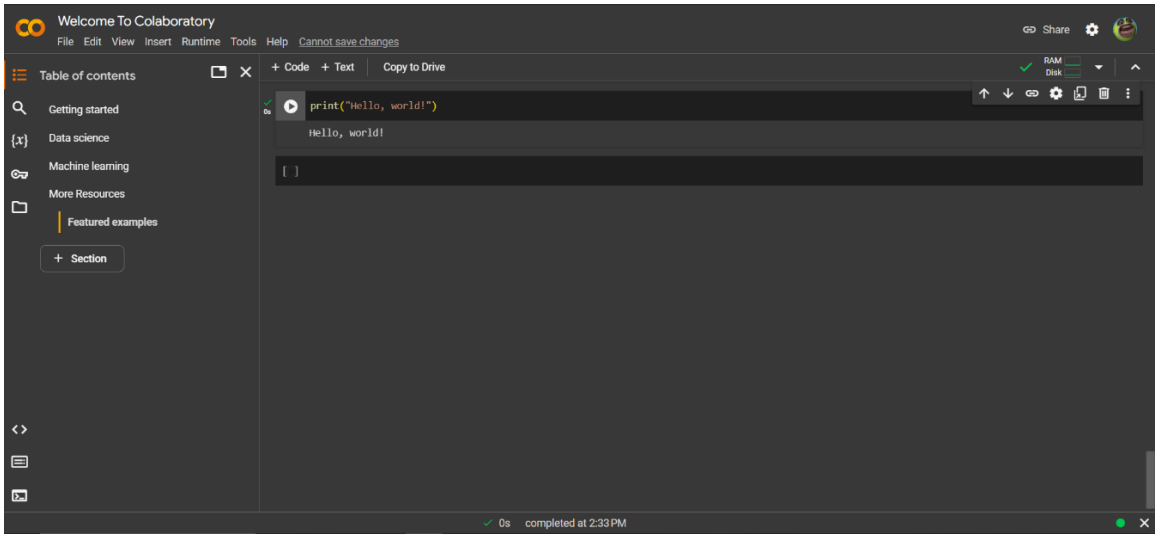
4.2 Prostředí Google Colab

Google Colab je webová služba společnosti Google, která umožňuje bezplatné používání prostředí Jupyter Notebook v cloudu. Poskytuje zdarma výpočetní zdroje, včetně možnosti využití GPU a TPU, což pomáhá k rychlejšímu trénování modelů a manipulaci s většími datovými soubory i pro uživatele s omezeným výpočetním výkonem.

Jupyter Notebooky přímo v prostředí Google Colab lze vytvářet a sdílet nebo skrze integrovaný Google Drive pro snadný import a export dat či notebooků a interaktivní vývoj a dokumentaci práce uživatelů.

Také je zde možnost importu notebooků z prostředí GitHub. Je podporován různými programovací jazyky, včetně Pythonu s knihovnami a frameworky pro strojové učení, jako je TensorFlow, PyTorch nebo OpenCV. Využitím Google Colab mohou vývojáři jednoduše a efektivně trénovat modely pro detekci UI prvků, což může přispět k vylepšení uživatelských zkušeností a vývoji sofistikovanějších uživatelských rozhraní. Z těchto důvodů bylo pro implementaci, trénování a testování jednotlivých modelů použito v praktické části bakalářské práce toto prostředí.

Obrázek 22 – Ukázka prostředí Google Colab



Zdroj: Vlastní zpracování

4.3 Stanovení metrik pro analýzu výsledků

Důležitou součástí praktické části bylo stanovit metriky pro výkon testování jednotlivých modelů a klasifikační metriky pro analýzu testovaných snímků. Pro objektivní vyhodnocení výkonu těchto modelů bylo zvoleno několik klíčových metrik, které se nejčastěji používají u detekce prvků v obraze.

Metrikou pro určení výkonnosti trénováné modelu byla zvolena metrika mAP (mean Average Precision), která je běžně používanou metrikou pro hodnocení výkonnosti algoritmů detekce objektů v oblasti Computer vision a strojového učení. Tato metrika kombinuje přesnost detekce objektů (precision) a úplnost detekce objektů (recall) a vypočítává průměrnou hodnotu přesnosti při různých úrovních odříznutí (thresholds) pro detekci. Metrika mAP0,5 představuje průměrnou přesnost detekce při prahování s IoU (Intersection over Union) 0,5. IoU je míra podobnosti mezi predikovanými a skutečnými

oblastmi objektů a prah 0,5 je často používaným standardem pro určení toho, zda je detekce považována za správnou.

Metrika mAP0,5:0,95 rozšiřuje koncept mAP tím, že bere v úvahu rozsah prahů pro IoU od 0,5 do 0,95. To znamená, že se bere v úvahu průměrná přesnost při prahování od 0,5 do 0,95. Tato metrika poskytuje komplexnější pohled na výkonnost detekčního modelu a umožňuje lépe porozumět jeho schopnosti detekovat objekty s různými úrovněmi přesnosti.

V obou případech je vyšší hodnota mAP žádoucí, protože to značí lepší výkonnost detekčního modelu.

První klasifikační metrikou pro analýzu výsledných obrázků je Accuracy, (procentuální úspěšnost detekcí) což je poměr správně detekovaných instancí k celkovému počtu instancí. Tato metrika poskytuje obecný přehled o celkovém výkonu modelu. Vzorec pro výpočet Accuracy je:

$$Accuracy = \frac{Počet\ detekovaných\ instancí}{Celkový\ počet\ instancí\ k\ detekování}$$

Druhou klasifikační metrikou je Avg. precision (průměrná přesnost), která detailněji měří výkon modelu na základě toho, jaká je míra správně identifikovaných pozitivních instancí mezi všemi pozitivními předpověďmi pro určení přesnosti klasifikace přes všechny třídy objektů v datasetu. Precision je poměr správně identifikovaných pozitivních instancí k celkovému počtu detekcí. Avg. precision tedy udává, jak dobře model detekoval objekty všech tříd.

Vzorec pro výpočet metriky precision:

$$Precision = \frac{TP}{TP + FP}$$

- TP – počet pozitivních detekovaných instancí;
- FP – počet falešných detekovaných instancí.

Třetí klasifikační metrikou a také metrikou pro detailnější měření výkonu modelu je Avg. recall (průměrná úplnost). Určuje, jaká je míra správně identifikovaných pozitivních instancí mezi všemi skutečnými pozitivními instancemi přes všechny třídy objektů v datasetu. Avg. recall udává, jak dobře model identifikoval všechny skutečné pozitivní instance objektů přes všechny třídy. Vzorec pro výpočet metriky recall:

$$Recall = \frac{TP}{TP + FN}$$

- TP – počet pozitivních detekovaných instancí;
- FN – počet nedetekovaných instancí.

Čtvrtou klasifikační metrikou je Avg. F1-score, která je harmonickým průměrem Avg. precision a Avg. recall a poskytuje vyvážený ukazatel výkonu modelu.

Společně tyto klasifikační metriky poskytují komplexní a objektivní analýzu výkonu modelů z oblasti Computer Vision pro detekci UI prvků. Jejich kombinace umožňuje detailní posouzení schopností modelů a porovnání jejich účinnosti při identifikaci a lokalizaci různých typů UI prvků. Vzorec pro výpočet klasifikační metriky F1-score:

$$F1 - score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

4.4 Sestavení modelů

Modely byly vybrány na základě jejich vhodné implementace v prostředí Google Colab s využitím knihoven PyTorch a TensorFlow. Dalším kritériem výběru byla optimalizace vzhledem k omezeným výpočetním zdrojům, konkrétně omezení CPU (12,7 GB) a GPU (15 GB) dostupných pro trénování modelů. Proces sestavení modelů zahrnoval přípravu konkrétních modelů z repositářů, spolu s veškerými potřebnými součástmi pro konfiguraci a trénování s vlastními daty, která byla rovněž načtena do úložiště Google Colab. Hyperparametry trénování byly pro všechny modely nastaveny na stejné hodnoty, aby bylo dosaženo konzistence v procesu trénování. Jde konkrétně o tyto hodnoty hyperparametrů:

- epochs 5;
- batch 4;
- třídy 8 + 1 pro background (pozadí).

SSD model svojí nepříliš vhodnou architekturou, která byla primárně navržena pro detekci objektů ve scénách s různými objekty a jeho dalšího potřebného ladění pro detekci UI prvků v uživatelském rozhraní, zejména pokud je v rozhraní pouze malý počet prvků nebo jsou tyto prvky relativně jednoduché, není vhodnou metodou pro mojí praktickou část bakalářské práce. Tato skutečnost byla zhodnocena při trénování modelu RetinaNet, který je svojí architekturou velice podobný a je také one stage detektorem jako SSD model. V těchto

případech mohou být two stage detector modely založené na Faster R-CNN nebo hybridní variantě DETR více vhodné, vzhledem k jejich menší náročnosti a rychlosti trénování. Z těchto důvodů a na základě literatury v teoretické části bylo rozhodnuto SSD model do praktické části nezařadit. Pro reprezentaci one stage detectorů, byly raději zvoleny modely YOLOv8 a RetinaNet. Řádek kódu pro stažení datasetu do prostředí Google colab:

```
!curl -L
"https://universe.roboflow.com/ds/sFmEHAPxDv?key=eCONZV1Ulp" >
roboflow.zip; unzip roboflow.zip; -d custom_data; rm roboflow.zip
```

Script 1 – Stažení datasetu pro trénování modelů

4.4.1 Model DETR

Příprava struktury adresáře byla poskytnuta z:

```
!git clone https://github.com/sovit-123/vision_transformers.git
```

Script 2 – Příprava adresáře pro model DETR

Po spuštění souboru requirements.txt z adresáře pro doinstalovaní potřebných knihoven jako např. (albumentations, Torchinfo, tqdm, deep-sort-realtime, Pycocotools, torchmetrics) a poté byl vytvořen konfigurační soubor custom_data.yaml, který obsahuje cesty k adresářům pro trénování s obsahem obrázků a jejich anotací. Dále definuje názvy tříd a počet tříd v datasetu. Volba SAVE_VALID_PREDICTION_IMAGES určuje, zda se mají ukládat obrázky s predikcemi z validační části. Tento soubor custom_data.yaml byl vytvořen k nastavení parametrů pro trénink a validaci modelu detekce UI prvků:

```
%%writefile /content/vision_transformers/data/custom_data.yaml
TRAIN_DIR_IMAGES: '/content/custom_data/train'
TRAIN_DIR_LABELS: '/content/custom_data/train'
VALID_DIR_IMAGES: '/content/custom_data/valid'
VALID_DIR_LABELS: '/content/custom_data/valid'

CLASSES:   ['__background__',   'button',   'checkbox',   'dropdown',
'icon', 'input', 'label', 'radio', 'switch']
NC: 10

SAVE_VALID_PREDICTION_IMAGES: True
```

Script č. 3 – Vytvoření config.py pro nastavení cest k datasetu pro trénování modelu DETR

Trénink detektoru a detekce na vybraném obrázku je spuštěn pomocí příkazového řádku v Pythonu. Specifikuje se počet epoch (5), velikost dávky (4), konfigurační soubor custom_data.yaml obsahující informace o datasetu a modelu, který lze měnit dle potřeby na detr_resnet50, detr_resnet50_dc5, detr_resnet101 nebo detr_resnet101_dc5. Parametr

--name určuje název tréninkového experimentu, který bude použit pro ukládání vážených modelů a výsledků tréninku:

```
!python tools/train_detector.py --epochs 5 --batch 4 –data
data/custom_data.yaml --model detr_resnet50_dc5 --name
detr_resnet50_dc5
```

Script 4 – Spuštění trénování modelu DETR

--input určuje cestu k obrázku, na kterém se má provést detekce.

```
!python tools/inference_image_detect.py --weights
runs/training/detr_resnet50_dc5/best_model.pth –input
"/content/TEST_images/TEST_1_UI.jpg"
```

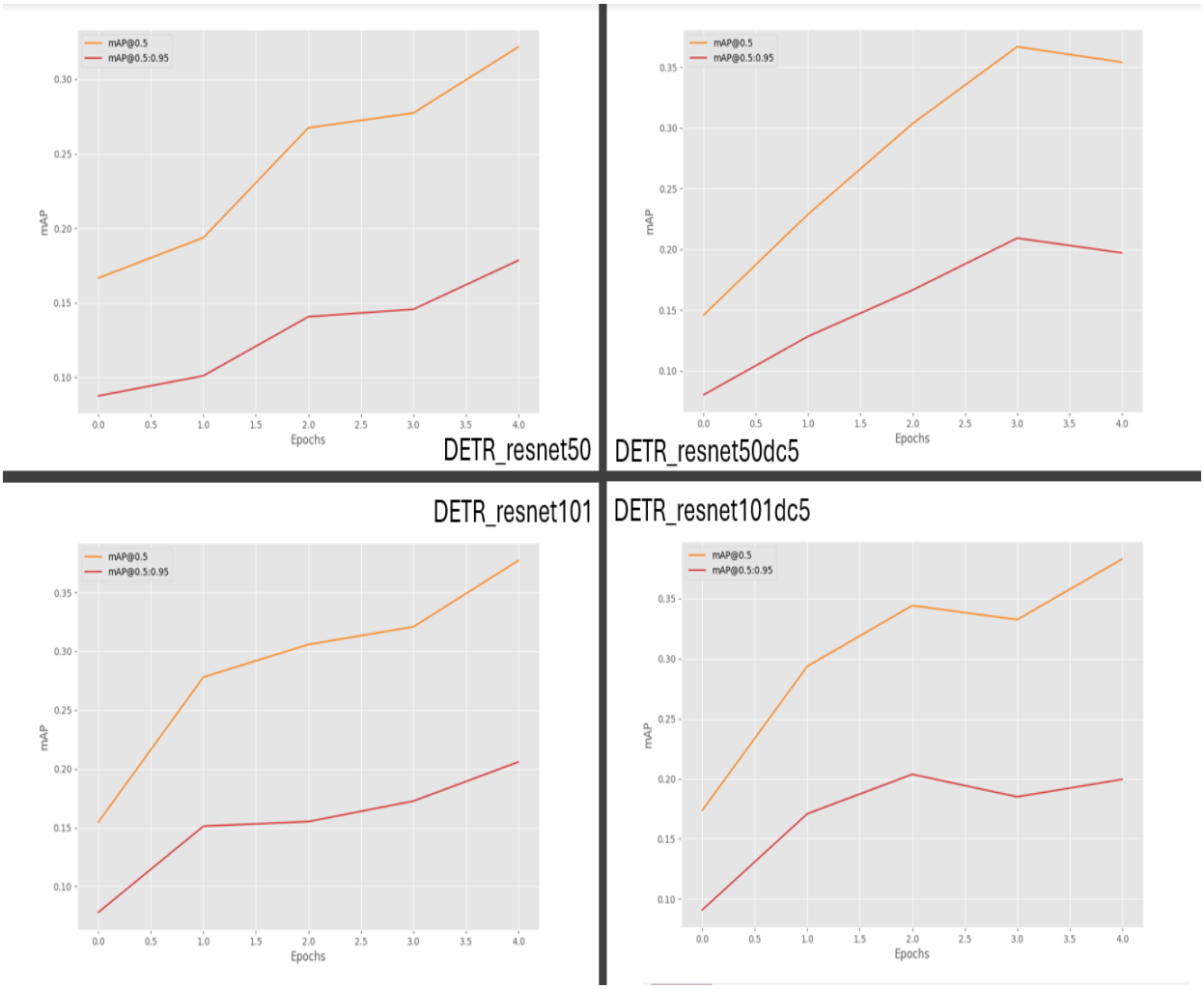
Script 5 – Detekce vybraného obrázku z test datasetu modelem DETR

Testování a detekovaní bylo provedeno na různých verzí modelu DETR. To může poskytnout poznatky o jejich schopnosti detekce a klasifikace. Porovnáním výkonu těchto verzí na stejném datasetu lze získat ucelenější povědomí o jejich efektivitě a možnostech nasazení.

Grafy výsledků trénování modelů DETR

Níže na (Graf 1) jsou grafy výsledků trénování a výsledky detekcí jednotlivých modelů, kde jsou zobrazeny metriky mAP0,5 a mAP0,5:0,95. Oranžová čára představuje metriku mAP0,5 a červená mAP0,5:0,95.

Graf 1 – Výsledky trénování modelů DETR



Zdroj: Vlastní zpracování

Z grafů je patrné, že verze resnet50, resnet101 a resnet101dc5 modelu DETR mají poměrně dobrou míru trénování na datasetu s UI prvky. U modelu resnet50dc5 již docházelo k „overfittingu“ tzv. přeučením počtem snímků obsažených v datasetu pro trénování.

4.4.2 Model Faster R-CNN

Struktura adresáře byla provedena z:

```
!git clone https://github.com/sovit-123/fastercnn-pytorch-training-pipeline.git
```

Script 6 – Příprava adresáře pro model Faster R-CNN

Podobně jako u metody DETR se spustí instalace přes textový soubor requirements.txt s potřebnými knihovnami např. (albumentations, ipython, jupyter, matplotlib, opencv-python, Pillow, PyYAML, scikit-image, scikit-learn, scipy, torch,

torchvision, numpy, protobuf, pandas, tqdm) a upraví se konfigurační soubor custom_data.yaml, ale s rozdílnou cestou k datasetu.

```
%%writefile data_configs/custom_data.yaml
TRAIN_DIR_IMAGES:                '/content/fastercnn-pytorch-training-
pipeline/custom_data/train'
TRAIN_DIR_LABELS:                '/content/fastercnn-pytorch-training-
pipeline/custom_data/train'
VALID_DIR_IMAGES:                '/content/fastercnn-pytorch-training-
pipeline/custom_data/valid'
VALID_DIR_LABELS:                '/content/fastercnn-pytorch-training-
pipeline/custom_data/valid'

CLASSES:   ['__background__',   'button',   'checkbox',   'dropdown',
'icon', 'input', 'label', 'radio', 'switch']

NC: 9

SAVE_VALID_PREDICTION_IMAGES: True
```

Script č. 7 – Úprava config.py pro nastavení cest k datasetu pro trénování modelu Faster R-CNN

Dále jsou části pro trénování a detekce obdobné jako u DETRu s rozdílem různých cest k souborům. Script pro trénovaní:

```
!python train.py --data /content/fastercnn-pytorch-training-
pipeline/data_configs/custom_data.yaml --epochs 5 --model
fasterrcnn_vgg16 --name custom_training --batch 4
```

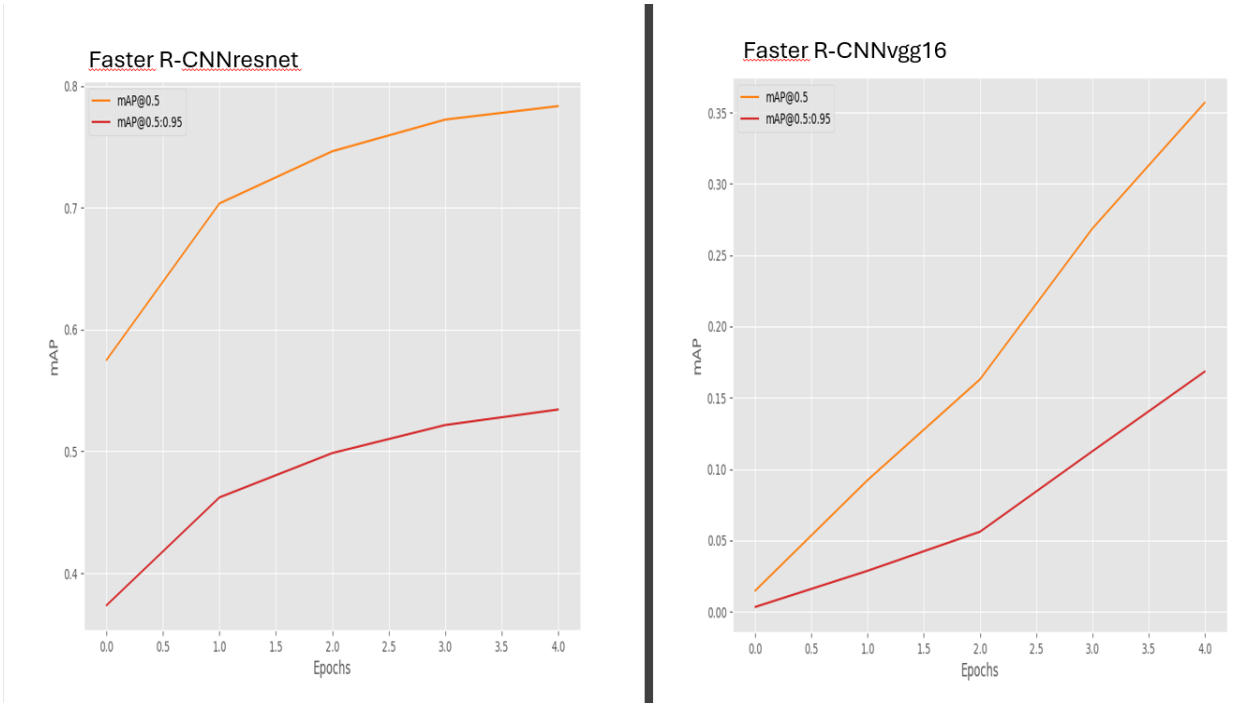
Script č. 8 – Spuštění trénování modelu Faster R-CNN

Script pro detekci:

```
!python inference.py --input data/inference_data/TEST_3_UI.jpg --
weights outputs/training/custom_training/best_model.pth
```

Script č. 9 – Detekce vybraného obrázku z test datasetu modelem Faster R-CNN

Graf 2 – Výsledky trénování modelů Faster R-CNN



Zdroj: Vlastní zpracování

Z porovnání grafů je patrné, že verze využívající neuronovou síť resnet má velice vysokou míru trénování oproti vgg16. V celkovém porovnání, ale model vgg16 vyšel obstojně. Verze Faster R-CNN vyšla nejlépe co se týče výkonu trénování. Očekává se tedy, že v testu nezávislým datasetem vyjde nejlépe.

4.4.3 Model RetinaNet

Adresář byl nahrán z Google drive pomocí instalačního kódu:

```
Import shutil
from google.colab import drive
drive.mount('/content/drive')
zip_path =
'/content/drive/MyDrive/20230515_Train_PyTorch_RetinaNet_on_Cu
stom_Dataset.zip'
extract_path = '/content/'
shutil.unpack_archive(zip_path, extract_path)
```

Script 10 – Nahrání adresáře s modelem RetinaNet z Google drive

V adresáři byl upraven script config.py a jeho cesty k datasetu. Byly nastaveny také hodnoty hyperparametrů batch_size a num_epochs.

Úprava config.py:

```
import torch

BATCH_SIZE = 4
NUM_EPOCHS = 5

DEVICE = torch.device('cuda') if torch.cuda.is_available()
else torch.device('gpu')
TRAIN_DIR = 'data/custom_data/train'
VALID_DIR = 'data/custom_data/valid'

CLASSES = [
    '__background__',    'button',    'checkbox',    'dropdown',
'icon', 'input', 'label', 'radio', 'switch'
]

NUM_CLASSES = len(CLASSES)

VISUALIZE_TRANSFORMED_IMAGES = True

OUT_DIR = 'outputs'
```

Script 11 – Úprava config.py pro nastavení cest k datasetu pro trénování modelu RetinaNet

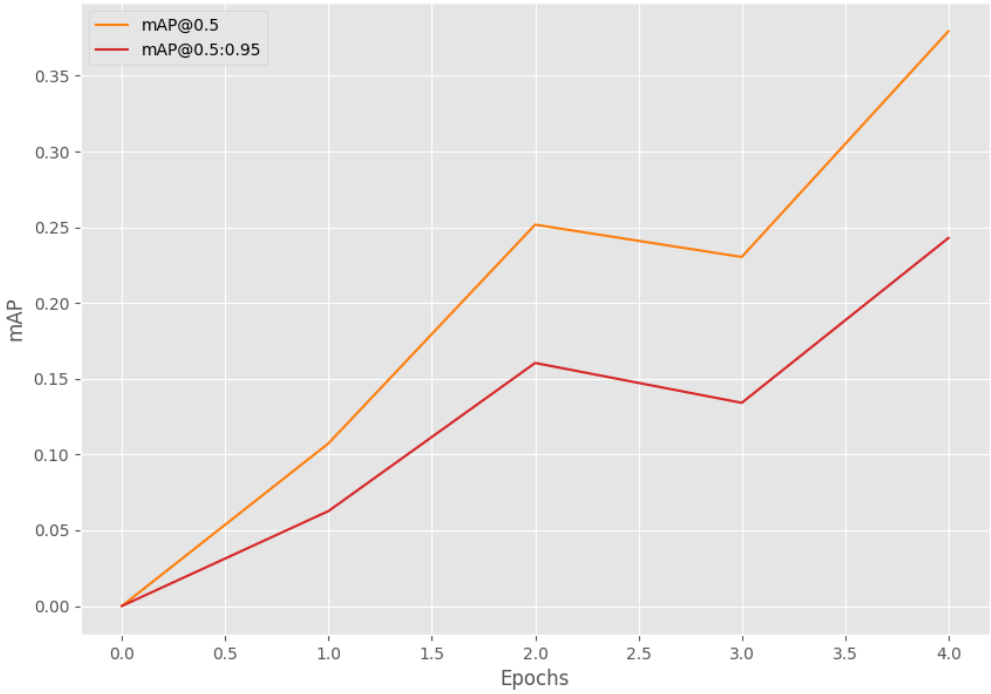
Poté je puštěn script pro trénování modelu RetinaNet a po dokončení trénování modelu script pro detekci dat:

```
!python/content/20230515_Train_PyTorch_RetinaNet_on_Custom_Dataset/tr
ain.py

!python/content/20230515_Train_PyTorch_RetinaNet_on_Custom_Dataset/in
ference.py --input "/content/inference_data"
```

Script 12 – Části kodu pro trénování a spuštění detekce modelu RetinaNet

Graf 3 – výsledek trénování modelu RetinaNet



Zdroj: Vlastní zpracování

Na základě grafu RetinaNet má výsledky podobné jako předešlé modely DETR a Faster R-CNN vgg16, jde ale o čistě trénování a validaci v rámci datasetu použitý pro trénování.

4.4.4 Model YOLOv8

Pro trénování modelem YOLOv8 byly implementovány v praktické částí dvě verze, YOLOv8s a YOLOv8n. YOLOv8n a YOLOv8s se liší ve své architektuře a výkonu. "n" ve jméně značí "nový" a "s" značí "rychlý". YOLOv8n je navržen pro dosažení lepší přesnosti detekce objektů, zatímco YOLOv8s je optimalizován pro rychlost. Pro implementaci YOLOv8 byl nahrán pomocí příkazu z knihovny Ultralytics:

```
!pip install ultralytics==8.0.196
from IPython import display
display.clear_output()
import ultralytics
ultralytics.checks()
from ultralytics import YOLO
from IPython.display import display, Image
```

Script 13 – Příprava adresáře Ultralytics

Využíváme dataset, který byl vytvořen v prostředí platformy Roboflow. Pro implementaci YOLOv8 přímo zde využíváme specifický postup, který umožňuje jeho načtení do Google Colabu a není potřeba dalších úprav:

```
!pip install roboflow -quiet
from roboflow import Roboflow
rf = Roboflow(api_key="**************")
project = rf.workspace("cha-ebebh").project("cingoz_")
version = project.version(2)
dataset = version.download("yolov8")
```

Script 14 – Stažení datasetu z Roboflow ve formátu pro YOLOv8 model

API klíč je v tomto příkladě zamazaný z důvodu součástí autentizačního procesu, který umožňuje přístup k účtu na Roboflow a provádění operací jako stahování datasetů nebo modelů. Je důležité udržovat API klíč jako soukromý, protože s ním může kdokoli, kdo ho zná, provádět akce v rámci účtu uživatele.

Trénování YOLOv8 se spustí přes následující skript, kde je možné volit model mezi yolov8n a yolov8s, dle potřeby:

```
!yolo task=detect mode=train model=yolov8n.pt
data={dataset.location}/data.yaml epochs=5 batch=4 plots=True
```

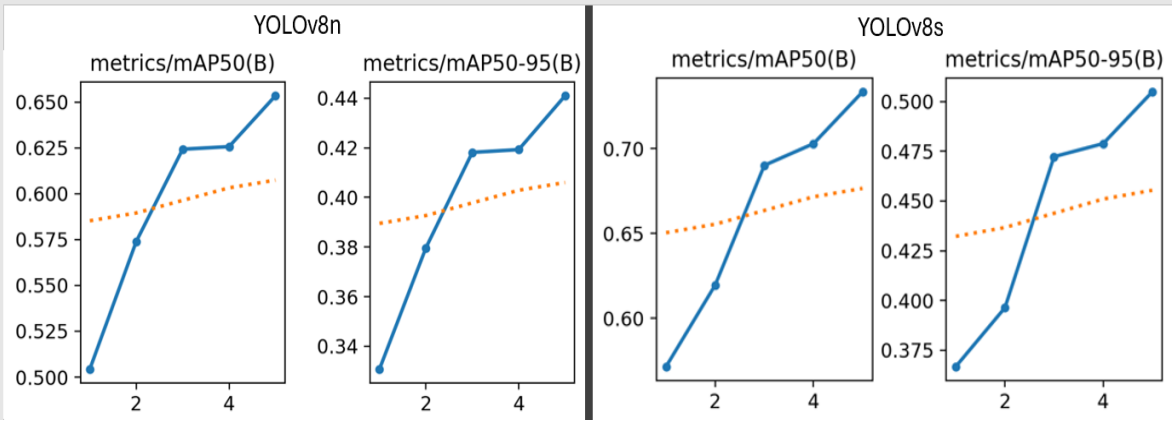
Script 15 – Spuštění trénování modelem YOLOv8

Detekování objektů v testovacích snímcích natrénovaným YOLOv8 bylo provedeno kódem:

```
!yolo task=detect mode=predict
model={HOME}/runs/detect/train2/weights/best.pt
source=/content/runs/detect/predict/TEST_czupef.png save=True
```

Script č. 16 - Detekce vybraného obrázku z test datasetu modelem YOLOv8

Graf 4 – Grafy výsledků trénování modelů Yolov8



Zdroj: Vlastná zpracování

Míra trénování modelů YOLOv8n a YOLOv8s je dle grafu vyšší než u modelů DETR, Faster R-CNN vgg16 a Retinanet. Předpokládá se tedy, že bude v testu nezávislým datasetem, dosahovat vyšších měr výkonnosti než tyto zmíněné modely.

5 Zhodnocení výsledků

V této části budou prezentovány vizuální výstupy detekčních modelů v podobě obrazových materiálů, tabulkových výsledků pro komparativní analýzu a závěry získané z obsahu tabulek s výsledky.

5.1 Vyhodnocení výsledků na nezávislém testu

Celkové výsledky detekcí jednotlivých modelů jsou obsaženy v tabulkách v příloze A. Data z tabulek byla získána v souladu s metodikou, která zahrnovala kontrolu a následné spočítání celkového počtu detekovatelných instancí a počtu detekovaných instancí na snímcích z nezávislého datasetu, které byly detekovány modelem. Tyto instance byly poté klasifikovány do tří kategorií na základě výsledků detekce: pozitivně detekované (TP), falešně detekované (FP) a nedetekované (FN). Pro ověření přesnosti zápisu hodnot do tabulky během analýzy snímků z nezávislého testovacího souboru, tzv. "nevynechané" nebo "neopomenuté" instance, byl vytvořen speciální řádek v tabulce označený jako "CLK_PČ_DET", jehož hodnota musela odpovídat hodnotě v řádku tabulky označeném jako "Počty prvků v UI", aby byla zajištěna integrita dat. Po zaznamenání všech instancí detekovaných daným modelem na všech čtyřech obrazech byly vypočteny klasifikační metriky pro každý model Accuracy, Precision, Recall a F1 skóre pro jednotlivé třídy UI prvků. Tyto klasifikační metriky byly dále použity k vytvoření a jejich zapsání do tabulky průměrného Accuracy, průměrného Precision, průměrného Recall a průměrného F1 skóre pro porovnání výkonu jednotlivých modelů mezi sebou. Zde je ukázka Tabulka 2, která byla výslednou tabulkou detekcí modelu YOLOv8s z Přílohy A:

Tabulka 2 – Ukázka výsledné tabulky jednoho z modelů

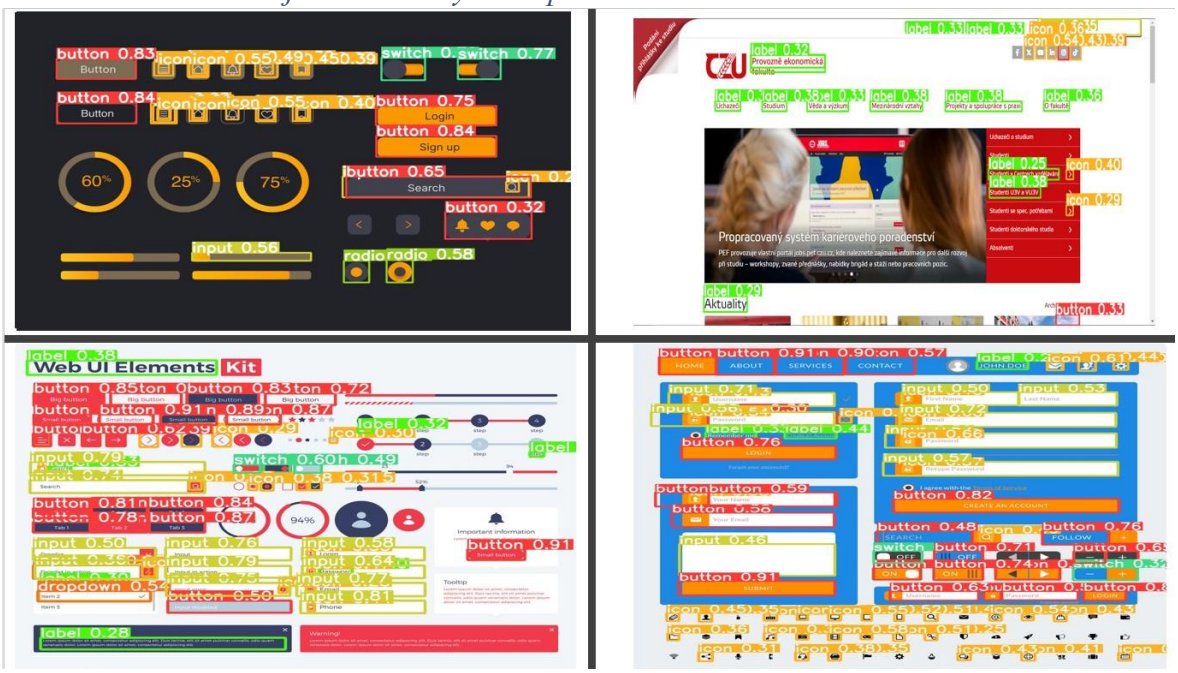
	57,55%	Button	Checkbox	dropdown	icon	input	label	radio	switch
YOLOv8s	TP	31	0	1	52	18	18	2	6
	FP	10	2	4	6	4	0	3	3
	FN	12	1	1	43	0	48	13	0
	CLK_PČ_DET	53	3	6	101	22	66	18	9
	Počty prvků v UI	53	3	6	101	22	66	18	9
	Precision	0,7561	0	0,2	0,8965517	0,818	1	0,4	0,6667
	Recall	0,72093	0	0,5	0,5473684	1	0,2727	0,1333	1
	F1-score	0,7381	0	0,28571	0,6797386	0,9	0,4286	0,2	0,8
	F1-score (Avg.)	0,50401	Precision (Avg.)	0,59219	Recall (Avg.)	0,522			

Zdroj: Vlastní zpracování

5.2 Prezentace vizuálních výsledků

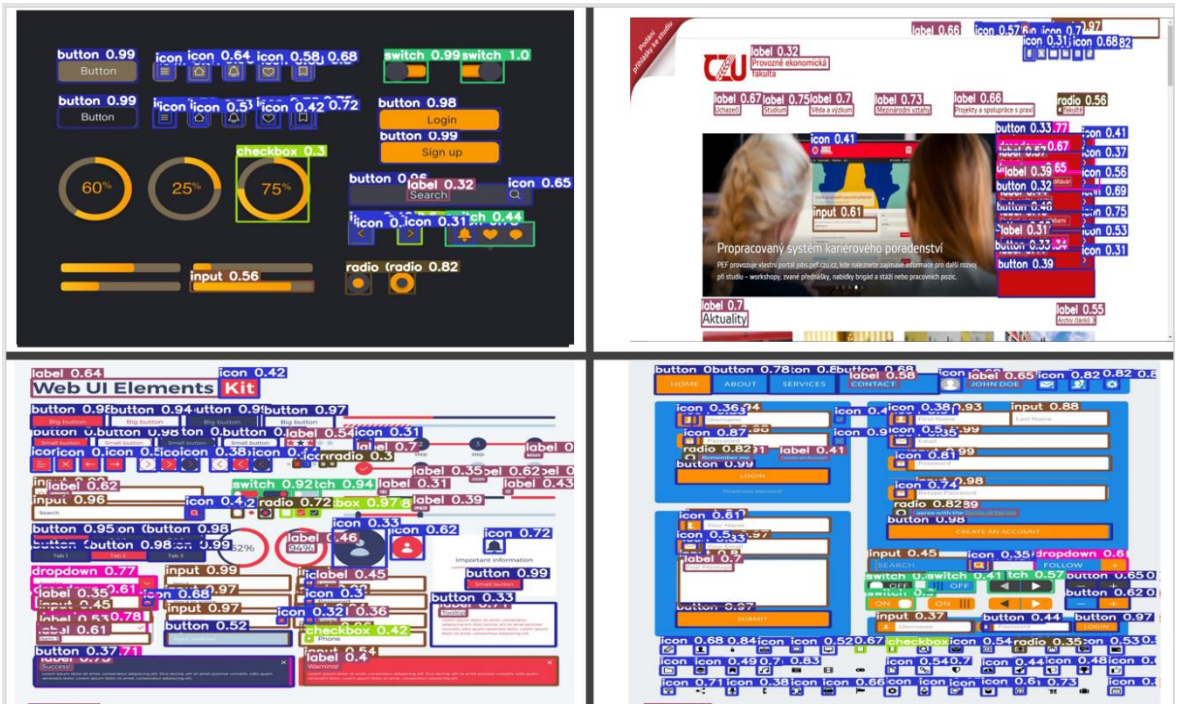
V následující sekci je prezentována vizuální analýza detekce uživatelských rozhraní (UI) prostřednictvím obrázků z nezávislého testovacího setu. Tyto obrázky ilustrují detekci UI prvků provedenou vybranými detekčními modely, což umožňuje porovnání jejich výkonu v této konkrétní oblasti:

Obrázek 23 – Ukázka již detekovaných UI prvků modelem YOLOv8s



Zdroj: Vlastní zpracování

Obrázek 24 – Ukázka již detekovaných UI prvků modelem Faster R-CNN resnet

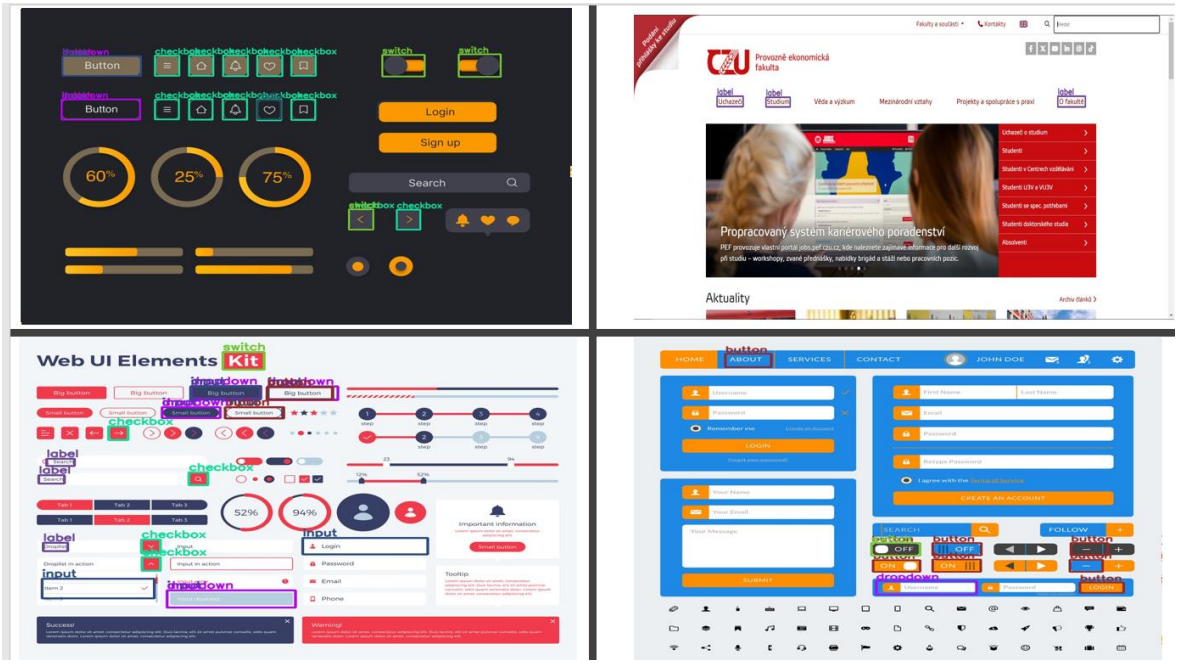


Zdroj: Vlastní zpracování

Obrázek 25 – Ukázka již detekovaných UI prvků modelem DETRresnet101

Zdroj: Vlastní zpracování

Obrázek 26 – Ukázka již detekovaných UI prvků modelem RetinaNet



Zdroj: Vlastní zpracování

5.3 Srovnání celkových výsledků a jejich analýza

Z Tabulka 3 a podle Grafu 5 výsledků porovnání klasifikačních metrik jednotlivých modelů vyplývá, že Faster R-CNN dosáhl nejlepších výsledků ve všech sledovaných metrikách, podobně jako v práci průzkumu Object Detection for Graphical User Interface:

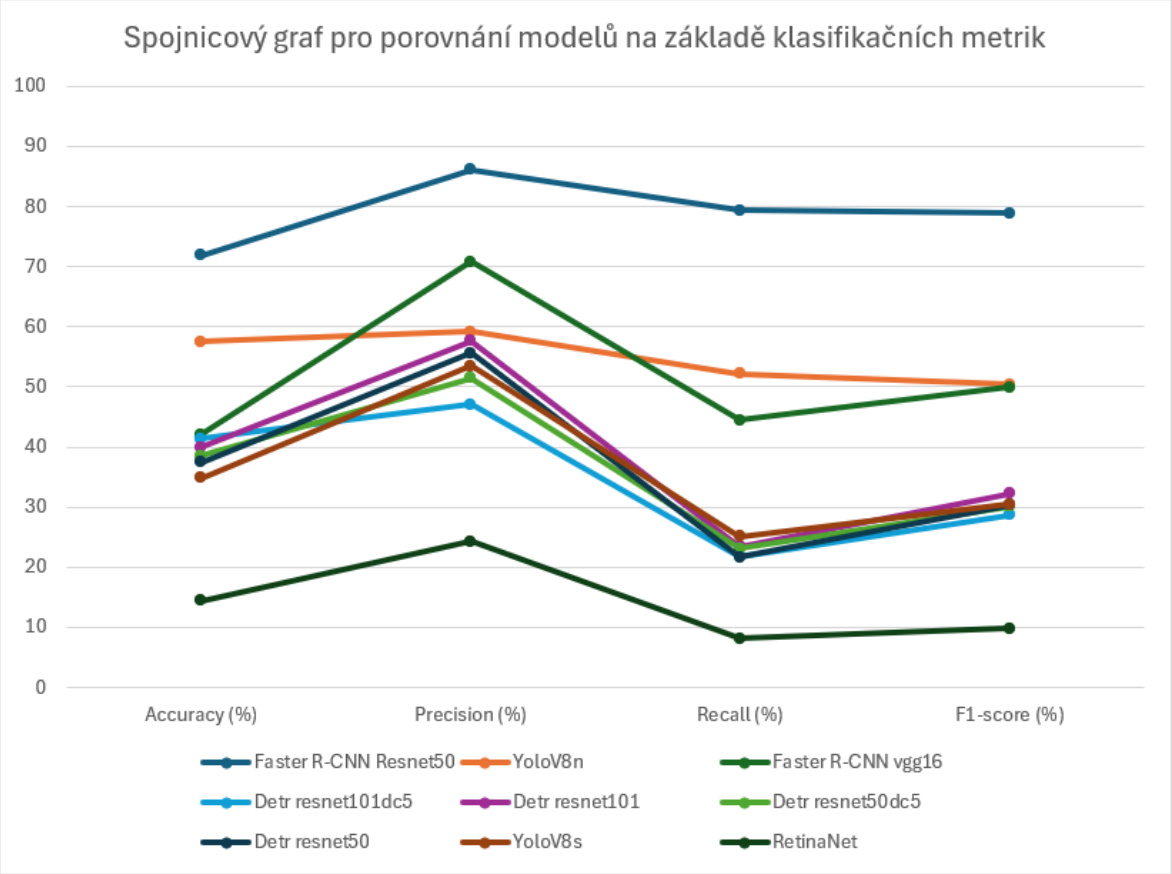
Old Fashioned or Deep Learning or a Combination? autorů (Chen, Xie, Xing, Chen, Xu, Zhu, 2020), kde se při porovnání umístil na prvním místě mezi tradičními metodami pro detekci. Naopak RetinaNet se v experimentu této práce umístil na posledním místě v celkovém srovnání. Metrika Accuracy je v kontextu této práce klíčová, neboť poskytuje informaci o procentu správně detekovaných prvků z celkového počtu detekovaných. V případě této studie se jednalo o 278 prvků. Pozoruhodným zjištěním byl rozdíl v počtu detekovaných prvků mezi modely YOLOv8s (160) a YOLOv8n (97), přičemž jejich rozdílem je v architektuře sítí, kdy YOLOv8s využívá sítě pro rychlejší detekci s menší přesností. To se také ukázalo v Tabulce 3, kdy YOLOv8n dosahovalo přesnější detekce než YOLOv8s s celkových detekcí.

Tabulka 3 – Výsledek porovnání klasifikačních metrik modelů

Srovnání na základě Accuracy				
	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
Faster R-CNN Resnet50	71,94	86,11	79,36	78,90
YoloV8n	57,55	59,22	52,18	50,40
Faster R-CNN vgg16	42,09	70,87	44,48	49,90
Detr resnet101dc5	41,36	47,10	21,84	28,69
Detr resnet101	39,93	57,69	23,43	32,27
Detr resnet50dc5	38,48	51,47	23,06	30,14
Detr resnet50	37,41	55,59	21,68	30,29
YoloV8s	34,89	53,45	25,16	30,45
RetinaNet	14,39	24,26	8,14	9,87

Zdroj: Vlastní zpracování

Graf 5 – Zobrazení výsledků z Tabulky 3 do spojnicového grafu



Zdroj: Vlastní zpracování

V Grafu 5 lze vidět také rozdíly v klasifikační metrice Accuracy mezi modely DETR resnet101dc5, DETR resnet101, DETR resnet50dc5 a DETR resnet50 jsou procentuální rozdíly malé, jelikož se jedná o rozdíly mezi 1,07 % (3 prvky) až 3,95 % (11 prvků) počtu detekovaných prvků. Modely YOLOv8n a Faster R-CNN vgg16 mají mezi sebou procentuální rozdíl 15,46 % (43 prvků) počtu detekovaných prvků, ovšem v procentuální úspěšnosti klasifikační metriky Precission má vyšší procentuální úspěšnost Faster R-CNN vgg16. Dosažených výsledků modelem Faster R-CNN vgg16 byly následující:

- Precision ≐ 70,87 %;
- Recall ≐ 44,48 %;
- F1-Score ≐ 49,90 %.

Dosažených výsledků modelem YOLOv8n byly následující:

- Precision ≐ 59,22 %;
- Recall ≐ 25,16 %;
- F1-Score ≐ 50,40 %.

Tento rozdíl ukázal, že Faster R-CNN vgg16 přesněji identifikoval relevantní objekty ve srovnání s modelem YOLOv8n. Nicméně, je třeba vzít v úvahu, že model Faster R-CNN vgg16 dosáhnul nižších hodnot klasifikačních metrik Accuracy, Recall a F1 skóre. To naznačuje, že přesnost detekce objektů není dostatečně vyvážená mezi falešně pozitivními a falešně negativními detekcemi, což může vést ke ztrátě relevantních objektů nebo k zahrnutí nadbytečných objektů do výsledku.

Dalším podnětným výsledkem z Příloh A byla neúspěšná detekce prvků checkbox a dropdown většině modelů. Všechny prvky třídy checkbox byly detekovány modelem Faster R-CNNresnet a Faster R-CNNvgg16, prvek dropdown byl v detekcích často zaměňován za checkbox, icon nebo button. Případ zaměňování třídy prvku jinou třídou prvku byla také pozorována u buttonů, které byly vyhodnoceny místo buttonu za inputy a buttonu za iconu kdy modely mezi sebou tyto tři třídy prvku často zaměňovaly. Návrhem na řešení tohoto problému je bližší optimalizace, jaký prvek je která třída a výběr obrázků datasetu pro trénování modelů přímo s těmito příklady v rozdílech jejich vizuální sféře.

5.4 Doporučení

Do budoucího výzkumu by bylo vhodné zařadit více modely typu two stage detector nebo hybridní modely typu jako je model DETR či kombinaci modelů s architekturou two stage a one stage detectoru. Modely s architekturou one stage detector nejsou pro detekci UI prvků vhodné z důvodů nepřesné detekce na malých či jemných typech jako jsou tlačítka, radia a ikony. Zvýšení výpočetních zdrojů GPU zakoupením vyššího plánu v prostředí Google colab nebo zakoupením výkonnějšího vlastního hardwaru. Zavedením obrázků do datasetu pro trénink UI prvků vytvořených v grafických editorech nebo kreslících aplikacích či ručně malovanými jako tomu je u projektu UIED autora MulongXie. Vytvoření vlastního datasetu, kdy je možnost ovlivnit více výběr prostředí s UI prvky a bližší specifikace konkrétních prvků pro eliminaci záměn už v procesu trénování modelů, případné přidání většího množství tříd UI prvků, kterými ImageView, Slider, Progress bar, Spinner a Tab. Pro úplnost porovnání jednotlivých modelů lze přidat také jiné faktory trénování, jako zátěž výpočetních zdrojů GPU a CPU nebo také faktor času, kdy se bude měřit časovou náročnost trénování modelů.

Závěr

Bakalářská práce se zabývala výběrem a porovnáním vhodných metod z oblasti Computer vision pro klasifikaci a identifikaci prvků uživatelského rozhraní. Práce se zaměřila na teoretická východiska v oblasti umělé inteligence, strojového učení a hlubokého učení se základní definicí hlavních pojmů těchto oblastí, aby poskytla nezbytný kontext pro pochopení problematiky detekce UI prvků. Dále se podrobněji zabývala používáním neuronových sítí a strukturou architektur konvolučních neuronových sítí, jako je R-CNN, Fast R-CNN, Faster R-CNN, SSD, YOLO a DETR, které jsou využívány pro detekci objektů v obrazech. Dále byla provedena analýza trhu obdobných řešení pro detekci UI prvků jiných autorů.

Vlastní práce byla zaměřena na výběr vhodného datasetu s nejčastěji využívanými prvky UI a implementaci tohoto vytvořeného datasetu pro detekci UI prvků. Dalším krokem bylo nalezení již existujících modelů pro jejich správné sestavení s funkčním využitím vybraného datasetu pro jejich trénování a analýzu výsledné detekce, kdy bylo vytvořeno v podobě grafů posouzení výkonu trénování. Pro práci bylo využito prostředí Google Colab z důvodu nedostatku výpočetních zdrojů GPU a CPU, kde zde byly implementovány čtyři hlavní modely pro detekci UI prvků: DETR, Faster R-CNN, RetinaNet a YOLOv8. Každý model byl následně trénován a testován na sestaveném nezávislém testovém setu obsahujícím snímky a reálným prostředím úvodní stránky pef.czu.cz s UI prvky.

V kapitole Zhodnocení výsledků byly následně porovnány výsledky na základě již detekovaných prvků ve snímcích z nezávislého testového setu. Jednotlivé výsledky modelů DETR, Faster R-CNN, RetinaNet, YOLOv8 byly poté porovnány mezi sebou a byla provedena analýza jejich úspěšnosti. Závěrem byl doporučen postup pro potenciální vývoj v této oblasti a byla zdůrazněna důležitost správného výběru modelu a výběru datasetu pro detekci UI prvků s ohledem na konkrétní požadavky a omezení.

Seznam použitých zdrojů

CHOLLET, Francois, 2019. Deep learning v jazyku Python: knihovny Keras, Tensorflow. Grada Publishing, as.

VONDRÁK, Ivo, 2019. Umělá inteligence a neuronové sítě. 3. vyd. Ostrava: VŠB - Technická univerzita Ostrava. ISBN 978-80-248-1981-5.

SIKKA, Bharat, 2021. Elements of Deep Learning for Computer Vision: Explore Deep Neural Network Architectures, PyTorch, Object Detection Algorithms, and Computer Vision Applications for Python Coders (English Edition). BPB Publications.

SOLEM, Jan Erik, 2012. Programming Computer Vision with Python: Tools and algorithms for analyzing images. " O'Reilly Media, Inc.".

ALPAYDIN, Ethem. 2014. Introduction to Machine Learning. 1. Cambridge: MIT Press, 639 s. Adaptive Computation and Machine Learning Ser., 3. ISBN 9780262325745. Dostupné také z: https://ebookcentral-proquest-com.infozdroje.czu.cz/lib/czup/detail.action?docID=3339851&query=Machine+learning#

WALTLOVÁ, Andrea, 2019. Umělá inteligence a učení. EdTech KISK. Brno. Dostupné z: https://medium.com/edtech-kisk/u%C4%8D%C3%ADc%C3%AD-se-spole%C4%8Dnost-informace-901718db7acb

KAĎOUSKOVÁ, Barbora, 2022. UMĚLÁ INTELIGENCE (AI): HISTORIE A TRENDY PRO ROK Rascasone. Praha. Dostupné z: https://www.rascasone.com/cs/blog/umela-inteligence-ai-trendy

DAMASSINO, Nicola, 2020. The Questioning Turing Test. Minds & Machines 30, 563–587. Dostupné z: https://doi.org/10.1007/s11023-020-09551-6

MIKELSTEN, Daniel, Vasil TEIGENS a Peter SKALFIST. Umělá inteligence: Čtvrtá průmyslová revoluce. Cambridge Standford Books. ISBN 9781005168490.

ELFOULY, Sharif, 2019. R-CNN (Object Detection) [online]. Medium [cit. 2022-08-15]. Dostupné z: https://medium.com/@selfouly/r-cnn-3a9beddfd55a

JIRKOVSKÝ, Jaroslav, 2017. Metody Deep Learning k segmentaci obrazu. Děčín. Automa – ČAT. ISSN 1210-9592.

JIRKOVSKY, Jaroslav, 2017. Deep Learning v prostředí Matlab. Děčín. Automa – ČAT. ISSN 1210-9592.

BADRINARAYANAN, Vijay, Alex KENDALL a Roberto CIPOLLA, 2015. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. ArXiv. 3. arXiv, (151100561), 1. Dostupné z: https://doi.org/10.48550/arXiv.1511.00561

MIKULSKÝ, Petr, 2021. Detekce pohybujících se objektů ve videu s využitím neuronových sítí. Brno. Diplomová práce. Vysoké učení technické v Brně. Vedoucí práce Vojtěch Myška.

DWIVEDI, Priya, 2019. Understanding and Coding a ResNet in Keras. Towards Data Science [online]. [cit. 2022-08-14]. Dostupné z: https://towardsdatascience.com/understanding-and-coding-a-resnet-in-keras-446d7ff84d33

HITCHHIKER, Marco, 2019. Hitchhiker's Guide to Residual Networks (ResNet) in Keras. Towards Data Science [online]. [cit. 2022-08-14]. Dostupné z: https://towardsdatascience.com/understanding-and-coding-a-resnet-in-keras-446d7ff84d33

MALADKAR, Kishan, 2018. Why ResNets Are A Major Breakthrough In Image Processing. Analytics India Magazine [online]. [cit. 2022-08-14]. Dostupné z: https://analyticsindiamag.com/why-resnets-are-a-major-breakthrough-in-image-processing/

KHAN, Salman, 2018. et al. A guide to convolutional neural networks for computer vision. Synthesis lectures on computer vision, 8.1: 1-207.

GANDHI, Rohith, 2018. *R-CNN, Fast R-CNN, Faster R-CNN, YOLO — Object Detection Algorithms: Understanding object detection algorithms* [online]. 1. Towards Data Science [cit. 2022-08-15]. Dostupné z: https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e

GIRSHICK, Ross, 2015. Fast r-cnn in proceedings of the ieee international conference on computer vision (pp. 1440–1448). Piscataway, NJ: IEEE.[Google Scholar].

LIU, Wei, 2016. et al. Ssd: Single shot multibox detector. In: European conference on computer vision. Springer, Cham. p. 21-37.

NIXON, Mark; AGUADO, Alberto, 2019. Feature extraction and image processing for computer vision. Academic press. Dostupné z: https://books.google.cz/books?hl=cs&lr=&id=KcW-DwAAQBAJ&oi=fnd&pg=PP1&dq=Nixon,+Mark%3B+Aguado,+Alberto+(2019).+Feature+Extraction+and+Image+Processing+for+Computer+Vision&ots=10nw4qWB5S&sig=SV45VOe5QmO8_umrq18bzsEh78M&redir_esc=y#v=onepage&q&f=false

SHAHDADPURI, Nakul, 2020. Real Image of Computer Vision Application and its Impact: Future and Challenges. *ResearchGate*. Gwalior. 13(53), 62 - 75.

REDMON, Joseph, Santosh DIVVALA, Ross GIRSHICK a Ali FARHADI, 2016. You Only Look Once: Unified, Real-Time Object Detection. Institute of Electrical and Electronics Engineers Inc. [online]. 345 E 47TH ST, NEW YORK, NY 10017 USA, 2016(4), 779-788 [cit. 2023-12-01]. ISSN 1063-6919. Dostupné z: doi:10.1109/CVPR.2016.91

KARMINI, Grace, 2023. Introduction to YOLO Algorithm for Object Detection [online]. [cit. 2023-11-30]. Dostupné z: https://www.section.io/engineering-education/introduction-to-yolo-algorithm-for-object-detection/?fbclid=IwAR2hOP4uF302MERjEtTGistSzQ1nSuzEVGtp9asZeB5Rlg3x0dI0eaLb7y4

RUSTAMY, Fahim, 2023. DEtection TRansformer (DETR) vs. YOLO for object detection. [online]. [cit. 2023-11-30]. Dostupné z: https://medium.com/@faheemrustamy/detection-transformer-detr-vs-yolo-for-object-detection-baeb3c50bc3

FANG, Wei, Lin WANG a Peiming REN, 2020. Tinier-YOLO: A Real-Time Object Detection Method for Constrained Environments. Institute of Electrical and Electronics Engineers Inc. [online]. School of IoT Engineering, Jiangnan University, Wuxi, 214122, China, 2020(8), 1935-1944 [cit. 2023-12-01]. ISSN 21693536. Dostupné z: doi:10.1109/ACCESS.2019.2961959

SÁNCHEZ LEAL, Isaac, Eiraj SAQIB, Irida SHALLARI, Axel JANTSCH, Silvia KRUG a Mattias O'NILS, 2023. Waist Tightening of CNNs: A Case study on Tiny YOLOv3 for Distributed IoT Implementations [online]. 10.1145/3576914.3587518. Mid Sweden University, Sundsvall, Sweden: Association for Computing Machinery 241-246 [cit. 2023-12-01]. ISBN 979-840070049-1. Dostupné z: doi:10.1145/3576914.3587518

KUSHWAH, Shivam, 2023. Object Detection Using Yolov4. AURIGA IT CONSULTING PVT LTD. Aurigait.com [online]. [cit. 2023-12-06]. Dostupné z: https://aurigait.com/blog/object-detection-using-yolov4/

NELSON, Joseph a Jacob SOLAWETZ, 2020. Responding to the Controversy about YOLOv5. © 2023 ROBOFLOW, INC. Roboflow [online]. [cit. 2023-12-06]. Dostupné z: https://blog.roboflow.com/yolov4-versus-yolov5/

CARION, Nicolas, Francisco MASSA, Gabriel SYNNAEVE, Nicolas USUNIER a Alexander KIRILLOV, 2020. End-to-End Object Detection with Transformers [online]. 1-26 [cit. 2023-12-06]. Dostupné z: doi:10.48550/arXiv.2005.12872

JOCHER, Glenn a Ayush CHAURASIA, 2023. Ultralytics [online]. [cit. 2023-12-07]. Dostupné z: https://docs.ultralytics.com/#yolo-licenses-how-is-ultralytics-yolo-licensed

CHEN, Jieshan, Mulong XIE, Zhenchang XING, Chunyang CHEN, Xiwei XU a Liming ZHU, 2020. Object Detection for Graphical User Interface: Old Fashioned orDeep Learning or a Combination? ResearchGater [online]. 1-13 [cit. 2024-03-08]. Dostupné z: doi: https://doi.org/10.48550/arXiv.2008.05132

SOLAWETZ, Jacob a FRANCESCO, 2023. What is YOLOv8? The Ultimate Guide. [online]. [cit. 2023-12-07]. Dostupné z: https://blog.roboflow.com/whats-new-in-yolov8/

MULONGXIE, 2022. UIED – UI element detection, detecting UI elements from UI screenshots or drawnings. Dostupné z: https://github.com/MulongXie/UIED

SUCHUAN, 2021. How Do You Use Deep Learning to Identify UI Components? Dostupné z: https://www.alibabacloud.com/blog/how-do-you-use-deep-learning-to-identify-ui-components_597859

ALTINBAS, MEHMET & SERIF, TACHA. (2022). GUI Element Detection from Mobile UI Images Using YOLOv5. 10.1007/978-3-031-14391-5_3.

Dataset UI_elements od uživatele cingoz_ - https://universe.roboflow.com/cha-ebebh/cingoz_/dataset/2

Seznam obrázků, tabulek, grafů a zkratek

5.5 Seznam obrázků

Obrázek 1 – Schéma Umělé inteligence, strojového učení a hlubokého učení Zdroj: Chollet, 2019
12
Obrázek 2 – Gestaltovy zákony – Principy percepčního pole Zdroj: https://slidetodoc.com/pednka-pro-ppravn-kurz-ke-studiu-psychologie-djiny/ 16
Obrázek 3 – Model neuronu Zdroj: https://portal.matematickabiologie.cz/index.php?pg=analyza-a-hodnoceni-biologickych-dat--umela-inteligence--neuronove-site-jednotlivy-neuron--jednotlivy-neuron--matematicky-model-a-aktivni-dynamika-neuronu 17
Obrázek 4 – Jednoduchá architektura vícevrstvé neuronové sítě Zdroj: Mikulovský, 2021 19
Obrázek 5 – Architektura VGGnet Zdroj: https://www.researchgate.net/figure/VGGNet-architecture-19_fig2_333242381 20
Obrázek 6 – Architektura sítě SegNet Zdroj: https://www.researchgate.net/figure/SegNet-architecture_fig5_343566178 21
Obrázek 7 – Standardní zbytkový blok a zbytkový blok s úzkým hrdlem Zdroj: https://www.researchgate.net/figure/Standard-Residual-vs-Bottleneck-Residual-Block-He-et-al-2015_fig4_337486420 22
Obrázek 8 – Příklad síťových architektur pro ImageNet Zdroj: https://arxiv.org/pdf/1512.03385.pdf 23
Obrázek 9 – Konvoluční síť v Deep Learningu Zdroj: https://www.technickytydenik.cz/rubriky/ict/deep-learning-pro-segmentaci-obrazu_42430.html 24
Obrázek 10 – Architektura konvoluční vrstvy Zdroj: https://www.optixs.cz/slovnik-17/dekonvoluce-a-konvoluce-68s 25
Obrázek 11 – Architektura R-CNN Zdroj: https://medium.com/dair-ai/papers-explained-14-rcnn-ede4db2de0ab 28
Obrázek 12 – SVM metoda za použití konvoluční vrstev pro klasifikaci obrazu Zdroj: https://www.datasciencecentral.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms/ 28
Obrázek 13 – Architektura Fast R-CNN Zdroj: https://arxiv.org/pdf/1504.08083.pdf 29
Obrázek 14 – Architektura SSD Zdroj: https://medium.com/@aiclubvast/road-damage-detection-system-895d3a36613a 30
Obrázek 15 – SSD příklad postupu Zdroj: https://www.researchgate.net/figure/SSD-default-boxes-GT-stands-for-ground-truth-Figure-taken-from-1_fig2_329747508 30
Obrázek 16 – Architektura YOLOv3 Zdroj: https://miro.medium.com/v2/resize:fit:2000/format:webp/1*d4Eg17IVJ0L41e7CTWLLSg.png 32
Obrázek 17 – Struktura vrstev Tiny-YOLOv3 Zdroj: https://www.researchgate.net/publication/338162578/figure/fig1/AS:839998031032320@1577282545408/The-network-structure-of-Tiny-YOLO-V3.jpg 33
Obrázek 18 – Model DETR Zdroj: https://lh5.googleusercontent.com/rPXh8HHA-KnqdFuTjXsqBLS3A1y1QGZ5eQnqt3RFV-

ZcogCZlku7VpnFFeIcIBmIYN4nevzDlWOEq69nS2d_ZDWM5fO_RgZ7OJrAh6X5h3OvLLXlqBDc3vGvTvfQRTG2KAu7Mv3tH4QJ4Nl_OtSHxdo 35
Obrázek 19 – Příklad vstupního snímku pro UIED 37
Obrázek 20 – Ukázka obrázků nezávislého test setu TEST_1_UI Zdroj: https://www.peppersquare.com/blog/4-critical-ui-elements-of-remarkable-interfaces/ TEST_2_UI Zdroj: https://www.vectorstock.com/royalty-free-vector/set-of-flat-design-ui-elements-for-website-vector-4742917 TEST_3_UI Zdroj: https://www.vectorstock.com/royalty-free-vector/user-interface-elements-web-ui-element-mobile-vector-33133683 41
Obrázek 21 – Ukázka obrázků označených instancí nezávislého test setu 41
Obrázek 22 – Ukázka prostředí Google Colab 43
Obrázek 23 – Ukázka již detekovaných UI prvků modelem YOLOv8s 56
Obrázek 24 – Ukázka již detekovaných UI prvků modelem Faster R-CNN resnet 56
Obrázek 25 – Ukázka již detekovaných UI prvků modelem DETRresnet101 56

5.6 Seznam scriptů

Script 1 – Stažení datasetu pro trénování modelů 46
Script 2 – Příprava adresáře pro model DETR 46
Script 3 – Vytvoření config.py pro nastavení cest k datasetu pro trénování modelu DETR 46
Script 4 – Spuštění trénování modelu DETR 47
Script 5 – Detekce vybraného obrázku z test datasetu modelem DETR 47
Script 6 – Příprava adresáře pro model Faster R-CNN 48
Script 7 – Úprava config.py pro nastavení cest k datasetu pro trénování modelu Faster R-CNN 49
Script 8 – Spuštění trénování modelu Faster R-CNN 49
Script 9 – Detekce vybraného obrázku z test datasetu modelem Faster R-CNN 49
Script 10 – Nahrání adresáře s modelem RetinaNet z Google drive 50
Script 11 – Úprava config.py pro nastavení cest k datasetu pro trénování modelu RetinaNet 51
Script 12 – Části kodu pro trénování a spuštění detekce modelu RetinaNet 51
Script 13 – Příprava adresáře Ultralytics 52
Script 14 – Stažení datasetu z Roboflow ve formátu pro YOLOv8 model 53
Script 15 – Spuštění trénování modelem YOLOv8 53
Script 16 - Detekce vybraného obrázku z test datasetu modelem YOLOv8 53

5.7 Seznam tabulek

Tabulka 1 – Celkový počet instancí k detekci 42
Tabulka 2 – Ukázka výsledné tabulky jednoho z modelů 55
Tabulka 3 – Výsledek porovnání klasifikačních metrik modelů 58

5.8 Seznam grafů

Graf 1 – Výsledky trénování modelů DETR 48

Graf 2 – Výsledky trénování modelů Faster R-CNN 50
Graf 3 – výsledek trénování modelu RetinaNet..... 52
Graf 4 – Grafy výsledků trénování modelů Yolov8 53
Graf 5 – Zobrazení výsledků z Tabulky 3 do spojnicového grafu..... 59

5.9 Seznam použitých zkratek

CV	Computer Vision
UI	User Interface
CNN	Convolitonal Neural Network
VGG	Visual Geometry Group
SSD	Single Shot Multibox Detector
RPN	Regional Personal NetworkS
YOLO	You Only Look Once

Přílohy

Příloha A

Tabulka DETR

DETR									
resnet50 - 37,41%	TP	5	0	0	35	12	33	0	2
	FP	4	0	2	3	0	1	7	0
	FN	44	3	4	63	10	32	11	7
	CLK_PČ_DET	53	3	6	101	22	66	18	9
	Počty prvků v UI	53	3	6	101	22	66	18	9
	Precision	0,555555556	0	0	0,92105263	1	0,97058824	0	1
	Recall	0,102040816	0	0	0,35714286	0,54545455	0,50769231	0	0,2222
	F1-score	0,172413793	0	0	0,51470588	0,70588235	0,66666667	0	0,3636
	F1-score (Avg.)	0,302913132	Precision (Avg.)	0,555899553	Recall (Avg.)	0,21681909			

	38,48%	Button	Checkbox	dropdown	icon	input	label	radio	switch
resnet50Dc5	TP	6	0	0	40	11	29	0	2
	FP	6	0	3	3	5	0	2	0
	FN	41	3	3	58	6	37	16	7
	CLK_PČ_DET	53	3	6	101	22	66	18	9
	Počty prvků v UI	53	3	6	101	22	66	18	9
	Precision	0,5	0	0	0,93023256	0,6875	1	0	1
	Recall	0,127659574	0	0	0,40816327	0,64705882	0,43939394	0	0,2222
	F1-score	0,203389831	0	0	0,56737589	0,66666667	0,61052632	0	0,3636
	F1-score (Avg.)	0,301449383	Precision (Avg.)	0,51471657	Recall (Avg.)	0,23056223			

	39,93%	Button	Checkbox	dropdown	icon	input	label	radio	switch
resnet101	TP	9	0	0	60	9	14	0	4
	FP	4	0	2	5	0	0	4	0
	FN	40	3	4	36	13	52	14	5
	CLK_PČ_DET	53	3	6	101	22	66	18	9
	Počty prvků v UI	53	3	6	101	22	66	18	9
	Precision	0,692307692	0	0	0,92307692	1	1	0	1
	Recall	0,183673469	0	0	0,625	0,40909091	0,21212121	0	0,4444
	F1-score	0,290322581	0	0	0,74534161	0,58064516	0,35	0	0,6154
	F1-score (Avg.)	0,322711747	Precision (Avg.)	0,576923077	Recall (Avg.)	0,23429125			

	41,36%	Button	Checkbox	dropdown	icon	input	label	radio	switch
resnet101Dc5	TP	13	0	0	51	9	11	0	2
	FP	9	0	1	4	3	0	10	2
	FN	31	3	5	46	10	55	8	5
	CLK_PČ_DET	53	3	6	101	22	66	18	9
	Počty prvků v UI	53	3	6	101	22	66	18	9
	Precision	0,590909091	0	0	0,92727273	0,75	1	0	0,5
	Recall	0,295454545	0	0	0,5257732	0,47368421	0,16666667	0	0,2857
	F1-score	0,393939394	0	0	0,67105263	0,58064516	0,28571429	0	0,3636
	F1-score (Avg.)	0,28687348	Precision (Avg.)	0,471022727	Recall (Avg.)	0,21841161			

Tabulka Faster R-CNN

Faster R-CNN	71,94%	Button	Checkbox	dropdown	icon	input	label	radio	switch
resnet50	TP	38	3	3	77	16	19	10	8
	FP	12	0	3	6	3	1	1	0
	FN	3	0	0	18	3	46	7	1
	CLK_PČ_DET	53	3	6	101	22	66	18	9
	Počty prvků v UI	53	3	6	101	22	66	18	9
	Precision	0,76	1	0,5	0,927710843	0,842105263	0,95	0,909090909	1
	Recall	0,926829268	1	1	0,810526316	0,842105263	0,292307692	0,588235294	0,8888889
	F1-score	0,835164835	1	0,666666667	0,865168539	0,842105263	0,447058824	0,714285714	0,9411765
	F1-score (Avg.)	0,788953289	Precision (Avg.)	0,861113377	Recall (Avg.)	0,79361159			

	42,09%	Button	Checkbox	dropdown	icon	input	label	radio	switch
vgg16	TP	26	3	0	21	23	19	4	1
	FP	11	0	3	0	0	0	1	5
	FN	16	0	3	80	1	47	13	3
	CLK_PČ_DET	53	3	6	101	24	66	18	9
	Počty prvků v UI	53	3	6	101	22	66	18	9
	Precision	0,702702703	1	0	1	1	1	0,8	0,1666667
	Recall	0,619047619	1	0	0,207920792	0,958333333	0,287878788	0,235294118	0,25
	F1-score	0,658227848	1	0	0,344262295	0,978723404	0,447058824	0,363636364	0,2
	F1-score (Avg.)	0,498988592	Precision (Avg.)	0,708671171	Recall (Avg.)	0,444809331			

Tabulka RetinaNet

RetinaNet	14,39%	Button	Checkbox	dropdown	icon	input	label	radio	switch
	TP	5	0	0	0	1	6	0	2
	FP	7	0	4	8	2	1	0	4
	FN	41	3	2	93	19	59	18	3
	CLK_PČ_DET	53	3	6	101	22	66	18	9
	Počty prvků v UI	53	3	6	101	22	66	18	9
	Precision	0,416666667	0	0	0	0,333333333	0,857142857	0	0,33333333
	Recall	0,108695652	0	0	0	0,05	0,092307692	0	0,4
	F1-score	0,172413793	0	0	0	0,086956522	0,166666667	0	0,36363636
	F1-score (Avg.)	0,098709168	Precision (Avg.)	0,242559524	Recall (Avg.)	0,081375418			

Tabulka YOLOv8

YOLOv8	34,89%	Button	Checkbox	dropdown	icon	input	label	radio	switch
YOLOv8n	TP	33	0	0	14	8	7	2	3
	FP	10	2	4	6	4	1	1	2
	FN	10	1	2	81	10	58	15	4
	CLK_PČ_DET	53	3	6	101	22	66	18	9
	Počty prvků v UI	53	3	6	101	22	66	18	9
	Precision	0,76744186	0	0	0,7	0,666666667	0,875	0,666666667	0,6
	Recall	0,76744186	0	0	0,147368421	0,444444444	0,107692308	0,117647059	0,4285714
	F1-score	0,76744186	0	0	0,243478261	0,533333333	0,191780822	0,2	0,5
	F1-score (Avg.)	0,304504285	Precision (Avg.)	0,534471899	Recall (Avg.)	0,25164569			

	57,55%	Button	Checkbox	dropdown	icon	input	label	radio	switch
YOLOv8s	TP	31	0	1	52	18	18	2	6
	FP	10	2	4	6	4	0	3	3
	FN	12	1	1	43	0	48	13	0
	CLK_PČ_DET	53	3	6	101	22	66	18	9
	Počty prvků v UI	53	3	6	101	22	66	18	9
	Precision	0,756097561	0	0,2	0,896551724	0,818181818	1	0,4	0,6666667
	Recall	0,720930233	0	0,5	0,547368421	1	0,272727273	0,133333333	1
	F1-score	0,738095238	0	0,285714286	0,679738562	0,9	0,428571429	0,2	0,8
	F1-score (Avg.)	0,504014939	Precision (Avg.)	0,592187221	Recall (Avg.)	0,521794907			