Detekce nasazení roušky/respirátoru ve videu

Matěj Vaculík

Bakalářská práce
2023

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav informatiky a umělé inteligence

Akademický rok: 2022/2023

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Matěj Vaculík**
Osobní číslo:	**A20387**
Studijní program:	**B0613A140020 Softwarové inženýrství**
Forma studia:	**Prezenční**
Téma práce:	**Detekce nasazení roušky/respirátoru ve videu**
Téma práce anglicky:	**Mask/respirator Detection in Video**

Zásady pro vypracování

1. Vypracujte literární rešerši zabývající se modely strojového učení pro detekci objektů v obraze/videu.
2. Stručně popište proces učení a aplikace hlubokých neuronových sítí.
3. Vyberte několik různých detekčních modelů vhodných pro detekci nasazení roušky/respirátoru. Při výběru modelů se zaměřte na požadavek použití výsledného modelu pro detekci ve videu.
4. Natrénujte vytipované modely na veřejně dostupném datasetu Face Mask Detection.
5. Vyberte nejlepší z natrénovaných modelů a otestujte jeho funkci ve videu.

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

[1] ROSEBROCK, Adrian. Deep learning for computer vision with Python: Practitioner Bundle. PyImageSearch, 2017. ISBN 978-1-986723817.
[2] ROSEBROCK, Adrian. Deep learning for computer vision with Python: Starter bundle. PyImageSearch, 2017. ISBN 9781986538138.
[3] ROSEBROCK, Adrian. Deep learning for computer vision with Python: ImageNet bundle. PyImageSearch, 2017. ISBN 978-1-986723848.
[4] GOODFELLOW, Ian, Yoshua BENGIO a Aaron COURVILLE. Deep learning: ImageNet bundle. Adaptive computation and machine learning. Cambridge, Massachussetts ; London: The MIT Press, 2016. ISBN 9780262035613.
[5] SHAOHUI, Lin, Cai LING, Lin XIANMING a Ji RONGRONG. Masked face detection via a modified LeNet. Volume 218. Neurocomputing, 2016. ISBN 197–202.
[6] ADHINATA, F. D., D. P. RAKHMADANI, M. WIBOWO a A. JAYADI. A Deep Learning Using DenseNet201 to Detect Masked or Non-masked Face. 9(1). JUITA Jurnal Informatika, 2021. ISBN 115-121.

Vedoucí bakalářské práce: **Ing. Alžběta Turečková**
Ústav informatiky a umělé inteligence

Datum zadání bakalářské práce: **2. prosince 2022**
Termín odevzdání bakalářské práce: **26. května 2023**

L.S.

doc. Ing. Jiří Vojtěšek, Ph.D. v.r.
děkan

prof. Mgr. Roman Jašek, Ph.D., DBA v.r.
ředitel ústavu

Ve Zlíně dne 7. prosince 2022

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne 26.5.2023

Matěj Vaculík
podpis studenta

ABSTRAKT

Tato bakalářská práce se zaměřuje na detekci správného nasazení roušek nebo respirátorů ve videu v kontextu pandemie Covid-19. Teoretická část zkoumá koncepty a techniky strojového učení a konvolučních neuronových sítí pro detekci objektů. Praktická část zahrnuje výběr, přípravu a trénování detekčních modelů, s důrazem na model YOLOv5. Testování prokázalo účinnost detekce nošení masek s možností zlepšení pro nesprávně nasazené masky. Práce představuje využití strojového učení pro detekci nošení masek ve videu a navrhuje implementaci v soukromých klinikách a menších zdravotnických zařízeních pro sledování dodržování pravidel nošení roušek nebo respirátorů.

Klíčová slova: detekce roušek, strojové učení, trénování modelů, model YOLOv5

ABSTRACT

This bachelor's thesis focuses on the detection of correct mask or respirator usage in videos within the context of the Covid-19 pandemic. The theoretical part explores concepts and techniques of machine learning and convolutional neural networks for object detection. The practical part involves the selection, preparation, and training of detection models, with an emphasis on the YOLOv5 model. Testing has demonstrated the effectiveness of mask-wearing detection with potential improvements for incorrectly worn masks. The thesis showcases the application of machine learning for mask-wearing detection in videos and proposes its implementation in private clinics and smaller healthcare facilities to monitor compliance with mask-wearing rules.

Keywords: mask detection, machine learning, model training, YOLOv5

Chtěl bych poděkovat vedoucí bakalářské práce Ing. Alžbětě Turečkové za konzultace, rady a odborné vedení při vypracování práce.

Prohlašuji, že odevzdaná verze bakalářské/diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD

Bakalářská práce se zaměřuje na detekci nasazení roušky a respirátoru ve videu.

Problém správného nasazení ochranných pomůcek, jako jsou roušky a respirátory, se stal jedním z klíčových témat v souvislosti s pandemií Covid-19.

Teoretická část práce je zaměřena konceptům a technikám strojového učení a hlubokých konvolučních neuronových sítí, které jsou základem pro detekci objektů. Tato část zahrnuje přehled různých modelů detekce objektů, jako je R-CNN, Fast R-CNN, Faster R-CNN, SSD a YOLO. Dále se zaměřujeme na procesy učení, včetně standardů datasetů a využití předtrénovaných modelů, a na metriky pro evaluaci modelů.

V praktické části je práce zaměřena na výběr datasetu z platformy Kaggle, jeho přípravu a konfiguraci. Následuje výběr a trénování detekčních modelů na vybraných datech, vyhodnocení a srovnání výsledků modelů. Práce je ukončena výběrem nejlepšího detekčního modelu, jeho implementací a testováním ve videu. Na závěr jsou provedena vyhodnocení a diskuze o dosažených výsledcích. Diskutuje se o možných vylepšeních a dalších přínosech.

Cílem této práce je porozumět procesu detekce nasazení roušky a respirátoru ve videu pomocí moderních technik strojového učení a hlubokých neuronových sítí, identifikovat a implementovat nejvhodnější model pro tuto úlohu.

I. TEORETICKÁ ČÁST

1 STROJOVÉ UČENÍ

Strojové učení je podoblast umělé inteligence, která se zaměřuje na vývoj algoritmů, jež umožňují počítačům učit se a adaptovat se na nová data, aniž by byly explicitně naprogramovány. Cílem strojového učení je vytvořit modely, které dokáží rozpoznávat a klasifikovat objekty, předpovídat budoucí události nebo optimalizovat strategie pro dosažení cílů.

Detekce objektů je jedním z hlavních úkolů strojového učení v oblasti počítačového vidění. Cílem detekce objektů je identifikovat a lokalizovat objekty v obraze nebo ve videu. K tomu se používají různé metody a techniky, které mohou být založeny na klasických přístupech nebo na hlubokém učení. [1]

Ve strojovém vidění rozlišujeme čtyři základní úlohy: klasifikace, sémantická segmentace, detekce objektu a segmentace instance. Tyto úlohy z oblasti zpracování obrazu jsou často řešeny za pomoci modelů umělé inteligence [2]

1.1 Klasifikace

Klasifikace je proces kategorizace objektů, dat nebo jevů do předem definovaných tříd nebo kategorií. Tento proces se používá v mnoha oblastech, jako je například strojové učení, statistika, biologie, psychologie, ekonomie a další. [3]

Při klasifikaci se data analyzují a přiřazují se jim určité třídy nebo kategorie na základě charakteristických vlastností, které jsou pro danou třídu typické. Tyto charakteristiky se mohou odlišovat v závislosti na aplikaci a jsou založeny na různých typech vstupních dat, například textu, obrazu, zvuku nebo číselných hodnotách. [4]

Klasifikace se obvykle provádí pomocí algoritmů strojového učení, které jsou trénovány na základě množiny dat s již přiřazenými třídami. Tyto algoritmy poté mohou být použity k automatické klasifikaci nových dat. [1]

Příklady aplikací klasifikace jsou například rozpoznávání obrazů, detekce spamových zpráv v e-mailových schránkách, diagnostika nemocí a další. Klasifikace je důležitým nástrojem v oblasti strojového učení a umožňuje automatizaci mnoha procesů, které by jinak byly časově náročné a náchylné k chybám. [5]

1.2 Sémantická segmentace

Sémantická segmentace (viz Obrázek 1.) je proces rozdělení obrazu nebo scény na sémantické části, které mají význam v kontextu dané aplikace. Tento proces se používá v počítačovém vidění a strojovém učení a je důležitým krokem pro mnoho aplikací, jako je například autonomní řízení, rozpoznávání objektů, detekce překážek a další. [4]

Při sémantické segmentaci se každému pixelu v obrázku nebo části scény přiřadí sémantická třída, například osoba, auto, strom , atd.. Tento proces může být proveden pomocí různých technik, včetně konvolučních neuronových sítí, které jsou schopny naučit se rozpoznávat různé sémantické třídy v obraze na základě trénovacích dat.

Sémantická segmentace umožňuje strojům lépe porozumět vizuálnímu obsahu obrazů a scén, to umožňuje vytvořit pokročilé aplikace v oblasti strojového učení, jako jsou například autonomní vozidla, robotika, detekce objektů a další. [3]

1.3 Detekce objektu

Detekce objektů (viz Obrázek 1.) je proces automatického nalezení a lokalizace objektů v digitálním obraze nebo videu. Cílem detekce objektů je identifikovat, zda se v obraze nachází určitý typ objektu a určit jeho polohu.

Detekce objektů se používá v mnoha oblastech, jako jsou například počítačové vidění, průmyslová automatizace, bezpečnostní systémy, medicína a další. Mezi běžně detekované objekty patří například lidé, vozidla, zvířata, budovy, předměty a další. [3]

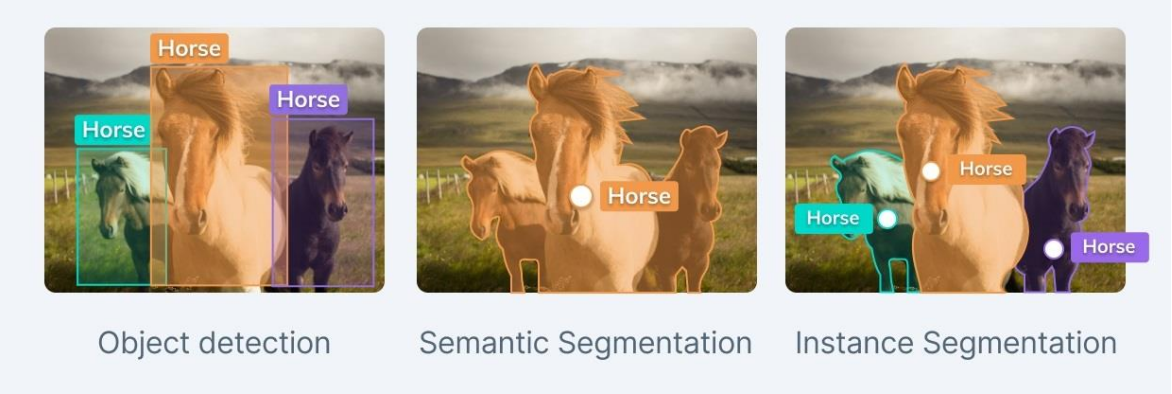
Proces detekce objektů obvykle zahrnuje několik kroků. Nejprve se získá digitální obraz nebo video, které se bude analyzovat. Poté se použije algoritmus detekce objektů, který hledá určité znaky, jako jsou hrany, tvary, textury nebo barvy, které jsou charakteristické pro hledaný objekt. Pokud je objekt nalezen, algoritmus určí jeho polohu v obraze a přiřadí mu určitou kategorii nebo třídu. [6]

1.4 Segmentace instance

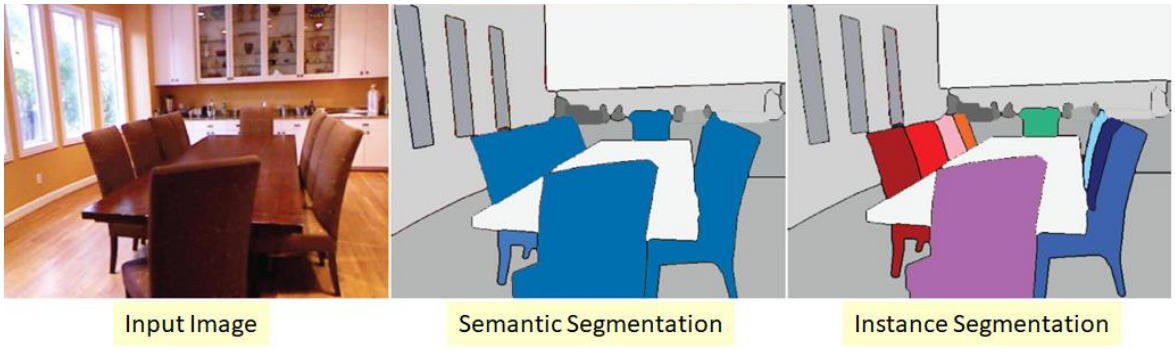
Segmentace instance (viz Obrázek 1.) je proces rozdělení obrazu nebo scény na jednotlivé instance objektů, přičemž každá instance je identifikována jako samostatný a jedinečný objekt. Oproti sémantické segmentaci, která rozděluje obraz nebo scénu do sémantických tříd, segmentace instance umožňuje rozpoznat každý jednotlivý objekt v obraze a přiřadit mu jedinečnou identifikaci (viz Obrázek 2.). [7]

Při segmentaci instance se každá instance objektu v obraze rozliší od ostatních a to poskytuje počítači nejen identifikovat každý objekt v obraze, ale také sledovat jeho pohyb a změny v čase. To je užitečné pro použití, jako je například počítačové vidění, sledování objektů, autonomní řízení a další. [3]

Segmentace instance může být provedena pomocí různých metod, jako jsou například detekce objektů, sledování objektů, klasifikace objektů a další. Tyto metody umožňují rozpoznat jednotlivé objekty v obraze a přiřadit jim jedinečné identifikátory. [4]



Obrázek 1. Detekce objektů, sémantické segmentace, segmentace instance [8]



Obrázek 2. Příklad hlavního rozdílu [9]

2 HLUBOKÁ KONVOLUČNÍ NEURONOVÁ SÍŤ

Hluboká konvoluční neuronová síť (Deep Convolutional Neural Network), je typ konvoluční neuronové sítě s velkým počtem vrstev, což umožňuje síti učit se složitější a abstraktnější reprezentace dat. Používá se zejména pro zpracování obrazových a vizuálních dat a byla úspěšně aplikována v širokém spektru úloh, jako jsou klasifikace obrazů, detekce objektů a segmentace obrazů. [10]

Architektura hluboké konvoluční sítě je podobná architektuře konvoluční neuronové sítě. Hlavní rozdíl mezi nimi spočívá v hloubce sítě, která se odvíjí od počtu vrstev. Hluboké konvoluční sítě obsahuje více vrstev než klasická konvoluční neuronová síť, což zlepšuje její schopnost generalizace a přesnosti.

Hluboké konvoluční sítě jsou používány v různých úlohách zpracování obrazu, jako jsou klasifikace, sémantická segmentace a detekce objektů. Tyto úkoly jsou zaměřeny na různé aspekty analýzy obrazu. Níže je každý z těchto úkolů popsán v kontextu hlubokých konvolučních sítí. [11]

2.1 Neuronová síť

Neuronová síť je typ počítačového modelu inspirovaného strukturou a fungováním lidského mozku. Základní jednotkou neuronové sítě je „umělý neuron“, který je navržený tak, aby napodoboval funkci biologických neuronů v mozku.

Umělé neurony jsou propojeny do sítě, která umožňuje komplexní zpracování informací. Každý neuron přijímá informace od jiných neuronů, zpracovává je a předává je dál. Tento proces je řízen sadou vah, které určují, jak silně každý vstup ovlivňuje výstup neuronu. [12]

Neuronové sítě se učí přizpůsobováním těchto vah v reakci na tréninková data. To se obvykle provádí pomocí algoritmu zvaného „zpětná propagace chyby“, který postupně upravuje váhy sítě tak, aby minimalizoval rozdíl mezi jejím aktuálním výstupem a požadovaným výstupem.

Neuronové sítě se vyznačují svou schopností učit se z příkladů, generalizovat z těchto příkladů na nové situace a tolerovat určitý stupeň šumu nebo nejistoty v datech. To z nich dělá silný nástroj pro řadu úloh, včetně rozpoznávání obrazu, rozpoznávání řeči, předpovědí, rozhodování a mnoho dalších využití.

Existují různé typy neuronových sítí, včetně vícevrstvých perceptronů, konvolučních neuronových sítí, rekurentních neuronových sítí a hlubokých neuronových sítí. Každý typ má své výhody a nevýhody a je vhodný pro různé druhy problémů. [2]

2.2 Konvoluční neuronová síť

Konvoluční neuronová síť (Convolutional Neural Network), neboli zkráceně CNN, je typ umělé neuronové sítě, která byla speciálně navržena pro zpracování obrazových a vizuálních dat. CNN využívá konvoluční vrstvy, které se automaticky učí detekovat různé vzory a vlastnosti ve vstupních datech, jako jsou hrany, textury a tvary. Tyto konvoluční vrstvy umožňují zachování prostorových informací, což je důležité pro úkoly, jako je klasifikace obrazů, detekce objektů a segmentace obrazů.

Architektura konvoluční neuronové sítě se skládá z několika typů vrstev, jako jsou konvoluční vrstvy, pooling vrstvy a klasifikace, neboli plně propojené vrstvy. Konvoluční vrstvy jsou zodpovědné za extrakci rysů z dat, zatímco pooling vrstvy slouží k redukci prostorových rozměrů a ke zvýšení invariance na posun. Plně propojené vrstvy se často nacházejí na konci sítě a slouží k integraci rysů a produkci výsledných predikcí. [13]

2.3 Modely

Každý model má své vlastní výhody a nevýhody a je důležité vybrat ten nejvhodnější model pro danou úlohu na základě požadavků a dostupných dat. Výběr vhodného modelu závisí na řadě faktorů, jako jsou dostupnost dat, výkon a požadovaná přesnost a rychlost.

Důležitou součástí využití modelů umělé inteligence pro zpracování obrazu je také předzpracování dat. To zahrnuje například normalizaci obrazů, odstranění šumu nebo převedení obrazů do vhodného formátu.

V poslední době se také využívají tzv. předtrénované modely, které jsou natrénovány na velkém množství dat a umožňují snadnější využití pro konkrétní úlohy. [14]

2.4 Detekce objektu

Detekce objektů v obraze nebo videu se skládá z několika kroků, které umožňují identifikovat a lokalizovat objekty v obraze. Tyto kroky zahrnují získání obrazu nebo videa, předzpracování obrazu, výběr zájmových oblastí, extrakci příznaků a klasifikaci a lokalizaci objektů. Každý z těchto kroků má svůj význam a hraje důležitou roli při detekci objektů.

Poté následuje samotný proces detekce objektů, který se skládá ze tří kroků:

1. **Výběr zájmových oblastí (region proposal)**

   Cílem tohoto kroku je identifikovat oblasti obrazu, ve kterých je pravděpodobné, že se nacházejí objekty. Tento krok může být prováděn různými způsoby, jako jsou například metodou selektivního vyhledávání (Selective Search) nebo metodou založenou na konvolučních neuronových sítích (RPN - Region Proposal Network).

2. **Extrakce příznaků z těchto oblastí**

   Po výběru zájmových oblastí se provádí extrakce příznaků z těchto oblastí. Tento krok se obvykle provádí pomocí konvoluční neuronové sítě CNN, která umožňuje převést obrazová data na vektorovou reprezentaci, která obsahuje důležité informace o objektu v zájmové oblasti.

3. **Klasifikace a lokalizace objektů**

   Nakonec se provádí klasifikace a lokalizace objektů na základě příznaků extrahovaných z oblastí zájmu. Klasifikace objektů zahrnuje identifikaci, o jaký typ objektu se jedná, například auto, pes, budova apod. Lokalizace objektů zahrnuje určení pozice objektu v obraze, například souřadnice levého horního a pravého dolního rohu. Tento krok se obvykle provádí pomocí regresních sítí, které určují pozici a velikost detekovaných objektů. [15]

Toto rozdělení „multi-stage“ je typické pro R-CNN architektury, které provádí proces detekce v několika krocích. Existují již modernější metody, které tyto kroky provádí současně. Ty se označují jako „single-stage“ detektory. Příklady těchto „single-stage“ detektorů jsou SSD (Single Shot MultiBox Detector) a YOLO (You Only Look Once).

Rozdíl oproti „multi-stage“ detektorům spočívá v tom, že „single-stage“ detektory nevyžadují zvláštní krok pro výběr zájmových oblastí. Místo toho provádějí detekci objektů a klasifikaci přímo na celém obraze najednou. [16]

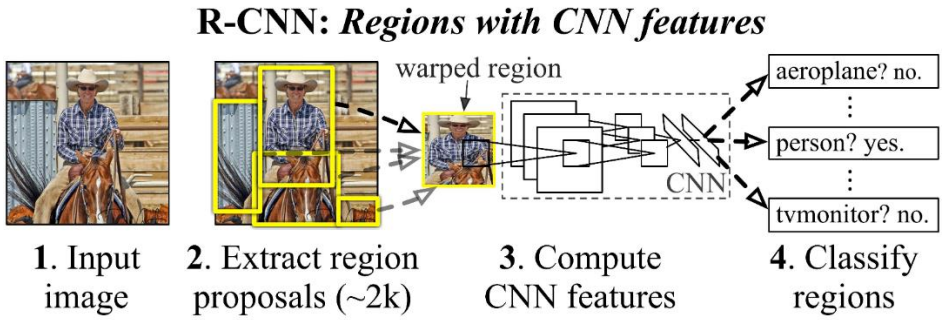
2.5 Detekční modely objektu

Detekční modely, které jsou schopny efektivně zpracovávat velké množství vizuálních dat. Jednotlivé modely se liší ve svých architekturách a způsobech, jakými generují oblasti zájmu a predikují ohraničující boxy a třídy objektů.

2.5.1 R-CNN

R-CNN (viz Obrázek 4.), neboli Region Based Convolutional Neural Network detekuje objekty pomocí algoritmu selektivního vyhledávání oblastí (Selective Search) k získání kandidátských oblastí. Následně jsou tyto oblasti klasifikovány pomocí konvoluční neuronové sítě CNN. Extrahované příznaky jsou poté předány do klasifikačního modelu SVM (Support Vector Machines) k rozhodnutí, zda daná oblast obsahuje objekt a jaký typ objektu to je (viz Obrázek 3.). R-CNN nabízí vysokou přesnost detekce objektů při použití na různých typech obrázků a pro detekci různých typů objektů. [17]

Nicméně, R-CNN má několik nevýhod, jako jsou pomalá rychlost výpočtu, vysoké nároky na výpočetní výkon a nutnost mít k dispozici velké množství trénovacích dat. Také vykazuje citlivost na umístění objektu v obrazu. [18]



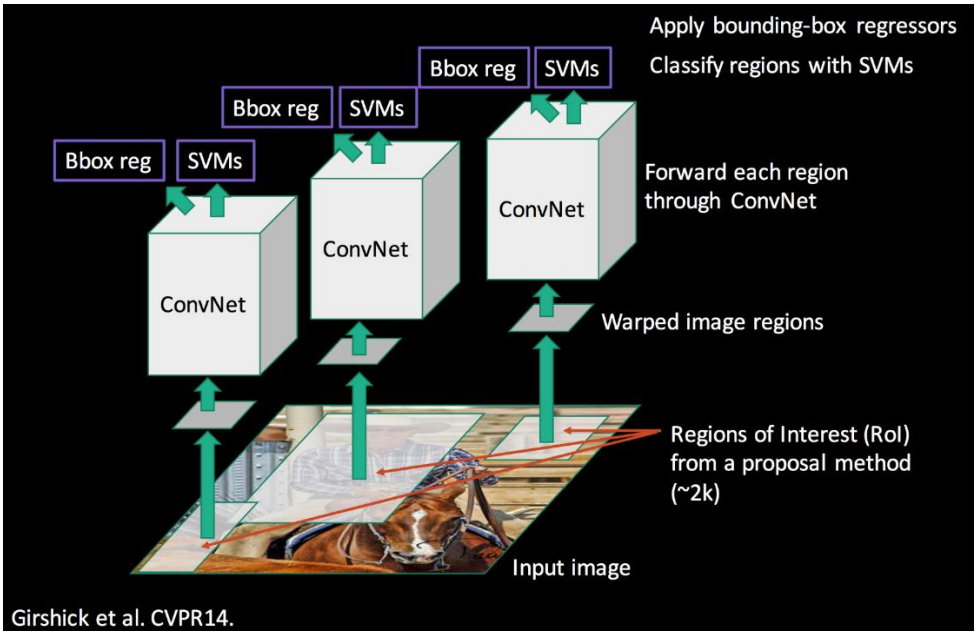
Obrázek 3. Model R-CNN [18]

Selektivní vyhledávání

Tato metoda se soustředí na identifikaci a výběr tzv. Region of Interest (ROI) v obrázku nebo videu. Tento přístup umožňuje redukovat množství dat, které musí být zpracovány, a zaměřit se na klíčové části, kde je nejpravděpodobnější výskyt objektů.

Selektivní vyhledávání využívá různé strategie, jako je tvar, barva nebo textura, k segmentaci obrázku na několik malých oblastí. Tyto oblasti jsou poté postupně sloučeny na základě určitých podobností, dokud nezůstane jen několik velkých oblastí.

Důležité je, že tento proces nezahrnuje žádné třídění nebo klasifikaci objektů. Jeho hlavním úkolem je jen identifikovat oblasti, které by mohly obsahovat potenciální objekty. Selektivní vyhledávání je často používáno jako první krok v systémech pro detekci objektů, který následuje dalším zpracováním, jako je klasifikace pomocí hlubokého učení. [19]



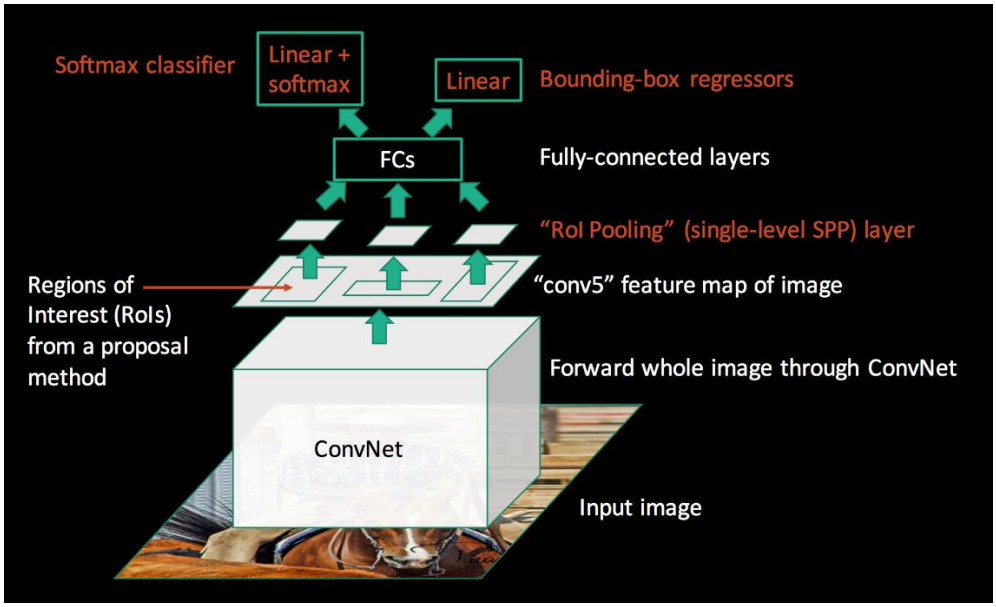
Obrázek 4. Architektura procesu R-CNN [17]

2.5.2 Fast R-CNN

Fast R-CNN se snaží řešit některé z problémů, které byly spojeny s původním modelem R-CNN. Zejména se jedná o vysokou výpočetní náročnost a pomalou rychlost zpracování, která byla způsobena potřebou provádět násobné konvoluce pro každou navrženou oblast.

Fast R-CNN (viz Obrázek 5.) oproti původnímu modelu R-CNN používá jednu konvoluční síť pro celý obrázek namísto použití konvoluční sítě pro každou navrženou oblast zájmu. To výrazně zlepšuje efektivitu a rychlost detekce objektů. Je založen na třech hlavních komponentách [18] :

1. **Konvoluční síť** - Tato síť je zodpovědná za extrakci vlastností z celého obrázku. Vlastnosti jsou pak využity k detekci objektů.
2. **Region of Interest (RoI) Pooling** - Tato metoda je klíčová pro transformaci funkcí získaných konvolučními sítěmi z různých oblastí zájmu na vektory s pevnou délkou, které pak mohou být efektivně zpracovány pomocí plně propojených vrstev.
3. **Plně propojené vrstvy** - Tyto vrstvy mají na starosti identifikaci a lokalizaci objektů. Klasifikační vrstva specifikuje typ rozpoznaného objektu, zatímco lokalizační vrstva definuje umístění a velikost ohraničujícího rámečku kolem daného objektu.



Obrázek 5. Architektura procesu Fast R-CNN [17]

Nicméně, tento model stále závisí na externí metodě pro generování návrhů oblastí. To může být pomalé a neefektivní. Dále Fast R-CNN vyžaduje velké množství paměti, protože musí uchovávat funkční mapy pro každý vstupní obraz a to může být náročné, zejména pro větší obrázky.

2.5.3 Faster R-CNN

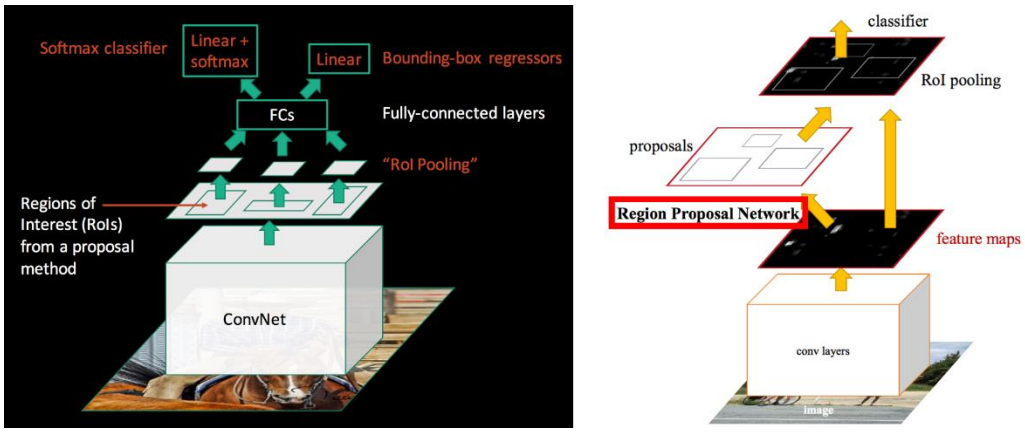
Faster R-CNN (viz Obrázek 6.) je pokročilý model pro detekci objektů, který byl navržen tak, aby byl rychlejší a efektivnější než jeho předchůdci. Model je založen na konvoluční neuronové síti CNN a využívá mechanismus nazvaný Region Proposal Network (RPN) k automatickému navrhování oblastí, kde lze objekty nalézt. Skládá ze dvou hlavních částí:

1. **Region Proposal Network** - je síť zodpovědná za vytváření oblastí (tzv. návrhových oblastí), do kterých lze umístit objekty. Prochází celý vstup a vrací sadu obdélníkových kandidátů na objekty (bounding boxes) a skóre udávající, zda se objekt v oblasti nachází, či nikoli.
2. **Detekční síť** - tato část sítě přebírá návrhy oblastí od RPN a provádí klasifikaci a regresi ohraničujících rámečků. Klasifikace určuje třídu objektu a regrese ohraničujícího rámečku upravuje rozměry a polohu návrhu oblasti tak, aby přesně odpovídala objektu.

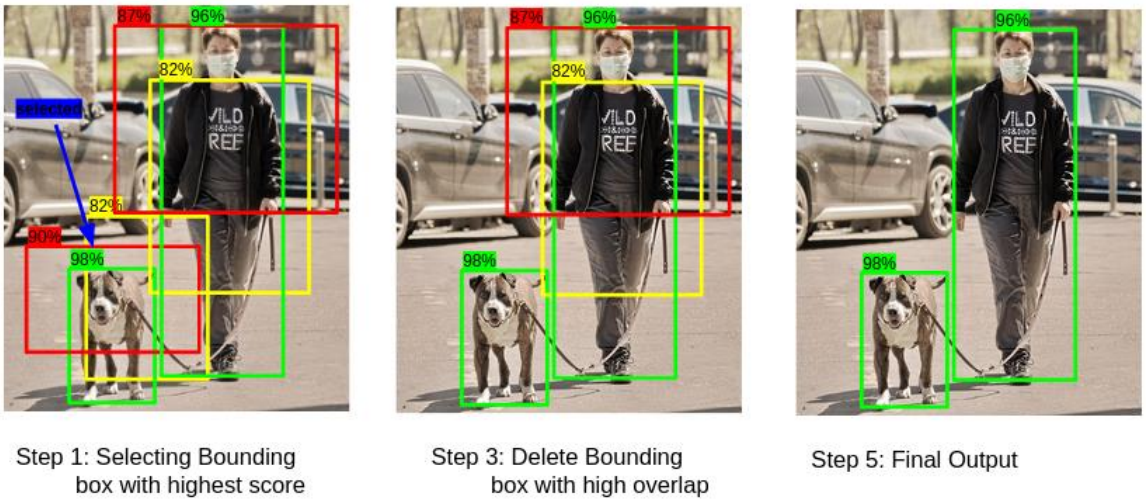
Obě tyto části jsou trénovány společně, což umožňuje jednotné end-to-end trénování modelu Faster R-CNN. Tento přístup výrazně zvyšuje rychlost a přesnost detekce objektů ve srovnání s předchozími modely. [20]

Vstupní obrázek je nejprve zpracován konvoluční neuronovou sítí, která extrahuje mapy vlastností. Tyto mapy vlastností obsahují důležité informace o různých vlastnostech obrázku, jako jsou barvy, tvary a textury. Mapy vlastností jsou poté zpracovány RPN. Každý návrh oblasti je převeden na pevně veliký vektor vlastností pomocí techniky zvané RoI Pooling (Region of Interest). Tato technika umožňuje zpracování návrhů oblastí různých velikostí a tvarů a zachovává prostorové informace z map vlastností. Vektory vlastností jsou následně zpracovány plně propojenými vrstvami, které provádějí klasifikaci a regresi ohraničujících rámečků. Klasifikační vrstva určuje třídu objektu v každém návrhu oblasti, zatímco regresní vrstva upravuje rozměry a polohu ohraničujícího rámečku pro přesnější lokalizaci objektu.

Na konci procesu se provádí non-maximum suppression (NMS) technika, která odstraňuje překrývající se detekce. NMS vybere nejpravděpodobnější detekci a poté odstraní všechny ostatní detekce, které se s ní významně překrývají (viz obrázek 7.).



Obrázek 6. Architektura procesu Faster R-CNN [17]



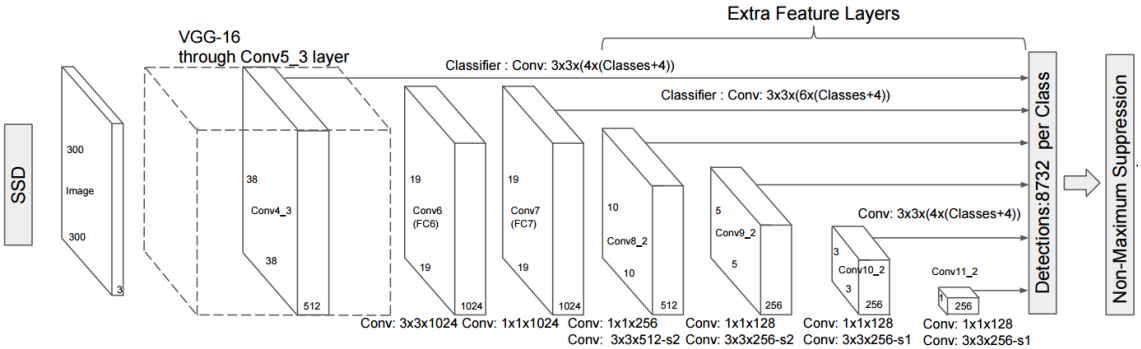
Obrázek 7. Proces non-maximum suppression [21]

Jedním z hlavních znaků Faster R-CNN je to, že RPN a detekční síť sdílejí své konvoluční vrstvy. To znamená, že konvoluční operace se vykonává jen jednou a její výsledek se poté společně využívá v obou částech sítě. To výrazně zrychluje celý proces detekce. [21]

2.5.4 SSD

SSD model (viz Obrázek 8.) přijímá vstupní obrázek jehož velikost se může lišit v závislosti na specifické variantě. Jeho hlavní předností je to, že dokáže detekovat objekty v různých měřítcích na obrázku v jediném průchodu (single shot). Poté provede extrakci vlastností pomocí konvoluční neuronové sítě CNN. Pro tuto část procesu je často používaná varianta modelu VGG16. Po každém bloku konvoluční vrstvy se vytvoří příznakové mapy různých velikostí a to umožňuje detekovat objekty různých velikostí. [22]

Detekční síť



Obrázek 8. Architektura SSD se vstupem 300x300x3 [22]

Každá mapa příznaků je rozdělena na mřížku, přičemž každá buňka mřížky odpovídá konkrétní oblasti na vstupním obrázku. Ke každé buňce mřížky je přiřazeno několik výchozích bounding boxů, také známých jako „anchor boxes“. Tyto výchozí boxy jsou různých aspektů (poměrů šířky a výšky) a měřítek a jsou navrženy tak, aby pokrývaly různé typy objektů, které mohou být na obrázku přítomny. každý výchozí box predikuje třídu objektu a odchylku od výchozí polohy a tvaru boxu. Tyto predikce jsou založeny na vlastnostech, které byly extrahovány z oblastí obrázku, který odpovídá buňce mřížky. Model ke konci používá techniky jako jsou non-maximum suppression (NMS) a Intersection over Union (IoU) pro zpracování těchto predikcí a eliminaci překrývajících se detekcí. [23]

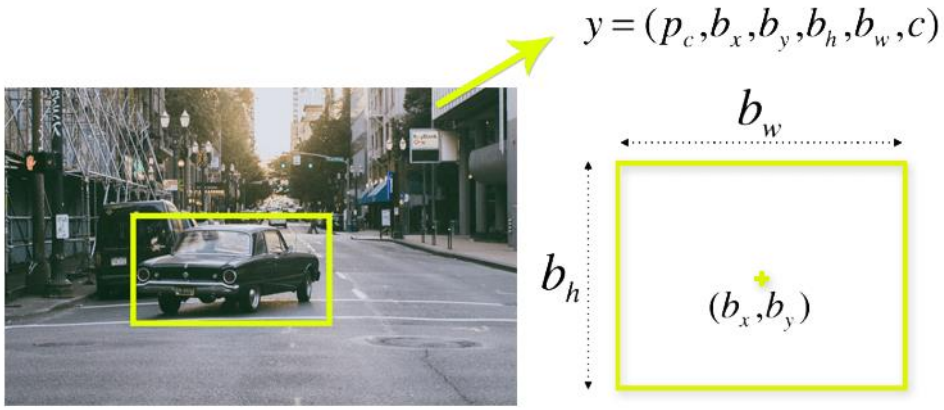
2.5.5 YOLO

Model YOLO (You Only Look Once) je algoritmus pro detekci objektů. Na rozdíl od výše uvedených algoritmů založených na oblastech, které identifikují a analyzují různé části obrázku nezávisle na sobě, YOLO přistupuje k obrázku jako k celku.

Jádrem tohoto modelu je jedna konvoluční neuronová síť, která provádí regresi pro předpověď bounding boxů objektů a jejich klasifikace. To znamená, že YOLO model se dívá na obrázek pouze jednou, odtud název „You Only Look Once“, aby identifikoval a klasifikoval objekty.

Konkrétně, model dělí obrázek do mřížky a každá buňka této mřížky je zodpovědná za předpověď určitého počtu bounding boxů a pravděpodobností tříd. Každý bounding box je definován pomocí čtyř parametrů (viz Obrázek 9.) [24]:

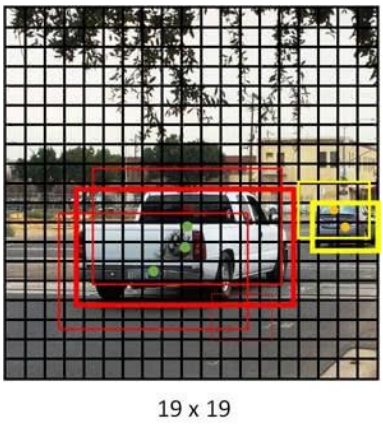
- Souřadnice středu bounding boxu (bx, by),
- Šířky (bw),
- Výšky (bh),
- Třída reprezentována písmenem (c).



Obrázek 9. Atributy bounding boxů [24]

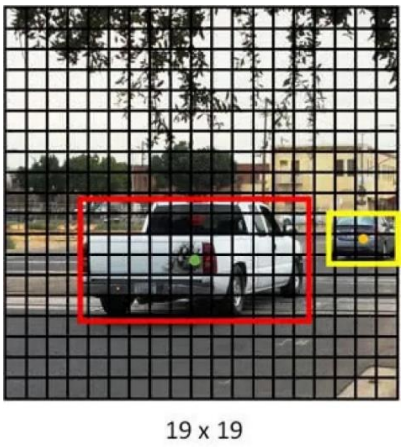
Obrázek je rozdělen do mřížky. Každá buňka v této mřížce je zodpovědná za předpověď bounding boxů a pravděpodobností tříd. Například vstupní obrázek rozdělí do mřížky o velikosti 19x19. [25]

Každá buňka v této mřížce má za úkol detekovat a identifikovat objekty, které se v jejím prostoru objevují. Určitý objekt je považován za součást konkrétní buňky mřížky, pouze pokud střed jeho bounding boxu spadá do prostoru této buňky. Tento postup má klíčový význam pro výpočet souřadnic bounding boxů. Středové souřadnice bounding boxu jsou vypočítány relativně k pozici buňky v mřížce, zatímco šířka a výška bounding boxu jsou vypočítány v poměru k celkové velikosti obrázku (viz Obrázek 10.). [26]



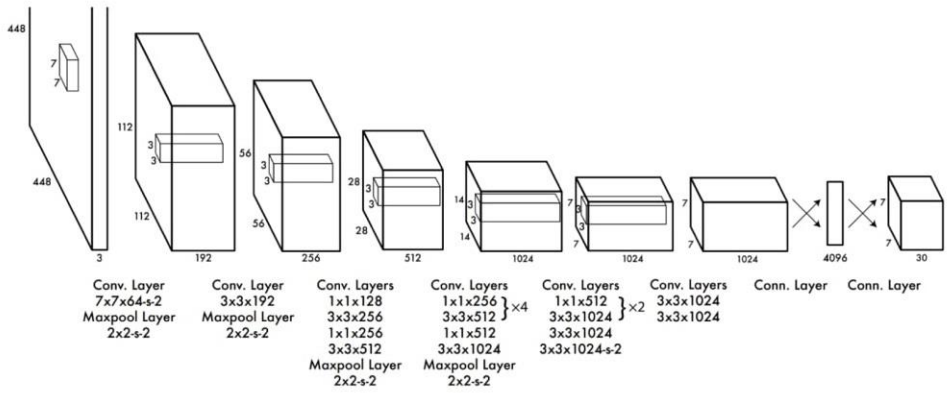
Obrázek 10. Vygenerované boxy [26]

Nakonec je provedena operace zvaná NMS, která se snaží odstranit překrývající se bounding boxy (viz Obrázek 11.). Pro každou třídu se bounding boxy seřadí podle skóre. Bounding box s nejvyšším skóre je označen za „detekováno“ a všechny ostatní boxy, které se s ním významně překrývají, jsou odstraněny. Tento proces se opakuje, dokud nezůstanou žádné bounding boxy, které se významně překrývají. Tímto způsobem se odstraní všechny redundantní detekce, takže každý objekt je detekován pouze jednou. Výstup modelu YOLO je sada bounding boxů a příslušných tříd objektů, které byly detekovány na obrázku. Každý bounding box je definován svými souřadnicemi (střed, šířka a výška) a každá detekce má přiřazenu třídu objektu a skóre, které odráží pravděpodobnost, že detekce je správná. [26]



Obrázek 11. Výsledek NMS [26]

Algoritmus YOLO používá jednoduchou hlubokou konvoluční neuronovou síť. Architektura modelu CNN, který tvoří páteř algoritmu YOLO (viz Obrázek 12.).



Obrázek 12. Architektura YOLO [27]

Základní architektura YOLO zahrnuje 24 konvolučních vrstev následovaných plně propojenou vrstvou. Tyto konvoluční vrstvy se skládají z různých filtrů pro extrakci příznaků z obrázků. Na konvoluční vrstvy navazuje plně propojená vrstva, která provádí konečné předpovědi tříd objektů a jejich bounding boxů.

Přestože původní YOLO model byl velmi úspěšný, bylo vytvořeno několik verzí modelu, které se snažily zlepšit jeho přesnost a efektivitu. [28]

YOLOv5

YOLOv5 představuje významný pokrok v oblasti detekce objektů, přinášejíc mnohá zlepšení, která zvyšují jak přesnost, tak efektivitu modelu. Model má vylepšenou architekturu, která je menší a rychlejší, ale stále poskytuje vynikající výkon. YOLOv5 se může přizpůsobit specifickým potřebám uživatele a to znamená, že lze jednoduše přeškolit na nových datech pro detekci širokého spektra objektů.

Model rovněž nabízí zlepšení v oblasti zpracování obrazu, včetně více detekcí objektů na jednom obrázku, rozšířeného sledování objektů a lepšího odhadování polohy. To zlepšuje schopnost modelu detekovat a sledovat objekty. I přes svou rychlost a efektivitu však YOLOv5 čelí výzvám, jako je zlepšení přesnosti detekce v obtížných podmínkách, například při nízkých světelných podmínkách nebo rušivém pozadí.

YOLOv5 je založen na konvolučních neuronových sítích CNN, které zpracovávají obrazová data a učí se rozpoznávat objekty na základě vzorců a charakteristik v datech. Tato schopnost umožňuje YOLOv5 provádět detekci objektů s vysokou rychlostí a přesností. YOLOv5 také zahrnuje řadu vylepšení, která zvyšují jeho výkon a efektivitu, včetně zlepšené architektury sítě a schopnosti adaptace na nové datové sady. To zlepšuje jeho flexibilitu a umožňuje detekci široké škály objektů. Navíc nové funkce, jako je schopnost detekovat více objektů na jednom obrázku a vylepšené sledování objektů, zvyšují užitečnost modelu v různých aplikacích. [29]

3 PROCES UČENÍ

V kontextu strojového učení a umělé inteligence je proces učení postup, kterým se strojový model přizpůsobí na základě dat a naučí se provádět určitou úlohu. Během procesu učení jsou parametry modelu upravovány na základě trénovací množiny dat tak, aby minimalizovaly chybu výstupu a maximalizovaly jeho přesnost.

Úspěšnost procesu učení detekce objektů závisí na mnoha vlivů. Výběr dat, jejich příprava a zpracování, výběr vhodné architektury a správná konfigurace parametrů modelu je klíčová pro dosažení vysoké přesnosti detekce objektů.

Proces učení má několik fází, které se obvykle opakují několikrát, dokud není dosaženo požadované přesnosti. [5]

3.1 Sběr dat

Získání dostatek kvalitních dat je dalším hlavním faktorem při trénování umělé inteligence. Bez kvalitních dat nelze dosáhnout vysoké přesnosti a účinnosti. Data mohou být získána různými způsoby, jako například manuálním sběrem nebo automatickým sběrem pomocí senzorů. Poté jsou data předzpracována tak, aby byla vhodná pro vstup do modelu. To zahrnuje úpravu velikosti dat, normalizaci a přizpůsobení formátu. [30]

3.2 Výběr architektury modelu

Architektura modelu představuje způsob, jakým jsou jednotlivé vrstvy propojeny a navrženy tak, aby umožnily modelu plnit určitou úlohu. Existuje mnoho různých typů architektur s vlastními přednostmi a nedostatky. Proto při výběru architektury modelu je třeba brát v úvahu úkol, který má model plnit, a charakteristiku dat, s nimiž model pracuje. Některé architektury jsou vhodné pro zpracování obrazových dat, jiné pro úkoly zpracování řeči. Důležitým faktorem je také velikost datové sady a dostupný výpočetní výkon. Některé architektury mohou vyžadovat speciální hardware, protože jsou velmi náročné na výpočetní kapacitu. [11]

3.3 Trénování modelu

Trénování modelu je proces, při kterém se model učí rozpoznávat vzorce a vztahy v datech a přizpůsobuje se úkolu, pro který byl navržen. Proces trénování zahrnuje prezentaci dat

modelu, odhadování parametrů modelu a optimalizaci těchto parametrů pomocí algoritmů učení.

Během trénování je důležité monitorovat výkon modelu na validačních datech, aby se minimalizovala chyba generalizace a předešlo se přetrénování. Existuje mnoho technik, které pomáhají minimalizovat riziko přetrénování, jako například regularizace, early stopping nebo použití dropout vrstev.

Proces trénování může být náročný na výpočetní výkon a zdroje dat. Proto se často používají předtrénované modely a transfer learning, kde jsou již naučené modely použity jako výchozí body pro nové úkoly. Také se často používají cloudové služby nebo GPU (Graphics Processing Unit) a TPU (Tensor Processing Unit) pro rychlejší trénování modelu. [30]

Vývoj umělé inteligence a proces učení jsou neustále v pohybu a výzkumníci stále hledají nové a lepší způsoby, jak je trénovat a vylepšovat její výkon. [8]

Přetrénování

Overfitting, česky přetrénování nebo přeučení, je problém, který se objevuje při trénování modelů. Když model přeučíme, dokáže sice velmi přesně reprezentovat trénovací data, ale má potíže s předvídáním výsledků na nových datech. [31]

Nedotrénování

Nedotrénování, anglicky underfitting může být stejně problematické jako přetrénování. Zatímco přetrénovaný model je příliš zaměřen na trénovací data a nedokáže generalizovat na nová data, nedotrénovaný model ani nezvládá přesně popsat trénovací data. Obě situace vedou k neefektivním modelům, které nedokážou správně předpovědět nebo klasifikovat nová data. [31]

3.4 Validace modelu

Validace modelu je nezbytnou součástí procesu učení. Je to proces testování a ověřování, zda je výstup modelu přesný a spolehlivý. Cílem validace je zajistit, že umělá inteligence je schopna správně klasifikovat a předpovídat výsledky pro data, která jí nebyla předložena během trénování. [31]

3.5 Testování a nasazení modelu

Testování a nasazení modelu jsou důležitými kroky, protože umožňují ověřit, zda model skutečně funguje a jak dobře se osvědčuje v reálných podmínkách. Pokud se během testování odhalí nějaké problémy, mohou být řešeny před nasazením modelu do ostrého provozu. Správné testování a nasazení jsou klíčové pro úspěch umělé inteligence a zajištění toho, aby byla přínosem pro uživatele. [2]

3.6 Standarty datasetů

Pro trénování modelů detekce objektů jsou používány standardizované datasety. Zde jsou některé z nejznámějších standardů datasetů pro detekci objektů:

1. **COCO (Common Objects in Context)**

   Common Objects in Context (COCO) je velmi rozsáhlá databáze obrázků používaná pro trénování strojového učení a hloubkového učení modelů v oblasti počítačového vidění.

   COCO je unikátní, protože pokrývá širokou škálu objektů v mnoha různých kontextech. Byla navržena tak, aby umožnila výzkum nových modelů pro detekci objektů, segmentaci a popisování scén.

   Databáze obsahuje více než 200 tisíc obrázků s anotacemi pro 80 kategorií objektů. Každý obrázek je anotován a obsahuje informace o poloze a typu každého objektu na obrázku. Díky těmto anotacím je COCO velmi užitečná pro trénování modelů, které se snaží rozpoznat a lokalizovat objekty na obrázcích. [32]

2. **Pascal VOC (Visual Object Classes)**

   Pascal VOC je další standard datasetu pro detekci objektů, který obsahuje více než 11 tisíc obrázků s anotacemi a více než 27 tisíc anotací objektů. Dataset zahrnuje 20 různých kategorií objektů, jako jsou osoby, auta, letadla, stromy, nábytek a další. I když je Pascal VOC dataset menší než některé jiné datasety, tak je často používán jako standard pro porovnávání výkonu algoritmů pro detekci objektů. Tento dataset je výjimečný díky vysoké kvalitě anotací objektů, což umožňuje přesnou a spolehlivou detekci objektů.

3.7 Předtrénované modely

Předtrénované modely jsou již natrénované modely, které lze použít pro různé účely. Pro přenos učení na nový dataset nebo novou úlohu. Toto přenosové učení může být využito, když není k dispozici dostatečný počet dat pro natrénování nového modelu, nebo když se chceme vyhnout dlouhému trénování modelu od začátku. Při použití předtrénovaných modelů je také důležité pochopit jejich omezení. Modely jsou trénovány na obrovských datasetech, které nemusí být vždy zcela relevantní pro konkrétní úlohu, na kterou je model používán. Nemáte kontrolu nad tím, jaké data byla použita k trénování modelu, což může vést k nepřesnostem. Modely, které byly trénovány před nějakou dobou, nemusí být nejnovější nebo nejúčinnější. Je třeba je pravidelně aktualizovat a přizpůsobovat novým výzkumným trendům a technologiím, které se velmi rychle vyvíjejí. [2]

4 EVALUACE

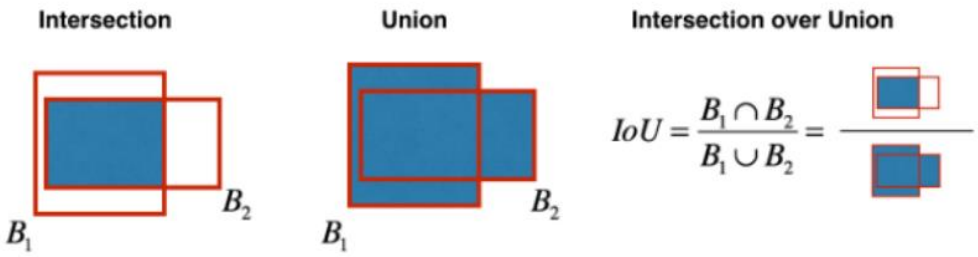
Evaluace, neboli vyhodnocení je součástí procesu strojového učení a umělé inteligence. Jedná se o proces měření, jak dobře model funguje. Toto vyhodnocení lze provádět různými způsoby v závislosti na typu úlohy, kterou se model snaží vyřešit. Tento proces je velmi důležitý, abychom mohli určit, jak dobře model funguje, a také nám umožňuje porovnávat různé modely a techniky. Je důležité si uvědomit, že žádná metrika není dokonalá a každá má své vlastní omezení. Proto je často užitečné použít více metrik k hodnocení výkonu modelu. [33]

4.1 Metriky evaluace

Metriky evaluace jsou nástroje používané k měření výkonnosti modelů strojového učení. Tyto metriky se liší v závislosti na typu úlohy strojového učení. [33]

4.1.1 Intersection over Union (IoU)

Intersection over Union je metrika používaná k hodnocení přesnosti detekce objektů. Měří podíl průniku mezi skutečným (Ground-truth) a predikovaným (Predicted) bounding boxem ku sjednocení těchto dvou bounding boxů (viz Obrázek 13.). [34]



Obrázek 13. Intersection over Union [35]

Hodnota IoU se pohybuje mezi 0 a 1, kde 0 znamená žádné překrytí a 1 znamená perfektní překrytí (viz Obrázek 14.). Při evaluaci modelů detekce objektů se často nastavuje prahová hodnota pro IoU (například 0,5) a pokud predikovaný bounding box splňuje tuto prahovou hodnotu v porovnání se skutečným bounding boxem (Ground-truth bounding box), je považován za správný (True Positive). [34]

Obrázek 14. Příklad výpočtu IoU pro různé bounding boxy [34]

4.1.2 Mean Average Precision (mAP)

Tato metrika kombinuje dva klíčové aspekty detekce. Jak přesné jsou jednotlivé detekce (precision) a kolik skutečných objektů bylo detekováno (recall). Precision je definována jako počet správných detekcí (True positives) děleno celkovým počtem detekcí (True positives + False positives) znázorněném na vzorci (1).

$$Presicion = \frac{True\ positive}{True\ positive + False\ positive} \tag{1}$$

Recall je metrika definována podle následujícího vzorce (2), udávající počet správných detekcí (True positives) děleno počtem skutečných objektů (True positives + False negatives).

$$Recall = \frac{True\ positive}{True\ positive + False\ negative} \tag{2}$$

Další důležitou metrikou je average precision (AP), česky průměrná přesnost, a vypočítá se jako plocha pod křivkou precision-recall. To se provádí rozdělením recall hodnot rovnoměrně na 11 částí: {0, 0.1, 0.2, ..., 0.9, 1} a výpočtem průměru přesnosti pro každý z těchto bodů. Matematicky se AP vyjadřuje pomocí vzorce (3). [36]

$$AP = \frac{1}{11} \sum_{Recall_i} Precision(Recall_i) = 1 \tag{3}$$

Mean average precision (mAP) je pak průměr AP napříč všemi třídami, který je vyjádřen vzorcem (4). [37]

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i \tag{4}$$

II. PRAKTICKÁ ČÁST

5 MASK DATASET

Při výběru datasetu je potřeba zvážit několik faktorů. Prvním z nich je velikost datasetu. Často platí, že zvýšení velikosti datasetu může zlepšit výkon modelu tím, že mu poskytne více informací pro učení, může to také zvýšit dobu trénování. Je tedy důležité najít rovnováhu mezi dostatečně velkým datasetem pro dobrou generalizaci modelu a schopností efektivně trénovat model v rozumném časovém rámci. Dalším faktorem je kvalita dat. Data by měla být čistá, relevantní a bez šumu. Kvalita dat také zahrnuje to, jak jsou data anotována. Posledním faktorem je rozmanitost dat. Dataset by měl obsahovat dostatečně rozmanitá data, aby model mohl pochopit dané aspekty problému, který potřebujeme řešit.

V rámci této bakalářské práce byly modely určené k detekci nasazení roušky a respirátoru trénovány na datasetu, který byl sestaven z dat pocházejících z různých zdrojů na platformě Kaggle. Záměrem bylo zajistit co nejširší spektrum a dostatečný objem dat, obsahujících různé typy obličejů s rouškou nebo respirátorem.

5.1 Kaggle

Kaggle je platforma pro datovou vědu a strojové učení, kterou vlastní společnost Google. Umožňuje uživatelům vyhledávat, publikovat a vytvářet modely ve webovém prostředí.

Kaggle, obsahuje více než 14 milionů anotovaných obrázků. Kaggle poskytuje dostatečnou velikost a rozmanitost dat pro naše potřeby a je široce používán v komunitě strojového učení pro účely klasifikace obrázků.

Datasety Kaggle jsou součástí této platformy, kde mohou uživatelé vyhledávat, zkoumat a stahovat veřejně dostupné datasety. Tyto datasety pochází z různých zdrojů a pokrývají širokou škálu témat, od zdravotnictví a biologie přes finance a ekonomiku až po text a obrázky pro trénování jazykových a vizuálních modelů. Ke každé datové sadě jsou připojena metadata, například popis, značky usnadňující orientaci a v mnoha případech ukázkový kód pro práci s daty. [38]

5.2 Příprava a nastavení datasetu

V této části práce je popsán proces přípravy a organizace dat. V úvodu je prezentován soubor prepare.py, jehož úkolem je připravit dataset obsahující obrázky a anotace pro účely detekce roušek. Následně je vysvětleno rozdělení dat na tři klasifikační třídy, které se týkají různých stavů nošení roušky, poté je popsán proces organizace dat do různých částí pro lepší

manipulaci a efektivitu při trénování modelu. V závěru textu je podrobně vysvětlen proces nastavení datasetu pro trénování detekčního modelu za použití knihovny PyTorch, včetně popisu vytvoření vlastní funkce datasetu a metody pro manipulaci s daty.

Příprava datasetu

Pomocí souboru prepare.py (dostupný na CD ve složce training) dataset rozdělí obsahujícího obrázky a anotace pro detekci roušky a respirátory. Dataset je uložen ve formátu ZIP, a obsahuje obrázky ve formátu PNG a anotace ve formátu XML. Kód rozdělí dataset na tréninkovou a validační množinu a převede anotace do formátu vhodného pro použití v detekčním modelu.

Soubor definuje tři klasifikační třídy, zaměřené na rozpoznávání nasazení roušky a respirátoru. Na tyto třídy byly poté modely přizpůsobeny. Klasifikačními třídami jsou:

- with mask,
- without mask,
- mask weared incorrect.

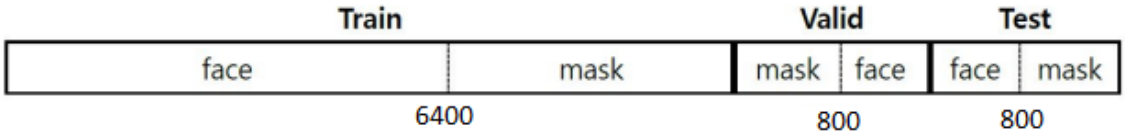
Pro lepší organizaci dat soubor dataset rozdělí na části, které zajišťují, že modely jsou trénovány a testovány na správných typech dat.

- **Annotations (Anotace)** - Tato část obsahuje informace o objektech nebo jiných entitách v obrazech, jako jsou jejich pozice, rozměry nebo kategorie. Tyto informace jsou obvykle vytvářeny ručně nebo s využitím nástrojů pro anotaci a jsou důležité pro trénování modelů pro úlohy jako je detekce objektů nebo segmentace.
- **Images (Obrázky)** - Obrázky, na kterých jsou trénovány a testovány modely strojového učení. Tyto obrázky mohou být reprezentovány v různých formátech.
- **Processed annotations (Zpracované anotace)** - Zpracované anotace zahrnují informace o anotacích, které byly upraveny nebo předzpracovány pro použití při trénování modelů. To může zahrnovat transformaci anotačních bodů do jiného souřadnicového systému nebo sloučení několika anotací do jedné.
- **Train (Trénovací data)** - Data, která jsou použita pro ověření úspěšnosti trénování modelu a pro ladění hyperparametrů. Tyto data by měla být podobná trénovacím

datům, ale neměla by být použita k trénování modelu, aby se minimalizovalo riziko přetrénování.

- **Valid (Validace)** - Používá se k testování výkonu modelů, když jsou nasazeny v produkčním prostředí. Tyto data by měla být odlišná od trénovacích a validačních dat, aby se minimalizovalo riziko přeučení a maximalizovala se obecná schopnost modelu generalizovat.

Data byla rozdělena do různých segmentů. Počátečních 6400 obrázků tvoří naši trénovací množinu, 800 obrázků je vyčleněno jako ověřovací množina a posledních 800 obrázků tvoří naši testovací množinu (viz Obrázek 15.).



Obrázek 15. Rozdělení datasetu

Nastavení datasetu

Dále bylo potřeba vytvořit vlastní funkce datasetu pro trénování detekčního modelu za použití knihovny PyTorch. Dataset je určen pro obrázky a odpovídající anotace ve formátu XML. Nejprve do souboru dataset.py (dostupný na CD ve složce training) byly importovány potřebné knihovny: **OS** pro interakci s operačním systémem, **torch** pro manipulaci s PyTorch, **PIL** pro načítání a manipulaci s obrázky, **xml.etree.ElementTree** pro parsování XML souborů a **torchvision.transforms** pro předzpracování obrázků. Dále byla vytvořena třída **CustomDataset**, která je odvozena od abstraktní třídy **torch.utils.data.Dataset**. Třída **CustomDataset** definuje metody pro inicializaci datasetu, parsování anotací, načítání dat z datasetu a předzpracování obrázků.

V konstruktoru **__init__** třídy **CustomDataset** je inicializován dataset, kde se nachází cesta k datům, typ detekčního modelu a transformace, které mají být na obrázky aplikovány.

Extrakci informací z anotací ve formátu XML zajišťuje funkce **parse_annotation**. Tato funkce načítá XML soubor, iteruje přes všechny objekty v XML a získává informace o třídách a bounding boxech.

Metoda **__getitem__** je zodpovědná za načítání dat z datasetu. Tato metoda načte obrázek a odpovídající anotaci, konvertuje obrázek do RGB, načte bounding boxy a štítky z anotace. Dále vytvoří tensor pro každý bounding box, štítek a následně vytvoří slovník target, který obsahuje všechny tyto informace. Pokud byly během inicializace definovány transformace, jsou tyto aplikovány na obrázek. K předzpracování obrázků slouží funkce **ssd_transforms** a **faster_rcnn_transforms**. Tyto funkce mění velikost obrázků, normalizuje a konvertuje obrázky na tensory.

Pro vrácení počtu dostupných obrázků v datasetu slouží jednoduchá funkce **__len__**.

Na začátek v každém ze souborů train_ssd.py, train_faster_rcnn.py (dostupné na CD ve složce training) byly importovány potřebné knihovny a moduly. Vlastní moduly jako **CustomDataset**, pro manipulaci s datasetem, **train_one_epoch** a **evaluate** ze souboru engine.py (dostupný na CD ve složce training) pro trénování a hodnocení modelu.

6 TRÉNOVÁNÍ DETEKČNÍCH MODELŮ

Praktická část projektu zahrnuje výběr detekčních modelů pro detekci nasazení roušky a respirátoru. Při výběru modelů je důležité zaměřit se na jejich použití pro detekci ve videu, která bude součástí výsledného modelu. Vybrané modely budou muset být především rychlé, aby byly schopny detekovat přítomnost roušky a respirátoru v reálném čase. Je tedy nutné pečlivě vybírat mezi dostupnými detekčními modely a zvolit ty, které nejlépe splňují požadavky.

V rámci této bakalářské práce byly vybrány celkem tři vhodné modely pro detekci nasazení roušky a respirátorů ve videu: YOLO (You Only Look Once), SSD (Single Shot MultiBox Detector) a Faster R-CNN (Region-based Convolutional Neural Network).

6.1 Použité technologie

Pro práci jsem si vybral jsem si vybral skriptovací jazyk Python verze 3.11.2. Hlavním důvodem bylo rozsáhlé množství knihoven, nástrojů a osobní zkušenost s jazykem. Kromě toho má Python velmi velkou komunitu vývojářů a uživatelů, což znamená, že existuje mnoho zdrojů, které mohou pomoci při řešení problémů.

Knihovna PyTorch

PyTorch je výkonná knihovna pro strojové učení a hluboké neuronové sítě. Poskytuje automatizaci výpočetních gradientů, umožňuje práci s modely a tenzory a akceleraci výpočtů na GPU. S jeho dynamickým grafem lze flexibilně vytvářet a optimalizovat výpočetní grafy, což usnadňuje vývoj složitých modelů.

Součástí PyTorch je také knihovna torchvision, která se zaměřuje na práci s daty pro počítačové vidění. Poskytuje předtrénované modely, transformace obrázků a standardní datové sady pro úlohy v oblasti počítačového vidění. Tím usnadňuje využití předem natrénovaných modelů a manipulaci s obrázky při trénování a evaluaci modelů. PyTorch a torchvision spolu tvoří mocný nástroj pro vývoj a nasazení pokročilých modelů strojového učení v oblasti počítačového vidění. [39]

6.2 Trénování modelů

Tato část se budeme zabývat procesem trénování vybraných modelů, konkrétně se zaměříme na modely pro detekci respirátorů a masek. Cílem je objasnit, jak se takové modely vytvářejí, trénují a testují.

Proces trénování modelů pro detekci objektů, jako jsou roušek a respirátory, je poměrně komplexní a zahrnuje řadu kroků. Začíná shromažďováním a předzpracováním dat, pokračuje volbou a trénováním vhodného modelu a končí jeho testováním, validací a nakonec implementací.

Klíčové body, které budeme v následujících kapitolách podrobněji prozkoumávat, zahrnují shromažďování a anotaci dat, předzpracování dat a trénování modelu, validaci a testování modelu.

V rámci práce je demonstrována implementace trénovací funkce **train_one_epoch**, která provádí trénování modelu po dobu jedné epochy.

Během každé epochy se model snaží „učit se“ na datech, což znamená, že se snaží optimalizovat své váhy tak, aby minimalizoval chybu mezi svými předpověďmi. Po každé epoše se obvykle vyhodnotí výkon modelu na validačních datech, které nebyly použity během trénování, aby se zjistilo, jak dobře model generalizuje na neznámá data. [11]

Počet epoch je jedním z hyperparametrů trénovacího procesu a musí být pečlivě nastaven. Pokud je počet epoch příliš nízký, model se nemusí dostatečně naučit a může podat slabý výkon. Pokud je počet epoch naopak příliš vysoký, model se může přeučit (overfit) [31].

Dalším důležitým hyperparametrem je „batch“, česky dávka, odkazuje na skupinu úloh nebo příkazů, které se vykonávají současně a postupně. Namísto toho, aby se každá úloha nebo příkaz vykonávala okamžitě, jsou seskupeny do jednoho balíčku, který je zpracován jako celek. [2]

Funkce **train_one_epoch** přijímá následující parametry:

- **model**: Hluboká neuronová síť, která bude trénována.
- **optimizer**: Optimalizátor je metoda použitá k aktualizaci vah modelu v průběhu trénování.
- **data_loader**: DataLoader je třída v PyTorch, která spravuje načítání trénovacích a testovacích dat. Umožňuje efektivní iteraci přes data v miniblokovém (batch) módu a také může automaticky míchat data před každou epochou, což je často žádoucí pro zabránění přeučení modelu.
- **device**: Tento parametr určuje, zda bude výpočet proveden na CPU nebo GPU.
- **epoch**: Tento parametr reprezentuje číslo aktuální epochy trénování.

- **print_freq**: Frekvence, s jakou se vypisují trénovací metriky.

Funkce nejprve nastaví model do trénovacího módu. Poté inicializuje **metric_logger**, což je nástroj pro sledování a zaznamenávání trénovacích metrik. V případě, že se jedná o první epochu, je nastaven **lr_scheduler**, aby se zabránilo náhlým skokům, které by mohly vést k nestabilitě.

Ve smyčce, která prochází všechny dávky dat v **data_loader**, jsou data zpracována a převedena na zařízení určené pro trénování. Následně model vrátí slovník ztrát **loss_dict** po provedení průchodu. Tyto ztráty jsou poté zredukovány a agregovány pro výpočet celkové ztráty.

Pokud je celková ztráta nekonečná, což by mohlo naznačovat problémy s trénováním, je trénování ukončeno. Jinak se provede zpětný průchod a optimalizátor provede krok, čímž aktualizuje váhy modelu.

Pokud byl nastaven **lr_scheduler**, je po každém kroku optimalizátoru proveden také krok **lr_scheduler**. Trénovací metriky jsou poté aktualizovány a zaznamenány pomocí **metric_logger**.

Po trénování jedné epochy je **metric_logger** vrácen jako výsledek, což umožňuje sledovat průběh trénování. Tento výsledek se uloží do textového dokumentu.

6.2.1 Faster R-CNN

Dále je načten předtrénovaný model Faster R-CNN s backbone architekturou ResNet50 a Feature Pyramid Network (FPN). Počet tříd je nastaven na tři, protože chceme detekovat osoby s rouškou, bez roušky a osoby s nesprávně nasazenou rouškou. Box predictor v modelu je poté nahrazen vlastním FastRCNNPredictorem s odpovídajícím počtem vstupních a výstupních vlastností.

Následně se vytvoří instance **CustomDataset** pro načtení a zpracování dat pro model. Pro tento model je DataLoader konfigurován tak, aby data nahrával v dávkách **batch** o velikosti 2 a promíchal **shuffle** pro lepší generalizaci modelu.

Následně je zjištěno, zda je dostupné CUDA pro výpočty na GPU. Pokud ano, model je převeden na GPU, nebo na CPU.

Poté jsou definovány parametry modelu pro optimalizaci, konkrétně jsou vybrány ty parametry, které vyžadují gradient (jsou upravitelné). Jako optimalizátor je použit Stochastic Gradient Descent (SGD) s danou learning rate (lr), momentem a weight decay. Pro úpravu learning rate během tréninku je použit **lr_scheduler** s daným krokem a gama parametrem.

Základní cyklus tréninku, který probíhá přes definovaný počet epoch (v tomto případě 100), je pro každou epochu zavolána funkce **train_one_epoch**, která model trénuje, a poté je zavolána funkce **evaluate**, která vyhodnocuje výkon modelu na validačním datasetu.

6.2.2 SSD

Na začátek v souboru train_ssd.py (dostupný na CD ve složce training) byly importovány potřebné knihovny a moduly. Vlastní moduly jako CustomDataset, pro manipulaci s datasetem, **train_one_epoch** a **evaluate** ze souboru engine.py (dostupný na CD ve složce training) pro trénování a hodnocení modelu.

Následně se vytvoří instance **CustomDataset** pro načtení a zpracování dat pro model. Pro tento model je **DataLoader** konfigurován tak, aby data nahrával v dávkách **batch** o velikosti 32 a promíchal **shuffle** pro lepší generalizaci modelu.

Model ssd300_vgg16 , který je předtrénovaný model detekce objektů. Počet tříd je nastaven na tři, protože chceme detekovat osoby s rouškou, bez roušky a osoby s nesprávně nasazenou rouškou. Poté model přesuneme na dostupné zařízení GPU, pokud je to možné, jinak CPU. Optimalizátor je nastaven na Stochastic Gradient Descent (SGD) s learning rate 0.01, momentum 0.9 a weight decay 0.0005. Learning rate scheduler je použit pro snižování learning rate každých 10 epoch o faktor 0.1.

Následuje hlavní tréninková smyčka, kde počet epoch je nastaven na 200. V každé epoše se provede jedno kolo tréninku pomocí funkce **train_one_epoch**. Aktualizuje se rychlost učení pomocí funkce **lr_scheduler.step().**

Na konci každé epochy je stav modelu uložen do souboru, takže trénink může být kdykoli přerušen a poté pokračovat od poslední uložené epochy.

6.2.3 YOLO

Při trénování modelu YOLOv5, byl vytvořen podadresář nazvaný „yolov5“ (dostupný na CD) s obsahem repozitáře YOLOv5, který je volně dostupný na oficiálním profilu společnosti Ultralytics na platformě GitHub.

Po úspěšném klonování repozitáře YOLOv5 bylo nutné nainstalovat potřebné knihovny. Tyto knihovny a balíčky Pythonu jsou použity v implementaci modelu YOLOv5. V repozitáři YOLOv5 je soubor s názvem „requirements.txt“, který obsahuje seznam všech potřebných knihoven a jejich příslušných verzí.

Další důležitou částí je konfigurace modelu. Konfigurační soubory jsou ve formátu YAML pro definování parametrů modelu a cest k datasetům.

Jedním z klíčových parametrů, který je třeba nastavit, je **NC**, což značí počet tříd, které model rozpoznává. V případě této práce, kdy detekujeme roušky a respirátor, je počet tříd nastaven na tři (with_mask, without_mask, mask_weared_incorrect).

Dále se v konfiguračním souboru nachází parametry jako **depth_multiple** a **width_multiple**, které určují velikost modelu. **Depth_multiple** ovlivňuje hloubku modelu tím, že určuje počet opakování některých vrstev, zatímco **width_multiple** ovlivňuje šířku modelu tím, že určuje počet neuronů v jednotlivých vrstvách.

Konfigurační soubor také obsahuje definici kotvících bodů anchors, které jsou klíčové pro proces detekce objektů. Anchors, neboli kotvící body určují různé velikosti a poměry stran bounding boxů, které model bude předpovídat.

Jakmile máme vše připraveno, můžeme začít s trénováním. YOLOv5 používá soubor konfigurace YAML pro nastavení různých parametrů trénování, včetně cesty k trénovacím a validačním datům, modelu architektury, počtu epoch a dalších. Soubor konfigurace může být upraven podle potřeb konkrétního projektu.

Pro trénování slouží soubor train.py (dostupný na CD ve složce yolov5). Tento soubor natrénuje model YOLOv5 na datech. Model v této práci bude trénován po dobu 100 epoch s batch 16 a velikostí obrázku 640.

Testování modelu pak probíhá na testovací sadě dat, která byla modelu dosud neznámá. Tato fáze nám poskytuje konečné hodnocení účinnosti modelu. V případě YOLOv5 lze evaluaci a testování provést pomocí souboru test.py a detect.py (dostupné na CD v souboru yolov5) pro testování detekce na nových obrázcích..

6.3 Vyhodnocení výsledků a srovnání modelů

Vyhodnocení tří vybraných detekčních modelů (uvedených výše v textu) si probereme v této části bakalářské práce. Každý model se liší v mnoha aspektech, včetně přesnosti, rychlosti

detekce a výpočetní náročnosti. Modely **Faster R-CNN**, **SSD** a **YOLOv5** budou nyní porovnány (viz Tabulka 1.).

V první řadě je důležité porozumět metrikám, na jejichž základě bude vyhodnocení provedeno. Precision je metrika, která ukazuje, jak velký podíl pozitivních případů detekovaných modelem je skutečně pozitivní. Recall potom ukazuje, jaký podíl skutečných pozitivních případů byl modelem správně identifikován. Metrika Mean Average Precision (mAP) je průměrná přesnost přes všechny třídy při daném prahu Intersection Over Union (IOU), zatímco mAP 0,5:0,95 je průměrná přesnost přes všechny třídy při různých prazích IOU od 0,5 do 0,95.

6.3.1 Faster R-CNN

Model předvedl v celkovém hodnocení (viz Tabulka 1.) solidní výsledky. Při prahové hodnotě 0,5:0,95 dosáhla průměrná přesnost (mAP) hodnoty 0,78. Čím vyšší je hodnota mAP, tím lépe model detekuje objekty při různých prazích detekce.

FPS 18,8 je hodnota, která ukazuje na schopnost modelu zpracovávat snímky za sekundu. Tento výsledek je důležitý pro použití ve videu, kde je rychlost zpracování kritickým faktorem.

Přesnost 0,94 a úplnost 0,96 jsou obě vynikající hodnoty a ukazují, že model je schopen s vysokou pravděpodobností přesně identifikovat a klasifikovat objekty ve vstupních datech. Zvláště úplnost 0,96 je indikátorem toho, že model dokáže detekovat téměř všechny objekty v datech.

Tabulka 1. Parametry 3 modelů

Testovaný model	Faster R-CNN	SSD	YOLOv5
Velikost vstupu	3x600x600	300x300	640x640
Počet epoch	100	200	100
Batch	2	32	16
Výsledek mAP@0,5:0,95	0,78	0,71	0,68
FPS	18,8	36,5	54,5
Precision	0,94	0,86	0,82
Recall	0,96	0,91	0,90

6.3.2 SDD

Výsledky vyhodnocení ukázaly (viz Tabulka 1.), že model dosáhl mAP (průměrná přesnost) hodnoty 0,71 při IOU (Intersection over Union) prahovém intervalu od 0,5 do 0,95. Tato hodnota mAP poskytuje přehled o celkové přesnosti modelu při detekci objektů v obrazech.

Dále byla změřena rychlost zpracování (FPS - Frames Per Second) modelu, která dosáhla hodnoty 36,5. V přesnosti detekce objektů dosáhl model hodnoty 0,86. Přesnost je míra, která vyjadřuje schopnost modelu správně klasifikovat objekty, tj. kolik z detekovaných objektů je skutečně relevantních. Úplnost modelu dosáhla hodnoty 0,91. Model dokáže detekovat alespoň 91 % všech relevantních objektů ve vstupních obrazech.

6.3.3 YOLO

Po ukončení trénování bylo provedeno vyhodnocení modelu (viz Tabulka 1.). Výsledkem byla průměrná přesnost detekce objektů (mAP) při prahování pravděpodobností od 0,5 do 0,95, která dosáhla hodnoty 68 %.

Další důležitou metrikou je rychlost zpracování snímků (FPS - Frames Per Second), která udává, kolik snímků model zvládne zpracovat za sekundu. V případě tohoto modelu dosáhla hodnoty 54,5 FPS.

Kromě toho byla vyhodnocena také přesnost a úplnost modelu. Přesnost se udává jako podíl správně detekovaných objektů k celkovému počtu detekcí a dosáhla hodnoty 0,82. Úplnost pak vyjadřuje podíl správně detekovaných objektů k celkovému počtu skutečných objektů a dosáhla hodnoty 0,90.

6.4 Srovnání modelů

V této části bakalářské práce se budeme zabývat srovnáním, výše uvedených, detekčních modelů, konkrétně modelů **Faster R-CNN**, **SSD** a **YOLOv5**.

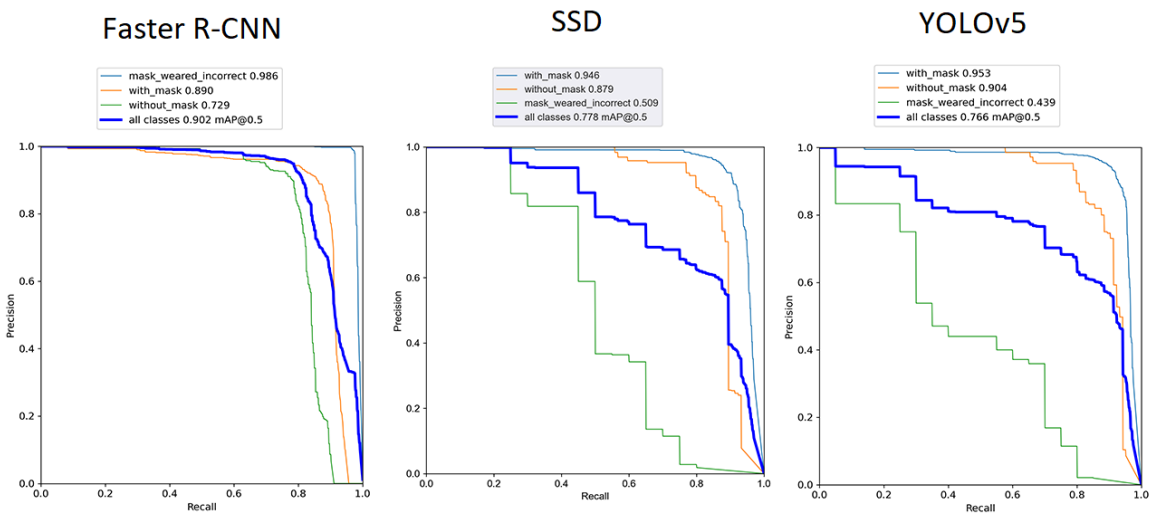
V grafu (viz Obrázek 16.) PR (Precision-Recall), který zobrazuje průměrnou přesnost napříč všemi třídami při prahové (IoU) hodnotě překrytí 0,5. Tato hodnota měří přesnost překrývání bounding boxů. To znamená, že aby byla predikce považována za správnou, musí

predikovaný (predicted) bounding box pokrývat alespoň 50% skutečného bounding boxu. Tento graf hodnotí dva hlavní aspekty detekčních modelů: přesnost a úplnost.

Ve srovnání výkonu modelů detekce objektů se model Faster R-CNN ukázal jako mírně dominantní, dosahující hodnoty 0.9 v metrice mAP@0.5, což znamená, že pokrývá 90% plochy pod křivkou v grafu PR (viz Obrázek 16.).

Model SSD vykazuje také silný výkon. Dosáhl mírně nižší hodnoty 0,78 a to značí pokrytí 78% plochy (viz Obrázek 16.).

Model YOLOv5 se umístil těsně za SSD s hodnotou 0,77 s pokrytím 77% plochy pod křivkou (viz Obrázek 16.). Přes tyto rozdíly, všechny tři modely vykazují vysoký výkon v detekci objektů, přičemž Faster R-CNN vykazuje mírně lepší výsledky.



Obrázek 16. Grafy Precision-Recall s prahovou hodnotou překrytí 0.5

Graf mAP@0.5:0.95 (viz Obrázek 17.) znázorňuje výkonnost modelu detekce objektů v průběhu jednotlivých epoch výpočtu.

Na ose x je zobrazen počet epoch, tedy kolikrát model prošel celým tréninkovým datasetem. Osa y značí metriku mAP@0.5:0.95, což je průměrná přesnost detekce objektů s různými úrovněmi prahové hodnoty překryvu (IoU) mezi predikovaným a skutečným umístěním objektu, konkrétně v rozmezí od 0.5 do 0.95. Vyšší hodnota znamená lepší výkonnost modelu. Pokud hodnota mAP@0.5:0.95 na grafu (viz Obrázek 17.) stoupá s počtem epoch, model se v průběhu tréninku zlepšuje. Pokud hodnota klesá, model se zhoršuje.

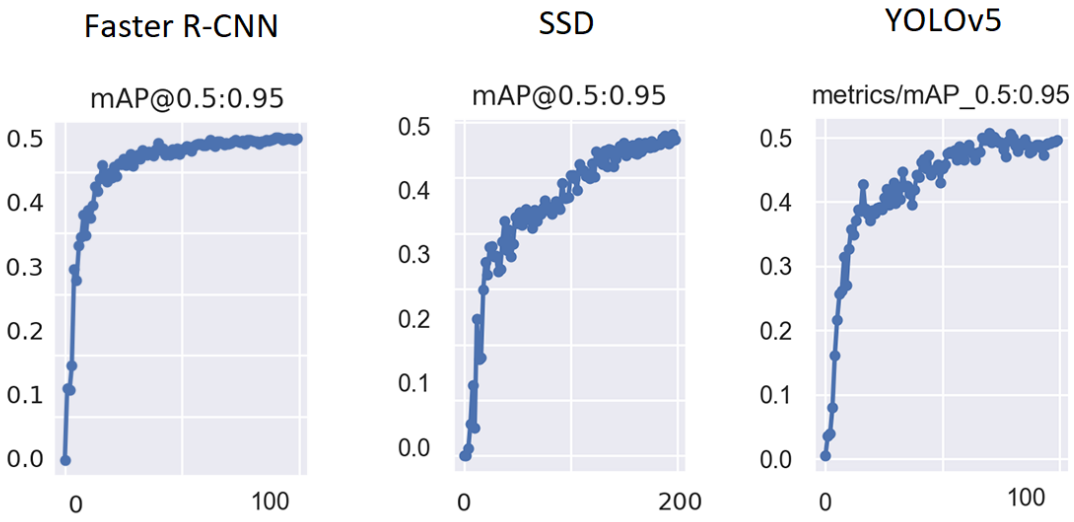
V tabulce srovnání (viz Tabulka 2.) model Faster R-CNN dosahuje nejvyšší precision 0,94 a recall 0,96, což ukazuje, že tento model je nejpřesnější a také nejlépe detekuje všechny relevantní případy v datovém setu. Tento model dosahuje hodnoty mAP 0,90 při prahu 0,5 a 0,78 při prahu 0,5:0,95 (viz Obrázek 17.). Nicméně, jeho rychlost snímků za sekundu (FPS) je nejnižší z těchto modelů, konkrétně 18,8 FPS.

SSD model nabízí vyváženější kompromis mezi přesností a rychlostí, s hodnotami 0,86 a 0,91. Jeho mAP skóre je 0,78 při prahu 0,5 a 0,71 při prahu 0,5:0,95 (viz Obrázek 17.). I přesto, že jeho účinnost je nižší než u Faster R-CNN, tento model je téměř dvojnásobně rychlejší s hodnotou 36,5 FPS.

V poslední řadě model YOLOv5. Ačkoli se řadí na třetím místě, dosahuje stále slušných výsledků s hodnotami přesností 0,82 a úplnost 0,90. Jeho mAP skóre je 0,77 při prahu 0,5 a 0,68 při prahu 0,5:0,95 (viz Obrázek 17.). Avšak, tento model je nejrychlejší ze všech tří modelů s hodnotou 54,5 FPS a proto je ideální volbou pro využití, kde rychlost hraje klíčovou roli.

Tabulka 2. Výsledné metriky 3 modelů

Detekční modely	**Přesnost** (Presicion)	**Úplnost** (Recall)	**mAP** (0.5)	**mAP** (0.5:095)	**FPS**
Faster R-CNN	0,94	0,96	0,90	0,78	18,8
SSD	0,86	0,91	0,78	0,71	36,5
YOLOv5	0,82	0,90	0,77	0,68	54,5



Obrázek 17. Grafy mAP@0.5:0.95

7 TESTOVÁNÍ NEJLEPŠÍHO MODELU VE VIDEU

V této kapitole se podrobně zabýváme testováním vybraného modelu YOLOv5. Model byl vybrán na základě rychlosti a uspokojivých hodnot přesnosti a úplnosti detekce. Testování proběhlo na videu, kde byla hodnocena schopnost modelu detekovat a klasifikovat osoby s rouškou, bez roušky a s nesprávně nasazenou rouškou.

7.1 Výběr nejlepšího detekčního modelu

V rámci kapitoly 6.4 Srovnání modelů, byly jednotlivé modely podrobně analyzovány a porovnány. Každý z modelů byl pečlivě posouzen na základě kritérií jako je rychlost detekce, přesnost, úplnost a mAP hodnoty. Srovnání bylo provedeno v kontextu konkrétních požadavků a cílů této bakalářské práce a proto byl vybrán YOLOv5 jako nejvhodnější model pro detekci respirátorů/masek ve videu. Ačkoliv je přesnost modelu nižší než u druhých dvou modelů, stále dosahuje relativně vysokých hodnot přesnosti pro reálné použití a díky své výrazně vyšší detekční rychlosti je nejvhodnějším modelem pro real time zpracování videa.

7.2 Testování modelu

V této sekci byl otestován model YOLOv5 pro detekci roušky a respirátoru ve videu. Nejprve byla připravena testovací data. V případě této práce je testovací datový soubor video s názvem "vid1.mp4" (dostupné na CD ve složce data/testovani) o rozlišení 1280x948 a rychlost 60 snímků za sekundu.

Poté, co byla data připravena, model YOLOv5 bylo potřeba inicializovat a nastavit pro testování. Výhodou tohoto detekčního modelu je, že přímo zpracovává video soubory a to eliminuje potřebu předzpracování dat. Model je již natrénovaný, proto další krok zahrnuje načtení modelu a jeho konfigurace. Po načtení je model aplikován na testovací data.

Obrázek 18. Snímky z testovacího videa (vid1.mp4)

7.3 Výsledky a diskuze

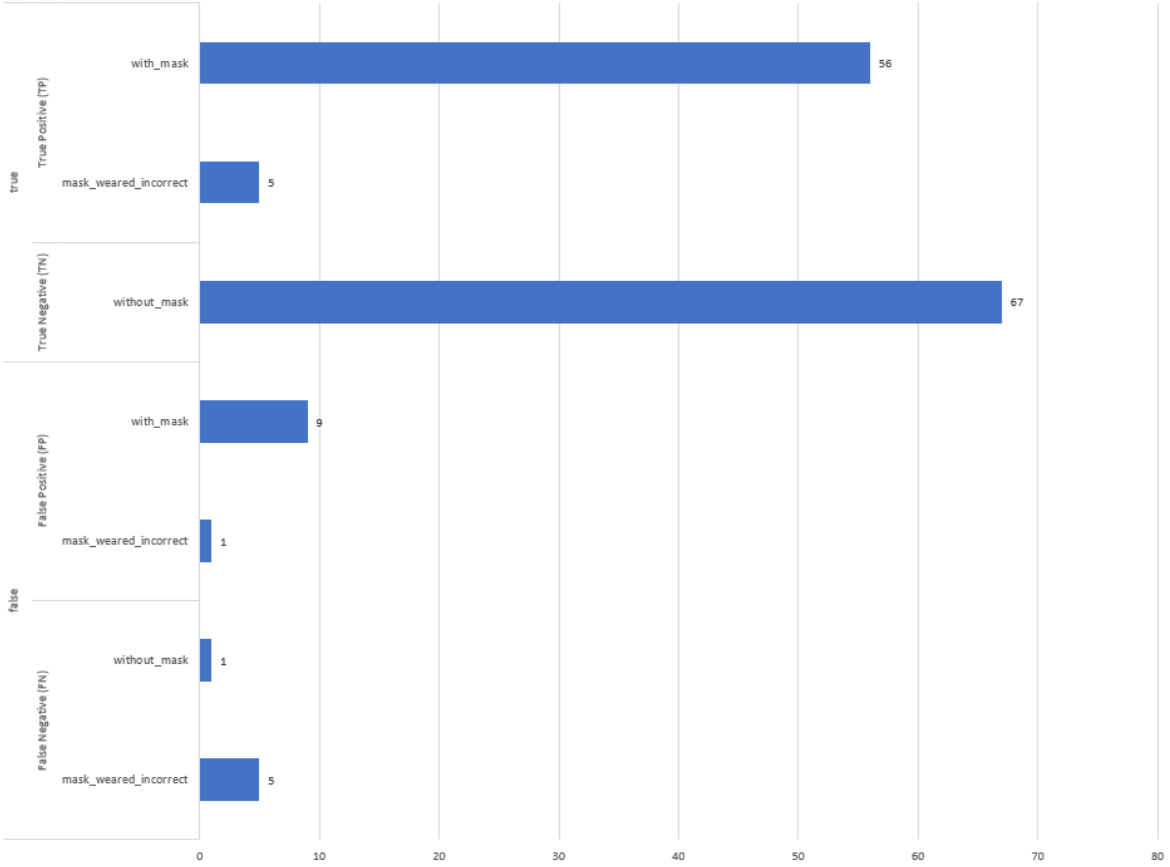
Když je model aplikován na testovací data, následuje fáze vyhodnocení výsledků. To znamená procházet seznam detekcí, které model identifikoval, a porovnávat je s očekávanými výsledky (Expected results). Očekávané výsledky se nachází na CD (ve složce data/testovani).

Hodnocení výkonnost tohoto modelu bylo rozděleno na kvantitativně a kvalitativně.

Kvantitativně (viz Obrázek 22.) byl měřen výkon modelu v detekci tří různých tříd: osoba s rouškou (with_mask), osoba bez roušky (without_mask) a osoba s nesprávně nasazenou rouškou (mask_weared_incorrect). Každé detekci byla přiřazena jedna z následujících kategorií v závislosti na tom, zda model správně, nebo nesprávně klasifikoval daný snímek:

- **True Positive (TP)** - model správně identifikoval situaci.
- **True Negative (TN)** - model správně identifikoval absenci situace.
- **False Positive (FP)** - model nesprávně identifikoval situaci, která tam nebyla.
- **False Negative (FN)** - model nesprávně identifikoval absenci situace, která tam byla.

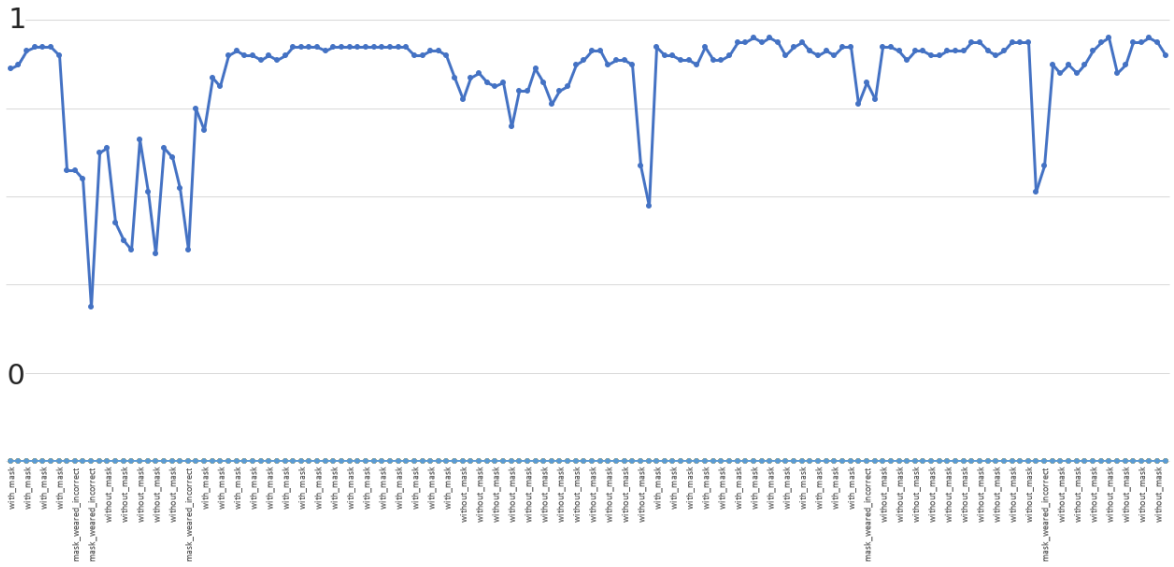
Celkově bylo detekováno 144 obličejů (viz Obrázek 19.), z nichž 56 bylo správně identifikováno jako (with_mask), 67 jako (without_mask) a 5 jako (mask_weared_incorrect). Na druhé straně, model udělal celkem 10 chybných detekcí, 9 pro (with_mask), 1 pro (mask_weared_incorrect). Model také nesprávně označil 6 snímků jako „False Negative“, 1 pro (without_mask) a 5 pro (mask_weared_incorrect).



Obrázek 19. Porovnání detekčního výkonu modelu pro různé třídy

Kvalitativně bylo zjištěno, že model YOLOv5 se vyznačuje vysokou přesností při detekci osob s rouškou a osob bez roušky, ale měl potíže při detekci osob s nesprávně nasazenou rouškou. To naznačuje, že model je schopen efektivně rozlišit mezi obličeji s rouškou a bez roušky, ale může mít problémy se zvláštním případem nesprávně nasazené roušky. Na grafu (viz Obrázek 23.) lze vidět výsledek v sekvenci 10 snímků za sekundu (10 FPS), který je vyjádřen jako číslo mezi 0 a 1. Číslo blízké 0 znamená, že model si je téměř jistý, že daný objekt nespadá do dané třídy. Naopak číslo blízké 1 znamená, že model si je téměř jistý, že daný objekt spadá do dané tříd.

Konkrétně graf (viz Obrázek 20.) zobrazuje vztah mezi detekovanou třídou ve snímku a mírou jistoty, kterou má detekovaný model v této detekci. Na ose x jsou zobrazeny tři třídy (with_mask, without_mask, mask_weared_incorrect), které mohou být detekovány ve snímku, zatímco na ose y je vyjádřena míra jistoty, kterou model přiřazuje každé třídě.



Obrázek 20. Časový průběh pravděpodobnosti detekce tříd

K vyhodnocení celkové výkonnosti modelu byla využita metriku F1 skóre, která je harmonický průměr přesnosti a návratnosti a dává nám jednotné skóre, které zohledňuje obě tyto metriky. F1 skóre se vypočítá jako 2 * (Precision * Recall) / (Precision + Recall). [40]

Výsledek tohoto skóre pro každou třídu je následující:

- with_mask: F1 = 0.91
- without_mask: F1 = 0.94
- mask_weared_incorrect: F1 = 0.57

Na základě těchto výsledků můžeme konstatovat, že YOLOv5 model je schopný účinně detekovat nošení roušky na sekvenci snímků. Vysoké F1 skóre pro detekci (with_mask) a (without_mask) naznačuje, že model je silný v těchto kategoriích. Nicméně, pro (mask_weared_incorrect), model ukázal prostor pro zlepšení. Další práce by mohly zahrnovat rozšíření datasetu o více obrázků, na kterých jsou masky nošeny nesprávně. Tato strategie by měla umožnit modelu lépe se naučit rozpoznávat a správně klasifikovat tuto situaci. Tento přístup

je v souladu s obecnou praxí strojového učení, kde rozšiřování datasetu o více vzorků cílové kategorie může pomoci modelu lépe se naučit identifikovat tuto kategorii. Souhrnně řečeno, model YOLOv5 dosahuje obecně dobrých výsledků v detekci masek, existuje prostor pro zlepšení, zejména v detekci nesprávně nošených masek.

ZÁVĚR

Tato bakalářská práce je zaměřena na detekci nošení roušek a respirátorů ve videu. Práce byla rozdělena do dvou částí: teoretická a praktická.

Teoretická část zkoumá koncepty jako strojové učení, hluboké konvoluční neuronové sítě, proces učení a evaluace. Tato část sloužila jako základ pro další praxi a poskytla rámec pro výzkum v oblasti detekce objektu.

Testování prokázalo, že model YOLOv5 je obecně účinný v detekci nošení roušek a respirátorů. Přestože byl model úspěšný ve většině případů, bylo zjištěno, že existuje prostor pro zlepšení, zejména v případě detekce nesprávně nasazených masek. Budoucí práce by mohly zahrnovat rozšíření datasetu o více obrázků, na kterých jsou roušky nošeny nesprávně, což by mohlo vést k zlepšení výsledků modelu.

Z hlediska potenciálního využití výsledného modelu v soukromých klinikách a menších zdravotnických zařízeních, může tato práce představovat základ pro implementaci automatického systému pro monitorování dodržování opatření spojených s nošením roušek a respirátorů. Tento systém by mohl pomoci zdravotnickým pracovníkům snížit riziko šíření infekčních nemocí a zároveň by mohl přispět ke zlepšení kvality a bezpečnosti poskytované péče.

Celkově tato práce představuje komplexní pohled na detekci roušek a respirátorů ve videu. Model YOLOv5 ukázal, jak může být strojové učení použito k řešení konkrétních problémů a jak lze využít různých metrik pro hodnocení účinnosti modelu.

SEZNAM POUŽITÉ LITERATURY

[1] MITCHELL, Tom M. Machine learning. Boston: McGraw-Hill, 1997. ISBN 0070428077.

[2] ROSEBROCK, Adrian. Deep learning for computer vision with Python: Practitioner Bundle. PyImageSearch, 2017. ISBN 978-1-986723817.

[3] Classification, Object Detection and Image Segmentation - Qualcomm Developer Network. Mobile Development - Qualcomm Developer Network [online]. Copyright ©2023 Qualcomm Technologies, Inc. and [cit. 2021]. Dostupné z: https://developer.qualcomm.com/software/qualcomm-neural-processing-sdk/learning-resources/image-segmentation-deeplab-neural-processing-sdk/classification-object-detection-segmentation.

[4] Difference Between Semantic and Instance Segmentation. Roboflow Blog [online]. Copyright © [cit. 13.10.2022]. Dostupné z: https://blog.roboflow.com/difference-semantic-segmentation-instance-segmentation.

[5] LANTZ, Brett. Machine learning with R. Birmingham, UK: Pack Publishing, c2013. Community experience distilled. ISBN 978-1-78216-214-8.

[6] Object Detection and Tracking in 2020 | by Borijan Georgievski | Netcetera Tech Blog | Medium. Medium – Where good ideas find you. [online]. Dostupné z: https://blog.netcetera.com/object-detection-and-tracking-in-2020-f10fb6ff9af3.

[7] Semantic Segmentation: The easiest possible implementation in code! | by Garima Nishad | Analytics Vidhya | Medium. Medium – Where good ideas find you. [online]. Dostupné z: https://medium.com/analytics-vidhya/semantic-segmentation-the-easiest-possible-implementation-in-code-193bf27b86b8.

[8] Object Detection: Models, Architectures & Tutorial [2023]. V7 - AI Data Platform for Computer Vision [online]. Copyright ©V7Labs Dostupné z: https://www.v7labs.com/blog/object-detection-guide.

[9] What is the difference between semantic segmentation, object detection and instance segmentation? | stackexchange [online]. [cit. 15.05.2019]. Dostupné z: https://datascience.stackexchange.com/questions/52015/what-is-the-difference-between-semantic-segmentation-object-detection-and-insta?ref=blog.roboflow.com.

[10] ADHINATA, F. D., D. P. RAKHMADANI, M. WIBOWO a A. JAYADI. A Deep Learning Using DenseNet201 to Detect Masked or Non-masked Face. 9(1). JUITA Jurnal

Informatika, 2021. ISBN 115-121.

[11] GOODFELLOW, Ian, Yoshua BENGIO a Aaron COURVILLE. Deep learning: Image-Net bundle. Adaptive computation and machine learning. Cambridge, Massachussetts

; London: The MIT Press, 2016. ISBN 9780262035613.

[12] The Essential Guide to Neural Network Architectures. Pragati Baheti | V7 - AI Data Platform for Computer Vision [online]. Copyright ©V7Labs [cit. 08.07.2021]. Dostupné z: https://www.v7labs.com/blog/neural-network-architectures-guide.

[13] Convolutional Neural Networks, Explained. Mayank Mishra | Towards Data Science [online]. Medium - Towards Data Science [cit. 26.08.2020]. Dostupné z: https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939.

[14] Transfer Learning | Pretrained Models in Deep Learning. Analytics Vidhya | Learn everything about Data Science, Artificial Intelligence and Web 3.0 [online]. Dostupné z: https://www.analyticsvidhya.com/blog/2017/06/transfer-learning-the-art-of-fine-tuning-a-pre-trained-model.

[15] Region Proposal Network (RPN) : A Complete Guide. ListenData [online]. Dostupné z: https://www.listendata.com/2022/06/region-proposal-network.html.

[16] Bibliometric Analysis of One-stage and Two-stage Object

Detection : Aditya Lohia, Kalyani Dhananjay Kadam, Rahul Raghvendra Joshi, Dr. Anupkumar M. Bongale | University of Nebraska [online]. Dostupné z: https://digitalcommons.unl.edu/cgi/viewcontent.cgi?article=9123&context=libphilprac.

[17] R-CNN vs Fast R-CNN vs Faster R-CNN - A Comparative Guide. Analytics India Magazine | Artificial Intelligence, Data Science, Machine Learning [online]. Copyright © Analytics India Magazine Pvt Ltd [cit. September 10, 2021]. Dostupné z: https://analyticsindiamag.com/r-cnn-vs-fast-r-cnn-vs-faster-r-cnn-a-comparative-guide.

[18] Faster R-CNN Explained for Object Detection Tasks | Paperspace Blog. Paperspace Blog [online]. Copyright © Copyright by [cit. 2020]. Dostupné z: https://blog.paperspace.com/faster-r-cnn-explained-object-detection.

[19] Selective Search for Object Recognition - Hannan Syed | McGill University. [online]. [cit. 2016]. Dostupné z: https://www.cs.mcgill.ca/~hsyed2/selectivesearch/Object-Recognition/Selective_Search_for_Object_Recognition.pdf.

[20] Region Proposal Network (RPN) : A Complete Guide. ListenData [online]. Dostupné z: https://arxiv.org/pdf/1506.01497.pdf.

[21] Selecting the Right Bounding Box Using Non-Max Suppression (with implementation) | Aishwarya Singh | Learn everything about Data Science, Artificial Intelligence and Web 3.0 [online]. Dostupné z: https://www.analyticsvidhya.com/blog/2020/08/selecting-the-right-bounding-box-using-non-max-suppression-with-implementation.

[22] Understanding SSD MultiBox - Real-Time Object Detection In Deep Learning | Eddie Forson | Towards Data Science | Nov 18, 2017 [online]. Dostupné z: https://www.analyticsvidhya.com/blog/2020/08/selecting-the-right-bounding-box-using-non-max-suppression-with-implementation.

[23] Single shot detector (ssd) - Object Detection In Deep Learning | leonardoaraujosantos | [online]. Dostupné z: https://leonardoaraujosantos.gitbook.io/artificial-inteligence/machine_learning/deep_learning/single-shot-detectors/ssd.

[24] Introduction to YOLO Algorithm for Object Detection | Engineering Education (EngEd) Program | Section. Globally distributed container hosting | Section [online]. Copyright © 2022 Section [cit. 2021]. Dostupné z: https://www.section.io/engineering-education/introduction-to-yolo-algorithm-for-object-detection.

[25] You Only Look Once (YOLO) - NeuralCeption - [cit. 2022]. Dostupné z: https://www.neuralception.com/objectdetection-yolo.

[26] Gentle guide on how YOLO Object Localization works with Keras (Part 2) - Heartbeat - [cit. 2018]. Dostupné z: https://heartbeat.comet.ml/gentle-guide-on-how-yolo-object-localization-works-with-keras-part-2-65fe59ac12d.

[27] YOLO — You only look once, real time object detection explained| datacamp [online]. Copyright © [cit. September 2022]. Dostupné z: https://towardsdatascience.com/yolo-you-only-look-once-real-time-object-detection-explained-492dc9230006.

[28] YOLO Object Detection Explained | Manish Chablani | Towards Data Science [online]. Copyright © [cit. August 2017]. Dostupné z: https://www.datacamp.com/blog/yolo-object-detection-explained.

[29] A Guide to the YOLO Family of Computer Vision Models. Data Phoenix [online]. Data Phoenix [cit. 12.01.2023]. Dostupné z: https://dataphoenix.info/a-guide-to-the-yolo-family-of-computer-vision-models.

[30] Training Data Quality: Why It Matters in Machine Learning. V7 - AI Data Platform for Computer Vision [online]. V7- Labs [cit. 2022]. Dostupné z: https://www.v7labs.com/blog/quality-training-data-for-machine-learning-guide.

[31] Overfitting and underfitting in machine learning | SuperAnnotate. The ultimate training data platform for AI | SuperAnnotate [online]. SuperAnnotate AI, Inc. All rights reserved. [cit. 2022]. Dostupné z: https://www.superannotate.com/blog/overfitting-and-underfitting-in-machine-learning.

[32] COCO - Common Objects in Context. COCO - Common Objects in Context [online]. Dostupné z: https://cocodataset.org.

[33] Object Detection Metrics With Worked Example - Kiprono Elijah Koech | Towards Data Science [online]. Dostupné z: https://towardsdatascience.com/on-object-detection-metrics-with-worked-example-216f173ed31e.

[34] Intersection over Union (IoU) for object detection - PyImageSearch. PyImageSearch - You can master Computer Vision, Deep Learning, and OpenCV. [online]. [cit. 2016]. Dostupné z: https://pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection.

[35] YOLO — You Only Look Once – Manishgupta | Towards Data Science. [online]. [cit. 2020]. Dostupné z: https://pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection.

[36] mAP (mean Average Precision) might confuse you! Shivy Yohanandan - Towards Data Science [online]. Deval Shah [cit. 2020]. Dostupné z: https://www.v7labs.com/blog/mean-average-precision.

[37] Mean Average Precision (mAP) Explained: Everything You Need to Know. V7 - AI Data Platform for Computer Vision [online]. Deval Shah [cit. 2022]. Dostupné z: https://www.v7labs.com/blog/mean-average-precision.

[38] Kaggle: Your Machine Learning and Data Science Community. Kaggle: Your Machine Learning and Data Science Community [online]. Dostupné z: https://www.kaggle.com.

[39] Understanding Torchvision Functionalities (for PyTorch)-Part 1 | Lars Nielsen | The Startup [online]. [cit. 2021]. Dostupné z: https://medium.com/swlh/understanding-torchvision-functionalities-for-pytorch-391273299dc9.

[40] Classification Accuracy is Not Enough: More Performance Measures You Can Use | Jason Brownlee | [2019]. Dostupné z: https://machinelearningmastery.com/classification-accuracy-is-not-enough-more-performance-measures-you-can-use.

[41] Object Detection in 2023: The Definitive Guide - viso.ai. Viso Suite - No-Code Computer Vision Platform - viso.ai [online]. Copyright © 2023 viso.ai [cit. 2023]. Dostupné z: https://viso.ai/deep-learning/object-detection.

[42] Understanding of Convolutional Neural Network (CNN) — Deep Learning | by Prabhu | Medium. Medium – Where good ideas find you. [online]. Dostupné z: https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148.

[43] ROSEBROCK, Adrian. Deep learning for computer vision with Python: Starter bundle. PyImageSearch, 2017. ISBN 9781986538138.

[44] ROSEBROCK, Adrian. Deep learning for computer vision with Python: ImageNet bundle. PyImageSearch, 2017. ISBN 978-1-986723848.

[45] SHAOHUI, Lin, Cai LING, Lin XIANMING a Ji RONGRONG. Masked face detection via a modified LeNet. Volume 218. Neurocomputing, 2016. ISBN 197–202.

[46] How single-shot detector (SSD) works? | ArcGIS API for Python. ArcGIS Developers [online]. Copyright © [cit. 06.03.2023]. Dostupné z: https://developers.arcgis.com/python/guide/how-ssd-works.

[47] Image classification and object detection. Ambolt AI - konsulent og udviklingshus [online]. Copyright © Ambolt 202 [cit. 2020]. Dostupné z: https://ambolt.io/en/image-classification-and-object-detection.

[48] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press. http://www.deeplearningbook.org.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

CNN	Convolution Neural Network
RPN	Region Proposal Network
R-CNN	Regions with Convolution Neural Network
SSD	Single Shot Detector
SVM	Support Vector Machines
ROI	Region Of Interest
NMS	Non-Maximum Suppression
IoU	Intersection over Union
YOLO	You Only Look Once
GPU	Graphics Processing Unit
TPU	Tensor Processing Unit
COCO	Common Objects in Context
AP	Average Precision
mAP	Mean average precision
ZIP	Zipped File
XML	Extensible Markup Language
PNG	Portable Network Graphics
VGG	Visual Geometry Group
RGB	Red Green Blue
SGD	Stochastic Gradient Descent
YAML	YAML Ain't Markup Language
FPS	Frames Per Second
DNN	Deep Neural Network
FPN	Feature Pyramid Network

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I: OBSAH CD

Přiložené CD obsahuje:

➢ Adresář **data**

- Adresář **detected**
    - Soubor **vid2.mp4**
    - Soubor **results_detected.ods**
- Adresář **testing**
    - Soubor **vid1.mp4**

➢ Adresář **models**

- Soubor **mask_faster_rcnn.pt**
- Soubor **mask_ssd.pt**
- Soubor **mask_yolov5.pt**

➢ Adresář **results**

- Soubor **test-results.png**
- Soubor **withmask.jpg**
- Soubor **withoutmask.jpg**
- Soubor **maskwearedincorrect.jpg**

➢ Adresář **training**

- Soubor **dataset.py**
- Soubor **engine.py**
- Soubor **prepare.py**
- Soubor **train_faster_rcnn.py**
- Soubor **train_ssd.py**
- Soubor **utils.py**

➢ Adresář **yolov5**

➢ Soubor **fulltext** – text bakalářské práce ve formátu PDF/A