VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2021 Bc. Petr Mikulský

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

DETEKCE POHYBUJÍCÍCH SE OBJEKTŮ VE VIDEU S VYUŽITÍM NEURONOVÝCH SÍTÍ
OBJECT DETECTION IN VIDEO USING NEURAL NETWORKS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Petr Mikulský**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Vojtěch Myška**
SUPERVISOR

BRNO 2021

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní program **Telekomunikační a informační technika**

Ústav telekomunikací

Student: Bc. Petr Mikulský ***ID:*** 183803
Ročník: 2 ***Akademický rok:*** 2020/21

NÁZEV TÉMATU:

Detekce pohybujících se objektů ve videu s využitím neuronových sítí

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se s technikami konvolučních neuronových sítí, problematikou detekce objektů v obraze a zpracujte aktuální stav vědy a techniky v této oblasti. V teoretické části získané vědomosti využijte pro návrh funkčního řešení problematiky klasifikace pohybujících se účastníků dopravního provozu. Za tímto účelem navrhněte neuronové sítě nebo využijte existující předtrénované modely, které budou zpracovávat data z video záznamů. Případně využité existující modely je nutné dotrénovat na sesbírané datové množině. Dosažené výsledky vhodně srovnejte a diskutujte.

DOPORUČENÁ LITERATURA:

[1] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, “Yolov4: Optimalspeed and accuracy of object detection,” 2020.

[2] Keras. Developer guides [online]. Dostupné z: https://keras.io/guides/

Termín zadání: 1.2.2021 ***Termín odevzdání:*** 24.5.2021

Vedoucí práce: Ing. Vojtěch Myška

prof. Ing. Jiří Mišurec, CSc.
předseda rady studijního programu

ABSTRAKT

Tato diplomová práce řeší detekci pohybujících se objektů ve videu s využitím neuronových sítí. Cílem práce byla detekce účastníků silničního provozu na video záznamech. Pro praktické řešení práce byl použit předtrénovaný detekční model YOLOv5. V rámci řešení byla vypracována vlastní datová množina ze záběrů dopravní komunikace s třídami: osobní automobil, autobus, dodávkový automobil, motocykl a kamion. Celkově finální podoba datové množiny čítá 5404 snímků a 6467 anotovaných objektů. Dotrénovaný model YOLOv5 dosáhl na testovací množině úspěšnosti detekce vozidel mAP 0,995, preciznosti 0,995 a úplnosti odhadu predikce 0,986. V závěru jsou popsány kroky, které vedly ke konečné podobě vlastního datasetu.

KLÍČOVÁ SLOVA

Detekce vozidel, konvoluční neuronové sítě, PyTorch, strojové učení, YOLOv5.

ABSTRACT

This diploma thesis deals with the detection of moving objects in a video recording using neural networks. The aim of the thesis was to detect road users in video recordings. Pre-trained YOLOv5 object detection model was used for a practical part of the thesis. As part of the solution, an own dataset of traffic road video recordings was created and annotated with following classes: a car, a bus, a van, a motorcycle, a truck and a trailer truck. Final version of this dataset comprise 5404 frames and 6467 annotated objects in total. After training, the YOLOv5 model achieved 0.995 mAP, 0.995 precision and 0.986 recall on the dataset. All steps leading to the final form of the dataset are described in the conclusion chapter.

KEYWORDS

Vehicle detection, convolution neural networks, PyTorch, machine learning, YOLOv5.

MIKULSKÝ, Petr. *Detekce pohybujících se objektů ve videu s využitím neuronových sítí*. Brno, 2021, 69 s. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce: Ing. Vojtěch Myška

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Detekce pohybujících se objektů ve videu s využitím neuronových sítí“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu Ing.Vojtěchu Myškovi za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci. Také bych rád poděkoval rodině za podporu během celého studia.

Obsah

Seznam obrázků

Seznam tabulek

Úvod

V posledním desetiletí zažíváme obrovský rozvoj a praktické využití strojového učení v aplikacích z mnoha různorodých oblastí. Současné moderní dostupné technologie umožňují zpracovávat obrovské množství dat během relativně krátké doby, a to díky nárůstu výpočetního výkonu moderních systémů a intenzivnímu výzkumu v oblasti umělé inteligence. Algoritmy strojového učení nacházejí uplatnění v širokém spektru nejrůznějších oborů od zdravotnictví, přes vzdělávání, bezpečnostní technologie, bankovnictví, dopravu atd.

Právě aplikace algoritmů umělé inteligence pro detekci pohybujících se objektů ve videu v oblasti silniční dopravy je náplní této diplomové práce.

Silniční doprava je nejrozšířenější druh dopravy především z důvodu flexibilnosti a dostupnosti používaných dopravních prostředků. Kromě nezanedbatelných negativních vlivů na životní prostředí vlivů však rostoucí hustota dopravy na pozemních komunikacích přináší různé další problémy – v praxi je nutné řešit např. nedostatečnou kapacitu stávajících komunikací, časté rekonstrukce a opravy stávajících komunikací, problémy s organizací plynulé a bezpečné dopravy atd.

Předmětem této diplomové práce je využití již existujících předtrénovaných modelů, pomocí kterých bude z video záznamů, pořízených na pozemních komunikacích, prováděna detekce a klasifikace jednotlivých účastníků silničního provozu. Takto získaná data mají vysokou vypovídající schopnost a mohou být nadále analyticky zpracovávána. Přínos mohou mít například pro vytváření dopravních map, sledování zatížení silnic, určení priorit při opravě silnic nebo budování nových silničních tahů apod.

V současné době je nejpopulárnějším řešením aplikace algoritmů umělé inteligence pro detekci pohybujících se objektů ve videu použití konvolučních neuronových sítí. Ty jsou hlavním trendem v oblasti zpracování obrazu a počítačového vidění, jelikož jsou schopny provádět klasifikaci i detekci objektů.

Obsahem této diplomové práce jsou základní informace o konvolučních neuronových sítích a o aktuálních metodách jejich využití při detekci objektů v obraze. V rámci praktické části je využití existujících předtrénovaných modelů k detekci a klasifikaci objektů na shromážděných videozáznamech.

První kapitola diplomové práce je věnována úvodu do umělých neuronových sítí, je zde zpracován jejich popis, popis základních stavebních prvků, dále jsou v kapitole popsány konvoluční neuronové sítě. Popis konvolučních neuronových sítí se věnuje zejména architektuře a typům jednotlivých vrstev konvolučních neuronových sítí.

Druhá kapitola mapuje aktuální stav vědy a techniky v oblasti detekce objektů ve videu. V kapitole jsou popsány nejmodernější a nejznámější architektury konvolučních neuronových sítí pro detekci objektů.

V praktické části diplomové práce je na základě získaných vědomostí z teoretické části proveden výběr detekční architektury. Jsou zde popsány všechny použité nástroje a zařízení, které byly v rámci praktického řešení práce použity. Rovněž jsou zde popsány použité datové množiny a proces jejich vytváření.

V poslední kapitole jsou uvedeny průběhy trénování a výsledky testování všech provedených experimentů. Pozornost je věnována především preciznosti a úplnosti detekce natrénovaných modelů. Jsou zde diskutovány výsledky a vzniklé nepřesnosti jednotlivých experimentů. Tyto chyby a nepřesnosti detekce vozidel jsou řešeny a optimalizovány.

1 Umělé neuronové sítě

Umělá neuronová síť je datová struktura, jejíž inspirací jsou biologické neuronové sítě, zejména lidský mozek. Cílem umělých neuronových sítí je napodobit základní funkce svých biologických vzorů. Podstatným rozdílem oproti běžné tvorbě uživatelských programů, kde je potřeba vytvářet algoritmy transformující vstupní množinu dat na množinu dat výstupní, je schopnost učení. Způsob transformace vstupních dat na výstupní určuje fáze učení, založená na expozici vzorků popisující řešenou problematiku (trénovací množina). Algoritmizace úlohy je nahrazena předložením trénovací množiny umělé neuronové síti a jejím učením [1].

1.1 Biologický neuron

Základním stavebním funkčním prvkem nervové soustavy je nervová buňka neuron. Neuron je specializovaný na zpracování, uchování a přenos informace. Biologický neuron se skládá ze čtyř hlavních částí:

- soma – tělo neuronu s buněčným jádrem,
- dendrity – tvoří vstupy neuronu,
- axon – tvoří výstup z neuronu,
- synapse – zakončení axonu, zprostředkovávají přenos informací mezi navzájem spolupracujícími neurony. Tvoří jakési informační rozhraní. Při procesu učení se průchodnost synapsí mění.

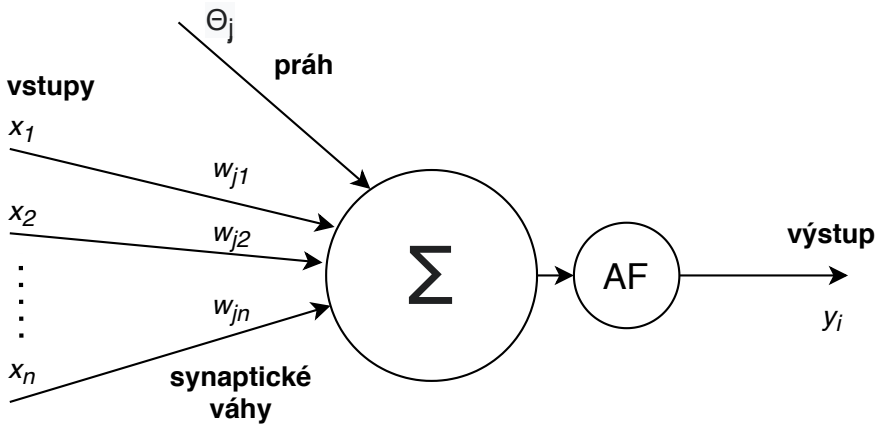
Neurony jsou velmi komplexní buňky. Výpočetní procesy uvnitř neuronů umožňují obrovskou variabilitu mechanismů, které nejsou dodnes detailně prozkoumány. Snaha matematicky popsat biologický neuron vyústila ve vznik formálního neuronu. [2, 3].

1.2 Formální neuron

Základním stavebním prvkem umělé neuronové sítě je matematický model neuronu. Typy používaných modelů neuronů se liší v používaných matematických funkcích a ve složitosti modelu. Nejrozšířenějším modelem je tzv. formální neuron, známý také podle svých autorů jako McCulloch-Pittsův model. Každý model neuronu se skládá ze dvou částí:

- obvodová funkce – určuje způsob, kterým budou kombinovány vstupní parametry uvnitř neuronu,
- aktivační funkce – definuje přenosovou funkci, určuje jak budou vstupní parametry transformovány na výstup neuronu.

Formální neuron má n obecně reálných vstupů x_i, modelující dendrity. Každý vstup je ohodnocen odpovídající synaptickou váhou w_i, která určuje jeho propustnost. Změnou synaptických vah během učení je formální neuron schopen adaptovat nově nabyté zkušenosti. Θ_j značí práh j-tého neuronu (bias). Neuron je aktivní pouze tehdy, pokud je vážená sumu vstupů větší než práh [2, 4].



Obr. 1.1: Model formálního neuronu [2].

1.2.1 Aktivační funkce

Úkolem aktivační funkce je transformace vstupních parametrů na výstup. Vstup aktivační funkce tvoří suma všech vstupů s odpovídajícími vahami. Po aplikování aktivační funkce získáváme výstup celého neuronu. Míra úspěšnosti predikce neuronové sítě záleží nejen na počtu vrstev sítě, ale také na typu použitě aktivační funkce. Příklady často používaných aktivačních funkcí [5]:

- Lineární aktivační funkce:
$$f(x) = ax \tag{1.1}$$
- Sigmoida:
$$f(x) = \frac{1}{1 + e^{-x}} \tag{1.2}$$
- Hyperbolický tangens:
$$f(x) = \frac{1 - e^{-x}}{1 + e^{-x}} \tag{1.3}$$
- ReLU:
$$f(x) = \begin{cases} 0 & \text{pro } x \leq 0 \\ x & \text{pro } x > 0 \end{cases} \tag{1.4}$$

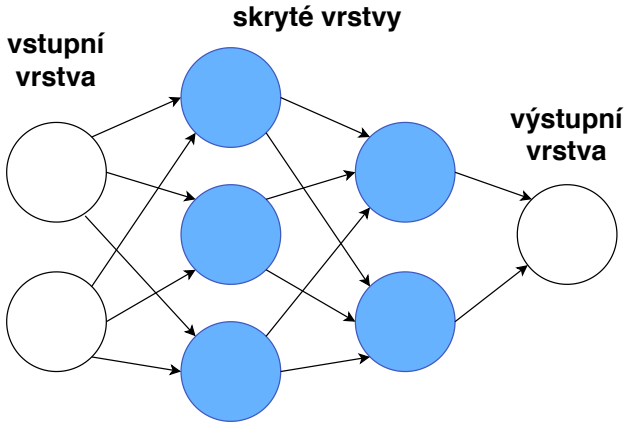
- Leaky ReLU:

$$f(x) = \begin{cases} 0,01x & \text{pro } x \leq 0 \\ x & \text{pro } x > 0 \end{cases} \tag{1.5}$$

1.3 Vícevrstvé neuronové sítě

Propojením více neuronů mohou být vytvářeny rozličné neuronové sítě. Každá neuronová síť je charakterizována svojí architekturou, počtem neuronů a typem učení. Vícevrstvé neuronové sítě jsou tvořeny neurony sestavenými do vrstev. Podle propojení neuronů ve vrstvách rozlišujeme sítě s dopředným šířením informace (feedforward) a rekurentní sítě (recurrent networks). Vícevrstvé neuronové sítě jsou tvořeny vždy minimálně třemi vrstvami. Mezi tyto vrstvy patří:

- vstupní vrstva – distribuční vrstva vstupních dat, s daty neprobíhají žádné operace,
- skrytá vrstva – přijímá data, která následně transformuje a posílá na vstup navazující vrstvy,
- výstupní vrstva – obsahuje výsledná data po průchodu celou neuronovou sítí.



Obr. 1.2: Jednoduchá vícevrstvá dopředná neuronová síť [2].

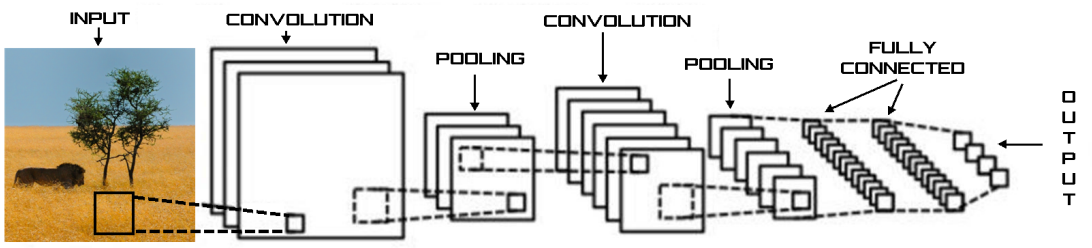
1.4 Konvoluční neuronové sítě

Konvoluční neuronové sítě (CNN) patří mezi nejpopulárnější typ umělých neuronových sítí, obzvláště pro zpracování dat velkých rozměrů, jako jsou obrázky a videa. CNN jsou úspěšně používány pro mnohé úlohy jako je rozpoznávání objektů, číslic apod. Mezi základní CNN vrstvy patří:

- konvoluční vrstva,

- pooling vrstva,
- plně propojená vrstva.

CNN fungují podobně jako standardní neuronové sítě, důležitým rozdílem je, že každá jednotka v konvoluční vrstvě je vícerozměrný filtr (kernel), který je pomocí konvoluce aplikován na vstup této vrstvy. Formálně je každý obrázek reprezentován jako trojrozměrná matice pixelů (šířka, výška a barva). Úkolem filtrů je ve vstupním obraze detekovat jednoduché lokální vzory, např. hrany. Použití konvolučních vah výrazně snižuje počet parametrů, které se síť musí naučit [6, 7].



Obr. 1.3: Příklad architektury CNN, převzato z [9].

1.4.1 Konvoluční vrstva

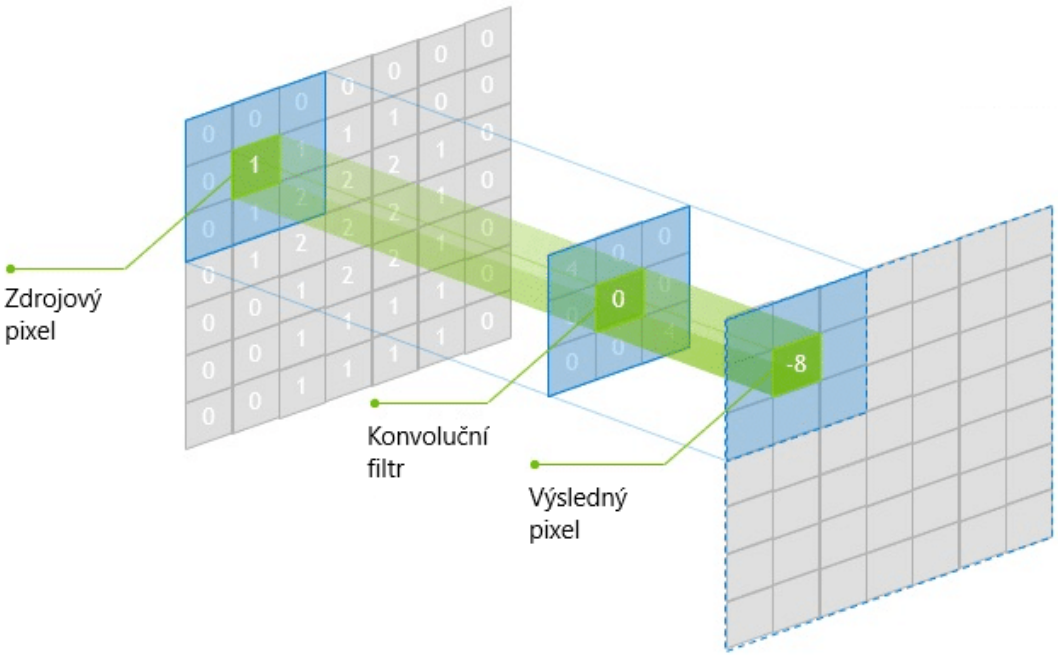
Konvoluční vrstva je nejdůležitější komponenta CNN, která obsahuje sadu filtrů (konvolučních jader). Každý filtr v konvoluční vrstvě má definovanou velikost a váhy, které nastavovány během trénování CNN. Typické rozměry konvolučních filtrů jsou $3 \times 3 \times h$, $5 \times 5 \times h$ nebo $7 \times 7 \times h$. Hloubka h závisí na hloubce vstupních dat (např. pro RGB obrázek $h = 3$). Tyto filtry se nejčastěji používají pro zpracování obrazů o velikostech $110 \times 110 \times h$, $224 \times 224 \times h$ a větších. Tato struktura CNN přináší dvě klíčové vlastnosti. Počet parametrů, které je potřeba se naučit, je výrazně snížen při použití malých filtrů. Malé filtry také zajistí, že charakteristické vzory se CNN učí z lokálních částí obrazu. Výstupem konvoluční vrstvy je příznaková mapa (feature map). Operace diskrétní konvoluce je definována jako:

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a). \tag{1.6}$$

V aplikacích CNN je vstupem většinou vícerozměrné pole dat a stejně tak i filtr je vícerozměrné pole parametrů. V praxi můžeme implementovat nekonečný součet jako součet přes nekonečný počet prvků pole. V CNN používáme konvoluci na více než jedné ose najednou. Pokud použijeme dvojrozměrný obraz jako vstup I a jako filtr dvourozměrné konvoluční jádro K, bude matematický zápis diskrétní konvoluce

vypadat takto [6, 8]:

$$S(i,j) = (I * K)(t) = \sum_m \sum_n I(m,n)K(i-m, j-n). \tag{1.7}$$



Obr. 1.4: Ukázka konvoluce jednoho pixelu, převzato z [10].

1.4.2 Pooling vrstva

Pooling vrstva je další částí CNN, obvykle je řazena periodicky za konvoluční vrstvu. Hlavní úkol této vrstvy je redukce rozměrů dat. Operace pooling efektivně podvzorkovává vstupní příznakovou mapu. V neposlední řadě také pooling vrstva snižuje počet parametrů pro učení, díky tomu se sníží výpočetní náročnost trénování CNN a snižuje riziko přetrénování. Podobně jako u konvoluční vrstvy je potřeba definovat rozměr a délku kroku posunu filtru (stride). Nejčastěji používanou funkcí v praxi je max pooling, kdy z hodnot k zpracování vybereme nejvyšší hodnotu. Další možná funkce je average pooling, při které se počítá průměr hodnot. Tato operace byla využívaná dříve [6, 11].

1.4.3 Plně propojená vrstva

Plně propojená vrstva v podstatě odpovídá konvoluční vrstvě s filtry o velikosti 1 x 1. Jak již název napovídá, všechny neurony v plně propojené vrstvě se připojují

ke všem neuronům v předchozí vrstvě. Tato vrstva kombinuje všechny vzory naučené předchozími vrstvami napříč obrázkem, aby identifikovala větší vzory. V typické CNN jsou plně propojené vrstvy obvykle umístěny ke konci architektury. Jsou však známy některé úspěšné architektury, které tento typ vrstvy používají v mezilehlém umístění v rámci CNN. U problémů klasifikace odpovídá počet kanálů poslední plně propojené vrstvy počtu tříd datové sady [6, 12].

2 Současný vývoj v oblasti detekce objektů

Tato kapitola je věnována problematice detekce objektů ve videu a mapuje aktuální stav vědy a techniky. Jsou zde popsány nejpopulárnější detekční systémy a architektury. Zvláštní pozornost je věnována algoritmům YOLO, které byly použity pro vypracování praktické části diplomové práce.

2.1 Detekce objektů ve videu

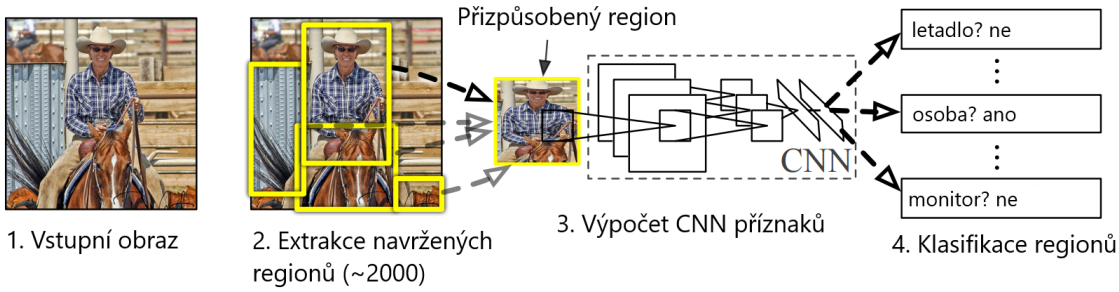
Člověk je schopen podívat se na obrázek a okamžitě ví, jaké objekty se na obrázku vyskytují, kde jsou a jak interagují. Lidský vizuální systém je rychlý a přesný, to lidem umožňuje provádět složité úkoly, jako je řízení, s malým vědomým myšlením. Rychlé a přesné algoritmy pro detekci objektů by například umožnily počítačům řídit automobily bez specializovaných senzorů. Detekce objektů ve videu s sebou ovšem nese plno problémů, které je potřeba řešit. Rozdíl mezi algoritmy detekce objektů a klasifikačními algoritmy spočívá v tom, že detekční algoritmy se pokoušejí nakreslit ohraničující rámeček kolem hledaného objektu a označit jej v obraze. Na snímku se nemusí nutně vyskytovat jen jeden hledaný objekt a předem není jasné, kolik takových objektů se v obraze vyskytuje. Detekční algoritmy musí být výpočetně nenáročné, aby bylo možné jejich nasazení v reálném čase [13].

2.2 Architektury CNN pro detekci

2.2.1 R-CNN

Architektura R-CNN (Region Based Convolutional Neural Networks) získala své jméno, protože kombinuje použití CNN s návrhy regionů hledaných objektů. Detekční model R-CNN vznikl v roce 2014 na University of California, Berkeley. Oproti předchozím modelům se R-CNN liší tím, že je množství navrhovaných regionů redukováno na 2000. Mechanismus funkce modelu R-CNN je vyobrazen na obrázku 2.1.

Na začátku procesu je načten vstupní obraz. V následujícím kroku je generováno 2000 návrhů kandidátských oblastí hledaných objektů. Těchto 2000 návrhů kandidátských oblastí je generováno pomocí algoritmu selektivního vyhledávání pro rozpoznávání objektů (Seq-NMS) viz [14]. Následně jsou regiony deformovány do čtvercových obrazců a poté jsou přivedeny na vstup CNN. Tato konvoluční neuronová síť extrahuje z přizpůsobených obrázků vektor příznaků. Extrahované vektory jsou 4096-ti rozměrné. Získané příznaky jsou dále zpracovávány sítí složenou z pěti konvolučních vrstvev a dvou plně propojených. V posledním kroku jsou extrahované

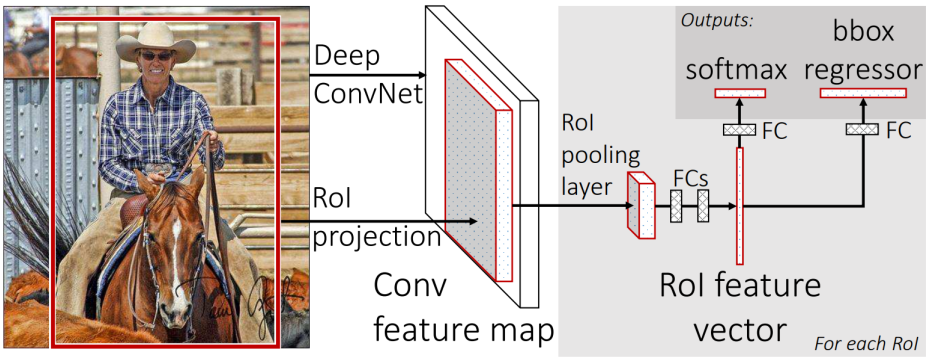


Obr. 2.1: Ukázka funkce detekčního systému R-CNN, převzato z [15].

příznaky klasifikovány pomocí metody podpůrných vektorů (Support vector machines). Hlavní výhodou R-CNN je vysoká přesnost, avšak tato architektura sebou nese i mnohá negativa. Hlavní nevýhodou je časová náročnost učení, pro každý zpracovávaný snímek je nutné klasifikovat 2000 návrhů kandidátských oblastí [15]. Tato technologie se kvůli své výpočetní a časové náročnosti nehodí pro aplikaci v reálném čase. Zpracování jednoho obrázku trvá 47 sekund (při použití GPU Nvidia K40 přetaktované na 875 Mhz) [16].

2.2.2 Fast R-CNN

Model Fast R-CNN (Fast Region Based Convolutional Neural Networks) je opět dílem autora R-CNN Rosse Girshicka. Tato architektura řeší některé z nedostatků R-CNN. Princip modelu je podobný jako u původní architektury, hlavní rozdíl spočívá v tom, že místo přivádění návrhů kandidátských oblastí na vstup CNN, přivádíme do CNN vstupní obraz. Na obrázku 2.2 je zachycena struktura funkčního modelu Fast R-CNN.



Obr. 2.2: Model Fast R-CNN, převzato z [16].

Pomocí CNN je vygenerována konvoluční příznaková mapa. Z konvoluční příznakové mapy jsou identifikovány kandidátské oblasti, ty jsou následně deformovány na čtverce. Pomocí RoI (Region of Interest) pooling vrstvy jsou čtverce upraveny na fixní rozměry, aby bylo je možno je přenést na vstup plně propojené vrstvy. K předpovědi patřičné třídy a offsetu pro ohraničující rámeček (bounding box) je použita vrstva softmax a RoI příznakový vektor. Důvodem, proč je Fast R-CNN oproti svému předchůdci výrazně rychlejší, je fakt, že není potřeba na vstup CNN přivádět 2000 kandidátských oblastí, ale konvoluční operace je prováděna pouze jednou u každého obrázku [16].

2.2.3 Faster R-CNN

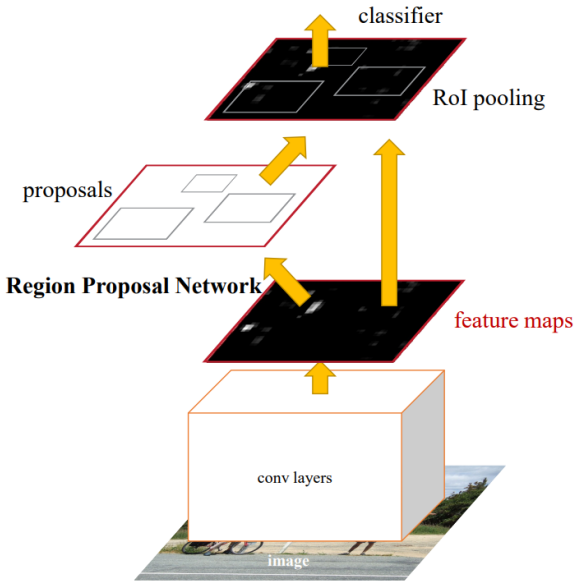
Předchozí pokrok v detekci objektů byl poháněn úspěchem algoritmů z rodiny R-CNN. Původní výpočetní náročnost u architektur založených na navrhování regionů byla drasticky snížena, díky sdílení konvoluce mezi návrhy regionů. Nejnovější inkarnace Fast R-CNN se přibližuje detekci objektů v reálném čase, je-li ignorován čas potřebný k návrhu regionů. Jak R-CNN tak i Fast R-CNN využívají pro návrh regionů selektivní vyhledávací algoritmy, které díky své výpočetní náročnosti činí tyto architektury nepoužitelnými pro real-time nasazení [17].

V roce 2015 Shaoqing Ren a spol. přišli s algoritmem Faster R-CNN (Faster Region Based Convolutional Neural Networks), který eliminuje algoritmus selektivního vyhledávání a umožňuje síti naučit se návrhy regionů. Architektura modelu Fast R-CNN je znázorněna na obrázku 2.3.

Tento detekční systém je složen z dvou modulů. První částí je hluboká konvoluční síť RPN (Region Proposal Network), která navrhuje regiony. Druhým modulem je detektor Fast R-CNN, který využívá navrhované regiony z prvního modulu. Celý systém tvoří jednotnou síť pro detekci objektů. Síť pro návrh regionů RPN „říká“ modulu Fast R-CNN, kde v obraze hledat cílové objekty.

Síť pro návrh regionů RPN

Na vstup RPN je přiveden obraz libovolných rozměrů, následně je vytvořena sada obdélníkových návrhů objektů. Ke každému návrhu je připojeno skóre objektivity, které značí příslušnost k třídě objektů proti pozadí. Předpokladem je, že RPN sdílí konvoluční vrstvy s detektorem Fast R-CNN, protože je cílem, aby tyto dva moduly sdílely výpočty. Návrh včetně skóre objektivity je získán přesunem malé neuronové sítě přes příznakovou mapu, kterou generuje CNN. Tato malá neuronová síť přijme na vstupu $n \times n$ prostorové okno ze vstupu příznakové mapy. Každé posuvné okno je mapováno do příznaku nižší dimenze. Příznak je dále přiveden do dvou příbuzných plně propojených vrstev (regresivní a klasifikační).



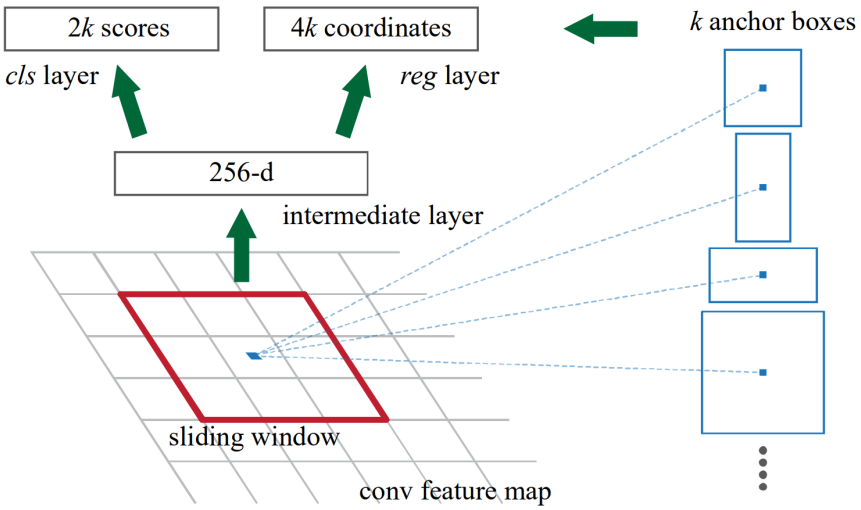
Obr. 2.3: Detekční architektura Faster R-CNN, převzato z [17].

Na obrázku 2.4 je zobrazena jedna pozice posuvného okna na příznakové mapě. Pro každou pozici posuvného okna je zároveň predikováno více návrhů regionů, kdy maximální počet návrhů regionů pro jeden posun je značen k. Výstup regresivní vrstvy tvoří $4k$ zakódovaných souřadnic ohraničujících rámečků. Výstupem klasifikační vrstvy je $2k$ odhadů pravděpodobností, zda v regionu je, či není hledaný objekt. Počet návrhů k je závislý na k referenčních rámečcích, které jsou označovány jako kotvy (anchors) [17].

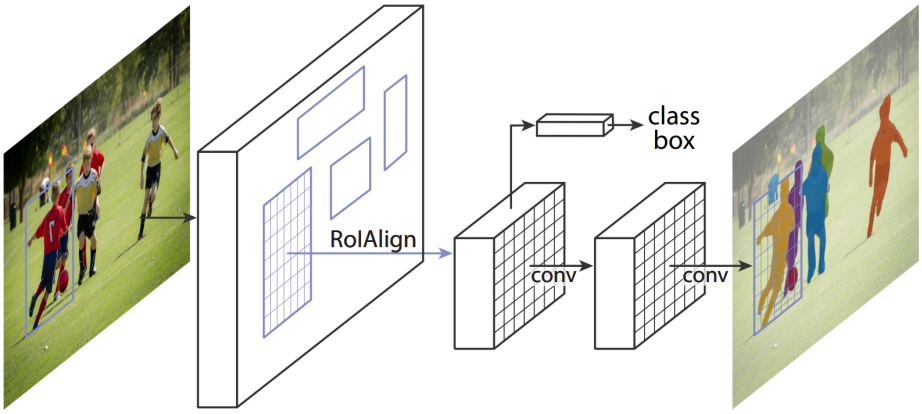
2.2.4 Mask R-CNN

V roce 2017 přišla výzkumná laboratoř FAIR (Facebook AI Research) s metodou rozšiřující Fast R-CNN o část predikující masku objektu paralerně s existující částí předpovídající bounding box. Segmentace instancí v obraze je náročná úloha, protože vyžaduje správnou detekci všech objektů a zároveň přesné segmentování každé instance. Tento framework tedy kombinuje prvky z klasických úkolů počítačového vidění detekce objektů, kde je cílem klasifikovat jednotlivé objekty a lokalizovat každý pomocí bounding boxů a přidává sémantickou segmentaci, která má za cíl klasifikovat každý pixel do pevné sady kategorií bez rozlišení třídy objektu.

Mask R-CNN používá stejný, dvoustupňový systém jako Faster R-CNN. Kompletní framework je zobrazen a popsán na obrázku 2.5. První modul architektury tvořen sítí pro návrh regionů (viz 2.2.3), který je zodpovědný za generování návrhů



Obr. 2.4: Region Proposal Network. (RPN), převzato z [17].



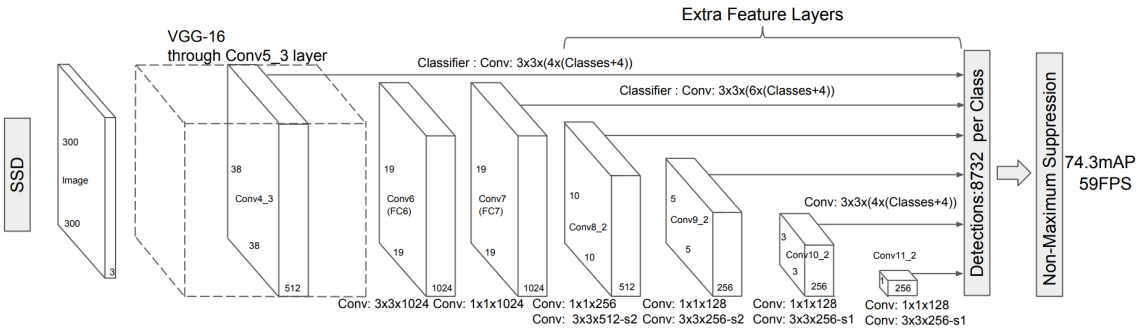
Obr. 2.5: Framework Mask R-CNN, převzato z [18].

regionů a skóre objektu. V druhé fázi převezme další neuronová síť návrhované regiony z první části. Paralelně je predikována třída objektu a bounding box, navíc Mask R-CNN generuje binární masku pro každý RoI. Postup vypadá podobně jako u RPN s tím rozdílem, že namísto využití kotev je použita operace RoIAlign [18].

2.2.5 SSD

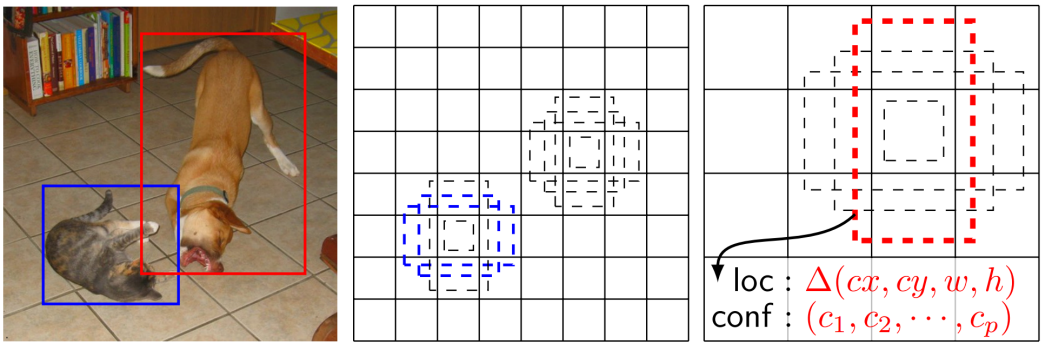
Metoda detekce objektů SSD (Single Shot Multibox Detector) byla uveřejněna v roce 2016 týmem, který tvořili: Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian

Szegedy, Scott Reed, Cheng-Yang Fu a Alexander C. Berg. V době svého vydání stanovila tato metoda rekord z hlediska výkonu a přesnosti detekce, kdy dosahovala přesnosti přes 74 % mAP (mean Average Precision) při 59 FPS (frames per second) na standartních datasetech jako např. COCO nebo PascalVOC. Hlavní zaměření této architektury je možnost nasazení v reálném čase.



Obr. 2.6: SSD framework, převzato z [20].

Přístup SSD je založen na jednoduché dopředné konvoluční neuronové síti, která produkuje kolekci bounding boxů a skóre přítomnosti třídy objektů v těchto boxech. Jak lze vidět na obrázku 2.6 výše, první část architektury SSD je postavena na známé architektuře VGG-16 [19], ale odstraňuje plně propojené vrstvy. Důvodem, proč byla použita architektura VGG-16 jako základ modelu, je její schopnost vysoce kvalitní klasifikace obrazu. Na konec modelu za VGG-16 část jsou přidány konvoluční funkční vrstvy, které umožňují detekci pro různá měřítka a poměry.



(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

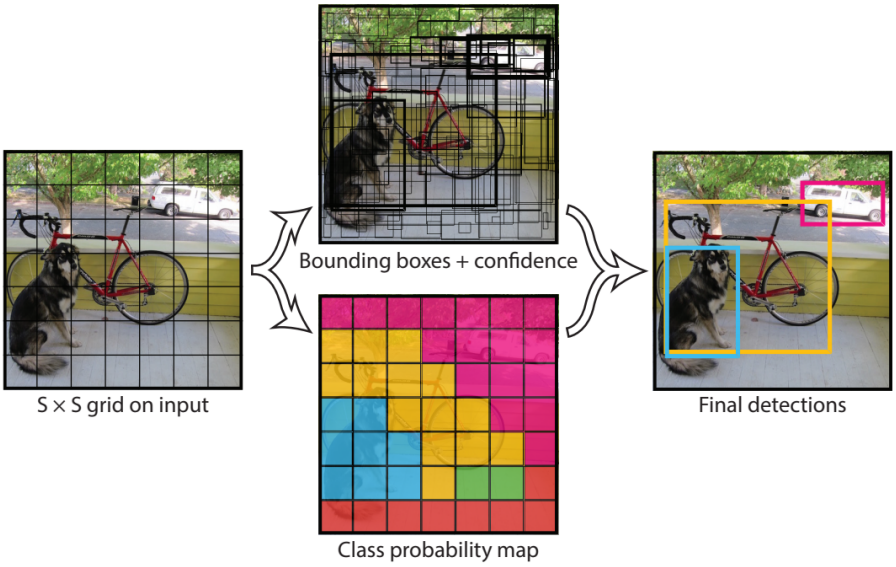
Obr. 2.7: Princip detekce SSD, převzato z [20].

Na rozdíl od Faster R-CNN model SSD nepoužívá síť pro návrh regionů RPN.

SSD řeší jak návrh regionů, tak klasifikaci za pomocí malých konvolučních filtrů. Část zodpovědná za detekci pracuje s odlišnými příznakovými mapami, což přináší vysoký počet predikcí. Každá příznaková mapa je rozdělená na buňky. Počet buněk určuje mřížka. Pro každou buňku jsou použity čtyři různé odsazení oproti originální poloze. Pro každou buňku je odhadován jak posun ohraničené oblasti, tak skóre pro všechny kategorie objektů. Proces je zobrazen na obrázku 2.7 [20].

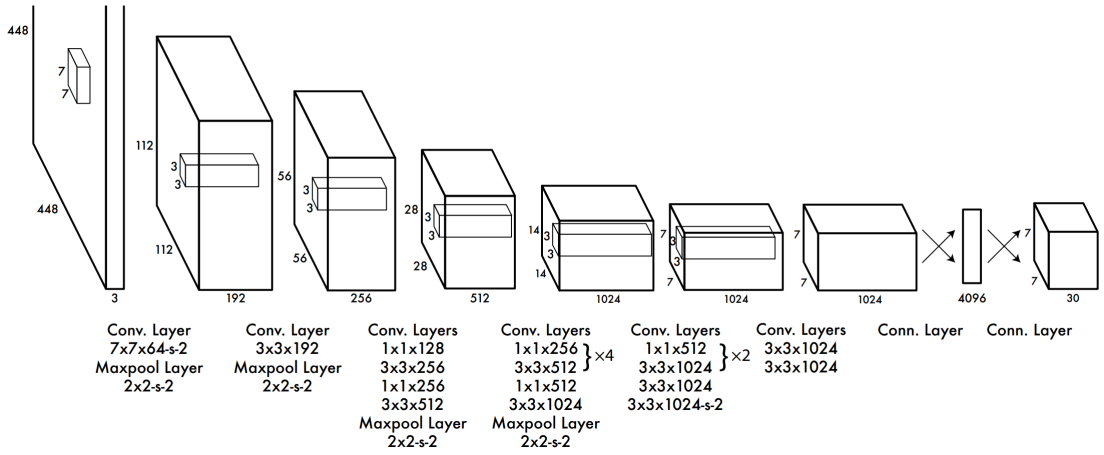
2.2.6 YOLO

Hlavní myšlenkou architektury YOLO (You Only Look Once), jak je již patrné z přiléhavého názvu, je podívat se na každý snímek pouze jednou. Jiné algoritmy jako např. R-CNN zpracovávají snímek dvakrát, kdy v prvním průchodu detekují hrany objektů, až v druhé části CNN objekt klasifikuje. Podobně jako SSD se YOLO řadí mezi single shot detektory. Další nespornou výhodou YOLO oproti jiným detektorů používajících metodu posuvného okno nebo návrhu regionů je schopnost zpracovávat kontextuální informace o objektech, což přispívá k robustnosti modelu.



Obr. 2.8: Postup YOLO při zpracování obrazu, převzato z [13].

YOLO sjednocuje jednotlivé komponenty pro detekci objektů do jedné neuronové sítě. Tato síť využívá příznaky z celého obrazu pro predikci a klasifikaci všech objektů na snímku současně. Model YOLO nejdříve rozdělí vstupní obraz na mřížku s $S \times S$ buňkami. Pro každou buňku předpoví B bounding boxů a C příslušnosti k jednotlivým třídám. Tyto predikce jsou vyjádřeny jako $S \times S \times (B \times 5 + C)$ tenzor.



Obr. 2.9: Architektura YOLO, převzato z [13].

Konvoluční vrstvy umístěné na začátku sítě extrahují příznaky ze snímku, zatímco plně propojené vrstvy predikují výstupní pravděpodobnosti a souřadnice. Tato síťová architektura je inspirována klasifikačním modelem GoogLeNet [21]. Model YOLO je tvořen z 24 konvolučních vrstev, a ze dvou plně propojených vrstech umístěných na konci modelu. Celá síť je zobrazena na obrázku 2.9. Oproti původním počátečním vrstvám z modelu GoogLeNet jsou použity *1* x *1* redukční vrstvy následované *3* x *3* konvolučními vrstvami. Model YOLO je schopen na GPU Titan X detekovat objekty rychlostí 45 FPS a rychlá verze pak dosahuje rychlosti zpracování více než 150 FPS [13].

2.2.7 YOLO9000

YOLO 9000 (YOLOv2) je real-time detektor, který dokáže klasifikovat přes 9000 tříd objektů. Jedná se o vylepšení původní modelu YOLO, verze YOLO9000 byla publikována v roce 2016 Josephem Redmonem s Ali Farhadim na University of Washington. Hlavním nedostatkem původního detektoru YOLO, byla jeho vysoká lokalizační chybovost, kdy dosahoval nižší přesnosti oproti modelu Faster R-CNN. Hlavním záměrem YOLO9000 je zvýšit přesnost detekce objektů při zachování vysoké rychlosti zpracování obrazu. Místo rozšiřování sítě byla síť zjednodušena a byla implementována tato nová důležitá vylepšení:

- Zavedení dávkové normalizace (batch normalization) vede k významnému zlepšení konvergence, přičemž eliminuje potřebu dalších forem regularizace. Přidáním batch normalizace na všechny konvoluční vrstvy v YOLO je dosaženo více než 2% zlepšení v mAP. Dávková normalizace také pomáhá regularizovat model. Díky této úpravě bylo možné z původního modelu odstranit dropout

metody, aniž by došlo k přetrénování.

- Použití high resolution klasifikátoru původní model YOLO trénoval klasifikační síť se vstupními snímky o velikosti 224 x 224 pixelů a navyšoval toto rozlišení na 448 pro účely detekce. Autoři tento problém vyřešili použitím klasifikátoru s vyšším rozlišením 448 x 448 pixelů pro 10 epoch na datové sadě ImageNet. Díky této inovaci byla zlepšena schopnost modelu detekovat objekty na snímcích s vyšším rozlišení, síť tímto dosáhla zlepšení téměr 4% mAP.
- Plně propojené vrstvy byly nahrazeny detekcí pomocí kotevních boxů (anchor boxes), podobně jako u architektry Faster R-CNN. Po procesu učení je síť zmenšena a pracuje se s snímky o velikosti 416 x 416 pixel. Autoři toto navrhli, aby dosáhli jedné buňky na středu obrazu. Velké objekty, mají tendenci zaujímat střed obrázku, takže je dobré mít jediné místo přímo na středu, než předpovídat čtyři buňky poblíž. Konvoluční vrstvy YOLOv2 podvzorkují obraz o faktor 32 a výstupem je příznaková mapa 13×13. Model je sestaven pouze z konvoluční a pooling vrstvy, a tudíž může zpracovávat vstupní data různých rozměrů (rozměry musí být násobkem 32).

Tab. 2.1: Výsledky detekčních architektur na datasetu PASCAL VOC 2007 [22].

Detection Frameworks	Train	mAP	FPS
Fast R-CNN	2007+2012	70,0	0,5
Faster R-CNN VGG-16	2007+2012	73,2	7
Faster R-CNN ResNet	2007+2012	76,4	5
YOLO	2007+2012	63,4	45
SSD300	2007+2012	74,3	46
SSD500	2007+2012	76,8	19
YOLOv2 288 × 288	2007+2012	69,0	91
YOLOv2 352 × 352	2007+2012	73,7	81
YOLOv2 416 × 416	2007+2012	76,8	67
YOLOv2 480 × 480	2007+2012	77,8	59
YOLOv2 544 × 544	2007+2012	78,6	40

Z tabulky 2.1 je patrné, že v době svého vydání model YOLO9000 dosahoval oproti konkurenčním detektorům vyšší rychlosti zpracování počtu snímků za sekundu při dosažení srovnatelné přesnosti mAP [22].

2.2.8 YOLOv3

V roce 2018 byla zásluhou Josepha Redmona a Ali Farhadiho z University of Washington publikována další verze YOLOv3, která přináší oproti svému předchůdci

YOLO9000 podstatná vylepšení. Podobně jako u YOLOv9000 jsou pro predikování bounding boxů použity anchor boxy. Pro výpočet skóre objektu YOLOv3 používá statistickou metodu logistická regrese. Podstatná inovace nastává v oblasti predikce třídy objektu. Nově může být každý bounding-box klasifikován zároveň do více tříd, které se překrývají (objekt může například patřit do třídy člověk a zároveň do třídy žena). Autoři modelu YOLOv3 nadále nepoužívají vrstvy softmax, jelikož nejsou potřebné pro dobrý výkon. Vrstvy softmax byly jendoduše nahrazeny nezávislými logistickými klafikátory, během fáze trénování jsou pro predikci využívány binární cross-entropy loss funkce.

Tab. 2.2: Síť Darknet-53 [23].

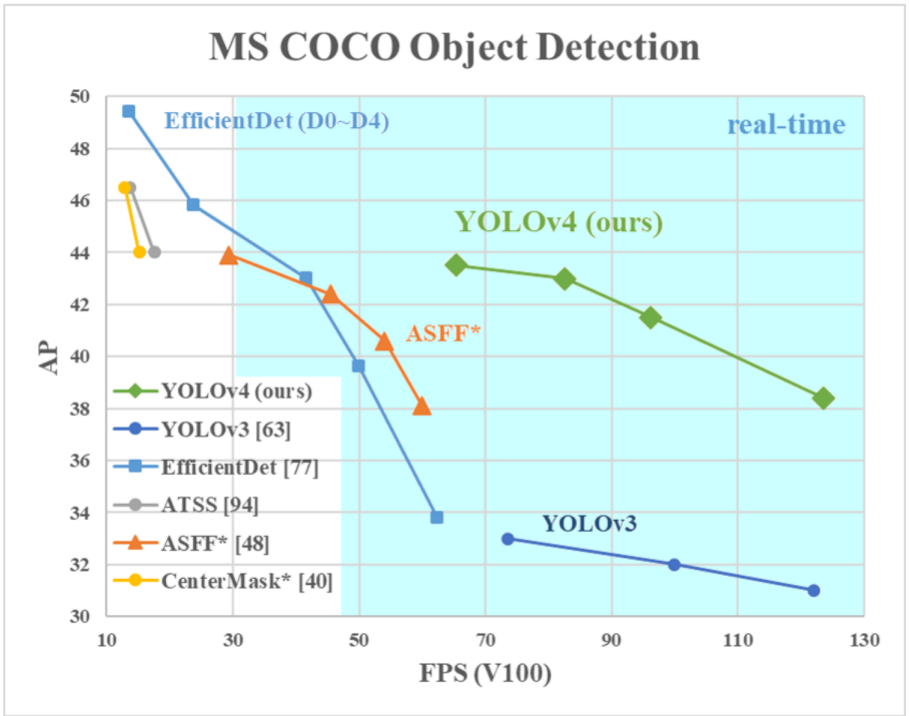
	Type	Filters	Size/Stride	Output
	Convolutional	32	3×3	256×256
	Convolutional	64	3×3 / 2	128×128
	Convolutional	32	1 x 1	
1x	Convolutional	64	3×3	
	Residual			128×128
	Convolutional	128	3×3 / 2	64×64
	Convolutional	64	1 x 1	
2x	Convolutional	128	3×3	
	Residual			64×64
	Convolutional	256	3×3 / 2	32×32
	Convolutional	128	1 x 1	
8x	Convolutional	256	3×3	
	Residual			32×32
	Convolutional	512	3×3 / 2	16×16
	Convolutional	256	1 x 1	
8x	Convolutional	512	3×3	
	Residual			16×16
	Convolutional	1024	3×3 / 2	8×8
	Convolutional	512	1 x 1	
4x	Convolutional	1024	3×3	
	Residual			8×8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Pro extrakci příznaků je použita nová síť Darknet-53. Oproti síti Darknet-19 používané modelem YOLOv2 je rozšířena na 53 konvolučních vrstev oproti původním

19. Tento nárůst počtu konvolučních vrstev má za následek daleko vyšší přesnost detekce oproti předchozí verzi. Zvýšení přesnosti je dosaženo za cenu snížení rychlosti modelu. Oproti klasifikačním architekturám ResNet dosahuje Darknet-53 při podobné přesnoti výrazně vyšší rychlosti [23].

2.2.9 YOLOv4

V roce 2020 tvůrce YOLO Joe Redmon ukončil svůj výzkum počítačového vidění, aby zabránil možnému zneužití technologie. Obavy měl zejména o vojenské uplatnění a dopadu, jaký může mít tato technologie na soukromí [24]. V dubnu roku 2020 byla publikována již čtvrtá verze YOLO, za kterou nově stojí Alexey Bochkovskiy, Chien-Yao Wang a Hong-Yuan Mark Liao. Hlavním cílem této verze je návrh vysokorychlostního detektoru s optimalizací pro paralelrní výpočty. Model YOLOv4 může být trénován na konvenčních grafických kartách (Nvidia 2080Ti nebo 1080Ti), přitom dosahuje vysoké přesnosti a umožňuje real-time nasazení. Tato čtvrtá verze YOLO sebou přináší vylepšení formou tzv. „bag-of freebies“ a „bag-of-specials“ metod používaných při trénování detektoru.



Obr. 2.10: Porovnání YOLOv4 s ostatními moderními detektory, převzato z [28].

Metody, které pouze mění strategii trénování nebo jenom navyšují náročnost trénování, jsou nazvány bag-of freebies. Co je často adaptováno u detekčních architektur a splňuje definici bag-of freebies je rozšiřování dat. Cílem rozšiřování dat

je zvýšení variability vstupního obrazu. Fotometrická a geometrická zkreslení jsou dva příklady běžně používaných metod pro rozšiřování dat, z kterých detekce objektů benefituje. Při použití fotometrické zkreslení, upravíme jas, kontrast, odstín, sytost a šum obrazu. U geometrické zkreslení přidáme náhodné měřítko, oříznutí, převrácení a otočení snímku.

Plugin moduly a metody následného zpracování, které způsobují mírný nárůst výpočetní náročnosti modelu, ale výrazně zvyšují přesnost detekce objektů, jsou označeny jako bag-of-specials. Obecně řečeno, plugin moduly slouží k vylepšení určitých atributů modelu, jako je rozšíření vjemových regionů (receptive field) nebo zavedení mechanismu pozornosti. Metody následného zpracování nachází uplatnění při monitorování výsledků predikce modelu. Běžné moduly, které lze použít pro zlepšení rozšíření vjemových regionů jsou SPP [25], ASPP [26] a RFB [27].

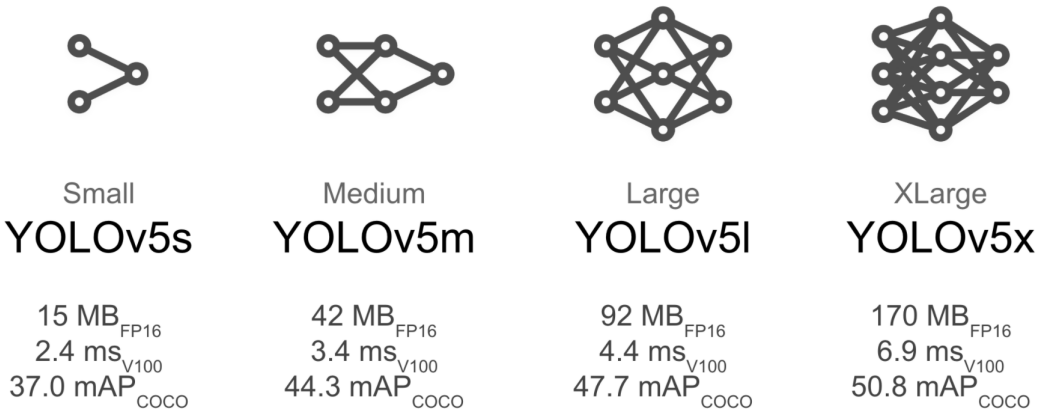
YOLOv4 vyjma možnosti použití konvenčních GPU přináší také nezanedbatelné zlepšení výkonu. Oproti předchozí třetí verzi dosahuje vyšší jak přesnosti AP (Average Precision), tak i rychlosti zpracování o 10 % respektive 15 % viz srovnání na obrázku 2.10 [28].

2.2.10 YOLOv5

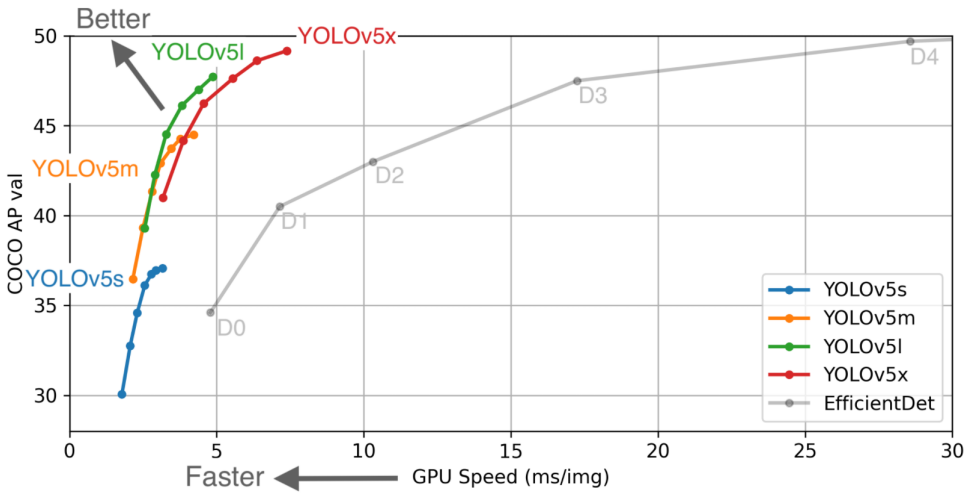
V červnu roku 2020, dva měsíce po vydání YOLOv4, byla představena již pátá verze YOLO. Tvůrcem YOLOv5 je Glenn Jocher z Ultralytics, který zveřejnil zdrojový kód formou GitHub repozitáře. Jedná se o open-source výzkum, který zahrnuje zkušenosti a osvědčené postupy Ultralytics z jejich předchozí práce na YOLOv3. Veškerý kód a modely jsou v stále aktivním vývoji. YOLOv5 nabízí čtyři předtrénované modely různých velikostí: YOLOv5s, YOLOv5m, YOLOv5l a YOLOv5x (small, medium, large a xlarge). Porovnání rychlostí a přesností jednotlivých modelů je zobrazeno na obrázku 2.12 [29].

Vydání této páté verze bylo kontroverzní hned z několika důvodů. Verze YOLOv5 nebyla zveřejněna formou oficiálního dokumentu, ale pouze formou GitHub repozitáře. V době psaní této práce stále nebyl vydán žádný oficiální dokument. Oproti předchozím modelům z rodiny detektorů YOLO využívajících Darknet, byla nejnovější verze implementována nativně v PyTorch [30].

Počáteční vydání YOLOv5 se jeví jako velmi rychlé, výkonné a snadno použitelné. Přestože YOLOv5 dosud nepředstavilo oficiální dokumentaci k novým vylepšením architektury modelů rodiny YOLO modelů, zavádí nově implementaci v PyTorch, čímž je výrazně urychlen čas potřebný k trénování sítě. U předchozích verzí byly dostupné pouze plnohodnotné modely a modely tiny, které byly určeny pro použití na zařízeních s nižším výkonem jak např. mobilní telefony. Tiny verze oproti standartním modelům výrazně zaostávaly v přesnosti detekce objektů. Oproti tomu



Obr. 2.11: Modely YOLOv5, převzato z [29].



Obr. 2.12: Srovnání výkonů modelů YOLOv5, převzato z [29].

YOLOv5 přináší modely čtyři, každé s jiným zaměřením (ať už jde o rychlost nebo přesnost detekce), a tím i větší možnou variabilitu použití [31].

3 Praktické řešení

Tato kapitola mapuje praktické řešení diplomové práce, jejíž cílem je implementovat algoritmus pro detekci objektů ve videu (konkrétně různé typy motorových vozidel). Je zde odůvodněn výběr detekční architektury, dále je popsán postup řešení, všechny nástroje, zařízení a data, které byly při řešení použity.

3.1 Výběr detekční architektury

Při výběru detekční architektury bylo zohledněno několik kritérií. Hlavní požadavkem byla taková rychlost modelu, která by umožnila real-time nasazení modelu. Dalším neméně důležitým kritériem výběru byla míra přesnost detekce objektů. Cílem této práce je pouze detekce, nikoli sématická segmentace masky, model Mask R-CNN byl proto vyřazen. Zbylé modely z rodiny R-CNN byly taktéž vyřazeny, protože míra jejich přesnosti a rychlost detekce byly překonány novějšími modely. Podobně je tomu i u architektury SSD. Pro praktické řešení této práce byl zvolen model YOLOv5, protože se počáteční vydání se jeví jako velmi rychlé, výkonné a snadno použitelné. V době psaní této práce se jedná o nejnovější detekční algoritmus, který navíc nabízí čtyři předtrénované modely různých velikost. Za účelem praktické řešení diplomové práce byly vybrány modely YOLOv5s a YOLOv5m, jelikož trénování probíhalo na konvenční grafické kartě.

3.2 Použité nástroje a zařízení

Pro trénování a testování detekčního modelu YOLOv5 bylo zapotřebí použít různé nástroje a zařízení. V této sekci jsou tyto nástroje a zařízení blíže specifikovány.

3.2.1 Anaconda

Anaconda je bezplatná a open-source [32] distribuce programovacích jazyků Python a R pro vědecké výpočty. Cílem tohoto sofwaru je zjednodušení správy balíčků a jejich použití. Distribuce zahrnuje datové vědecké balíčky vhodné pro Windows, Linux a macOS. Distribuce je vyvíjena a udržována společnost Anaconda, Inc. Společnost v roce 2012 založili Peter Wang a Travis Oliphant [33]. Verze balíků v Anacondě jsou spravovány pomocí systému správy balíčků conda [34]. Tento správce balíčků byl vyčleněn jako samostatný balíček s otevřeným zdrojovým kódem, protože byl užitečný sám o sobě a i pro jiné účely, než jen jako distribuce Python [35]. Anaconda přichází s více než 250 balíčky s automatickou instalací a z PyPI (Python Package Index) lze nainstalovat více než 7500 dalších open-source balíčků, stejně jako balíček

conda a správce virtuálního prostředí. Vedle rozhraní příkazového řádku Anaconda také disponuje grafickým uživatelským rozhraním Anaconda Navigator [36].

3.2.2 PyTorch

PyTorch byl vydán výzkumnou laboratoří AI Facebooku na počátku roku 2017. Jedná se o open source knihovnu pro programy psané v Pythonu, která má za cíl usnadnění tvorby projektů strojového učení. Jádrem stroje pro hluboké učení je složitá, komplexní matematická funkce, mapující vstupy na výstup. Pro usnadnění vyjádření této funkce definuje PyTorch základní datovou strukturu tenzor. Tenzor je vícerozměrné pole, které sdílí mnoho podobností s poli NumPy. PyTorch disponuje funkcemi pro provádění zrychlených matematických operací na vyhrazeném hardwaru. Hlavní výhodou PyTorch je silná podpora zrychlení výpočtů tenzorů na grafických kartách NVIDIA (balíček torch.cuda) [37, 38]. PyTorch framework umožňuje flexibilně a snadno vyvíjet modely hlubokého učení. S rámcem PyTorch je možné využívat balíčky Pythonu, jako jsou SciPy, NumPy atd. Nejčastější příklady užití PyTorch jsou klasifikaci obrázků a strojový překlad [39].

3.2.3 Hardware pro trénování YOLOv5

Trénování modelu YOLOv5 bylo provedeno na PC, jehož přesná specifikace je uvedena níže v tabulce 3.1.

Tab. 3.1: Technické parametry sestavy použité pro trénování modelu YOLOv5.

CPU	AMD Ryzen 7 3700X 8 jader, 16 vláken
GPU	NVidia ASUS GeForce RTX 2070 SUPER 2560 CUDA jader, 8 GB GDDR6 paměť
RAM	Patriot Viper 4 64 GB (2 x 32 GB) DDR4 3200 MHz CL16
PSU	EVGA 750 W GQ

Nejdůležitějším komponentem celé PC sestavy je grafická karta NVidia GeForce RTX 2070 SUPER, která byla uvedena na trh v roce 2019. Grafické jádro je postaveno na architektuře Turing. NVidia GeForce RTX 2070 SUPER disponuje 2560 výpočetními jednotkami označovanými jako CUDA cores. Dále je karta vybavena speciálními jádry Tensor Cores, které slouží k výpočtu RayTracingu [40]. Na těchto speciálních jádrech je možné akcelrovat výpočty matic [41]. Model YOLOv5 byl nativně implementován v PyTorch [29], z toho důvodu je zapotřebí jako grafickou kartu pro trénování tohoto modelu použít jeden z produktů od společnosti NVidia, na kterých je možné tento framework akcelrovat [39].

Další důležitou komponentou počítačové sestavy pro trénování detekční architektury YOLOv5 jsou paměti RAM. YOLOv5 umožňuje zrychlení trénování díky možnosti cachování obrázků do pamětí RAM, a tudíž je zapotřebí vysoká paměť RAM. Původní G.SKill TridentZ RGB 32 GB (4 x 8GB) DDR4 3200 CL16 byly vyměněny za Patriot Viper 4 64 GB (2 x 32 GB) DDR4 3200 MHz CL16, jelikož původní kapacita 32 GB nebyla dostatečná. Při trénování středního modelu YOLOv5m za použití finální datové množiny (velikost obrázků 864 x 486 pixelů, batch 16) bylo průměrně využito 56 GB z dostupných 64 GB.

3.2.4 Zařízení pro záznam

Pro řešení této diplomové práce nebyla použita pouze data z volně dostupných zdrojů, ale byla také vytvořena vlastní datová množina. Pro sběr dat byl použit stativ Velbon Sherpa 250R s digitálním fotoaparátem Nikon D3100, který podporuje nahrávání videa. Za pomocí fotoaparátu připevněného na stativu byly nahrávány statické záběry. Video záznamy byly pořízeny na mostech a pěších lávkách nad pozemními komunikacemi na několika místech v Praze a v Brně. Videa byla natočena v rozlišení Full HD (1920×1080 pixelů) při 24 snímcích za sekundu a v rozlišení HD (1280 x 720 pixelů) pří 30 snímcích za sekundu. V součtu bylo nahráno přes 7 hodin video záznamu.

3.2.5 Computer Vision Annotation Tool

Computer Vision Annotation Tool (CVAT) je bezplatný open source anotační nástroj, který byl vyvinutý společností Intel Corporation. CVAT je možné použít pro anotaci digitálních obrázků a videí. Po jednoduché implementaci nástroje pomocí Dockeru je CVAT snadno přístupný přes rozhraní webového prohlížeče (konkrétně je vyžadován Google Chrome), není nutná žádná další instalace. CVAT umožňuje uživatelům anotovat obrázky čtyřmi způsoby: za pomocí rámečků, polygonů (obecně i pro segmentační úkoly), křivky (které mohou být užitečné pro anotovaní značek na silnicích) a body (např. Pro anotaci orientačních bodů tváře nebo odhad pozice). Pro účely této diplomové práce bylo využito anotovaní pomocí rámečků, jelikož to vyžaduje použitý model YOLOv5. Velká výhoda nástroje je možnost využití automatické anotace videí, kdy je možné například použít interpolaci mezi klíčovými snímky. Demo anotačního nástroje CVAT je možné vyzkoušet na webové adrese www.cvat.org. CVAT umožňuje přímý export anotovaných dat ve formě datasetu YOLO, včetně vygenerováno YAML konfiguračního souboru [42, 43] .

3.2.6 Datumaro

Datumaro je komponentou anotačního nástroje CVAT. Tento CLI (Command Line Interface) nástroj je taktéž vyvíjen společností Intel Corporation. Je dostupný jako knihovna Pythonu. Tato komponenta slouží k analýze a správě datových množin pro počítačového vidění, podporuje širokou škálu formátů (MS COCO, PASCAL VOC, YOLO, atd.). Nástroj CVAT při exportu anotovaného videa rozdělí video na jednotlivé snímky. Tyto snímky poté defaultně exportuje ve formátu PNG, což má za následek značný nárůst velikosti exportované datové množiny oproti původnímu anotovanému videu. Nástroj Datumaro byl v rámci řešení této diplomové práce vyžit k exportu snímků anotovaných videí z CVAT ve formátu JPEG [44].

3.2.7 Weights & Biases

Weights & Biases (wandb) je softwarový nástroj, který slouží k logování výsledků experimentů strojového učení. Data z trénování neuronové sítě jsou ukládána na cloudový účet. Na cloudovém účtu je možné živé zobrazení metrik, protokolů terminálu a statistik systému. Data jsou vizualizována pomocí grafů. K implementaci Weights & Biases je zapotřebí pouze pět řádků kódu. Za nástrojem wandb stojí společnost Weights and Biases, Inc. Pro akademické a open source projekty je použití softwaru zdarma [45].

3.2.8 DaVinci Resolve 17

DaVinci Resolve je multifunkční softwarový nástroj, který kombinuje úpravy, korekci barev, vizuální efekty, pohyblivou grafiku a zvukovou postprodukci. Software je rozdělen na „karty“, z nichž každá slouží k jinému účelu. Úpravy se provádějí na Cut a Edit kartách, vizuální efekty a pohybová grafika na kartě Fusion, korekce barev na kartě Color, zvuk na kartě Fairlight a pro organizaci a výstup médií karta Deliver. Tento software je vyvíjen a distribuován společností Blackmagic Design. DaVinci Resolve je bezplatně dostupný, ale s omezenými funkcemi. Placenou verzí tohoto programu je DaVinci Resolve Studio [46].

3.2.9 Roboflow

Roboflow je webová aplikace, která poskytuje všechny nástroje potřebné k převodu nezpracovaných obrazů na vlastní trénovaný model počítačového vidění a jeho nasazení. Roboflow podporuje modely detekce a klasifikace objektů [47].

3.3 Datová množina Roboflow Vehicles

Pro prvotní pokusy trénování modelu YOLOv5 byla použita již existující datová množina. Jedná se o bezplatnou, veřejně přístupnou datovou sadu Roboflow Vehicles-OpenImages pro detekci vozidel vytvořenou 19. června 2020 [48]. Datová množina čítá celkem 627 anotovaných obrázků rozdělených do pěti objektových tříd. Objektové třídy včetně zastoupení v datasetu jsou uvedeny v tabulce 3.2.

Tab. 3.2: Zastoupení objektových tříd v datové sadě Roboflow Vehicles.

Třída objektu	Počet anotovaných objektů
Osobní automobil	651
Autobus	141
Motocykl	140
Nákladní automobil	136
Sanitní vůz	126

Z tabulky 3.2 je patrné, že jednotlivé třídy nejsou v datasetu zastoupeny rovnoměrně. To může mít negativní vliv na výslednou přesnost detekce natrénovaného modelu, kdy např. může model dosahovat vysoké přesnosti i přesto, že všechny objekty klasifikuje do jedné třídy, která je nejhojněji zastoupena. Tento problém je možné řešit redukcí snímku s nejčastěji anotovanými objekty, rozšířením datové množiny nebo úpravou vah pro jednotlivé třídy.

3.3.1 Formát dat

Všechny anotované snímky mají velkost 416 x 416 pixelů. Anotované obrázky byly rozděleny do jednotlivých kategorií pro trénování, validaci a testování. Toto rozdělení je zapsáno v tabulce 3.3.

Tab. 3.3: Rozdělení obrázků v datové sadě Roboflow Vehicles.

Data	Počet anotovaných obrázků
Trénovací	439
Validační	125
Testovací	63

Byl vytvořen YAML soubor `vehicle.yaml` , který definuje cestu k datům, počet tříd objektů a jejich názvy. Soubor `vehicle.yaml` vypadá následovně:

```
train: vehicle_data/images/train
val: vehicle_data/images/validation
```

```
nc: 5
names: ['Ambulance', 'Bus', 'Car', 'Motorcycle', 'Truck']
```

YOLOv5 požaduje, aby byla datová množina ve formátu darknet. Ke každému obrázku existuje textový soubor (label) se stejným názvem. Každý objekt, nacházející se na obrázku, je zapsán na jeden řádek. Souřadnice boundingboxu musí být normalizovány mezi 0 a 1. Formát zápisu:

```
class x_center y_center width height
```

Obr. 3.1: Ukázka obrázku z datasetu Roboflow Vehicles.

Label ke snímku z obrázku 3.1 vypadá takto:

```
1 0.747596153846 0.507211538461 0.4591346153846 0.801682692307692
1 0.33533653846153 0.484375 0.46033653846153844 0.7403846153846154
```

3.4 Vlastní datová sada

Tato vytvořená datová množina obsahuje 1650 snímků rozdělených do tříd: osobní automobil, autobus, motocykl, nákladní automobil a dodávkový automobil. Sběr dat probíhal na třech místech v Brně. Konkrétně se jedná o lávku přes čtyřproudovou komunikaci Hradecká, poblíž Technologického Parku Brno a.s., lávku vedoucí přes dálnici D1 v Tvarožné a pěší lávku vedoucí z Galerie Vaňkovka přes ulici Zvonařka. Na pěší lávky byl umístěn digitální fotoaparát Nikon D3100 připevněný na stativ, který nahrával videa v rozlišení HD při 30 FPS a FullHD při 24 FPS. Pořízená videa byla sestříhaná pomocí softwaru DaVinci Resolve 17 tak, aby ve výsledných videích byly zastoupeny všechny sledované objektové třídy. Veškerá získaná data pro tvorbu vlastních datových množin byla anotována pomocí bezplatného open source anotační nástroje CVAT. Export datové množiny z CVAT byl proveden pomocí

CLI komponenty Datumaro. Objektové třídy včetně zastoupení v datové sadě jsou uvedeny v tabulce 3.4.

Tab. 3.4: Zastoupení objektových tříd ve vlastní datové sadě.

Třída objektu	Počet anotovaných objektů
Osobní automobil	2613
Autobus	384
Nákladní automobil	189
Dodávkový automobil	689
Motocykl	51
Kamion	433

Z tabulky 3.4 je zřejmé, že jednotlivé třídy nejsou ve vlastní datové sadě zastoupeny rovnoměrně, podobně jako tomu je u volně dostupné datové množiny Roboflow Vehicles. Sběr dat pro tuto datovou sadu byl prováděn během ledna a února, je tudíž logické, že objektovou třídou s nejmenším výskytem je motocykl, kterých v tomto chladném období jezdí minimum. Nerovnoměrné zastoupení objektových tříd se může opět negativně projevit na výsledné přesnosti detekce natrénovaného modelu. Možné důsledky a jejich řešení řešení byly popsány výše 3.3.

3.4.1 Formát dat

Z důvodu nízké paměti grafické karty 8 GB, byly veškeré anotované snímky upraveny na velikost 1280 x 720 pixelů. Anotované obrázky byly pomocí webové aplikace Roboflow rozděleny do dvou skupin, a to 1170 do trénovací a 480 do validační skupiny. Výskyt jednotlivých objektových tříd v trénovací a validační sadě je zapsán v tabulce 3.5.

Tab. 3.5: Vlastní datová sada – výskyt objektových tříd ve skupinách.

Třída objektu	Trénovací sk.	Validační sk.
Osobní automobil	1860	753
Autobus	274	110
Nákladní automobil	135	54
Dodávkový automobil	471	218
Motocykl	37	14
Kamion	315	118

Žádná data z této sady nebyla přidělena do testovací množiny. Pro testování budou použita data z finální datové množiny. Použitím testovacích dat z finální da-

tové sady pro všechny datasety bude docíleno přímého porovnání úspěšnosti detekce vozidel.

Obr. 3.2: Ukázka obrázku z vlastní datové sady.

3.5 Vlastní rozšířená datová sada

Tato datová množina vznikla za účelem redukce nevyváženosti zastoupení jednotlivých objektových tříd v původním datasetu. Kvůli uzavření okresů během pandemické situace nebylo možné v tomto období natáčet mimo katastrální území obývané obce. Z tohoto důvodu byly pro rozšíření vlastní datové množiny použity fotky z internetu.

Majoritní část původní datové sady je tvořena snímky s osobními automobily, z důvodu snahy o lepší vyvážení objektových tříd byl původní dataset zredukován z 1650 na 1180 snímků. Veškeré nepoužité obrázky z původního vlastní datové sady zachycovaly pouze osobní automobily. Zredukovaná datová množina byla rozšířena o obrázky z internetu. Celkem tato rozšířená datová sada čítá 1731 anotovaných snímků.

Nevýhodou použitých snímků z internetu je úhel záběru, který je rozmanitý viz ukázkový obrázek 3.1. Nejedná se tedy vždy o záběr shora, jako je tomu u původní datové množiny. Tato skutečnost se může negativně projevit při detekci záběrů z dopravních kamer. Datová množina nebyla dostatečně rozšířena, aby byl zcela eliminován problém s nevyváženými objektovými třídami, nicméně rozdíl mezi zastoupením jednotlivých tříd není tak propastný, jako tomu je u prvního vlastního datasetu. Cílem této datové sady bylo, zda-li větší vyváženost objektových tříd spolu s použitými snímky vozidel z internetu dokáží pozitivně ovlivnit schopnost detekce vozidel v obraze.

Tab. 3.6: Zastoupení objektových tříd v rozšířené datové sadě.

Třída objektu	Počet anotovaných objektů
Osobní automobil	1830
Autobus	514
Nákladní automobil	222
Dodávkový automobil	705
Motocykl	210
Kamion	671

Obr. 3.3: Ukázka obrázku z rozšířené datové sady, převzato z [49].

3.5.1 Formát dat

Všechny rozšiřující anotované snímky byly upraveny na velikost 1280 x 720 pixelů. Datová množina byla opět pomocí webové aplikace Roboflow rozdělena na trénovací (1271 snímků) a validační část (460 snímků).

Tab. 3.7: Rozšířená datová sada – výskyt objektových tříd ve skupinách.

Třída objektu	Trénovací sk.	Validační sk.
Osobní automobil	1378	452
Autobus	344	170
Nákladní automobil	166	56
Dodávkový automobil	500	205
Motocykl	153	57
Kamion	543	128

3.6 Finální datová sada

Ve finální datové sadě byla použita část anotovaných obrázků z původní verze vlastní datové sady. Cílem při tvorbě této datové množiny byla vyváženost objektových tříd a použití pouze autorských snímků. Sběr dat probíhal na dvou místech v Praze. Pro sběr snímků autobusů byla použita pěší lávka přes ulici Bucharova, poblíž MHD zastávky Nušlova. Druhým místem sběru dat byl Pražský okruh, toto místo bylo vybráno za účelem pořízení snímků kamionů, nákladních automobilů a dodávek. Finální podoba vlastní datové sady byla rozšířena na 5404 anotovaných obrázků. Četnost výskytů sledovaných objektů v této datové množině je uvedeno v tabulce 3.8.

Tab. 3.8: Zastoupení objektových tříd ve finální datové sadě.

Třída objektu	Počet anotovaných objektů
Osobní automobil	1698
Autobus	942
Nákladní automobil	988
Dodávkový automobil	999
Motocykl	843
Kamion	1017

Pořízené záznamy ze sběru dat jen výjimečně obsahují vyvážený poměr všech sledovaných objektových tříd. Natočená videa bylo tedy nutné nejen vhodně sestříhat, ale také bylo zapotřebí kolem nedostatečně zastoupených objektů vytvořit masku aby nebylo navyšováno množství anotací nadměrně zastoupených tříd. Ukázku takto upraveného snímku je možné vidět na obrázku 3.4. Všechna videa byla sestříhána a následně vhodně editována v softwarovém nástroji DaVinci Resolve 17.

Obr. 3.4: Ukázka snímku z finální datové sady po aplikaci masky.

3.6.1 Formát dat

Videa byla nahrávána ve FullHD rozlišení při 24 FPS. Nahrávání videí v rozlišení FullHD bylo preferováno kvůli nižšímu počtu snímků za sekundu. Rozšiřující anotované snímky byly následně zmenšený na velikost 1280 x 720 pixelů, aby měly stejné rozlišení jako zbytek datové množiny. Anotované snímky byly ručně rozděleny do kategorií pro trénování (3658 snímků), validaci (1120 snímků) a testování (626 snímků). Rozdělení snímků do skupin bylo provedeno tak, aby se snímky z jednotlivých videí nevyskytovaly ve více skupinách. Bylo tak předcházeno situaci, kdy se konkrétní sledované vozidlo mohlo ocitnout např. v trénovací i validační skupině.

Tab. 3.9: Finální datová sada – výskyt objektových tříd ve skupinách.

Třída objektu	Trénovací sk.	Validační sk.	Testovací sk.
Osobní automobil	1213	397	88
Autobus	684	150	108
Nákladní automobil	684	210	94
Dodávkový automobil	676	203	120
Motocykl	495	206	122
Kamion	690	191	136

Testovací množina finální datové sady byla použita pro porovnání úspěšnosti detekce vozidel všech použitých datasetů. Testovací množina byla vytvářena s důrazem na vyváženost objektových tříd. Množina obsahuje záběry jak předních, tak i zadních částí všech sledovaných objektů. Díky těmto vlastnostem testovací sady bude možné korektně evaluovat výstupy testovaní použitých datových množin.

3.7 Postup

3.7.1 Nastavení prostředí a instalace součástí YOLOv5

Aby bylo možné začít trénovat a testovat model YOLOv5, je zapotřebí nejdříve naklonovat GitHub repozitář yolov5 od Ultralytics (repozitář je možno stáhnout zde: [29]). V této diplomové práci byla použita verze z 21. dubna 2021. V neposlední řadě je nutné nainstalovat další potřebné softwarové součásti, které jsou zapsány v textovém souboru `requirements.txt`.

Všechny tyto součásti byly nainstalovány do virtuální prostředí Anaconda. K ukládání logů na cloudový účet byl použit nástroj wandb, který je implentován v kódu detekčního modelu YOLOv5.

K zaházejí trénování je potřeba spustit pythonovský skript `train.py`, který má tyto parametry:

- img: definuje velikost vstupního obrázku,
- batch: určuje velikost batch,
- epochs: definuje počet tréninkových epoch,
- data: nastavení cesty k yaml souboru,
- cfg: upřesňuje, který model YOLOv5 bude použit,
- weights: nastavení cesty k vahám,
- name: název adresáře s výsledky,
- nosave: uložit pouze poslední kontrolní bod,
- cache: obrázky budou uloženy do mezipaměti pro rychlejší trénink.

Pro dosažení lepších výsledků je autory YOLOv5 doporučeno použití předtrénovaných vah `yolov5s.pt`, při zadání parametru `--weights ''` jsou použity náhodně inicializováné váhy [29].

Pro účely vyhodnocení natrénované přesnosti detekce modelu na validačních nebo trénovacích datech slouží pythonovský skript `test.py`. Skript `test.py` je volán ve skriptu `train.py` vždy po dokončení epochy, za účelem vyhodnocení změny vah. Skript má následující parametry:

- img: definuje velikost vstupního obrázku,
- batch: určuje velikost batch,
- data: nastavení cesty k yaml souboru,
- weights: nastavení cesty k vahám,
- name: název adresáře s výsledky,
- iou: definice velikosti IoU (defaultní hodnota 0,6),
- conf: práh spolehlivosti odhadu (defaultní hodnota 0,001),
- save-txt: uloží anotace obrázků ve formátu txt,
- save-json uloží anotace obrázků ve formátu json,
- verbose: ukázat výsledky pro jednotlivé třídy.

Pomocí `detect.py` skriptu je možné detekovat objekty na obrázcích nebo videích libovolných rozměrů. Skript `detect.py` má následující parametry:

- iou: definice velikosti IoU,
- save-txt: uloží anotace do textového souboru,
- save-json uloží anotace obrázků ve formátu json,
- source: definuje cestu k souborům,
- weights: nastavení cesty k vahám,
- classes: omezení na třídy, které budou detekovány,
- iou: definice velikosti IoU (defaultní hodnota 0,6),
- conf: práh spolehlivosti odhadu (defaultní hodnota 0,001),
- output: definuje cestu pro výstup.

4 Výsledky

Tato kapitola je věnována diskuzi průběhů trénování a dosažených výsledků. Sledovanými parametry jsou mAP s IoT (Intersection over Union) 0,5 ,preciznost (precision) a úplnost (recall) predikce objektů. Tyto parametry nejvíce vypovídají o průběhu trénování konvolučních neuronových sítí.

4.1 Trénování – datová množina Roboflow Vehicles

Pro prvotní pokusy trénování neuronové sítě byl použit pythonovský skript `train.py` a byl zvolen nejmenší model YOLOv5s. Pro dosažení co nejlepších výsledků byly využity předtrénované váhy `yolov5s.pt`. Parametr určující maximální počet současně zpracovávaných obrázků (batch) byl nastaven na nejvyšší možnou hodnotu 64. Tato nejvyšší možná hodnota je vždy omezena fyzickou pamětí grafické karty. V rámci řešení diplomové práce byla použita konvenční grafická karta Nvidia RTX 2070 SUPER s pamětí 8 GB .

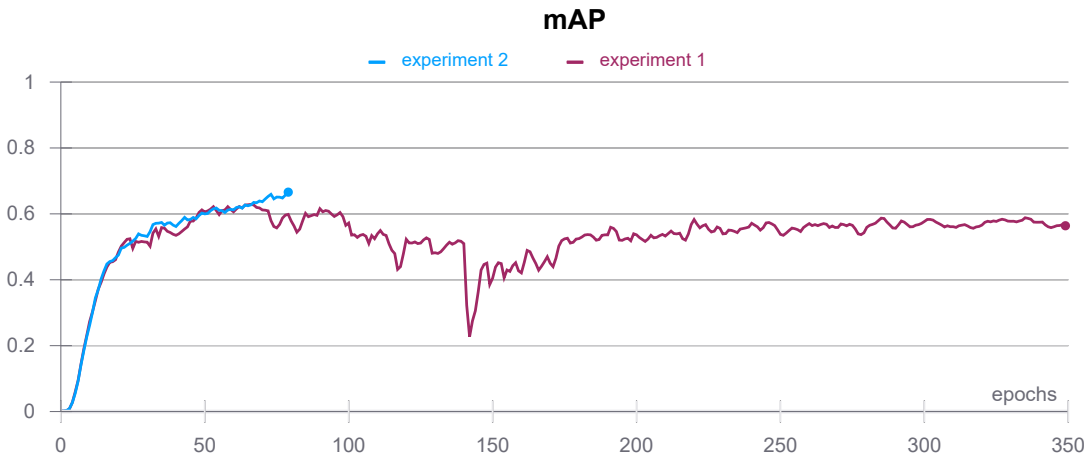
Pro experiment 1 probíhalo trénování neuronové sítě po dobu 350 epoch. Z tohoto experimentu bylo patrné, že při překročení 80 epoch dochází k přetrénování modelu. Z toho důvodu byl počet trénovacích epoch pro experiment 2 omezen na 80.

Tab. 4.1: Hodnoty sledovaných metrik při trénování pro experiment 2.

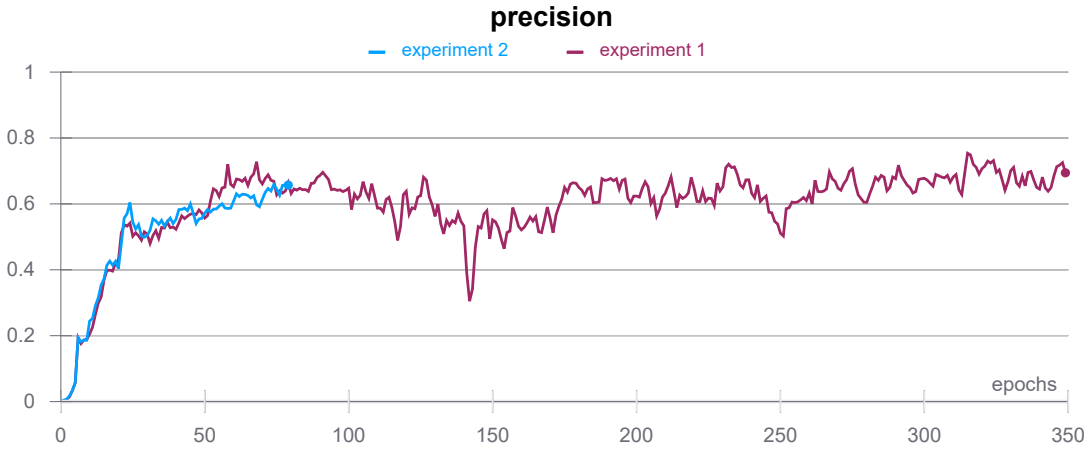
Metrika	Hodnota
mAP	0,677
Precision	0,657
Recall	0,697

Třída	mAP
Sanitka	0,852
Autobus	0,876
Os. automobil	0,554
Motocykl	0,768
Nákladní auto.	0,333

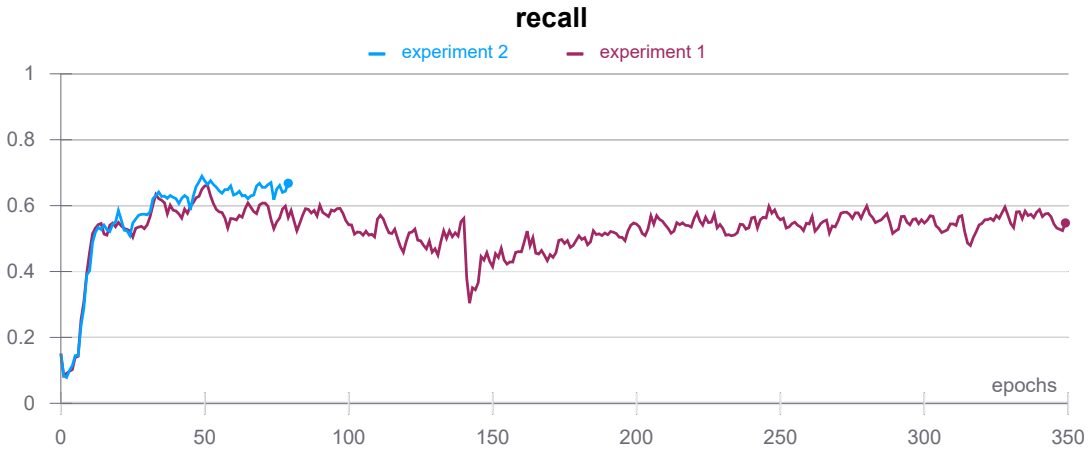
Přetrénování je patrné z grafů 4.1, 4.2 a 4.3, kde je u experimentu 1 jasně patrný pokles mAP, úplnosti a preciznosti predikce. U experimentu 2 bylo dosaženo následujících hodnot: Výsledné hodnoty sledovaných metrik, kterých bylo dosaženu u experimentu 2, jsou zapsány v tabulce 4.1.



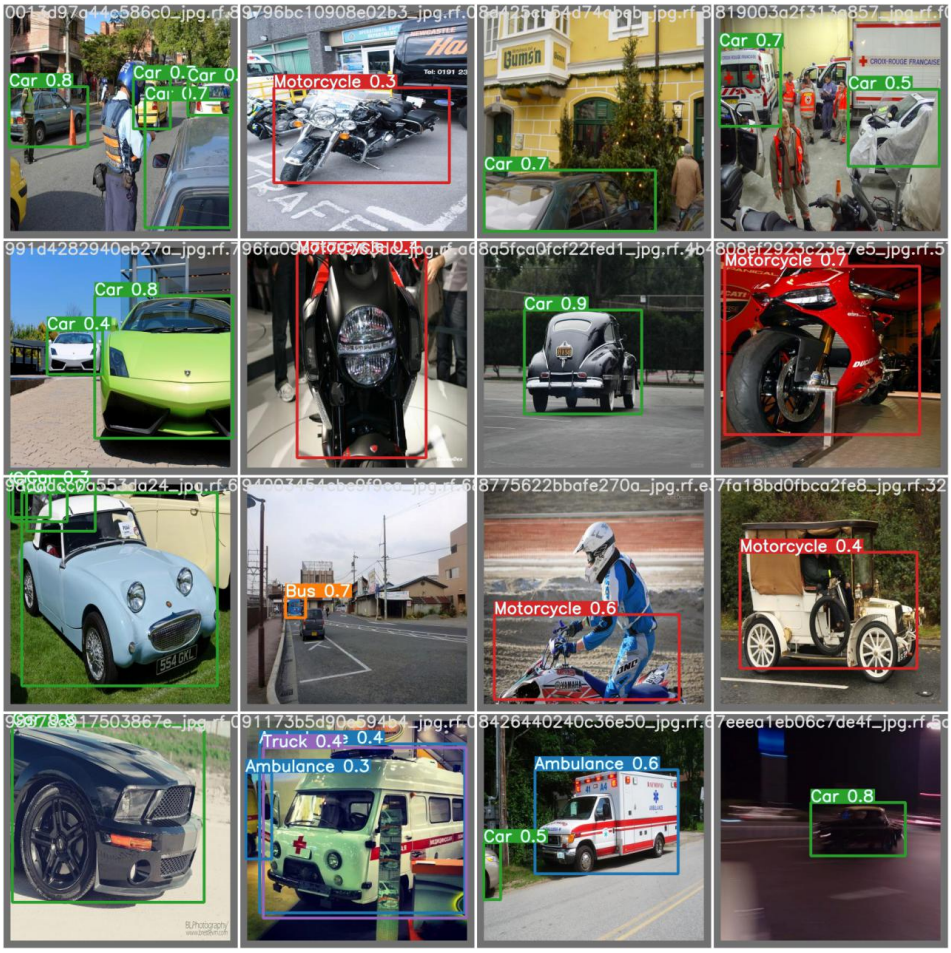
Obr. 4.1: Průběh mAP při trénování modelu YOLOv5s.



Obr. 4.2: Průběh preciznosti při trénování modelu YOLOv5s.



Obr. 4.3: Průběh úplnosti predikce při trénování modelu YOLOv5s.



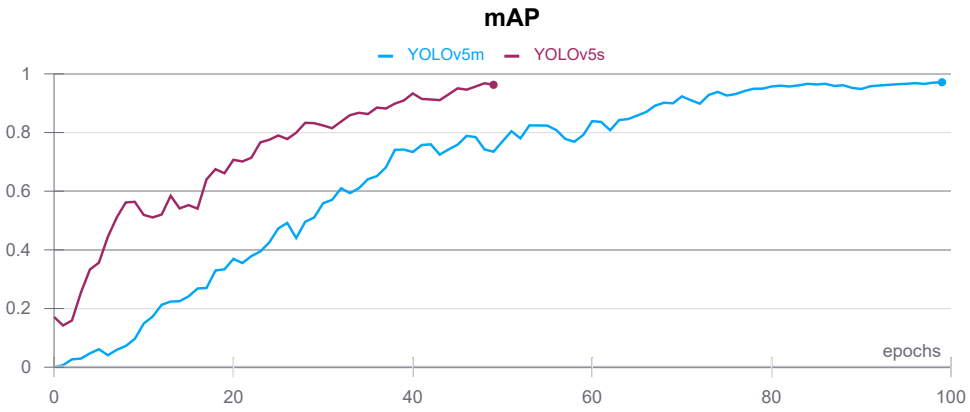
Obr. 4.4: Ukázka detekce objektů na množině validačních obrázků.

4.2 Trénování – vlastní datová množina

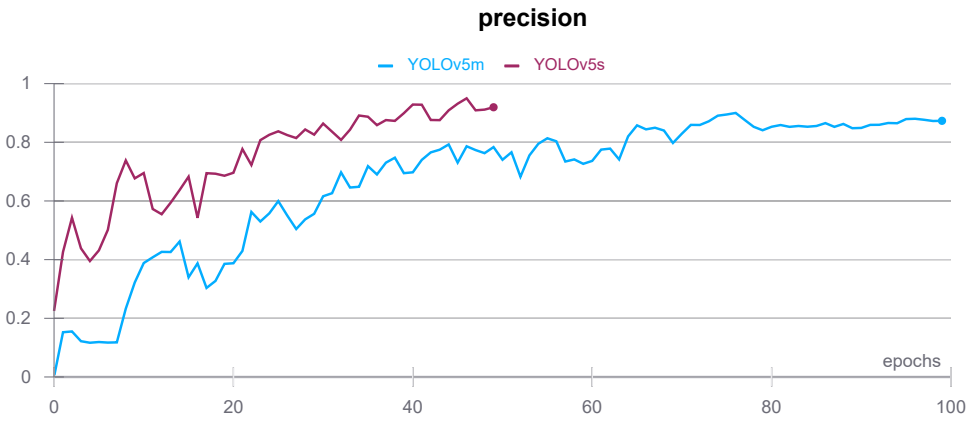
Za použití vlastní datové sady byly trénovány modely YOLOv5s a YOLOv5m. Při HD (1280 x 720 pixelů) rozlišení anotovaných snímků byla použita nejvyšší přípustná velikost parametru batch 16 pro model YOLOv5s a batch 8 pro model YOLOv5m. U menšího modelu probíhalo trénování po dobu 50 epoch, u modelu YOLOv5m byl počet epoch dvojnásobný.

Tab. 4.2: Vlastní datová sada – výsledné metriky při trénování.

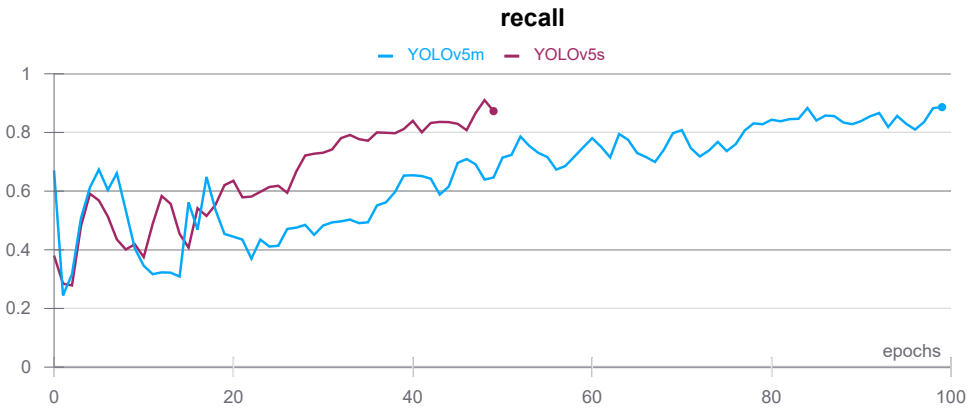
Metrika	YOLOv5s	YOLOv5m
mAP	0,9563	0,9729
Preciznost	0,8272	0,8904
Úplnost	0,9292	0,8739



Obr. 4.5: Průběh mAP při trénování YOLOv5s a YOLOv5m.



Obr. 4.6: Průběh preciznosti při trénování YOLOv5s a YOLOv5m.



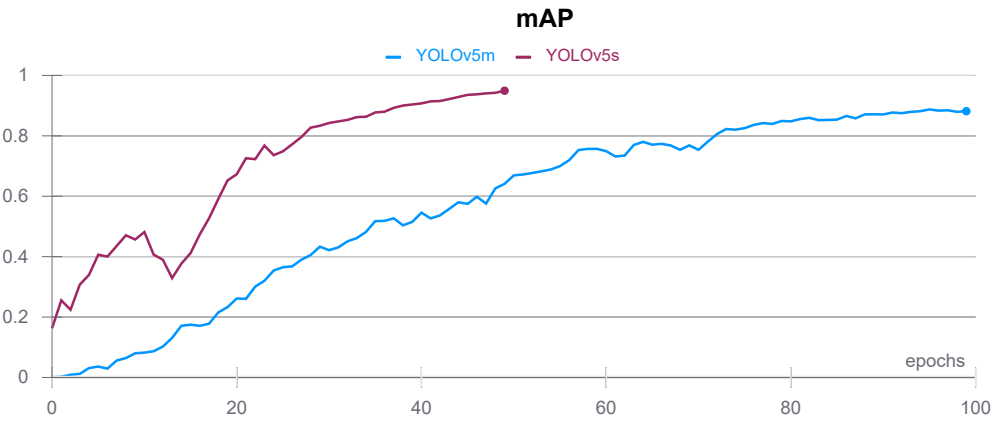
Obr. 4.7: Průběh úplnosti predikce při trénování YOLOv5s a YOLOv5m.

4.3 Trénování – rozšířená datová množina

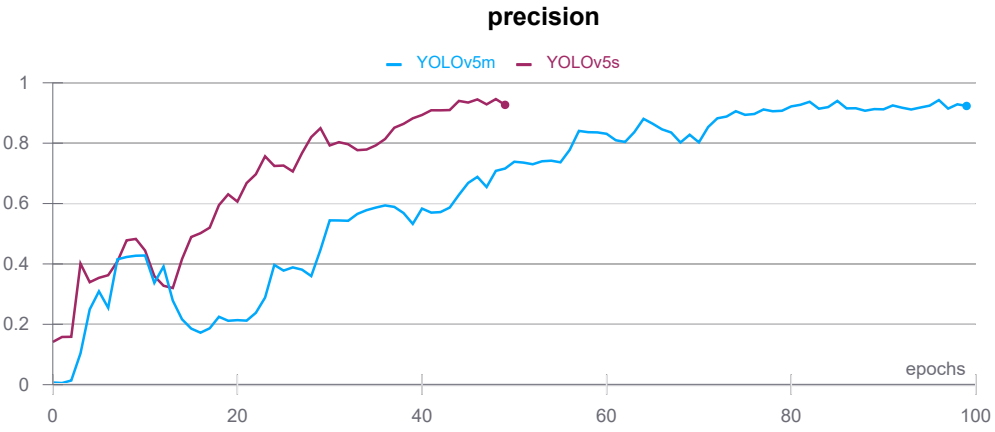
Postup trénování byl totožný jako u trénování vlastní datové množiny. Byla použita stejná hodnota parametru určujícího maximální počet současně zpracovávaných obrázků a shodný počet trénovacích epoch viz 4.2.

Tab. 4.3: Rozšířená datová sada – výsledné metriky při trénování.

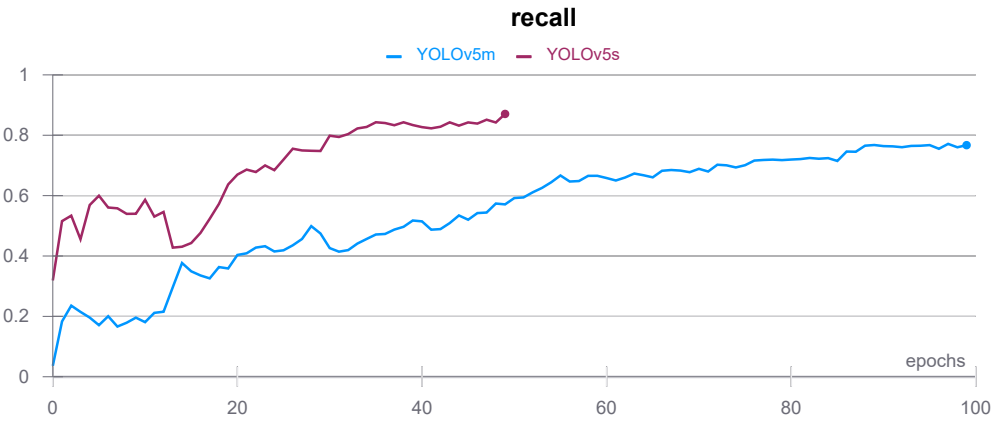
Metrika	YOLOv5s	YOLOv5m
mAP	0,9577	0,8841
Preciznost	0,9037	0,9166
Úplnost	0,9049	0,7757



Obr. 4.8: Průběh mAP při trénování YOLOv5s a YOLOv5m.



Obr. 4.9: Průběh preciznosti při trénování YOLOv5s a YOLOv5m.



Obr. 4.10: Průběh úplnosti predikce při trénování YOLOv5s a YOLOv5m.

4.4 Trénování – finální datová množina

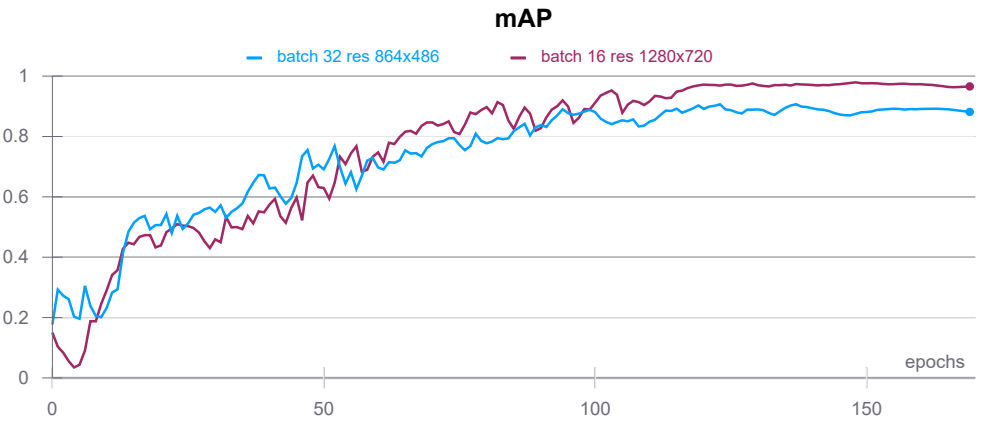
Tato podkapitola je rozdělena do dvou částí, z níž první část je věnována trénování nejmenšího modelu YOLOv5s, druhá část popisuje trénování modelu YOLOv5m. S pomocí finální datové množiny byly vykonány celkem čtyři experimenty trénování detektoru YOLO.

4.4.1 Model YOLOv5s

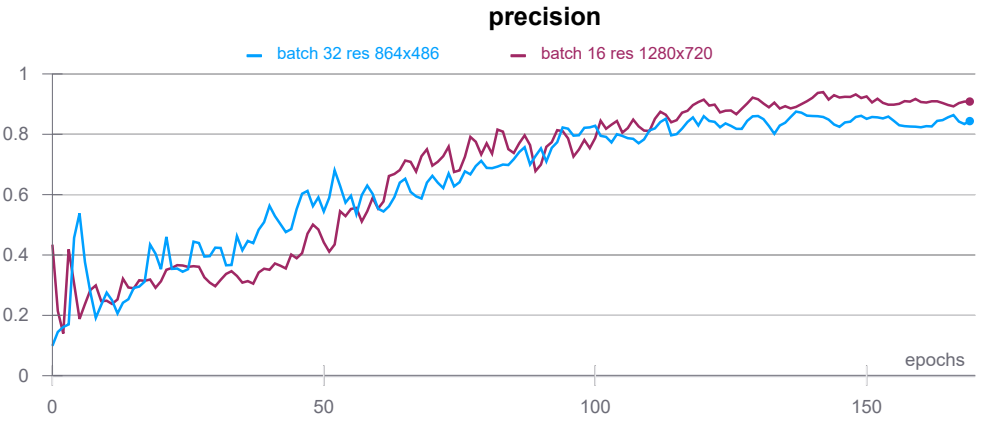
Za použití finální datové sady byly provedeny experimenty s dvojí velikostí snímků. Cílem experimentu byly zjistit, zda-li případně jaký vliv má rozlišení anotovaných snímků na výslednou přesnost detekčního modelu YOLO. Pro první experiment bylo rozlišení snímků sníženo na 864 x 486 pixelů a hodnota batch byla nastavena na maximální hodnotu 32. Pro druhý průběh trénování bylo zachováno originální rozlišení datové sady (1280 x 720 pixelů), parametr určující maximální počet současně zpracovávaných snímků byl opět nastaven na maximální přípustnou hodnotu 16, která je omezena pamětí grafické karty.

Metrika	864 x 486	1280 x 720
mAP	0,8786	0,9672
Preciznost	0,8555	0,9086
Úplnost	0,8264	0,8984

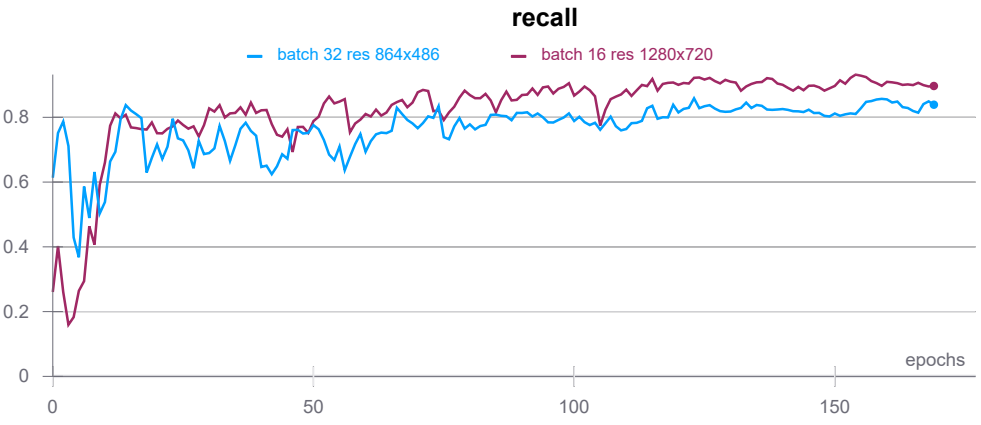
Tab. 4.4: Finální datová sada – výsledné metriky při trénování YOLOv5s.



Obr. 4.11: Průběh mAP při trénování modelu YOLOv5s.



Obr. 4.12: Průběh preciznosti při trénování modelu YOLOv5s.



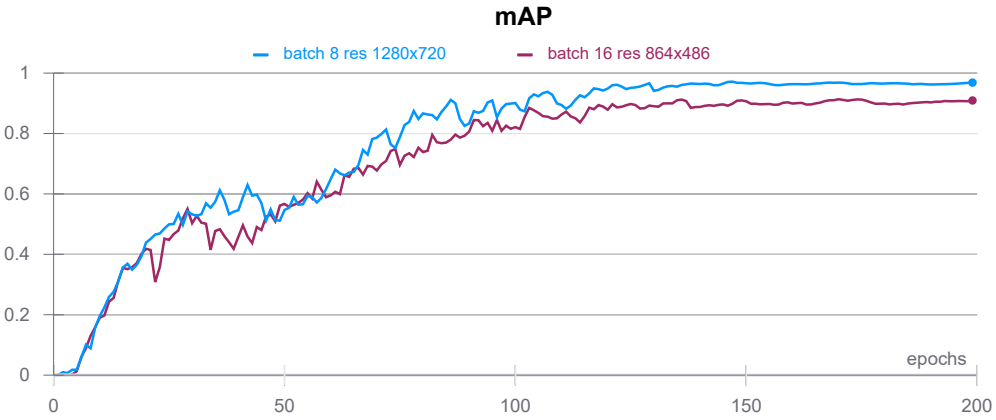
Obr. 4.13: Průběh úplnosti predikce při trénování modelu YOLOv5s.

4.4.2 Model YOLOv5m

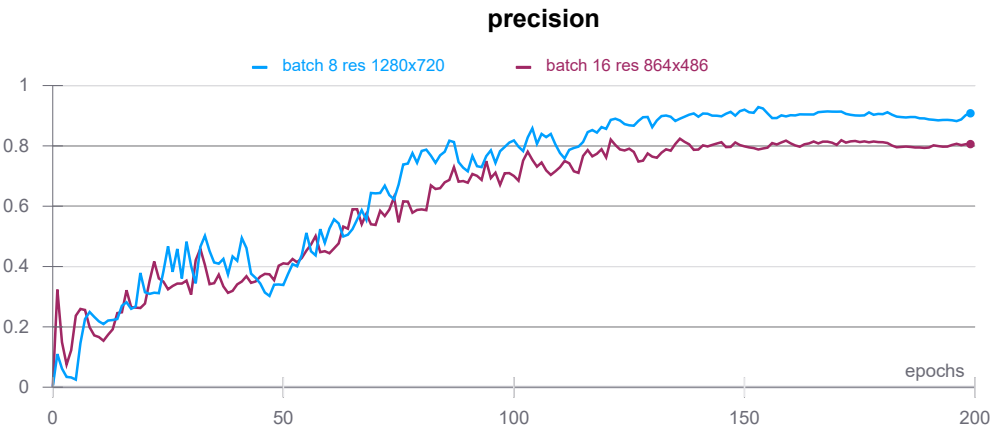
Postup trénování modelu YOLOv5m byl obdobný jako u menšího modelu. Byly taktéž provedeny dva experimenty průběhu trénování s různou velikostí obrázků. Rozdílné byly pouze hodnoty parametru batch, pro originální rozlišení nabýval tento parametr hodnoty 8, pro snížené rozlišení pak 16.

Metrika	864 x 486	1280 x 720
mAP	0,9121	0,9696
Preciznost	0,8061	0,9151
Úplnost	0,9041	0,9066

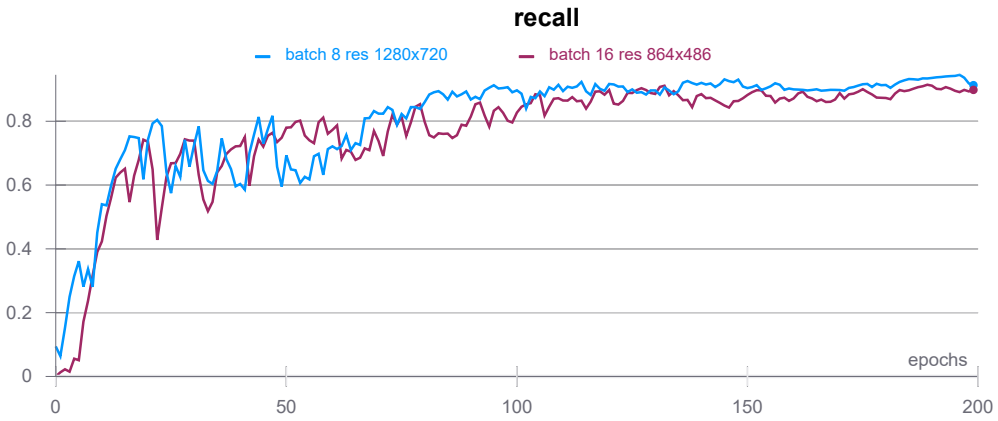
Tab. 4.5: Finální datová sada – výsledné metriky při trénování YOLOv5m.



Obr. 4.14: Průběh mAP při trénování modelu YOLOv5m.



Obr. 4.15: Průběh preciznosti při trénování modelu YOLOv5m.



Obr. 4.16: Průběh úplnosti predikce při trénování modelu YOLOv5m.

4.5 Testování – datová množina Roboflow Vehicles

Pro fázi testování neuronové sítě byl použit pythonovský `test.pt`, pro zobrazení výsledků úspěšnosti detekce jednotlivých objektových tříd byl využit nepovinný parametr `--verbose` . Výsledky testování jsou zapsány níže v tabulce 4.6.

Tab. 4.6: Výsledné metriky testování experimentu 2.

Třída	mAP	Preciznost	Úplnost
Celkově	0,740	0,722	0,688
Sanitka	0,963	0,809	1,000
Autobus	0,866	0,865	0,789
Osobní aut.	0,652	0,651	0,640
Motocykl	0,599	0,685	0,599
Nákladní aut.	0,620	0,598	0,620

4.6 Testování – vytvořené datové množiny

V této podkapitole jsou zapsány výsledky testování natrénovaného detektoru YOLOv5 za použití vlastních datových sad. Veškerá testovací data pochází z finální datové sady, tato vlastnost umožnila přímé porovnaní jednotlivých výsledků trénování. Testovací sada čítá celkem 626 snímků s rovnoměrným zastoupením konkrétních objektových tříd. U testovaných snímků bylo zachováno původní rozlišení 1280 x 720 pixelů, za účelem dosažení nejvyšší možné úspěšnosti detekce. U pythonovského skriptu `test.pt` pro testování byly parametry IoU a práh spolehlivosti odhadu ponechány na defaultních hodnotách (60,0 % a 0,1 %).

4.6.1 Vlastní datová množina

Tab. 4.7: Výsledné metriky testování natrénovaného modelu YOLOv5s.

Třída	mAP	Preciznost	Úplnost
Celkově	0,790	0,628	0,825
Osobní aut.	0,995	0,509	1,000
Autobus	0,938	0,796	1,000
Nákladní aut.	0,428	0,405	0,819
Dodávkový aut.	0,766	0,572	0,933
Motocykl	0,841	0,555	0,893
Kamion	0,770	0,993	0,306

Tab. 4.8: Výsledné metriky testování natrénovaného modelu YOLOv5m.

Třída	mAP	Preciznost	Úplnost
Celkově	0,432	0,612	0,399
Osobní aut.	0,991	0,281	1,000
Autobus	0,687	0,727	0,676
Nákladní aut.	0,478	0,405	0,345
Dodávkový aut.	0,246	0,572	0,267
Motocykl	0,167	0,720	0,107
Kamion	0,023	1,000	0,000

4.6.2 Rozšířená datová množina

Tab. 4.9: Výsledné metriky testování natrénovaného modelu YOLOv5s.

Třída	mAP	Preciznost	Úplnost
Celkově	0,836	0,769	0,872
Osobní aut.	0,995	0,976	1,000
Autobus	0,995	0,969	1,000
Nákladní aut.	0,848	0,405	0,905
Dodávkový aut.	0,862	0,572	0,730
Motocykl	0,346	0,349	0,797
Kamion	0,970	0,552	1,000

Tab. 4.10: Výsledné metriky testování natrénovaného modelu YOLOv5m.

Třída	mAP	Preciznost	Úplnost
Celkově	0,834	0,745	0,789
Osobní aut.	0,932	0,941	0,739
Autobus	0,994	0,652	1,000
Nákladní aut.	0,958	0,405	0,697
Dodávkový aut.	0,798	0,572	0,765
Motocykl	0,464	0,464	0,691
Kamion	0,827	0,656	0,842

4.6.3 Finální datová množina

Tab. 4.11: Výsledné metriky testování natrénovaného modelu YOLOv5s za použití snímků o rozlišení 864 x 486 pixelů.

Třída	mAP	Preciznost	Úplnost
Celkově	0,993	0,958	0,974
Osobní aut.	0,995	0,994	1,000
Autobus	0,995	1,000	0,954
Nákladní aut.	0,985	0,774	1,000
Dodávkový aut.	0,988	0,982	0,903
Motocykl	0,996	0,997	1,000
Kamion	0,996	1,000	0,990

Tab. 4.12: Výsledné metriky testování natrénovaného modelu YOLOv5m za použití snímků o rozlišení 864 x 486 pixelů.

Třída	mAP	Preciznost	Úplnost
Celkově	0,995	0,926	0,978
Osobní aut.	0,995	0,983	1,000
Autobus	0,995	1,000	0,944
Nákladní aut.	0,994	0,616	1,000
Dodávkový aut.	0,993	0,961	1,000
Motocykl	0,995	0,999	1,000
Kamion	0,996	1,000	0,924

Tab. 4.13: Výsledné metriky testování natrénovaného modelu YOLOv5s za použití snímků o rozlišení 1280 x 720 pixelů.

Třída	mAP	Preciznost	Úplnost
Celkově	0,993	0,935	0,971
Osobní aut.	0,996	0,984	1,000
Autobus	0,996	0,994	1,000
Nákladní aut.	0,981	0,634	1,000
Dodávkový aut.	0,996	1,000	0,974
Motocykl	0,996	0,995	1,000
Kamion	0,995	1,000	0,853

Tab. 4.14: Výsledné metriky testování natrénovaného modelu YOLOv5m za použití snímků o rozlišení 1280 x 720 pixelů.

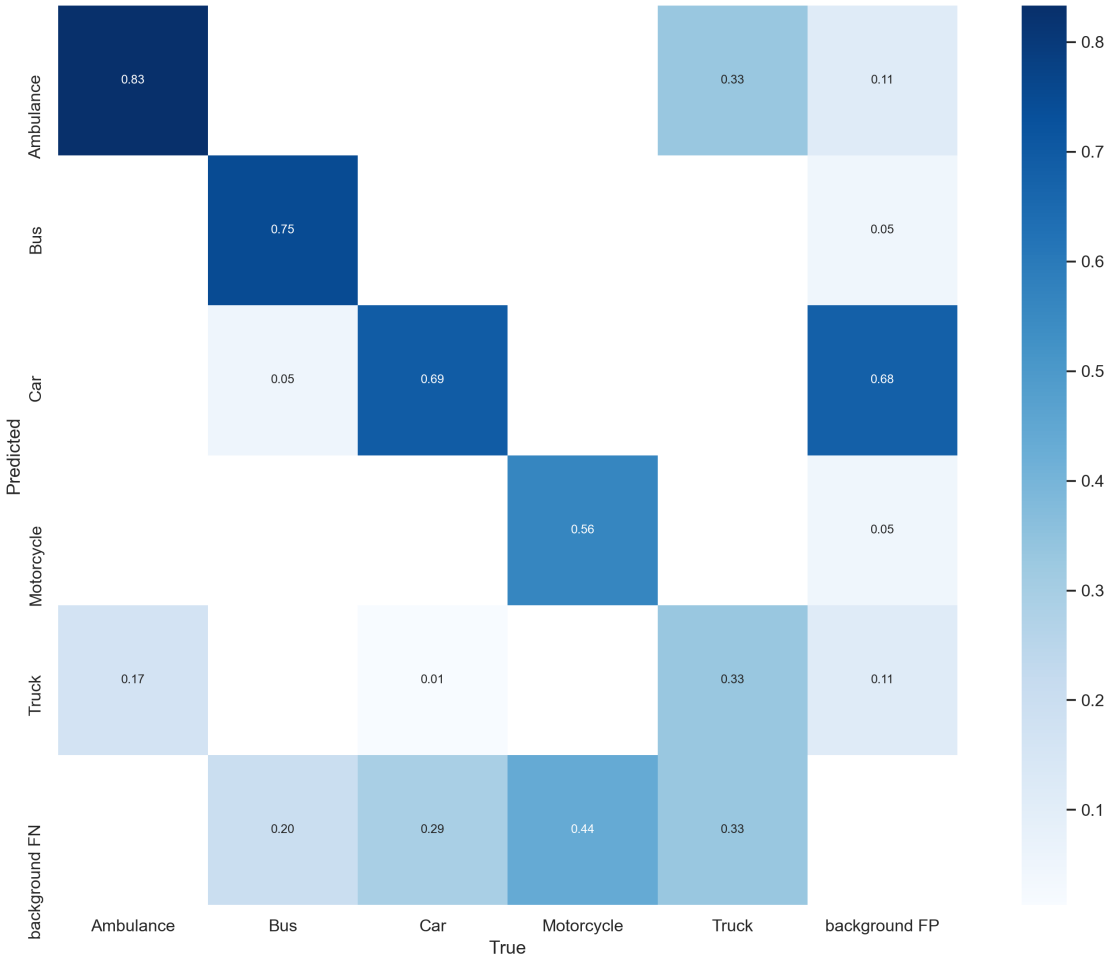
Třída	mAP	Preciznost	Úplnost
Celkově	0,995	0,995	0,986
Osobní aut.	0,995	0,998	1,000
Autobus	0,996	0,999	1,000
Nákladní aut.	0,995	0,989	1,000
Dodávkový aut.	0,995	0,994	0,975
Motocykl	0,996	1,000	1,000
Kamion	0,995	0,992	0,939

4.7 Diskuze dosažených výsledků

4.7.1 Datová množina Roboflow Vehicles

Z tabulky 4.1 a 4.6 můžeme vyčíst, že model YOLOv5s natrénovaný pomocí datové množiny Roboflow Vehicles-OpenImages dosahuje velmi nízkého hodnoty mAP 0,631 na validační a 0,740 na testovací sadě dat. Z matice záměn na obrázku 4.4 je patrné, že k nejvíce chybám dochází při záměně pozadí za osobní automobil, dále také při nedetekování osobního automobilu, motocyklu a nákladního automobilu na testovaných snímcích. Častou chybou je taktéž záměna sanitky za nákladní automobil, tato chyba mohla být způsobena podobným provedením karoserie u obou objektů. Příčinou tohoto problému se jeví nekvalitní datová sada, kdy jsou jednotlivé snímky rozdílně anotovány. Na části anotovaných snímků nejsou vytvořeny labely pro objekty v pozadí, když tyto neanotované objekty model správně detekuje, je

detekce vyhodnocena jako mylná. Mezi další problémy s použitou datovou sadou se řadí, že datová sada obsahuje nejen obrázky skutečných vozidel, ale také obrázky např. hraček, které jsou anotovány jako vozidla. Problematická je i třída nákladní automobil, do které byly autory datové množiny anotovány mimo nákladních automobilů také osobní automobily s provedením karoserie pick-up, které rozhodně do této třídy nepatří. V rámci řešení diplomové práce byla za účelem odstranění těchto problémů vytvořena vlastní datová sada ze záběrů dopravní komunikace s třídami: osobní automobil, autobus, dodávkový automobil, motocykl a kamion. Jasné rozdělení objektů do tříd a korektní anotace snímků eliminuje nedostatky tohoto datasetu.



Obr. 4.17: Matice záměn – datová množina Roboflow Vehicles.

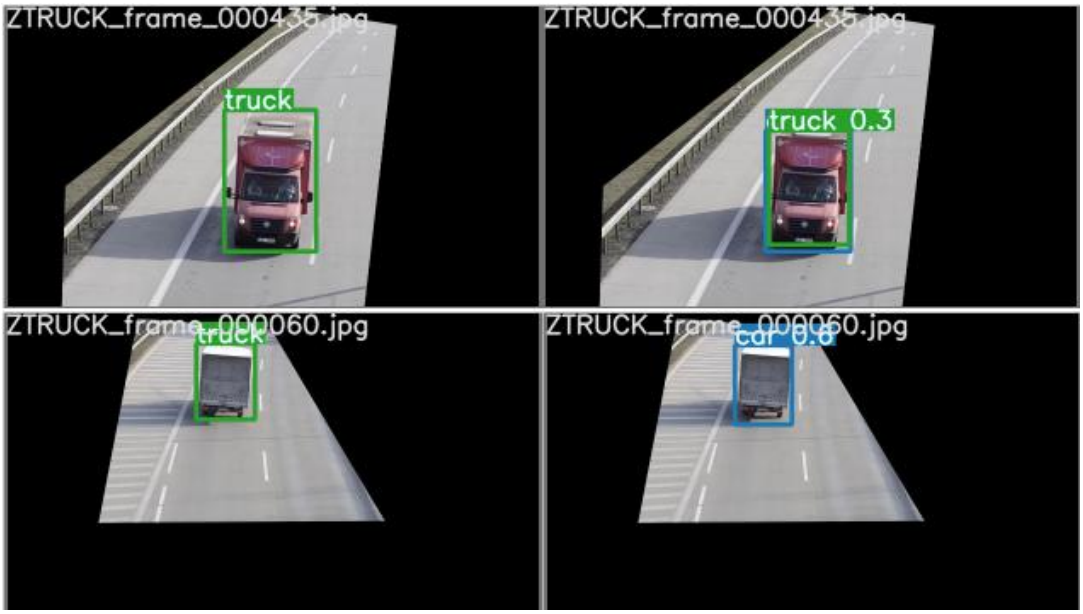
4.7.2 Vlastní datová množina

Z průběhů a výsledků trénování je patrné, že natrénovaný model YOLOv5m dosahuje mírně vyšší hodnoty mAP a preciznosti na validační sadě dat než menší model

YOLOv5s, který dosahuje vyšší úplnosti.

Výsledky testování jsou zcela odlišné od výsledných hodnot sledovaných metrik průběhu trénování. Model YOLOv5s dosáhl oproti původní hodnotě mAP 0,956 pouze 0,790. Veškeré sledované metriky úspěšnosti detekce dosáhly na testovací množině nižších hodnot oproti množině validační pro oba natrénované modely. Výsledky testování modelu YOLOv5m zaznamenaly ještě razantnější pokles až na mAP 0,432, přičemž objektová třída kamion nebyla v testovací sadě dat vůbec detekována.

Vlastní datová množina trpí nadměrným zastoupením objektové třídy osobní automobil, podobně jako tomu je u veřejně přístupné datové sady Roboflow Vehicles. Tento fakt se projevil zejména u modelu YOLOv5m, kdy většina detekovaných objektů byla mylně klasifikována jako osobní automobil viz obrázek 4.18.



Obr. 4.18: Ukázka detekce natrénovaného modelu YOLOv5m na testovací sadě.

Dalším faktorem, který negativně ovlivnil detekci na testovací množině, je způsob, kterým webová aplikace Roboflow rozdělila anotované snímky mezi validační a trénovací sadu. Data byla rozdělena v poměru 70 % na trénovací a 30 % na validační část. Snímky vznikly anotací videí, pořadí snímků nahraných do aplikace odpovídalo pořadí ve videosekvenci. Trajektorie konkrétního sledovaného vozidla tak byla přiřazena jak do skupiny pro trénování, tak do validační části. Výskyt snímků stejného vozidla v obou skupinách způsobil mimo jiné i nízký počet epoch potřebný k natrénování modelu. Natrénovaný model se tudíž naučil primárně detekovat pouze vozidla, která byla obsažena v trénovací a validační sadě. Tento negativní efekt je možné pozorovat na výsledcích detekce za použití testovacích dat pro oba natrénované modely.

4.7.3 Rozšířená datová množina

Výsledky trénování modelů YOLOv5m a YOLOv5s na rozšířené datové množině dopadly opačně než za použití vlastní datové množiny. Z grafů 4.8, 4.9 a 4.10 je možné vyčíst, že až na srovnatelnou preciznost dosáhl menší model YOLOv5s vyšší úplnosti predikce a hodnoty mAP.

Výsledky testování nedosahují u sledovaných metrik tak vysokých hodnot jako u trénování, avšak oproti vlastní datové sadě nedochází k tak výraznému poklesu viz tabulky 4.7, 4.8, 4.9 a 4.9. Oba natrénované modely dosáhly srovnatelných výsledků mAP (YOLOv5s 0,836, YOLOv5m 0,834) a preciznosti (YOLOv5s 0,769, YOLOv5m 0,745), z hlediska úplnosti predikce docílil lepšího výsledku model YOLOv5s.

Podobně jako u vlastní datové množiny bylo rozdělení anotovaných snímků provedeno za pomocí webové aplikace Roboflow, zastoupení jednotlivých objektových tříd taktéž není vyvážené. Důsledky těchto vlastností byly popsány výše v sekci 4.7.2. Motivací k vytvoření rozšířené datové sady bylo zjistit, zda-li větší vyváženost objektových tříd spolu s použitými snímky vozidel z internetu dokáží pozitivně ovlivnit schopnost detekce vozidel v obraze. Každý nově přidaný snímek zachycoval originální, dříve nepoužité vozidlo (případně vozidla). Toto rozšíření trénovací a validační sady dat o různá netotožná vozidla částečně omezilo negativní vliv nevhodného rozdělení snímků pomocí webové aplikace Roboflow. Nevýhodou rozšiřujících snímků z internetu je, že úhel záběru není totožný jako z dopravní kamery, a tudíž se tato vlastnost může negativně projevit při detekci na testovací sadě dat. I přes zmíněná negativa je možné pozorovat, že oproti modelům natrénovaným na vlastní datové množině došlo k výraznému zlepšení výsledků detekce na testovací sadě dat u obou trénovaných modelů. Na základě výsledků testování bylo jasné, že je třeba snímky mezi jednotlivé části korektně rozložit. Dalším nutným krokem bylo takové rozšíření datové sady, aby bylo docíleno vyváženosti objektových tříd.

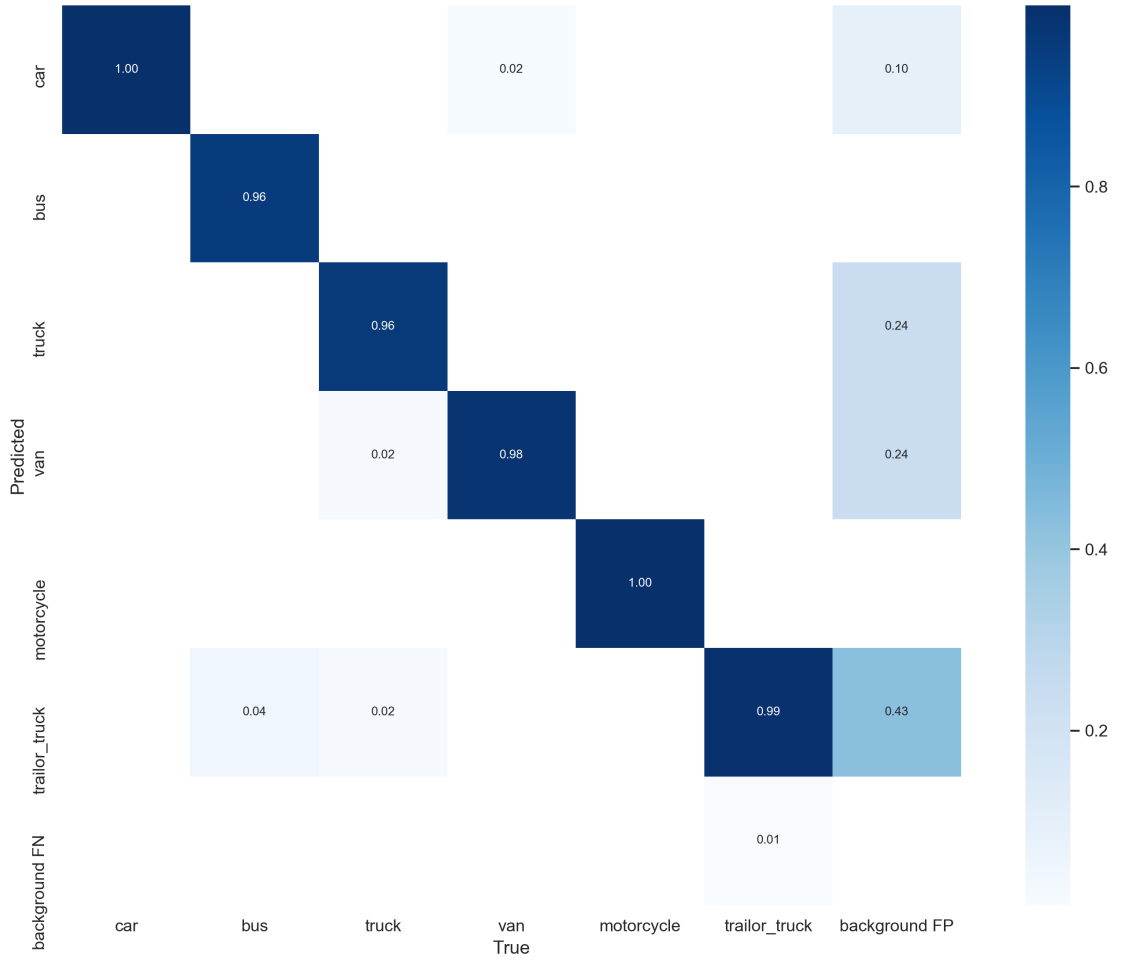
4.7.4 Finální datová množina

Finální datová množina byla vytvořena na základě výsledků předchozích experimentů. Problém s nevyváženými objektovými třídami byl vyřešen vhodným rozšířením datové sady. Veškeré obrázky z internetu byly nahrazeny vlastními sesbíranými daty. Toto nahrazení bylo provedeno za účelem předejití možných negativních důsledků, způsobených použitím snímků z jiných úhlů, než vozidla snímá dopravní kamera. Vyjma trénování modelů YOLOv5s a YOLOv5m na finální datové množině s originálním rozlišením 1280 x 720 pixelů, byly provedeny experimenty s redukovaným rozlišením 864 x 476 pixelů. Toto zredukované rozlišení umožnilo zvýšit hodnotu parametru batch. Cílem těchto bylo zjistit, která z použitých metod dosáhne nejlepších výsledků detekce vozidel.

Z průběhů grafů 4.11, 4.12 a 4.13 je možné vyčíst, že lepších výsledků trénování modelu YOLOv5s bylo dosaženo za použití datové sady s originálním rozlišením a nižší hodnotou batch. Souhlasný trend průběhu trénování je možné pozorovat i u většího modelu YOLOv5m viz 4.14, 4.15 a 4.16.

Výsledky detekce na testovací množině za použití finální datové sady zaznamenaly doposud nejvyšší hodnoty všech sledovaných metrik. Všechny experimenty dosáhly mAP alespoň 0,99.

Z dvojce trénovaných modelů YOLOv5s dopadl po testování lépe model trénovaný na datové sadě s redukovaným rozlišením 864 x 476 pixelů. Tento model docílil mAP 0,993, preciznosti 0,958 a úplnosti odhadu detekce 0,974.



Obr. 4.19: Matice záměn nejúspěšnějšího natrénovaného modelu YOLOv5m, za použití snímků o rozlišení 1280 x 720 pixelů a finální datové sady.

Nejúspěšnějším natrénovaným detektorem vozidel v obraze se stal model YOLOv5m za použití snímků o rozlišení 1280 x 720 pixelů a finální datové sady. Hod-

noty všech sledovaných metrik u žádné z objektových tříd neklesly pod hodnotu 0,93. Celkově model dosáhl mAP 0,995, preciznosti 0,995 a úplnosti odhadu detekce 0,986. Z matice záměn na obrázku 4.19 je zřejmé, že dochází k velmi nízkému výskytu mylné detekce, nejčastěji došlo k záměně kamionu za pozadí.

Závěr

Cílem této diplomové práce bylo seznámit se s technikami konvolučních neuronových sítí a problematikou detekce objektů v obraze, dalším cílem práce bylo zpracování aktuálního stavu vědy a techniky v této oblasti. Z oblasti praktické části bylo kladeno za cíl navržení neuronové sítě nebo využití již existujících předtrénovaných modelů, které budou zpracovávat data z video záznamů. Případné využité existující modely je nutné dotrénovat na sesbírané datové množině.

První kapitola obsahuje popis umělých neuronových sítí, pozornost je věnována jejich základním stavebním prvkům, následně jsou v kapitole popsány konvoluční neuronové sítě. Popis konvolučních neuronových sítí je zaměřen především na jejich architekturu a typy vrstev.

V druhé kapitole bylo provedeno zmapování aktuálního stavu vědy a techniky v oblasti detekce objektů ve videu. V kapitole byly popsány nejpopulárnější detekční systémy a architektury. Zvláštní pozornost byla věnována algoritmům YOLO, které byly použity pro vypracování praktické části práce.

V praktické části práce byly pro detekci objektů ve videu využity dva nejmenší modely YOLOv5s a YOLOv5m. Pro prvotní experimenty trénování byla použita veřejně přístupná datová sadu Roboflow Vehicles-OpenImages, která obsahuje pouze 627 obrázků, anotovaných do 5 kategorií (osobní automobil, autobus, motocykl, nákladní automobil a sanitní vůz.). Jednotlivé třídy nejsou v datové množině zastoupeny rovnoměrně, nadměrně zastoupena je třída osobní automobil. Dále v této části byla vytvořena vlastní datová sada čítající 1650 snímků rozdělených do tříd: osobní automobil, autobus, motocykl, nákladní automobil a dodávkový automobil. Pro sběr dat byl použit digitální fotoaparát Nikon D3100 připevněný na stativ. Vlastní datová sada podobně jako veřejně přístupná datová sadu Roboflow Vehicles-OpenImages trpí nevyvážeností objektových tříd, kdy je nadměrně zastoupena třída osobní automobil. Snahou zjistit, zda-li větší vyváženost objektových tříd spolu s použitými snímky vozidel z internetu dokáží pozitivně ovlivnit schopnost detekce vozidel v obraze vznikla rozšířená datová sada. Rozšířená datová sada obsahuje oproti původní vlastní datové množině snímky nedostatečně zastoupených objektových tříd stažených z internetu. Posledním použitým datasetem je finální datová množina, který eliminovala problém s nevyváženými objektovými třídami vhodným rozšířením datové sady. Veškeré obrázky z internetu byly nahrazeny vlastními sesbíranými daty. Celkově finální datová sada čítá 5404 snímků a 6467 anotovaných objektů.

V poslední kapitole byly diskutovány průběhy trénování detektoru YOLOv5 a dosažené výsledky při použití veřejně dostupné datové sady Roboflow Vehicles a vytvořených vlastních datových množin.

Natrénovaný model YOLOv5s za použití datové množiny Roboflow Vehicles do-

sáhl na testovací datové sadě nízké hodnoty mAP 0,740, preciznosti 0,722 a úplnosti predikce 0,688. Z matice záměn je patrné, že k nejvíce chybám dochází při záměně pozadí za osobní automobil a při nedetekování osobního automobilu na testovaných snímcích. Jednou z příčin problému nízké úspěšnosti detekce vozidel se jeví špatná anotace obrázků, datová sada obsahuje nejen fotky skutečných dopravních prostředků, ale také hraček. U části anotovaných snímků nejsou vytvořeny labely pro objekty v pozadí, když tyto neanotované objekty model správně detekuje je detekce vyhodnocena jako mylná. Další příčinou neuspokojivých dosažených výsledku je použití velmi malé datové sady.

První verze vlastní vytvořené datové sady trpěla podobně jako datová množina Roboflow Vehicles nevyvážeností objektových tříd. Dalším faktorem, který negativně ovlivnil detekci na testovací množině byl nesprávný způsob, kterým webová aplikace Roboflow rozdělila anotované snímky mezi trénovací a validační sadu dat. Nejlepšího výsledku při užití prvotní verze vlastního datové sady dosáhl natrénovaný model YOLOv5, obdobně jako u datasetu Roboflow Vehicles se jedná o nízké hodnoty, konkrétně mAP 0,790, preciznost 0,628 a úplnost predikce 0,825.

Za účelem zjištění, zda-li větší vyváženost objektových tříd dokáže pozitivně ovlivnit schopnost detekce vozidel v obraze byla prvotní verze datové množiny rozšířena o snímky z internetu. Tento předpoklad se potvrdil. Nejlépe natrénovaný model YOLOv5s dosáhl mAP 0,836, preciznosti 0,769 a úplnosti predikce 0,872.

Na základě předchozích zjištění byla vytvořena finální verze datové množiny. Vhodným rozšířením byl eliminován problém nevyvážených objektových tříd. Veškeré obrázky z internetu byly nahrazeny vlastními sesbíranými daty. Byly prováděny experimenty trénování YOLOv5s a YOLOv5m, dalším zkoumaným aspektem bylo použití snímků s redukovaným rozlišením. Nejlepšího výsledku docílil natrénovaný model YOLOv5m za použití snímků o rozlišení 1280 x 720 pixelů. Natrénovaný model dosáhl mAP 0,995, preciznosti 0,995 a úplnosti odhadu detekce 0,986. Z matice záměn je zřejmé, že dochází k minimálnímu výskytu mylné detekce.

Detekce objektů byla vyzkoušena i na videu z dopravní kamery snímající rušnou světelnou křižovatku. Video o rozlišení 1280 x 720 pixelů bylo na GPU Nvidia RTX 2070 Super zpracováno rychlostí 44,87 FPS, tím bylo dokázáno, že na použitém hardwaru je model možné provozovat v reálném čase pro videa o rozlišení 1280 x 720 pixelů.

Literatura

[1] VONDRÁK, Ivo. *Umělá inteligence a neuronové sítě*. 3. vyd. Ostrava: VŠB - Technická univerzita Ostrava, 2009. ISBN 978-80-248-1981-5.

[2] TUČKOVÁ, Jana. *Vybrané aplikace umělých neuronových sítí při zpracování signálů*. Praha: České vysoké učení technické v Praze, 2009. ISBN 978-80-01-04229-8.

[3] NOVÁK, Mirko. *Umělé neuronové sítě:* teorie a aplikace. Praha: C.H. Beck, 1998. Učebnice informatiky. ISBN 80-717-9132-6

[4] ŠÍMA, Jiří a Roman NERUDA. *Teoretické otázky neuronových sítí*. Praha: MATFYZPRESS, 1996. ISBN 80-858-6318-9.

[5] SHARMA, Sagar. Activation functions in neural networks. *Towards Data Science*, 2017, 6.

[6] KHAN, Salman, et al. A guide to convolutional neural networks for computer vision. *Synthesis Lectures on Computer Vision*, 2018, 8.1: 1-207.

[7] NGIAM, Jiquan, et al. Tiled convolutional neural networks. In: *Advances in neural information processing systems*. 2010. p. 1279-1287.

[8] GOODFELLOW, Ian, et al. *Deep learning*. Cambridge: MIT press, 2016.

[9] CIABURRO, Giuseppe; AYYADEVARA, V. Kishore; PERRIER, Alexis. *Hands-on Machine Learning on Google Cloud Platform: Implementing Smart and Efficient Analytics Using Cloud ML Engine*. Packt Publishing Ltd, 2018.

[10] MARTIN, Scott. What's the Difference Between a CNN and an RNN? In: *NVIDIA Blogs* [online]. 2018 [cit. 2020-10-30]. Dostupné z: https://blogs.nvidia.com/blog/2018/09/05/whats-the-difference-between-a-cnn-and-an-rnn/

[11] LI, Fei-Fei, Andrej KARPATHY a Justin JOHNSON. CS231n: *Convolutional Neural Networks for Visual Recognition* [online]. [cit. 2020-11-01]. Dostupné z: `https://cs231n.github.io/convolutional-networks/`

[12] Mathworks documentation R2020b, *Specify Layers of Convolutional Neural Network* [online]. [cit. 2020-11-5]. Dostupné z: `https://www.mathworks.com/help/deeplearning/ug/layers-of-a-convolutional-neural-network.html#bvobklb-8`

[13] REDMON, Joseph, et al. You only look once: Unified, real-time object detection. In: *Proceedings of the IEEE conference on computer vision and pattern recognition.* 2016. p. 779-788.

[14] UIJLINGS, Jasper RR, et al. Selective search for object recognition. *International journal of computer vision*, 2013, 104.2: 154-171.

[15] GIRSHICK, Ross, et al. Rich feature hierarchies for accurate object detection and semantic segmentation. In: *Proceedings of the IEEE conference on computer vision and pattern recognition.* 2014. p. 580-587.

[16] GIRSHICK, Ross. Fast R-CNN. In: *Proceedings of the IEEE international conference on computer vision.* 2015. p. 1440-1448.

[17] REN, Shaoqing, et al. Faster R-CNN: Towards real-time object detection with region proposal networks. In: *Advances in neural information processing systems.* 2015. p. 91-99.

[18] HE, Kaiming, et al. Mask R-CNN. In: *Proceedings of the IEEE international conference on computer vision.* 2017. p. 2961-2969.

[19] SIMONYAN, Karen; ZISSERMAN, Andrew. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv*:1409.1556, 2014.

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

[21] SZEGEDY, Christian, et al. Going deeper with convolutions. In: *Proceedings of the IEEE conference on computer vision and pattern recognition.* 2015. p. 1-9.

[22] REDMON, Joseph; FARHADI, Ali. YOLO9000: better, faster, stronger. In: *Proceedings of the IEEE conference on computer vision and pattern recognition.* 2017. p. 7263-7271.

[23] REDMON, Joseph; FARHADI, Ali. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[24] SAGAR, Ram. YOLO Creator Quits AI Research Citing Ethical Concerns. *Analytics India Magazine* [online]. 25.2.2020 [cit. 2020-11-30]. Dostupné z: `https://analyticsindiamag.com/yolo-creator-joe-redmon-computer-vision-research-ethical-concern/`

[25] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 37(9):1904–1916, 2015. 2, 4, 7

[26] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI),*40(4):834–848, 2017. 2, 4

[27] Songtao Liu, Di Huang, et al. Receptive field block net foraccurate and fast object detection. In: *Proceedings of the European Conference on Computer Vision (ECCV)*, 385–400, 2018. 2, 4, 11

[28] BOCHKOVSKIY, Alexey; WANG, Chien-Yao; LIAO, Hong-Yuan Mark. YOLOv4: Optimal Speed and Accuracy of Object Detection. *arXiv preprint arXiv: 2004.10934*, 2020.

[29] JOCHER, Glenn, ULTRALYTICS LLC. YOLOv5 GitHub repository [online] , [cit. 2020-11-30]. Dostupné z: `https://github.com/ultralytics/yolov5`

[30] NELSON, Joseph a Jacob SOLAWETZ. Responding to the Controversy about YOLOv5. In: *Roboflow Blog* [online]. 12.6.2020 [cit. 2020-12-02]. Dostupné z: `https://blog.roboflow.com/yolov4-versus-yolov5/`

[31] Jacob SOLAWETZ. YOLOv5 New Version - Improvements And Evaluation. In: *Roboflow Blog* [online]. 29.6.2020 [cit. 2020-12-02]. Dostupné z: `https://blog.roboflow.com/yolov5-improvements-and-evaluation/`

[32] Anaconda Commercial Edition. *Anaconda* [online]. [cit. 2020-12-03]. Dostupné z: `https://www.anaconda.com/blog/anaconda-commercial-edition-faq`

[33] About Us. *Anaconda* [online]. [cit. 2020-12-03]. Dostupné z: `https://www.anaconda.com/about-us`

[34] Conda Documentation. *Conda* [online]. [cit. 2020-12-03]. Dostupné z: `https://conda.io/en/latest/`

[35] FAQs: What's the difference between Anaconda, conda, and Miniconda? *Bio-Conda GitHub* [online]. [cit. 2020-12-03]. Dostupné z: `https://bioconda.github.io/contributor/faqs.html#conda-anaconda-minconda`

[36] Anaconda Navigator. *Anaconda Documentation* [online]. [cit. 2020-12-03]. Dostupné z: `https://docs.anaconda.com/anaconda/navigator/`

[37] *PyTorch* [online].[cit. 2020-11-10]. Dostupné z: https://pytorch.org/

[38] STEVENS, Eli, Luca ANTIGA a Thomas VIEHMANN. *Deep Learning with PyTorch.* Manning Publications Co., 2020. ISBN 9781617295263.

[39] Deep Learning Frameworks Documentation. *NVIDIA* [online]. [cit. 2020-12-03]. Dostupné z: https://docs.nvidia.com/deeplearning/frameworks/index.html

[40] GEFORCE RTX 2070 SUPER. *NVIDIA* [online]. [cit. 2020-12-03]. Dostupné z: https://www.nvidia.com/en-us/data-center/tensor-cores/

[41] NVIDIA Tensor Cores. *NVIDIA cloud & data center* [online]. [cit. 2020-12-03]. Dostupné z: https://www.nvidia.com/cs-cz/geforce/graphics-cards/rtx-2070-super/

[42] SEKACHEV, Boris, Nikita MANOVICH a Andrey ZHAVORONKOV. New Computer Vision Tool Accelerates Annotation of Digital Images and Video *Intel* [online]. [cit. 2021-04-20]. Dostupné z: https://www.intel.com/content/www/us/en/artificial-intelligence/posts/introducing-cvat.html

[43] SEKACHEV, Boris, Nikita MANOVICH a Andrey ZHAVORONKOV. Computer Vision Annotation Tool: A Universal Approach to Data Annotation *Intel* [online]. [cit. 2021-04-20]. Dostupné z: https://software.intel.com/content/www/us/en/develop/articles/computer-vision-annotation-tool-a-universal-approach-to-data-annotation.html

[44] Datumaro GitHub repository. *GitHub* [online] , [cit. 2021-04-20]. Dostupné z: https://github.com/openvinotoolkit/datumaro

[45] *Weights & Biases* [online] [cit. 2021-04-21]. Dostupné z: https://wandb.ai/site

[46] DaVinci Resolve 17. *Black Magic Design* [online], [cit. 2021-04-20]. Dostupné z: https://www.blackmagicdesign.com/products/davinciresolve/

[47] Roboflow Documentation. *Roboflow* [online] [cit. 2021-04-21]. Dostupné z: https://docs.roboflow.com/

[48] Public Datasets: Vehicles-OpenImages Dataset. *Roboflow* [online]. [cit. 2021-04-20]. Dostupné z: https://public.roboflow.com/object-detection/vehicles-openimages/1

[49] 20 Anni di Mercedes-Benz Sprinter. *Newsauto* [online]. [cit. 2021-04-20]. Dostupné z: https://www.newsauto.it/mercedes-benz/20-anni-di-mercedes-benz-sprinter-2016-54164

Seznam symbolů, veličin a zkratek

AF	Aktivační funkce
AI	Artificial Intelligence
AP	Average Precision
ASPP	Atrous Spatial Pyramid Pooling
CLI	Command Line Interface
CNN	Convolution Neural Network
CVAT	Computer Vision Annotation Tool
FAIR	Facebook AI Research
FPS	Frames Per Second
GPU	Graphics processing unit
HD	High Definition
IoU	Intersection over Union
mAP	Mean Average Precision
R-CNN	Region Based Convolutional Neural Networks
ReLU	Rectified Linear Unit
RFB	Receptive Field Block
RoI	Region of Interest
RPN	Region Proposal Network
Seq-NMS	Spatial Pyramid Pooling
SPP	Spatial Pyramid Pooling
SSD	Single Shot Multibox Detector)
YOLO	You Only Look Once

5 Obsah přiloženého DVD

Všechny datové množiny použité v rámci diplomové práce se nachází v adresáři Datasets. Naklonovaný GitHub repozitář YOLOv5 je umístěn ve složce Yolov5. Repozitář obsahuje všechny pythonovské skripty, konfigurační soubory, výsledky trénování a testování. Veškeré výsledné váhy trénování YOLO pro všechny experimenty jsou k nalezení v adresáři Weights. Základní struktura souborů je zapsána níže.

```
/ ..... kořenový adresář přiloženého DVD
├── Datasets ..... adresář s použitými datasety
│   ├── Roboflow
│   ├── Vehicles_final
│   ├── Vehicles_v1
│   └── Vehicles_v2
├── Yolov5 ..... stažený GitHub repozitář YOLOv5
└── Weights ..... výsledné váhy trénování YOLO
    ├── Roboflow_YOLOv5s.pt
    ├── Vehicles_final_864x486_YOLOv5m.pt
    ├── Vehicles_final_864x486_YOLOv5s.pt
    ├── Vehicles_final_1280x720_YOLOv5m.pt
    ├── Vehicles_final_1280x720_YOLOv5s.pt
    ├── Vehicles_v1_YOLOv5m.pt
    ├── Vehicles_v1_YOLOv5s.pt
    ├── Vehicles_v2_YOLOv5m.pt
    └── Vehicles_v2_YOLOv5s.pt
```