FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Detekce zaparkovaných aut z pohybujícího se dronu pro analytické účely
Student:	Bc. Peter Kanoš
Vedoucí:	Ing. Lukáš Brchl
Studijní program:	Informatika
Studijní obor:	Znalostní inženýrství
Katedra:	Katedra aplikované matematiky
Platnost zadání:	Do konce zimního semestru 2021/22

Pokyny pro vypracování

Cílem práce je návrh algoritmů a implementace aplikace, která bude detekovat zaparkovaná auta z obrazového záznamu pořízeného dronem. Implementaci proveďte v jazyce Python.

- Proveďte rešerši existujících řešení zabývající se detekcí aut z dronu.
- Proveďte návrh a implementaci algoritmů detekce s využitím algoritmů počítačového vidění.
- Prozkoumejte a implementujte vhodné výstupní statistiky.
- Zvažte možnost vytvoření vlastního datasetu pro vyhodnocení algoritmů nebo vyberte z existujících.
- Proveďte zhodnocení dosažených výsledků a navrhněte budoucí rozšíření.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Karel Klouda, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 17. února 2020

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Diplomová práca

Detekce zaparkovaných aut z pohybujícího se dronu pro analytické účely

Bc. Peter Kanoš

Katedra aplikované matematiky
Vedúci práce: Ing. Lukáš Brchl

3. augusta 2020

Poďakovanie

V prvom rade by som chcel poďakovať Bohu za všetko čo mám. Ďalej by som sa rád poďakoval mojim rodičom, ktorí pri mne celý čas stáli a verili mi. Takisto aj mojím súrodencom Adamkovi, Terezke a Janke, bez ktorých podpory by to bolo omnoho ťažšie. Moja vďaka patrí tiež starým rodičom a tak isto aj tete Jane za prebdené noci. Ďalej by som chcel poďakovať svojim spolužiakom, ktorí mi celý čas pomáhali a podporovali ma, hlavne Eliške, Dominkovi a Tánke. Mojim učiteľom a cvičiacim, že to so mnou tie roky vydržali. Najmä vedúcemu mojej diplomovej práce Ing. Lukášovi Brchlovi a vedúcemu mojej bakalárskej práce Ing. Danielovi Vašatovi Ph.D.. A nakoniec, Mati a Adi za ich ochotu a pomoc pri gramatických opravách tejto práce.

Prehlásenie

Prehlasujem, že som predloženú prácu vypracoval(a) samostatne a že som uviedol(uviedla) všetky informačné zdroje v súlade s Metodickým pokynom o etickej príprave vysokoškolských záverečných prác.

Beriem na vedomie, že sa na moju prácu vzťahujú práva a povinnosti vyplývajúce zo zákona č. 121/2000 Sb., autorského zákona, v znení neskorších predpisov, a skutočnosť, že České vysoké učení technické v Praze má právo na uzavrenie licenčnej zmluvy o použití tejto práce ako školského diela podľa § 60 odst. 1 autorského zákona.

V Prahe 3. augusta 2020 . . . . . .

Odkaz na túto prácu

Kanoš, Peter. *Detekce zaparkovaných aut z pohybujícího se dronu pro analytické účely.* Diplomová práca. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2020.

Abstrakt

Práca sa zaoberá návrhom a implementáciou aplikácie na rozpoznávanie áut z obrazového záznamu urobeného letiacim dronom.

V prvej teoretickej časti práca obsahuje úvod do oblasti detekovania objektov, ako aj prehľad state-of-art algoritmov. Táto časť je zameraná najmä na prístupy použité v tejto práci. Ďalšou časťou práce je rešeršná časť, kde je urobený prehľad doteraz implementovaných riešení použitých v rámci tejto problematiky.

Výsledkom práce sú štyri modely, pri ktorých trénovaní bol použitý dataset CARPK [1]. Tento dataset obsahuje fotky, ako aj videozáznamy urobené letiacim dronom. Celkom sa v datasete nachádza 90 000 zachytených áut na viac ako 1500 fotkách. Ďalej je v práci navrhnutý prístup detekcie objektov vo videu pomocou vytvorenia ortofotomapy z daného videa, na ktorej je následne prevedená detekcia. Celá práca je implementovaná v jazyku Python.

V záverečnej časti práce sú predstavené experimenty, ktoré boli vykonané na dátach mimo dataset.

Klíčová slova detekcia áut, rozpoznávanie, drony, rátanie objektov, počítačové videnie

Abstract

This thesis deals with the implementation of an application for vehicle recognition from images captured with flying unmanned aerial vehicle (UAV). In the first section of the thesis is an overview of already existing solutions from this topic. This section is focused mainly on algorithms used in this thesis. The next section is an overview of already implemented solutions for automatic car detection on parking lots. Besides, the thesis proposes and implements an algorithm for the automatic detection of parked vehicles. The result of the thesis is four models and approaches implemented for automatic car detection from UAV on parking lots. For training was used CARPK dataset which contains more than 90 000 captured cars on more than 1500 photos. Furthermore, the work proposes an approach to detecting objects in a video, by creating an orthophoto map, in which is subsequently provided detection. The whole thesis is implemented in the Python language.

Keywords cars detection, recognition, unnamed aerials vehicles, objects counting, computer vision

Obsah

Zoznam obrázkov

Zoznam tabuliek

Úvod

V oblasti strojového (počítačového) videnia a rozpoznávania obrazu sa urobil v posledných rokoch veľký skok vpred, čo má za následok jeho využitie v každej sfére nášho života. Vo využití počítačového videnia (computer vision) sú dva základné problémy: detekcia objektov (object detection), sledovanie objektov (object tracking). Hlavnou úlohou v oblasti detekcii objektov je na obrázku lokalizovať inštancie objektov z veľkého množstva preddefinovaných kategórií. Detekcia objektov sa ďalej delí vzhľadom na to z akého obrazového média sa objekty detekujú, a teda na detekciu objektov vo videách a v obrázkoch.

V posledných rokoch bol tiež viditeľný rozmach bezpilotných lietadiel UAV, ktorých využitie sa našlo v mnohých oblastiach nášho života, či už sa to týka výskumu, armádnych operácií, filmovania, stráženia objektov a podobne. Vzhľadom na zložitosť úloh, niektoré stále potrebujú k svojmu vykonaniu ľudskú asistenciu. Jednou z týchto oblastí je aj autonómne trackovanie a detekovanie dopravných prostriedkov (áut) či už na parkoviskách alebo priamo na cestách. Pri riešení tejto úlohy sa narazilo za posledné roky na mnoho problémov s nimi spojenými, a taktiež sa implementovalo mnoho zaujímavých riešení. [17]

Cieľom tejto práce bolo navrhnúť a implementovať aplikáciu, ktorá bude detekovať zaparkované autá z obrazového záznamu vyhotoveného pomocou dronu.

V prvej časti práce je poskytnutý teoretický základ metód používaných v oblasti detekcie objektov. Následne sú predstavené hlavné (state-of-the-art) algoritmy, ktoré sa na riešenie tejto problematiky používajú a ich krátke porovnanie (s odkazmi na príslušnú odbornú literatúru). V ďalšej časti práce je urobený prehľad už existujúcich riešení tejto problematiky ako aj datasetov, ktoré sú pri ich riešení používané. V nasledujúcej časti je predstavený vlastný výber implementovaného algoritmu ako aj diskusia nad jeho výberom. Tak isto je ďalej predstavený aj výber trénovacích dát so zreteľom na objektívne zhodnotenie ich vhodnosti na riešenie daného problému. Tretia časť je zame-

raná na samotný návrh riešenia. Na záver je vypracovaná analýza výsledného riešenia, experimenty a zhodnotenie dosiahnutých výsledkov.

Kapitola 1

Úvod do teórie detekcie objektov

Detekcia objektov je jednou z najrozsiahlejšie sa rozširovaných oblastí strojového videnia. Záujem o túto oblasť je spôsobený najmä jej širokou škálou využitia. Či už sa jedná o autopilotné autá, sledovanie áut, osôb na videu a pod. Detekcia objektov patrí medzi jednu z hlavných kategórií, ktorými sa strojové videnie zaoberá, ďalej medzi tieto kategórie patrí napríklad klasifikácia objektov (object classification), sémantická segmentácia (semantic segmentation) a segmentácia inštancií objektov (instance segmentation).

Detekcia objektov sa podľa [17] dá rozdeliť na dve základné skupiny: obecné detekovanie objektov (generic object detection) a špecifické detekovanie objektov. Prvé sa snaží o detekovanie inštancií špecifického objektu ako napríklad Pražský hrad, kdežto detekcia generických objektov sa snaží určovať obecnú skupinu, do ktorej objekt na obrázku spadá ako napríklad pes alebo kôň. Medzi hlavné dôvody napredovania tejto oblasti patrí najmä záujem verejnosti a široké možnosti jeho praktického využitia. Rozmach je podporený najmä rozširujúcim sa množstvom trénovacích datasetov, rôznych súťaží a výziev najmä čo sa týka klasifikácie objektov, ktoré sú s detekciou objektov úzko späté. Tieto majú ďalej za následok stále sa rozširujúcu rodinu state-of-the-art algoritmov na detekovanie objektov.

Táto kapitola je zameraná na základné architektúry používané v súčasnosti na detekciu objektov a zamerať sa hlavne na architektúry sietí použitých v práci.

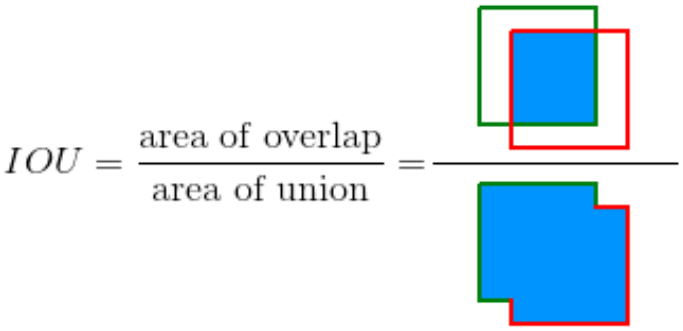
1.1 Používané metriky

V tejto sekcii predstavím základné metriky používané pre porovnávanie modelov na detekciu objektov.

1.1.1 Intersection over Union

Intersection over Union (IoU) je metrika na porovnávanie podobnosti dvoch priestorových objektov z hľadiska ich obsahu. Nech $A, B \subseteq S \in R^n$ potom IoU je definovaná ako:

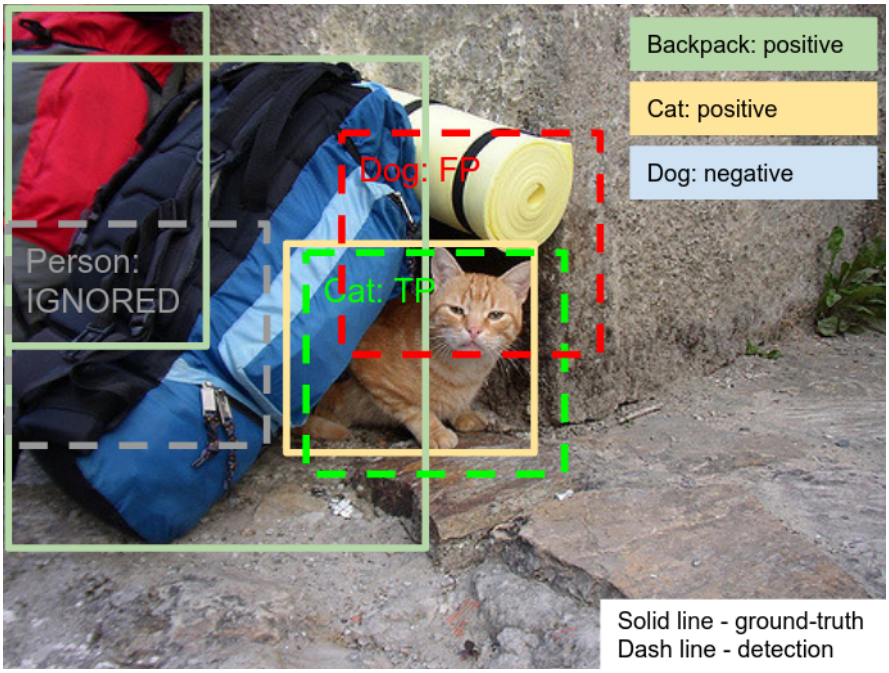
$$IoU = \frac{|A \cap B|}{|A \cup B|}.$$



Obr. 1.1: Vizuálizácia intersection over union [3]

Je to teda pomer medzi prienikom dvoch množín a ich zjednotením (viď obrázok 1.1). IoU je invariantná voči škále problému. Využíva sa u detekcie objektov na meranie presnosti danej detekcie voči pravdivému olabelovaniu daného objektu. Invariantnosť voči škálovaniu je vhodná pri detekcii objektov, kde sa využíva inferencia na rôznych škálach toho istého obrázku (multiscale object detection).

Pri detekcii objektov dochádza k procesu kedy sa prechádza medzi všetkými detekciami daného modelu a vyberajú sa tie, ktoré majú vyššie skóre istoty (confidence score) než zvolený prah. Následne sa spomedzi daných detekcií vyberie tá, ktorá má správnu triedu a najvyššiu hodnotu IoU s pravdivým olabelovaním daného objektu. Ak taká existuje, detekcia je braná ako pravdivo pozitívna (true positive) v opačnom prípade je falošne pozitívna (false positive). Ak je skóre istoty (confidence score) detekcie, ktorá predikuje správnu triedu nižšie než prah, je daná detekcia braná ako falošne negatívna (false negative). Ak je skóre istoty (confidence score) nižšie než daný prah a zároveň nedetekuje danú triedu je braná detekcia ako pravdivo negatívna (true negative) [3].



Obr. 1.2: Vizualizácia true a false positive detekcie [4]

1.1.2 Mean Average Precision

Na porovnanie detekcie sa bežne u modelov strojového učenia používa Precision a Recall. Kde precision je definovaná ako:

$$precision = \frac{TP}{TP + FP}.$$

Recall je definovaný ako pomer:

$$recall = \frac{TP}{TP + FN},$$

kde TP je počet pravdivo pozitívnych (true positive) detekcií, FP je počet falošne pozitívnych (false positive) detekcií, FN počet falošne negatívnych (false negative) detekcií. Zmenou prahu (thresholdu) skóre istoty (confidence score) modelu sa nám však mení aj precision a recall daného modelu. Túto závislosť je možné vykresliť pomocou precision-recall krivky. Na jednotné porovnanie presnosti modelov sa však pre jednoduchosť neporovnávajú celé krivky, ale používa sa ako bežná metrika jedno číslo, a tým je average precision. Average precision (AP) je precision priemerovaná cez všetky recall levely. Na zrátanie average precision sa najprv interpoluje precision pre daný recall. Tento precision je definovaný ako najvyšší precision pre daný recall level

$$r' \geq r :$$

$$p_{interp}(r) = maxp(r') :$$

kde $r' \geq r$. Jednou možnosťou voľby recall levelov je zvolenie 11 rovnomerne rozložených levelov, (0.0, 0.1...1.0) Average precision môže byť definovaná ako plocha pod interpolovanou precision-recall krivkou.

$$AP = \sum_{i=1}^{n-1} (r_{i+1} - r_i) p_{interp}(r_{i+1})$$

kde $r_1, r_2 ... r_n$ sú stupne recallu. MAP je rozšírením Average Precision pre K tried

$$mAP = \frac{\sum_{i=1}^{K} AP_i}{K}$$

.

1.1.3 Average Recall

Tak ako average precision, average recall (AR) je numerickou metrikou, ktorá je používaná na porovnanie detekčných modelov. Average recall je priemer cez IoU $\in$ [0.5, 1.0] a môže byť zrátaný ako dvakrát obsah pod recall-IoU krivkou teda:

$$AR = 2 \int_{0.5}^{1} recall(o) do$$

kde o je IoU a $recall(o)$ je recall k danému IoU. Mean average recall je následne definovaný znova ako priemer cez všetky triedy detekcie:

$$mAR = \frac{\sum_{i=1}^{K} AR_i}{K}$$

1.1.4 COCO challenge metriky

COCO challenge definuje vlastné prahy vyššie definovaných metrík pre porovnávanie výsledkov modelov v ich súťažiach. Jednou skupinou sú metriky pre rôzne veľkosti IoU:

- $mAP^{IoU=.50:.05:.95}$ - je mAP priemerované cez 10 IoU prahov (thresholds) (0.50, 0.55, ...,0.95),
- $mAP^{IoU=.50}$ - mAP pre 0.50 IoU,
- $mAP^{IoU=.75}$ - mAP pre 0.75 IoU.

Okrem IoU prahov (thresholds) je aj mAP rátaný cez rôzne veľkosti objektov, ktoré sú priemerované cez všetkých 10 IoU prahov (thresholds):

- mAP^{small} - mAP pre malé objekty, ktoré pokrývajú plochu menšiu než 32^2,

- mAP^{medium}- mAP pre objekty, ktoré pokrývajú plochu väčšiu než 32^2 ale menšiu než 96^2,
- mAP^{large} - mAP pre objekty, ktoré pokrývajú plochu väčšiu než 96^2.

Tak ako mAP aj mean average recall (mAR) metrika má tiež viacero variant, prvou je počet detekcií na snímok:

- $mAR^{max=1}$ - mAR pre jednu detekciu na obrázok,
- $mAR^{max=10}$ - mAR pre desať detekcií na obrázok,
- $mAR^{max=100}$ - mAR pre sto detekcií na obrázok.

Ďalšia sada mean Average Recallu je definovaná pre rôzne veľkosti objektov:

- mAR^{small} - mAR pre malé objekty, ktoré pokrývajú plochu menšiu než 32^2,
- mAR^{medium} - mAR pre objekty, ktoré pokrývajú plochu väčšiu než 32^2 ale menšiu než 96^2,
- mAR^{large} - mAR pre objekty, ktoré pokrývajú plochu väčšiu než 96^2.

[4]

1.2 Konvolučné neurónove siete

V tejto časti predstavím základnú architektúru konvolučných neuronových sietí, vysvetlím na príklade prečo sa v praxi pouívajú, a následne predstavím architektúry používané v tejto práci. Teoretická časť týkajúca sa základnej stavby konvolučných neurónových sietí je prevzatá z [5].

Predstavme si jednoduchý problém, chceme klasifikovať triedy objektov na obrázkoch tzn. určiť čo za objekt sa na obrázku nachádza. Uvažujeme pritom, že väčšina plochy vstupného obrázku zaberá práve objekt, ktorý chceme klasifikovať. Pre jednoduchosť vezmime obrázok rozmeru 32x32 pixelov. Ak uvažujeme, že obrázok je farebný ešte musíme rátať s jednou dimenziou pre každý kanál. Ak by sme chceli túto úlohu riešiť za pomoci FC neurónovej siete, jeden neurón na prvej skrytej vrstve by mal $32*32*3 = 3072$ váh. Pre obrázky väčších rozmerov sa počet parametrov na jeden neurón zväčšuje geometrickým radom, a tento počet parametrov by viedol k preučeniu (overffiting) siete.

Pre takýto typ úloh bol navrhnutý nový typ neurónových sietí - konvolučné neuronové siete convolutional neural networks (CNN), ktoré majú neuróny zostavené v troch dimenziách: šírka, výška a hĺbka. Hĺbka v tomto prípade značí počet dimenzií - na vstupnej vrstve pre farebné obrázky by to bolo 3 pre 3-RGB kanály. Obecne sa pre stavbu konvolučných sietí používajú štyri druhy vrstiev:

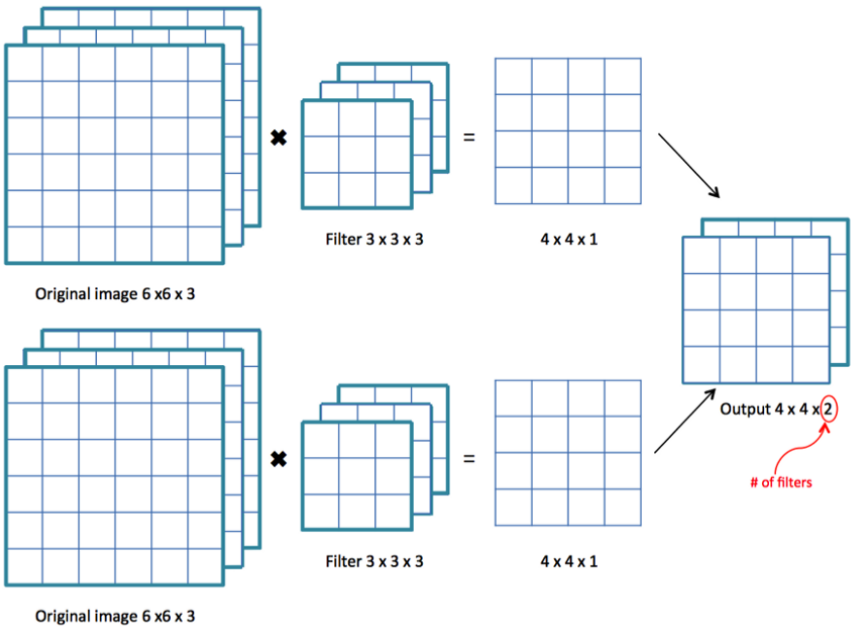
- konvolučná vrstva,
- pooling vrstva,
- FC vrstva,
- RELU vrstva.

1.2.1 Konvolučná vrstva

Parametre konvolučných vrstiev pozostávajú zo sady filtrov, ktoré sa tak ako váhy pri FC sieťach učia (obvykle pomocou backpropagation algoritmu [18]). Počas dopredného prechodu posúvame filter po vstupných dátach - a rátame skalárny súčin medzi hodnotami filtru a hodnotami vstupu na každej pozícii. Obvykle majú konvolučné vrstvy filtre nepárnej veľkosti napríklad $3x3xk, 5x5xk, 7x7xk$ to najmä z dôvodu výslednej veľkosti feature mapy po aplikovaní konvolúcie.

Týchto filtrov má každá konvolučná vrstva väčšie množstvo a môžeme si predstaviť, že každý z nich zachytáva určitú vlastnosť (feature) vstupu. Napríklad horizontálnu/vertikálnu hranu na obrázku a pod. Počet konvolučných filtrov danej vrstvy nám určuje počet dimenzii na výstupe pretože, výstupy jednotlivých konvolučných filtrov danej vrstve sa na výstupe skladajú na seba viď 1.3.

Veľkosť výstupu konvolučnej vrstvy sa riadi troma parametrami:



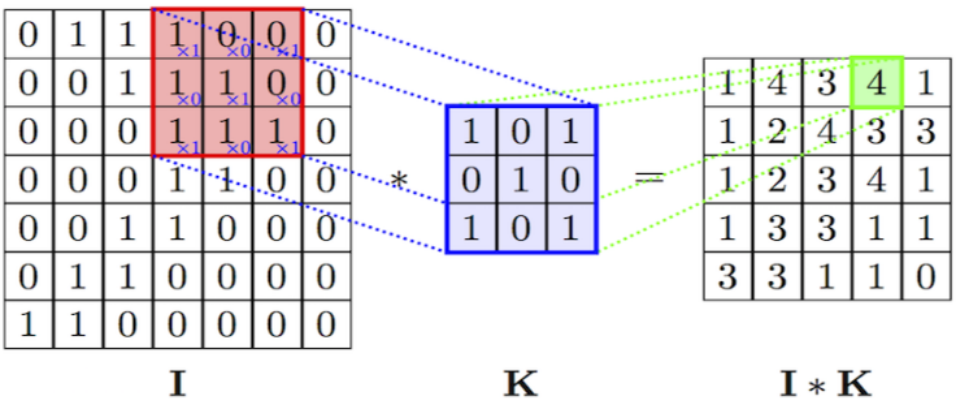
Obr. 1.3: Vizuálizácia konvolučných filtrov prevzaté z: [5]

- hĺbka (depth) počet filtrov v danej vrstve,
- krok (stride) veľkosti kokov posunu filtra po vstupnách dátach,
- nulová výplň (zero padding) použitá na vstupné dáta.

Vo vstupných dátach hĺbka predstavuje počet kanálov, pre farebný obrázok teda 3 RGB. Krok (stride) je udávaný v pixeloch, teda hodnota 2 určuje skoky filtra o hodnotu 2px. Nulová výplň (zero padding) sa aplikuje pre rovnomerné použitie vstupných pixelov a korekcii rozmerov výstupu. Vstupné dáta sú kvôli spôsobu ako konvolúcia prebieha nerovnomerne použité pre výpočet daného výstupu konvolučnej vrstvy. Pixely v centrálnych častiach obrázka sa ako vstup do výpočtu používajú častejšie ako tie na okrajoch. Pohyb filtra po vstupných dátach prebieha nasledovne: doplníme vstupné dáta o definovanú nulovú výplň (zero padding), následne aplikujeme konvolučný filter na štartovaciu pozíciu - ľavý horný roh vstupných dát a postupujeme z ľava do prava (po osi x) po krokoch veľkosti definovanou veľkosti kroku (stride). Pri každom kroku zrátame výsledok aplikovanej konvolúcie. Výpočet jednej konvolúcie je znázornený na obrázku 1.4. Pri skončení konvolúcie pre danú hladinu osi y presunieme filter znovu na ľavý okraj vstupných dát a posunieme sa o veľkosť kroku (stride) po osi y nižšie. Postup aplikujeme pre každý filter danej konvolučnej vrstvy, výsledné mapy konvolúcií skladáme na seba.

Veľkosť výstupu danej konvolučnej vrstvy je daná: $(W - F + 2P)/S + 1$. Kde W je veľkosť vstupu, F je veľkosť konvolučných filtrov, S je krok (stride)

aplikovaný pri konvolúcií a veľkosť nulovej výplne (zero padding) je vyjadrená ako P [5].



Obr. 1.4: Vizuálizácia konvolučnej operácie [6]

1.2.2 Pooling vrstva

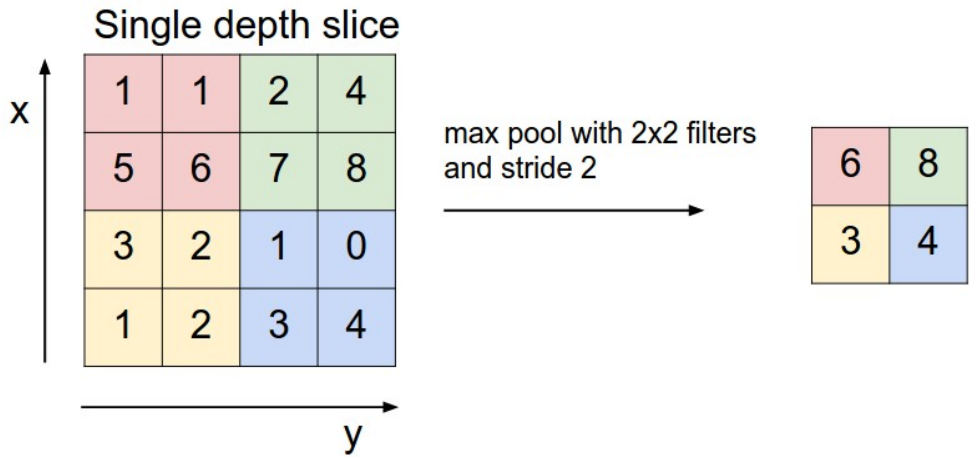
Funkciou pooling vrstvy je redukovať rozmery reprezentácie daného obrázku, a tým znížiť aj množstvo parametrov. Tak ako konvolučná vrstva, pooling vrstva pozostáva z filtrov, ktoré sú aplikované na vstupné dáta. Namiesto násobenia prvkov je tu použitá najčastejšie MAX operácia. Aplikovanie filtrov pooling vrstvy prebieha na každej hladine vstupných dát. Pritom veľkosť hĺbky dát zostáva nezmenená. Obecne pooling vrstva berie na vstupe dáta o veľkosti $W_1 * H_1 * D_1$. Kde W_1 je šírka, H_1 výška, a D_1 hĺbka, tak ako u konvolučnej vrstvy. Potrebujeme určiť dva parametre a to veľkosť používaneho filtra F a veľkosť stride S. Na výstupe produkuje dáta o veľkosti $W_2 * H_2 * D_2$ kde $W_2 = (W_1 - F)/S + 1$ $H_2 = (H_1 - F)/S + 1$ $D_2 = D_1$ Najpoužívanejšou formou v praxi je pooling vrstva s filtrami o veľkosti 2x2 aplikovaná so stride 2 [5].

1.2.3 Fully-connected vrstva

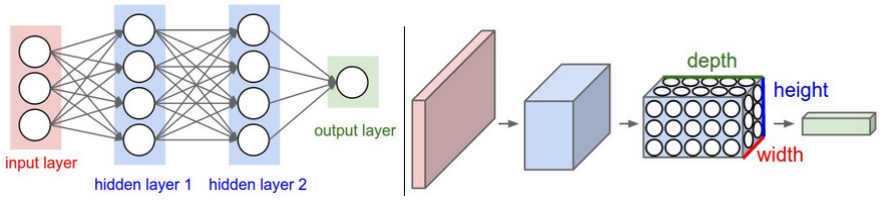
Neuróny v tejto vrstve sú spojené so všetkými neurónmi z predošlej vrstvy tak ako v klasickej FC neurónovej sieti. Pre ďalšie podrobnosti o tejto vrstve viď [19].

1.2.4 Detekcia objektov

Detekcia generických objektov je úloha, kde sa zameriavame na lokalizáciu a zároveň klasifikáciu daného objektu na obrázku. Lokalizácia je zväčša prevedená pomocou takzvaných bounding boxov (ohraničením objektu v rámci



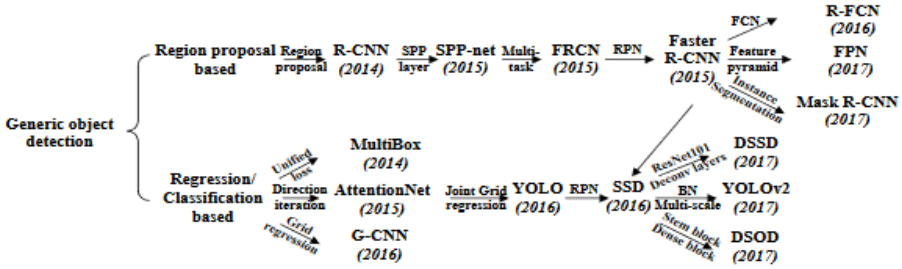
Obr. 1.5: Vizualizácia MAX pooling operácie prevzaté z: [5]



Obr. 1.6: Provnanie FC siete (vľavo) a konvolučnej siete (vpravo) prevzaté z: [5]

obrázku pomocou pravouhlého štvoruholníka). Klasifikácia objektu je následne robená klasickým olabelovaním objektu podľa toho, do akej triedy podľa predikcie spadá. Architektúry modelov na generickú detekciu objektov môžeme rozdeliť na dve skupiny viď 1.7:

- prvá založená na základe návrhov regiónov (Region proposal based)
- druhá založená na regresii/klasifikácii (regresion/classification based)



Obr. 1.7: Rozdelenie modelov prevzaté z: [7]

1.3 Region Proposal Based Framework

Hlavnými predstaviteľmi modelov založených na návrhoch regiónov sú:

- R-CNN - Region-based-CNN,
- Fast-R-CNN - zlepšenie region based,
- Faster-R-CNN - zlepšenie Fast-R-CNN,
- Mask-R-CNN - okrem detekcie objektov pridaná segmentácia inštancií objektov.

Základným princípom je generovanie návrhov regiónov (region proposals) kde by sa mohol vyskytovať objekt. V práci [20] bol tento jav dosiahnutý nahradením klasifikačných vrstiev predtrénovanej ConvNet siete regresnou sieťou a jej trénovaním na predikciou bounding boxov na rôznych škálach a lokalizáciách v rámci obrázku.

1.3.1 Model R-CNN - Regions with CNN features

Detekcia pomocou modelu R-CNN je rozdelená do troch fáz:

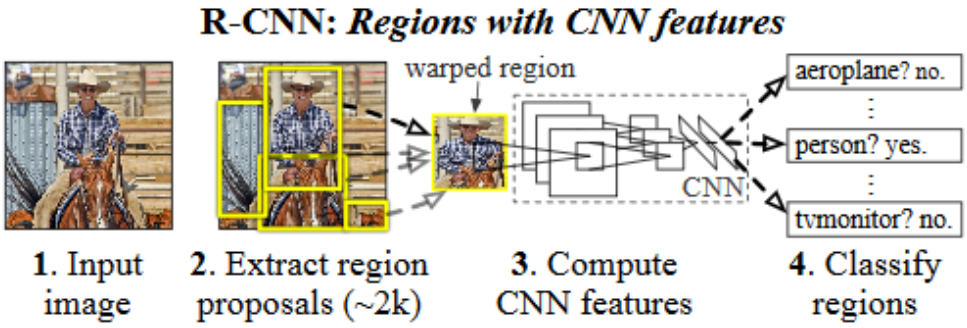
- generovanie regiónov,
- extrahovanie vlastností (features),
- vlastná klasifikácia.

Algoritmus v testovacom čase pomocou selective search generuje cez 2000 návrhov regiónov (region proposals). Tieto regióny sú kategoricky nezávislé, generované bez ohľadu na triedu objektu, ktorý by sa v nich mohol vyskytovať.

Selective search [21] prebieha nasledovne:

- Vygenerovanie počiatočnej sub-segmentácie - mnoho potencionálnych regiónov.

- Použitie greedy algoritmu na rekurzívne spájanie podobných regiónov do väčších.
- Použitie generovaných regiónov na produkovanie finálnych návrhov regiónov.



Obr. 1.8: Ukážka priebehu detekcie R-CNN modelu [8]

Návrhy regiónov sú následne vstupom pre konvolučnú neurónovú sieť. Konvolučná sieť z každého regiónu extrahuje feature vektor dĺžky 4096 prvkov. Následne sa klasifikuje každý región pomocou support-vector machines (SVM) pre každú triedu, na ktorú je model trénovaný. Každý SVM je špecifický pre jednu triedu, ktorú predikuje. R-CNN model je obecne nezávislý na použitej metóde pre generovanie návrhov regiónov (region proposals). Okrem predikovania prítomnosti objektu v danom regióne, algoritmus predikuje štyri hodnoty, ktoré predstavujú offsety zvyšujúce presnosť predikcie bounding boxov. Problémy modelu R-CNN:

- dlhý čas trénovania siete - hlavne kvôli 2000 návrhom pre jeden obrázok,
- pomalá evaluácia pri testovaní - nemožnosť real time behu,
- selective search sa netrénuje s celým modelom.

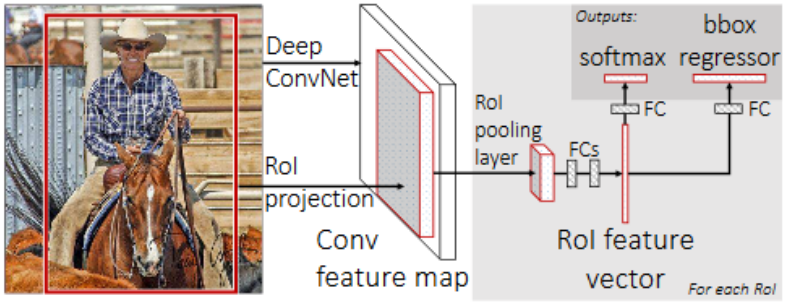
Následne na tieto ohodnotené regióny aplikujeme greedy non-maximum suppression, ktorý zamietne región ak má intersection over union s regiónom vyššieho skóre nad daným prahom (thresholdom). Odstránime duplicitné bounding boxy s nižšou hodnotou skóre, ktoré detekujú ten istý objekt.

1.3.2 Fast R-CNN

Vzhľadom na problémy, ktoré sa vyskytli pri sieti R-CNN napríklad pomalé trénovanie modelu, pamäťová, časová náročnosť a rýchlosť testovania modelu R-CNN. V práci [9] bolo navrhnuté zlepšenie modelu v podobe Fast R-CNN.

Fast R-CNN ponúka end-to-end trénovanie celého modelu. Namiesto feedovania regiónov do CNN siete jedného po druhom, vezme model celý obrázok

spolu so všetkými ponúkanými regiónmi (region proposals) ako vstup do CNN. Do architektúry je ďalej pridaná Region of Interest pooling vrstva medzi poslednú CONV vrstvu a prvú FC vrstvu, ktorá zabezpečuje extrahovanie feature vektoru fixnej dĺžky z feature mapy. Tieto features sú vstupom do sekvencie FC vrstiev, ktoré sa rozdeľujú do dvoch na predikciu výstupu. Predikcia triedy prebieha pomocou softmax vrstvy, ktorá nahradila SVM použité v R-CNN. Druhou výstupnou FC vrstvou je vrstva predikujúca 4 reálne čísla, ktoré predstavujú bounding box pre daný objekt. Architektúra je znázornená na obrázku 1.9 [9].



Obr. 1.9: Ukážka priebehu detekcie Fast R-CNN modelu [9]

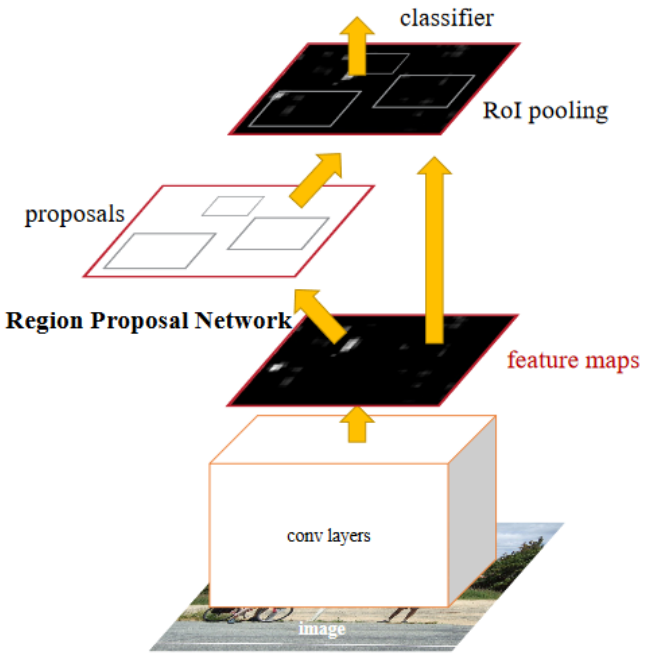
1.3.3 Faster R-CNN

Narozdiel od frameworkov R-CNN a Fast R-CNN sa táto architektúra vyhýba použitiu selective search algoritmu, ktorý je časovo náročný. Selective search je nahradený pomocou CNN siete, ktorá sa sama učí navrhovanie regiónov (region proposals) [10].

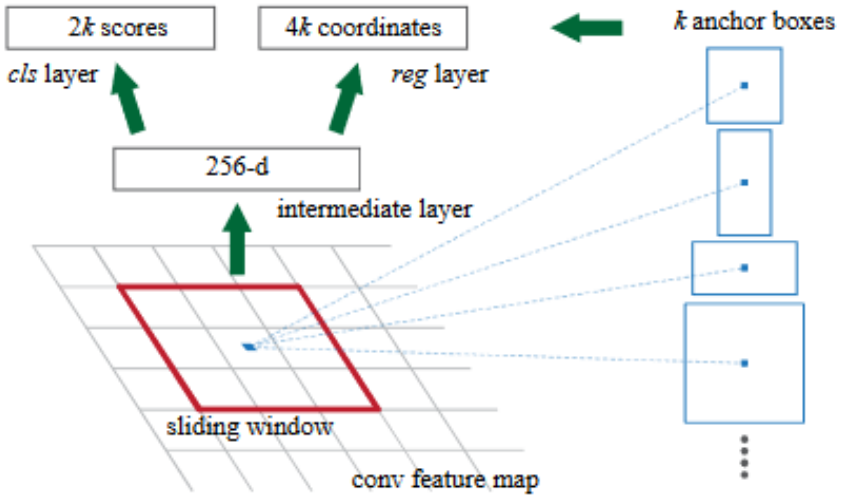
Použitá CNN sieť sa nazýva Region Proposal Network. Rozmery predikovaných regiónov sú zmenené pomocou Region of interest (ROI) pooling vrstvy, ktorá klasifikuje daný región a predikuje offsety pre bounding boxy. Celkovo je teda Faster-RCNN zložená z dvoch modulov. Prvým je hlboká plne konvolučná sieť, ktorá produkuje regióny region proposal network (RPN). Druhý modul je Fast-RCNN detektor. Architektúra siete je znázornená na obrázku 1.10. Obe tieto siete zdieľajú konvolučné vrstvy.

RPN najprv inicializuje k referenčných (reference) boxov - anchor boxov rôznych škál a rozlíšení na každej pozícii CONV feature mapy(výstup z konvolučnej siete viď 1.10). Každá oblasť je namapovaná na menej dimenzionálny vektor, ktorý je vstupom do dvoch FC vrstiev. Tieto vrstvy sú object category vrstva a box regression vrstva. Na obrázku 1.11 je znázornený príklad RPN siete s jednou lokáciou posuvného okna.

Anchor boxy Sú boxy obdĺžnikového tvaru ako môžeme vidieť na obrázku 1.11. Sú to vlastne tvary/pózy v akých sa budú objekty na obrázku nachádzať najčastejšie. V práci [10] sú použité anchor boxy o troch rôznych škálach a



Obr. 1.10: Ukážka priebehu detekcie Faster R-CNN modelu [10]



Obr. 1.11: Ukážka návrhov z jednej pozície pomocou Region proposal network z: [10]

pomeroch strán teda celkovo 9 anchor boxov. Ako sa ukázalo v [2] lepším riešením je tieto anchor boxy vygenerovať pomocou zhlukovacích algoritmov

na trénovacích dátach. Celkovo teda regresná a klasifikačná vrstva produkuje na každej pozícii posuvného okna maximálne $2k$ - pravdepodobností prítomnosti objektu, respektive $4k$ - súradnic bounding boxov.

1.4 One stage Frameworky

Je kategória prediktorov, kde na predikovanie tried a bounding boxov daných objektov stačí jeden priebeh konvolučnou sieťou. Detekcia a učenie týchto modelov sú menej výpočetne náročné. Vzhľadom na charakter architektúry modelov je možné učenie robiť priamo (end-to-end). Hlavnými predstaviteľmi sú:

- OverFeat,
- DetectorNet,
- YOLO,
- YOLOv2/YOLO9000,
- YOLOv3,
- SSD - Single Shot MultiBox Detector,
- CornerNet [17].

V ďalšej časti predstavím architektúry použité v tejto práci spolu s prvými dvomi verziami YOLO architektúr. To najmä z dôvodu, že sú dôležitými a prelomovými v oblasti rozpoznávania objektov. V súčasnej dobe je dostupná už štvrtá verzia YOLO architektúry viď [22].

1.4.1 YOLOv1

You Only Look Once (YOLO) [11] je unifikovaný detektor objektov, ktorý berie detekciu objektov ako regresný problém. Nedochádza tu ku žiadnemu generovaniu návrhov regiónov ako u region proposal frameworkov.

YOLO je jedna konvolučná sieť simultánne predikujúca viacero bounding boxov a pravdepodobností tried pre tieto boxy. Na rozdiel od region proposal architektúr pozerá na obrázok ako na celok. V prvom kroku obrázok rozdelí na bunky (cells) o veľkosti SxS. Objekt patrí do danej bunky obrázku, pokiaľ sa v nej nachádza jeho stred. Každá bunka rozdeleného obrázku (grid cell) je zodpovedná za predikciu objektov, ktoré sa v nej nachádzajú, a zároveň predikuje B bounding boxov a skóre istoty (confidence score) k nim príslušným. Tieto skóre znázorňujú, ako veľmi je model presvedčený, že sa v bounding boxe nachádza objekt, a ako presne bounding box popisuje daný objekt.

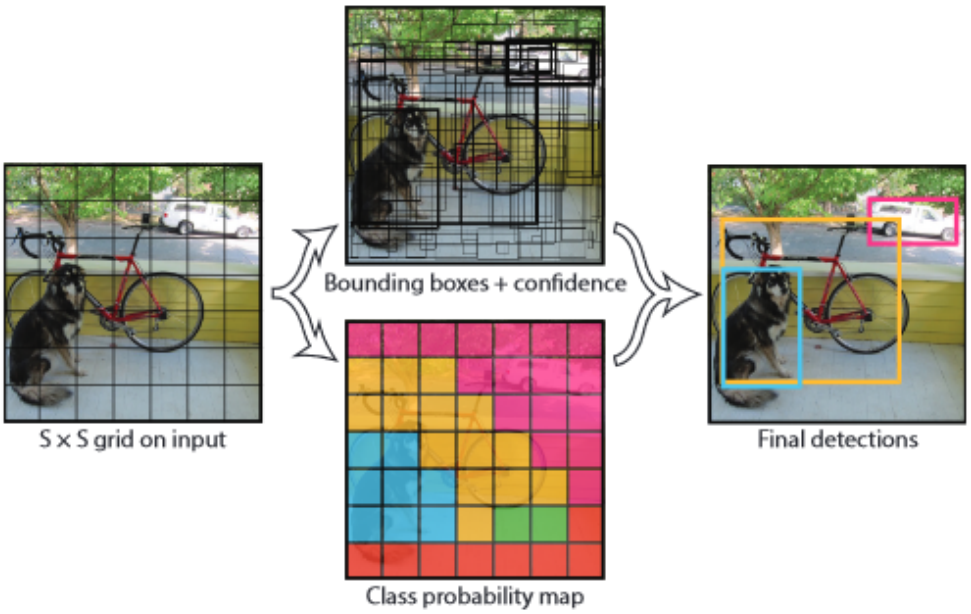
Formálne definujeme skóre istoty (confidence score) ako

$$(Pr(Object) * IOU_{pred}^{truth}.$$

Ak sa v danej časti objekt nenachádza skóre je rovné nule. Inak je skóre rovné IoU medzi predikovaným boxom a pravdivou hodnotou. Každý bounding box

pozostáva z 5 čísel: x, y, w, h a hodnoty skóre istoty. Box je reprezentovaný jeho stredom, ktorého súradnice sú značené ako x, y a ich hodnoty zodpovedajú pozícii v rámci bunky.

Šírka a výška je predikovaná vzhľadom na celý obrázok. Celá sieť má 24 konvolučných a následne dve FC vrstvy [11].



Obr. 1.12: Ukážka detekcie YOLOv1 frameworku prevzaté z: [11]

1.4.2 YOLOv2/YOLO9000 [2]

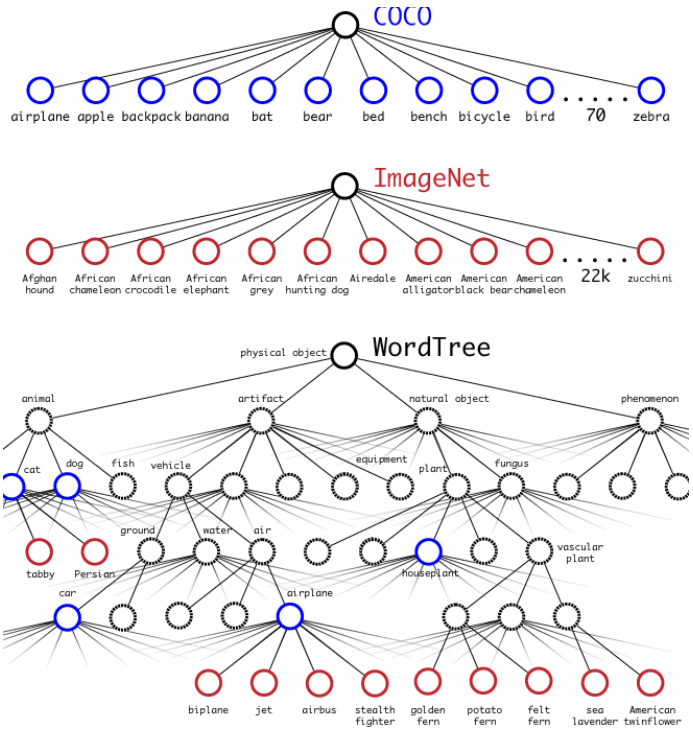
V práci [2] bola predstavená druhá verzia YOLO frameworku. Vzhľadom na to, že už prvá verzia bola značne rýchlejšia, než aktuálne region proposal frameworky, a práve v presnosti detekcie model zaostával, autori sa zamerali najmä na zlepšenie klasifikačnej presnosti. Ďalším problémom, ktorý je v práci riešený, je nepomerne menšie množstvo trénovacích dát na detekčné úlohy oproti tým klasifikačným.

V práci [2] boli zjednotené trénovacie dáta a YOLOv2 bolo trénované na dátach z ImageNet-u [23] a zároveň COCO datasetu. Namiesto pevne stanovených anchor boxov bol použitý k-means algoritmus na bounding boxy trénovacieho COCO a VOC datasetu. Výsledkom bolo 5 anchor boxov (oproti 9 v predošlom modele), ktoré priniesli značné zrýchlenie vo fáze trénovania. Ďalej došlo k úplnému odstráneniu FC vrstvy, čo umožnilo zlepšenie trénovania, a to v podobe viacškálového trénovania (multiscale training).

1.4.3 Viacškálové trénovanie (multiscale training)

Pri trénovaní sa namiesto fixnej veľkosti vstupného obrázku sieť mení každých pár iterácií. Konkrétne každých 10 dávok (batches) sieť náhodne volí novú veľkosť. Konvolučné vrstvy modelu zmenšujú (downsample) reprezentáciu faktorom 32. Rozmery vstupných obrázkov sa pohybujú medzi 320 a 608 (všetko násobky 32) pixelmi. Tento prístup poskytuje možnosť učenia detekcie na rôznych rozlíšeniach obrázkov.

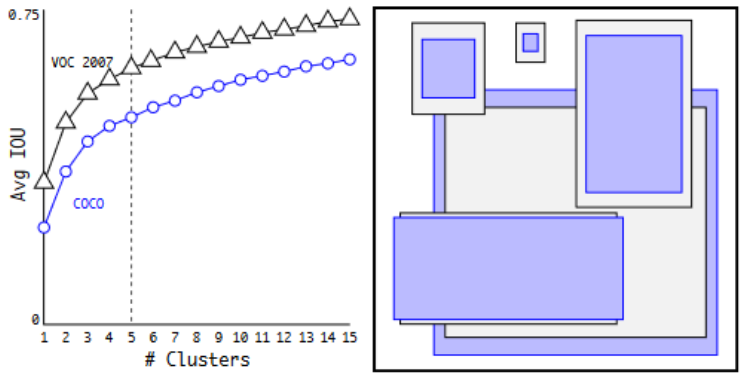
Ďalej bol v práci navrhnutý nový klasifikačný model s názvom Darknet-19. DarkNet-19 pozostáva z 19 konvolučných a 5 max-pooling vrstiev. Pre bližší popis viď [2].



Obr. 1.13: Kombinovanie datasetov ImageNet a COCO za použitia WordTree hierarchie prevzaté z: [2]

1.4.4 SSD

Single shot Multibox Detector (SSD) nepoužíva inú sieť na tvorenie návrhov regiónov (region proposals), namiesto toho predikuje umiestnenie a class skóre použitím malých konvolučných filtrov aplikovaných na feature mapy. Na extrahovanie týchto feature máp používa sieť VCG16. Na rozdiel od YOLOv2 (použitie k-means na hľadanie anchor boxov) sú anchor boxy vybrané manuálne.



Obr. 1.14: Použitie k-means na bounding boxy z VOC a COCO datasetov prevzaté z: [2]

Každá predikcia pozostáva z bounding boxu a $n + 1$ skóre/pravdepodobností pre každú triedu plus jedno skóre predstavujúce pozadie (background) [12].

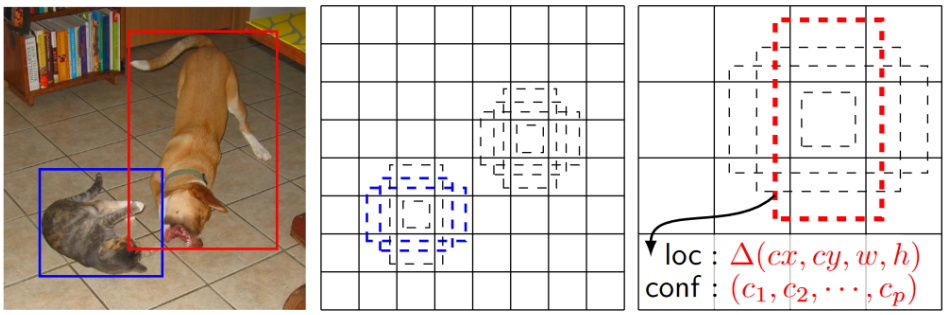
1.4.4.1 Multi scale feature maps

Detekčné modely majú problém s detekciou najmä malých objektov. Z tohto dôvodu bol v práci [12] navrhnutý prístup pre detekciu objektov rôznych rozmerov nazvaný multi scale feature maps. Model v ňom predikuje objekty z feature máp rôznych škál, teda už v priebehu extrahovania features. Konkrétne v SSD architektúre dochádza k predikcii objektov na každej zo šiestich pridaných konvolučných vrstiev viď 1.16.

Na detekciu objektov väčších rozmerov sa hodia feature mapy väčšej granularity, kdežto menšie objekty sú lepšie detekovateľné pomocou feature mapy jemnejšej granularity. Príklad použitia dvoch feature máp rôznych škál je zobrazený na obrázku 1.15. Na tomto obrázku je vidieť, že pes (väčší objekt) je detekovaný pomocou feature mapy 4x4 (väšia granularita), kdežto mačka (menší objekt) je detekovaná na feature mape 8x8 (jemnejšia granularita).

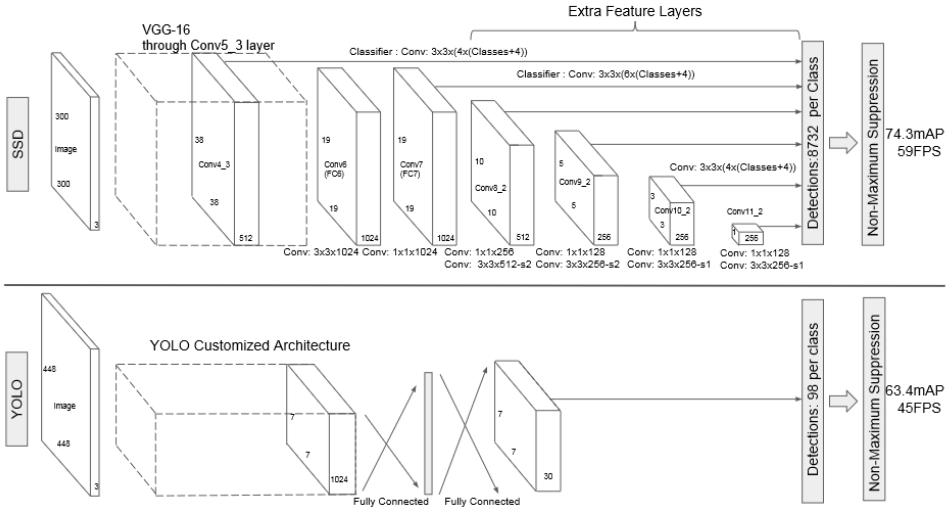
Pokiaľ je predikcia bounding boxov braná ako regresný problém, dochádza v priebehu trénovania k jeho nestabilite (najmä v počiatočných fázach). Dôvodom nestability je náhodná inicializácia váh a následné predikovanie pozícií bounding boxov v rámci celej plochy obrázku.

Pre tento problém sa rozhodli v práci [12] namiesto absolútnych súradníc, použiť offsety relatívne k danému oknu (cell). Predikcie sú brané ako pozitívne, ak daný bounding box má IoU väčšie než 0.5 s pravdivým bounding boxom (labelom daného objektu).



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

Obr. 1.15: Príklad použitia rôznych feature máp na detekciu objektov prevzaté z: [12]



Obr. 1.16: Porovnanie SSD a YOLO architektúry prevzaté z: [12]

1.4.4.2 Stratová funkcia

Nech $x_{ij}^p = 1,0$ je indikátor pre správnu predikciu i-teho default boxu na j-ty pravdivý box pre triedu p. Celková stratová funkcia je vážená suma medzi lokalizačnou stratovou funkciou (loc) a klasifikačnou stratovou funkciou (conf):

$$L(x, c, l, g) = \frac{1}{N}(L_{conf}(x, c) + \alpha L_{loc}(x, l, g)),$$

kde N je počet správnych predikcií default boxov. Lokalizačná stratová funkcia je vyhladená (smooth) L1 funkcia medzi parametrami predikovaného boxu (l) a pravdivého boxu (g). Celkovo počas tréningu dochádza k riešeniu regresného

problému, kde parametre, ktoré sa snažíme optimalizovať, sú umiestnenie stredu (cx, cy) defaultného bounding boxu (d), jeho šírka (w) a výška (h).

$$L(x, l, g) = \sum_{i \in Pos}^{N} \sum_{m \in \{cx,cy,w,h\}} x_{ij}^{k} smooth_{L1}(l_i^m - \hat{g}_j^m)$$

$$\hat{g}_j^{cx} = (g_j^{cx} - d_i^{cx})/d_i^w \quad \hat{g}_j^{cy} = (g_j^{cy} - d_i^{cy})/d_i^h$$

$$\hat{g}_j^w = \log(\frac{g_j^w}{d_i^w}) \quad \hat{g}_j^h = \log(\frac{g_j^h}{d_i^h})$$

Klasifikačná stratová funkcia je softmax loss cez viacero pravdepodobností tried (c).

$$L_{conf}(x, c) = -\sum_{i \in Pos}^{N} x_{ij}^p \log(\hat{c}_i^p) - \sum_{i \in Neg} \log \hat{c}_i^0 \quad \text{,kde} \quad \hat{c}_i^p = \frac{\exp c_i^p}{\sum_p \exp c_i^p}$$

α je nainicializovná na 1 pomocou cross validácie.

Na odstránenie duplicitných predikcií, ktoré detekujú ten istý objekt, je použitý non-maximum suppression.

1.4.4.3 Hard negative mining

Kvôli prevahe pozadia nad objektmi bol v práci použitý aj takzvaný Hard negative mining. Prevaha pozadia na objektmi spôsobuje, že dostávame oveľa viac negatívnych predikcií než pozitívnych. Negatívnymi predikciami sa myslí označenie triedy, ktorá sa na danom mieste nachádza za pozadie (oblasť bez vyskytujúceho sa objektu). Namiesto použitia všetkých negatívnych predikcií, sa tieto zoradia podľa hodnoty ich stratovej funkcie (confidence loss), a vyberú sa tie s jej najväčšou hodnotou tak, aby pomer medzi negatívnymi a pozitívnymi predikciami bol maximálne 3:1. Výsledky SSD modelu na COCO datasete viď 1.17.

Method	data	Avg. Precision, IoU:			Avg. Precision, Area:			Avg. Recall, #Dets:			Avg. Recall, Area:		
		0.5:0.95	0.5	0.75	S	M	L	1	10	100	S	M	L
Fast [6]	train	19.7	35.9	-	-	-	-	-	-	-	-	-	-
Fast [24]	train	20.5	39.9	19.4	4.1	20.0	35.8	21.3	29.5	30.1	7.3	32.1	52.0
Faster [2]	trainval	21.9	42.7	-	-	-	-	-	-	-	-	-	-
ION [24]	train	23.6	43.2	23.6	6.4	24.1	38.3	23.2	32.7	33.5	10.1	37.7	53.6
Faster [25]	trainval	24.2	45.3	23.5	7.7	26.4	37.1	23.8	34.0	34.6	12.0	38.5	54.4
SSD300	trainval35k	23.2	41.2	23.4	5.3	23.2	39.6	22.5	33.2	35.3	9.6	37.6	56.5
SSD512	trainval35k	**26.8**	**46.5**	**27.8**	**9.0**	**28.9**	**41.9**	**24.8**	**37.5**	**39.8**	**14.0**	**43.5**	**59.0**

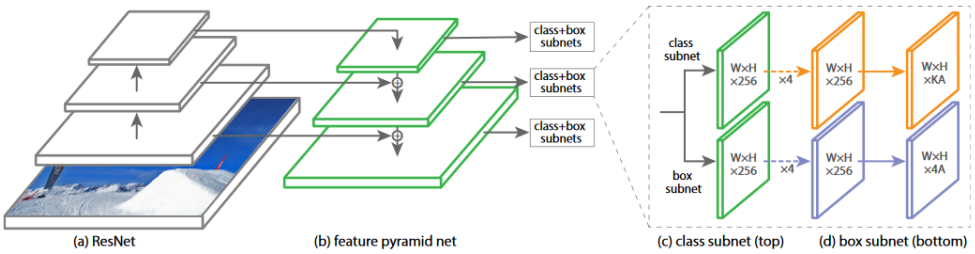
Obr. 1.17: Výsledky SSD modelu na COCO datasete porovnané s ostatnými metódami prevzdaté z: [12]

1.4.5 RetinaNet

V práci Focal Loss for Dense Object Detection [13] bol prezentovaný nový typ stratovej funkcie Focal loss, ktorý pridáva viac váhy pre ťažšie (zle) klasifikované prvky datasetu. Focal loss je dynamicky škálovaná stratová funkcia vzájomnej entropie (cross entropy loss), kde škalovací faktor klesá k nule so vzrastajúcou pravdepodobnosťou správnej klasifikácie. Tak ako bolo poukázané pri SSD modele, počas trénovania dochádza k nevyváženosti background a foreground tried, čo je hlavný dôvod ich nižšej presnosti oproti 2 stage detektorom. V predošlých prácach boli použité heuristiky, ako fixný pomer foreground to background detekcií (1:3) u SSD, alebo OHEM - online hard example mining.

S použitím tejto stratovej funkcie navrhli a natrénovali v práci [13] single stage RetinaNet detector, ktorý prekonal všetky doterajšie modely ako Faster R-CNN, tak aj YOLOv2 1.19.

RetinaNet sa skladá z Feature Pyramid siete, klasifikačnej podsiete a box regression podsiete [13]. Celá architektúra RetinaNet-u je znázornená na obrázku 1.18.



Obr. 1.18: Architektúra siete RetinaNet prevzaté z: [13]

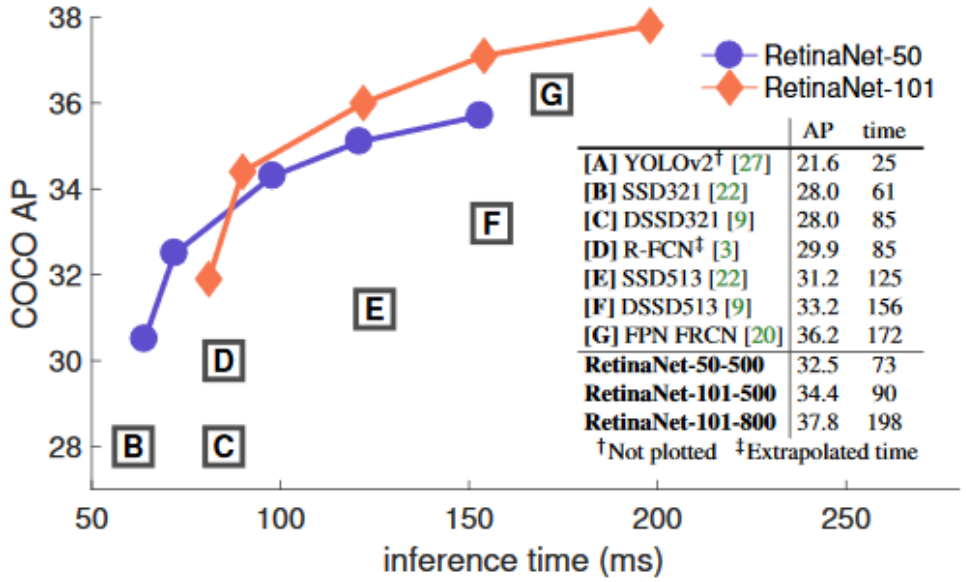
Focal loss Focal loss je definovaný ako

$$FL(p_t) = -(1 - p_t)^{\gamma} \log(p_t),$$

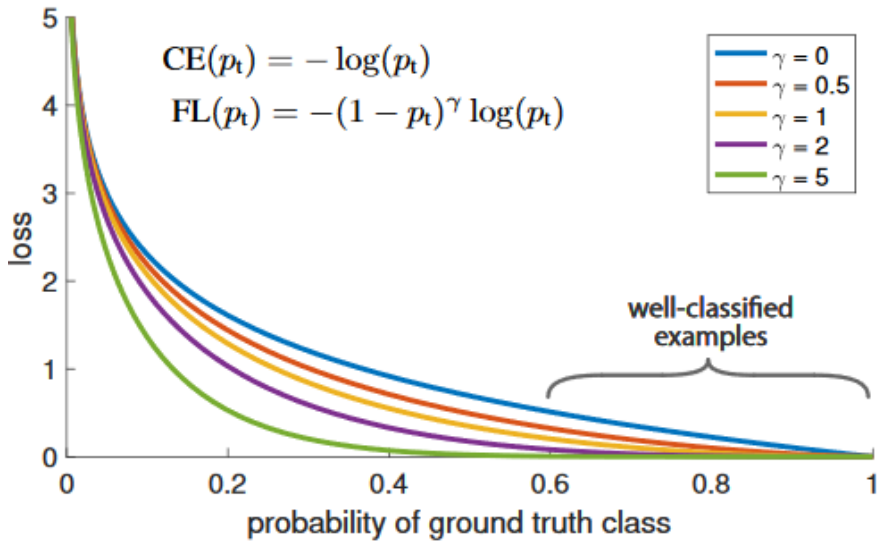
kde $-(1 - p_t)$ je modulačný faktor s ombänným (tunable) parametrom γ. V práci [13] boli zistené dve hlavné vlastnosti focal loss funkcie:

- keď je prvok nesprávne klasifikovaný a p_t je malé, modulačný faktor je blízko 1 a loss zostáva nezmenený,
- ako sa p_t blíži k 1, faktor ide k nule a loss pre správne klasifikované prvky sa znižuje.

Celý jav ako sa mení hodnota stratovej funkcie je znázornený na obrázku 1.20. Viac informácií a podrobnejší popis RetinaNet-u a Focal loss funkcie viď [13].



Obr. 1.19: Porovnanie average pracission RetinaNet s ostatnými state-of-the-art architektúrami prevzaté z: [13]



Obr. 1.20: Porovnanie klasickej stratovej funkcie (modrá) s Focal stratovou funkciou v práci: [13]

1.4.6 MobileNet

MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications [14] je trieda efektívnych mobilných modelov postavených na streamline architektúre ktorá používa novú depthwise separabilnú konvolúciu.

Štandardná konvolučná vrstva berie ako vstup feature mapu F s rozmermi $D_F \times D_F \times M$ a produkuje feature mapu G s rozmermi $D_G \times D_G \times N$, kde D_F je šírka a výška vstupnej feature mapy, viď 1.21. M je počet vstupných kanálov, D_G je šírka a výška výstupnej feature mapy a N je počet výstupných kanálov.

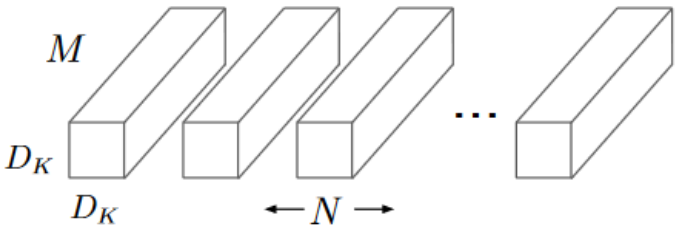
Štandardná konvolučná vrstva je parametrizovaná konvolučným kernelom K veľkosti $D_K \times D_K \times M \times N$, kde D_K je dimenzia kernelu, M počet vstupných a N počet výstupných kanálov.
Výpočetná náročnosť je $D_K \cdot D_K \cdot M \cdot N \cdot D_F \cdot D_F$. Štandardná konvolučná operácia má efekt filtrovania vlastností (features) založený na konvolučných kerneloch a kombinovaní vlastností (features) v snahe vytvoriť novú reprezentáciu.

Toto filtrovanie a kombinovanie môže byt' použitím faktorizovaných konvolúcií, nazývaných depthwise separabilná konvolúcia, rozdelené do dvoch krokov. Depthwise separabilná konvolúcia sa skladá z dvoch častí, depthwise konvolúcie a pointwise konvolúcie. Ich konvolučné filtre sú ilustrované na obrázkoch 1.23 a 1.22.

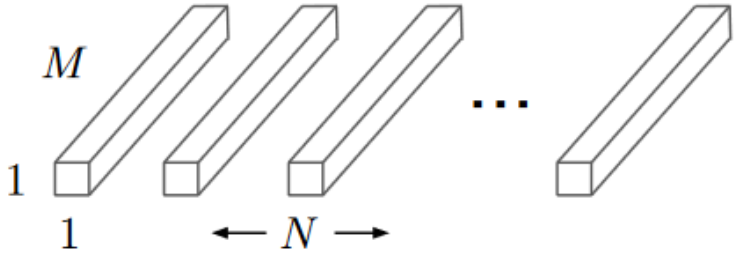
Depthwise konvolúcia aplikuje jeden filter na jeden kanál vstupných dát, pointwise konvolúcia s veľkosťou 1×1 následne vytvára lineárnu kombináciu výstupov depthwise vrstvy. MobileNets riešia túto výpočetnú náročnosť a najprv používajú depthwise separabilnú konvolúciu. Depthwise konvolúcia s jedným filtrom na vstupný kanál, môže byť vyjadrená ako:

$$\hat{G}_{k,l,m} = \sum_{i,j} \hat{K}_{i,j,m} \cdot F_{k+i-1,l+j-1,m}$$

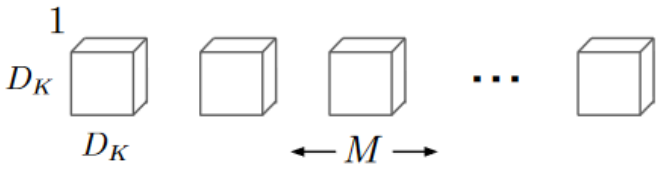
kde $\hat{K}$ je depthwise konvolučný kernel veľkosti $D_K \times D_K \times M$. Výpočetná cena depthwise konvolúcie je $D_K \cdot D_K \cdot M \cdot D_F \cdot D_F$.



Obr. 1.21: Štandardné konvolučné filtre prevzaté z: [14]



Obr. 1.22: Pointwise konvolučné filtre prevzaté z: [14]



Obr. 1.23: Depthwise konvolučné filtre prevzaté z: [14]

Oproti bežnej konvolúcii odpadá N výstupných kanálov (channels). Následné aplikovanie pointwise konvolúcie nám dáva celkovú zložitosť deptwise separabilnej konvolúcie ako $D_K \cdot D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F$. V pomere so štandardnou konvolúciou teda dostaneme

$$\frac{D_K \cdot D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F}{D_K \cdot D_K \cdot M \cdot N \cdot D_F \cdot D_F} = \frac{1}{N} + \frac{1}{D_K^2}$$

MobileNet je postavená na depthwise separabilnej konvolúcii, celkovo sa skladá z 27 konvolučných, jednej average pooling a jednej FC vrstvy. Na klasifikáciu je použitý softmax klasifikátor.

1.4.7 Image stitching

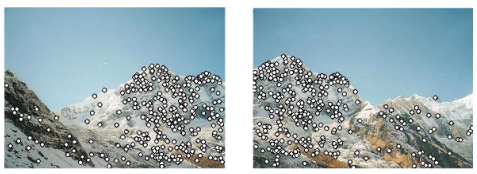
Vzhľadom na to, že našim cieľom je detekovať objekty vzhľadom k celému videu, teda x po sebe idúcich snímok, klasické detekovanie objektov, kde je prevedená detekcia na jednom snímku, je nedostačujúce. To hlavne z dôvodu opakovaného detekovania toho istého objektu viď ďalej ukážka v praktickej časti. Spájanie za sebou idúcich snímok (frames) videa tento problém rieši.

Oblasť automatického spájania obrázkov (automatic image stitching) do panorám je oblasťou, ktorá je dlhé roky v akademickej obci aktívne skúmaná. Výsledkom týchto prác sú už komerčne používané image stitching algoritmy či už v bežných smartfónoch, alebo pri upravovaní fotiek napríklad vo photoshope.

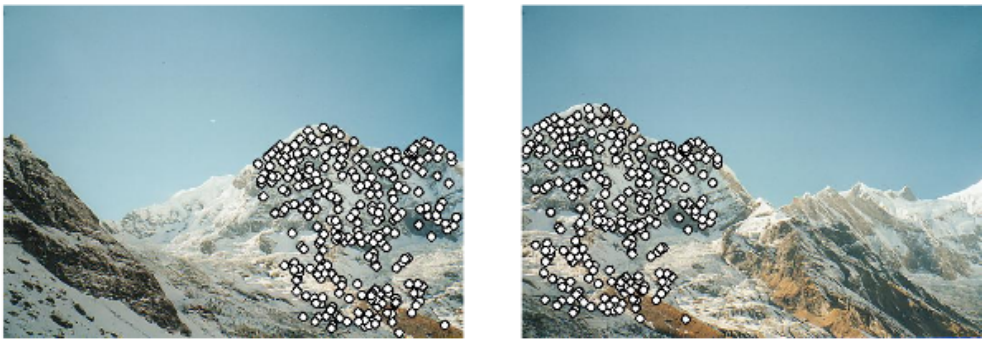
Metódy automatického spájania obrázkov sa dajú rozdeliť na dve skupiny, priame a feature based. Priame metódy majú výhodu použitia všetkých dát obrázkov, ale vyžadujú manuálnu inicializáciu, napríklad rotáciu obrázkov. Feature based metódy nevyžadujú inicializáciu, ale klasickým feature matching metódam chýba vlastnosť invariantnosti.

V práci [15] bol preto predstavený prístup invariantného spájania obrázkov. Hlavnými výhodami tohto prístupu sú napríklad možnosť rotácie, približovania a zmien svetla vo vstupných obrázkoch. Výhodou je aj možnosť hľadania a spájania panorám v nezoradenom datasete. Prvým krokom je extrahovanie a väzbenie SIFT features [24] medzi všetkými obrázkami. Následne nájdenie najbližších features pomocou k-NN (k-nearest neighbors). Následne musí dôjsť k nájdeniu správnych prekrývajúcich sa obrázkov. K tomuto dochádza za pomoci RANSAC (random sample consensus) algoritmu [15]. Ilustrácia priebehu algoritmu na obrázkoch 1.24, 1.25, 1.26, 1.27.

Obr. 1.24: Obrázky, ktoré sú spájané [15]



Obr. 1.25: Všetky SIFT features na oboch obrázkoch [15]



Obr. 1.26: Nájdené Sift matche[15]

Obr. 1.27: Zarovnané obrázky podľa najbližších SIFT features [15]

Kapitola 2

Rešerš / Súčasný stav riešenej problematiky

Počítačové videnie ako oblasť informatiky či znalostného inžinierstva má v praxi mnoho uplatnení, či už ide o oblasť rozpoznávania objektov, alebo ich detekcie. Medzi ukážky aplikované v praxi patria napríklad autonómne autá (self driving cars), rozpoznávanie tváre na smartfónoch a podobne [17].

Cieľom oblasti detekovania objektov je určiť a lokalizovať inštancie istého objektu z daných kategórií (napríklad mačka, človek, auto a podobne), ak prítomné sú, vyznačiť ich na obrázku napríklad pomocou bounding box [16]. Drony (Unmanned Aerial Vehicles) sa vďaka skvelým vlastnostiam, ako je veľkosť, možnosť ovládania na diaľku a vysoká miera mobility, začali vo veľkej miere používať v oblasti detekcie objektov a celkovo počítačového videnia. Riešenie problému automatickej detekcie a rátania vozidiel či už v premávke, alebo na parkoviskách sa stal jedným z hlavných cieľov tejto oblasti. V posledných rokoch bolo pre tento účel predstavených mnoho architektúr konvolučných neurónových sietí.

Jednými z hlavných algoritmov používanými pre tento účel, sú v tejto dobe Faster R-CNN a YOLOv3. Tieto dve metódy boli medzi sebou porovnané v práci [16]. Pre tento účel bol pri práci postavený nový UAV dataset, ktorý bol rozdelený na 218 trénovacích a 52 testovacích prvkov. Pri trénovaní bolo použité Tensorflow Object Detection API. Čo sa výsledkov týka, YOLOv3 vyšlo v porovnaní lepšie než Faster-RCNN viď obrázok 2.1.

Ďalšou prácou, ktorá sa venovala detekcii dopravných prostriedkov, v tomto prípade v reálnom čase, bola Real-Time Vehicle-Detection Method in Bird-View Unmanned-Aerial-Vehicle Imagery [25], kde bola predstavená sieť DRFBNet300, a tak isto aj dataset UAV-cars, ktorý obsahuje 5 035 obrázkov obsahujúcich 21 582 objektov. DRFBNet300 s DRFB modulom dosiahla najvyššie skóre spomedzi ostatných lightweight single-stage metód, v real time dosiahla 21 mAP na 45 frame per second (FPS).

Measure	Faster R-CNN (test dataset)	YOLOv3 (test dataset)
TP (True positives)	578	751
FP (False positives)	2	2
FN (False negatives)	150	7
Precision (TPR)	99.66%	99.73%
Sensitivity (recall)	79.40%	99.07%
F1 Score	88.38%	99.94%
Quality	79.17%	98.81%
Processing time (Av. in ms)	1.39 s	0.057 ms

Obr. 2.1: Výsledky porovnanie YOLOv3 a Faster R-CNN v práci [16]

V práci Drone-based Object Counting by Spatially Regularized Regional Proposal Networks [26] bola predstavená nová architektúra modelov Layout Proposal Networks na simultánne rátanie a lokalizáciu objektov. Tak isto s nimi bol prestavený aj nový dataset zaparkovaných aut CARPK obsahujúci 90 000 aut zachytených na rôznych parkoviskách. V práci boli výsledky siete porovnávané s YOLOv2 a Faster-RCNN, a to na datasete PUCPR+ [27] a CARPK. Na oboch datasetoch dosiahla sieť Layout Proposal Networks najlepšie výsledky spomedzi porovnávaných architektúr. Porovnateľné výsledky dosiahla aj Faster R-CNN so small Region Proposal Network.

Ďalej v práci [28] bola predstavená implementácia algoritmu na automatické detekovanie a rátanie dopravných prostriedkov zachytených dronom. Pre trénovanie a vyhodnotenie výsledkov v tejto práci bol použitý tak isto CARPK [26] a PUCPR+ [27] dataset. Riešenie v práci je postavené na YOLOv3 predtrénovanej na COCO datasete [29]. V práci boli dosiahnuté state-of-the-art výsledky (tvrdia autori). Podľa výsledkov dosahuje riešenie na CARPK datasete recall 95 % a presnosť (precision) 97 %. Na Datasete PUCPR+ kde test set pozostával z 25 obrázkov, bol recall 86 % a presnosť (precision) 95 %.

Výsledky boli porovnávané s riešením v práci [30], kde bola predstavená sieť ShuffleDet. V práci bolo prevedené porovnávanie na datasetoch CARPK a PUCPR+. V porovnaní s touto prácou je ShuffleDet rýchlejšia (14 FPS proti 4 FPS), avšak error ShuffleDet je 7-krát väčší na CARPK datasete a 23-krát väčší na PUCPR+ datasete.

Významnou prácou v oblasti detekovania dopravných prostriedkov pomocou dronov je aj [31]. V tejto práci je predstavené porovnanie (benchmark) v oblasti detekcie a sledovania (tracking) objektov. Obrázky a videá použité v porovnaní (benchmarku) boli zachytené na rozličných miestach v 14 mestách Číny. Konkrétne ide o dataset s 263 videami a 10 209 anotovanými obrázkami. Celkovo je v datasete viac ako 2.5 milióna anotovaných inštancií v 179 264 snímkach. Porovnanie (benchmark) je rozdelené na štyri podúlohy:

- detekciu objektov z obrázkov (object detection in images),
- detekciu objektov z videa (object detection in videos),

- sledovanie jedného objektu (single object tracking),
- sledovanie viacerých objektov (multi-object tracking).

V diplomovej práci [32] bol takisto použitý dataset CARPK spolu s CNR parking lot datasetom [33]. Na detekciu boli použité YOLOv2 a YOLOv3 modely. Práca sa, takisto ako mnoho predtým spomenutých, zameriavala na detekciu obsadenosti daných miest parkoviska z aktuálnej snímky. Na zlepšenie detekcie bola pridaná jedna detekčná vrstva, s ktorou podľa výsledkov bola dosiahnutá hranica 95 % presnosti. Avšak z výsledkov nie je zrejmé, na akej vzorke testovacích dát.

V práci [34] bola predstavená Guided attention network na detekciu a rátanie objektov pomocou dronov. Sieť bola postavená na VGG-16 a ResNet-50. Experimenty znovu prebehli na CARPK, PUCPR+ spolu s UAVDT datasetom. Výsledky na CARPK datasete ukazujú, že architektúra prekonala SSD, Faster R-CNN ako aj YOLOv3 architektúru.

V práci [35] sa autor venoval detekcii áut na parkovisku. Na trénovanie bol použitý Pklot dataset. Počas práce boli natrénované dva modely Faster R-CNN atrous a Faster R-CNN 101. Následne boli prevedené experimenty ohľadom techník trénovania modelov. V práci sa ukázala atrous verzia R-CNN modelu ako lepšia, to hlavne z dôvodu vysokej miery falošne negatívnych (false negative) predikcií modelu Faster R-CNN-101.

Datasety používané na detekciu áut z dronu

V tejto časti predstavím datasety zamerané na detekciu dopravných prostriedkov. Väčšina z týchto datasetov zachytáva autá v bežnej premávke. Okrem týchto datasetov samozrejme existuje veľké množstvo iných trénovacích dát pre detekciu objektov z dronu. Najčastejšie nimi bývajú práve chodci.

V práci [36] bol predstavený benchmark porovnávajúci datasety používané pre detekciu a trackovanie jedného a viacerých objektov (single/multiple object tracking/detection). Vrámci práce bol skonštruovaný nový UAVDT dataset obsahujúci 80 000 olabelovaných snímok pomocou bounding boxov. Dataset je zameraný na tri základné úlohy: detekciu objektov, sledovanie jedného objektu (single object tracking) a sledovanie vicerých objektov (multiple object tracking). Dataset obsahuje snímky zachytené pomocou dronu v rôznych komplexných scénach. Tento fakt je však prakážkou pri použití v tejto práci, pretože okrem dát obsahujúcich autá zachytené na parkoviskách, obsahuje aj bežne vyskytujúce sa zábery z cestnej premávky a podobne.

Ďalej je tu CNRPark+EXT dataset [33], ktorý obsahuje dva subsety CNRPark a CNR-EXT. Celkovo ide o približne 150 000 snímok zachytávajúcich parkovisko z rôznych statických kamier. Pri CNR-EXT datasete sa jedná o dáta zbierané počas 23 dní z deviatich rôznych kamier. CNRPark je jeho predchodca pozostávajúci z 12 000 snímkov zachytených z dvoch statických kamier. V práci "Deep learning for decentralized parking lot occupancy detection" [37] bol tento dataset použitý na implementáciu systému vizuálne detekujúceho ob-

sadenosť parkovísk. Vzhľadom na statické snímanie miest však nebol vhodný pre túto prácu.

PUCPR dataset [1], ktorého rozšírená verzia (PUCPR) sa nachádza aj pri datasete použitom v práci, obsahuje parkovisko zachytené z desiateho poschodia administratívnej budovy v rôznom počasí a časoch.

Pklot dataset [27] obsahuje 12 417 obrázkov parkovísk a 695 899 obrázkov parkovacích miest. Tieto boli zachytené na dvoch parkoviskách. UFPR04 a UFPR05 predstavujú subsety obrázkov zachytávajúcich to isté parkovisko z dvoch rôznych miest, konkrétne štvrté a piate poschodie budovy UFPR. A ďalším subsetom je PUCPR, už vyššie zmienený dataset.

Kapitola 3

Realizácia

V tejto kapitole predstavím praktickú časť mojej práce, použité postupy pri implementácii riešenia, augmentáciu dát aj použité datasety. Takisto popíšem priebeh trénovania vybraných modelov a ich dosiahnuté výsledky. Problémy, ktoré sa počas ich trénovania vyskytli, ako aj možné prístupy pre ich riešenie. Implementované modely otestujem a porovnám ich dosiahnuté výsledky. Na záver zhodnotím dosiahnuté výsledky a navrhnem možné zlepšenia navrhovaného riešenia, ako aj ďalšie možnosti jeho rozšírenia.

3.1 Navrhované riešenie

Problém, ktorý sa práca snaží riešiť, je detekcia áut z obrazového záznamu vytvoreného pomocou letiaceho dronu. Konkrétne využitie si môžeme predstaviť tak, že autonómny dron vylietava z jedného štartovacieho bodu, obletí svoju trasu, počas ktorej sníma cieľovú oblasť, a dáta pošle na server, kde prebieha detekcia na daných snímkach, alebo urobí detekciu dát priamo počas letu v reálnom čase. Týmto spôsobom sa vytvára obraz o obsadenosti daného parkoviska.

Ku detekcii objektov na videu sa dá pristupovať ako ku sledovaniu viacerých objektov (multi object tracking) problému spojenému s detekciou viacerých objektov (multi object detection). V prvom kroku prevedieme detekciu objektov na aktuálnej snímke videa, následne prevedieme multi object tracking - teda označenie už detekovaných objektov tak, aby sme po príchode ďalšej snímky videa presne vedeli, ktoré objekty sú nové, ktoré na snímke len zmenili svoju pozíciu a ktoré sa z obrazu stratili. Takýmto iterovaním všetkých snímok získame celkový pohľad na zaplnenosť daného parkoviska. Tento spôsob je výpočetne náročný a vzhľadom na výpočetnú silu hardwaru, ktorý unesie daný dron (mobilné zariadenia), by bolo jeho real time prevedenie obtiažne.

Druhou možnosťou je skladanie jednotlivých snímok (frames) videa do ortofotomapy a následné prevedenie detekcie nad celým takto získaným obra-

zom. Tento spôsob so sebou prináša určité nepresnosti spojené s prevádzaním záznamu do jednej snímky, ale aj určitú časovú latenciu. Vzhľadom na fakt, že detekcia mala prebiehať ako dávkové prelietavanie dronu nad parkoviskom, bola však detekcia prevádzaná na serveri (s určitou latenciou) dostačujúca. Kvôli jednoduchšiemu prístupu k finálnej detekcii objektov a vyhnutiu sa multi object tracking problému bol tak nakoniec pre prácu zvolený daný prístup.

Celkový priebeh fungovania danej aplikácie by mal teda vyzerať nasledovne:

- dané parkovisko je nasnímané pomocou dronu,
- zo získaného záznamu je vyskladaná ortofotomapa parkoviska,
- nad zloženou ortofotomapou je prevedená detekcia pomocou zvoleného object detection modelu.

3.1.1 Výber datasetu

Na trénovanie modelov bol vybraný CARPK dataset [1]. Tento, ako bolo spomenuté v rešeršnej časti práce, bol hojne využívaný v prácach v posledných rokoch. Výber ovplyvnila najmä kvalita nasnímaných záznamov a charakter datasetu. Tento dataset na rozdiel od ostatných bol nasnímaný na 4 rôznych parkoviskách. Obsahoval teda výhradne fotky a videové záznamy z parkovísk. Jeho ďalšou výhodou boli snímky zachytávajúce parkovisko v priebehu letu dronu, tieto boli ideálne na neskoršie skladanie do ortofotomapy. Snímky v datasete boli zhotovené z približne 40 metrovej výšky, celkom sa v datasete nachádza 1500 snímok s vyše 90 000 objektmi.

3.1.2 Tvorenie ortofoto

Na skladanie ortofotomáp zo snímok bol vytvorený script v Pythone. Tu bola využitá knižnica opencv [38], konkrétne trieda Stitcher.

3.1.3 Výber modelov

Na trénovanie všetkých modelov bolo použité Tensorflow object detection API [39]. Toto API som si vybral hlavne z dôvodu veľkého množstva predtrénovaných modelov na bežných object detection datasetoch ako COCO a Kitti a relatívne jednoduchého rozhrania na trénovanie týchto modelov pre vlastné datasety. V čase písania tejto diplomovej práce bol zverejnený len object detection Zoo pre Tensorflow v1.x. Z tohto dôvodu bol pri trénovaní modelov použitý Tensorflow v1.15.3 [40], a preto boli zvolené aj dané modely.

V nasledujúcej časti predstavím modely, ktoré boli počas práce použité. Prvý uvediem názov modelu, ktorý zodpovedá názvu uvedenému v object detection API model Zoo pre Tensorflow v1. Následne bližšie vysvetlenie archi-

tektúry daného modelu. Všetky použité modely boli predtrénované na COCO datasete.

V práci som sa pokúsil vybrať rôznorodé modely z hľadiska presnosti a rýchlosti detekcie nameranej na COCO datasete. To najmä z dôvodu, že pri trénovaní na tomto datasete sa počet tried z pôvodných 91 (z COCO datasetu) zníži na jednou triedu (auto). Predpokladal som, že by tým pádom mohla ich presnosť (ktorá bola meraná na COCO datasete) výrazne vzrásť. Modely boli vyberané od najrýchlejšieho s najnižšou nameranou presnosťou po najpomalší s najvyššou nameranou presnosťou.

- ssd_mobilenet_v1_coco - SSD s Mobilenet v1,
- ssd_resnet_50_fpn_coco - RetinaNet - SSD s Resnet 50 v1 FPN feature extraktorom, shared box prediktorom a použitím focal loss,
- faster_rcnn_resnet101_coco - Faster R-CNN s Resnet-101 (v1),
- faster_rcnn_inception_resnet_v2_atrous_coco - Faster R-CNN s Inception Resnet v2, Atrous verzia.

3.1.4 SSD s Mobilenet v1

Tento model bol zvolený z dôvodu jeho vysokej rýchlosti, 30ms na obrázok, ktorú dosahoval pre COCO dataset [4]. Jeho presnosť na tomto datasete bola 21 mAP pre 14 minival set [39]. Avšak vzhľadom na to, že COCO dataset obsahuje 91 tried som predpokladal, že by sa mohla presnosť modelu značne zvýšiť. Faktorom ovplyvňujúcim rýchlosť evaluácie je fakt, že čas 30ms na obrázok bol nameraný pre obrázky veľkosti 600x600 pixelov. Vzhľadom na to, že môj trénovací dataset pozostáva z obrázkov o veľkosti 1280x720 pixelov predpokladalo sa spomalenie evaluácie.

3.1.5 RetinaNet

Druhým použitým modelom v práci bol RetinaNet model. Tento model podľa práce [13] dosahoval v implementácií RetinaNet50 na COCO datasete 32.5 mAP a rýchlosť 73ms na snímku. Model bol znovu predtrénovaný na COCO datasete. Podľa nameraných údajov uvedených pri danom modele dosahoval 35 mAP a rýchlosť 76ms, čo sú porovnateľné údaje s uvedenými v oficiálnom článku.

3.1.6 Faster R-CNN resnet 101

Ďalším použitým modelom je Faster R-CNN model konkrétne teda implementácia faster_rcnn_resnet101_coco. Presnosť modelu na COCO datasete dosahovala 32 mAP pri rýchlosti 106ms na snímku.

3.1.7 Faster R-CNN s Inception Resnet v2 (Atrous verzia)

Faster R-CNN s Inception Resnet v2 a Atrous v object detection API predtrénovaný model s názvom faster_rcnn_inception_resnet_v2_atrous_coco. Tento model bol použitý najmä z dôvodu vysokej presnosti 37 mAP. Jeho rýchlosť 620ms na snímku však zďaleka nestačí na real time detekciu. Avšak vzhľadom na charakter navrhovaného riešenia nebol real time beh modelu potrebný. Okrem iného v práci [35] bolo poukázané na nižšiu mieru False positive predikcií v porovnaní s resnet-101 modelom.

3.2 Implementácia

Všetky použité modely boli trénované pomocou Tensorflow Object Detection API [39]. Voľba parametrov bola prevedená na základe predpripravených konfiguračných súborov, ktoré boli použité pri samotnom trénovaní daných modelov. Podľa týchto konfigurácií sa nechali stávajúce hodnoty väčšiny parametrov. Zmenené boli najmä časti pre augmentáciu dát a, samozrejme, počty trénovacích krokov. Následne bol implementovaný script v jazyku Python na automatické vytváranie ortofotomáp. Tento mal slúžiť na jednoduché zjednotenie framov videa na ktorých sa vyskytujú autá, ktoré sme chceli detekovať. Daný prístup mal zabezpečiť validné detekovanie áut na parkovisku.

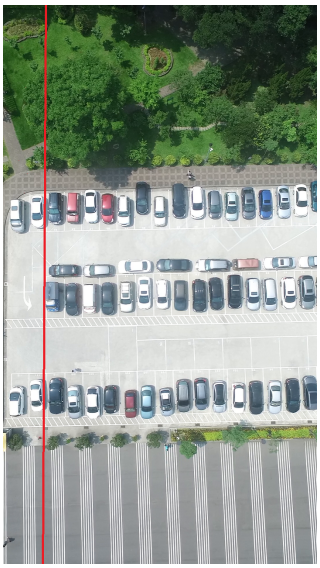
Na nasledujúcom príklade vysvetlím výhody a nevýhody prevedenia detekcie na vytvorenej ortofotomape. Na príklade na obrázku číslo 3.1 a 3.2 môžeme vidieť zachytené parkovisko s autami počas letu dronu (obrázky pochádzajú z CARPK datasetu).

Ako si môžeme všimnúť na oboch obrázkoch sa nachádza značná časť zachyteného parkoviska, ktorú majú spoločnú (oblasti označené červenou čiarou). Ak chceme detekovať všetky autá nachádzajúce sa na danom úseku, musíme urobiť zjednotenie týchto obrázkov a následný odpočet ich prieniku (princíp inklúzie a exklúzie). Tento mechanizmus nám zabezpečí vytvorenie otrofotomapy.

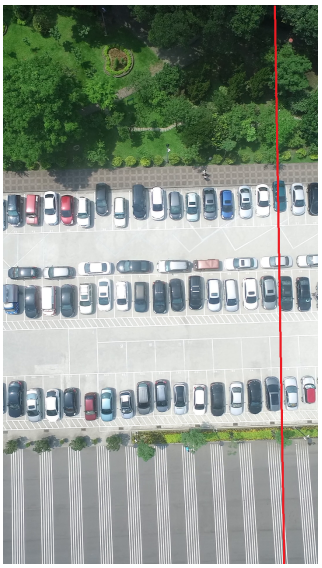
Výsledná ortophotomapa týchto obrázkov viď 3.3. Následne môžeme na obrázkoch 3.4, 3.5 a 3.6 vidieť prevedenie detekcie áut. Táto detekcia bola urobená pomocou Faster R-CNN modelu. Celkovo je na výslednom obrázku 66 áut s tým, že jedno na pravom okraji je ťažko identifikovateľné voľným okom.

3.3 Dataset

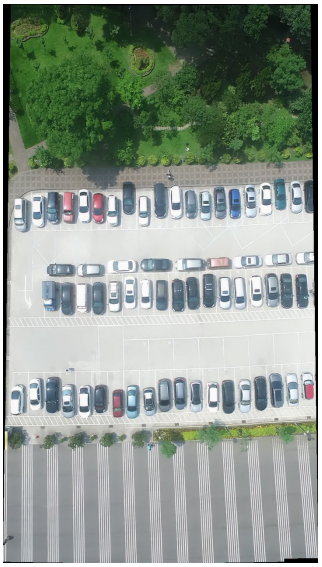
Na trénovanie modelov bol použitý CARPK dataset. Tento obsahuje 90 000 áut zachytených z dronu na 4 rôznych parkoviskách. Snímky sú zachytávané z približne 40 metrovej výšky. Všetky snímky majú rozlíšenie 1280 x 720 pixelov, čo je klasické 16:9 rozlíšenie. Na labelovanie objektov sú použité bounding



Obr. 3.1: Prvý frame videa



Obr. 3.2: Druhý frame videa



Obr. 3.3: Spojenie týchto frameov do jedného

Obr. 3.4: Detekcia objektov na prvom obrázku počet detekovaných objektov: 58

Obr. 3.5: Detekcia objektov na druhom obrázku počet detekovaných objektov: 62

Obr. 3.6: Detekcia objektov na spojení oboch obrázkov počet detekovaných objektov: 66.

boxy. Dataset nerozlišuje typy vozidiel, obsahuje teda len jednu triedu, a to auto. Dataset je súčasťou datasetu použitého v práci [1], v ktorom sa ešte

nachádza aj PUCPR+ dataset. PUCPR+ je rozšírením pôvodného PUCPR datasetu [27]. Na zachytenie snímok bol použitý dron PHANTOM 3 PROFESSIONAL. Súčasťou datasetu sú textové súbory, ktoré obsahujú rozdelenie datasetu na testovaciu a trénovaciu množinu. Ďalej sa tu nachádzajú scripty v jazyku Python 2 na vykresľovanie bounding boxov podľa daných labelov apod. Tieto však pri práci neboli použité.

3.3.1 Príprava dát

Dataset CARPK obsahuje už prerozdelené dáta na trénovací a testovací subset, toto prerozdelenie bolo použité aj v práci. Anotácie dát sú v textových súboroch (*.txt) a labelujú jedno auto na jeden riadok. Názov súboru zodpovedá názvu obrázku, ktorého sa anotácia týka. Anotácia jedného auta popisuje súradnice ľavého horného a pravého dolného rohu bounding boxu. Tieto súradnice sú absolútne vzhľadom na celkové rozmery obrázka. Tzn. obrázky sú rozmeru 1280x720px, súradnice bounding boxu $(1019, 521)$, $(1129, 571)$ v tvare (x, y), kde $(1019, 521)$ je súradnica ľavého horného rohu a $(1129, 571)$ je súradnica pravého dolného rohu. Tieto bolo nutné previesť do tfrecord formátu s relatívnymi hodnotami súradníc. Pre tento účel bol implementovaný Jupyter notebook - data_preparation.

Dáta boli rozdelené tak, že testovacia množina obsahuje 460 snímok a trénovacia množina 990 snímok (bez použitia augmentácie). Jednotlivé snímky sa nachádzajú v zložkách images/test a images/train.

3.3.2 Augumentácia dát

Augmentácia dát bola prevedená priamo pomocou object detection API. V konfiguračných súboroch (*.config) na to slúži parameter data_augmentation_options. Všetky dostupné možnosti augmentácie dát sa nachádzajú v možnosti augmentácie dát. Augmentácia dát bola na začiatku experimentov zachovaná tak, ako v trénovaní pôvodných modelov. Počas evaluácie na orthophoto sa však ukázalo, že detekcia na horizontálne zaparkovaných autách funguje značne horšie, než na vertikálne zaparkovaných. Z tohto dôvodu bola augmentácia dát doplnená najmä o ďalšie rotácie obrázkov. Použité metódy v práci sú:

- random_crop_image - Náhodne oreže obrázok a bounding boxy.
- random_rotation90 - Náhodne rotuje obrázok a detekcie o 90 stupňov v smere hodinových ručičiek - defaultná pravdepodobnosť otočenia 50 %.
- random_vertical_flip - Náhodne rotuje obrázok a detekcie o 90 stupňov v smere hodinových ručičiek - defaultná pravdepodobnosť otočenia 50 %.

- random_adjust_brightness - Náhodne mení svetlosť obrázka do maximálneho delta parametra, obrázok bude saturovaný medzi hodnotami 0 až 1. Defaultná hodnota 0.2.
- random_horizontal_flip - Náhodne horizontálne otočí obrázok a detekcie so špecifikovanou pravdepodobnosťou - defaultná pravdepodobnosť 50 %.

Bližší popis týchto metód a ich parametrov je možné nájsť v zdrojových kódoch Object Detection API. Počas experimentov pri trénovaní modelov boli vyskúšané aj možnosti automatickej augmentácie dát. Tieto sa zadávajú ako parameter autoaugment_image do konfiguračného súboru pre trénovanie modelu. Sada týchto augmentácií bola navrhnutá v práci [41]. Počas trénovania však značne spomaľovala trénovací progress a nevykazovala dobré výsledky, preto ďalej v práci nebola použitá.

Tak isto bolo počas práce trénovaných niekoľko ďalších modelov, ako napríklad SSD s mobilenet verzie 2. Ďalej boli trénované stávajúce modely Faster R-CNN s menšími hodnotami anchor boxov, z dôvodu zlej detekcie malých objektov. Ani jeden z týchto pokusov však nepriniesol zlepšenie výkonnosti modelov, preto v práci boli testy na týchto modeloch vynechané.

3.4 Trénovanie modelov

V tejto časti popíšem použitý hardware a sofware, ako aj vlastnosti vybraných predtrénovaných modelov. Predstavím použité hyperparametre na ich trénovanie a argumentujem zvolený výber.

3.4.1 Použitý hardware a software

Na trénovanie modelov bol použitý počítač z laboratória spracovania obrazu na Fakulte informačných technológií ČVUT v Prahe - ImproLab s názvom Fractal. Tento má nasledujúce parametre:

Procesor	Intel Core i5 7600K CPU @ 3.80GHz
Grafická karta	Nvidia GeForce RTX 2080 Ti

Tabuľka 3.1: Hardware použitý na trénovanie model

Pri písaní scriptov na trénovanie modelov a spájanie ortofoto boli použité okrem bežných knižníc opencv, PIL, TensorFlow v1.15.3, TensorFlow v2, matplotlib, numpy. Tieto aj ostatné použité knižnice sa nachádzajú v súbore requirements_training.txt a requirements_inferention.txt.

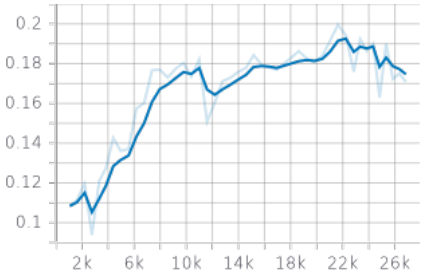
3.4.2 SSD Mobilenet v1

Na trénovanie bol použitý predtrénovaný model ssd_mobilenet_v1_coco. Konfigurácia tréningu bola robená na základe konfigurácie ssd_mobilenet_v1_pets, táto sa nijako nelíši od konfigurácie použitej na COCO datasete (okrem počtu klasifikačných tried). Na základe empirických experimentov bol použitý dropout s pravdepodobnosťou 0.8 na zachovanie väzby. Batch size bol nastavený na 32 prvkov. Augmentácia dát bola robená pomocou metód:

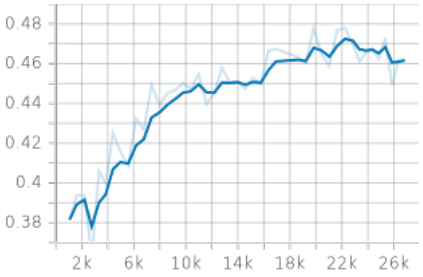
- random_crop_image,
- random_rotation90,
- random_vertical_flip,
- random_adjust_brightness,
- random_horizontal_flip.

Vo všetkých týchto metódach boli ponechané defaultné hodnoty viď zdrojové kódy práce. Veľkosť vstupných dát bola ponechaná na defaultných 300 x 300 pixelov. Trénovanie modelu bolo zastavené po 26 000 krokoch, kde model dosiahol 0.47 mean average precision pre 50% Intersection over Union ($mAP^{IoU=.50}$) viď 3.8. Jeho mean average recall pre sto detekcií na obrázok ($mAR^{max=100}$) bol 0.34.

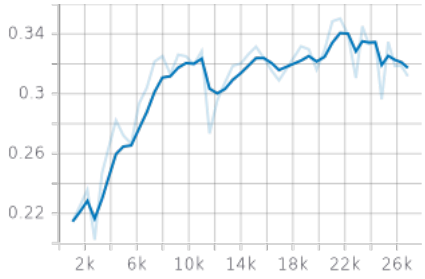
Všetky nasledujúce grafy vykresľujú metriky namerané na validačných dátach.



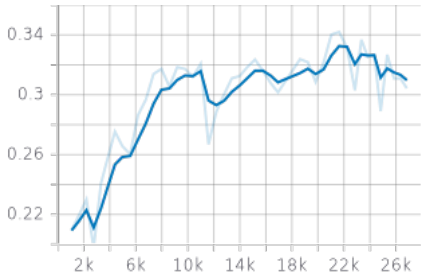
Obr. 3.7: SSD Mobilenet: Vývoj mAP



Obr. 3.8: SSD Mobilenet: Vývoj mAP@.50IOU



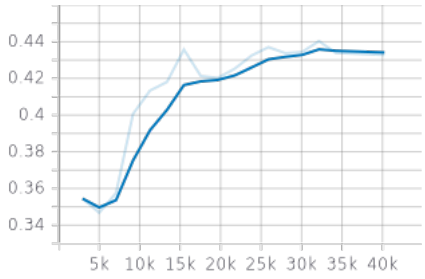
Obr. 3.9: SSD Mobilenet: Vývoj AR@100 pre médium objekty



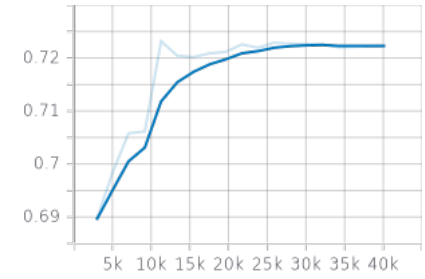
Obr. 3.10: SSD Mobilenet: Vývoj AR@100

3.4.3 Faster R-CNN resnet 101

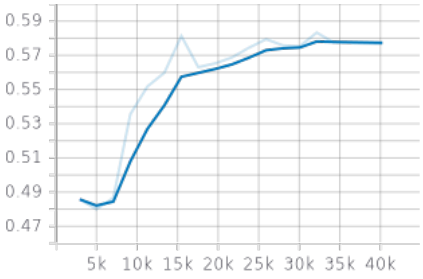
Na dotrénovanie bol použitý predtrénovaný model faster_rcnn_resnet101_coco_2018_01_28. Konfigurácia tréningu bola robená na základe príslušnej konfigurácie použitej pri trénovaní faster_rcnn_resnet101_coco. Tu bola kvôli veľkosti trénovacích dát zmenená veľkosť image_resizer parametru na 720x1280 pixelov. Na základe empirických experimentov bol použitý dropout s pravdepodobnosťou 0.8 na zachovanie väzby. K už použitým dátovým augmentáciám bola pridaná 90-stupňová náhodná rotácia, náhodne vertikálne otočenie a náhodná zmena svetlosti. Počet krokov bol zmenený na 80 000. Celý konfiguračný súbor použitý pri trénovaní viď zdrojový kód práce. Trénovanie modelu bolo zastavené po 40 000 krokoch, kedy nenastávalo väčšie zlepšenie ako mAP tak aj mAR. Nasledujúce grafy vykresľujú všetky metriky namerané na validačných dátach. Model dosiahol 0.72 $mAP^{IoU=.50}$ viď 3.8 a $mAR^{max=100}$ 0.57.



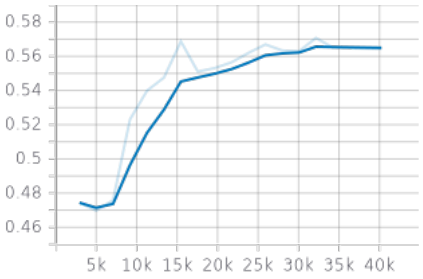
Obr. 3.11: Faster-R-CNN: Vývoj mAP



Obr. 3.12: Faster-R-CNN: Vývoj mAP@.50IOU



Obr. 3.13: Faster-R-CNN: Vývoj AR@100 - medium objekty



Obr. 3.14: Faster-R-CNN: Vývoj AR@100

3.4.4 RetinaNet

Podľa výsledkov presnosti predtrénovanej RetinaNet architektúry mala byť RetinaNet rýchlejšia, avšak aj presnejšia, než Faster R-CNN. Namerané hodnoty podľa [39]. Faster R-CNN:

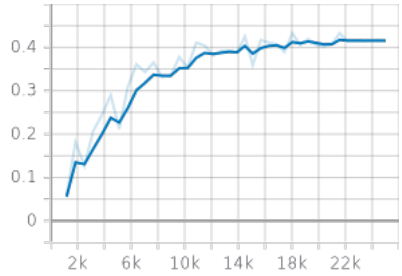
- rýchlosť: 106ms,
- presnosť: 32mAP.

RetinaNet:

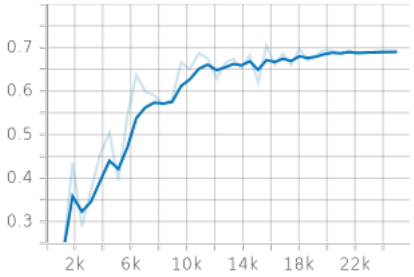
- rýchlosť: 76ms,
- presnosť: 35mAP.

Na jej trénovanie bola využitá jej predtrénovaná verzia ssd_resnet_50_fpn_coco. Parametre trénovania vychádzali z konfiguračného súboru použitého na predtrénovanie ssd_resnet50_v1_fpn_shared_box_predictor_640x640_coco14_sync. Resizovanie vstupných dát bolo zanechané na 640x640 pixelov. Batch size bol kvôli veľkosti pamäte nastavený na 8 prvkov. V augmentácii dát boli zachované random_horizontal_flip a takisto aj random_crop_image so stávajúcimi parametrami viď zdrojové kódy. Naviac boli pridané random_rotation90, random_vertical_flip a random_andjust_brightness. Všetky s defaultnými hodnotami. Počet warmup krokov bol zmenený na 2000. Ostatné parametre viď zdrojový kód práce.

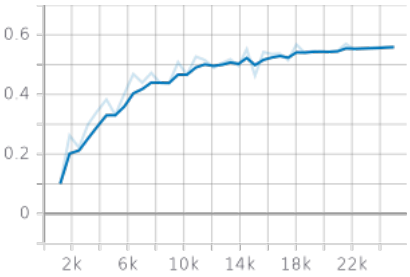
Ako môžeme vidieť z nasledujúcich grafov, model dosiahol $mAP^{IoU=.50}$ 0.69 čo je oproti Mobilenet modelu značne lepšie, avšak o tri stotiny horší výsledok, než Faster R-CNN. Jeho $mAR^{max=100}$ bol taktiež o dve stotiny horší, oproti 0.57 na Faster R-CNN, čo sú však zanedbateľné hodnoty. Tak ako u predošlých modelov, aj tu grafy vykresľujú hodnoty namerané na validačných dátach.



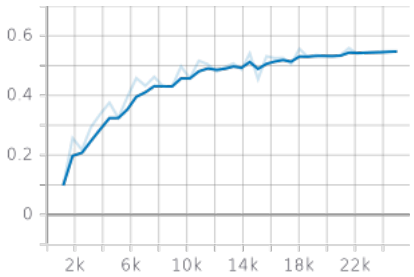
Obr. 3.15: RetinaNet: Vývoj mAP



Obr. 3.16: RetinaNet: Vývoj mAP@.50IOU



Obr. 3.17: RetinaNet: Vývoj AR@100 - medium objekty



Obr. 3.18: RetinaNet: Vývoj AR@100

3.4.5 Faster R-CNN s Inception Resnet v2 (Atrous verzia)

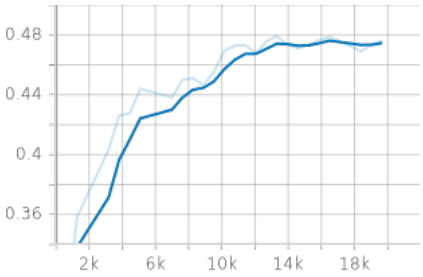
Kvôli nižšej miere falošne pozitívnych (false positive) detekcií som sa rozhodol ešte natrénovať Faster R-CNN vo verzii s inception resnet v2. Táto bola trénovaná na predtrénovanom modele faster_rcnn_inception_resnet_v2_atrous_coco. Pri trénovaní boli použité parametre vychádzajúce z parametrov použitých na jej predtrénovanie na COCO datasete, konkrétne teda faster_rcnn_inception_resnet_v2_atrous_coco.config. Predtrénovaný model detekoval rýchlosťou 620ms na snímok s presnosťou 37mAP. Tieto hodnoty boli znova namerané na COCO datasete.

Zmenený bol maximálny počet krokov na hodnotu 80 000, avšak trénovanie bolo zastavené po 20 000 krokoch. Tak isto boli upravené medze zmien learning rates, kde initial learning rate bola znížená na 0.0002 kvôli príliš veľkým zmenám váh. Tieto pri pokuse s hodnotou initial learning rate 0.0003 spôsobovali problémy s konvergenciou. Augmentácia dát tak ako u predošlých modelov zahŕňa:

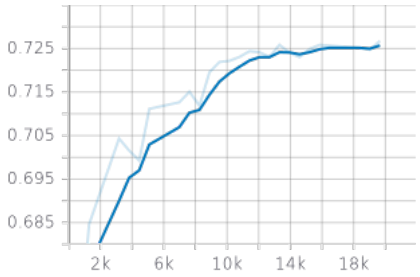
- random_crop_image,
- random_rotation90,

- random_vertical_flip,
- random_adjust_brightness,
- random_horizontal_flip.

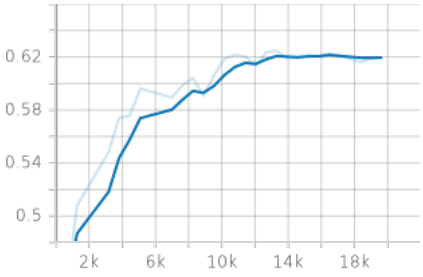
Trénovanie modelu bolo zastavené na hranici 19 668 krokov, tu dosahoval model $mAP^{IoU=.50}$ hodnotu 0.726. $mAR^{max=100}$ dosahoval 0.605, čo sú znova porovnateľné hodnoty s dvoma predošlými modelmi.



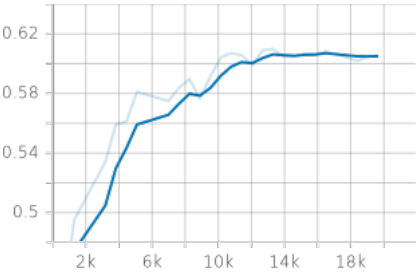
Obr. 3.19: Faster-R-CNN Atrous: Vývoj mAP



Obr. 3.20: Faster-R-CNN Atrous: Vývoj mAP@.50IOU



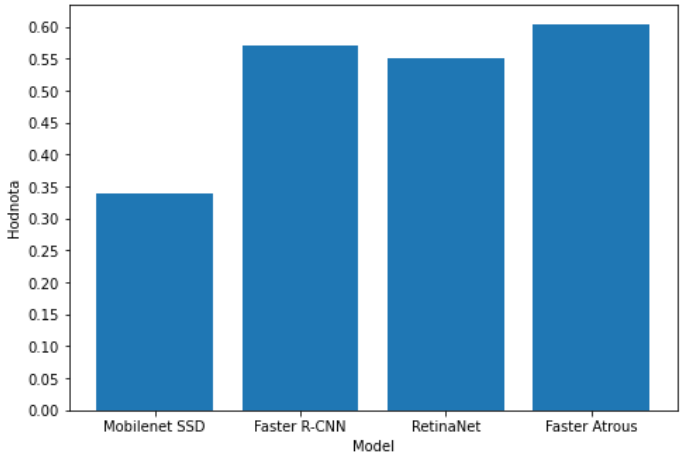
Obr. 3.21: Faster-R-CNN Atrous: Vývoj AR@100 - medium objekty



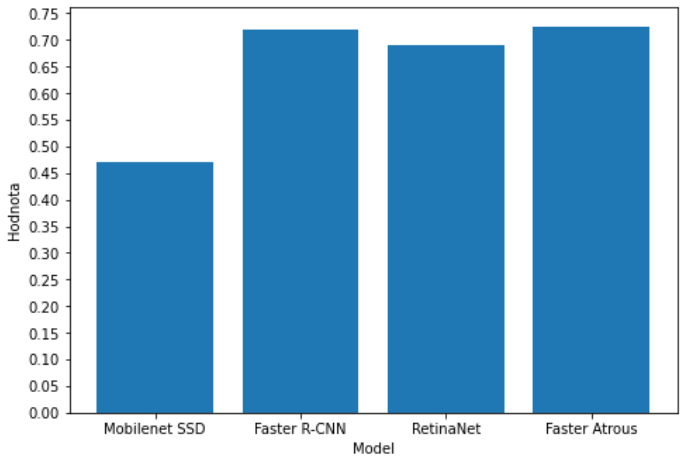
Obr. 3.22: Faster-R-CNN Atrous: Vývoj AR@100

3.4.6 Výsledky trénovania

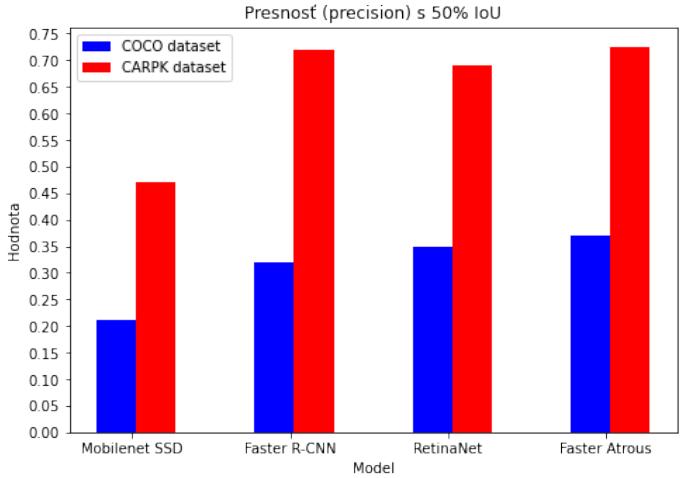
Ns nasledujúcom grafe 3.24 sú znazornené dosiahnuté hodnoty mAP modelov. Ako môžeme vidieť, podľa očakávania Faster R-CNN a RetinNet modely dosiahli lepšie výsledky než Mobilenet. Čo je však trochu prekvapujúce je výsledok RetinaNet oproti Faster R-CNN modelom, RetinaNet mal byť podľa výsledkov uvedených v oficiálnom paperi presnejší. Ďalej je na grafe číslo 3.23 znázornený dosiahnutý recall modelov. Na grafe 3.25 je znázornené porovnanie mAP hodnôt dosiahnutých na COCO datasete proti tým na CARPK datasete. Podľa očakávania došlo k značnému zlepšeniu. Z grafu je vidieť, že najväčšie zlepšenie dosiahol Faster R-CNN s resnet101 (značený na grafe len ako Faster R-CNN).



Obr. 3.23: Porovnanie dosiahnutých recall hodnôt na CARPK datasete



Obr. 3.24: Porovnanie dosiahnutých hodnôt presností (precision) na CARPK datasete



Obr. 3.25: Porovnanie mAP hodnôt dosiahnutých modelmi na COCO datasete oproti dosiahnutým na CARPK datasete po dotrénovaní

3.5 Skladanie ortofotomapy

Pre vyskladanie ortofotomáp bol použitý script implementovaný v jazyku Python. Script berie ako vstup zložku, v ktorej sú snímky z daného preletu dronom. Testovaný bol na snímkach z CARPK datasetu a takisto nalietaných snímkach na ulici Kolejní, ktorá sa nachádza pri budove Fakulty informačných technológií. Snímky boli urobené v rozmedzí jednej sekundy. Predzpracovanie snímok zahŕňalo ich rotáciu v smere preletu z zľava doprava, ďalej zmenšenie rozlíšenia čo zabezpečilo zrýchlenie behu skladania a detekcie.

Script podľa zadaných parametrov zloží ortofotomapu z daných vstupných dát. Parametrami je možné nastavovať či bude ortofotomapa skladaná naraz zo všetkých snímok, alebo postupne po častiach kde parametrom je počet snímkov skladaných v jednu iteráciu. Následne sú tieto čiastkové ortofotomapy zložené do jednej finálnej. Táto metóda sa v práci pomocou empirických pokusov ukázala ako časovo a kvalitatívne najvýkonejšia. Defaultným parametrom sú štyri snímky na jednu iteráciu.

Ďalej je možné skladať ortofotomapu len z vybraných snímok. Jednou možnosťou je brať každú x-tú snímku z danej zložky, to ale v prípadoch, keď sú prekryvy snímok malé, môže spôsobiť zlyhanie skladania.

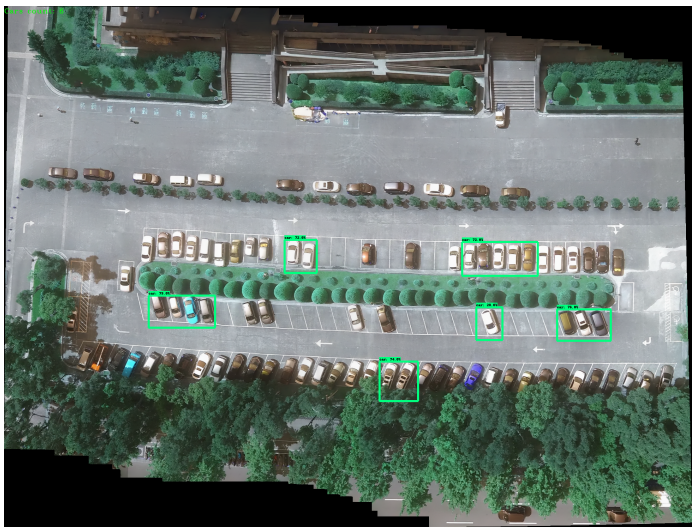
Ďalšou možnosťou je použiť parameter na výber snímok pomocou heuristiky, ktorá používa bitový xor na zistenie rozdielov medzi danými obrázkami. V prípade, že je rozdiel medzi obrázkami v danom rozsahu, je daná snímka vybraná do skladania pre ortofotomapy. Týmto je zabezpečené, že v prípade snímok robených vo veľmi malých intervaloc,h sa nebudú skladať dv snímky, ktorých rozdiel medzi časom ich zachytenia je príliš malý (plochy, ktoré zachytávajú sú takmer totožné). Zároveň sa však vyberú snímky s dostatočnými prekryvmi na spájanie.

Obr. 3.26: Ukážka vyskladanej ortofoto z nalietaných dát, snímok je vyskladaný z 20 frameov.

3.6 Experimenty

V tejto časti popíšem experimenty vyskúšané počas práce. Tieto sa týkajú najmä prístupu k skladaniu ortofotomáp a následnej inferencie nad jej výstupom. Ďalej bude predstavená časť s experimentmi mimo trénovací CARPK dataset na porovnanie výkonnosti výsledných modelov.

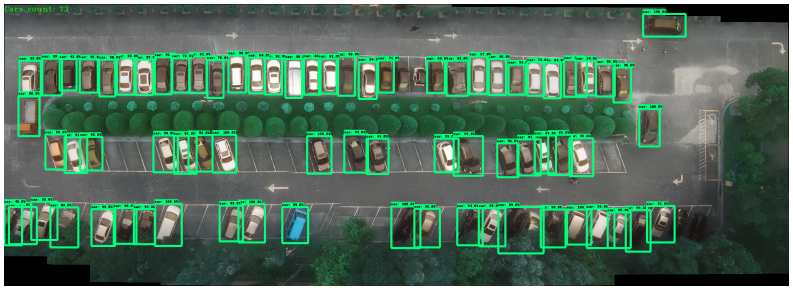
Ako sa počas práce ukázalo, skladanie orofotomapy naráža na viacero problémov. Prvým a najzávažnejším z nich je, že zachytené objekty sú na príliš malé na detekciu pomocou použitých modelov. Teda pokiaľ sú na vytvorenie ortofotomapy použité snímky, ktoré zachytávajú autá vo vysokom rozlíšení, z relatívne veľkej výšky (presná hodnota v metroch záleží na rozlíšení snímky), po prevedení na veľkosť vstupu pre danú sieť sú objekty menšie než 30x30 pixelov. Tieto následne nie je možné pomocou modelov detekovať. Tento jav je vidieť na ukážke z obrázka číslo 3.27. Detekcia tu bola prevedená pomocou Faster R-CNN modelu. Veľkosť snímky je 1949x1477 pixelov. Objekty majú na pôvodnom obrázku veľkosť približne 70x30 pixelov, po prevedení obrázku na veľkosť vstupu do siete, teda 1280x720 pixelov, dostaneme obrázok s rozmermi 950x720 pixelov a objekty o veľkosti 34x14 pixelov. Ako môžeme vidieť, žiaden z objektov nebol na snímke detekovaný. To aj napriek tomu, že séria snímok, z ktorej je ortofoto vyskladané, pochádza z trénovacích dát.



Obr. 3.27: Ukážka neúspešnej detekcie na ortophotomape vygenerovanej zo série snímok pochádzajúcich z trénovacích dát.

Naproti tomu v ukážke na obrázku číslo 3.28 bolo detekovaných 73/78 áut správne. To z dôvodu odlišného pomeru veľkosti zachytených objektov ku veľkosti snímky. Detekcia bola prevedená pomocou Faster R-CNN modelu. Veľkosť snímky je 2217x796 pixelov. Objekty na pôvodnom obrázku majú veľkosť približne 90x40 pixelov, po prevedení na veľkosť vstupu do siete, teda

1280x720 pixelov, dostaneme vstupný obrázok s rozmermi 1280x460 pixelov a objekty o veľkosti 52x26 pixelov. Ako môžeme vidieť, tieto objekty sú už pomocou Faster R-CNN zachytiteľné.



Obr. 3.28: Ukážka úspešnej detekcie na ortofotomape vygenerovanej zo série snímok pochádzajúcich z trénovacích dát.

Z tohto dôvodu je nutné po zložení frameov do ortofotomapy dodatočné rozdelenie výslednej snímky na časti. Toto rozdelenie, vzhľadom na známosť domény, na ktorej je detekcia prevádzaná (predpokladá sa konkrétne parkovisko), je možné robiť automaticky, na základe tvaru daného parkoviska. Tento prístup bol použitý aj v práci počas testovania. Aby sme sa pri inferencii vyhli duplicitnému detekovaniu tých istých objektov, je možné používať heuristiky, ktoré detekujú, že sa jedná o rozdelený objekt. Týmito môže byť napríklad minimálna veľkosť detekovaného objektu spolu s jeho súradnicami.

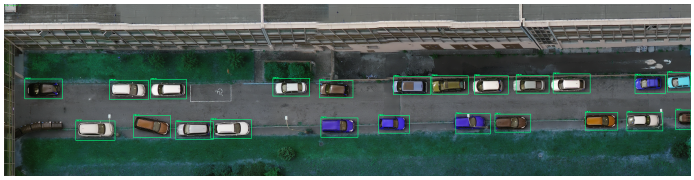
Druhou možnosťou je použitie vhodnejšieho nastavenia kamery a výšky letiaceho dronu, a to s ohľadom na výslednú veľkosť objektov po poskladaní ortofotomapy. To s ohľadom na výslednú veľkosť objektov po poskladaní orthophotomapy.

Treťou možnosťou je, samozrejme, natrénovanie ešte presnejšieho modelu, než aké boli použité v tejto práci. Vhodným kandidátom by mohol byť napríklad YOLOv4.

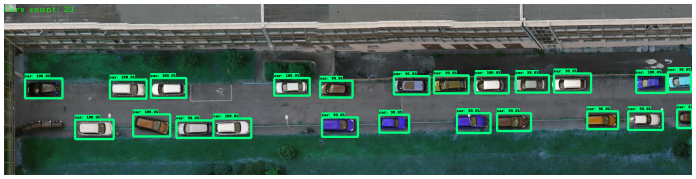
Počas práce boli vyskúšané rôzne prístupy použitia nástroja na skladanie obrázkov do ortofoto. Z týchto vyplýva, že veľkosť rozlíšenia obrázkov nemá vplyv ako na skladanie ortofotomapy, tak ani na detekciu objektov čo sa kvality výsledkov týka. Tu sa samozrejme zvažujú rozlíšenia obrázkov väčšie než vstupné dáta pre danú sieť. Naopak, ako som prepokladal s rastúcim rozlíšením rastie čas potrebný ako na vyskladanie ortofoto, tak na výslednú detekciu.

Ako môžeme vidieť z obrázkov 3.29 a 3.30, detekcie na týchto obrázkoch sú totožné, avšak čas ich skladania a inferečný čas sa výrazne líšia.

Preto bolo ako krok predspracovania pridané zníženie rozlíšenia daných snímok (pri zachovaní pometu strán danej snímky, aby sa zachoval tvar objektov).



Obr. 3.29: Detekcia objektov na obrázku s rozlíšením 7276x1812 pixelov pomocou Faster R-CNN



Obr. 3.30: Detekcia objektov na preškálovanom obrázku s rozlíšením 1800x448 pixelov pomocou Faster R-CNN

Do predspracovania musel byť v niektorých prípadoch pridaný krok orezania originálnych snímok. Toto sa týkalo nalietaných dát na ulici Kojení, kde boli z veľkej časti snímky zachytené okolité budovy. Tie spolu s meniacim zorným uhlom letiaceho dronu spôsobovali zlú kvalitu vyskladaných ortofotomáp.

3.7 Testy na dátach mimo dataset

V tejto časti otestujem všetky štyri natrénované modely. Test je urobený na spojených testovacích dátach z CARPK datasetu s nalietanými dátami z ulice Kolejní. Celkovo tento validačný dataset obsahuje 25 snímok z rozdelených ortofotomáp. Viac informácií o dátach uvediem v nasledujúcej časti.

3.7.1 Použité dáta

Na testovanie bolo použitých 460 snímok z testovacej množiny CARPK datasetu. Tieto snímajú jedno parkovisko, takže celkovo z nich boli vyskladané dve ortofotomapy, ktoré boli následne rozsekané na menšie časti. Veľkosť týchto úsekov nie je jednotná. Okrem týchto snímok boli pomocou dronu DJI Mavic 2 Pro z ulice Kolejní nasnímané dáta v rámci 140 snímok. Výška letu dronu bola v prvej iterácií 40 metrov nad zemou, čo sa ukázalo ako príliš nízko. V ďalšej časti boli preto snímané zábery z výšky 60 metrov. Rozlíšenie snímok je 5472x3648 pixelov. Z týchto následne vznikli 3 ortofotomapy, ktoré boli tiež rozdelené na menšie časti.

Na labelovanie oboch sád snímok bol použitý nástroj labelImg [42]. Kód na prevod YOLO formátu do tfrecord formátu je znova dostupný v Jupyter notebooku v zdrojovom kóde práce. Na testy boli použité COCO metriky.

Validačný set celkovo obsahoval 25 snímok - 13 snímok tvoriacich ortophotomapu z testovacích dát CARPK datasetu a 12 snímok z troch orthophotomáp z ulici Kolejní.

3.7.2 Dosiahnuté výsledky

Všetky dosiahnuté výsledky sa nachádzajú v dodatku A. Ako môžeme vidieť, u small objects sú namerané nízke hodnoty, čo môže byť spôsobené tým, že modely pre malé objekty nefungujú ideálne, ako aj nízkym počtom daných objektov vo validačnom datasete.

V tabuľke 3.7.2 sa nachádzajú namerané hodnoty $mAP^{IoU=.50}$ a $mAR^{max=100}$ pre všetky použité modely. Celkovo na všetkých snímkach najlepšie obstál Faster R-CNN atrous s celkovým $mAR^{max=100}$ 0.54 a $mAP^{IoU=.50}$ s hodnotou 0.933.

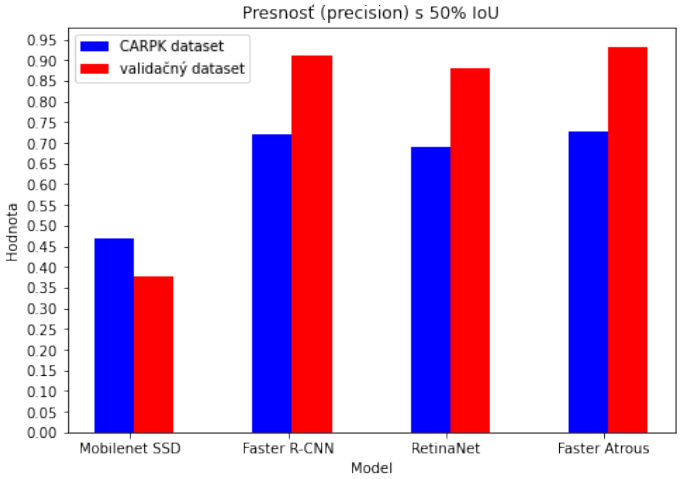
Namerané priemerné presnosti (average precisions) u všetkých testovaných modelov, až na SSD Mobilenet v1, prekonali hodnoty dosiahnuté na validačnom datasete pri ich trénovaní. Tento fakt je znázornený v grafe 3.31.

Porovnanie dosiahnutých $mAP^{IoU=.50}$ danými modelmi je v grafe 3.33, porovnanie $mAR^{max=100}$ hodnôt viď 3.32.

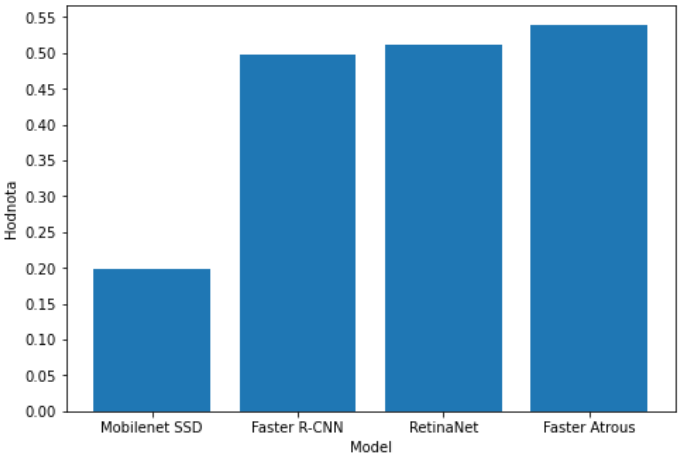
Ako sme mohli vidieť v grafe 3.25, predpoklad zlepšenia presnosti modelov na používanom datasete sa naplnil. Zlepšenie Mobilenet v1 modelu však nebolo dostatočné, následne sa na validačnom datasete ukázalo, že na týchto dátach jeho presnosť ešte viac klesla.

Model	$mAP^{IoU=.50}$	$mAR^{max=100}$
SSD Mobilenet v1	0.376	0.199
Faster R-CNN resnet 101	0.911	0.498
RetinaNet	0.881	0.512
Faster R-CNN s Inception Resnet v2 (Atrous verzia)	0.933	0.540

Tabuľka 3.2: Výsledky modelov na validačnom datasete

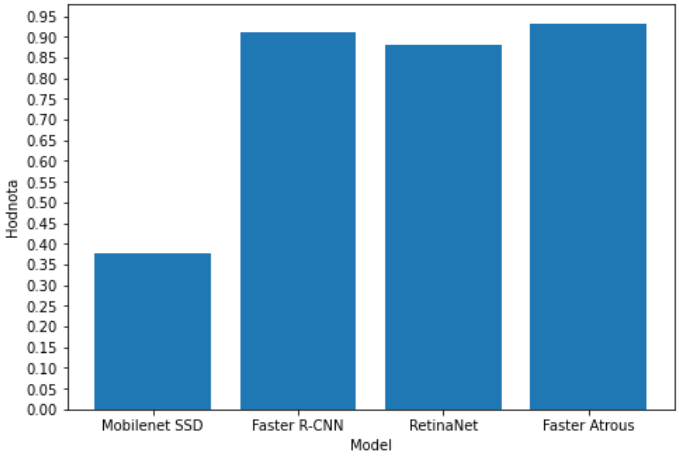


Obr. 3.31: Porovnanie dosiahnutých hodnôt presností (precision) na validačnom datasete oproti CARPK datasetu



Obr. 3.32: Porovnanie dosiahnutých recall hodnôt na validačnom datasete

Lepšie výsledky ostatných troch architektúr môžu byť spôsobené väčšími objektmi, ktoré boli detekované vo finálnom validačnom datasete (z dôvodu



Obr. 3.33: Porovnanie dosiahnutých hodnôt presností (precision) na validačnom datasete

rozsekania ortofoto).

Výsledky dosiahnuté Faster R-CNN architektúrami sú porovnateľné s RetinaNet architektúrou.

Model Faster R-CNN dosiahol najvyšší $mAR^{max=100}$ spomedzi vybraných modelov čím sa potvrdila správnosť jeho voľby. Z tohto dôvodu je používaný vo finálnej implementácií práce.

Záver

V tejto práci som sa zaoberal detekciou zaparkovaných áut z pohybujúceho sa dronu. Cieľom práce bol návrh a implementácia aplikácie, ktorá bude detekovať zaparkované autá z obrazového záznamu vytvoreného pomocou dronu.

Počas práce som implementoval algoritmus, ktorý na základe sekvencie snímok urobených pomocou letiaceho dronu vyskladá ortofotomapu daného parkoviska. Následne na nej prevedie rozdelenie na časti vhodné na detekciu a na týchto čiastkových snímkach detekuje všetky objekty, ktoré sa nachádzajú na danom parkovisku. Tento prístup umožňuje uspokojivé výsledky detekovania aj pre menej presné modely, vzhľadom na veľkosť detekovaných objektov získaných po rozdelení snímkov.

Počas práce bol takisto vytvorený vlastný dataset. Na tomto datasete bolo následne prevedené testovanie danej aplikácie. Možných zlepšení navrhovaného spôsobu detekcie áut na parkovisku sa ponúka hneď niekoľko. Prvou možnosťou ako nadviazať na prácu, je navrhnutie modelu, ktorý dokáže zachytiť aj objekty menšie než približne 25x25 pixelov, a týmto zabezpečiť úspešnú detekciu na snímkach s veľkým rozlíšením.

Ďalšia možnosť je celkové zlepšenie navrhovaného spôsobu skladania orthophotomapy hlavne s cieľom zrýchlenia tohto procesu.

Treťou možnosťou je vytvorenie, resp. použitie heuristík na zamedzenie dvojitej detekcie toho istého objektu rozdeleného do dvoch snímok.

Literatúra

[1] Hsieh, M.-R.; Lin, Y.-L.; Hsu, W. H.: Drone-based Object Counting by Spatially Regularized Regional Proposal Networks. In *The IEEE International Conference on Computer Vision (ICCV)*, IEEE, 2017.

[2] Redmon, J.; Farhadi, A.: YOLO9000: Better, Faster, Stronger. *CoRR*, ročník abs/1612.08242, 2016, 1612.08242. Dostupné z: http://arxiv.org/abs/1612.08242

[3] da Silva, R. P. S. L. N. E. A. B.: Survey on Performance Metrics for Object-Detection Algorithms. *International Conference on Systems, Signals and Image Processing (IWSSIP)*, 2020.

[4] Lin, T.; Maire, M.; Belongie, S. J.; aj.: Microsoft COCO: Common Objects in Context. *CoRR*, ročník abs/1405.0312, 2014, 1405.0312. Dostupné z: http://arxiv.org/abs/1405.0312

[5] Li, F.-F.; Krishna, R.; Xu, D.: Stanford University CS231n: Convolutional Neural Networks for Visual Recognition. 2020. Dostupné z: http://cs231n.stanford.edu/

[6] S. Mohamed, I.: *Detection and Tracking of Pallets using a Laser Rangefinder and Machine Learning Techniques*. Dizertačná práce, 09 2017, doi:10.13140/RG.2.2.30795.69926.

[7] Zhao, Z.; Zheng, P.; Xu, S.; aj.: Object Detection with Deep Learning: A Review. *CoRR*, ročník abs/1807.05511, 2018, 1807.05511. Dostupné z: http://arxiv.org/abs/1807.05511

[8] Girshick, R. B.; Donahue, J.; Darrell, T.; aj.: Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, ročník abs/1311.2524, 2013, 1311.2524. Dostupné z: http://arxiv.org/abs/1311.2524

[9] Girshick, R. B.: Fast R-CNN. *CoRR*, ročník abs/1504.08083, 2015, 1504.08083. Dostupné z: http://arxiv.org/abs/1504.08083

[10] Ren, S.; He, K.; Girshick, R. B.; aj.: Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *CoRR*, ročník abs/1506.01497, 2015, 1506.01497. Dostupné z: http://arxiv.org/abs/1506.01497

[11] Redmon, J.; Divvala, S. K.; Girshick, R. B.; aj.: You Only Look Once: Unified, Real-Time Object Detection. *CoRR*, ročník abs/1506.02640, 2015, 1506.02640. Dostupné z: http://arxiv.org/abs/1506.02640

[12] Liu, W.; Anguelov, D.; Erhan, D.; aj.: SSD: Single Shot MultiBox Detector. *CoRR*, ročník abs/1512.02325, 2015, 1512.02325. Dostupné z: http://arxiv.org/abs/1512.02325

[13] Lin, T.; Goyal, P.; Girshick, R. B.; aj.: Focal Loss for Dense Object Detection. *CoRR*, ročník abs/1708.02002, 2017, 1708.02002. Dostupné z: http://arxiv.org/abs/1708.02002

[14] Howard, A. G.; Zhu, M.; Chen, B.; aj.: MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *CoRR*, ročník abs/1704.04861, 2017, 1704.04861. Dostupné z: http://arxiv.org/abs/1704.04861

[15] Brown, M.; Lowe, D. G.: Automatic Panoramic Image Stitching using Invariant Features. *Int. J. Comput. Vision*, ročník 74, č. 1, 2007: s. 59–73, ISSN 0920-5691, doi:10.1007/s11263-006-0002-3. Dostupné z: http://dl.acm.org/citation.cfm?id=1265138.1265141

[16] Benjdira, B.; Khursheed, T.; Koubaa, A.; aj.: Car Detection using Unmanned Aerial Vehicles: Comparison between Faster R-CNN and YOLOv3. 12 2018.

[17] Liu, L.; Ouyang, W.; Wang, X.; aj.: Deep Learning for Generic Object Detection: A Survey. *CoRR*, ročník abs/1809.02165, 2018, 1809.02165. Dostupné z: http://arxiv.org/abs/1809.02165

[18] Kelley, H. J.: Gradient theory of optimal flight paths. *Ars Journal*, ročník 30, č. 10, 1960: s. 947–954.

[19] Goodfellow, I.; Bengio, Y.; Courville, A.: *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[20] Sermanet, P.; Eigen, D.; Zhang, X.; aj.: OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks. 2013, 1312.6229.

[21] Uijlings, J.; Sande, K.; Gevers, T.; aj.: Selective Search for Object Recognition. *International Journal of Computer Vision*, ročník 104, 09 2013: s. 154–171, doi:10.1007/s11263-013-0620-5.

[22] Bochkovskiy, A.; Wang, C.-Y.; Liao, H.-y.: YOLOv4: Optimal Speed and Accuracy of Object Detection. 04 2020. Dostupné z: `https://arxiv.org/abs/2004.10934`

[23] Russakovsky, O.; Deng, J.; Su, H.; aj.: ImageNet Large Scale Visual Recognition Challenge. *CoRR*, ročník abs/1409.0575, 2014, `1409.0575`. Dostupné z: `http://arxiv.org/abs/1409.0575`

[24] Lowe, D. G.: Distinctive Image Features from Scale-Invariant Keypoints. *Int. J. Comput. Vision*, ročník 60, č. 2, Listopad 2004: s. 91–110, ISSN 0920-5691, doi:10.1023/B:VISI.0000029664.99615.94. Dostupné z: `http://dx.doi.org/10.1023/B:VISI.0000029664.99615.94`

[25] Han, S.; Kwon, S.: Real-Time Vehicle-Detection Method in Bird-View Unmanned-Aerial-Vehicle Imagery. *Sensors*, ročník 19, 09 2019: str. 3958, doi:10.3390/s19183958.

[26] Hsieh, M.-R.; Lin, Y.-L.; Hsu, W.: Drone-Based Object Counting by Spatially Regularized Regional Proposal Network. 10 2017, s. 4165–4173, doi:10.1109/ICCV.2017.446.

[27] Almeida, P.; Soares de Oliveira, L.; Jr, A.; aj.: PKLot - A Robust Dataset for Parking Lot Classification. *Expert Systems with Applications*, ročník 42, 02 2015, doi:10.1016/j.eswa.2015.02.009.

[28] Amato, G.; Ciampi, L.; Falchi, F.; aj.: Counting Vehicles with Deep Learning in Onboard UAV Imagery. 06 2019, s. 1–6, doi:10.1109/ISCC47284.2019.8969620.

[29] Lin, T.-Y.; Maire, M.; Belongie, S.; aj.: Microsoft COCO: Common Objects in Context. 04 2014, doi:10.1007/978-3-319-10602-1_48.

[30] Azimi, S.: ShuffleDet: Real-Time Vehicle Detection Network in On-board Embedded UAV Imagery. 11 2018.

[31] Zhu, P.; Wen, L.; Bian, X.; aj.: Vision Meets Drones: A Challenge. 04 2018.

[32] Ordonia, S.: Detecting Cars in a Parking Lot using Deep-Learning. 2020. Dostupné z: `https://scholarworks.sjsu.edu/cgi/viewcontent.cgi?article=1697&context=etd_projects`

[33] Amato, G.; Carrara, F.; Falchi, F.; aj.: Car parking occupancy detection using smart camera networks and deep learning. In *Computers and Communication (ISCC), 2016 IEEE Symposium on*, IEEE, 2016, s. 1212–1217.

[34] Yuanqiang, C.; Du, D.; Zhang, L.; aj.: Guided Attention Network for Object Detection and Counting on Drones. 09 2019.

[35] Holmström, A.: 2018. Dostupné z: `http://urn.kb.se/resolve?urn=urn:nbn:se:umu:diva-149689`

[36] Du, D.; Qi, Y.; Yu, H.; aj.: The Unmanned Aerial Vehicle Benchmark: Object Detection and Tracking. *CoRR*, ročník abs/1804.00518, 2018, `1804.00518`. Dostupné z: `http://arxiv.org/abs/1804.00518`

[37] Amato, G.; Carrara, F.; Falchi, F.; aj.: Deep learning for decentralized parking lot occupancy detection. *Expert Systems with Applications*, ročník 72, 2017: s. 327–334.

[38] Bradski, G.: The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[39] Huang, J.; Rathod, V.; Sun, C.; aj.: Speed/accuracy trade-offs for modern convolutional object detectors. *CoRR*, ročník abs/1611.10012, 2016, `1611.10012`. Dostupné z: `http://arxiv.org/abs/1611.10012`

[40] Abadi, M.; Agarwal, A.; Barham, P.; aj.: TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. 2015, software available from tensorflow.org. Dostupné z: `http://tensorflow.org/`

[41] Cubuk, E. D.; Zoph, B.; Mané, D.; aj.: AutoAugment: Learning Augmentation Policies from Data. *CoRR*, ročník abs/1805.09501, 2018, `1805.09501`. Dostupné z: `http://arxiv.org/abs/1805.09501`

[42] Tzutalin: LabelImg. Free Software: MIT License, 2015. Dostupné z: `https://github.com/tzutalin/labelImg`

Zoznam použitých skratiek

$mAP^{IoU=.50}$ mean average precision pre 50% Intersection over Union. 40–42, 44, 51

$mAR^{max=100}$ mean average recall pre sto detekcií na obrázok. 40–42, 44, 51

AP average precision. 5

AR average recall. 6

CNN convolutional neural networks. 8, 14

FC Fully Connected. xi, 8, 10, 11, 14, 18, 26

FPS frame per second. 29, 30

IoU Intersection over Union. 4, 6, 17, 20, 63–65

mAP mean average precision. xii, 6, 7, 29, 35, 36, 40, 41, 43–46

mAR mean average recall. 7, 41

ROI Region of interest. 14

RPN region proposal network. 14

SSD Single shot Multibox Detector. ix, 19, 20, 23, 31

SVM support-vector machines. 13, 14

UAV unmanned aerial vehicle. viii, 1, 29

YOLO You Only Look Once. 17, 18

Dodatok A

Dosiahnuté výsledky na validačnom datasete

Faster R-CNN - Kolejni + test dataset - 25 snímok				
Metrika	area	maxDets	IoU	Nameraná hodnota
Average Precision (AP)	all	100	0.50:0.95	0.410
Average Precision (AP)	all	100	0.50	0.911
Average Precision (AP)	all	100	0.75	0.262
Average Precision (AP)	small	100	0.50:0.95	0.006
Average Precision (AP)	medium	100	0.50:0.95	0.387
Average Precision (AP)	large	100	0.50:0.95	0.621
Average Recall (AR)	all	1	0.50:0.95	0.015
Average Recall (AR)	all	10	0.50:0.95	0.128
Average Recall (AR)	all	100	0.50:0.95	0.498
Average Recall (AR)	small	100	0.50:0.95	0.042
Average Recall (AR)	medium	100	0.50:0.95	0.488
Average Recall (AR)	large	100	0.50:0.95	0.674

Tabuľka A.1: Výsledky modelu Faster R-CNN na validačnom datasete

RetinaNet - Kolejni + test dataset - 25 snímok				
Metrika	area	maxDets	IoU	Nameraná hodnota
Average Precision (AP)	all	100	0.50:0.95	0.401
Average Precision (AP)	all	100	0.50	0.881
Average Precision (AP)	all	100	0.75	0.265
Average Precision (AP)	small	100	0.50:0.95	0.035
Average Precision (AP)	medium	100	0.50:0.95	0.382
Average Precision (AP)	large	100	0.50:0.95	0.659
Average Recall (AR)	all	1	0.50:0.95	0.016
Average Recall (AR)	all	10	0.50:0.95	0.125
Average Recall (AR)	all	100	0.50:0.95	0.512
Average Recall (AR)	small	100	0.50:0.95	0.083
Average Recall (AR)	medium	100	0.50:0.95	0.493
Average Recall (AR)	large	100	0.50:0.95	0.745

Tabuľka A.2: Výsledky modelu RetinaNet na validačnom datasete

SSD Mobilenet - Kolejni + test dataset - 25 snímok				
Metrika	area	maxDets	IoU	Nameraná hodnota
Average Precision (AP)	all	100	0.50:0.95	0.098
Average Precision (AP)	all	100	0.50	0.376
Average Precision (AP)	all	100	0.75	0.021
Average Precision (AP)	small	100	0.50:0.95	0.011
Average Precision (AP)	medium	100	0.50:0.95	0.075
Average Precision (AP)	large	100	0.50:0.95	0.532
Average Recall (AR)	all	1	0.50:0.95	0.012
Average Recall (AR)	all	10	0.50:0.95	0.080
Average Recall (AR)	all	100	0.50:0.95	0.199
Average Recall (AR)	small	100	0.50:0.95	0.033
Average Recall (AR)	medium	100	0.50:0.95	0.157
Average Recall (AR)	large	100	0.50:0.95	0.654

Tabuľka A.3: Výsledky modelu SSD Mobilenet na validačnom datasete

Faster R-CNN atrous - Kolejni + test dataset - 25 snímok				
Metrika	area	maxDets	IoU	Nameraná hodnota
Average Precision (AP)	all	100	0.50:0.95	0.447
Average Precision (AP)	all	100	0.50	0.933
Average Precision (AP)	all	100	0.75	0.296
Average Precision (AP)	small	100	0.50:0.95	0.077
Average Precision (AP)	medium	100	0.50:0.95	0.427
Average Precision (AP)	large	100	0.50:0.95	0.714
Average Recall (AR)	all	1	0.50:0.95	0.016
Average Recall (AR)	all	10	0.50:0.95	0.135
Average Recall (AR)	all	100	0.50:0.95	0.540
Average Recall (AR)	small	100	0.50:0.95	0.075
Average Recall (AR)	medium	100	0.50:0.95	0.526
Average Recall (AR)	large	100	0.50:0.95	0.756

Tabuľka A.4: Výsledky modelu Faster R-CNN atrous na validačnom datasete

Dodatok **B**

Obsah priloženého CD

- `src`
  - `model` ..... zdrojové kódy implementácie
    - `annotations` . tfrecords súbory pre trénovanie modelov a label_map
    - `good_models` ..... inferečné grafy troch najúspešnejších modelov
    - `kolejni_dataset` ..... dataset z ulice kolejní
    - `model_main.py` ... script na trénovanie prevzatý z Object Detection API
    - `resize_images.py` ..... script na predzpracovanie snímkov
    - `inferention.ipynb` ..... Jupyter notebook slúžiaci na inferenciu modelov
    - `stitcher.py` ..... script na spájanie snímkov
    - `training.ipynb` . Jupyter notebook slúžiaci na trénovanie modelov
    - `images`
      - `test` ..... snímky testovacieho datasetu
      - `train` ..... snímky trénovacieho datasetu
- `text` ..... text práce
  - `thesis.pdf` ..... text práce vo formáte PDF
  - `thesis.ps` ..... text práce vo formáte PS