VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

MONITOROVÁNÍ CHODCŮ POMOCÍ DRONU

MONITORING PEDESTRIAN BY DRONE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **VLADIMÍR DUŠEK,**
AUTHOR

VEDOUCÍ PRÁCE **Prof. Ing., Dipl.-Ing. MARTIN DRAHANSKÝ, Ph.D.**
SUPERVISOR

BRNO 2019

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav inteligentních systémů (UITS) Akademický rok 2018/2019

Zadání bakalářské práce

Student: **Dušek Vladimír**
Program: Informační technologie
Název: **Monitorování chodců pomocí dronu**
Monitoring Pedestrian by Drone
Kategorie: Zpracování obrazu

Zadání:

1. Prostudujte a sumarizujte možnosti detekce osob v obraze pomocí klasifikačních algoritmů a neuronových sítí. Zaměřte se především na řešení, která umožňují detekci osob z výšky.
2. Seznamte se s dostupnými drony na trhu, analyzujte jejich vlastnosti a vybrané 3-4 porovnejte s ohledem na monitorování chodců.
3. Navrhněte algoritmus pro detekci chodců, který umožní detekovat chodce ve videu. Ke každému chodci pak bude vykreslovat trajektorii do panoramatického snímku z videa.
4. Navržený algoritmus implementujte jako multiplatformní aplikaci. K implementaci použijte programovací jazyk Python.
5. Proveďte experimenty na videích pořízených z dronu. Na základě zjištěných poznatků proveďte zhodnocení funkce algoritmu pro detekci chodců. Vyhodnoťte správnost trajektorií pro jednotlivé chodce napříč snímky videa.

Literatura:

- Reid P., Biometrics for Network Security. Prentice Hall Professional, 2004. ISBN 0-13-101549-4.
- JAIN, Anil K.; LI, Stan Z. Handbook of face recognition. New York: springer, 2011. ISBN 978-0-85729-932-1.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 a 2

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce: **Drahanský Martin, prof. Ing., Dipl.-Ing., Ph.D.**
Konzultant: Goldmann Tomáš, Ing., UITS FIT VUT
Vedoucí ústavu: Hanáček Petr, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 15. května 2019
Datum schválení: 3. prosince 2018

Abstrakt

Tato práce se zabývá monitorováním lidí na videozáznamu pořízeným dronem. Detekce osob v obraze je realizována pomocí natrénovaného modelu detekční sítě RetinaNet. Každé detekované osobě je extrahován příznakový vektor pomocí barevných histogramů. Jednoznačná identifikace detekovaných osob je uskutečněna porovnáváním jejich příznakových vektorů s ohledem na jejich vzdálenost ve snímku. Nakonec je vykreslena trajektorie pohybů všech detekovaných osob do výsledného panoramatického obrázku. Úspěšnost detektoru na těžkých validačních datech je 58,6 %. Chybovost je částečně vyřešena způsobem navrhnutí algoritmu pro vizualizaci trajektorií. Pro korektní vykreslení trajektorie osoby ji není nutné úspěšně detekovat v každém snímku. Zároveň statické objekty, kde je vysoká pravděpodobnost, že se nejedná o člověka, nejsou vizualizovány vůbec. Algoritmů zabývajících se detekcí lidí je velké množství, avšak přístupů zaměřených se na pohled z výšky je velmi poskromnu.

Abstract

This thesis is focused on monitoring people in a video footage captured by drone. People are detected by trained model of detector RetinaNet. A feature vector is extracted for each detected person using color histograms. Identification of people is realized by comparing their feature vectors with respect to their distance in the frame. In the end the trajectories of all people are visualized in a panorama image. Accuracy of the trained RetinaNet detector on difficult validation data is 58.6 %. Error rate is partially reduced by the way of algorithm design for trajectory visualisation. It's not necessary to successfully detect person on every frame for correct visualization of its trajectories. At the same time, static objects which are detected as person but are not moving are not consider as people and are not visualized at all. There is a lot of algorithms dealing with people detection however only a few approaches are focused on detection people from an aerial footage.

Klíčová slova

umělá inteligence, strojové učení, klasifikace, klasifikační algoritmy, hluboké učení, neuronové sítě, konvoluční neuronové sítě, rozpoznávání obrazu, počítačové vidění, detekce objektů, detekce lidí, detekce chodců, reidentifikace osob, RetinaNet, dron

Keywords

artificial intelligence, machine learning, classification, classification algorithms, deep learning, neural networks, convolutional neural networks, image recognition, computer vision, object detection, human detection, pedestrian detection, person re-identification, RetinaNet, drone

Citace

DUŠEK, Vladimír. *Monitorování chodců pomocí dronu*. Brno, 2019. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Prof. Ing., Dipl.-Ing. Martin Drahanský, Ph.D.

Monitorování chodců pomocí dronu

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Prof. Ing., Dipl.-Ing. Martina Drahanského, Ph.D. Další informace mi poskytl konzultant Ing. Tomáš Goldmann. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Vladimír Dušek
12. května 2019

Poděkování

Chtěl bych poděkovat panu Ing. Tomáši Goldmannovi za veškeré odborné konzultace a poskytnuté rady v celém průběhu tvorby této práce. Dále děkuji skupince dobrovolníků, která se zúčastnila pořizování testovacích záběrů dronem a ochotně dbala mých instrukcí.

Obsah

1 Úvod **4**

2 Detekce objektů v obraze **5**
2.1 Proces detekce objektů 5
2.2 Klasifikační algoritmy 6
2.3 Neuronové sítě 9
2.4 Učení neuronových sítí 11
2.5 Vícevrstvé neuronové sítě 15
2.6 Konvoluční neuronové sítě 19
2.7 Detekční sítě 22

3 Přehled dronů a jejich srovnání **28**
3.1 Využití dronů 28
3.2 Dostupné drony na trhu 29

4 Trénování detektoru **33**
4.1 Implementace detektoru 33
4.2 Dataset 34
4.3 Proces trénování 35

5 Algoritmus monitorování chodců **41**
5.1 Návrh 41
5.2 Implementace algoritmu 43
5.3 Extrakce a porovnávání příznakových vektorů 44
5.4 Experimenty s dílčími částmi algoritmu 44
5.5 Implementace demonstrační aplikace 47

6 Experimenty **50**
6.1 Úspěšnost a rychlost detektoru 51
6.2 Úspěšnost reidentifikace lidí 52

7 Závěr **60**

Literatura **61**

A Obsah přiloženého paměťového média **65**

Seznam obrázků

2.1 Schéma procesu detekce objektů v obraze 6
2.2 Příklady Haarových příznaků 7
2.3 Příklad aplikace Haarových příznaků 7
2.4 Výpočet plochy na základě integrálního obrazu 8
2.5 Schéma algoritmu HOG 9
2.6 Reprezentace snímku histogramy orientovaných gradientů 9
2.7 Stavba biologického neuronu 10
2.8 Matematický model neuronu (perceptron) 11
2.9 Příznaky přetrénování 14
2.10 Příklad přetrénovaného klasifikátoru 14
2.11 Architektura vícevrstvé neuronové sítě 15
2.12 Graf funkce jednotkového skoku 17
2.13 Graf lineární funkce 17
2.14 Graf sigmoidální funkce 18
2.15 Graf funkce hyperbolický tangens 18
2.16 Graf funkce ReLU 19
2.17 Příklad výpočtu konvoluce 20
2.18 Příklad max a averige poolingu 21
2.19 Ukázka napojení vrstev konvolučních sítí 22
2.20 Motivace k algoritmům detekce objektů 22
2.21 Ukázky selektivního vyhledávání detektoru R-CNN 23
2.22 Schéma fungování detektoru R-CNN 24
2.23 Chybová funkce Focal loss 25
2.24 Schéma Feature Pyramid Network 25
2.25 Zbytkový ResNet model 26
2.26 Architektura sítě ResNet 26
2.27 Porovnání přesnosti několika různých detektorů 27

3.1 Ukázky dronů podle typu použití 29
3.2 DJI Mavic 2 Pro 30
3.3 DJI Spark 31
3.4 Parrot Bebop 2 32

4.1 Ukázka ze datasetu Stanford Drone Dataset 35
4.2 Chyba a validační chyba prvního trénování 36
4.3 Validace modelu z prvního trénování na validačních snímcích 37
4.4 Validace modelu z prvního trénování na testovacích snímcích 37
4.5 Chyba a validační chyba druhého trénování 38
4.6 Validace modelu z druhého trénování na validačních snímcích 38

4.7 Validace modelu z druhého trénování na testovacích snímcích 39
4.8 Chyba a validační chyba třetího trénování 39
4.9 Validace modelu z třetího trénování na validačních snímcích 40
4.10 Validace modelu z třetího trénování na testovacích snímcích 40

5.1 Výřezy detekovaných osob neuronovou sítí 41
5.2 Příklad RGB histogramu 42
5.3 Návrhový diagram aplikace People Detector 42
5.4 První verze algoritmu identifikace chodců 45
5.5 Druhá verze algoritmu identifikace chodců 45
5.6 Třetí verze algoritmu identifikace chodců 46
5.7 Ukázka grafického rozhraní aplikace People Detector 48
5.8 Další ukázka grafického rozhraní aplikace People Detector 49

6.1 Přesnost natrénovaného modelu 51
6.2 Ukázka z vyhodnocování přesnosti modelu 51
6.3 Ukázky detekce lidí na testovacích obrázcích 52
6.4 Otestování algoritmu na scéně Coupa 53
6.5 Výsledná mapa s trajektoriemi osob ze scény Coupa 53
6.6 Otestování algoritmu na scéně Little 54
6.7 Výsledná mapa s trajektoriemi osob ze scény Little 55
6.8 Otestování algoritmu na scéně Death Circle 56
6.9 Výsledná mapa s trajektoriemi osob ze scény Death Circle 57
6.10 Otestování algoritmu na prvním záznamu z dronu 58
6.11 Výsledná mapa s trajektoriemi osob z první scény 58
6.12 Otestování algoritmu na druhém záznamu z dronu 59
6.13 Výsledná mapa s trajektoriemi osob z druhé scény 59

Kapitola 1

Úvod

Technologický vývoj v oblasti hardware zaznamenal v posledních desítkách let obrovský pokrok. Počítače dosahují několikanásobného výpočetního výkonu a disponují mnohokrát větší pamětí. Zároveň díky internetu a moderním technologiím dnešní doby je zaznamenáváno daleko více dat. Tyto skutečnosti vytváří ideální prostředí pro rozvoj oblasti strojového učení, speciálně pak oboru neuronových sítí. Jejich teoretický základ předběhl technologickou vyspělost tehdejší doby. Nyní neuronové sítě konečně dokáží uplatnit svůj potenciál a odstartovaly revoluci například v odvětví zpracování obrazu a zpracování lidské řeči.

Cílem práce je prostudovat možnosti detekce osob v obraze z výšky, z videozáznamu pořízeným dronem. Každý detekovaný člověk bude v průběhu zpracování videa identifikován. Jednotliví lidé budou od sebe rozlišeni, budou monitorováni v průběhu celého záznamu a nakonec trajektorie jejich pohybů budou vizualizovány v panoramatickém snímku. Bude implementována aplikace pro otestování a demonstraci funkcionality.

V kapitole 2 je čtenáři představena oblast detekce objektů v obraze. Ve stručnosti je nastíněno, jak celý proces probíhá a jsou vysvětleny některé starší klasifikační algoritmy, které ve své době byly významné až pokrokové. Dále kapitola pojednává o umělých neuronových sítích a jejich využití pro zpracování obrazu. Je vysvětlena analogie s biologickými neuronovými sítěmi, jak umělé neuronové sítě fungují, jak probíhá proces učení, jak jsou jednotlivé neurony na sebe napojeny a jak fungují konvoluční neuronové sítě pro klasifikaci dat. Na konci kapitoly je vysvětleno, jak se dají využít pro detekci objektů a je představeno několik konkrétních detektorů.

Problematika dronů je nastíněna v kapitole 3. Přes historii vývoje dronů, po jejich současnou podobu a využití dnes i v blízké budoucnosti, je srovnáno několik konkrétních modelů s ohledem na jejich možné využití pro monitorování chodců.

Kapitola 4 představuje zvolenou implementaci detektoru. Dále vybraný dataset a veškeré manipulace s ním. Je popsán proces trénování vlastního modelu. V průběhu jsou prezentovány výsledky a na jejich základě další modifikace datasetu.

V kapitole 5 je představen návrh algoritmu pro monitorování chodců. Více je rozvedena problematika jejich identifikace a reidentifikace. Nakonec je popsána vytvořená demonstrační aplikace People Detector.

Výsledky a provedení experimentů jsou shrnuty v kapitole 6. Nejprve je vyhodnocena úspěšnost detektoru a poté je na několika ukázkách představena funkčnost algoritmu reidentifikace lidí a zakreslení jejich trajektorií do snímku. Je vyhodnoceno v jakých situacích mechanismus funguje a kdy naopak ne.

Závěrečná 7. kapitola pak shrnuje provedenou práci, zamýšlí se nad výsledky, možným budoucím vývojem a dalšími rozšířeními.

Kapitola 2

Detekce objektů v obraze

Detekce objektů je počítáčová technologie z oblasti počítačového vidění (*computer vision*) a zpracování obrazu (*image processing*). Zabývá se rozpoznáním objektů v digitálních obrázcích či videích a jejich následným zařazením (klasifikací) do předem definovaných kategorií (tříd). Může se jednat například o detekci lidí, automobilů, zvířat či v podstatě čehokoliv jiného, záleží pouze na množině trénovacích dat. O trénování více v sekci 2.4.

V poslední době zažívá sféra detekce objektů v obraze velký rozmach a vzbuzuje značný zájem v oblasti umělé inteligence. S rozvojem architektury konvolučních neuronových sítí, za kterými stojí velké množství trénovacích dat a pokročilé výpočetní technologie, se schopnost strojů rozpoznávat objekty ve scénách výrazně zlepšila. Počítače nyní, za určitých specifických nastavení, dokáží v některých konkrétních úlohách dokonce překonat člověka, například v rozpoznávání lidských tváří [22].

Již dnes lze najít velké množství konkrétních využití detekce objektů. Například optické rozpoznávání vytištěných či ručně psaných znaků lze využít pro automatické rozeznávání státních poznávacích značek aut na záznamech z kamer či digitalizaci různých typů tištěných dokumentů. V míčových sportech lze využít detekci objektů pro sledování míče. Na sociálních sítích je zase implementována detekce lidských obličejů pro rozpoznávání tváří uživatelů na fotografiích. Další příklad využití detekce objektů jsou samořídící auta. Ty jsou sice otázkou až blízké budoucnosti, je ale zřejmé, že musí umět velmi spolehlivě rozeznávat veškeré objekty kolem sebe (jiné auto, člověk, semafor, ...) a podle toho reagovat.

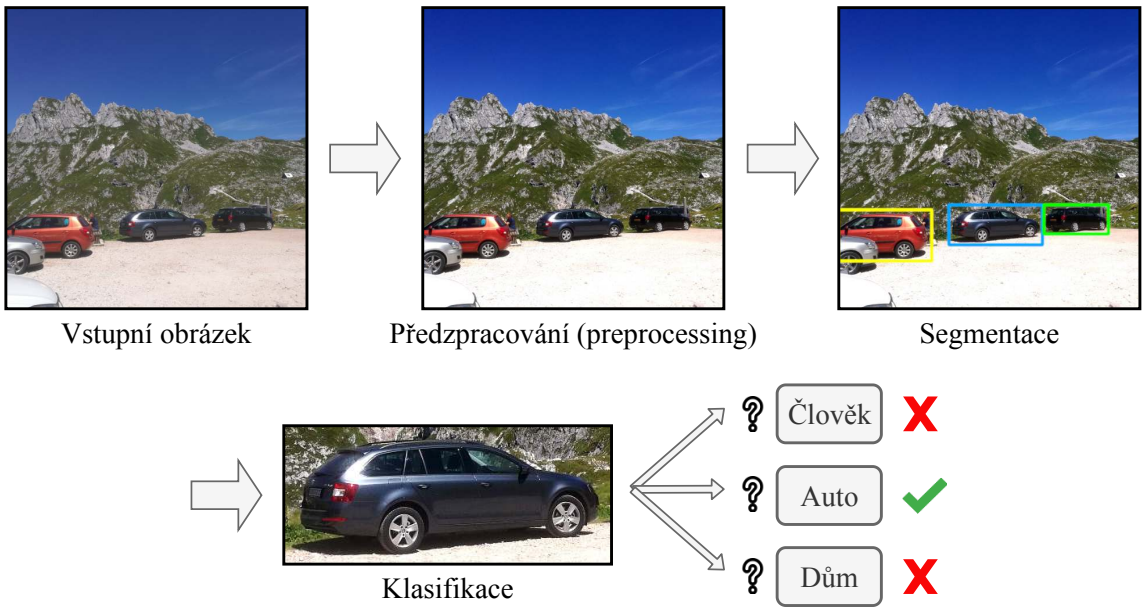
2.1 Proces detekce objektů

Kompletní proces detekce objektů je běžně rozdělen do několika fází. Za úplně první krok může být považováno pořízení vstupních digitálních obrazových dat fotoaparátem nebo videokamerou. Získaná data jsou následně předzpracována jednoduchými úpravami (*preprocessing*). Například se jedná o změnu rozlišení, zvýraznění hran objektů, úpravy jasu, kontrastu, normalizace barev apod. Druh předzpracování záleží na konkrétní metodě detekce, některé jej nevyužívají vůbec.

Dalším krokem je segmentace obrazu, kdy jsou vybrána místa, kde by se potenciálně mohl vyskytovat nějaký hledaný objekt. Nejjednodušší metody pracují s oknem, které se s určitým krokem posouvá po obraze a výřezy jsou předávány klasifikátoru. Pro detekci objektů s různým poměrem stran a orientací je nutné použít okno opakovaně s různými parametry. Problém těchto metod je velký počet výřezů, což má za následek značné výpočetní

nároky. V dnešní době valná většina detektorů funguje na základě neuronových sítí. Ty jsou schopny selektivně vybírat vhodné hraniční oblasti, a tím výrazně snížit počet výřezů.

Posledním krokem je klasifikace objektu. Z vyříznuté oblasti je extrahován příznakový vektor, který je porovnán se vzorovými příznaky jednotlivých tříd. Příznaky slouží jako popis samostatného objektu. Mohou jimi být například barva, struktura nebo tvar. U neuronových sítí je extrakce příznaků prováděna pomocí konvolučních vrstev. Porovnávání příznaků provádí klasifikátor, ten rozhodne, zda se jedná o objekt z předem známých kategorií (tříd).



Obrázek 2.1: Schéma procesu detekce objektů v obraze.

2.2 Klasifikační algoritmy

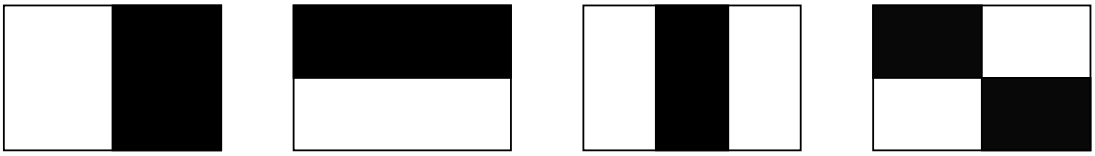
Algoritmy pro detekci objektů v obraze je možné rozdělit podle toho, jakým způsobem jsou jim nastaveny příznaky (*features*). Algoritmy se snaží tyto příznaky v datech hledat a na jejich základě provádět klasifikaci. První jsou klasifikátory, které mají příznaky předprogramované (*hand coded*). Mají zaměření na konkrétní typ dat a přeučit je na jiné vyžaduje přeprogramování příznaků. Tímto typem klasifikátorů se zabývá tato sekce. Druhým typem jsou klasifikátory, jejichž příznaky jsou dynamicky upravovány trénováním. Po naprogramování nemají žádné zaměření, a ještě nejsou schopny klasifikovat žádná data. Tuto schopnost získají až po procesu natrénování na trénovacích datech. Mezi tento typ klasifikátorů patří neuronové sítě, viz sekce 2.3.

Algoritmus Viola–Jones

Ačkoliv se lidé oblasti detekce objektů v obraze pomocí počítačů věnují již od 60. let 20. století, dlouho se nedařilo přijít s výsledky, které by bylo možné k něčemu reálně využít. Zlom nastal až v novém tisíciletí, kdy v roce 2001 přišli Paul Viola a Michael Jones s algoritmem, který pojmenovali po sobě samých [38]. Algoritmus byl primárně navržen pro detekci obličejů, lze jej však upravit pro detekci různých objektů s výraznými hranami.

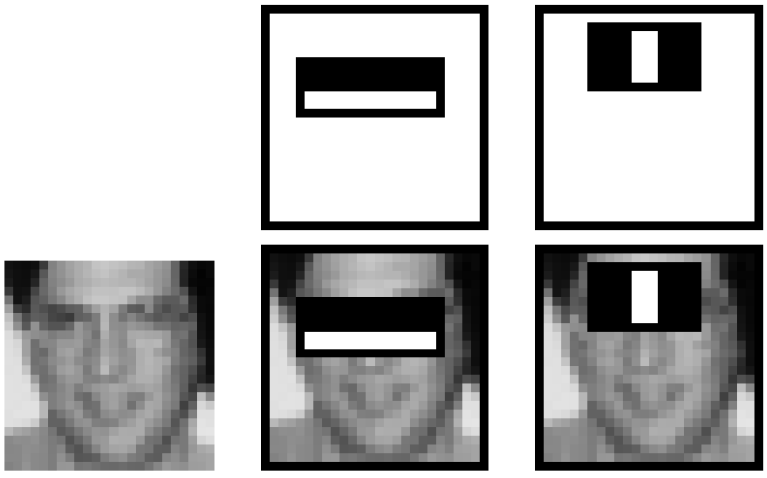
Demonstrační aplikace, kterou Paul Viola a Michael Jones představili, byla schopná v reálném čase detekovat obličeje na videu z webkamery. Byla zde jistá omezení: Celá plocha obličeje musela směřovat přesně k fotoaparátu a obličej nesměl být nijak nakloněn. I přesto se jednalo v té době o revoluční objev, který demonstroval potenciál počítačového vidění. Algoritmus byl brzy implementován do knihovny OpenCV a stal se synonymem pro detekci obličejů.

Algoritmus využívá toho, že lidské tváře sdílí podobné rysy. Například oblast očí je obvykle tmavší než líce, nosní přepážka je zase světlejší než oblast očí. Využívá také celkové kompozice obličeje, pozici a velikosti očí, pusy a nosu. Snímek je na začátku zpracování převeden do stupně šedi (*greyscale*), kde tyto vlastnosti lépe vyniknou. Jejich míra shody je poté zjišťována pomocí Haarových příznaků.



Obrázek 2.2: Příklady Haarových příznaků. První dva mohou sloužit pro detekci hran, třetí pro detekci vertikální linie a čtvrtý pro detekci šikmé čáry.

Algoritmus pomocí aplikace Haarových příznaků detekuje ve vstupním obrázku polohu očí, nosu a pusy. Následně spočítá jejich vzájemnou polohu a vzdálenost. Poté tyto metriky porovná se vzorovými metrikami – jak jsou v obličeji od sebe průměrně vzdáleny oči, nos a pusa. Z vysvětlení tohoto přístupu už jsou jasná omezení fungování algoritmu. Obličej musí být natočen přesně ke kameře, aby bylo možné detekovat důležité rysy. V jakkoliv nakloněném obličeji jsou zase tyto rysy odlišné. Stejně tak algoritmus nemůže fungovat pokud jsou tyto důležité rysy něčím překryty.

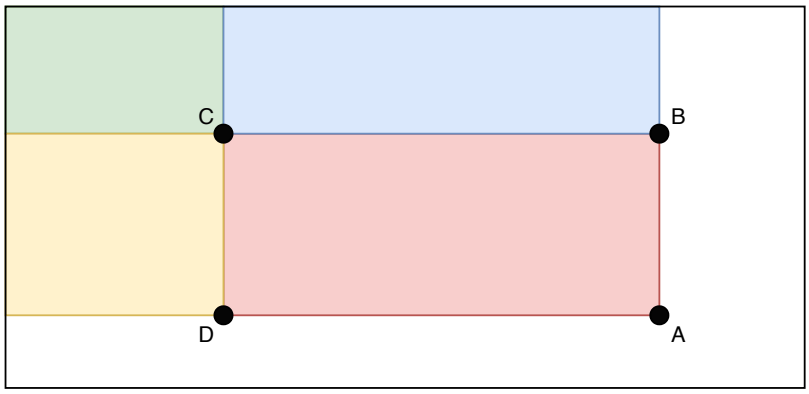


Obrázek 2.3: Příklad aplikace Haarových příznaků. První měří rozdíl intenzity mezi oblastí očí a líci, druhý porovnává intenzitu v oblasti očí a nosní přepážky. Obrázek je převzatý [38].

Každý příznak je spjat se specifickou oblastí ve vyříznutém okně (částí snímku). Hodnota příznaku se spočítá jako rozdíl součtu intenzity pixelů pod černými a bílými částmi, viz následující rovnice.

$$\text{Value} = \sum(\text{pixels in black area}) - \sum(\text{pixels in white area}) \tag{2.1}$$

Je zřejmé, že i pro relativně malý obrázek bude spočítáno velké množství Haarových příznaků. Vzhledem k tomu, že algoritmus vyžaduje iterovat přes všechny příznaky, musí být spočítány efektivně. Z tohoto důvodu Viola a Jones navrhli postup, kdy je původní obraz převeden do integrálního obrazu (*integral image*). Jedná se o způsob reprezentace obrazu tak, že každý bod představuje součet hodnot předchozích pixelů doleva a nahoru, tedy pravý spodní bod obsahuje součet všech pixelů původního obrázku. Reprezentovat obraz tímto způsobem je velmi výhodné, jelikož při počítání intenzity v segmentu obrazu lze pracovat pouze s jeho čtyřmi okrajovými body a není nutné procházet všechny pixely v dané oblasti.



Obrázek 2.4: Příklad výpočtu plochy na základě integrálního obrazu.
Red area $= A - B - D + C$.

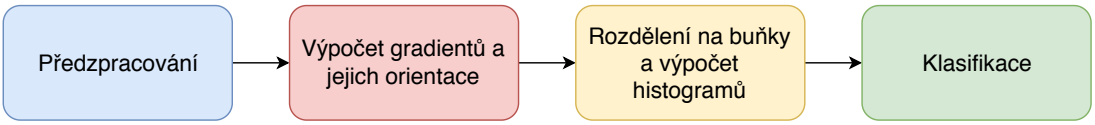
Histogramy orientovaných gradientů (HOG)

Koncept rozpoznávání obrazu pomocí techniky histogramů orientovaných gradientů byl znám již od druhé poloviny 80. let 20. století. Do širšího povědomí se však algoritmus dostal až v roce 2005, kdy ho Navneet Dalal a Bill Triggs použili pro detekci chodců ve statických obrázcích ve své práci *Histograms of Oriented Gradients for Human Detection* [2]. Dosahovali výrazně lepších výsledků než do té doby dostupná řešení. Později detektor rozšířili i pro použití ve videu a detekci dalších objektů (aut, zvířat). V současnosti je HOG detektor implementován v běžně dostupných knihovnách pro počítačové vidění, např. OpenCV.

Základní myšlenka algoritmu spočívá v porovnávání, jak je určitá část obrazu tmavá v porovnání s částmi, které ji obklopují. V prvním kroku je možné využít předzpracování pomocí normalizace kontrastu a barev. Dalal a Triggs ale ve své práci demonstrovali, že předzpracování přináší jen velmi malé zpřesnění výsledků, proto je u mnoha implementací úplně vynecháváno.

Po případném předzpracováním je pro každý pixel ve snímku spočítáno, který pixel kolem něho je nejtmavší. Následně je pixel nahrazen orientovaným gradientem (šipkou), který míří směrem k nejtmavšímu sousedovi. Gradienty nám tak ukazují tok světla napříč celým obrázkem. Z původní reprezentace snímku (matice pixelů) dostáváme matici orientovaných

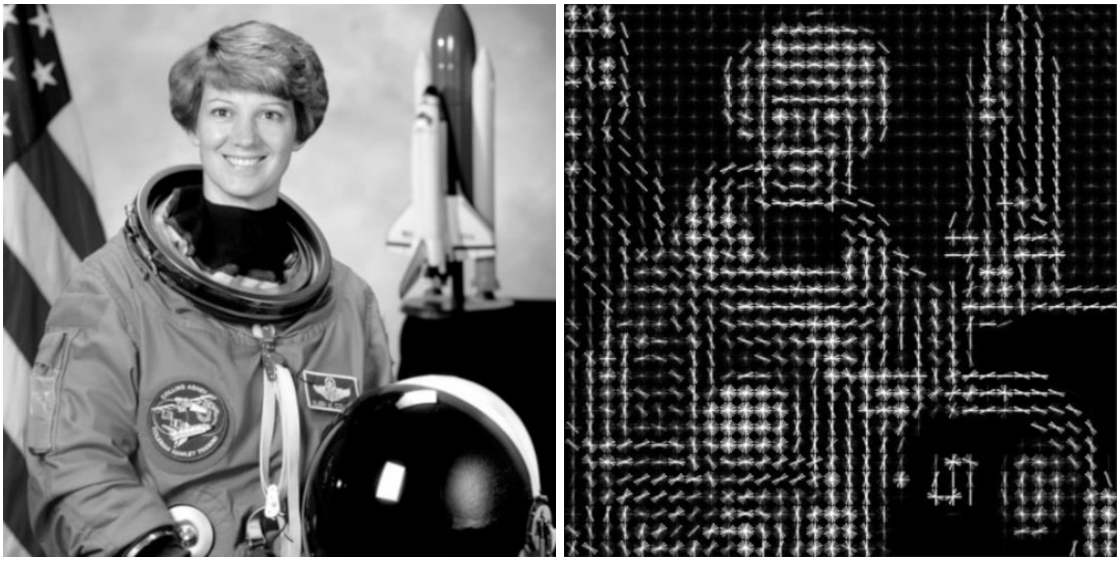
gradientů. U barevných fotografií je gradient vypočten zvlášť pro každý kanál a pro daný pixel je zvolen ten s nejvyšší hodnotou.



Obrázek 2.5: Schéma celého algoritmu, který představili Dalal a Triggs.

Ve druhém kroku je obraz rozdělen na buňky (malé čtverce) 16×16 pixelů. Pro každou buňku je spočítán histogram orientovaných gradientů, a tím je zjištěn jejich převažující směr – dominantní tok světla v buňce. Celá buňka je následně nahrazena převažujícím gradientem.

Tímto způsobem je původní obrázek převeden do poměrně jednoduché reprezentace, která však neztratí informaci o toku světla ve snímku. V tomto formátu probíhá klasifikace, tedy porovnávání zachycených objektů se vzory a jejich následné zařazení do rozpoznávaných tříd. Algoritmus využívá lineární klasifikátor SVM (Support Vector Machine). Reprezentace histogramy orientovaných gradientů také například dokáže velmi dobře zachytit základní rysy obličeje.



Obrázek 2.6: Reprezentace snímku histogramy orientovaných gradientů. Obrázek je převzatý [32].

2.3 Neuronové sítě

Neuronové sítě jsou jedním z výpočetních modelů používaných v umělé inteligenci. Jeden z průkopníků neuronových sítí, Dr. Robert Hecht-Nielsen, definuje neuronovou síť jako:

> „Počítačový systém tvořený řadou jednoduchých, vysoce propojených procesních prvků, které zpracovávají informace svou dynamickou reakcí na externí vstupy.“ [5]

Díky značnému vývoji výpočetního výkonu počítačů a zároveň dostupnosti velkého množství dat, začaly neuronové sítě dosahovat v oblasti umělé inteligence výsledků, které byly před pár lety ještě nepředstavitelné. Momentálně nabízejí nejlepší řešení v řadě oblastí umělé inteligence, zejména pak v oboru zpracování obrazu a rozpoznávání lidské řeči.

Nápad algoritmu založeném na biologickém neurononu vznikl už ve 40. letech 20. století. V 70. a 80. letech byly snahy nápad zrealizovat, ale výpočetní technologie byly absolutně nedostačující pro jakékoliv trénování neuronových sítí [40]. Další výzkum tedy musel počkat na vyšší výpočetní výkon. Během toho byla vymyšlena komplexnější architektura neuronových sítí včetně zpětné propagace, sekce 2.5, a konvolučních vrstev, sekce 2.6.

První opravdové úspěchy neuronových sítí přišly v rozpoznávání ručně psaných číslic v 90. letech, o to se postaral Yann LeCun [18].

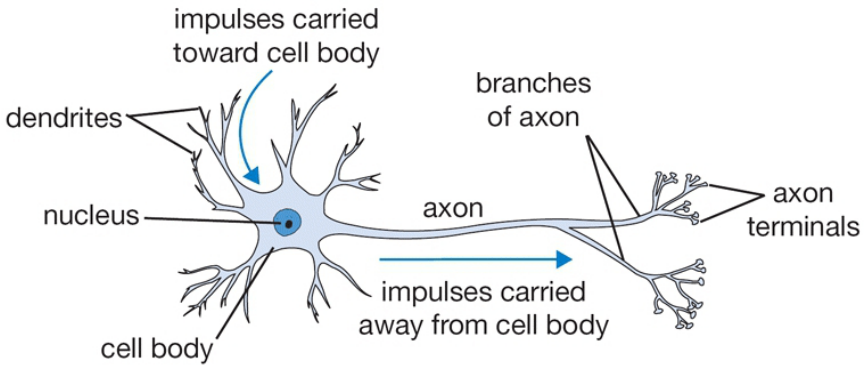
V roce 2006 přišel Geoff Hilton s algoritmem pro výrazně rychlejší učení hlubokých neuronových sítí [11]. Později ukázal, že pro rozpoznávání lidského hlasu dokáže vytrénovat neuronovou síť v rámci týdnů.

V roce 2012 v soutěži ImageNet Large Scale Visual Recognition (ILSVR), což je soutěž, kde vědci srovnávají výkonnost svých algoritmů, Alex Krizhevsky se svojí konvoluční neuronovou sítí dosáhl suverénně nejlepších výsledků [15].

Na neuronovou síť, přesněji umělou neuronovou síť, se můžeme dívat také jako na výpočetní model inspirovaný biologickou neuronovou sítí, konkrétně zpracováním informací v lidském mozku.

Biologická inspirace

Základní výpočetní jednotkou mozku je neuron. Jedná se o živou buňku, která je zaměřená na sběr, zpracování a přenos informací. V lidském nervovém systému se nachází kolem 86 miliard neuronů [5]. Každý z nich má v průměru 10 tisíc spojů s jinými neurony. Neuron se skládá z těla (*cell body*), do kterého přicházejí informace ze vstupních větví – dendritů (*dendrites*). Výstupní signál, který závisí na vstupech, je dále propagován axonem. Ten je na svém konci mnohačetně rozvětven. Propojení neuronů je uskutečněno pomocí speciálních výběžků – synapsí. Ty jsou napojeny na dendrity nebo přímo na těla jiných neuronů.



Obrázek 2.7: Stavba biologického neuronu. Vstupní informace jsou ze synapsí přijímány dendrity, které impulsy propagují dále do jádra neuronu. Případná reakce neuronu na vstupní podněty je pak přenášena axonem do synapsí a následně buď na dendrit dalšího neuronu, nebo na neuronem ovládanou svalovou buňku. Obrázek je převzatý [5].

Matematický model neuronu

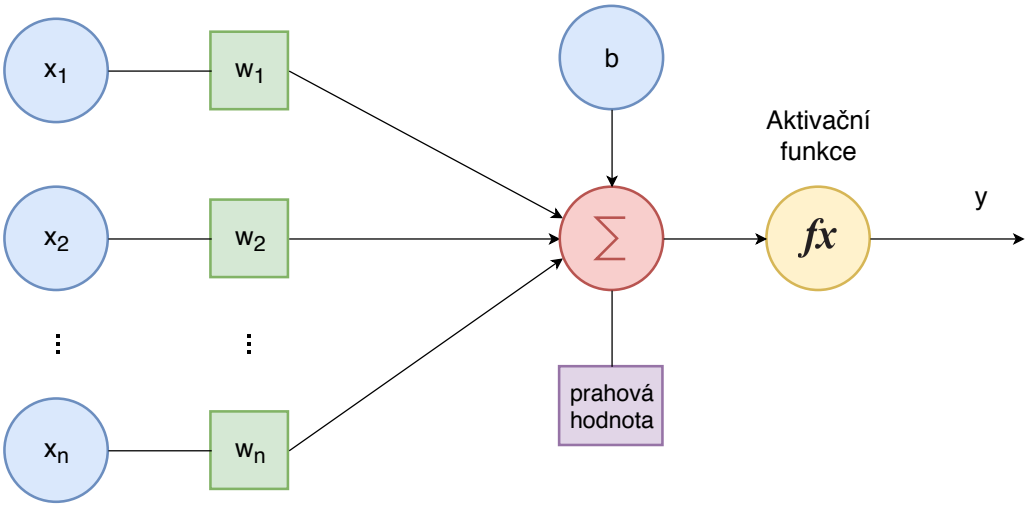
Základní jednotka matematického modelu neuronové sítě je získána přeformulováním zjednodušené funkce biologického neuronu do matematického jazyka. Tato základní jednotka, matematický neuron, je někdy nazývána jako perceptron. Ilustrace matematického modelu neuronu je zachycena na obrázku 2.8.

Vstup neuronu je získáván z výstupu jiných neuronů, či externího zdroje. Každému vstupu je přiřazen váhový koeficient, který určuje důležitost vstupu v porovnání s ostatními [8]. Výstup je vypočítán pomocí tzv. váženého součtu. Jedná se o součet veškerých hodnot vstupů v závislosti na jejich váhovém koeficientu. S výsledkem pak pracuje aktivační funkce (o nich více v sekci 2.5), která určí výstup neuronu, viz následující rovnice.

$$y = f(b + \sum_{i=1}^{n} w_i x_i) \tag{2.2}$$

Aktivační funkce je značena f, vstupy neuronu jako x, váhy vstupů w (*weigh*) a b je značeno zkreslení (*bias*). Výstupní hodnota neuronu je označena y.

Myšlenka váhových koeficientů je učenlivost, kontrola míry závislosti jednoho neuronu na jiném. Koeficient může nabývat jak pozitivních, tak negativních hodnot. Jestliže je výsledný součet vyšší než prahová hodnota, je neuron probuzen a indikuje signál na svém výstupu ve formě aktivační funkce (též přenosové funkce).



Obrázek 2.8: Matematický model neuronu (perceptron). Vstupy jsou značeny x, jejich váhy w (*weigh*), zkreslení b (*bias*) a výstup y. Vše s příslušnými indexy.

2.4 Učení neuronových sítí

Cílem učení (trénování) neuronových sítí je nastavit je tak, aby poskytovaly co nejpřesnější výsledky. V biologických sítích jsou zkušenosti uloženy v dendritech. V umělých neuronových sítích jsou zkušenosti uloženy v jejich matematickém ekvivalentu – váhách. V průběhu trénování se tak váhové koeficienty postupně mění a to takovým způsobem, aby nakonec poskytovaly správné hodnoty výstupního signálu na dané vstupní signály. Množina dat, na kterých se síť učí, se nazývá dataset. Po procesu naučení neuronové sítě lze na síť pohlížet

jako na černou skříňku (*black box*), která je vhodná k nasazení ve zvolených aplikačních rovinách [36].

Strategie učení

Učení s učitelem (*supervised learning*) je strategie, kdy je využíváno zpětné vazby. Neuronové síti je předložen vzor a na základě aktuálních hodnot váhových koeficientů je vypočítán výsledek. Ten je porovnán s očekávaným výsledkem a následně je spočítána chyba – jak moc se aktuálně vypočítaný výsledek neuronovou sítí liší od toho očekávaného. Je-li chyba vyšší než stanovená minimální hranice, je spočítána korekce a jsou upraveny hodnoty vah tak, aby hodnota chyby byla snížena. Toto je opakováno až do dosažení požadované minimální hranice chyby. Poté je síť považována za adaptovanou (naučenou) [8]. Při rozpoznávání objektů v obraze je typické použití strategie učení s učitelem, kdy se síť učí na předem označených (anotovaných) datech.

Učení bez učitele (*unsupervised learning*) na rozdíl od učení s učitelem s výstupem vůbec nepracuje. Neuronové síti jsou postupně předkládány vstupní data bez požadovaných výstupů. Síť je donucena si sama hledat relevantní příznaky, pomocí kterých dokáže data sama roztřídit do skupin. To vede ke shlukování (*clustering*) vstupních dat. Síť se tak naučí reagovat na typického zástupce shluku. Váhy sítě se nastavují tak, aby výstup byl konzistentní, tedy aby síť poskytovala stejnou odezvu na budící signál při stejných nebo podobných vektorech vstupu [8].

Chybové funkce

Pro proces učení neuronové sítě je nutný algoritmus, který provádí změnu vah vstupů a výši zkreslení. Obecně až do takové fáze, kdy při vstupu dat z datasetu výstup odpovídá očekávanému výsledku. Ke zjištění toho, do jaké míry se výstup sítě liší od očekávaného výsledku, se využívá chybových funkcí. Z jejich výstupu je určeno, jakým směrem a zhruba o kolik je nutné změnit váhové koeficienty jednotlivých vstupů. Správný průběh učení neuronové sítě lze pozorovat z hodnot chybové funkce. Ty by se měly každou iteraci snižovat a síť by tak měla lépe aproximovat funkci specifickou pro daný úkol.

Mean squared error (MSE) je základní funkce pro výpočet chyby. Je definována rovnicí 2.3. $\hat{Y}$ je vektor predikcí, Y je vektor očekávaných výsledků, n velikost těchto vektorů [19]. Díky umocnění se odstraní záporné hodnoty a dojde ke zvýraznění větších chyb.

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 \tag{2.3}$$

Cross entropy loss (CE) je chybová funkce definována rovnicí 2.4. x značí vstupní vektor o velikosti N, $y(x_n, w)$ je výstup z neuronové sítě za použití vah w a t_n je očekávaný výstup. Funkce se využívá v případech, kdy výstup může nabývat hodnot z několika tříd, které nejsou vzájemně výlučné. V praxi se může jednat například o rozpoznání objektu v obraze [3].

$$\mathrm{CE} = \sum_{n=1}^{N}\sum_{k=1}^{K} t_{n,k}\ln(y(x_n, w)) \tag{2.4}$$

Inicializace vah

Za inicializaci vah se považuje proces před samotným učením neuronové sítě, kdy se váhám vstupů přiřadí počáteční hodnoty. Správné počáteční hodnoty mohou proces trénování sítě velmi urychlit [37].

Úplně nejjednodušší způsob je vygenerování čistě náhodných počátečních hodnot. To z pochopitelných důvodů nemusí být ideální, proto se standardně využívají chytřejší techniky. Způsob vhodnosti generování se liší od typů aktivačních funkcí daného neuronu. Typicky je vygenerována hodnota na základě Gaussova rozdělení a následně je vynásobena nějakým výrazem.

Pro aktivační funkci hyperbolický tangens

1. Je vygenerována náhodná hodnota dle Gaussova rozdělení se středem 0 a odchylkou 1.
2. Hodnota je vynásobena výrazem $\sqrt{2/n_i}$, kde n_i je počet vstupních jednotek vrstvy, ve které se neuron nachází.

Někdy může dojít k problémům známým jako *vanishing gradient* nebo *exploding gradient* [27]. Omezení těchto problémů řeší například Xavierova inicializace, která zohledňuje i počet výstupů vrstvy [37].

Xavierova inicializace pro aktivační funkci hyperbolický tangens

1. Je vygenerována náhodná hodnota dle Gaussova rozdělení se středem 0 a odchylkou 1.
2. Hodnota je vynásobena výrazem $\sqrt{2/(n_i + n_o)}$, kde n_i je počet vstupních jednotek a n_o počet výstupních jednotek vrstvy, ve které se neuron nachází.

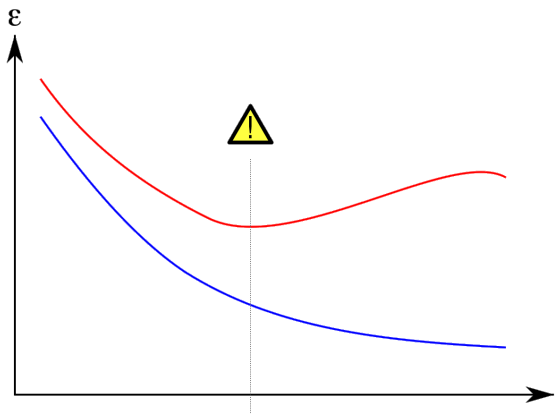
Pro inicializaci vah je možné rovněž použít hodnoty z již naučené neuronové sítě, která byla vytrénována k podobným účelům.

Přetrénování a podtrénování

Při učení neuronové sítě je běžné, že se model po určitém počtu iterací dostane do stavu, kdy podává velmi přesné výsledky na trénovacích datech (chybová funkce udává velmi nízké hodnoty), ale při použití nových vstupních dat je přesný méně, než tomu bylo u nižšího počtu iterací trénování. Tento stav se označuje jako **přetrénování** (*overfitting, overtraining*) [42].

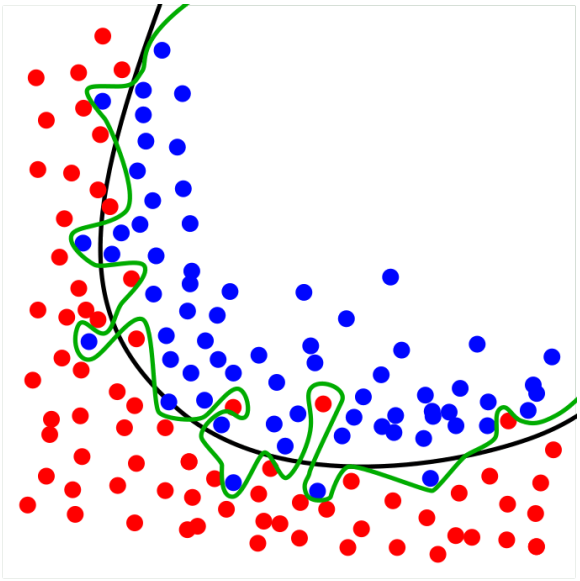
Pro ideální natrénování modelu a zabránění přetrénování je dataset běžně rozdělen na dvě části, trénovací a validační. Trénovací část je typicky značně větší a obsahuje trénovací data – data na kterých se model učí. Druhá část jsou tzv. validační data. Ta neslouží pro trénování, ale pouze pro hlídání přetrénování. Na jejich základě se s váhami modelu nehýbe. Po každé epoše[1] se spočítá chyba z validačních dat. Pokud chyba na trénovacích datech klesá, ale chyba na validačních datech roste, znamená to, že se model dostává do stavu přetrénování.

[1] Epocha – iterace trénování, během jedné epochy je předložen síti celý dataset.



Obrázek 2.9: Graf trénovací a validační chyby v závislosti na počtu odtrénovaných iterací. Trénovací chyba je zakreslena modře, validační chyba červeně. V momentě, kdy validační chyba začala růst, ale trénovací chyba pokračovala v klesání, dochází k přetrénování. Nejideálněji natrénovaný model je po počtu iterací, kdy je validační chyba na svém globálním minumu. Obrázek je převzatý [42].

K přetrénování dochází jelikož množství trénovacích dat je omezené. Model se tak po určité době začne zaměřovat na detaily, které jsou specifické pouze pro danou sadu dat a detektor tak ztrácí na robustnosti, obecnosti (*generalization*). V extrémním případě si pak model zapamatuje všechna trénovací data a je tak schopen podávat 100 % správné výsledky.



Obrázek 2.10: Příklad přetrénovaného klasifikátoru na dvou typech dat (červené a modré). Zelená linie reprezentuje přetrénovaný model, černá správně natrénovaný, který dobře aproximuje původní funkci. Zatímco zelená linie perfektně klasifikuje trénovací data, je pro ně příliš uzpůsobená. Je tak pravděpodobné, že při nových datech by podávala vyšší chybovost než klasifikátor černý. Obrázek je převzatý [42].

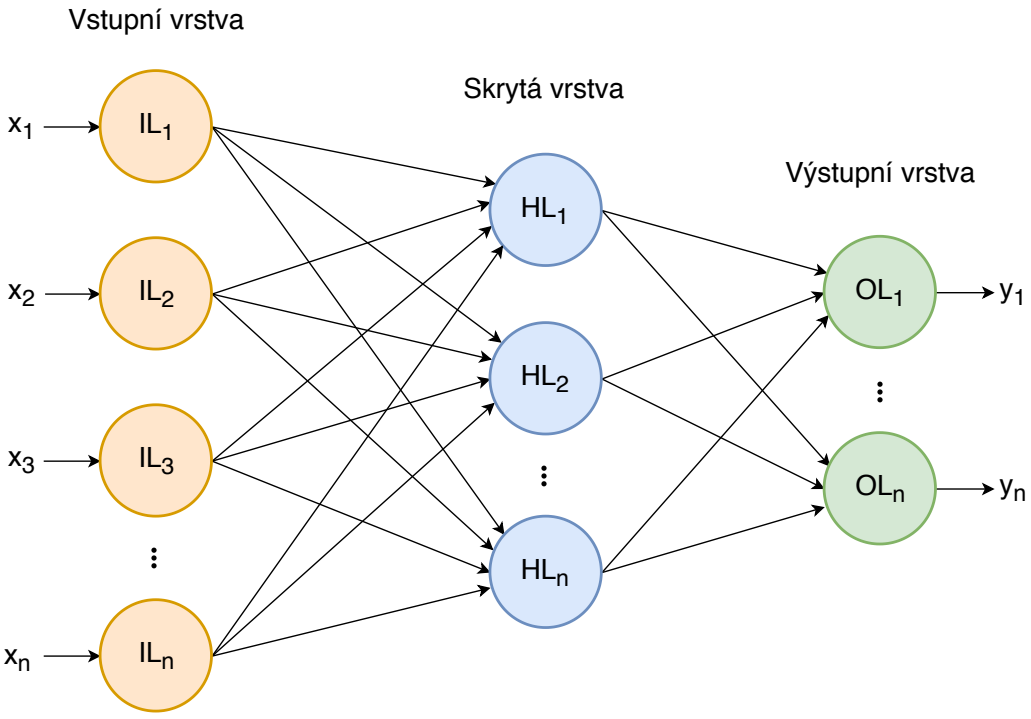
Podtrénování (*underfitting, undertraining*) je jev opačný. Přetrénovaný model dokáže perfektně klasifikovat data, na kterých byl trénován, ale podává špatné výsledky na datech,

které nikdy neviděl [42]. Podtrénovaný model pak není schopen ani uspokojivě klasifikovat data, na kterých byl trénován a na datech, která nikdy před tím neviděl, pohoří pravděpodobně úplně. Jednoduše chybová funkce je stále vysoká a přesnost modelu nízká.

K podtrénování dochází, protože model nemůže adekvátně zachytit základní strukturu dat, případně na to neměl dostatek času. Model zkrátka nemusí být vhodný pro naučení se dané struktury. Jedním z řešení může být úprava modelu. Zvýšení počtu vrstev, zvýšení počtu neuronů, či změnění typů vrstev a jejich přeskládání. Problém ale může být i na straně datasetu, objekt nemusí mít dostatečně specifické příznaky, na základě kterých by ho bylo možné identifikovat.

2.5 Vícevrstvé neuronové sítě

Doposud nebylo vysvětleno, jak taková neuronová síť vlastně vypadá. Použití pouze jednoho neuronu není příliš efektivní a dá se využít pouze na velmi jednoduché úlohy (např. klasifikace do dvou tříd) [36]. Ve skutečnosti se setkáváme s vícevrstvými neuronovými sítěmi (MultiLayer Perceptron – MLP). Ty jsou tvořeny opakováním základního modelu neuronu, které jsou na sebe napojeny ve vrstvách. Výstup jednoho neuronu je pak vstupem jednoho nebo více neuronů další vrstvy. Způsob propojení neuronů pak určuje architekturu (topologii) sítě.



Obrázek 2.11: Architektura vícevrstvé neuronové sítě. Síť se skládá z vrstvy vstupní (*input layer*), skryté (*hidden layer*) a výstupní (*output layer*). Ve vrstvě vstupní neprobíhají žádné výpočty, pouze se propagují externí informace do další vrstvy. Ve vrstvě skryté se informace zpracovávají a poté jsou propagovány do vrstvy výstupní. Ta poskytuje výstup neuronové sítě.

Počet vstupních neuronů je dán počtem vstupů matematického modelu. Počet neuronů ve skryté vrstvě je volen s ohledem na složitost úlohy. Čím složitější úloha, tím více neuronů je potřeba pro naučení se potřebných příznaků a pochopení struktury vzoru [36]. Počet výstupních neuronů obvykle odpovídá počtu klasifikačních tříd. Vícevrstvá neuronová síť je standardně učena strategií učení s učitelem. Neunorová síť může mít více skrytých vrstev, poté o ní hovoříme jako o hluboké neuronové síti (Deep Neural Network – DNN). Jednotlivé skryté vrstvy pak mají různé formy specializace. O takových sítích mluvíme jako o konvolučních neuronových sítí a jsou podrobně popsány v sekci 2.6. Data, se kterými skryté vrstvy pracují, jsou směrem od vstupu více abstraktní a ke konci sítě lze pomocí příznaků definovat i velmi abstraktní datové struktury. Proces učení se s robustností sítě stává náročnější.

Zpětné šíření chyby

Zpětné šíření chyby (*backpropagation*) je adaptační algoritmus, pomocí kterého lze vypočítat podíl jednotlivých neuronů na celkové chybě sítě a upravit váhy jednotlivých neuronů tak, aby byla minimalizována celková chyba. Jedná se o nejrozšířenější adaptační algoritmus vícevrtsvých neuronových sítí, je používán přibližně v 80 % všech aplikací neuronových sítí [39].

Samotný algoritmus se skládá ze tří etap. Nejprve je vstupní signál šířen sítí dopředu (*feedforward*). Ze vstupní vrstvy do vrstev skrytých, kde je spočítán váhový součet všech vstupů a dle aktivační funkce je výstup neuronů směrován do dalších vrstev. Tímto způsobem je signál propagován až do vrstvy výstupní, kde je spočítána celková chyba dle chybové funkce. Dále je chyba šířena neuronovou sítí zpětně od vrstev vyšších k vrstvám nižším. Každému neuronu je spočítána korekce vah. Ty jsou nakonec aktualizovány a celý proces se opakuje s dalšími vstupy.

S konceptem zpětného šíření chyby souvisí problémy *vanishing gradient* a *exploiding gradient*.

Aktivační funkce

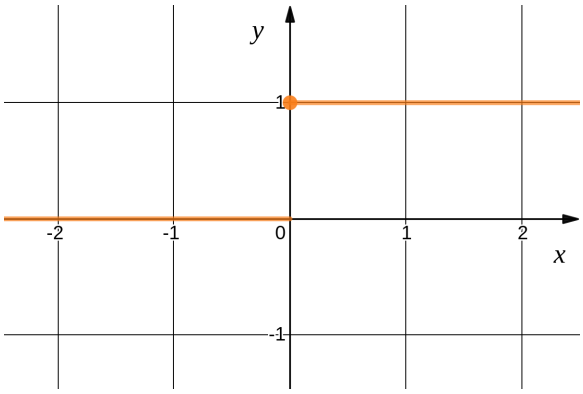
Aktivační funkce (také přenosová) slouží pro výpočet výstupní hodnoty neuronu v závislosti na jeho vektoru vstupních hodnot. Je-li neuron probuzen, aktivační funkce moduluje výstupní signál, ten je směrován do další úrovně neuronové sítě. Bez aktivačních funkcí by výstup neuronu mohla být jakákoliv hodnota od mínus nekonečna až do plus nekonečna. Aktivační funkce poskytuje tedy jisté hranice, mezi kterými může neuron produkovat hodnoty [33]. Aktivační funkce dávají vědět neuronům další úrovně, zda mají považovat neuron za aktivovaný, nebo ne. Volba aktivační funkce má výrazný vliv na dobu učení (trénování) neuronové sítě. Její výběr je často ovlivněn zkušenostmi z řešení obdobných úloh, mnohdy je nutné experimentovat.

Funkce jednotkového skoku (také Heavisideova funkce) je nejjednodušší přenosová funkce. Jedná se o nespojitou funkci, která nabízí pouze dvě možné hodnoty výstupu. 1, pokud vážený součet je vyšší nebo roven prahové hodnotě (t – *treshold*) a 0, pokud nikoliv [33].

$$f(x) = \begin{cases} 0 & x < t \\ 1 & x \geq t \end{cases} \tag{2.5}$$

Funkce jednotkového skoku může posloužit jako binární klasifikátor. Při klasifikaci do více tříd by bylo problémové síť tvořenou neurony s funkcí jednotkového skoku dovést

ke konvergenci. V takovém případě by byla žádoucí funkce, která by dokázala vyjádřit například aktivováno z 20 %, ze 70 % apod.



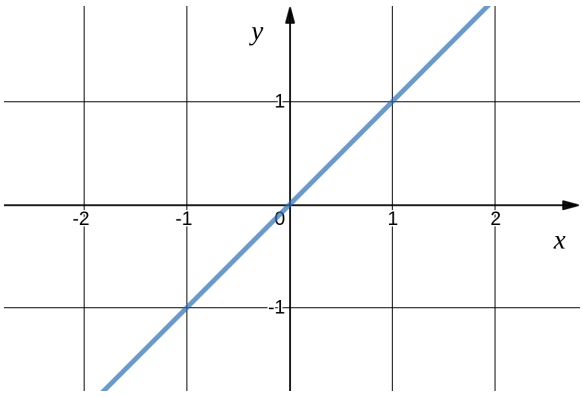
Obrázek 2.12: Graf funkce jednotkového skoku.

Lineární funkce podává výstup proporcionálně roven vstupům – váženému součtu hodnot na vstupu a přičtení zarovnání. Aktivace mohou mít různou sílu, problém jako u funkce jednotkového skoku tu tak není [33].

$$f(x) = ax \tag{2.6}$$

Problém však nastává při výpočtu gradientního sestupu (*gradient descent*), který slouží pro optimalizaci při trénování. Derivace lineární funkce je totiž konstantní, není nijak závislá na hodnotě vstupu x. Pokud nastane chyba v predikci, změny prováděné zpětnou propagací jsou konstantní a nejsou závislé na změně vstupu x.

Toto ovšem není jediný problém. Mějme vícevrstvou neuronovou síť, kde je v každé vrstvě použita lineární aktivační funkce. První vrstva je aktivována lineární funkcí. Tato aktivace zase vstupuje do další úrovně jako vstup. Další vrstva vypočítává vážený součet a podává výstup opět na základě jiné lineární funkce. To má za následek, že poslední aktivační funkce bude pouze lineární funkcí vstupů první vrstvy a všechny vrstvy by tedy bylo možné nahradit pouze jedinou.



Obrázek 2.13: Graf lineární funkce.

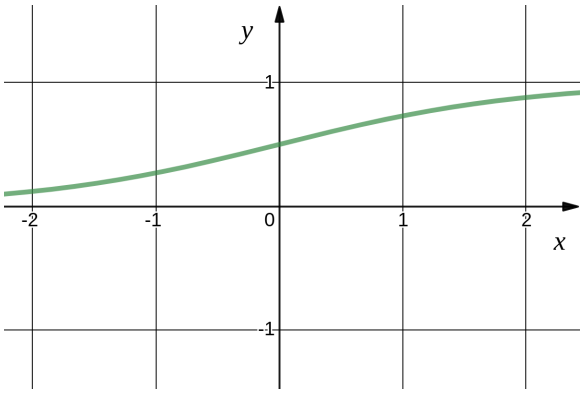
Sigmoidální přenosová funkce dává výstupy z internalu $(0, 1)$. Funkce a její kombinace jsou nelineární. Její derivace je tak závislá na hodnotách vstupů. Problémy s výpočtem

gradientního sestupu jsou pryč a stejně tak je možné efektivně skládat vrstvy neuronů za sebe [33].

$$f(x) = \frac{e^x}{e^x + 1} \tag{2.7}$$

Funkce roste v intervalu $(-2, 2)$ velmi strmě. Což znamená, že jakákoliv malá změna vstupů x v tomto intervalu, má za následek poměrně výraznou změnu na výstupu y. To má za následek, že funkce má tendenci vracet hodnoty y blízko hraničním hodnotám. Pokud jsou hodnoty vstupu naopak mimo zmíněný interval, rozdíly ve vrácené hodnotě mohou být velmi malé. To znamená, že síť se v některých případech nedokáže v rozumném čase učit. Tento problém je znám jako *vanishing gradient*.

Tyto vlastnosti činí funkci sigmoid oblíbenou a často používanou v hlubokých neuronových sítích.

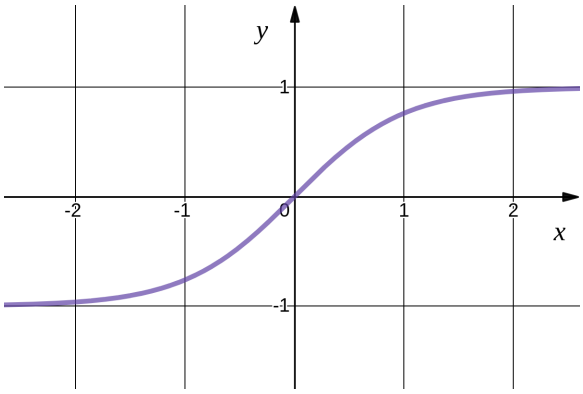


Obrázek 2.14: Graf sigmoidální funkce.

Hyperbolický tangens je hyperbolická funkce definována pomocí hyperbolického sinu a cosinu. Její průběh je podobný jako u sigmoidní funkce a potýká se se stejnými problémy (*vanishing gradient*). Obor hodnot $(-1, 1)$ je vycentrovaný okolo nuly, což usnadňuje optimalizace [33].

$$f(x) = \tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1} \tag{2.8}$$

Hyperbolický tangens je tak také velmi populární a široce používaná aktivační funkce.

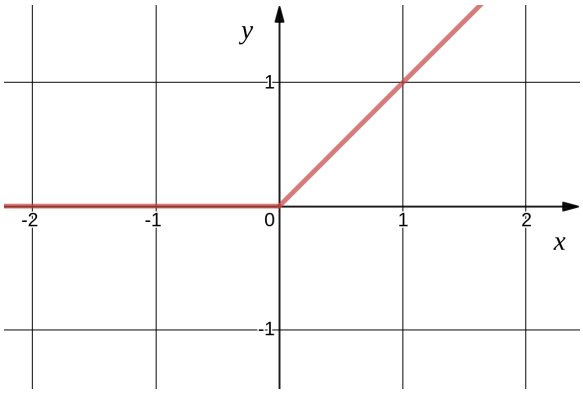


Obrázek 2.15: Graf funkce hyperbolický tangens.

Rectified linear unit (ReLU) je aktivační funkce, která vrací 0 pro záporné hodnoty a x pro hodnoty kladné. Na první pohled se může zdát, že se bude potýkat se stejnými problémy jako lineární funkce. ReLU a její kombinace jsou ale nelineární povahy a ve skutečnosti dokáží dobře aproximovat jakoukoliv funkci [33].

$$f(x) = \begin{cases} 0 & x < 0 \\ x & x \geq 0 \end{cases} \tag{2.9}$$

Díky tomu, že ReLU zahrnuje jednodušší matematické operace, jsou její výpočetní nároky oproti sigmoidní funkci a hyperbolickému tangentu výrazně nižší. Proto je v umělých neuronových sítí často využívána [31].



Obrázek 2.16: Graf funkce ReLU.

Funkce softmax se využívá v posledních vrstvách neuronových sítí, zejména u klasifikačních úloh, například přiřazení objektu do předem definovaných tříd, které nejsou vzájemně výlučné. Funkce je definována následující rovnicí [33].

$$f_i(\vec{x}) = \frac{e^{x_i}}{\sum_{n=1}^{N} e^{x_n}} \tag{2.10}$$

Funkce softmax normalizuje N rozměrný vektor $\vec{x}$ do N rozměrného vektoru $f(\vec{x})$, jehož jednotlivé prvky dávají v součtu hodnotu 1. Jelikož není možné načrtnout obecnou křivku funkce softmax, je místo grafu uveden příklad s konkrétními hodnotami na následující rovnici.

$$f([1;2;3;4;5]) = \frac{[e;e^2;e^3;e^4;e^5]}{e+e^2+e^3+e^4+e^5} = [0,012;0,032;0,086;0,234;0,636] \tag{2.11}$$

2.6 Konvoluční neuronové sítě

Konvoluční neuronové sítě (Convolutional Neural Networks – CNN) jsou hluboké neuronové sítě, kde jednotlivé skryté vrstvy mají jistou formu zaměření pro konkrétní činnost. Jsou speciálně navrženy pro rozpoznávání dvojrozměrných tvarů s vysokým stupněm invariance. Jinými slovy, rozpoznávaný objekt může být nějakým způsobem deformován, může být zachycen v různých rozlišeních či se lišit v detailech, i přesto by ho konvoluční neuronová síť měla rozpoznat. [8] Typy vrstev používané při zpracování obrazu jsou vysvětleny dále.

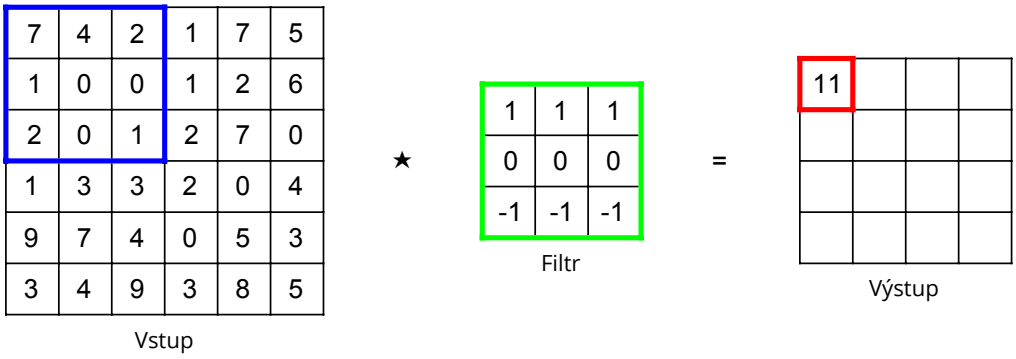
Typy vrstev

Plně propojená vrstva (*fully connected layer* nebo *dense layer*) spojuje každý neuron z dané vrstvy s každým neuronem vrstvy následující. Jedná se o stejný princip jako u klasických vícevrstvých neuronových sítí, které byly představeny v sekci 2.5. Ačkoliv jsou sítě s plně propojenými vrstvami schopny dobře aproximovat libovolnou funkci, jejich vysoký počet spojení výrazně zvyšuje nároky na výpočetní kapacity. Proto se vyskytují spíše až v koncových vrstvách konvolučních sítí, kde mají na starosti klasifikaci objektů [23].

Konvoluční vrstvy (*convolutional layers*) jsou hlavní stavební kameny konvolučních neuronových sítí. Jejich název je převzat od jména matematické operace, která se ve vrstvě provádí – konvoluce. Obraz je reprezentován diskrétními hodnotami pixelů, konkrétně maticí těchto hodnot, proto nás bude zajímat diskrétní 2D konvoluce [17]. Ta je definována rovnicí 2.12.

$$f(x,y) \star h(x,y) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} f(x-i, y-j) \cdot h(i,j) \tag{2.12}$$

Symbol $\star$ je operátor konvoluce nad funkcemi $f(x,y)$ a $h(x,y)$. První operand, v našem případě funkce f, obvykle značí vstupní obraz a druhý operand, funkce h, značí konvoluční jádro (filtr).



Obrázek 2.17: Příklad výpočtu konvoluce. Vstup jsou náhodně vygenerované hodnoty, pro zpracování obrazu by se jednalo o hodnoty jednotlivých RGB kanálů případně hodnoty jasu. Konvoluční filtr může vypadat například takto, ale může se jednat i o matici jiné velikosti.

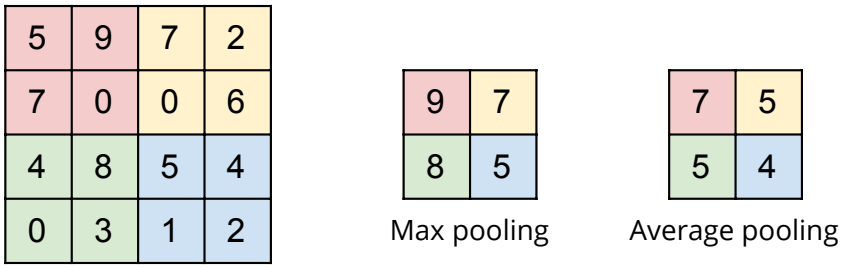
Výstupní matice je menší než vstupní. To je důsledkem toho, že pro výpočet jednoho výsledného bodu potřebujeme na vstupu matici o velikosti filtru. Pokud chceme zmenšení rozlišení předejít, dá se původní matice rozšířit. A to buď nulovými body (*zero padding*), případně jinými, vhodně vybranými hodnotami.

Parametry konvoluční vrstvy se skládají z množiny konvolučních filtrů. Samotná síť se učí optimalizací jejich jednotlivých hodnot. Filtry jsou po vstupu posouvány po určitém kroku a je počítána výsledná dvoudimenzionální mapa [17]. Konvoluční filtry mohou mít různá uplatnění. Například pro rozmazaní či doostření obrazu, ale také pro detekci hran. Hrana je rozpoznána prudkou změnou světelných podmínek. Podstata filtru je nalézt gradient, tj. směr maximální změny, který hranu charakterizuje [1].

Vlivem výpočtu v konvolučních vrstvách může dojít k tomu, že některé hodnoty v matici nabudou záporných hodnot. Barvy jsou ale reprezentovány třemi kanály v rozmezí 0 – 255.

Proto je používána přenosová funkce ReLU, viz obrázek 2.16, která odstraní negativní hodnoty z aktivační mapy. Dají se použít i jiné funkce, ale ReLU se ukázala jako nejefektivnější [15].

Pooling vrstva je další typický druh vrstvy konvolučních neuronových sítí. *Pooling*, neboli sdružování slouží pro podvzorkování – zmenšení rozměrů dvoudimenzionální mapy. Tím se zmenší počet parametrů sítě a dojde ke snížení výpočetních nároků pro další vrstvy. Realizuje se agregací několika sousedních buněk do jedné. Nejběžnější způsoby jsou *average pooling*, kdy se počítá průměrná hodnota z agregovaných buněk a *max pooling*, kdy se vybírá maximální hodnota [17].



Obrázek 2.18: Příklad *average pooling* a *max pooling* na jednoduché 4×4 matici do matice o rozměrech 2×2.

Za sdružováním stojí myšlenka, že úplně přesná lokace příznaku není pro klasifikaci důležitá. Naopak může být na škodu a podpořit problém přetrénování sítě, viz sekce 2.4. Důležitější je tedy relativní lokace příznaku vzhledem k ostatním. Tím se dosáhne větší obecnosti detektoru (generalizace) [17].

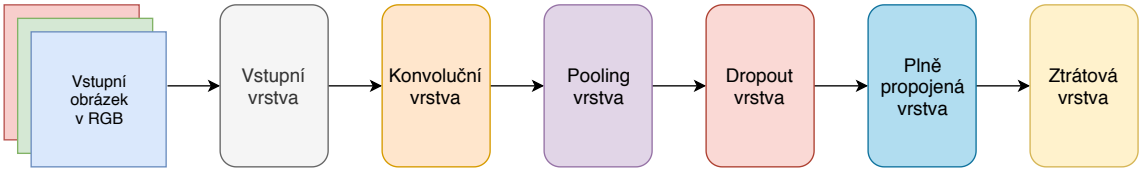
Pooling vrstva se běžně vkládá mezi jednotlivé konvoluční vrstvy. V současnosti se nejvíce používá přístup *max pooling*, který v praxi funguje nejlépe [30].

Dropout vrstva má za úkol snížit náchylnost sítě k přetrénování. Spočívá v nastavení výstupu každého neuronu ve skryté vrstvě na 0 s pravděpodobností 50 %. K deaktivaci neuronů dochází u každého vzorku dat při trénování, do učení je tak uměle přidáván šum, díky kterému je síť schopna více generalizovat. Dropout vrstva zvýšuje počet potřebných iterací k dosažení konvergence [15].

Ztrátová vrstva (*loss layer*) specifikuje jak proces trénování penalizuje rozdíl mezi predikcí sítě (výstupem) a skutečnými hodnotami (označená trénovací data). Jedná se o finální vrstvu neuronové sítě. Existuje mnoho chybových funkcí určené pro různé typy úloh, viz sekce 2.4.

Architektura konvolučních neuronových sítí

Samotných architektur konvolučních neuronových sítí je velké množství. Jednotlivé implementace se od sebe liší typem použitých vrstev a jejich množstvím. Na obrázku 2.19 je příklad napojení jednotlivých vrstev za sebe.



Obrázek 2.19: Ukázka napojení jednotlivých vrstev za sebe.

Na samém začátku je vstupní obraz reprezentovaný většinou třemi barevnými kanály. Vstupní vrstva čte hodnoty pixelů a propaguje je do dalších vrstev. V konvoluční vrstvě probíhá konvoluce. Je jich většinou několik a každá se snaží porozumět jiným základním vzorům. Například první konvoluční vrstva se snaží rozpoznávat hrany, další se snaží porozumět tvaru, barvě apod. Během procesu učení se mění hodnoty buněk jednotlivých filtrů. Konvoluční vrstvy obsahují přechodovou funkci ReLU pro odstranění případných záporných hodnot. Pooling vrstva provede podvzorkování pro dosáhnutí větší generalizace a snížení výpočetních nároků. Dropout vrstva zabraňuje přetrénování. Výstupy z některých neuronů nejsou použity, tím se do sítě zanese uměle vytvořený šum. Síť tak nebude tolik závislá na trénovacím datasetu, ale bude fungovat obecněji. Plně propojená vrstva provádí klasifikaci, typicky jich také bývá více. Při učení nastavujeme váhy pro každý vstup každého neuronu. Ztrátová vrstva dává zpětnou vazbu, jak moc se výsledek sítě liší od označených trénovacích dat.

2.7 Detekční sítě

Doposud bylo představeno využití neuronových sítí pouze pro klasifikaci obrazu. Předpokládalo se, že na snímku je pouze jeden dominantní objekt, který bude porovnáván se známými vzory a bude vyhodnocena jejich vzájemná podobnost. V reálném světě se ale ve snímcích vyskytuje velká spousta různých objektů odlišných velikostí. Je nutné zajistit nějaký mechanismus, který rozpozná, že se jedná o objekt hodný klasifikace. Obraz kolem něho ořízne a danou výseč předá klasifikátoru, který následně rozhodne, kam objekt zařadit.

Obrázek 2.20: Na levé straně je obrázek kočky. Na snímku je zachycen a perfektně oříznut pouze jeden objekt. To z obrázku dělá ideální snímek pro klasifikaci. Na pravé straně je pro změnu fotka ulice, kde se nachází spousta objektů, s různým pozadím a různě se navzájem překrývajících.

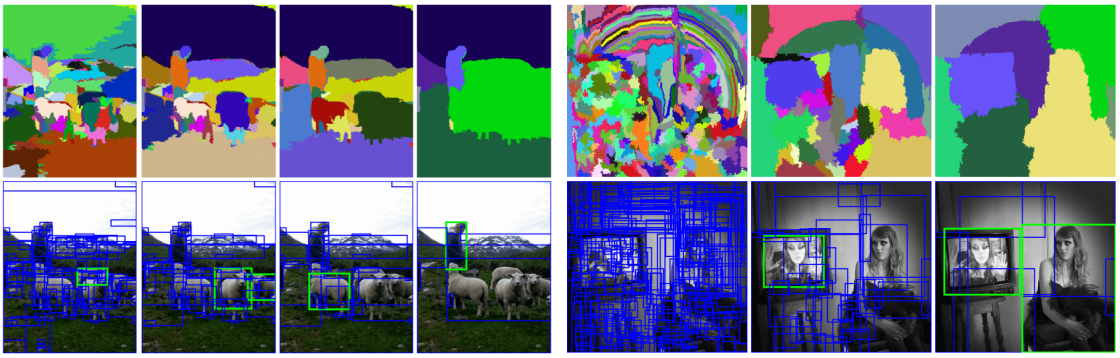
Naivní algoritmus detekce objektů by mohl spočívat v posouvání různě velikých oken po obraze a následné výseče předávat nějaké konvoluční neuronové síti, která by segment klasifikovala. Toto je velmi silový (*brute force*) způsob, který by byl velmi výpočetně náročný a neefektivní. Existují lepší řešení, které dokáží vybírat podezřelé oblasti z obrazu a ty poté předávat klasifikátoru.

Rozlišujeme detektory jednostupňové (*one-stage*) a dvoustupňové (*two-stage*). První jmenované zvládají detekci i klasifikaci v jednom kroku, jsou typicky rychlejší, ale méně přesné. Dvoustupňové pak rozdělují proces detekce a klasifikace zvlášť – obraz se musí zpracovávat dvakrát. To má za následek pomalejší celkové zpracování, ale větší přesnost (10 – 40 %) [21].

R-CNN

R-CNN (Region-based Convolutional Neural Network) a další detektory z něj vycházející byly významným pokrokem v oblasti detekce objektů. Jedná se o dvoustupňový detektor [26].

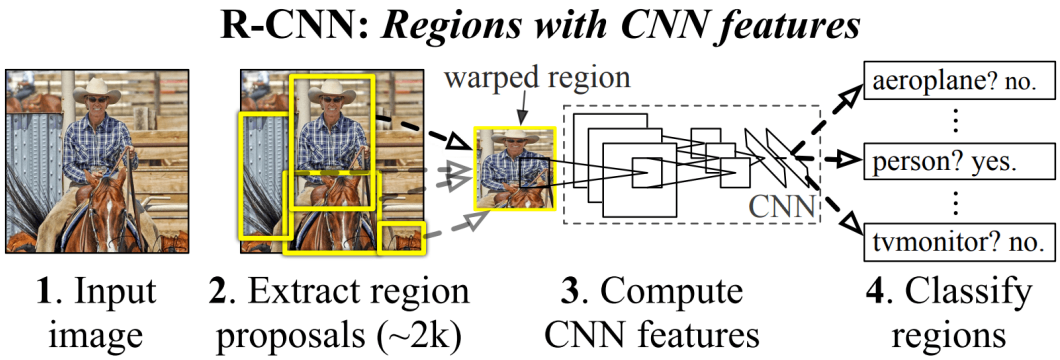
Nejprve je využit algoritmus pro detekci podezřelých oblastí (*region proposals*) nezávisle na tom, do jaké kategorie by objekt mohl spadat. Jedná se o algoritmus selektivního vyhledávání, který prochází zdrojový obraz a snaží se spojit pixely stejné nebo podobné barvy [7]. U nich lze předpokládat, že náleží jednomu objektu.



Obrázek 2.21: Dvě ukázky selektivního vyhledávání a jeho práci s rozdílnými měřítkami pro spojování barevných pixelů. Obrázky jsou převzaté [26].

Těchto výsečí je nalezeno kolem 2000. Po nalezení jsou zdeformovány a předány konvoluční neuronové síti (AlexNet [15]). Ta provede výpočet příznaků, na základě kterých pak klasifikátor SVM (Support Vector Machine) rozhodne, zda se skutečně jedná o objekt a jaké třídě s jakou pravděpodobností náleží [7].

Původní model nebyl dostatečně rychlý především kvůli rozdělení celého procesu do několika postupných fází (regrese ohraničení, extrakce příznaků pomocí konvoluční sítě a klasifikace pomocí SVM). Pro každý snímek musela konvoluční síť AlexNet spočítat příznaky pro 2000 výsečí. Z tohoto důvodu přišlo několik dalších detektorů, které původní R-CNN zdokonalily.



Obrázek 2.22: Schéma fungování detektoru R-CNN. Ze vstupního obrazu (1) jsou vyříznuty oblasti s potenciálním výskytem objektů (2). Z těchto výsečí jsou pomocí konvoluční sítě (3) získány příznaky, které jsou předány SVM klasifikátoru (4). Obrázek je převzatý [7].

Detektor **Fast R-CNN** tyto problémy řeší sdílením výpočtů konvolucí mezi jednotlivými okny s objekty. Zároveň nahrazuje klasifikátor SVM vrstvou konvoluční sítě s přechodovou funkcí softmax. Ke zlepšení výkonu také pomohlo použití sítě VGG16 oproti AlexNet [6].

Další iterace detektoru **Faster R-CNN** pak přinesla zdokonalení hledání podezřelých oblastí. Je jich nutné prohledat méně (asi 300 oproti 2000) a navíc je celý proces zrychlen. Autoři uvádějí zpracování až 5 snímků za sekundu na GPU [28].

Mask R-CNN je detektor vyvinutý výzkumným týmem Facebooku a přichází s něčím úplně novým. Oproti předchozím detektorům nepodává jako výstup souřadnice ohraničující objekt, ale provádí řazení do jednotlivých kategorií pro každý pixel obrazu [9].

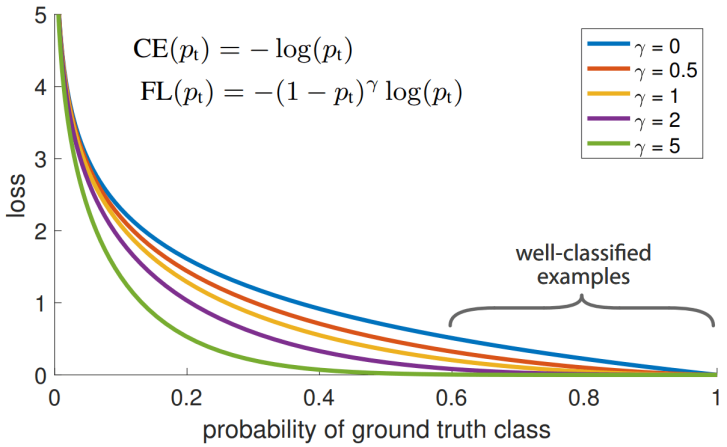
RetinaNet

Model RetinaNet je na rozdíl od detektorů rodiny R-CNN jednostupňový. Avšak díky nové chybové funkci dosahuje přesnosti vyšší než jiné jednostupňové detektory (Single Shot Multibox Detector, Deformable Parts Model, You Only Look Once, ...). Je dokonce srovnatelný s detektory dvoustupňovými [21].

Jeden z problémů dosavadních jednostupňových detektorů je nevyrovnanost tříd (*class imbalance*). Tyto detektory vyhodnocují mezi 10 000 – 100 000 možných výsečí pro každý vstupní obraz. Avšak jen velmi málo z těchto výsečí skutečně obsahuje objekt. Poměr výřezů objektů vůči pozadí je až 1 : 1000 [21]. To má za následek, že většina výsečí je klasifikátorem snadno rozpoznána jako pozadí, respektive nerozpoznána jako žádný objekt. Trénování detektoru je při vysoké nevyrovnanosti tříd neefektivní. Většina výsečí jsou snadno rozpoznatelné negativní detekce, jelikož se jedná o pozadí. Tento vysoký počet lehkých detekcí má za následek pomalé trénování případně dokonce degeneraci modelu. Detektor se totiž nejvíce učí z těžkých dat, těch je ale malé množství v porovnání se snadno identifikovatelným pozadí.

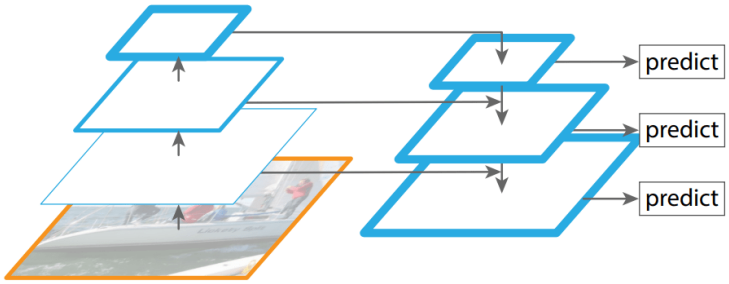
Autoři detektoru RetinaNet tento problém řeší novou chybovou funkcí **Focal loss** [21]. Ta vychází z chybové funkce Cross entropy (sekce 2.4), která je široce využívaná jak u jednostupňových tak dvoustupňových detektorů. Obohacuje ji o parametr γ, pomocí kterého je možné korigovat váhu učení. Tvar chybové funkce v závislosti na parametru γ je zachycen

na obrázku 2.23. Čím vyšší má hodnotu, tím více funkce zvýhodňuje učení z obtížnějších, ale méně často se vyskytujících, dat.



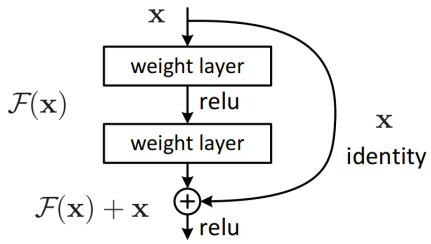
Obrázek 2.23: Funkce zachycuje hodnotu chybové funkce v závislosti na pravděpodobnosti správně klasifikovaného objektu. Je-li parametr γ roven 0, funkce je ekvivaletní s Cross entropy. Obrázek je převzatý [21].

Detektor RetinaNet dále využívá strukturu **FPN** (Feature Pyramid Network) [20]. Ta pracuje se vstupním snímkem v různých rozlišení a tím se snaží příznaky objektů zobecnit. Dokáže tak extrahovat slabé příznaky objektů zachycené v nízkém rozlišení i silné příznaky objektů v rozlišením vyšším (pyramida příznaků). Výsledkem je, že síť dokáže produkovat mapy příznaků v různých škálách obrazu, což přispívá jak ke klasifikaci, tak k regresi ohraničení.



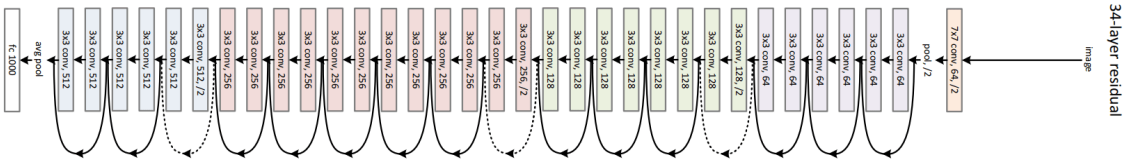
Obrázek 2.24: Příznaky jsou ze vstupního snímku extrahovány v různém rozlišení a poté porovnávány s příznaky pořízenými také v různém rozlišení. Obrázek je převzatý [20].

Je možné zvolit různé architektury konvolučních neuronových sítí pro klasifikaci (páteřní síť – *backbone*). Často je využívána síť **ResNet** (Residual Network) od výzkumného týmu Microsoftu [10]. Jedná se o jednu z nejkomplexnějších architektur, která má ve svých variacích 50, 101 nebo až 152 vrstev. S roustoucí hloubkou sítě by se přesnost teoreticky měla zvyšovat. Avšak s roustoucí hloubkou je zpětně propagovaný signál od konce sítě k jejímu počátku stále slabší. To ve výsledku znamená, že nižší vrstvy jsou učeny velmi málo či téměř zanedbatelně. ResNet tento problém řeší prostřednictvím tzv. zbytkových modelů (*residual model*).



Obrázek 2.25: Stavební blok zbytkového modelu. Obrázek je převzatý [10].

Na soutěžním datasetu ILSVRC v roce 2015 dosáhla ResNet chybovosti pouhých 3,57 %. Vzhledem k faktu, že lidé dosahují chybovosti 5 – 10 % (v závislosti na zkušenostech řešitele), se jedná o výsledek, který se blíží maximální teoreticky možné úspěšnosti [10].

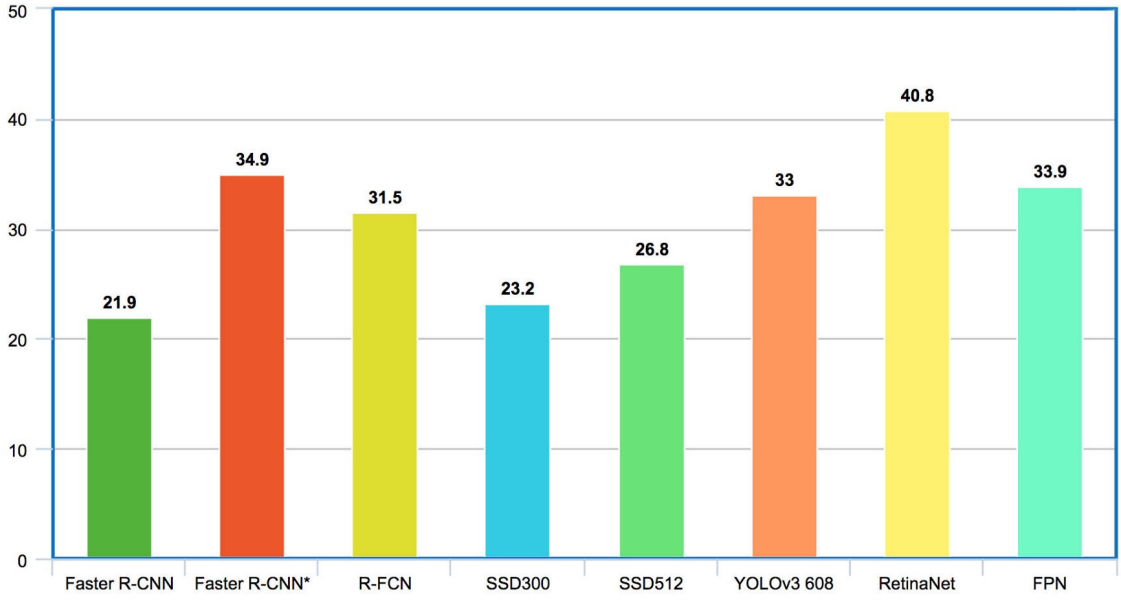


Obrázek 2.26: Architektura ResNet se znázorněním konceptu zbytkových modelů. Obrázek je převzatý [10].

Výběr detekční sítě

Dvě hlavní kritéria ovlivňující výběr detekční sítě jsou přesnost a rychlost detekce. V oblasti rychlosti si práce neklade velké cíle, není účelem dosáhnout detekce v reálném čase (*real-time*). Detektor bude detekovat osoby z výšky z pohledu ze shora. To může být poměrně náročný úkol. Jak je představeno v sekci 4.2, člověk z výšky je snadno zaměnitelný objekt. Vzhledem k typu objektů, které detektor bude detekovat, je přesnost daleko významnější kritérium.

Na obrázku 2.27 je vidět, že detektor RetinaNet překonává v přesnosti ostatní algoritmy [12]. Alespoň v měřeních na složitých datech z datasetu COCO. Navíc zachovává rozumnou rychlost detekce. Proto je vhodným detektorem pro účely této práce.



Obrázek 2.27: Porovnání přesnosti několika různých detektorů na datasetu COCO. Pro oblast detekce objektů se jedná o složitější dataset, proto detektory dosahují relativně nízké přesnosti. Obrázek je převzatý [12].

Kapitola 3

Přehled dronů a jejich srovnání

Dron, nebo také bezpilotní letadlo (UAV – *unmanned aerial vehicle*), je létající objekt, který nemá posádku. Může být ovládaný na dálku, mít předprogramovanou trasu letu nebo využívat složitější dynamické autonomní systémy. Nejčastější jsou dálkové modely letadel, vrtulníků a multikoptér. Naváděné střely, ačkoliv jsou bezpilotní a v některých případech řízeny vzdáleně, nejsou klasifikovány jako UAV, protože tyto zbraně jsou pouze na jedno použití [41].

Pokud člověk létá s jakýmkoliv dronem, stává se účastníkem leteckého provozu a musí dodržovat jistá pravidla. Létání je třeba provozovat v bezpečné vzdálenosti od lidí. Srážky jistých modelů s člověkem mohou mít závažné zdravotní následky. Také je nutné vyhnout se leteckému provozu – je zakázáno létat v blízkosti letišť a celkově létat výše než 300 metrů nad zemí. Nelze také létat v zakázaném leteckém prostoru (bezletové zóně) [41].

Počátky bezpilotních letounů sahají k počátkům 20. století. Už ve 20. letech se stroje začínaly podobat dnešním dronům, ačkoliv disponovaly podstatně většími rozměry. Původně měly sloužit hlavně pro přepravu lidské posádky, to ale nezaznamenalo výraznější úspěch. Postupem času se jejich oblast využití změnila. Dnes obvykle nesou kamery, termokamery či jiná monitorovací zařízení, kterými poskytují při leteckém snímkování důležitá data pro nejrůznější účely [41].

3.1 Využití dronů

Dnes se nejčastěji setkáváme s malými drony, které jsou dálkově ovládané a oblast jejich využití je velmi široká [16]. Stavební firmy je používají pro monitorování důležitých infrastruktur, ropovodů nebo dálnic. V energetice se pomocí dronů a speciálních kamer měří elektrické vedení. Ve filmové či hudební produkci se pomocí dronů natáčejí záběry ze vzduchu. Drony také zvládají 3D mapování povrchu, díky čemuž se vytváří geografické modely a mapy. Využívat se dají i v zemědělství, například při sledování kvality hnojení nebo osevu. Bezpilotní letadla se používají v armádě k průzkumným i útočným letům.

Drony jsou také skvělou pomůckou v oblasti bezpečnosti. Záchranáři je mohou využívat pro rychlé vyhledání nehody, policie pro sledování a průzkum terénu, hasiči pro zachycení velikosti požáru, či dokonce k jeho hašení. V oblasti zabezpečení areálů se drony používají k monitorování oblastí. Mohou také posloužit pro kontrolování a focení těžko dostupných míst, například větrných elektráren, určitých typů střech apod. Nahraná dokumentace z dronu může posloužit jako důkazní materiál.

3.2 Dostupné drony na trhu

To, jak dron vypadá, kolik stojí a jaké má rozměry, samozřejmě záleží od účelu využití, ke kterému byl konstruován. Největší a nejdražší jsou armádní bojové drony – bezpilotní letouny. Ty disponují rozpětím křídel až desítkami metrů a stojí v přepočtu stovky miliónů korun. Mají velkou dráhu doletu a vysokou vzletovou výšku. Může jít o ozbrojené drony, s nimiž armáda likviduje cíle, aniž by byly ohroženy životy pilotů, nebo mohou sloužit k rozsáhlým špionážním akcím [16].

Filmařské drony jsou samozřejmě vybaveny kvalitní nahrávací technikou. Disponují obvykle několika vrtulemi pro zajištění větší stability. Příchod dronů změnil styl novodobého natáčení leteckých scén. Pronájem, potažmo koupě reálného vrtulníku, je samozřejmě vysoce nákladná záležitost, letecké záběry se tak staly mnohonásobně dostupnější. Cena špičkových filmařských dronů se pohybuje v řádech sta tisíců korun [16].

Donáškové drony jsou zatím spíše futuristická záležitost. Zahraniční technologické firmy, jako Amazon nebo Google, již testují své služby [14]. Zboží o váze několik kilogramů chtějí rozvážet menšími vrtulovými drony. Problém ani tak není technologický vývoj, ale zákony. V současnosti ve většině světa není takovýto způsob dopravy zboží legislativně nijak ošetřen. Jde však pravděpodobně o otázku času, než se začnou donáškové drony využívat.

Komerční drony stojí řádově desítky tisíc korun a jsou tak cenově nejdostupnější. Lze s nimi snadno manévrovat a dokáží pořizovat dostatečně kvalitní záběry pro běžnou spotřebu. Oblíbené tak jsou u cestovatelů, kteří dokáží ocenit především panoramatické záběry, na které by jinak neměli nárok, tak u leteckých nadšenců a amatérských pilotů. Několik konkrétních modelů je popsáno ve zbytku této kapitoly [13].

Obrázek 3.1: Ukázky dronů podle typu použití. Armádní, filmařský, doručovací a komerční dron. Obrázky jsou převzaté [41, 14].

DJI Mavic 2 Pro

Čínská firma DJI je v současnosti nejzvučnější jméno v oblasti civilních dronů. Produkují několik odlišných řad, pro různé cílové skupiny. Inovují, udávají trendy a v současnosti pokrývají asi 70% trhu [13].

Mavic 2 Pro je nástupce velmi úspěšného dronu Mavic Pro. Váží 907 g, dokáže létat rychlostí až 72 km / h a ve vzduchu vydrží až 31 minut. Dá se ovládat až do vzdálenosti 18 km. Má špičkovou kameru s rozlišením 20 MP, která dokáže natáčet video až ve 4 K rozlišení a v 60 snímcích za vteřinu. Nabízí spoustu módů pro natáčení a focení, například *hyperlapse* pro stabilizované časosběrné záznamy. Dron se dá složit do kompaktních rozměrů – $214 \times 91 \times 84$ mm). Mezi komerčními drony se jedná o drona z vyšší cenové kategorie, pořizovací cena je 1 500 $. Technické parametry jsou převzaté z oficiálních stránek výrobce [35].

Obrázek 3.2: DJI Mavic 2 Pro. Obrázek je převzatý [35].

S ohledem na monitorování chodců se jedná o ideálního drona. Důležité jsou zejména kvalitní snímky ve vysokém rozlišení, které umožní lidi ve videu lépe identifikovat. Čím vyšší rozlišení, tím z větší výšky je možné chodce spolehlivě detekovat. Výdrž baterie se dá považovat za limitující, bohužel v současné době je toto technologické maximum. Žádná konkurence nenabízí výrazně delší výdrž. Dosah letu až 18 km je naprosto dostačující, i s ohledem na maximální dobu letu.

DJI Spark

Spark je malý, lehký, kompaktní dron, kterého není problém strčit do batohu či dokonce kapsy od bundy. Váží pouhých 300 g, je schopen nabrat rychlost až 50 km / h a létat vydrží 16 minut. Dosah doletu je 2 km v ideálních podmínkách. Kamera má rozlišení 12 MP a je schopna natáčet video v rozlišení Full HD ve 30 snímcích za vteřinu. Spark se pohybuje v jiné cenové kategorii než Mavic 2 Pro a stojí 500 $. Technické parametry jsou převzaté z oficiálních stránek výrobce [34].

Obrázek 3.3: DJI Spark. Obrázek je převzatý [34].

Spark není dron pro náročné piloty. Jedná se spíše o rodinného drona, který je snadný na ovládání a může dobře posloužit například pro focení na dovolené. Spark je 3× levnější než Mavic Pro, daň za to je nižší výdrž baterie a horší kvalita videa. Je to pochopitelné, cílová skupina obou dronů je rozdílná. Pro monitorování chodců se samozřejmě dá použít, ale všechny jeho kompromisy jsou v tomto ohledu na škodu. Rozlišení omezí výšku, ze které by bylo možné osoby detekovat. Kapacita baterie zase dobu, po kterou je možné monitorování realizovat. Spark má také výrazně kratší doletovou vzdálenost.

Parrot Bebop 2

Dron od francouzské společnosti Parrot SA. Narozdíl od předchozích dronů od společnosti DJI, se Bebop 2 ovládá pomocí aplikace v chytrém telefonu nebo tabletu. Váží 500 g, dokáže létat rychlostí až 60 km / h a ve vzduchu vydrží až 25 minut. Maximální dosah letu je pouhých 300 m. Má 14 MP čočku kamery a zvládne pořizovat videozáznam v rozlišení Full HD při 30 snímcích za sekundu. Jedná se o drona ze stejné cenové kategorie jako DJI Spark a pořizovací cena se pohybuje kolem 500 $. Technické parametry jsou převzaté z oficiálních stránek výrobce [25].

Obrázek 3.4: Parrot Bebop 2. Obrázek je převzatý [25].

Bebop 2 se dá považovat za konkurenta Sparka. Je těžší a větší, díky tomu má větší kapacitu baterie a vydrží létat delší dobu. V letové rychlosti a ceně jsou na tom se Sparkem velmi podobně. Bebop 2 má ale nízký maximální dolet. To by při monitorování mohlo být limitující, pilot by se musel nacházet velmi blízko u monitorované oblasti. Rozlišení pořízovaného záznamu je shodné, takže platí to co u Sparka.

Kapitola 4

Trénování detektoru

Při trénování neuronových sítí probíhá spoustu relativně snadných výpočtů. Například aplikace konvolučních filtrů na vstupní obraz nebo výpočet podvzorkování v pooling vrstvách. Tyto jednotlivé výpočty jsou na sobě vzájemně nezávislé, proto mohou být prováděny paralelně. Doba trénování tedy může být mnohonásobně urychlena použitím grafických karet, které mají běžně tisíce jader.

Využitá knihovna Tensorflow podporuje akcelaraci pomocí grafických karet s podporou architektury CUDA (Compute Unified Device Architecture) [24]. Jedná se o platformu pro urychlení výpočetně náročných úloh. Typicky takových, kde je možnost snadno využít paralelizaci. Použití této architektury je omezeno pouze na grafické akcelerátory a výpočetní karty společnosti nVIDIA, která ji vyvíjí.

Pro účely hlubokého učení stojí nad architekturou CUDA ještě knihovna cuDNN (CUDA Deep Neural Network library) [24]. Ta poskytuje vysoce vyladěné implementace pro standardní úlohy, jako jsou konvoluce, podvzorkování, normalizace a aktivace vrstev. Toto API pak už přímo využívají vývojáři knihoven pro hluboké učení a odprošťuje je od ladění nízkoúrovňového výkonu GPU.

Samotné trénování pak probíhalo na stolním počítači s grafickou kartou nVIDIA GeForce GTX 1070, která má 1920 CUDA jader. Bylo nutné manuálně doinstalovat veškeré závislosti varianty Tensorflow využívající GPU akceleraci. Fizyr RetinaNet vyžaduje knihovnu Keras verze 2.2.4, ta je součástí až Tensorflow 1.13. Tato verze Tensorflow podporuje pouze CUDA 10.0 a cuDNN 7.4.1, případně novější. CUDA 10.0 požaduje ovladač grafické karty verze alespoň 410. Bylo nutné vše podřídit těmto závislostem. Po jejich úspěšném nainstalování bylo trénování realizováno na čisté instalaci Ubuntu 18.04 s ovladačem GPU 418.43, CUDA 10.0 a cuDNN 7.4.2.

4.1 Implementace detektoru

Jak bylo vysvětleno v sekci 2.7 detekční síť RetinaNet se ukázala vhodná pro účely této práce. Zvolil jsem implementaci od společnosti Fizyr.[1] Jedná se o nizozemskou firmu, která se specializuje na vytváření software v oblasti hlubokého učení. Implementace je *open-source* a je veřejně přístupná na repozitáři GitHub. Síť je napsána v jazyce Python pomocí knihovny Keras v její nejnovější verzi 2.2.4 využívající knihovnu Tensorflow jako *backend*. Keras poskytuje vysokoúrovňové rozhraní pro rychlé a pohodlné vytváření vlastních modelů neuronových sítí. Další výhodou knihovny je, že nabízí jednotné rozhraní pro práci

[1] Github repozitář Fizyr RetinaNet je dostupný na https://github.com/fizyr/keras-retinanet

s různými *backend* knihovnami z oblasti strojového učení. V současné době je to kromě již zmíněného Tensorflow od Googlu také Theano, vyvíjené na univerzitě v Montrealu a Microsoft Cognitive Toolkit.

Předtrénované modely sítě RetinaNet jsou schopné rozpoznávat kolem 80 různých tříd objektů. Je mezi nimi samozřejmě i člověk. Experimentování ale potvrdilo předpoklad, že z pohledu ze shora síť chodce spolehlivě nerozezná, ani nedetekuje jako žádný jiný objekt. Proto bylo nutné opatřit vhodný dataset a natrénovat vlastní model.

4.2 Dataset

Typický záběr monitorování oblasti dronem, kde se mohou nacházat lidé, je z výšky z pohledu ze shora. Proto pro úspěšné natrénování modelu bylo potřeba obstarat dataset obsahující záběry z tohoto pohledu a souřadnice ohraničující pozici člověka ve snímku (anotovaný dataset).

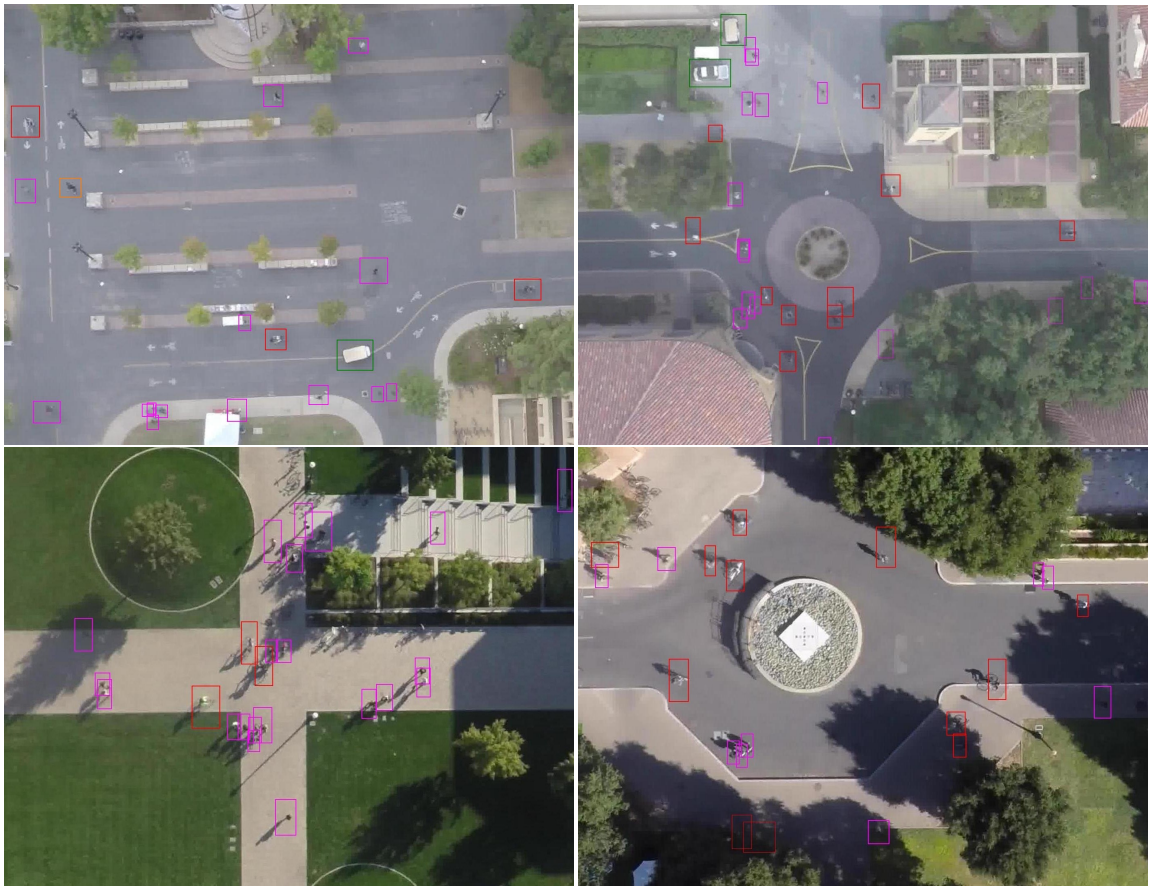
Na klasických uložištích datasetů, jako jsou například Google Open Images, Kaggle, Pascal VOC (The PASCAL Visual Object Classes) nebo COCO (Common Objects in Context), je velké množství datasetů různých kategorií. Ačkoliv dat zachycujících člověka je mnoho, nepodařilo se mi najít žádnou specifickou kategorii pro rozpoznání člověka z výšky. Bylo proto nutné hledat jiný zdroj.

Stanford Drone Dataset (dále už jenom SDD) je dataset záběrů frekventovaných míst univerzitního kampusu na americké univerzitě ve Stanfordu.[2] Dataset vznikl v rámci práce *Learning Social Etiquette: Human Trajectory Understanding in Crowded Scenes* [29], která se zabývá analýzou a předpovídáním pohybů nejenom chodců, ale i cyklistů, aut, autobusů apod.

Dataset obsahuje videozáznamy ve formátu mp4 z 8 různých lokací o celkové velikosti 72 GiB. Každému videozáznamu přísluší anotační textový soubor, kde je zaznamenána poloha objektů na každém snímku videa. Druhy zaznamenávaných objektů jsou chodec, cyklista, člověk na skateboardu, auto, autobus a golfový vozík. Každá anotace také obsahuje tři příznaky: *lost*, *occluded* a *generated*. Příznak *lost* je zde z důvodu, pro který byl dataset vytvářen – predikce pohybu cíle. Značí, zda se objekt nachází mimo zaznamenávaný obraz. Tedy cíl se objeví v zaznamenávané ploše, a když ji opustí, je odhadováno kudy se mohl pohybovat dále. Příznak *occluded* značí, že něco objekt překrývá směrem ke kameře. Například chodec zašel pod strom a není na kameře vidět. A *generated* říká, zda byla anotace automaticky interpolována. Některé anotace byly tvořeny manuálně, zbylé byly vygenerovány (většina).

Použitá implementace RetinaNet zahrnuje trénovací skript `retinanet-train` pro doučení vlastního modelu. Skript umí formáty anotací největších skladišť datasetů. Zvolený dataset pro trénování však žádné z nich neodpovídá. Vlastní anotace je třeba skriptům předat ve formátu CSV a tvaru `path_to_image,x1,y1,x2,y2,class_name`. Z charakteristiky datasetu SDD a zvolené implementace sítě vyplývá, že pro účely této práce bude muset být dataset upraven. A to jak samotná trénovací data tak anotace.

[2] Stanford Drone Dataset je dostupný na http://cvgl.stanford.edu/projects/uav_data



Obrázek 4.1: Ukázka z použitého datasetu Stanford Drone Dataset.

Kvůli syntaxi anotací bylo nutné videozáznamy rozložit po jednotlivých snímcích a upravit informace z původních anotací do požadovaného formátu. Zároveň pro správné naučení sítě bylo třeba z anotací vyřadit objekty jiných tříd než chodců a také všechny objekty označené jako *lost* a *occluded*. Veškeré úpravy datasetu probíhaly pomocí skriptů napsaných v jazyce Python využívajících knihovny OpenCV.

Upravený dataset obsahoval přes 520 000 obrázků ve formátu jpg o celkové velikosti 150 GiB, anotovaných chodců bylo více než 3 800 000. Zda anotace opravdu odpovídají konkrétním snímkům, bylo otestováno pomocí debugovacího nástroje `retinanet-debug`, který je taktéž součástí použité implementace. Umožňuje vizualizovat anotované objekty – označí objekt v daném snímku podle souřadnic uvedených v anotaci.

4.3 Proces trénování

Samotné trénování probíhalo pomocí již zmíněného trénovacího skriptu `retinanet-train`. Váhy byly u každého trénování inicializovány z předtrénovaného modelu na datasetu ImageNet. To by mělo zajistit rychlejší konvergenci než u náhodné inicializace. Jako páteřní síť byla zvolena dopředná konvoluční síť ResNet s 50 vrstvami. Byla vybrána příslušná grafická karta a adresář pro ukládání tensorboard statistik z trénování. U každého trénovacího snímku byla jistá šance, že se před trénováním otočí, jelikož síť by neměla být zavislá na tom, jak je chodec natočený ke kameře a jakým směrem jde. Každé trénování probíhalo 60 epoch – 60× byl síti představen celý dataset. Při trénování neuronové sítě je běžná

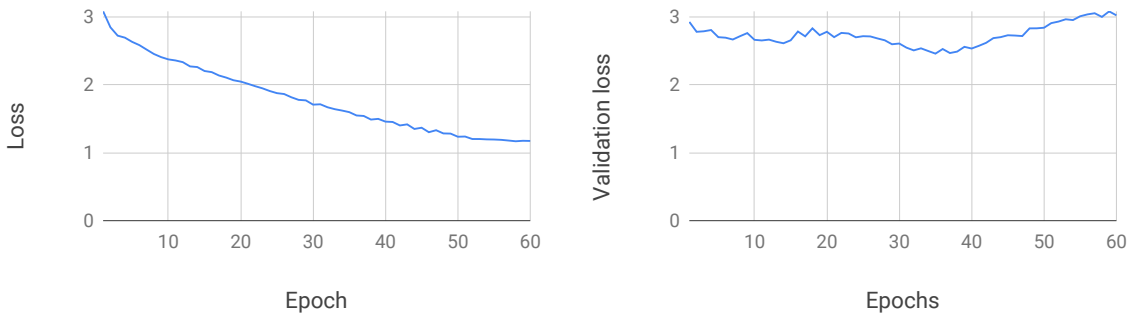
praxe, že se obrázky síti nepředkládají po jednom, ale po balíčcích (*batches*) určité velikosti (*batch size*). Trénování se poté skládá z kroků (*steps*), které jsou definovány rovnicí 4.1.

$$steps = \frac{velikost\ datasetu}{batch\ size} \tag{4.1}$$

Počet kroků byl zvolen na 2 500, což s množstvím snímků v průměru 10 000 znamená, že síti byly předkládány anotační snímky po zhruba 4 kusech. Dále z každé scény byly vybrány snímky z jednoho videa, které sloužily jako validační data. Jednalo se zhruba o 10 % dat. Každé trénování, celých 60 epoch, proběhlo za čas kolem 20 hodin. Veškeré hromadné manipulace a modifikace datasetu probíhaly pomocí skriptů napsaných v jazyce Python či Bash.

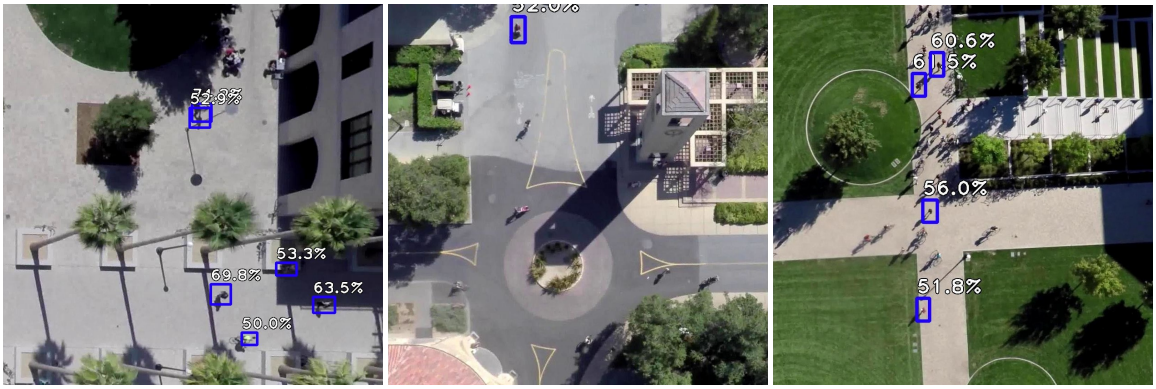
První trénování

Po úpravách datasetu zmíněných v sekci 4.2 bylo k dispozici přes 520 000 obrázků s 3 800 000 anotovanými chodci. Snímky jsou však pořízeny z videa o 30 fps. Dá se předpokládat, že v průběhu jedné vteřiny, se příliš věcí v záběru nezmění. Chodci budou posunuti o pár pixelů dále, ale že by každý snímek přinášel nové relevantní informace o tom, jak chodec vypadá, je nepravděpodobné. Použití všech snímků by znamenalo pouze prodloužení procesu trénování, protože v podstatě stejná, či velmi podobná informace by byla zaznamenána na několika snímcích. Proto byl pro trénování použit až každý 50. snímek. To znamenalo, že pro první učení bylo pro trénování anotováno 67 386 chodců a pro validaci 9 907.



Obrázek 4.2: Porovnání chyby a validační chyby modelu z prvního trénování v závislosti na množství odtrénovaných epoch.

Nejlépe si trénovaný model vedl kolem 35. epochy, potom validační chyba začala růst a model začal jevit známky přetrénování. Z podrobnějšího prozkoumání a otestování výsledků prvního trénování vyplynulo, že spousta cyklistů je mylně rozpoznáno jako chodci a spousta chodců zase jako cyklisti. V původním datasetu tvořili anotovaní chodci zhruba 45 % a cyklisté 40 %. Z výšky je velmi těžké, někdy až nemožné rozeznat co je cyklista a co chodec. Na snímcích je zachycena spousta cyklistů, kteří vypadají jako chodci. Tyto anotace síť mátly a to byl jeden z důvodů vysoké validační chyby a nepřesnosti modelu.



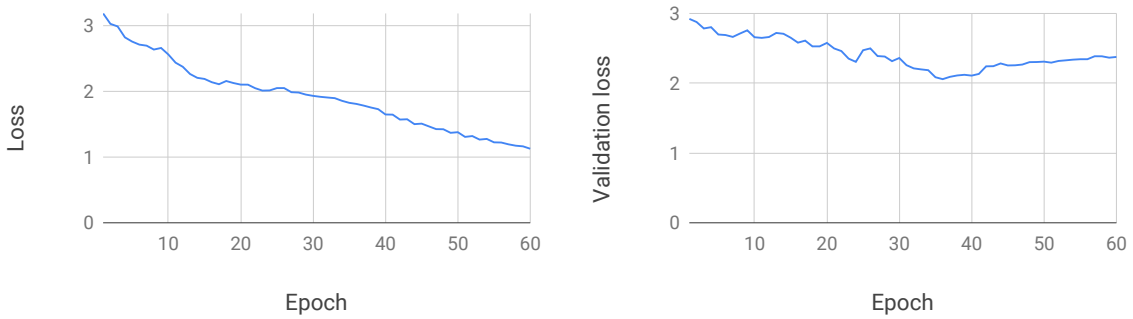
Obrázek 4.3: Validace modelu z prvního trénování na validačních snímcích.

Obrázek 4.4: Validace modelu z prvního trénování na testovacích snímcích, které nebyly součástí původního datasetu.

Druhé trénování

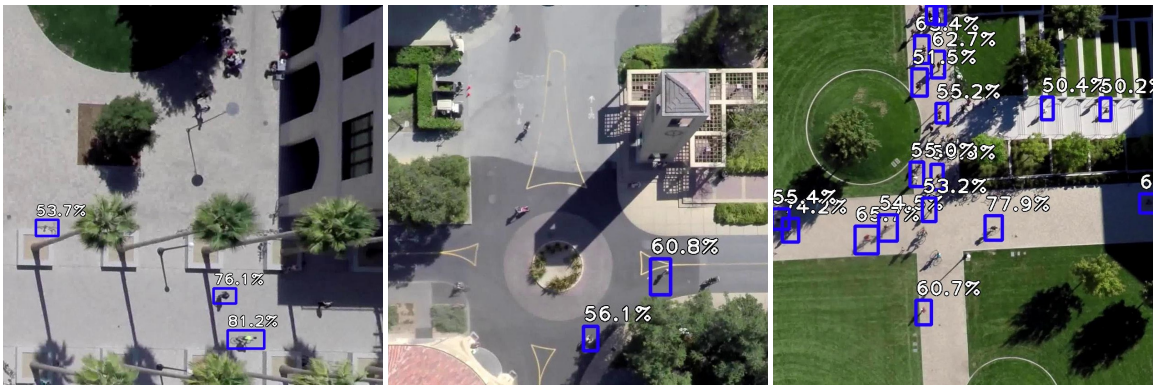
Na základě poznatků z prvního trénování byl dataset pro další trénování upraven. Do anotací byli zahrnuti nejen chodci, ale také cyklisté a lidé na skateboardu. To vše pod jednou společnou třídou.

Druhá úprava spočívala v úpravě počtu snímků z jednotlivých scén. Videa z některých scén byla výrazně kratší, či obsahovala výrazně odlišný počet anotací. To mohlo mít za následek, že některé typy záběrů byly sítí preferovány. Například z určité výšky, z určitého úhlu či daného barevného spektra ovlivněné světelnými podmínkami ve scéně. Proto pro další trénování nebyl vybrán každý 50. snímek z každé scény, ale snímkování bylo ovlivněno počtem anotací z jednotlivých scén. A to tak, aby každá scéna obsahovala zhruba stejný počet anotací. Pro druhé učení bylo trénovacích anotací 55 241 a validačních 8 263.



Obrázek 4.5: Porovnání chyby a validační chyby modelu z druhého trénování v závislosti na množství odtrénovaných epoch.

Jako u prvního trénování si model nejlépe vedl kolem 35. epochy. Výsledky byly lepší, ale ne uspokojivé. Dalším zkoumáním bylo zjištěno, že spousta anotací je uvedena úplně mylně. Například příznaky *occluded* nebyly uvedeny ve 100 % případech, a tak v anotacích byla spousta anotovaných objektů například pod stromem či pod střechou domu. Jindy byly zase anotace úplně mimo objekt nebo objekt pokrývaly jen zčásti. Tyto neduhy datasetu byly pravděpodobně způsobeny skutečností, s jakým účelem byl dataset vytvářen – predikce pohybů objektů. V takovém případě není důležité, zda je chodec na kameře skutečně vidět, je skrytý pod stromem nebo zda je anotován pouze zčásti. Bohužel chybných anotací bylo poměrně velké množství a pro lepší a přesnější výsledky bude nutné dataset ručně projít a obrázky s chybnými anotacemi vymazat.

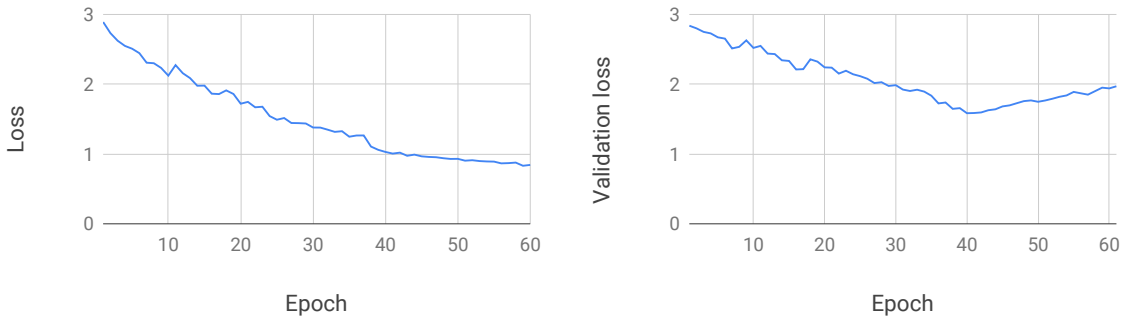


Obrázek 4.6: Validace modelu z druhého trénování na validačních snímcích.

Obrázek 4.7: Validace modelu z druhého trénování na testovacích snímcích, které nebyly součástí původního datasetu.

Třetí trénování

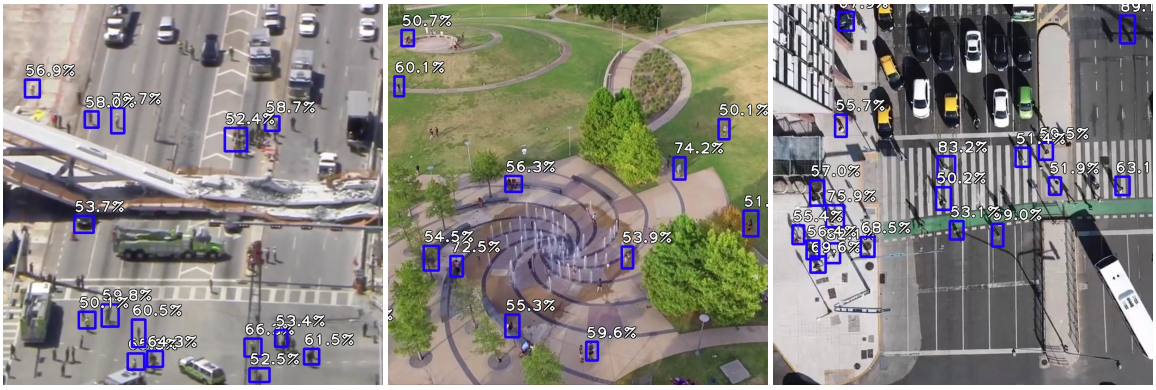
Pro třetí trénování byl dataset ručně protřízen. Pomocí skriptu `retinanet-evaluate`, který je součástí použité implementace RetinaNet, byly vykresleny anotace přímo do příslušných snímků. Poté byly ručně zkontrolovány a ty, kde byly anotace nepřesné či úplně chybné, byly vyřazeny. Celkově bylo vymazáno 1 926 snímků z 9 549 vygerovaných stejným způsobem jako u druhého trénování, aby anotace z žádné scény nepřevažovaly. Anotací pro trénování zůstalo 63 433 a anotací pro validaci 14 010.



Obrázek 4.8: Porovnání chyby a validační chyby modelu z třetího trénování v závislosti na množství odtrénovaných epoch.

Model tentokrát dosáhl nejlepších výsledků kolem 40. epochy, ty byly opět lepší než v předchozím trénování. Validační chyba dosáhla minimální hodnoty 1,58. Tyto výsledky již považuji vzhledem k výpočetním možnostem a rozmanitosti datasetu za dostačující.

Obrázek 4.9: Validace modelu z třetího trénování na validačních snímcích.



Obrázek 4.10: Validace modelu z třetího trénování na testovacích snímcích, které nebyly součástí původního datasetu.

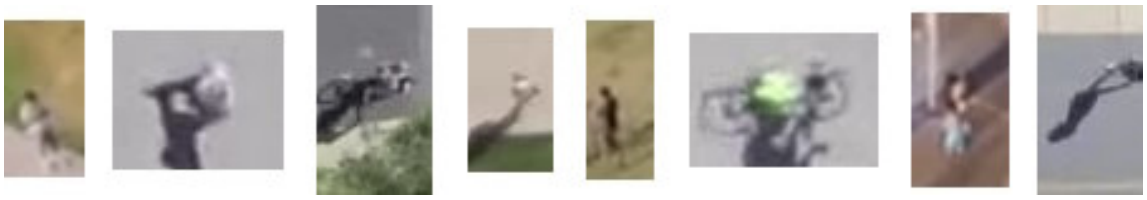
Kapitola 5

Algoritmus monitorování chodců

Neuronová síť je po natrénování schopná s jistou přesností detekovat osoby v obraze. Díky tomu je možné určit, na jakých souřadnicích a s jakou pravděpodobností se na nich člověk nachází. Nyní je třeba navrhnout algoritmus identifikace osob – jednotlivé chodce od sebe navzájem rozlišit a vizualizovat trajektorie jejich pohybů v průběhu celého videa. Také vyřešit otázku zpracování videa, tedy postupného dávkování snímků detektoru. A nakonec implementovat demonstrační aplikaci pro kompletní otestování funkcionality a provedení experimentů.

5.1 Návrh

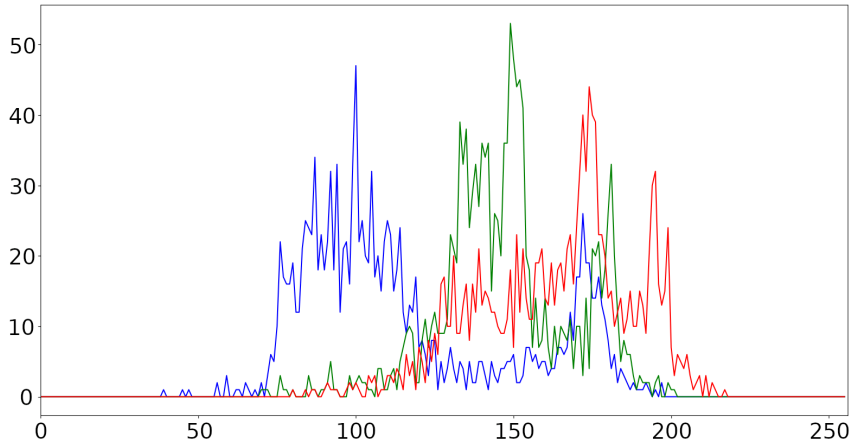
Vzhledem k tomu s jakými typy záběrů se pracuje, jsou detekovaní lidé běžně zaznamenáni v matici o výšce a šířce řádově desítek pixelů. To není mnoho, navíc z toho značnou část tvoří pozadí. Lidé také mohou stát přímo kolmo ke kameře. V takových záběrech pak tělo člověka není zachyceno téměř vůbec. Jak je vidět na obrázku 5.1, síť si také někdy pomáhá detekcí typického stínu člověka, pokud jsou záběry pořízeny ve světelných podmínkách, které to umožňují. Díky těmto aspektům není možné využít nové sofistikované algoritmy pro reidentifikaci lidí, jako například AlignedReID [43].



Obrázek 5.1: Výřezy detekovaných osob neuronovou sítí.

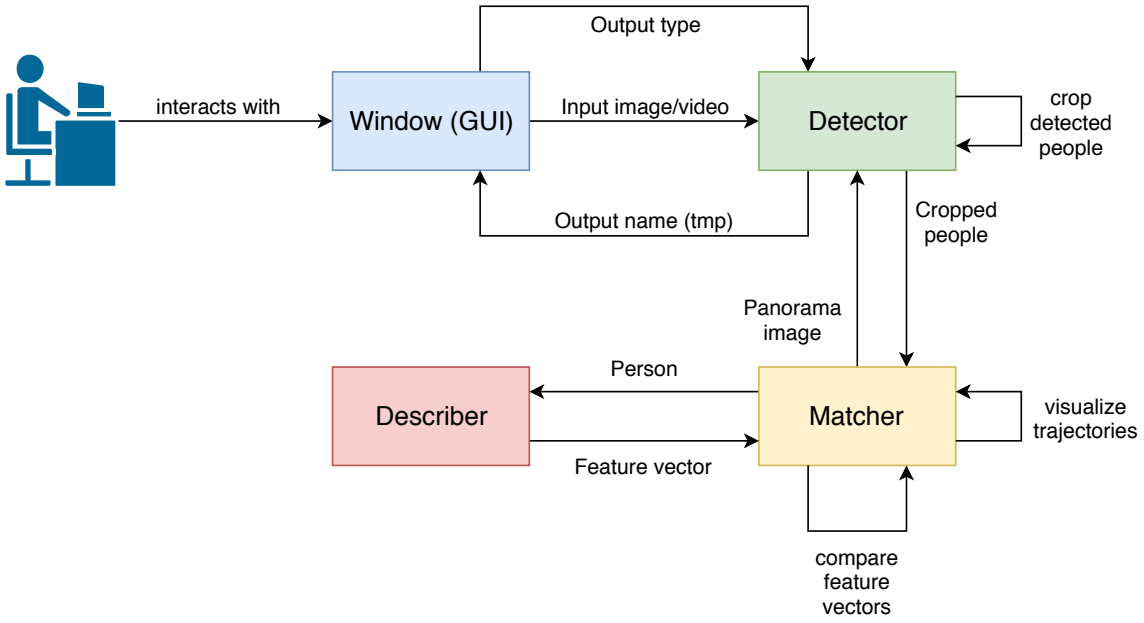
Díky výše zmíněným skutečnostem je zvolen jednodušší a přímočařejší přístup, a sice extrakce příznaků pomocí **barevných histogramů**. Každé detekované osobě ve snímku bude spočítán barevný histogram – jak jsou jednotlivé barevné kanály v daném segmentu rozloženy. Barevný histogram bude sloužit jako příznakový vektor (*feature vector*). Jedná se o seznam hodnot, který popisuje konkrétní obrázek a může být porovnáván s jinými příznakovými vektory. Detekované osoby, kterým bude spočítán barevný histogram, budou označeny jako viděné a jejich příznakové vektory budou uloženy. V dalším snímku budou opět vypočítány příznakové vektory pro každého detekovaného chodce. Následně budou porovnávány s příznakovými vektory již viděných chodců. Pokud bude nalezena dostatečná

shoda, chodec bude považován za reidentifikovaného, jeho příznakový vektor bude nahrazen aktuálnějším a jeho souřadnice v aktuálním snímku budou uloženy.



Obrázek 5.2: RGB histogram jednotlivých barevných složek pro první osobu zleva z obrázku 5.1. Na ose x je vyznačeno 256 možných hodnot jednotlivých barevných složek. Na ose y počet pixelů s touto barevnou hodnotou.

Výsledná aplikace People Detector pak bude fungovat následovně. Uživatel přes grafické uživatelské rozhraní specifikuje vstup a spustí detekci. Aplikační okno dávkuje vstupní soubor po jednotlivých snímcích a postupně je předává detektoru. Ten v každém snímku detekuje osoby, vyřízne je a předá matcheru. Matcher předává výřezy dále describeru, který mu vrací příznakové vektory. Matcher příznakové vektory ukládá a porovnává. Po zpracování všech výřezů vizualizuje trajektorie chodců a výslednou mapu předá detektoru. Uživatel poté může pokračovat v práci s aplikací. Například si výstup uložit, zadat nový soubor či spustit detekci znovu. Návrhový diagram celé aplikace je na obrázku 5.3.



Obrázek 5.3: Návrhový diagram aplikace People Detector.

5.2 Implementace algoritmu

Po spuštění aplikace je vytvořena instance detektoru, která načte natrénovaný model sítě RetinaNet ve formátu h5. Výchozí cesta je nastavena k natrénovanému modelu po 40. epoše z třetího trénování, které bylo popsáno v sekci 4.3. Použitý model ale není problém změnit. Lze tak libovolně využít jiný natrénovaný model, stačí specifikovat cestu v detektoru.

Uživatel interaguje s aplikací přes její grafické uživatelské rozhraní. Zadá cestu ke vstupnímu souboru (obrázek nebo video ve formátu jpg, jpeg, png, bmp, mp4, avi, wmv, mov nebo mkv) a vybere typ výstupu. Má na výběr ze dvou možností: Za 1. vizualizovat pouze ohraničující obdélníky kolem lidí a zobrazit pravděpodobnost, jak moc si je aplikace jejich detekcí jistá nebo za 2. vizualizovat trajektorie, kudy se osoba napříč videem pohybovala. 2. možnost pochopitelně nemá smysl v kombinaci s obrázkem na vstupu.

Pokud uživatel zadá jako vstup video, je třeba, aby bylo detektoru předáváno po snímcích. To je realizováno následujícím algoritmem.

```
def process_video(window, input_file, output_type):
    for frame in input_file:
        window.actualize_input_label(frame)
        objects_detected = detect_objects(frame)
        image_with_boxes = draw_bounding_boxes(frame, objects_detected)
        if output_type == BORDERS:
            out_file.write(image_with_boxes)
        elif output_type == PANORAMA:
            objects_cropped = crop_objects(frame, objects_detected)
            matcher.process_objects(frame, objects_cropped)
        window.actualize_output_label(image_with_boxes)
        window.progress_bar += step
    if output_type == PANORAMA:
        panorama_image = matcher.get_panorama()
        window.actualize_output_label(panorama_image)
```

Výpis 5.1: Funkce v pseudokódu Pythonu pro zpracování vstupu v detektoru.

Po zadání vstupů uživatel spustí detekci. Okno aplikace dávkuje vstupní soubor po snímcích a ty postupně předává detektoru. Detektor pomocí natrénovaného modelu neuronové sítě detekuje ve vstupních datech osoby. Pokud je jako výstup požadováno pouze vyznačení ohraničujících obdélníků, je tak učiněno a výstupní soubor je uložen do dočasného adresáře. Cesta k němu je předána oknu aplikace v případě, že by si uživatel přál výstup uložit. Pokud je jako výstup požadovaný panoramatický snímek, lidé jsou z obrazu vyříznuti a jsou předáni matcheru.

Matcher využije describer, aby pomocí barevných histogramů získal vektor příznaků jednotlivých osob. Ty poté porovnává a jejich výskyty zaznamenává do výsledného panoramatického snímku. Ten následně předá detektoru, který výstupní soubor uloží do dočasného adresáře a cestu k němu předá oknu, odkud si uživatel může soubor uložit.

Není implementován žádný *image stitching* algoritmus pro spojování snímků napříč videem. Trajektorie pohybu lidí jsou vykreslovány do prvního snímku videa. Z toho plyne zásadní omezení, a sice že pořízené video musí být ze statického bodu – dron se v průběhu pořizování záznamu nesmí hýbat. Každému rozpoznanému chodci jsou propojena všechna místa, kde byl v průběhu videa identifikován. Kolem chodce je vyznačen ohraničující obdélník (*bounding box*) a z jeho středu vede linie, kudy se bude dál ve videu pohybovat.

5.3 Extrakce a porovnávání příznakových vektorů

Skutečně použitý histogram se od toho na obrázku 5.2 liší. Byl použit třídimenzionální histogram a jednotlivé barevné kanály byly rozděleny do 8 binů. Kategorizace obrazových bodů probíhá následujícím způsobem: Kolik pixelů má hodnotu červené složky z daného intervalu, zelené z jiného a modré zase z jiného. Například červené z $\langle 0, 31 \rangle$, zároveň zelené z $\langle 32, 63 \rangle$ a modré z $\langle 64, 95 \rangle$. Reálný počet binů je tak $8^3 = 512$.

Histogram je následně normalizován, aby velikost rozlišení vstupního snímku nezkreslovala výsledky. Každá hodnota v histogramu je vydělena maximální možnou hodnotou, tím jsou všechny hodnoty přeškálovány do intervalu $\langle 0, 1 \rangle$. Pro extrakci histogramů a jejich následnou normalizaci byla použita knihovna OpenCV.

Spočítání podobnosti výřezů obrázků lze realizovat porovnáním jejich příznakových vektorů. Cílem je spočítat jejich vzájemnou vzdálenost. K tomu lze použít různé distanční metriky, například euklidovskou vzdálenost,

$$d(u, v) = \sqrt{\sum_{i=1}^{n} (u_i - v_i)^2} \tag{5.1}$$

manhattanskou vzdálenost,

$$d(u, v) = \sum_{i=1}^{n} |u_i - v_i| \tag{5.2}$$

nebo korelační vzdálenost,

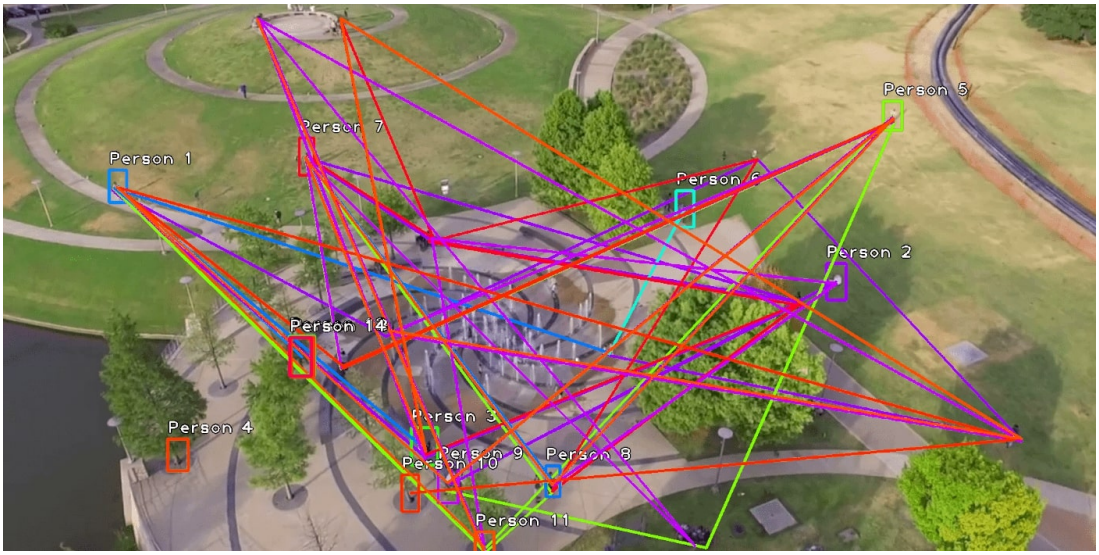
$$d(u, v) = \frac{\sum_{i=1}^{n} (u_i - \hat{u})(v_i - \hat{v})}{\sqrt{\sum_{i=1}^{n} (u_i - \hat{u}) \sum_{i=1}^{n} (v_i - \hat{v})}} \tag{5.3}$$

kde u, v jsou příznakové vektory a $\hat{u}$, $\hat{v}$ jsou průměrné hodnoty.

Experimentálně bylo zjištěno, že pro tento typ dat bude nejvhodnější korelační vzdálenost, kde byly rozdíly ve vzdálenostech mezi stejnými objekty napříč snímky vůči jiným objektům nejvýraznější.

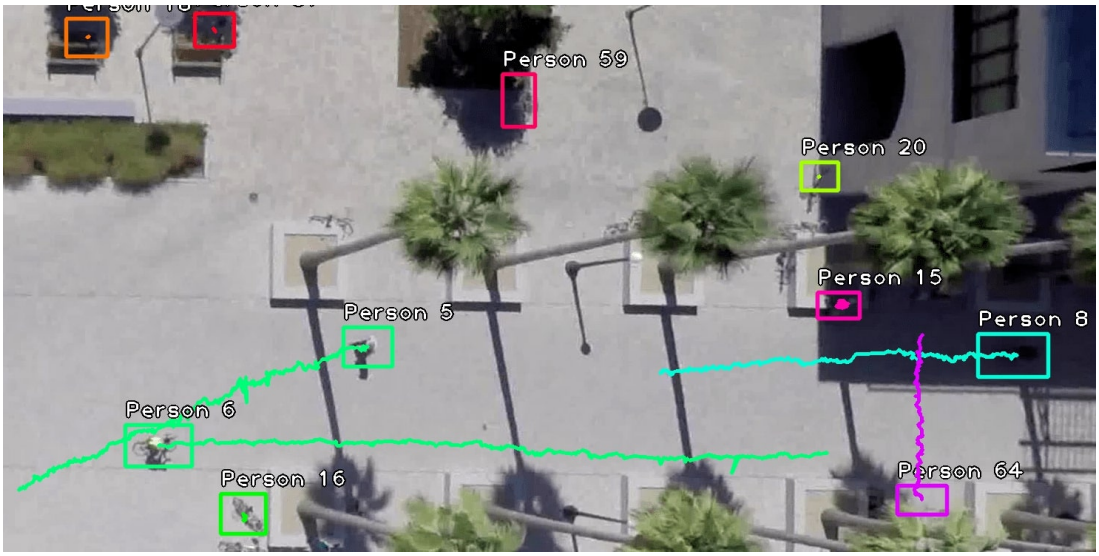
5.4 Experimenty s dílčími částmi algoritmu

Pro dokončení algoritmu bylo nutné provést několik experimentů, a tím zjistit, zda porovnání pouze příznakových vektorů je dostačující. Hned první experiment (obrázek 5.4) ukázal, že tento přístup není ideální. Pokud je ve snímku zachycen větší počet lidí, je pravděpodobné, že si jejich příznakové vektory mohou být velmi podobné, a díky tomu jako nejpodobnější může být vyhodnocen úplně jiný.



Obrázek 5.4: Vizualizace trajektorií chodců první verzí algoritmu. Chodci jsou porovnáváni pouze na základě příznakových vektorů.

Je nemožné, aby se člověk v rámci jednoho snímku videa přesunul o výraznou vzdálenost. Vzhledem k tomu byl algoritmus upraven, aby příznakový vektor osoby byl porovnáván pouze s příznakovými vektory osob v její blízkosti. To zajistilo, že detekce nejsou spojovány křížem krážem napříč snímkem. Z tohoto přístupu vyplývá problém v následujícím případě: Člověk projde pod nějakou překážkou, a tím se kameře schová. Jeho další detekce, ačkoliv o spoustu snímků později, může tak být velmi vzdálená od té předchozí. Stejný případ může nastat, pokud člověk záběr opustí a poté se do něho vrátí v jiném místě.



Obrázek 5.5: Vizualizace trajektorií chodců druhou verzí algoritmu. Při porovnávání příznakových vektorů chodců je brána v potaz i jejich reálná vzdálenost ve snímku.

Výsledek nyní vypadá podstatně lépe. Avšak ve snímku jsou vyznačeny statické objekty, které jistě nejsou lidé. To je dáno chybou neuronové sítě, která v některém snímku vrátila jako osobu objekt, který osoba není. Druhý problém je vidět na osobě 64. Tento člověk se ve

videu objevil až později, nebyl na prvním snímku. Proto je zakreslen ohraničující obdélník kolem prázdného místa.

Dá se předpokládat, že ve většině záznamů se lidé budou pohybovat, zvlášť bude-li videozáznam delší. Pokud se člověk pohybovat nebude, nemá pro něho vykreslení trajektorie pohybu smysl. Algoritmus byl upraven tak, aby pro statické objekty, které se v rámci celého pořízeného záběru nepohnou, nebylo vykresleno nic. Tato úprava vyřešila problém občasné chybné detekce neuronovou sítí nějakého statického objektu jako člověka.

Druhá úprava se pak postarala o problém prázdných obdélníků pro osoby, které se v záběru objevily až později. Při jejich prvním výskytu je obraz s nimi vyříznut a překopírován do panoramatického snímku.

Obrázek 5.6: Vizualizace trajektorií chodců třetí verzí algoritmu. Nyní jsou ve snímku vyznačeni pouze lidé a trajektorie, kterými bude směrován jejich pohyb.

Výsledné algoritmy pro zpracování objektů a vykreslení trajektorií jsou zachyceny na následujících výpisech.

```
def get_panorama_image(image, known_objects):
    for obj in known_objects:
        if is_static_object(obj):
            continue
        if not detected_in_first_frame(obj):
            insert_cutout_into_image(image, obj.cutout)
        draw_bounding_box(image, obj)
        draw_caption(image, obj)
        previous_point = obj.points.pop(0)
        for point in obj.points:
            draw_line(image, prev_point, point)
            previous_point = point
    return image
```

Výpis 5.2: Výsledná funkce v pseudokódu Pythonu pro vizualizaci trajektorií.

```
def process_objects(known_objects, frame, objects):
    for obj in objects:
        if frame == 0:
            obj.compute_histogram()
            obj.detected_in_first_frame = True
            known_objects.append(obj)
        else:
            obj.compute_histogram()
            most_similar_obj = None
            shortest_hist_distance = MAX_INT
            for known_obj in known_objects:
                euclidean_distance = compute_euclidean_distance(obj, known_obj)
                if euclidean_distance < EUCLIDEAN_DISTANCE_TRESHOLD:
                    hist_distance = compute_correlation_distance(obj, known_obj)
                    if hist_distance < shortest_hist_distance:
                        shortest_hist_distance = hist_distance
                        most_similar_obj = known_obj
            if shortest_hist_distance < HIST_SIMILARITY_TRESHOLD:
                most_similar_obj.compute_hist(obj)
                most_similar_obj.add_point(obj)
            else:
                known_objects.append(obj)
```

Výpis 5.3: Výsledná funkce v pseudokódu Pythonu pro zpracování objektů detekovaných neuronovou sítí.

5.5 Implementace demonstrační aplikace

Pro kompletní otestování funkcionality a provedení experimentů bylo nutné naimplementovat demonstrační aplikaci. Vzhledem k požadavku na multiplatformnost aplikace byla brána v potaz složitost přenosu aplikace na různé operační systémy a s tím spojené komplikace, se kterými by se uživatel musel vypořádávat při instalaci potřebných závislostí.

Samotná aplikace je naprogramovaná v jazyce Python v aktuální verzi 3.7.3. Její vývoj probíhal ve vývojovém prostředí (IDE) PyCharm od společnosti JetBrains. JetBrains poskytují licence ke svému software ve verzi Ultimate pro studenty a akademické účely zdarma.

Grafické uživatelské rozhraní bylo realizováno pomocí knihovny PyQt[1] ve verzi 4.8.7. PyQt je odnož frameworku Qt pro implementaci uživatelských rozhraní v Pythonu. Qt je široce používaný multiplatformní framework, ve kterém lze vyvíjet aplikace v odlišných programovacích jazycích pro různé platformy.

Pro otevírání, ukládání a manipulaci s obrázky byla využita knihovna PIL (Python Imaging Library). Konkrétně její fork Pillow[2] verze 6.0.0, který od Pythonu verze 3 nahradil původní implementaci, jelikož její vývoj byl ukončen.

Použitá implementace detekční sítě RetinaNet, která byla blíže popsána v sekci 4.1, vyžaduje knihovnu Keras[3] verze 2.2.4 nebo novější. Ta je dostupná buď samostatně nebo přímo jako součást Tensorflow[4] od verze 1.13.

[1] https://www.riverbankcomputing.com/software/pyqt
[2] https://pillow.readthedocs.io/en/stable/
[3] https://keras.io/
[4] https://www.tensorflow.org/

Pro načítání, zpracování obrazových dat, manipulaci s video soubory, počítání a normalizací barevných histogramů a vizualizaci ohraničujících boxů či trajektorií byla použita knihovna OpenCV[5] verze 4.1.0.

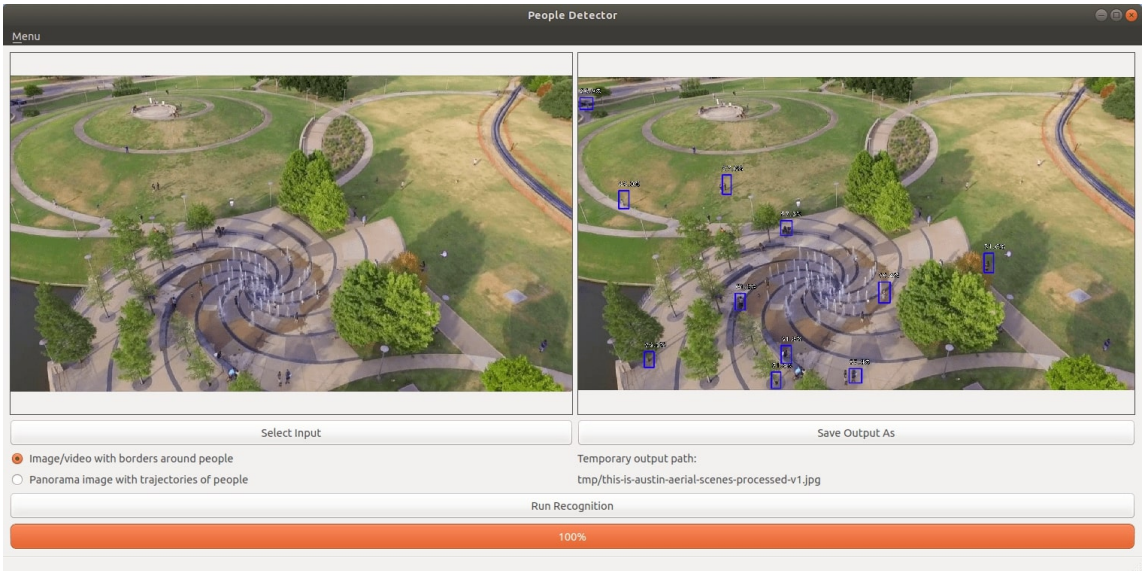
Počítání distančních metrik mezi vektory příznaků bylo realizováno pomocí knihovny SciPy[6] ve verzi 1.2.1.

Výše zmíněné knihovny pracují s reprezentací obrazových dat jako s multidimenzionálním polem. Konkrétně s polem NumPy array a operacemi implementovanými nad ním v knihovně NumPy[7]. Použita byla aktuální verze 1.16.3.

Veškeré použité nástroje jsou platformně nezávislé. Aplikaci by tak neměl být problém spustit na většině operačních systémů. Potřebné Python knihovny je možné doinstalovat pomocí správce balíků (*package manager*) Pip z oficiálního repozitáře PyPI (Python Package Index). Výjimku tvoří pouze knihovna PyQt, která zastává pouze funkci wrapperu nad C++ implementací Qt. Python knihovna tak poskytuje pouze pohodlné rozhraní pro práci s ní. Z toho důvodu je třeba ji instalovat přes správce balíků operačního systému. Například přes DNF v případě linuxových distribucí založených na RPM nebo přes APT v případě distribucí založených na Debian. Pro účely této práce byla aplikace otestována na operačním systému Fedora 29, Ubuntu 18.04 a Windows 10 (Verze 1809, OS Build 17763.437).

Grafické uživatelské rozhraní

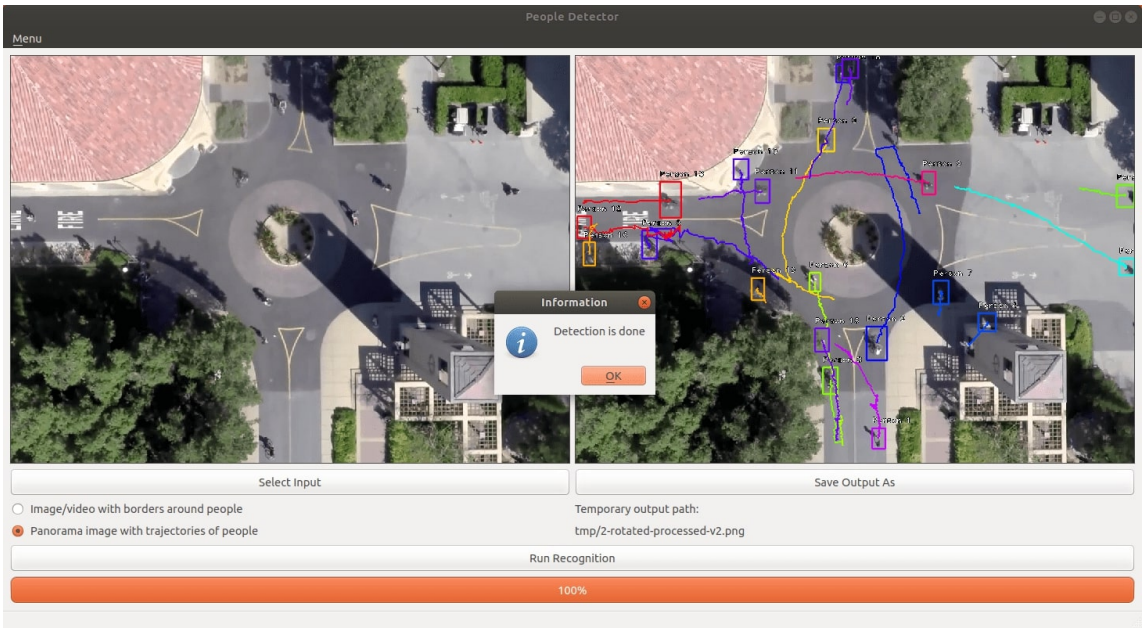
Grafické uživatelské rozhraní poskytuje tři tlačítka (*push button*) pro základní ovládání aplikace. Otevření vstupního souboru, spuštění procesu detekce a uložení výstupu. Všechny tyto akce jsou přístupné i z hlavního menu (*menu bar*) nebo přes klávesové zkratky. Posledním ovládacím prvkem je přepínač (*radio button*) mezi typy výstupů.



Obrázek 5.7: Ukázka využití aplikace přes jeho grafické uživatelské rozhraní. Jako vstup je zvolen snímek a typ výstupu snímek s ohraničujícími obdélníky kolem lidí. Uživatel si nyní může výstup uložit, zvolit další vstup nebo jenom změnit typ výstupu a spustit rozpoznávání znovu.

[5] https://opencv.org/
[6] https://www.scipy.org/
[7] https://www.numpy.org/

Aplikace také obsahuje štítky (*label*) pro vizualizaci vstupních a výstupních dat. V případě, že je na vstupu vybráno video, jsou štítky při každém zpracovaném snímku aktualizovány. Uživatel tak může v průběhu procesu zpracování pozorovat, jak je aplikace s detekcí úspěšná.



Obrázek 5.8: Další ukázka použití aplikace. Tentokrát je jako vstup zvoleno video a výstup panoramatická mapa.

Posledním prvkem grafického rozhraní je ukazatel průběhu zpracování vstupu (*progress bar*). Předem je spočítána velikost kroku, o který se ukazatel bude posunovat. Ta je odvozena od počtu snímků ve videu. Po každém zpracovaném snímku se pak ukazatel o danou velikost kroku posune. Při zpracování fotky nabývá ukazatel pouze hodnot 0 % a 100 %.

Kapitola 6

Experimenty

Vyhodnocení přesnosti detektoru bylo provedeno metrikou Averige Precision (AP). Jedná se o způsob, který se používá na soutěžích Pascal VOC Challenge. Rozhodnutí, jestli detekce byla správná či nikoliv, se uskutečňuje pomocí metody Intersection over Union (IoU). Je spočítán podíl průniku a sjednocení predikované oblasti a oblasti skutečně anotované. Oblast predikovaná je označena A a oblast skutečně anotovaná B. Výpočet pak vypadá následovně.

$$IoU(A, B) = \frac{A \cap B}{A \cup B} \tag{6.1}$$

Pokud podíl vyjde vyšší než 0,5 je detekce považována za úspěšnou a označena jako skutečně pozitivní (*true positive*). V opačném případě je označena za falešně pozitivní (*false positive*). Případ, kdy není anotovaný objekt nalezen vůbec, je označen jako falešně negativní (*false negative*). Z těchto ukazatelů lze vypočítat metriky *precision* a *recall*. AP potom shrnuje tvar $precision\,/\,recall$ křivky [4].

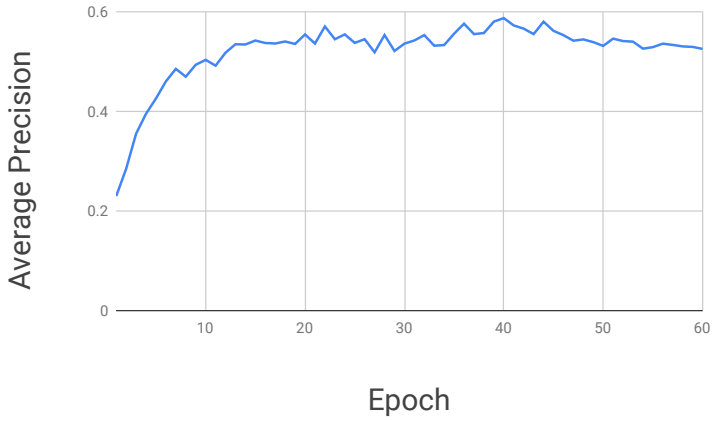
$$precision = \frac{true\,positives}{true\,positives + false\,positives} \tag{6.2}$$

$$recall = \frac{true\,positives}{true\,positives + false\,negatives} \tag{6.3}$$

Zhodnocení úspěšnosti reidentifikace osob a správnosti vykreslených trajektorií bylo provedeno subjektivně po otestování na několika demonstračních videích. Bylo vyhodnoceno, s čím má algoritmus problémy a kde naopak funguje.

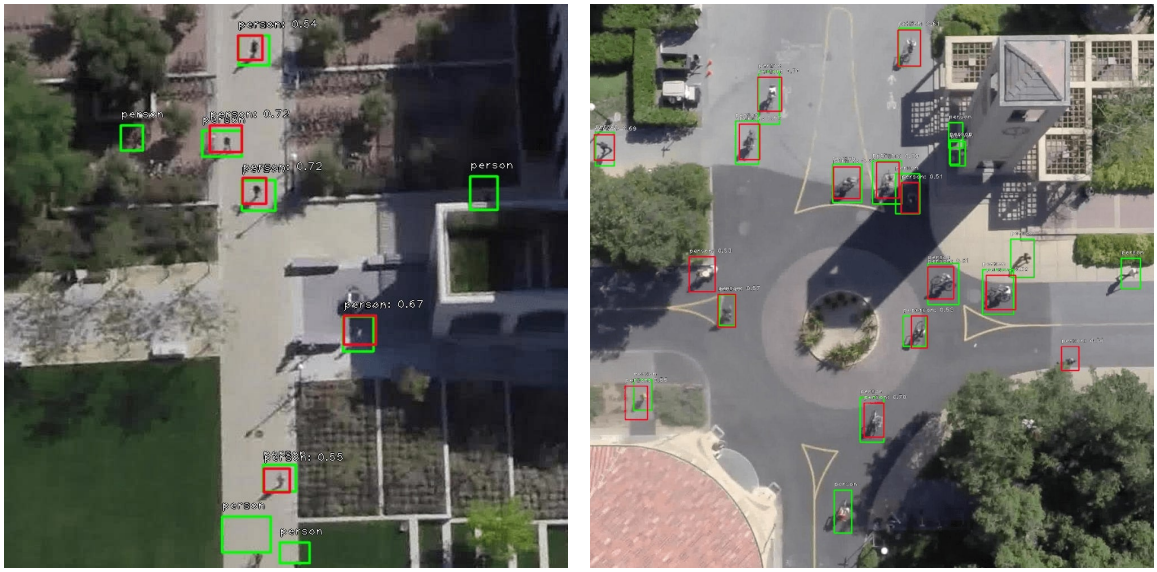
6.1 Úspěšnost a rychlost detektoru

Vyhodnocení přesnosti detektoru bylo měřeno na validační části datasetu SDD (sekce 4.2). Přesnost byla měřena na modelech z finálního třetího trénování, kde se dosáhlo nejnižší validační chyby. Nejvyšší přesnosti AP dosáhl model po 40. trénovací epoše a to sice 58,6 %.

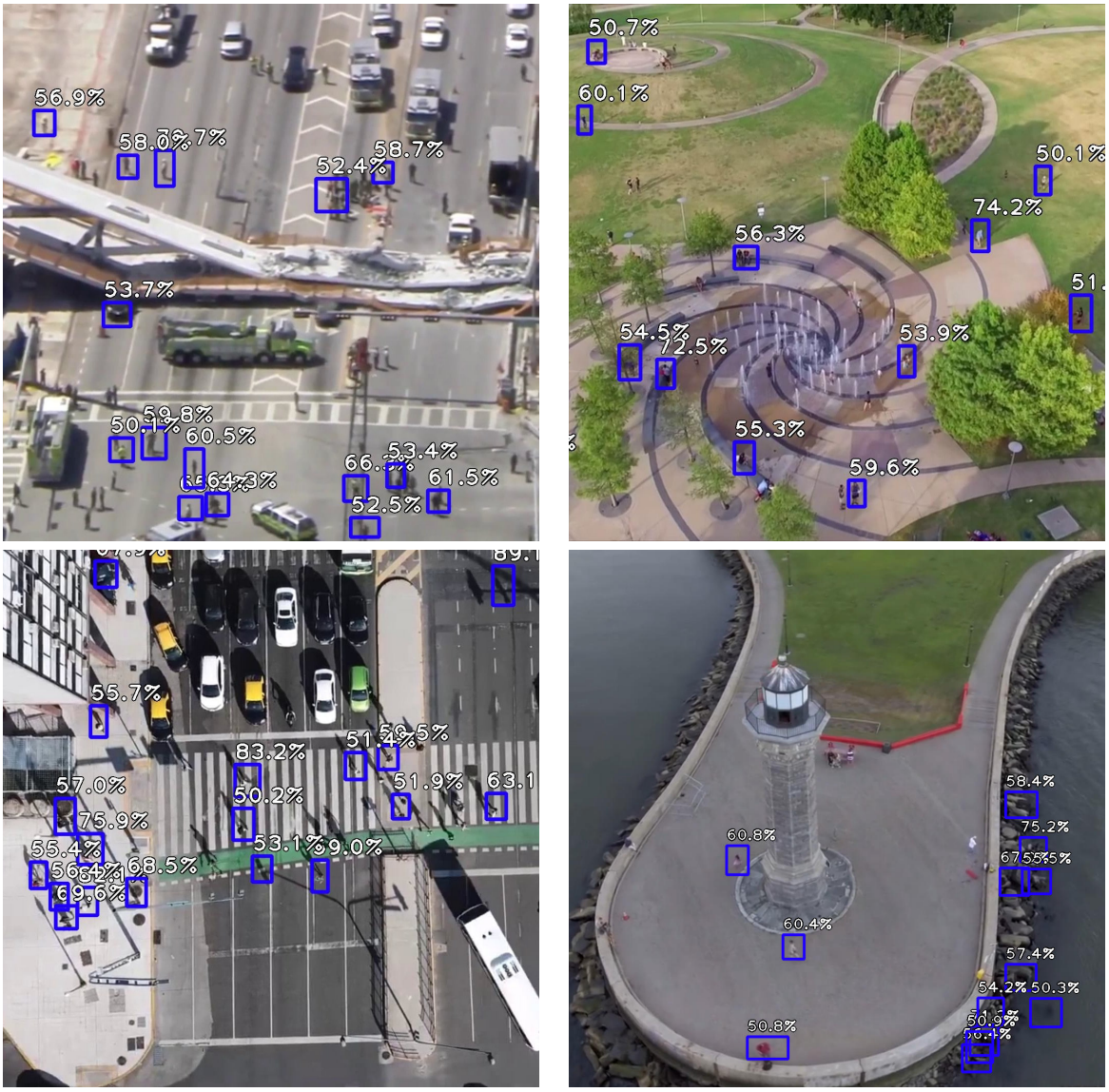


Obrázek 6.1: Vývoj přesnosti AP v závislosti na množství odtrénovaných epoch.

Měření rychlosti modelu probíhalo na operačním systému Ubuntu 18.04 s interpretem Pythonu verze 3.7.3 a knihovnou Tensorflow verze 1.13. Procesor Intel Core i7 8565U Whiskey Lake potřeboval na zpracování jednoho snímku kolem 3 sekund. Zpracování obrázku s akcelerací pomocí grafické karty nVIDIA GeForce GTX 1070 trvalo zhruba 0,1 vteřiny.



Obrázek 6.2: Ukázka z vyhodnocování přesnosti na validačních datech. Zeleně jsou vyznačeni skutečně anotovaní lidé, červeně rozpoznaní. Je zřejmé, že někde nejsou anotace vyznačeny přesně, jak bylo popsáno v sekci 4.3. To kazí celkovou naměřenou přesnost.



Obrázek 6.3: Ukázky detekce lidí na testovacích obrázcích, které nebyly součástí původního datasetu. Je zřejmé, že detektor všechny osoby nedetekuje a zároveň ho některé objekty dokáží zmást. Obrázky jsou převzaty z videí na Youtube.[1]

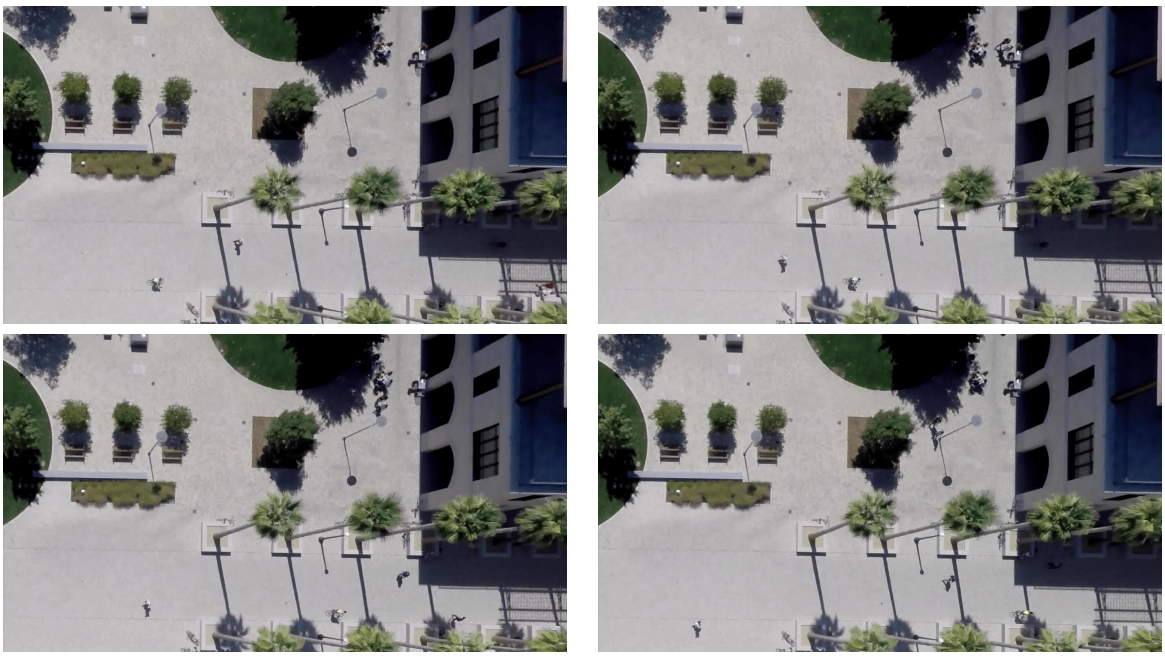
6.2 Úspěšnost reidentifikace lidí

Vyhodnocení úspěšnosti reidentifikace lidí a zakreslení správnosti jejich trajektorií probíhalo na záběrech z validační části použitého datasetu a na pořízených záznamech z dronu DJI Spark z výšky 35 m. Z datasetu byly vybrány takové záběry, kde je možnost reidentifikace aspoň trochu možná, tedy spíše záběry v lepší kvalitě a pořízené z nižší výšky. Dataset jinak obsahuje záběry z velmi vysoké výšky, případně v nízké kvalitě, kde je v podstatě

[1] https://youtu.be/zJq5_7Au13o, https://youtu.be/7ns6fFhahbw, https://youtu.be/O9MOb65LVL0, https://youtu.be/HiZXABMNCUY

člověk viděn pouze jako tmavá čmouha. Tam pochopitelně není možné od sebe jednotlivé lidi rozlišit.

Scéna Coupa



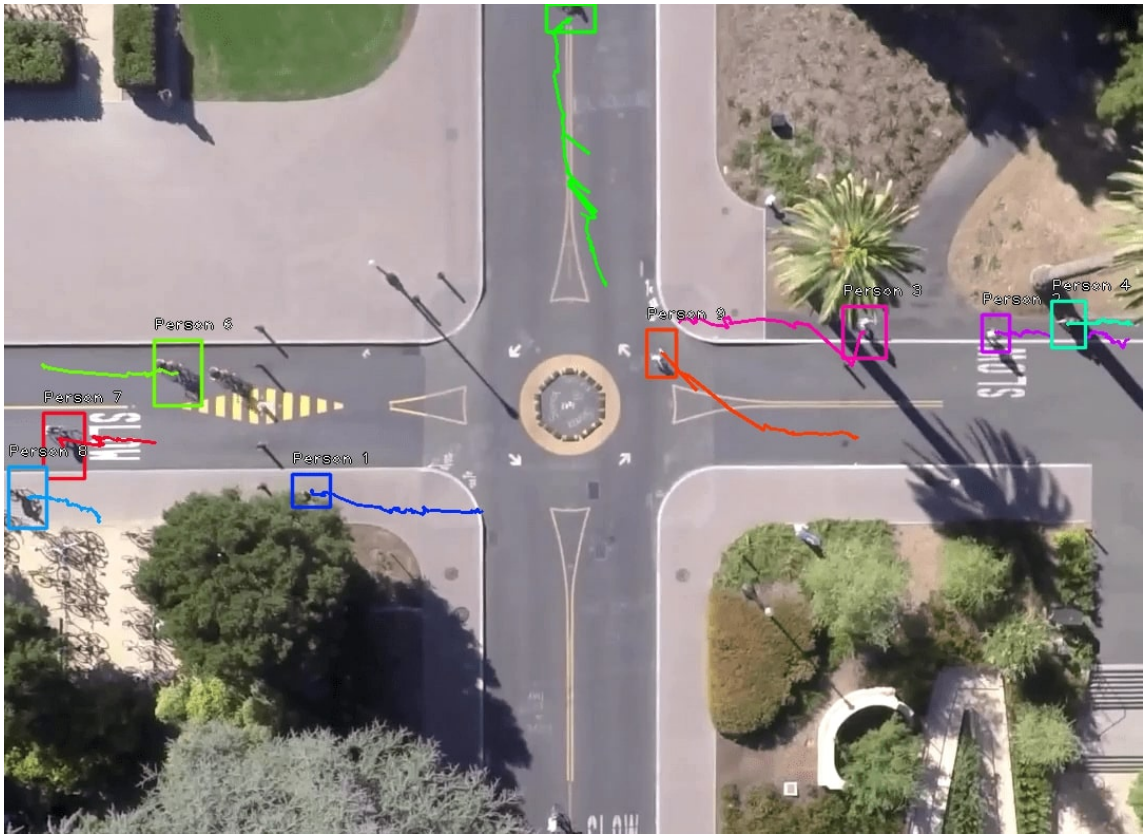
Obrázek 6.4: Vstupní 10 vteřinové video ze scény Coupa. V pravé horní části obrazu se nachází po celou dobu několik sedících osob. V průběhu kolem nich projde další dvojice osob. V dolní části se pak pohybuje asi čtveřice osob, kdy jedna z nich se v záběru objeví až později. V pravém dolním rohu odchází ze záběru dvojice osob.

Obrázek 6.5: Výsledná mapa s trajektoriemi osob ze scény Coupa.

Pravý horní roh způsobil velký zmatek. Dvojice chodců, která procházela kolem skupinky sedících lidí, nebyla dostatečně rozlišena. Procházeli velmi blízko sebe a díky tomu, že není nijak vyřešeno oříznutí pozadí chodce a histogram je počítán pro celou výseč, je výsledek takto špatný. Vektory příznaků si jsou příliš podobné. Dolní část obrazu dopadla o poznání lépe. Všem 4 osobám byly trajektorie vykresleny správně. Osoby se potkaly, ale pomocí vektoru příznaků od sebe byly správně rozpoznány. Osoba 7 navíc vyšla ze stínu, algoritmus se přesto zachoval správně. Další problém nastal s dvojicí osob v pravém dolním rohu, která byla v obraze pouze chvilku a procházela přes částečný stín. Osoby nebyly úspěšně reidentifikovány, jejich příznakové vektory se napříč snímky příliš lišily.

Scéna Little

Obrázek 6.6: Vstupní 7 vteřinové video ze scény Little. V obrázku se pohybují chodci a cyklisté. Někteří procházejí skrz stíny, pod lehkým zákrytem stromů nebo do záběru vstupují až v průběhu záznamu.

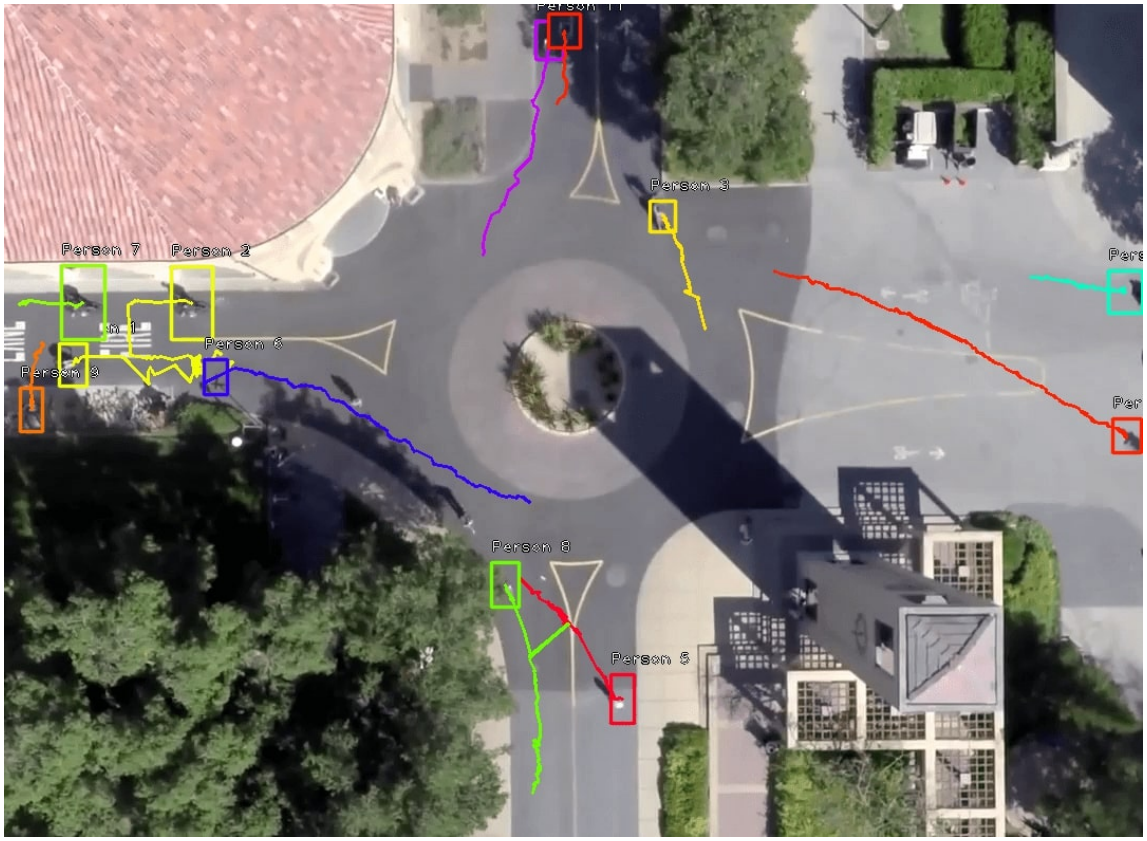


Obrázek 6.7: Výsledná mapa s trajektoriemi osob ze scény Little.

Cyklistu 6 síť nerozpoznala ihned, ale až po několika snímcích, proto je jeho výřez vidět dvakrát. Výřez více vpravo je pozice cyklisty v prvním snímku. Označený výřez vlevo je moment, kdy byl sítí poprvé rozpoznán. Osoby 7, 8 se v záběru objevily až později a byly správně detekovány a reidentifikovány. Člověk 1 je na počátku schovaný pod stromem, v záběru se ukáže až v průběhu videa a je správně rozpoznán. Osoby 2, 4 projdou stínem, přesto jsou správně reidentifikovány. Člověk 3 projde stínem, částečně pod zákrytem stromu a je opět správně rozpoznán. Cyklista 9 se uprostřed křižovatky ztratí, síť ho přestane na několik snímků detekovat. Po jeho zpětném rozpoznání je už považován za jinou osobu, protože se mezitím dostal příliš daleko.

Scéna Death Circle

Obrázek 6.8: Vstupní 5 vteřinové video ze scény Death Circle. Záznam je podobného typu jako ze scény Little. V záběrech se objevují cyklisté a chodci, procházejí skrz stíny, případně pod zákryty stromů.



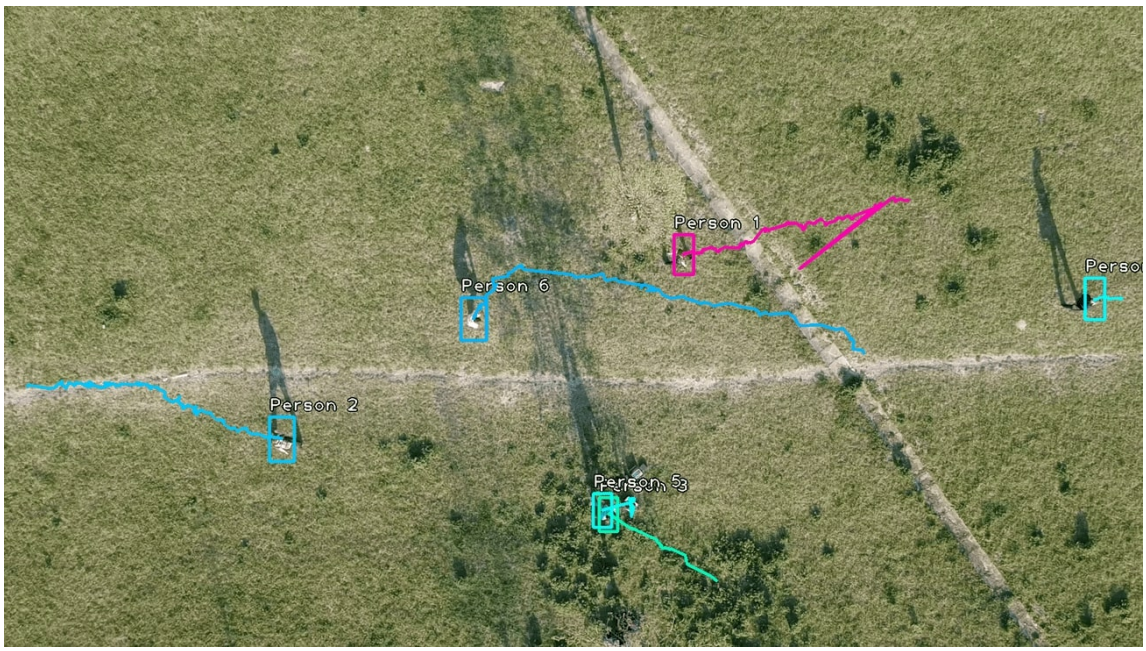
Obrázek 6.9: Výsledná mapa s trajektoriemi osob ze scény Death Circle.

Ve spodní části videa jdou proti sobě lidé označeni čísly 5 a 8. Osoba 8 se na krátký okamžik detektoru ztratí. Jako nejbližší a nejpodobnější detekce je pak vyhodnocena osoba 5, proto ta zelená linie k červené. V pravé horní části obrazu se vyskytuje 5 osob, jejichž detekce i predikce trajektorií je bezchybná. V levé části jede mimo záznam cyklista 2 a proti němu cyklista 1, později identifikován jako cyklista 6. V momentě, kdy si jsou nejblíže, nastává pro cyklistu 2 zmatek a jeho trajektorie uhýbá směrem k cyklistovi 1. O několik snímků později je cyklista 2 identifikován jako cyklista 7 a dále je jeho pohyb zaznamenán správně. Cyklista 1 je od doby, co není zaměňován s cyklistou 2 identifikován jako cyklista 6 a jeho trajektorie je také vizualizována správně. V této části se nachází ještě osoba 9, pro kterou je detekce korektní.

První scéna z natáčení dronem

Pro další experimentování a vyhodnocování úspěšnosti byly pořízeny záběry dronem DJI Spark. Skupinka osob se náhodně pohybovala po natáčeném prostoru. Vzniklo několik záběrů, které byly následně sestříhány a využity pro další otestování algoritmu. Hranice, pro vyhodnocení detekce neuronovou sítí jako úspěšné, byla snížena z výchozí hodnoty 0,5 na 0,4.

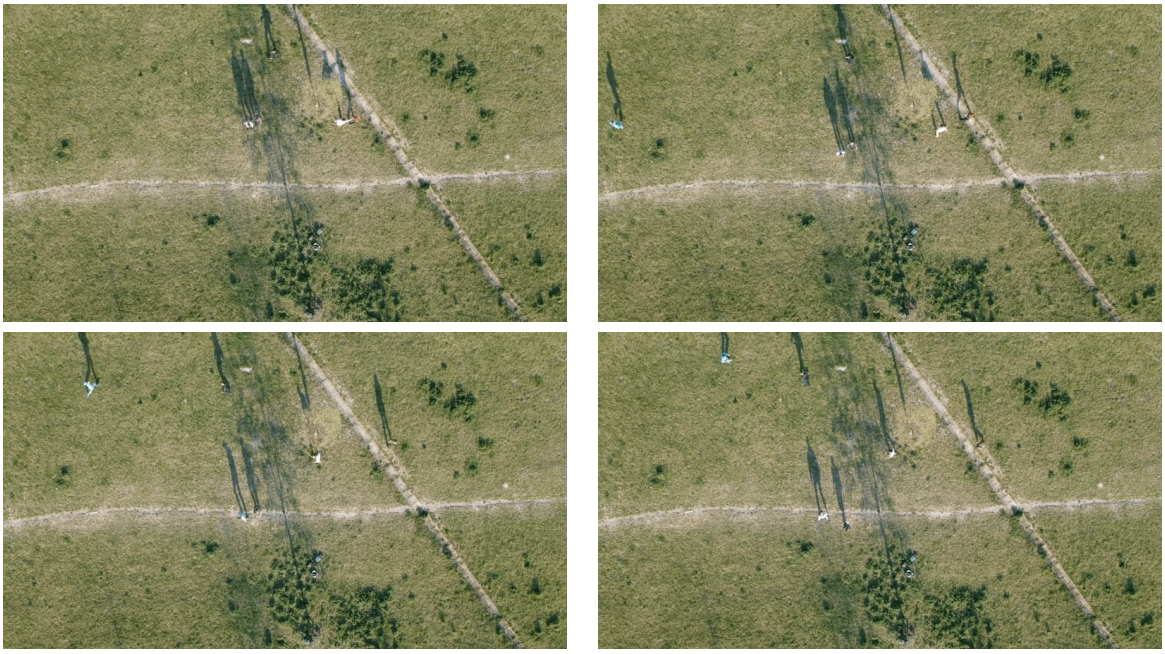
Obrázek 6.10: Vstupní 6 vteřinové video. První záznam pořízený dronem. Na počátku je v obraze 5 lidí a postupně všichni odcházejí mimo záběr.



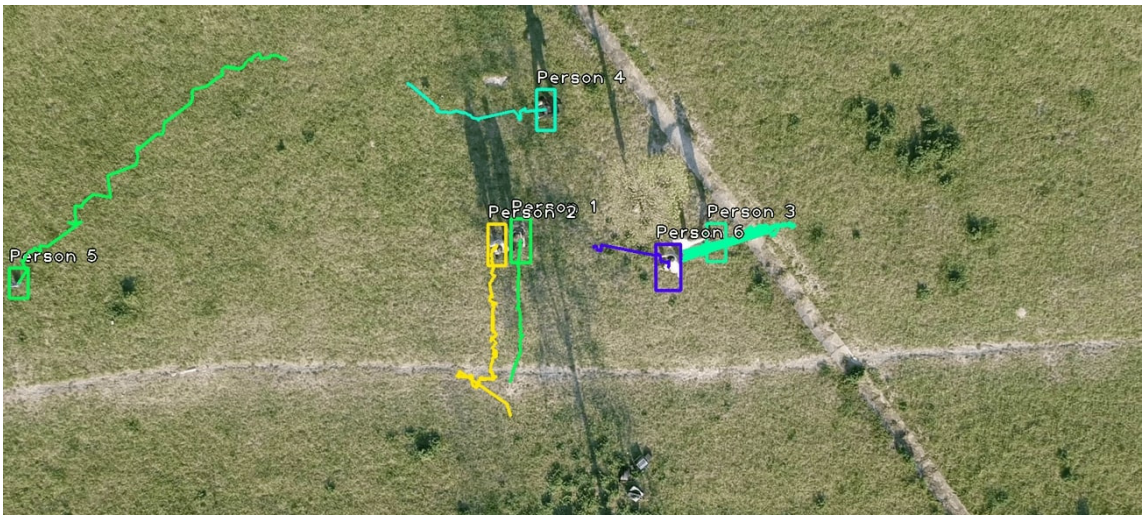
Obrázek 6.11: Výsledná mapa s trajektoriemi osob z první scény.

Všechny osoby byly v průběhu záznamu detekovány a jejich trajektorie zakresleny. Osoba 1 byla ke konci mylně spojena s jiným bodem. Pravděpodobně síť osobu na moment ztratila a zároveň rozpoznala jiný bod jako člověka o kus vedle. Osoba 5 a 3 jsou na počátku detekováni jako dvě osoby, jedná se pouze o jednu. Zbytek reidentifikací a vykreslení trajektorií je správné.

Druhá scéna z natáčení dronem



Obrázek 6.12: Vstupní 5 vteřinové video. Druhý záznam pořízený dronem. 6 osob pohybujících se v záběru.



Obrázek 6.13: Výsledná mapa s trajektoriemi osob z druhé scény.

Lidé 1 a 2 jdou vedle sebe, jsou od sebe správně rozlišeni na základě vektoru příznaků a jejich trajektorie jsou korektně vykresleny. Osoby 6 a 3 běží od sebe. 6 je rozpoznána až později, nejdříve je několikrát mylně zaměněna s osobou 3. Výsledkem je několik dlouhých, rovných linií. Lidé 5 a 4 jsou reidentifikovány po celou dobu správně a jejich trajektorie jsou korektní.

Kapitola 7

Závěr

Práce se zabývá detekcí lidí v obrazovém materiálu pořízeném dronem. Detekované osoby jsou od sebe vzájemně rozlišeny a jejich pohyb je monitorován v průběhu celého videozáznamu. Po zpracování celého videa jsou trajektorie jednotlivých osob vizualizovány.

K rozpoznání osob byla využita detekční síť RetinaNet. Předtrénované modely nebyly schopny detekovat lidi z dostatečné výšky. Z toho důvodu byl natrénován vlastní model na datasetu Stanford Drone Dataset. Proběhly celkem 3 trénování, kdy bylo s datasetem manipulováno na základě průběžných výsledků. Dále byl implementován algoritmus identifikace chodců, podle kterého byla každé osobě vykreslena trajektorie pohybu. Detekované objekty, které se v průběhu videa nepohybovaly, jsou považovány za chybu detektoru a nejsou vizualizovány. Trajektorie pohybů osob jsou vykresleny do prvního snímku videa. Pro demonstraci funkcionality byla implementována aplikace s grafickým rozhraním.

Dosáhnutá přesnost detektoru 58,6 % na validační části datasetu je spíše slabá. Tato nízká hodnota je způsobena jednak složitostí problému, kdy lidé z velké výšky mohou být snadno zaměnitelní za jiné objekty, tak také nepřesnými anotacemi použitého datasetu, a to i přes jeho ruční protřízení.

Ani identifikace chodců pomocí porovnávání jejich příznakových vektorů založených na barevných histogramech nefunguje perfektně. Nijak není realizováno oříznutí pozadí kolem chodce. Histogramy tak jsou počítány pro celý segment vrácený detektorem. To znamená, že histogram se může výrazně změnit pokud chodec změní pozadí. Proto si algoritmus identifikace pomáhá využitím vzájemné fyzické vzdálenosti detekovaných osob.

Výsledné trajektorie jsou zaznamenávány do prvního snímku videa. Z toho vyplývá omezující podmínka pro korektní vykreslování trajektorií pohybu: pořízený záznam musí být statický, dron se v průběhu natáčení nemůže hýbat.

Potenciální budoucí rozšíření může spočívat v implementaci nějaké formy *image-stitching* algoritmu. Pomocí něho by jednotlivé snímky videozáznamu mohly být pospojovány a vznikla by panoramatická mapa, která by zaznamenávala celou zabranou oblast ve videu. Pro správné zakreslení trajektorií chodců by následně bylo nutné implementovat mechanismus, pomocí kterého by byl každý jednotlivý snímek v panoramatu detekován. Celý program by pak fungoval i pro videozáznamy, kde se dron pohybuje.

Dosáhnutí vyšší přesnosti detektoru by bylo možné za použití lépe anotovaného datasetu. Oblast neuronových sítí je velmi dynamicky se rozvíjející obor. Téměř každý rok jsou představovány dokonalejší a přesnější architektury. I z tohoto důvodu nebude problém přesnost zde představeného detektoru překonat.

Literatura

[1] Andrýsková, J.; Horký, L.; Procházka, D.: *Digitální fotografie, eLearningová opora k předmětu*. 2008, [Online; navštíveno 7. 2. 2018].
URL https://is.mendelu.cz/eknihovna/opory/index.pl?opora=1083

[2] Dalal, N.; Triggs, B.: Histograms of Oriented Gradients for Human Detection. In *Computer Society Conference on Computer Vision and Pattern Recognition, San Diego, CA, USA*, June 2005, ISSN 1063-6919, s. 886–893.
URL https://lear.inrialpes.fr/people/triggs/pubs/Dalal-cvpr05.pdf

[3] de Boer, P.-T.; Kroese, D.; Mannor, S.; aj.: A Tutorial on the Cross-Entropy Method. *Annals of operations research*, ročník 134, č. 1, 2005: s. 19–67, ISSN 0254-5330.
URL http://web.mit.edu/6.454/www/www_fall_2003/gew/CEtutorial.pdf

[4] Everingham, M.; Gool, L.; Williams, C. K.; aj.: The Pascal Visual Object Classes (VOC) Challenge. *Int. J. Comput. Vision*, Červen 2010: s. 303–338, ISSN 0920-5691.
URL http://host.robots.ox.ac.uk/pascal/VOC/pubs/everingham10.pdf

[5] Fumo, D.: *A Gentle Introduction To Neural Networks Series*. Srpen 2017, [Online; navštíveno 21.12.2018].
URL https://towardsdatascience.com/a-gentle-introduction-to-neural-networks-series-part-1-2b90b87795bc

[6] Girshick, R. B.: Fast R-CNN. *CoRR*, ročník abs/1504.08083, 2015.
URL http://arxiv.org/pdf/1504.08083.pdf

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

[8] Haykin, S.: *Neural Networks: A Comprehensive Foundation*. Upper Saddle River, NJ, USA: Prentice Hall PTR, druhé vydání, 1998, ISBN 0132733501.

[9] He, K.; Gkioxari, G.; Dollár, P.; aj.: Mask R-CNN. *CoRR*, ročník abs/1703.06870, 2017.
URL http://arxiv.org/pdf/1703.06870.pdf

[10] He, K.; Zhang, X.; Ren, S.; aj.: Deep Residual Learning for Image Recognition. *CoRR*, ročník abs/1512.03385, 2015.
URL http://arxiv.org/pdf/1512.03385.pdf

[11] Hinton, G. E.; Osindero, S.; Teh, Y.-W.: A Fast Learning Algorithm for Deep Belief Nets. In *Neural Computation*, MIT Press, Červenec 2006, s. 1527–1554.
URL https://www.cs.toronto.edu/~hinton/absps/fastnc.pdf

[12] Hui, J.: Object detection: speed and accuracy comparison. Březen 2018, [Online; navštíveno 5.5.2019].
URL https://medium.com/@jonathan_hui/object-detection-speed-and-accuracy-comparison-faster-r-cnn-r-fcn-ssd-and-yolo-5425656ae359

[13] Joshi, D.: *Here are the world's largest drone companies and manufacturers to watch and invest in*. Červenec 2017, [Online; navštíveno 20.12.2018].
URL https://www.businessinsider.com/top-drone-manufacturers-companies-invest-stocks-2017-07

[14] Kanter, J.: Google just beat Amazon to launching one of the first drone delivery services. Duben 2019, [Online; navštíveno 10.5.2019].
URL https://www.businessinsider.com/google-beats-amazon-to-launching-drone-delivery-service-in-australia-2019-4

[15] Krizhevsky, A.; Sutskever, I.; Hinton, G. E.: ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems 25*, Curran Associates, Inc., 2012, s. 1097–1105.
URL https://www.cs.toronto.edu/~kriz/imagenet_classification_with_deep_convolutional.pdf

[16] Kroh, K.; House, A.: What are aerial drones good for? 2018, [Online; navštíveno 6.5.2019].
URL https://www.dummies.com/consumer-electronics/drones/what-are-aerial-drones-used-for/

[17] LeCun, Y.; Bengio, Y.; Hinton, G. E.: Deep learning. *Nature*, ročník 521, č. 7553, 2015: s. 436–444, doi:10.1038/nature14539.
URL http://graveleylab.cam.uchc.edu/WebData/mduff/MEDS_6498_SPRING_2016/deep_learning_nature_2015.pdf

[18] LeCun, Y.; Boser, B. E.; Denker, J. S.; aj.: Handwritten Digit Recognition with a Back-Propagation Network. In *Advances in Neural Information Processing Systems*, Morgan Kaufmann, 1990, s. 396–404.
URL http://yann.lecun.com/exdb/publis/pdf/lecun-90c.pdf

[19] Lehmann, E. L.; Casella, G.: *Theory of Point Estimation*. New York, NY, USA: Springer-Verlag, druhé vydání, 1998.

[20] Lin, T.; Dollár, P.; Girshick, R. B.; aj.: Feature Pyramid Networks for Object Detection. *CoRR*, ročník abs/1612.03144, 2016.
URL http://arxiv.org/pdf/1612.03144.pdf

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

[22] Lu, C.; Tang, X.: Surpassing Human-Level Face Verification Performance on LFW with GaussianFace. *CoRR*, ročník abs/1404.3840, 2014.
URL http://arxiv.org/pdf/1404.3840.pdf

[23] Mehrotra, K.; Mohan, C.; Ranka Preface, S.: *Elements of Artificial Neural Nets*. Cambridge, MA, USA: MIT Press, 01 1997, ISBN 0-262-13328-8.

[24] Nvidia Corporation: CUDA Toolkit Documentation. Březen 2019, [Online; navštíveno 10.5.2019].
URL https://docs.nvidia.com/cuda/

[25] Parrot AR.Drone: PARROT BEBOP 2. 2016, [Online; navštíveno 10.5.2019].
URL https://www.parrot.com/eu/drones/parrot-bebop-2

[26] Parthasarathy, D.: A Brief History of CNNs in Image Segmentation: From R-CNN to Mask R-CNN. Duben 2017, [Online; navštíveno 10.5.2019].
URL https://blog.athelas.com/a-brief-history-of-cnns-in-image-segmentation-from-r-cnn-to-mask-r-cnn-34ea83205de4

[27] Pascanu, R.; Mikolov, T.; Bengio, Y.: Understanding the exploding gradient problem. *CoRR*, ročník abs/1211.5063, 2012.
URL http://arxiv.org/pdf/1211.5063.pdf

[28] 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.
URL http://arxiv.org/pdf/1506.01497.pdf

[29] Robicquet, A.; Sadeghian, A.; Alahi, A.; aj.: Learning Social Etiquette: Human Trajectory Understanding In Crowded Scenes. In *Computer Vision – ECCV 2016*, Cham: Springer International Publishing, 2016, s. 549–565.
URL https://infoscience.epfl.ch/record/230262/files/ECCV16social.pdf

[30] Scherer, D.; Müller, A.; Behnke, S.: Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition. In *Proceedings of the 20th International Conference on Artificial Neural Networks: Part III*, ICANN'10, Berlin, Heidelberg: Springer-Verlag, 2010, s. 92–101.
URL http://ais.uni-bonn.de/papers/icann2010_maxpool.pdf

[31] Schmidhuber, J.: Deep Learning in Neural Networks: An Overview. *CoRR*, ročník abs/1404.7828, 2014.
URL http://arxiv.org/pdf/1404.7828.pdf

[32] scikit-image team: Histogram of Oriented Gradients. 2019, [Online; navštíveno 10.5.2019].
URL https://scikit-image.org/docs/0.11.x/auto_examples/plot_hog.html

[33] Sharma, A.: *Understanding Activation Functions in Neural Networks*. Březen 2017, [Online; navštíveno 3.5.2019].
URL https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0

[34] SZ DJI Technology Co., Ltd.: SPARK SPECS. 2017, [Online; navštíveno 10.5.2019].
URL https://www.dji.com/cz/spark/info#specs

[35] SZ DJI Technology Co., Ltd.: MAVIC 2 SPECS. 2018, [Online; navštíveno 10.5.2019].
URL https://www.dji.com/cz/mavic-2/info#specs

[36] Trenz, O.; Fejfar, J.; Popelka, O.; aj.: *Umělá inteligence, eLearningová opora k předmětu Umělá inteligence 1*. 2010, [Online; navštíveno 21. 12. 2018].
URL https://is.mendelu.cz/eknihovna/opory/index.pl?opora=2068

[37] Vasudev, R.: *How to Initialize weights in a neural net so it performs well?* Květen 2018, [Online; navštíveno 25.1.2019].
URL https://hackernoon.com/how-to-initialize-weights-in-a-neural-net-so-it-performs-well-3e9302d4490f

[38] Viola, P.; Jones, M.: Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Dec 2001, ISSN 1063-6919.
URL
https://www.cs.cmu.edu/~efros/courses/LBMV07/Papers/viola-cvpr-01.pdf

[39] Volná, E.: *Neuronové sítě 1*. 2008, [Online; navštíveno 27. 12. 2018].
URL http://www1.osu.cz/~volna/Neuronove_site_skripta.pdf

[40] Wikipedia: *Artificial neural network*. Prosinec 2018, [Online; navštíveno 21.12.2018].
URL https://en.wikipedia.org/wiki/Artificial_neural_network

[41] Wikipedia: *Unmanned aerial vehicle*. Prosinec 2018, [Online; navštíveno 20.12.2018].
URL https://en.wikipedia.org/wiki/Unmanned_aerial_vehicle

[42] Wikipedia: *Overfitting*. Únor 2019, [Online; navštíveno 3.5.2019].
URL https://en.wikipedia.org/wiki/Overfitting

[43] Zhang, X.; Luo, H.; Fan, X.; aj.: AlignedReID: Surpassing Human-Level Performance in Person Re-Identification. *CoRR*, ročník abs/1711.08184, 2017.
URL http://arxiv.org/pdf/1711.08184.pdf

Příloha A

Obsah přiloženého paměťového média

Na přiloženém paměťovém médiu se nachází:

- app/ – demonstrační aplikace People Detector
- app/models/ – natrénované modely na datasetu Stanford Drone Dataset
- app/src/ – zdrojové kódy aplikace
- app/requirements.txt – seznam potřebných knihoven pro zprovoznění aplikace
- app/LICENSE – licence
- examples/ – testovací obrázky a videa pro People Detector
- tex/ – adresář se zdrojovými texty technické zprávy v jazyce LaTeX
- readme.txt – manuál pro zprovoznění aplikace a další informace
- text.pdf – technická zpráva
- text-print.pdf – technická zpráva pro tisk (odkazy jsou černé)