VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

SLEDOVÁNÍ POHYBU MÍČE VE VIDEU

BALL TRACKING IN SPORTS VIDEO

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. MATÚŠ MOTLÍK**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. VOJTĚCH BARTL**
SUPERVISOR

BRNO 2019

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2018/2019

Zadání diplomové práce

Student: **Motlík Matúš, Bc.**
Program: Informační technologie Obor: Počítačová grafika a multimédia
Název: **Sledování pohybu míče ve videu**
Ball Tracking in Sports Video
Kategorie: Zpracování obrazu

Zadání:

1. Prostudujte existující metody pro sledování pohybu míče.
2. Navrhněte vhodnou metodu, která bude schopna detekovat pozici míče a trajektorii jeho pohybu ze statických záběrů.
3. Zpracujte a připravte testovací množinu dat z poskytnutých videí.
4. Implementujte řešení podle navržené metody. Očekávaným výstupem je pozice míče v každém snímku.
5. Ověřte funkčnost a spolehlivost výsledné implementace na poskytnutých datech, zdokumentujte a navrhněte možná vylepšení.
6. Vytvořte stručné video prezentující vaši práci, její cíle a výsledky.

Literatura:

- dle pokynů vedoucího

Při obhajobě semestrální části projektu je požadováno:

- Body 1 až 3.

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

Vedoucí práce: **Bartl Vojtěch, Ing.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 22. května 2019
Datum schválení: 1. listopadu 2018

Abstrakt

Táto diplomová práca sa zaoberá automatickou detekciou a sledovaním futbalovej lopty v zázname športového stretnutia. Na základe predstavených techník zameraných na sledovanie malých objektov vo vysokom rozlíšení sú navrhnuté a implementované efektívne konvolučné neurónové siete, ktoré ďalej využíva upravený sledovací algoritmus SORT pre automatickú detekciu objektov v obraze. Za účelom preskúmania možností spracovania v reálnom čase pri čo najnižšej strate presnosti sledovania sú uskutočnené experimenty so spracovávaním snímok v rôznych rozlíšeniach a s rôznou frekvenciou získavania detekcií. Získané výsledky experimentov sú prezentované a využité pre návrh ďalších rozšírení, ktoré by viedli k zlepšeniu úspešnosti sledovania pri zachovaní dostatočnej rýchlosti spracovania.

Abstract

This master's thesis deals with automatic detection and tracking of a soccer ball in sports videos. Based on the introduced techniques focusing on tracking of small objects in high-resolution videos, effective convolutional neural networks are designed and used by a modified version of tracking algorithm SORT for automatic object detection. A set of experiments with the processing of images in different resolutions and with various frequencies of detection extraction is carried out in order to examine the trade-off between processing speed and tracking accuracy. The obtained results of experiments are presented and used to form proposals for future work, which could lead to improvements in tracking accuracy while maintaining reasonable processing speed.

Kľúčové slová

detekcia objektov, sledovanie objektov, futbal, strojové učenie, hlboké neurónové siete

Keywords

object detection, object tracking, soccer, machine learning, deep neural networks

Citácia

MOTLÍK, Matúš. *Sledování pohybu míče ve videu.* Brno, 2019. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Vojtěch Bartl

Sledování pohybu míče ve videu

Prehlásenie

Prehlasujem, že som túto diplomovú prácu vypracoval samostatne pod vedením pána Ing. Vojtěcha Bartla. Ďalšie informácie mi poskytol pán Ing. Igor Potůček, Ph.D. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

.....
Matúš Motlík
22. mája 2019

Poďakovanie

Týmto by som sa chcel poďakovať Ing. Vojtěchovi Bartlovi za odborné vedenie, pomoc a cenné rady pri tvorbe tejto práce. Rovnako ďakujem Ing. Igorovi Potůčkovi, Ph.D. za poskytnutie dát pre túto prácu a informácie predané pri konzultáciách.

Obsah

Kapitola 1

Úvod

Využitie informačných technológií v športe zaznamenáva v poslednom období veľký rozmach. Inovatívne technologické riešenia pomáhajú zefektívňovať športové výkony atlétov, sú využívané pri analýze športových stretnutí, sumarizovaní zápasov a množstvo ďalších vecí. Futbal, ktorý patrí medzi celosvetovo jeden z najpopulárnejších športov preto za ostatnými nezaostáva. Hráči a tréneri využívajú informačné systémy pre zdokonalenie svojej taktiky a výsledkov, kamerové systémy pomáhajú rozhodcom riešiť komplikované situácie počas hry a v neposlednom rade nástup nových technológií napríklad umožnil divákom sledovať zápas svojho obľúbeného tímu odkiaľkoľvek. Tieto služby poskytuje množstvo spoločností, pôsobiacich po celom svete.

Jednou z nich je Brnenská firma CamVision s.r.o.[1] zaoberajúca sa poskytovaním systému pre nahrávanie a analýzu futbalových zápasov a tréningov profesionálnym európskym tímom. Poskytovaný systém sa skladá z dvoch hlavných častí. Prvá časť realizuje automatické nahrávanie zápasov, druhú časť tvorí aplikácia pre analýzu zápasu, okrem toho poskytuje možnosť sledovať živý prenos práve hranej hry alebo záznam nejakého zápasu, či tréningu. Futbalové stretnutie je nahrávané pomocou dvojice kamier vo vysokom rozlíšení. Zábery kamier, z ktorých každá zaznamenáva polovicu ihriska sú spojené do širokého panoramatického záberu poskytujúceho pohľad na celú hraciu plochu. Z tohto veľkého záberu sa automatickým softvérom vyberá iba určitá časť, typicky tá, kde sa práve hrá. Automaticky sa tak vytvára záber simulujúc nahrávanie kameramanom. Pre vytvorenie takéhoto výrezu panoramatického záberu je potrebné získať dodatočné informácie o prebiehajúcej hre. Týmito informáciami sú predovšetkým poloha hracej lopty a jednotlivých hráčov v každej snímke video záznamu.

Spôsobov sledovania objektov v obraze existuje veľké množstvo. V dnešnej dobe je však stále populárnejšie riešiť podobnú analýzu obrazu využitím hlbokých neurónových sietí na rozdiel od tradičnejších, ručne navrhnutých algoritmov. Nespornou výhodou neurónových sietí je schopnosť učiť sa, čo znamená možnosť znovu použitia rovnakej implementácie modelu siete na riešenie podobných problémov. Samotné riešenie problému preto nezávisí iba na štruktúre použitej siete, ale predovšetkým na dátovej sade, kde vzťahy medzi dátami popisujú riešený problému.

Táto práca má preto za cieľ navrhnúť a implementovať systém pre sledovanie futbalovej lopty v panoramatickom video zázname využitím hlbokých neurónových sietí. Následne môže byť tento systém využitý ako samostatný modul v programe na analýzu a spracovanie futbalových zápasov, napríklad pre získanie informácie v ktorej časti ihriska práve prebieha

[1] https://www.camvision.cz/

hra. Táto informácia môže byť ďalej využitá pre zaostrenie a priblíženie virtuálnej kamery na túto časť obrazu, čím vznikne efekt simulujúc prácu skutočného kameramana.

V jednotlivých kapitolách tejto práce sú postupne popísané metódy riešenia problému sledovania predovšetkým malých objektov. Na začiatku kapitoly 2 je predstavený základný princíp neurónových sietí a možnosti ich využitia pre detekciu objektov v obraze, s dôrazom predovšetkým na detekciu malých objektov. V závere kapitoly sú spomenuté vybrané sledovacie algoritmy. Kapitola 3 popisuje získanú datovú sadu. V nasledujúcej kapitole 4 je podobne definovaný riešený problém a navrhnutý systém na jeho riešenie. Stručný popis implementácie a následné vyhodnotenie implementovaného systému a jeho častí sú popísané v závere tejto práce v kapitolách 5 a 6.

Kapitola 2

Sledovanie objektov v obraze využitím neurónových sietí

Sledovanie objektov v obraze je možné riešiť viacerými prístupmi, z ktorých jeden z najpoužívanejších je tzv. sledovanie na základe detekcií (*tracking by detection*). Problém je v tomto prípade možné rozdeliť na dve hlavné časti, a to detekciu objektov v obraze, ktorú je možné riešiť tradičnými algoritmami, akými sú napríklad SVM, AdaBoost a mnohé iné. V dnešnej dobe sú však populárnejšie riešenia založené na hlbokých neurónových sieťach, ktoré dosahujú vo viacerých prípadoch výrazne lepšie výsledky. Druhou časťou je zostrojenie trajektórií sledovaných objektov na základe získaných detekcií. Nasledujúca kapitola preto ponúka prehľad používaných techník.

2.1 Neurónové siete

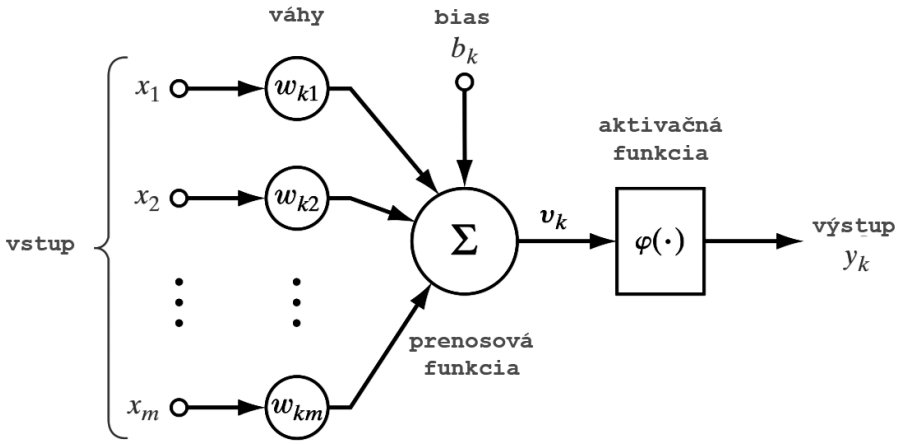
Neurónové siete sú biologicky inšpirované matematické modely strojového učenia. Ich podstata spočíva vo všeobecnej aproximácii funkcií, vďaka čomu sú schopné učiť sa a rozpoznávať závislosti v rôznych dátach. Hodia sa preto na riešenie rozličných problémov, napr. klasifikáciu, segmentáciu alebo detekciu objektov v obraze, problémy spracovania prirodzeného jazyka, predikcie dát a množstvo iných úloh.

Základnou jednotkou neurónových sietí je neurón. Na vstupe má vektor hodnôt $\mathbf{x}$, z ktorých sa pomocou vektoru váh $\mathbf{w}_k$, posunu b_k (*bias*) a aktivačnej funkcie φ vypočíta predikcia y_k. Matematicky je možné neurón popísať vzorcom 2.1.

$$y_k = \varphi\left(\sum_{i=1}^{m} w_{ki}x_i + b_k\right) \tag{2.1}$$

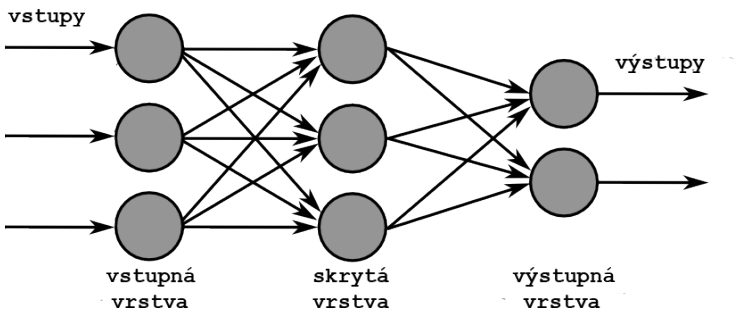
Neuróny sú organizované vo vrstvách, ktoré v prípade doprednej (*feedforward*) neurónovej siete, viď schému 2.2, delíme na vstupné, výstupné a skryté vrstvy. Výstup z jednej vrstvy je v tomto prípade vstupom tej nasledujúcej. Samotné vrstvy sa ďalej rozlišujú podľa typu funkcie, ktorú implementujú, napríklad na plne-prepojené, konvolučné, rekurentné, tzv. *pooling* vrstvy a rôzne ďalšie.

Dôležitou časťou jednotlivých vrstiev sú aktivačné funkcie vo vzorci 2.1 označené ako φ. Ich základný princíp spočíva v zavedení nelinearity do výpočtu modelu siete, ktorou sa snažíme reprezentovať komplexné nelineárne mapovanie vstup na výstup. Sú inšpirované akčným potenciálom biologickej neurónovej bunky. Vážený súčet vstupného vektoru je po-



Obr. 2.1: Schematická vizualizácia neurónu. Prevzaté z [12]

mocou tejto funkcie prevedený nelineárne na novú hodnotu definovanú typom funkcie. Bez týchto nelineárnych funkcií by samotná sieť bola iba kombináciou rôznych lineárnych funkcií, a teda celá by tiež implementovala iba lineárnu funkciu. To by však malo negatívny dopad na flexibilitu modelu, rýchlosť učenia a jeho konvergenciu, a teda aj celkovú presnosť neurónovej siete. Nelineárnych funkcií existuje veľké množstvo, nie každá je však vhodná pre použitie v konkrétnom prípade. Dôležitou vlastnosťou týchto funkcií je ich derivovateľnosť. Medzi najpoužívanejšie aktivačné funkcie patria *sigmoid*, *hyperbolický tangens* a predovšetkým *ReLU*.



Obr. 2.2: Vizualizácia doprednej (*feedforward*) neurónovej siete[1]

Trénovanie

Dôležitou vlastnosťou neurónových sietí je schopnosť naučiť sa nájsť riešenie konkrétneho problému, inými slovami nájsť netriviálne mapovanie vstupných dát na výstupné. Trénovanie siete je možné chápať ako iteratívne upravovanie váh a posunov jednotlivých neurónov modelu tak, že sa progresívne aproximuje vzťah vstupných a výstupných dát v známej trénovacej datovej sade. V jednej iterácií učenia model najprv vypočíta predikciu z trénovacích dát, čo je tiež označované ako dopredný prechod (*forward-pass*) modelom siete. To zahŕňa výpočet váženého súčtu vstupných dát a následne aplikovanie aktivačnej funkcie na tento

[1]Prevzaté z http://web.utk.edu/~wfeng1/spark/fnn.html

súčet v každom neuróne každej vrstvy siete. Výstupom poslednej vrstvy siete je samotná predikcia. Potom je potrebné vypočítať celkovú chybu predikcie a upraviť váhy a posuny neurónov tak, aby sa minimalizovala táto chyba v každej nasledujúcej iterácii. Úprava váh prebieha pomocou algoritmu spätného prechodu modelom siete. Počas trénovania sa model učí predikovať hodnoty v trénovacej datovej sade, pričom spracovanie celej sady je označované ako epocha. Učenie typicky prebieha v niekoľkých epochách, v závislosti na riešenom probléme a type samotných dát.

Výpočet derivácií a algoritmus spätného prechodu sieťou

Základným zdrojom informácií o progrese učenia je hodnota chybovej funkcie. Táto funkcia v podstate indikuje rozdiel medzi očakávaným a skutočným výstupom neurónovej siete [12]. V závislosti na type problému sa používajú rôzne chybové funkcie akými sú napríklad stredná kvadratická chyba (*mean square error*), *cross entropy* a iné.

Ako bolo spomenuté vyššie, cieľom trénovania je iteratívne upravovať váhy modelu siete v závislosti na tejto chybe. Zmeny parametrov sa vypočítajú pomocou algoritmu gradientného zostupu (*gradient descent*), ktorý využíva algoritmus spätného prechodu (*backpropagation*) pre výpočet komplexných gradientov chybovej funkcie v závislosti na parametroch modelu siete. Tento algoritmus počíta gradienty iteratívne v jednotlivých vrstvách siete pomocou tzv. reťazového pravidla (*chain-rule*) [12].

Gradient je matematicky derivácia vektoru funkcie, ktorá je závislá na viac než jednej premennej. Inými slovami, gradient ukazuje smer 'stúpania' funkcie. V opačnom smere gradient ukazuje smer k lokálnemu minimu funkcie. Pomocou algoritmu gradientného zostupu je preto možné nájsť lokálne minimum chybovej funkcie a teda efektívne minimalizovať chybu predikcie úpravou parametrov modelu. Váhy a posuvy modelu siete v jednotlivých vrstvách sa aktualizujú podľa hodnoty vypočítaného gradientu a hyper-parametru miery učenia (*learning rate*), ktorá umožňuje kontrolovať veľkosť zmeny váh neurónov, čím v konečnom dôsledku vplýva na rýchlosť učenia celej neurónovej siete [12]. Príliš veľký parameter miery učenia však môže spôsobiť, že algoritmus gradientného zostupu nebude konvergovať k lokálnemu minimu. V praxi sa najčastejšie používa praktika postupného znižovania tohto parametru po určitom počte iterácií, čo umožňuje rýchlejšiu konvergenciu na začiatku trénovania a zároveň nájdenie presnejšieho minima vďaka menším úpravám parametrov modelu v neskorších iteráciách.

Typy neurónových sietí

Hlboké neurónové siete pozostávajú z množstva neurónov organizovaných v rôznych vrstvách, prípadne doplnených o rôzne trénovateľné funkcie a filtre. Používajú veľké množstvo rôznych chybových a aktivačných funkcií v závislosti na riešenom probléme. Všetky tieto vlastnosti umožňujú neurónovým sieťam riešiť celú škálu rôznych úloh, napríklad detekciu objektov v obraze, spracovanie prirodzeného jazyka, predpovedanie a množstvo ďalších.

Vo všeobecnosti je možné rozdeliť architektúry neurónových sietí do troch kategórií:

- **Dopredné neurónové siete** - Patria k najjednoduchším typom neurónových sietí. Dáta sú v tomto prípade spracovávané samostatne bez vplyvu na iné dáta, jedným smerom cez vstupnú až po výstupnú vrstvu bez akýchkoľvek cyklov. Príkladmi sú predovšetkým konvolučné a plne-prepojené siete.
- **Rekurentné neurónové siete** - Tieto siete sú navrhnuté na spracovávanie sekvenčných dát, aktuálny výstup je ovplyvnený súčasným vstupom a prípadne jedným alebo

viacerými predchádzajúcimi vstupmi. Dokážu preto zachytiť dlhodobejšie závislosti v dátach, a preto sú vhodné napríklad pre riešenie problémov prirodzeného jazyka.

- **Symetricky prepojené siete** - Medzi ich predstaviteľov patria Hopfieldove siete a Boltzmannove stroje.

2.2 Základné konvolučné neurónové siete

Konvolučné neurónové siete sa počas trénovania učia na určitej datovej sade v jednotlivých vrstvách realizovať rôzne druhy filtrov, ktoré extrahujú vhodné príznaky popisujúce relatívne malé lokálne okolie vstupu. Inými slovami, hierarchicky filtrujú vstupný obraz čím extrahujú veľké množstvo latentných príznakov, ktoré sémanticky popisujú vstup siete. V prípade tradičných detekčných algoritmov je naopak nutné ručne implementovať podobné filtrovacie funkcie, čo je do značnej miery náročné a často krát vyžadujúce pokročilé znalosti riešeného problému. Oproti iným typom sietí navyše vyžadujú výrazne menšie množstvo parametrov, práve aplikovaním jedného konvolučného jadra na celý vstup, vďaka čomu sú pamäťovo a výpočetne efektívnejšie.

Medzi najznámejšie základné konvolučné siete patria *Alexnet* [16], *VGG siete* [29], prípadne z novších *Resnet* [14], *Xception* [3], alebo *MobileNet* [28], ktoré sú pôvodne navrhnuté pre klasifikáciu obrazu. Tieto siete sú však pomerne často využívané aj ako tzv. extraktory príznakov v zložitejších modeloch sietí navrhnutých pre iné úlohy ako samotná klasifikácia. Výstupom týchto extraktorov sú aktivačné mapy, ktoré sú napojené na ďalšie vrstvy neurónovej siete, ktoré z nich napríklad detegujú objekty, alebo počítajú segmentáciu. Používané extraktory bývajú rôzne zložité v závislosti na riešenom probléme.

K jedným z dôvodov výrazného rozšírenia vyššie spomenutých sietí patrí aj to, že ich implementácia býva často súčasťou knižníc určených pre hlboké učenie, v niektorých prípadoch aj s už natrénovanými váhami na veľkých datasetoch, akými je napríklad *ImageNet*[2]. Tieto predtrénované modely sa často používajú na dotrénovanie siete pre riešenie inej, príbuznej úlohy, čo má výhodu predovšetkým v prípadoch s malou datovou sadou, pri ktorej by trénovanie od nuly (*from scratch*) nemuselo konvergovať k očakávanému riešeniu.

2.2.1 ResNet

Neurónové siete zvyčajne realizujú mapovanie $\mathcal{H}(\mathbf{x})$ vstupu na výstup pomocou niekoľkých za sebou postavených vrstiev. Teoreticky, použitím viac takýchto vrstiev by malo byť možné aproximovať stále komplikovanejšie mapovania, teda na riešenie zložitých problémov by malo stačiť použiť hlbšie siete. V praxi sa však ukázalo trénovanie takto hlbokých sietí ako veľmi problematické, predovšetkým kvôli tzv. exlodujúcim gradientom a degradačnému problému [14]. Architektúra ResNet [14] však pre riešenie týchto problémov počas trénovania siete zavádza nový druh základného bloku tzv. reziduálny blok, znázornený na obrázku 2.3, vďaka čomu je možné úspešne používať aj neurónové siete skladajúce sa z desiatok až stoviek vrstiev.

Autori zistili, že na rozdiel od hľadania aproximácie priameho mapovania $\mathcal{H}(\mathbf{x})$ je jednoduchšie implementovať tzv. reziduálne mapovanie $\mathcal{F}(\mathbf{x}) = \mathcal{H}(\mathbf{x}) - \mathbf{x}$. Realizovaná funkcia je teda $\mathcal{F}(\mathbf{x}) + \mathbf{x}$, ktorá tiež dokáže aproximovať hľadané mapovanie vstupu na výstup, avšak pri výrazne jednoduchšom učení. V dopredných neurónových sieťach je možné túto

[2] http://www.image-net.org/

formuláciu implementovať pomocou tzv. *shortcut* spojení, viď obrázok 2.3a, ktoré preskakujú jednu alebo viac vrstiev siete a implementujú mapovanie identity. Ich výstup je potom jednoducho pričítaný k výstupu týchto preskočených vrstiev.

2.2.2 MobileNetV2

MobileNetV2 [28] je príkladom architektúry navrhnutej pre mobilné zariadenia a platformy s obmedzenými zdrojmi. Signifikantne redukuje množstvo parametrov siete a počet pamäťových operácií pri zachovaní porovnateľnej úspešnosti s výrazne robustnejšími sieťami, akými sú vyššie spomenuté ResNet, VGG a pod. Vďačí za to špeciálnemu typu základného bloku, tzv. invertovanému reziduálnemu bloku s *bottleneck* vrstvou, schematicky znázornenému na obrázku 2.3b.

Tento blok využíva tzv. reziduálne *shortcut* spojenia medzi tenkými *bottleneck* vrstvami. Medzi týmito vrstvami sú príznaky filtrované tzv. *depthwise* konvolučným blokom, ktorý najprv rozširuje málo dimenzionálny vstup do viac kanálového a ten je následne filtrovaný tzv. *depthwise separable* konvolučnou vrstvou. Nakoniec sú získané príznaky premietnuté späť do nízko dimenzionálnej reprezentácie pomocou lineárnej konvolúcie.

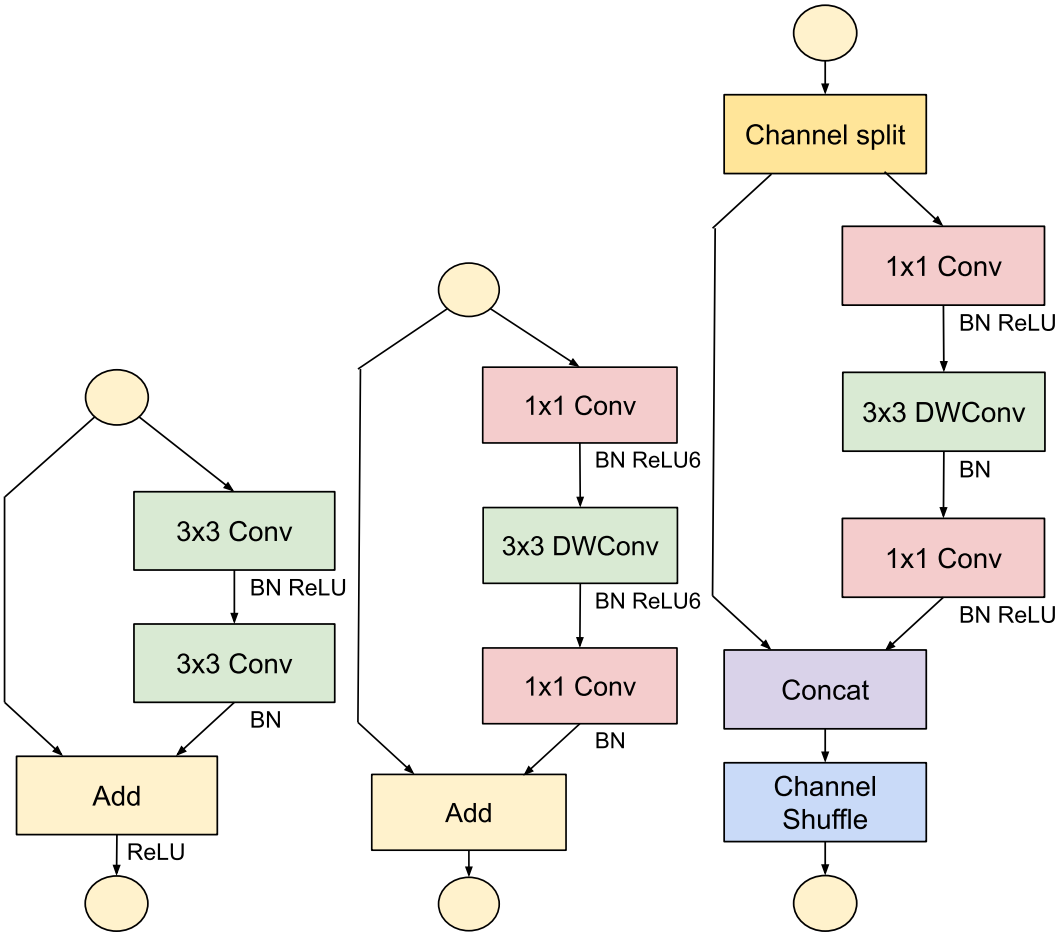
Tzv. *depthwise separable* konvolúcie, prvý krát predstavené v architektúre Xception [3] sú dnes používané v mnohých neurónových sieťach [22, 28] vďaka svojej efektivite. Ich myšlienka spočíva v nahradení plnej konvolučnej vrstvy dvoma oddelenými vrstvami, čím sa výrazne redukuje potrebné množstvo parametrov na vykonanie danej operácie. Prvá v poradí, tzv. *deptwise* konvolučná vrstva aplikuje jedno konvolučné jadro oddelene na každý vstupný kanál. Získané príznaky v jednotlivých vrstvách sú následne lineárne kombinované v *pointwise* konvolučnej vrstve pomocou konvolučného jadra 1×1, čím sa získa výstup odpovedajúci aplikácií klasickej konvolučnej vrstvy.

Formálne zapísané, štandardná konvolučná vrstva má na vstupe tenzor L_i veľkosti $h_i \times w_i \times d_i$, kde h, w je výška, resp. šírka tenzoru a d je jeho hĺbka (počet kanálov). Aplikovaním konvolučného jadra $K \in \mathcal{R}^{k \times k \times d_i \times d_j}$ na tenzor L_i, kde k je veľkosť jadra, vznikne tenzor L_j veľkosti $h_j \times w_j \times d_j$. Výpočetná náročnosť tejto operácie je preto $h_i * w_i * d_i * d_j * k * k$. Oproti tomu tzv. *depthwise separable* konvolučné vrstvy nahradzujú štandardnú konvolúciu, pričom ich výpočetná náročnosť je súčtom náročností tzv. *depthwise* a *pointwise* konvolúcie, teda formálne $h_i * w_i * d_i(k^2 + d_j)$. V prípade architektúry *MobileNetV2* redukujú výpočet oproti implementácií s klasickými konvolučnými vrstsvami 8 až 9 násobne za cenu nepatrnej straty presnosti [28].

Špecifikom tejto architektúry je možnosť parametrizovať počet parametrov invertovaných reziduálnych blokov a tým vplývať na celkovú veľkosť a rýchlosť modelu pomocou parametru `width-multiplier`. V prípade zníženia počtu parametrov však dôjde z veľkou pravdepodobnosťou k zníženiu presnosti predikcií modelu.

2.2.3 ShuffleNetV2

Ďalším príkladom výpočetne a pamäťovo efektívnej architektúry je tzv. *ShuffleNet* [31, 22]. Pre zníženie výpočetnej náročnosti sú aj v tomto prípade využívané tzv. *bottleneck* základné bloky podobné tým, využívaným v MobileNet a ResNet sieťach. V základnom bloku sú však navyše implementované špeciálne operácie *channel split* a *channel shuffle*. Operácia *channel split*, rozdelí tenzor L veľkosti $h \times w \times d$ do dvoch vetiev s približne rovnakým počtom kanálov $d - d'$ a d'. Prvá vetva dáta žiadnym spôsobom nespracováva, teda pôsobí ako identita. Druhá naopak pozostáva z troch konvolúcií, jednej *depthwise* a dvoch 1×1 konvolúcií. Výstup týchto dvoch vetiev je následne konkatenovaný a náhodne premiešaný operáciou



Obr. 2.3: Znázornenie základných reziduálnych blokov jednotlivých sietí. (a) ResNet-18 [14]. (b) MobileNetV2 [28] a (c) ShuffleNetV2 [22]

channel shuffle. Táto operácia umožňuje prúdenie informácií medzi rozdelenými skupinami kanálov. Vďaka týmto zmenám je možné použiť v modely *ShuffleNetV2* viac kanálov vo vrstvách siete a teda zvýšiť úspešnosť pri zachovaní porovnateľnej komplexnosti výpočtu oproti tradičným architektúram sietí, akými sú napríklad ResNet [14], alebo Xception [3]. Základný blok modelu ShuffleNetV2 je znázornený na obrázku 2.3c.

2.3 Detekcia objektov v obraze pomocou neurónových sietí

Konvolučné neurónové siete pre detekciu objektov je možné rozdeliť na dve základné skupiny podľa spôsobu spracovania vstupného obrazu na tzv. jedno-priechodové detektory [24, 25, 21, 20, 8] a detektory založené na regiónoch, príp. nazývané aj dvoj-priechodové detektor [10, 27, 19, 13]. Prvé spomenuté riešia detekciu objektov ako regresný a klasifikačný problém implementovaný v jednom modeli. Majú za cieľ čo najrýchlejšie spracovanie vstupu za cenu čo najmenších chýb pri detekcii. Dvoj-priechodové detektory sú vo všeobecnosti pomalšie, no dosahujú vo všeobecnosti lepšie výsledky. Tieto modely spracúvajú vstupný obraz v dvoch oddelených krokoch. Najprv sa generujú kandidátne regióny (*regions proposal*) a to

buď algoritmicky [11], alebo pomocou rôznych modelov konvolučných sietí [27, 19] (tzv. základná sieť detektoru). Následne prebieha rozpoznávanie a detekcia objektov v samotných regiónoch pomocou tzv. klasifikačných a regresných podsietí.

2.3.1 Detektory založené na regiónoch

Medzi prvých predstaviteľov týchto detektorov patrí R-CNN [11]. Pre generovanie kandidátnych regiónov používa algoritmy podobné selektívnemu prehľadávaniu [30], *Edge Boxes* [17] a iné. Tieto algoritmy generujú približne 2000 regiónov, v ktorých sa následne klasifikujú a detegujú jednotlivé objekty separátne pomocou konvolučnej neurónovej siete. Takéto spracovanie je však výpočetne neefektívne a komplikuje trénovanie. Ďalším detektorom, ktorý výrazne zrýchlil detekciu a zjednodušil trénovanie modelu je Fast R-CNN [10]. Fast R-CNN upravil spôsob spracovania kandidátnych regiónov, kedy sa namiesto extrakcie príznakov separátne v jednotlivých regiónoch spracováva celý vstupný obraz niekoľkými konvolučnými vrstvami. Týmto spôsobom sa získa mapa príznakov (*feature map*), z ktorej sa vyberú iba časti odpovedajúce navrhovaným regiónom. Tie sú následne spracovávane podobne ako u R-CNN.

Faster R-CNN a odvodené

Napriek týmto úpravám modelu sa pre generovanie kandidátnych regiónov z mapy príznakov používa relatívne pomalý algoritmus *selective search*. Preto bol vo **Faster R-CNN** [27] detektore nahradený konvolučnou neurónovou sieťou, označenou ako tzv. **RPN** (*Regions Proposals Network*), ktorá dokáže pomocou tzv. *Anchor boxov* generovať regióny v rôznych rozlíšeniach a pomeroch strán. Vďaka efektívnejšiemu generátoru kandidátnych regiónov bolo výrazne zrýchlené celé spracovanie dát a dokonca zlepšená celková úspešnosť modelu siete oproti predchádzajúcim verziám [27]. Výraznejší problém však má s detekciou a lokalizáciou malých objektov [32].

Na základe Faster R-CNN boli následne postavené ďalšie práce, ktoré sa snažili zlepšiť detekciu objektov modifikovaním RPN siete. Jedným z príkladov sú tzv. **Deformovateľné konvolučné siete** (*Deformable Convolutional Networks*), ktoré nahradili tradičné konvolučné a *pooling* operácie tzv. deformovateľnými konvolúciami s cieľom naučiť model dynamické vnímavé pole (*receptive field*) bez predchádzajúcej úpravy dát, napr. augmentácie [5, 18]. Medzi ďalšie príklady patrí tzv. **FPN** sieť (*Feature Pyramid Network*) [19], ktorá generuje mapy príznakov v rôznych rozlíšeniach pomocou hierarchie konvolučných vrstiev organizovaných v štruktúre podobnej pyramíde. Podrobnejšie je tento model popísaný v podkapitole 2.4.1. Na základe FPN modelu bola navrhnutá **Mask R-CNN** sieť [13] používaná predovšetkým na sémantickú segmentáciu obrazu. Ďalší výskum bol okrem iného zameraný na zjednodušenie modelu siete pre dosiahnutie efektívnejšieho a pamäťovo menej náročného spracovania. Jedným z príkladov je **Light Head R-CNN**, kde bola zjednodušená tzv. hlava detektoru použitím tenkej mapy príznakov vďaka čomu bola výrazne zrýchlená detekcia objektov [18].

R-FCN (*Region-based Fully Convolutional Network*) [4] inšpirovaná plne konvolučnými klasifikačnými sieťami eliminuje separátne spracovanie regiónov objektov pomocou pozične závislých máp príznakov následne spracovaných pomocou pozične závislej *pooling* vrstvy. Tieto modifikácie umožňujú na rozdiel od Faster R-CNN výrazné zjednodušenie trénovania, sieť sa trénuje ako celok. R-FCN dosahuje presnejšie výsledky a rýchlejšie spracovanie obrazu v porovnaní s Faster R-CNN [4].

Výhody a nevýhody detektorov založených na oblastiach

Pre dosiahnutie čo najvyššej presnosti je potrebné generovať veľké množstvo kandidátnych regiónov relatívne komplikovaným spôsobom. Tie sú následne spracovávané pomerne robustnou neurónovou sieťou. To všetko negatívne vplýva na výpočetnú náročnosť celého modelu siete a preto sú vo všeobecnosti pomalšie v porovnaní s jedno-priechodovými detektormi. Dosahujú však presnejšie výsledky[18, 32].

2.3.2 Jedno-priechodové detektory objektov

Oproti detektorom založeným na oblastiach upravujú spôsob spracovania vstupu. Algoritmus generovania kandidátnych regiónov a následná klasifikácia a detekcia objektov sú spojené do jednej siete. Celý vstupný obrázok je typicky spracovaný základnou konvolučnou sieťou, označovanou aj tzv. *back-bone* sieť. Jej výstupom je aktivačná mapa v ktorej sa následne klasifikujú jednotlivé objekty pomocou klasifikačnej podsiete. Pre získanie ohraničujúcich obdĺžnikov sa spracuje aktivačná mapa inou, tzv. regresnou podsieťou. Kandidátne regióny sú typicky dopredu definované napríklad fixnou mriežkou, ktorou je rozdelený obraz. Každá bunka tejto mriežky má typicky definovaných niekoľko tzv. *anchor boxov*, v ktorých sa následne detegujú jednotlivé objekty. Výstupom spracovania vstupného obrazu je zoznam klasifikovaných objektov a súradníc ich ohraničujúcich obdĺžnikov. Často krát je však pre rovnaký objekt vygenerovaných príliš veľa obdĺžnikov, ktoré je potrebné eliminovať, napríklad metódou *non-maxima supression*.

Vzhľadom k minimalizácií náročného výpočtu kandidátnych regiónov a relatívne jednoduchým klasifikačným a regresným podsieťam, sú tieto detektory výrazne rýchlejšie oproti dvoj-priechodovým detektorom, avšak za cenu menšej presnosti [32].

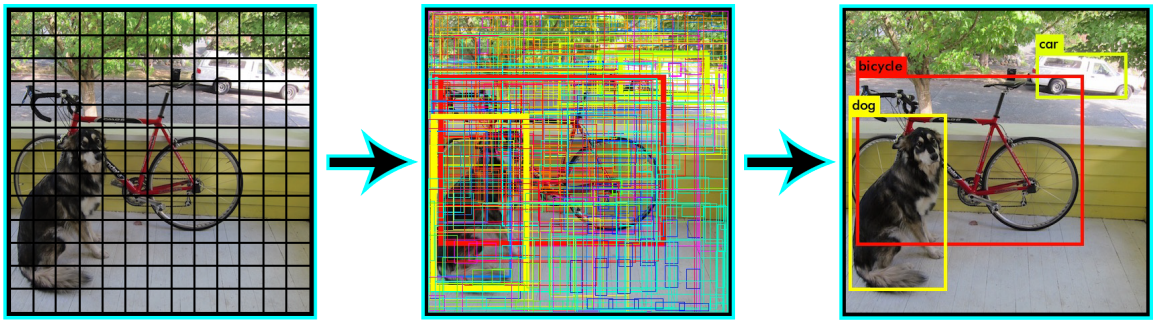
SSD

SSD (Single Shot Detector) [21] je detekčná sieť, ktorej základom je modifikovaná VGG16 sieť pre extrakciu mapy príznakov v ktorej sa následne pomocou ďalších konvolučných vrstiev detegujú a klasifikujú objekty. SSD dokáže lokalizovať rôzne veľké objekty vďaka preddefinovaným ohraničujúcim obdĺžnikom (nazývanými aj *anchor boxy*) v rôznom rozlíšení, vzhľadom k tomu, že sú navrhované z rôznych častí základnej VGG siete.

YOLO

Z predstaviteľov jedno-priechodových detektorov je často používaný model **YOLO** (You Only Look Once [24]. Táto sieť rozdelí vstupný obraz na mriežku veľkosti $S \times S$, viď obrázok 2.4, kde každá bunka je zodpovedná za predpovedanie objektu, ktorého stred sa nachádza práve v nej. V každej bunke je predikovaných B ohraničujúcich obdĺžnikov a ich odpovedajúca istota (confidence score), ktorá reprezentuje pravdepodobnosť výskytu objektu v danom ohraničujúcom obdĺžniku. Následne, podľa prahu istoty, sa vo vybraných obdĺžnikoch predikuje príslušnosť objektu do jednotlivých tried. Na predikované ohraničujúce obdĺžniky je následne aplikovaný algoritmus *non-maxima supression*, ktorý eliminuje duplicitné predikcie rovnakých objektov. Problémom tejto siete je však detekcia malých objektov a objektov stojacich príliš blízko sebe práve kvôli použitiu mriežky $S \times S$ a tomu, že v každej bunke je predikovaný práve jeden objekt.

Ďalšie verzie detektoru **YOLOv2** a **YOLO9000** [25] sa snažia eliminovať nedostatky a zlepšiť celkovú presnosť a rýchlosť modelu. Zavádzajú nové techniky ako normalizácia dát



Obr. 2.4: Princíp detekcie objektov v YOLO. Vstupný obraz je rozdelený na regióny, pre ktoré sa následne predikujú ohraničujúce obdĺžniky a predikcie tried objektov. Nakoniec sa eliminujú duplikáty pomocou metódy *non-maxima supression*. Prevzané z [24]

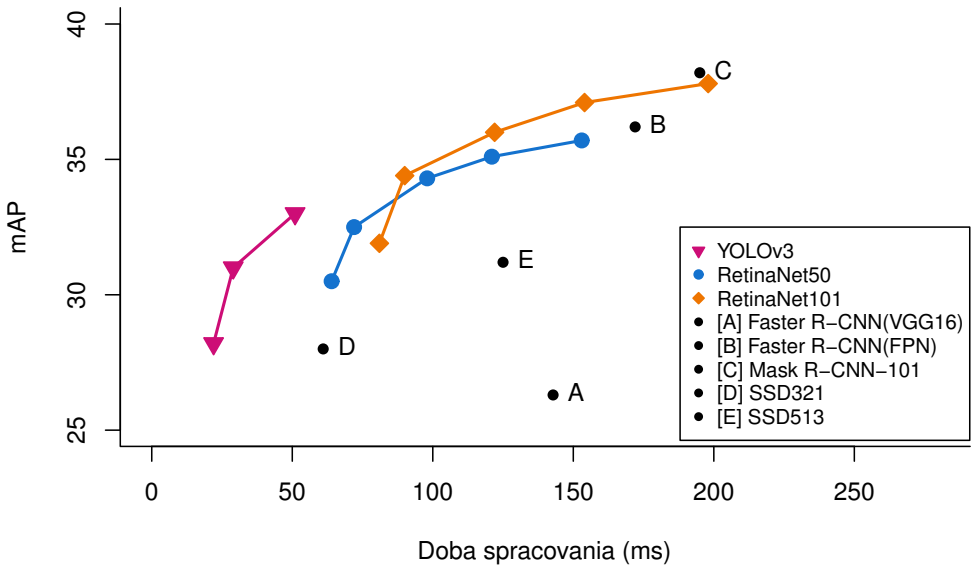
v rámci skupiny (*batch-normalization*), *anchor boxy* alebo trénovanie siete na vstupe v rôznom rozlíšení tzv. *multi-scale* trénovanie. **YOLOv3** [26] prináša ďalšie zlepšenia v podobe klasifikácie do viacerých tried súčasne (*multilabel classification*), predikovanie ohraničujúcich obdĺžnikov v troch rôznych rozlíšeniach pomocou konceptu podobnému FPN, viď [19] a zmenou siete extrahujúcej príznaky. Vďaka týmto zmenám YOLOv3 prekonáva predovšetkým rýchlosťou spracovania snímok ostatné tzv. rýchle detektory (SSD, RetinaNet, Faster R-CNN). V presnosti detekcie však stále zaostáva napríklad za modelom RetinaNet a novším RetinaMask [9], viď porovnanie na obrázku 2.5. Oproti predchádzajúcim verziám detektoru YOLO, tretia verzia dosahuje lepšie výsledky aj v detekcii malých objekto [26].

2.4 Detekcia malých objektov

V podkapitole 2.3 bolo popísaných niekoľko vybraných konvolučných sietí pre detekciu objektov v obraze, označované aj ako tzv. *state-of-the-art* siete. Viaceré z nich dosahujú vysokú presnosť na známych datasetoch, napríklad *MS COCO*[3], viď porovnanie v tabuľke 2.1 a grafe 2.5. Z porovnaní je možné pozorovať, že tzv. jednopriechodové detekčné siete dosahujú horšie výsledky avšak sú výrazne rýchlejšie oproti sieťam založeným na regiónoch. Vo všeobecnosti majú problém s detekciou malých objektov. Je to spôsobené predovšetkým malým rozlíšením aktivačných máp, kvôli čomu sú príznaky malých objektov často krát nedetekovateľné. Spomenuté detektory dokážu spracovávať vstupný obraz v rozličnom rozlíšení, vďaka použitiu konvolučných vrstiev. S rastúcim rozlíšením vstupu dochádza k zlepšeniu úspešnosti detekcie, viď. graf 2.5, avšak doba spracovania výrazne rastie. Je tiež možné pozorovať rozdiel úspešnosti v detekcii malých objektov u detektoroch *Faster R-CNN, YOLOv3 a RetinaNet*, ktoré vo svojej základnej sieti využívajú techniku tzv. pyramídy príznakov.

RetinaNet [20] je príkladom jedno-priechodových detektorov zložená so základnej siete typu FPN a dvoch podsietí pre klasifikáciu objektov a regresiu ohraničujúcich obdĺžnikov. Špecifikom tejto siete je použitie špeciálne navrhnutej chybovej funkcie, tzv. FocalLoss, ktorá má za ciel eliminovať negatívny vplyv výrazného nepomeru objektov popredia a pozadia počas trénovania a teda zlepšiť celkovú úspešnosť siete.

[3] http://cocodataset.org



Obr. 2.5: Porovnanie presnosti a rýchlosti konvolučnej neurónovej siete YOLOv3 s vybranými detekčnými sieťami na datasete *MS COCO*. mAP označuje priemernú presnosť (*mean Average Precision*). Pre model YOLOv3 sú postupne znázornené tri hodnoty odpovedajúce spracovaniu vstupov v rozlíšeniach 320×320 px, 416×416 px a 608×608 px. Hodnoty v prípade modelov typu RetinaNet zodpovedajú veľkostiam vstupu 400×400 px, 500×500 px, 600×600 px, 700×700 px a 800×800. px. Údaje boli prevzaté z [20, 26].

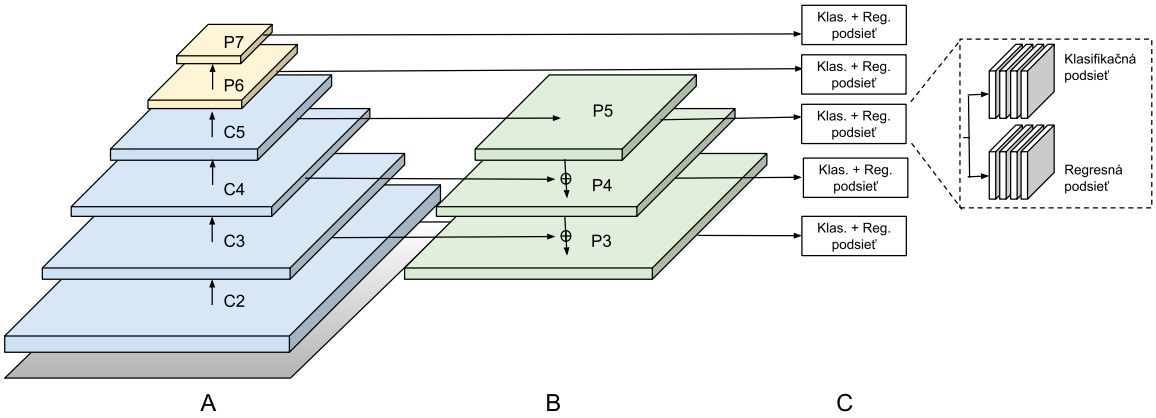
2.4.1 Štruktúra modelu RetinaNet

RetinaNet model pozostáva z plne konvolučnej siete typu *ResNet-FPN* a dvoch podsietí je znázornený na obrázku 2.6. Základná sieť, označovaná aj ako tzv. *back-bone*, má architektúru typu *ResNet* [14], z ktorej sú použité natrénované konvolučné vrstvy C_1 až C_5 doplnené o ďalšie dve konvolučné vrstvy P_6 a P_7. Na tieto vrstvy je pripojená pomocou tzv. bočných spojení FPN sieť (*Feature Pyramid Network*) navrhnutá podľa [19]. FPN sieť generuje tzv. pyramídy príznakov, teda aktivačné mapy z vrstiev P_3 až P_7 ktoré sú v rôznych rozlíšeniach, predovšetkým pre detekciu rôzne veľkých objektov. Vrstva P_i má rozlíšenie 2^i menšie ako vstupný obraz a všetky majú 256 kanálový výstup. Tieto aktivačné mapy sú následne spracovávané dvoma podsieťami, tzv. klasifikačnou a regresnou, ktoré sú podrobnejšie popísané v podkapitole 2.4.2.

Pyramídová štruktúra základnej siete bola zvolená pre generovanie sémanticky vyvážených príznakov v každej aktivačnej mape. Táto štruktúra umožňuje kombinovať sémanticky silné príznaky v malom rozlíšení (získané hlbšími konvolučnými vrstvami siete) so sémanticky slabšími príznakmi vo väčšom rozlíšení pomocou tzv. cesty zhora dole (*top-down*) a bočných spojení (*lateral connections*). V každej aktivačnej mape, ktorá je výstupom jednotlivých vrstiev FPN siete, je rozmiestnených niekoľko tzv. *anchor boxov* v ktorých sú detegované jednotlivé objekty. Na rozdiel od niektorých vyššie spomenutých modelov sietí nie je potrebné definovať veľké množstvo *anchor boxov* v rôznych rozlíšeniach. To je už

Metóda	AP	AP^S	Rýchlosť (ms)	Rozlíšenie snímky
Faster R-CNN(VGG16) [27]	26.3		142.8	$\sim 1000 \times 600$
Faster R-CNN(FPN-ResNet101) [19]	36.2	18.2	172	$\sim 1300 \times 800$
Mask R-CNN-101 [13]	38.2	20.1	195	800×800
SSD513(ResNet101) [8]	31.2	10.2	125	512×512
YOLOv2 [25]	21.6	5.0	14	416×416
YOLOv3 [26]	33.0	18.3	51	608×608
RetinaNet50 [20]	32.5	13.9	73	500×500
RetinaNet101 [20]	39.1	21.8	198	800×800

Tabuľka 2.1: Porovnanie presnosti a rýchlosti vybraných konvolučných sietí na datasete *MS COCO*. AP označuje priemernú presnosť (*average precision*), AP^S označuje priemernú presnosť na triede malých objektov.



Obr. 2.6: Zjednodušená architektúra modelu RetinaNet: Základná sieť pre generovanie aktivačných máp v rôznom rozlíšení pozostáva z niekoľkých vrstiev ResNet siete (**A**) a FPN siete (*Feature Pyramid Network*) (**B**). Každú aktivačnú mapu následne spracováva dvojica podsietí (**C**) - klasifikačnej pre predikciu triedy a regresnej pre *bounding box* objektu.

implicitne dané rôznym rozlíšením aktivačných máp v každej výstupnej vrstve FPN siete. Autori preto definovali tieto obdĺžniky iba v troch rôznych pomeroch strán $[1:1, 1:2, 2:1]$ a troch veľkostiach $\{2^0, 2^{\frac{1}{3}}, 2^{\frac{2}{3}}\}$ [20]. V jednej aktivačnej mape existuje celkovo deväť rôznych *anchor boxov*. V súčte cez všetky aktivačné mapy FPN siete tieto obdĺžniky pokrývajú oblasti od 32 do 813 pixelov vo vstupnom obraze.

2.4.2 Klasifikácia a detekcia objektov

Základná FPN sieť produkuje celkovo 5 aktivačných máp P_3 až P_7, z ktorých každá je následne spracovaná dvoma konvolučnými podsieťami, viď obrázok 2.6. Prvá, tzv. klasifikačná podsieť rieši problém klasifikácie K tzv. *anchor boxov* zo všetkých aktivačných máp do N tried objektov. Počas trénovania je pre výpočet chyby klasifikácie na výstup tejto podsiete aplikovaná chybová funkcia FocalLoss. Druhá, tzv. regresná podsieť sa snaží predikovať ofsety súradníc (dx_1, dy_1, dx_2, dy_2) ohraničujúcich obdĺžnikov relatívne k stredu jednotlivých *anchor boxov*. Chyba predikcie týchto súradníc je počas trénovania určená pomocou funkcie *SmoothL1Loss* [20].

Výstupom týchto podsietí je $K \times N$, resp. $K \times 4$ predikcií. Z týchto všetkých predikcií je však väčšinou pomerne veľké množstvo duplicitných, teda pre jeden objekt jednej triedy je predikovaných niekoľko ohraničujúcich obdĺžnikov, preto je potrebné duplicity eliminovať. Podobne ako u vyššie spomenutých detektorov, jednou z možností je napríklad metóda *non-maxima supression*.

Pre výpočet chyby predikcie je potrebné pred začiatkom trénovania anotované dáta transformovať do vyššie spomenutých *anchor* boxov. V prípade, že medzi anotovaným obdĺžnikom a *anchor* boxom je pomer prieniku a zjednotenia, označovaný ako IoU ≥ 0.5 (*Intersection over Union*), je tomuto *anchor* boxu priradený identifikátor triedy objektu a štyri hodnoty určujúce rozdiel súradníc daného obdĺžnika a anotovaných súradníc. *Anchor* boxy kde IoU so žiadnym anotovaným objektom nie je väčší ako 0.4 sú označené ako pozadie a tie, ktoré majú najväčšie IoU v rozmedzí $\langle 0.4, 0.5)$ sú počas trénovania ignorované [20].

2.4.3 Chybová funkcia FocalLoss

Jedno-priechodové detektory musia riešiť problém s výrazne nerovnomerným rozdelením tried objektov. Tieto detektory typicky vyhodnocujú 10 000 až 100 000 kandidátnych regiónov (*anchor boxov*) v jednom obraze. Z týchto všetkých regiónov je však len malé množstvo regiónov popredia, teda hľadaných objektov, zvyšné reprezentujú pozadie. Tento nepomer spôsobuje predovšetkým dva zásadné problémy počas trénovania modelu:

- Väčšina oblastí je ľahko klasifikovateľná ako pozadie a teda neprinášajú žiaden progres v učení.
- Tieto ľahko klasifikovateľné regióny sa najviac podieľajú na hodnote celkovej chyby počas trénovania, čo vedie k neúčinným modelom s malou úspešnosťou. Inými slovami, model sa nedokáže naučiť detegovať objekty, pretože chyba spôsobená nimi je nevýrazná kvôli ich malému počtu.

Jedným z riešení, napr. použitým u vyššie spomenutých jedno-priechodových detektorov, je použitie techniky *hard negative mining*. K ďalším spôsobom riešenia tohto problému je úprava chybovej funkcie.

Chybová funkcia FocalLoss je navrhnutá tak, aby priradila nižšiu váhu tzv. jednoduchým príkladom objektov (objekty pozadia) a preto ich príspevok do celkovej chyby predikcie je malý v porovnaní s ich počtom. Inými slovami, táto chybová funkcia zameriava trénovanie modelu na pomerne malý počet objektov popredia. Táto funkcia ja založená na vzájomnej entropii (*cross entropy*) definovanej vzťahom 2.2 pre binárnu klasifikáciu.

$$\mathrm{CE}(p_t) = -\log(p_t) \tag{2.2}$$

kde p_t je definované vzťahom 2.3 kvôli zjednodušenému zápisu.

$$p_t = \begin{cases} p, & \text{ak } y = 1 \\ 1 - p, & \text{opačne} \end{cases} \tag{2.3}$$

Vo vyššie uvedenom vzťahu 2.3, označuje $y \in \{-1, +1\}$ skutočnú (*ground-truth*) triedu klasifikovaného objektu a $p \in \langle 0, 1\rangle$ je modelom predikovaná pravdepodobnosť, že objekt patrí do triedy $y = 1$ [20].

Chybová funkcia FocalLoss dopĺňa vzájomnú entropiu o tzv. modulovací faktor $(1-p_t)^\gamma$, s nastaviteľným exponentom $\gamma \geq 0$, ktorý upravuje pomer ktorým je tzv. jednoducho klasifikovateľným objektom znižovaná váha ich príspevku do celkovej chyby. Autori odporúčajú

nastaviť parameter $\gamma = 2$ [20]. Formálne je táto funkcia definovaná ako:

$$\mathrm{FL}(p_t) = -(1 - p_t)^{\gamma} \log(p_t) \tag{2.4}$$

Modulovací faktor teda znižuje príspevok ľahšie klasifikovaným objektov, napr. objektov pozadia, do celkovej chyby predikcie modelu a teda menej frekventovanejšie objekty majú väčší vplyv na priebeh celého učenia siete.

2.5 Sledovanie objektov v obraze

Cieľom sledovania objektov vo video zázname je predovšetkým získať ich polohu, prípadne odlíšiť jednotlivé objekty od seba a zostrojiť individuálne trajektórie pohybu. Ako bolo spomenuté v úvode tejto kapitoly, jedným z prístupov sledovania objektov je tzv. sledovanie podľa detekcií (*tracking by detection*). Týmto prístupom je každý snímok video spracovaný typicky v troch oddelených častiach — detekcie objektov, predikcie polohy z predchádzajúcich snímok a asociácie získaných detegovaných a predikovaných polôh objektov porovnaním ich podobnosti do jednotlivých trajektórií [2]. Predikciu polohy objektu v nasledujúcich snímkach je možné riešiť viacerými prístupmi, k najpoužívanejším prístupom patrí Kalmanov filter, časticový filter (*particle filter*), rekurentné neurónové siete alebo optický tok. Asociácia má za cieľ priradiť nové detekcie k existujúcim trajektóriám objektov porovnaním s predikciami získaných v predchádzajúcom kroku, prípadne vytvoriť nové trajektórie pre nové objekty v scéne.

Sledovacie algoritmy je možné okrem získania samotnej trajektórie a jej následnej analýzy využiť aj na urýchlenie celkového spracovania video záznamu. Výpočetne náročná detekcia objektov sa v takom prípade aplikuje iba na každú n-tú snímku videa alebo iba na začiatku videa. Vo zvyšných snímkach je poloha objektu získaná napríklad predikciou trajektórie, prípadne iným, efektívnejším algoritmom, ktorý sa snaží lokalizovať posunutý objekt v časti obrazu na základe podobnosti vybraných vizuálnych príznakov. Ako príklad takého sledovacieho algoritmu je možné uviesť systém GOTURN [15].

2.5.1 SORT

SORT (*Simple Online and Realtime Tracking*) [2] je príkladom online sledovacieho algoritmu viacerých objektov (*MOT*) navrhnutým predovšetkým pre použitie v aplikáciách bežiacich v reálnom čase. Online sledovací algoritmus má za úlohu zostaviť trajektórie pohybujúcich sa objektov asociovaním ich detekcií získaných zo súčasného a predchádzajúcich snímkov. Algoritmus SORT cieli na čo najvyššiu efektivitu spracovania, preto využíva oproti iným *state of the art* algoritmom výrazne jednoduchšie techniky. Pre výpočet predikcie pohybu a asociovanie detekcií do trajektórií využíva iba pozíciu ohraničujúceho obdĺžnika (*bounding box*) objektu a jeho veľkosť, na rozdiel od ostatných algoritmov, ktoré využívajú napríklad aj vizuálne vlastnosti objektov. SORT je taktiež menej robustný voči chybám detekcie, nerieši komplikované hraničné situácie akými sú zakrytie objektov a pod. Samotný detektor preto výrazne vplýva na celkovú úspešnosť sledovania objektov.

Pre aproximáciu zmeny polohy objektov medzi snímkami sa využíva tzv. konštantný lineárny model rýchlosti, ktorý je nezávislý od ostatných sledovaných objektoch a pohybe kamery. Aktuálny stav trajektórie je reprezentovaný ako:

$$\mathbf{x} = [u, v, s, r, \dot{u}, \dot{v}, \dot{s}]^T,$$

kde u a v označujú súradnice stredu objektu, s reprezentuje plochu a r pomer strán ohraničujúceho obdĺžnika. V prípade, že nejaká detekcia je priradená k trajektórií, aktuálny stav trajektórie sa aktualizuje pomocou tohto detegovaného obdĺžnika, kde rýchlostné komponenty sú optimálne vyriešené pomocou Kalmanového filtra. Naopak, pri chýbajúcej detekcii je stav predikovaný iba pomocou lineárneho rýchlostného modelu. Tento model je však príliš jednoduchý prediktor pre zachytenie komplikovaných trajektórií.

K asociovaniu nových detekcií do trajektórií sa využívajú predikované súradnice ohraničujúcich obdĺžnikov existujúcich trajektórií v aktuálnej snímke. Zostaví sa tzv. cenová matica (*assignment cost matrix*) medzi detegovanými a predikovanými obdĺžnikmi, v ktorej sú hodnoty určené ako pomer prieniku a zjednotenia (*IoU - Intersection over Union*). Matica je následne vyriešená pomocou tzv. Maďarského (Hungarian) algoritmu [2].

Detekcie, ktoré neboli priradené k žiadnej trajektórii sú použité pre inicializovanie nových trajektórií. Nová trajektória je však potvrdená až v prípade naakumulovania dostatočného počtu detekcií, kvôli eliminovaniu zbytočného sledovania tzv. falošných detekcií (*false positives*). Trajektórie sú ukončené v prípade chýbajúcich detekcií po dobu T_{lost} snímok.

Kapitola 3

Dáta

Spoločnosť CamVision s.r.o. poskytla pre účely tejto práce celkovo 10 anotovaných video záznamov futbalových stretnutí z rôznych štadiónov. Informácie o poskytnutých videách je znázornené v tabuľke 3.1. Celkovo videa obsahujú 142 minút anotovanej hry. Stretnutie je zaznamenané pomocou dvojice kamier vo vysokom rozlíšení, kde každá kamera zaznamenáva jednu polovicu ihriska. Záznam jednej kamery je navyše rozdelený na menšie časti, typicky v dĺžke piatich minút hry, predovšetkým kvôli zníženiu pamäťových nárokov pri spracovávaní videí. Ukážka dvojice záberov na celú hraciu plochy je možné vidieť na obrázku 3.1.

Názov záznamu	Trvanie (min)	Rozlíšenie snímky	Počet anotácií (ľavá/pravá kamera)	Počet extrahovaných anotácií
V1	3	4000×3000	135 : 147	282
T2	25	3840×2160	1208 : 1457	1333
V3	19	4000×3000	2617 : 3104	797
V4	24	4000×3000	3181 : 3915	907
T5	12	4000×3000	2719 : 4044	2029
T6	10	4000×3000	458 : 980	1438
T7	8	4000×3000	1083 : 978	1443
T8	19	4000×3000	2934 : 5405	2085
T9	12	3840×2160	3706 : 2198	1772
T10	10	4000×3000	837 : 890	1382

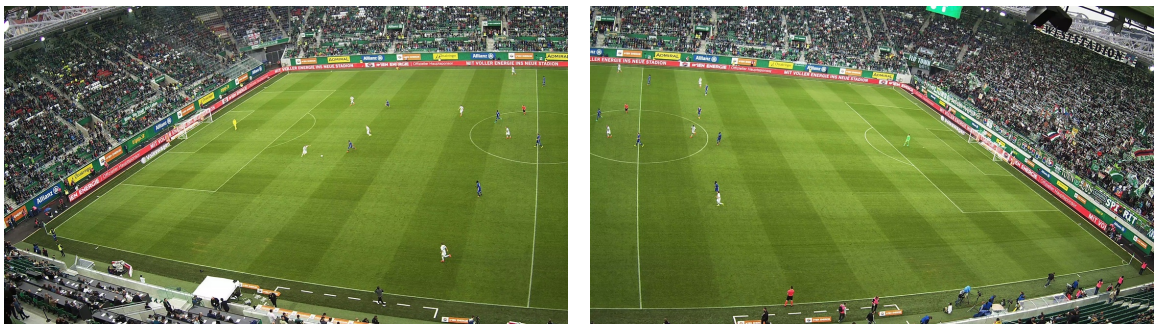
Tabuľka 3.1: Sumarizácia poskytnutých video záznamov a anotácií polôh hracej lopty. Jednotlivé videá sú nahrávané za odlišných svetelných podmienok a rôznych štadiónoch.

3.1 Analýza poskytnutých video záznamov

V dátovej sade sa okrem samotných video záznamov nachádzajú anotačné súbory. V nich je zaznamenaná presná poloha lopty vo vybranej snímke videa. Okrem samotnej polohy je zaznamenaná jej veľkosť, rýchlosť pohybu a prejdená vzdialenosť. V každej snímke videa je zaznamenaná vždy práve jedna futbalová lopta, teda tá, ktorá je aktuálne v hre. Lopty za hranicou hracej plochy sú ignorované. Pozícia lopty sa zaznamenáva iba počas aktívnej hry a iba v prípadoch, keď je viditeľná. Ostatné situácie kedy je hra prerušená, prípadne loptu drží hráč alebo je viditeľná iba čiastočne, sú ignorované. Príklady anotovaných lôpt

je možné vidieť na obrázkoch 3.3, kde sú zobrazené rôzne situácie, predovšetkým rôzna veľkosť anotovanej lopty a zhoršené vizuálne podmienky spôsobené napríklad rozmazaným obrazom alebo dymom svetlíc.

Samotné anotácie boli vytvorené pomocou interného programu vyvinutého spoločnosťou CamVision. Špecifikom týchto anotácií je, že nie sú anotované všetky za sebou idúce snímky. Poloha lopty je zaznamenaná iba vo vybraných situáciách, akými sú napríklad kontakt lopty a hráča, odrazenie lopty od zeme a pod. V prípade letu lopty vzduchom je celková trajektória rozdelená na menšie anotované úseky. Polohu v snímkach medzi jednotlivými anotáciami je však možné interpolovať.



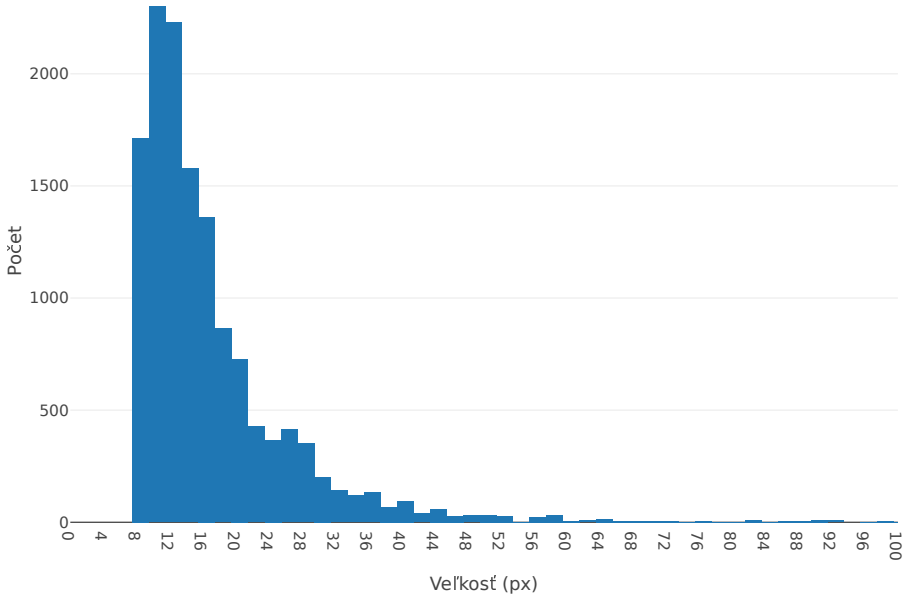
Obr. 3.1: Ukážka dát poskytnutých firmou CamVision s.r.o.. Na záberoch je možné vidieť pohľady z oboch kamier na jednotlivé polovice futbalového ihriska.

Poloha lopty $[rx, ry]$ je meraná relatívne voči ľavému hornému okraju záznamu kamery v pixeloch. Jej veľkosť, meraná v pixeloch, je určená v dvoch rozmeroch ako $[rw, rh]$, označujúcimi šírku a výšku. Anotácie sú uložené v XML súboroch zvlášť pre ľavý a pravý záznam hry. Sú členené do skupín organizovaných po tzv. udalostiach. Tie reprezentujú časti video záznamov, v ktorých je lopta jasne viditeľná, tzn. úsek hry, kedy je lopta na pozorovanej polovici hracej plochy. Každá udalosť pozostáva z jednotlivých anotácií lopty, z ktorých každá obsahuje niekoľko rôznych hodnôt. Pre účely tejto práce sa však používa iba 5 premenných, a to konkrétne časová známka anotácie, súradnice polohy lopty (rx, ry), výška (rw) a šírka (rh) lopty.

Analyzovaním poskytnutých dát je možné pozorovať niektoré vlastnosti anotovaných objektov, ktoré je možné ďalej využiť pri návrhu a optimalizácií algoritmu sledovania lopty v obraze alebo pri samotnom spracovaní video záznamov. Jednou z kľúčových vlastností je veľkosť hracej lopty, pričom priemerná veľkosť lopty je 16.65 px a najmenšia pozorovaná veľkosť je 8 px. V grafe 3.2 je možné pozorovať histogram veľkostí anotovaných futbalových lôpt. Priemerná rýchlosť pohybu lopty v anotovaných dátach je 425 px/s, resp. 17 px/snímok.

3.2 Príprava trénovacej a validačnej datovej sady

Pre trénovanie konvolučnej neurónovej siete je potrebné vytvoriť sadu snímok so známou polohou a veľkosťou futbalovej lopty. Tieto snímky sa z jednotlivých video záznamov extrahujú podľa časovej známky poznačenej v jednotlivých anotáciách. Vybrané snímky majú rovnaké rozmery, napríklad 512×512 pixelov a obsahujú loptu, ktorá sa nachádza kdekoľvek v danej snímke. Anotované súradnice lopty sú uložené vo formáte CSV v nasledujúcej podobe:



Obr. 3.2: Histogram velkostí anotovaných futbalových lôpt v poskytnutej datovej sade

```
val_data/V3L_1000646.png; 391.0; 209.0; 425.0; 243.0
```

kde je najprv uvedená cesta k súboru s výrezom nasledovaná súradnicami ohraničujúceho obdĺžnika. Súradnice sú určené relatívne k ľavému hornému okraju výrezu.

Dostupné videá, zosumarizované v tabuľke 3.1, boli rozdelené na trénovacie (`T2, T5, T6, T7, T8, T9, T10`) a validačné (`V1, V3` a `V4`). Vzhľadom v pomerne veľkej podobnosti za sebou idúcich anotácií a im odpovedajúcim snímkam videa, bola pre vytvorenie datovej sady použitá iba podmnožina všetkých anotovaných snímok. Zbytočne veľké množstvo veľmi podobných ukážok hracej lopty by malo negatívny vplyv na veľkosť datovej sady a dobu trénovania modelu siete za pomerne malý príspevok na celkovej úspešnosti detektoru. Napriek týmto obmedzeniam bolo vybraných celkovo 11482 snímok veľkosti 1536×1536 px určených pre trénovanie. Počty extrahovaných anotácií z jednotlivých videí je zosumarizovaný v tabuľke 3.1. Vo väčšine prípadov sa hracia lopta nachádza v strede obrazu, predovšetkým pre jednoduchšiu aplikáciu tzv. augmentácie dát popísanej nižšie.

Počas trénovania modelu prebieha tzv. augmentácia trénovacích dát. Tá ma za cieľ umelo zvýšiť variabilitu dát na ktorých sa učí model neurónovej siete, vďaka čomu sa zvyšuje jeho robustnosť a celková úspešnosť. Augmentácia pozostáva z troch operácií tzv. *ColorJitter*, horizontálne otočenie a náhodný výrez obrazu. Všetky tieto operácie sa uskutočňujú s určitou náhodnosťou. Najprv sa zvolí náhodný výrez trénovacej snímky tak, aby sa v ňom nachádzala celá anotovaná lopta. Implicitne má výrez veľkosť 512×512 px. Následne sa aplikuje s pravdepodobnosťou 50% tzv. horizontálne otočenie obrazu. Nakoniec sa náhodne upravia hodnoty jednotlivých farebných kanálov, prípadne jas snímky v tzv. operácií *ColorJitter*.

Pre validáciu modelu siete počas trénovania bolo vytvorených celkovo 1986 snímok veľkosti 768×768 px. V týchto snímkach je však poloha lopty určená náhodne. Na rozdiel od trénovacích dát, ktoré sú v každej iterácií učenia modelu vyššie popísaným spôsobom modifikované, sa validačné dáta neupravujú kvôli možnosti presne merať zlepšenie úspešnosti detekcie medzi jednotlivými epochami trénovania neurónovej siete.

Okrem týchto dvoch datových sád využitých počas trénovania sietí boli ručne vyselektované kratšie úseky anotovaných video záznamov určené pre validáciu systému na sledovanie lopty. Bolo vybraných celkovo 11 záznamov, v dĺžke od 20 do 100 sekúnd, pričom anotovaná lopta je viditeľná počas celej doby videí. Záznamy zachytávajú bežné prihrávky lopty, rohové kopy a autové vhadzovanie, výkop lopty brankárom alebo gólovú situáciu.

Obr. 3.3: Príklady anotovaných futbalových lôpt v rôznych veľkostiach a situáciách. Poloha lopty je znázornená žltou kružnicou, pričom celé číslo nad ňou reprezentuje veľkosť anotovaného objektu v pixeloch.

Kapitola 4

Návrh systému

Systém pre sledovanie lopty v športových videách má za cieľ analyzovať panoramatický záznam celej hracej plochy. V každej snímke videa musí detegovať a lokalizovať hraciu loptu a následne sledovať jej pohyb. Je potrebné spracovávať až dve videá vo vysokom 4K rozlíšení, ideálne v reálnom čase. Vzhľadom k priemernej veľkosti anotovanej lopty v datovej sade, viď podkapitolu 3.1, algoritmus musí byť schopný detegovať veľmi malé objekty, tzn. v priemere menšie ako 32 px. Predovšetkým kvôli týmto vlastnostiam nie je možné zefektívniť výpočet výrazným zmenšením rozlíšenia vstupných videí. Navrhnutý algoritmus musí preto rýchlo a čo najpresnejšie lokalizovať malú, rýchlo sa pohybujúcu hraciu loptu vo vysokom rozlíšení. Na základe pozorovaní samotnej kvality video záznamov je nutné, aby bol systém schopný potlačiť negatívny vplyv rozmazania obrazu, predovšetkým v okrajových častiach záberov, alebo spôsobený rýchlym pohybom samotnej lopty.

4.1 Štruktúra systému pre sledovanie lopty

Systém pre sledovanie lopty v športových video záznamov je možné rozdeliť do niekoľkých samostatných častí. Prvou časťou je načítanie a predspracovanie vstupov, nasledované samotnou detekciou objektov pomocou konvolučnej neurónovej siete a asociáciou detekcií do trajektórii. Nakoniec sa vyberie jedna výsledná trajektória, určujúca aktuálnu polohu hracej lopty v panoramatickej snímke hracej plochy.

Spracovanie vstupných video záznamov

Na vstupe sa očakávajú dve statické video záznamy, nahrávané dvoma fixnými kamerami, z ktorých každá nahráva polovicu hracej plochy. Ich zložením má vzniknúť panoramatický pohľad na celú hraciu plochu. Okrem toho sa na vstupe očakávajú kalibračné súbory, vhodné pre spätnú transformáciu obrazu spôsobenú distorciou šošoviek jednotlivých kamier. Ďalšou požiadavkou na vstupné videá je rovnaká frekvencia snímok a rovnaké rozlíšenie obrazu, predovšetkým kvôli eliminácii nadbytočného spracovania. Druhým vstupom systému je natrénovaná detekčná sieť. Tá je použitá pre samotnú detekciu lopty vo video zázname.

Detekcia a sledovanie objektov

Vzhľadom k tomu, že systém má spracovávať videá plne automaticky, sledovanie objektov musí byť preto riadené samotným detektorom objektov (*tracking by detection*). Princíp toho typu sledovania je popísaný v kapitole 2. Navrhnutý spôsob spracovania vstupných video záznamov je preto možné rozdeliť do nasledujúcich krokov:

- **Predspracovanie vstupných snímok** - Jednotlivé snímky video záznamov je nutné transformovať a upraviť do vhodného formátu pre spracovanie detektorom založenom na hlbokej konvolučnej neurónovej sieti.
- **Detekcia objektov** - V každej snímke videa je potrebné detegovať všetky objekty, v tomto prípade hracie lopty. Vzhľadom v vlastnostiam datovej sady predstavenej v kapitole 3 je potrebné, aby detektor dokázal spracovať čo najrýchlejšie snímky vo vysokom rozlíšení (veľkosti až 4K) a v nich čo najpresnejšie detegovať predovšetkým veľmi malé objekty (menšie ako 32 px).
- **Sledovanie objektov** - Detegovaným objektom je potrebné priradiť jedinečný identifikátor, teda priradiť detekcie do jednotlivých trajektórii využitím algoritmov popísaných v podkapitole 2.5.
- **Výber výslednej predikcie polohy lopty** - Očakávaným výstupom je vždy práve jedna lopta, preto je potrebné samotné trajektórie určitým spôsobom ohodnotiť a vybrať najvhodnejšieho kandidáta z aktívnych trajektórii. Ten zároveň určí polohu sledovanej lopty v aktuálnej snímke video záznamu.

Postprocessing

Výstupom systému je poloha lopty v každej snímke panoramatického video záznamu športového stretnutia. Tento výstup je nakoniec možné uložiť do súboru s časovou známkou konkrétnej snímky, prípadne vizualizovať graficky v samotnom videu.

4.2 Detekcia objektov

Spomenuté detektory nedokážu spracovávať vstupný obraz vo vysokom rozlíšení (viac ako 2000 px) v reálnom čase. Je preto potrebné navrhnúť efektívnejšiu architektúru zameranú predovšetkým na detekciu objektov veľkosti typicky od 8 px do 40 px, viď histogram veľkostí anotovaných lôpt 3.2. V rámci tejto práce bola preto upravená architektúra typu RetinaNet odstránením nadbytočných výpočtov, úpravou základnej siete generujúcej aktivačné mapy a zjednodušením klasifikačných a regresných podsietí.

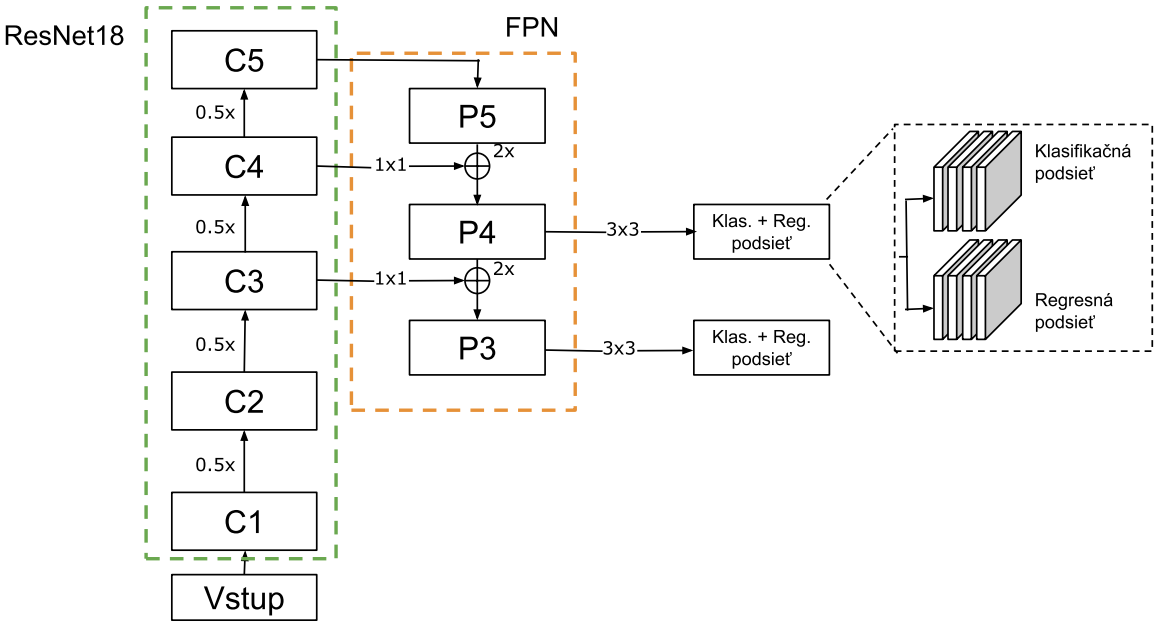
RetinaNet so základnou sieťou typu *FPN-ResNet50* popísaná v podkapitole 2.4.1 dokáže spracovávať RGB obraz veľkosti 2048×2048 px za približne 849.13 ms. Z výstupného tenzoru rozmerov $[785664, 5]$, teda 785664 predikcií rozmiestnených podľa tzv. *anchor boxov* (pri detekcii jedného objektu) je potrebné dekódovať finálne predikcie detegovaných objektov podľa preddefinovaných tzv. *anchor boxov*. Toto spracovanie trvá ďalších približne 14.53 ms. Táto sieť deteguje ohraničujúce obdĺžniky objektov vo veľkostiach 32 px až 512 px v deviatich rôznych pomeroch strán. Z analýzy dát v podkapitole 3.1 je zrejmé, že stačí detegovať iba štvorcové objekty do 64 px. Odstránením vrstiev P5, P6 a P7 a teda používaním iba dvoch aktivačným máp z vrstiev P3 a P4 pre regresiu a klasifikáciu objektov a zároveň úpravou tzv. *anchor boxov* iba na štvorce veľkosti 32×32 a 64×64 je možné zrýchliť výpočet siete o približne 29.97 ms a zredukovať výstupný tenzor na rozmery $[81920, 5]$ čím sa ušetrí pri dekódovaní ďalších približne 9 ms.

4.2.1 ResNet-18

Zmenou pomerne robustnej základnej siete typu *ResNet-50* na jednoduchšiu *ResNet-18* je možné výpočet siete urýchliť na 456.95 ms. Podrobnou analýzou doby spracovania jednotli-

vých vrstiev bolo zistené, že vrstvy patriace tzv. regresným a klasifikačným podsieťam, popísaným v podkapitole 2.4.2 zaberajú značnú časť celkovej doby výpočtu. Redukciou počtu kanálov z 256 na 32 v každej konvolučnej vrstve týchto podsietí je možné dosiahnuť ďalšie výrazné zrýchlenie. Táto varianta označovaná ako *LH-ResNet18* (*LightHead RetinaNet-ResNet18*) dosahuje rýchlosť spracovania 160.97 ms.

Navrhnutý model je možné vidieť v schéme 4.1, pričom jednotlivé vrstvy základnej siete *ResNet-18* sú podrobne popísané v tabuľke 4.1. Základná sieť je napojená na zvyšok FPN štruktúry obdobne, ako je tomu v pôvodnej verzii modelu RetinaNet, teda pomocou tzv. latentných 1×1 kovnolúcií a 3×3 konvolúcií produkujúcich finálne aktivačné mapy. Klasifikačné a regresné podsiete sa skladajú zo štyroch konvolučných vrstiev s jadrom $3 \times 3 \times 32$.



Obr. 4.1: Detailná štruktúra navrhnutého modelu LH-ResNet18.

4.2.2 MobileNetV2

Medzi ďalšie možnosti zefektívnenia výpočtu pôvodnej siete patrí využitie architektúr typu MobileNet, prípadne ShuffleNet, viď podkapitoly 2.2.2 a 2.2.3. Tieto klasifikačné siete je možné použiť pre extrakciu príznakov v základnej sieti výmenou za pôvodné ResNet vrstvy. V prvom prípade boli nahradené bloky C1 až C5 prvými 14 vrstvami architektúry *MobileNetV2*, ktoré sú usporiadané do štyroch skupín C1 až C4, na ktoré je napojený pomocou latentných vrstiev zvyšok tzv. *FPN* štruktúry. Detail navrhnutého modelu je možné vidieť v schéme 4.2. Podrobný popis základnej siete je popísaný v tabuľke 4.2.

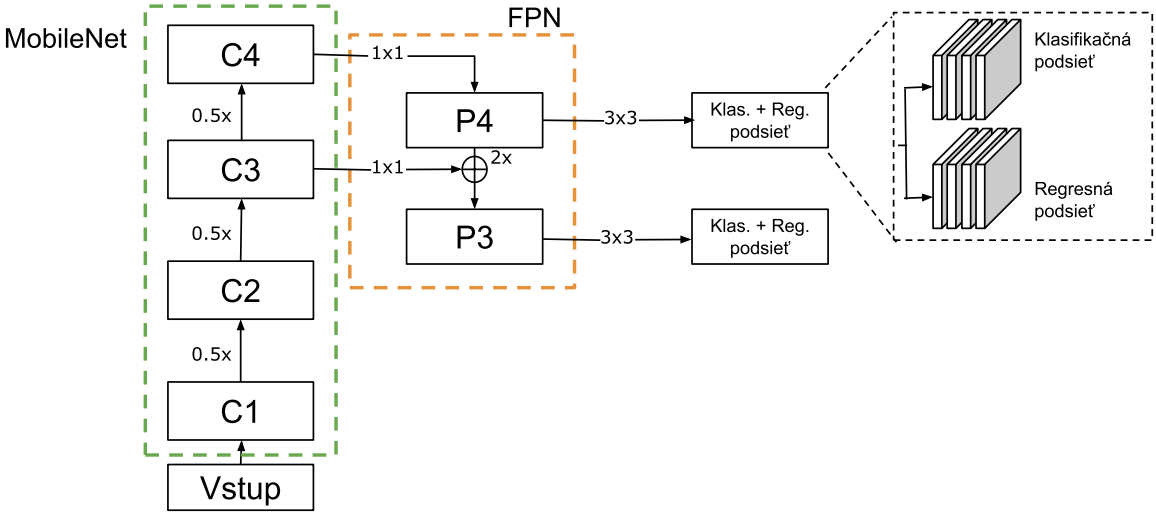
Aj v tomto prípade boli navrhnuté varianty s pôvodným počtom kanálov v regresných a klasifikačných podsieťach, ako aj ich odľahčené varianty, označené prefixom LH (*LightHead*) obsahujúce po 32 kanálov v ich konvolučných vrstvách. Výhodou použitia *MobileNet* siete je možnosť parametrizovať počet kanálov jednotlivých vrstiev základnej siete pomocou parametru `width_multiplier`, čím sa dá výrazne zredukovať počet potrebných parametrov siete a teda aj počet potrebných operácií. Táto redukcia však prinesie zníženie presnosti detekcie.

Vrstva	# iterácií	Veľkosť jadra	# filtrov	Stride
C1 (conv1)	1	$7 \times 7 \times 3$	64	2
C1 (maxPool)	1	3×3	1	2
C2	2	$3 \times 3 \times 64$	64	1
		$3 \times 3 \times 64$	64	1
C3	2	$3 \times 3 \times 64$	128	2/1
		$3 \times 3 \times 128$	128	1
C4	2	$3 \times 3 \times 128$	256	2/1
		$3 \times 3 \times 256$	256	1
C5	2	$3 \times 3 \times 256$	512	2/1
		$3 \times 3 \times 512$	512	1

Tabuľka 4.1: Detailne popísané vrstvy základnej siete typu *ResNet-18* naznačenej v schéme 4.1. Každá vrstva je nasledovaná tzv. *batch* normalizáciou a aktivačnou funkciou *ReLU*. Hodnota 2/1 znamená použitie *Stride* = 2 v prvej iterácií, v nasledujúcich je použitá hodnota 1. Navyše vo vrstvách $C2$ až $C5$ je použité reziduálne spojenie pomocou 1×1 konvolúcie.

Základný variant, označený ako *RetinaNet-MobileNet* dokáže spracovávať vstup veľkosti 2048×2048 px v priemere za 487.47 ms. Obsahuje 3, resp. 2 krát menej parametrov oproti sieti *RetinaNet-50*, resp. *RetinaNet-18*. Odľahčená verzia, tzv. *LH-MobileNet* v priemere spracuje tento vstup za 174.54 ms a má približne 15 krát menej parametrov oproti plnej verzii. Použitím parametru `width_multiplier` s hodnotu 0.25 je možné vstupný obraz spracovať za 357.71 ms v prípade varianty *RetinaNet-MobileNet-0.25*. Použitím *LH-MobileNet-0.25* je tento vstup spracovaný v priemere za 68.55 ms.

Okrem toho bola navrhnutá tzv. *FD-RetinaNet-MobileNet* inšpirovaná *FD-MobileNet* [23], ktorá v prvých konvolučných vrstvách rýchlejšie zníži rozlíšenie aktivačných máp a tým výrazne redukuje množstvo parametrov v nasledujúcich vrstvách siete. Jej *LightHead* varianta s parametrom `width_multiplier` = 0.25 dosahuje rýchlosť spracovania snímky 2048×2048 px za 32.89 ms.



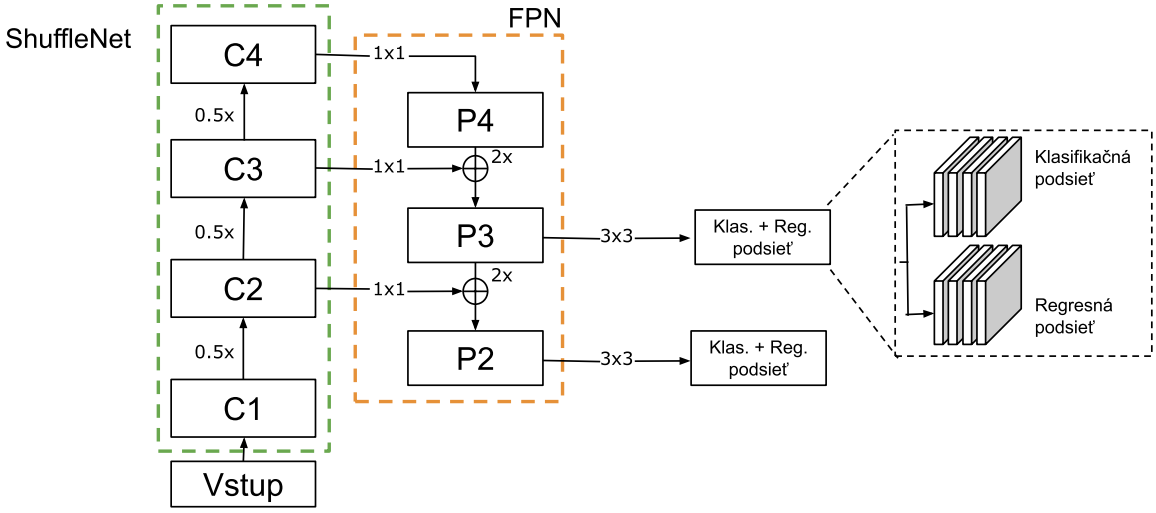
Obr. 4.2: Detailná štruktúra navrhnutého modelu LH-MobileNet.

Vrstva	# iterácií	Veľkosť jadra	# filtrov	Stride	Faktor expanzie
C1	1	$3 \times 3 \times 3$	32	2	-
C2	1	*InvResidual*	16	1	1
	2	*InvResidual*	24	2/1	6
C3	3	*InvResidual*	32	2/1	6
C4	4	*InvResidual*	64	2/1	6
	3	*InvResidual*	96	1	6

Tabuľka 4.2: Detailne popísané vrstvy základnej siete typu *MobileNet* naznačenej v schéme 4.2. Každá vrstva je nasledovaná tzv. *batch* normalizáciou a aktivačnou funkciou *ReLU6*. Hodnota 2/1 znamená použitie *Stride* = 2 v prvej vrstve, v nasledujúcich je použitá hodnota 1. Základný blok, tzv. invertovaný reziduál je navrhnutý podľa pôvodnej implementácie v [28].

4.2.3 ShuffleNetV2

V prípade použitia architektúry *ShuffleNetV2* sú vrstvy C1 až C5 pôvodnej základnej siete nahradené prvými 17 vrstvami modelu *ShuffleNetV2*, podrobne popísané v tabuľke 4.3. Zoskupené do štyroch blokov je ich výstup pomocou latentných vrstiev naviazaný na zvyšok architektúry. Detail navrhnutého modelu je možné vidieť v schéme 4.3. K tomuto modelu boli taktiež navrhnuté varianty s odľahčenými podsieťami detektoru ako aj redukované základné siete pomocou parametru `width_multiplier`. Model *RetinaNet-ShuffleNet* spracováva vstup veľkosti 2048×2048 px za približne 375.75 ms, jeho odľahčený variant označený ako *LH-ShuffleNet* dosahuje rýchlosti 81.33 ms a v prípade nastavenia parametru `width_multiplier` na polovicu je táto rýchlosť redukovaná až na 48.31 ms. Súhrnné porovnanie počtu parametrov, rýchlosti spracovania a veľkostí všetkých navrhnutých sietí je možné vidieť v tabuľke 4.4



Obr. 4.3: Detailná štruktúra navrhnutého modelu LH-ShuffleNet.

Vrstva	# iterácií	Veľkosť jadra	# filtrov	Stride
C1 (conv1)	1	$3 \times 3 \times 3$	24	2
C1 (maxPool	1	3×3	24	2
C2	4	*InvResidual*	116	2/1
C3	8	*InvResidual*	232	2/1
C4	4	*InvResidual*	464	2/1

Tabuľka 4.3: Detailne popísané vrstvy základnej siete typu *ShuffleNet* naznačenej v schéme 4.3. Každá vrstva je nasledovaná tzv. *batch* normalizáciou a aktivačnou funkciou *ReLU*. Hodnota 2/1 znamená použitie *Stride* = 2 v prvej vrstve, v nasledujúcich je použitá hodnota 1. Základný blok, tzv. invertovaný reziduál je navrhnutý podľa pôvodnej implementácie v [22].

4.3 Sledovanie objektov

Tento krok je zodpovedný za priradenie identifikátoru detegovaným objektom v jednotlivých snímkach využitím modifikovaného algoritmu *SORT* popísaného v podkapitole 2.5.1.

- **Aktualizácia trajektórií** - V každom kroku je aktualizovaný zoznam aktívnych trajektórií. Počas aktualizácie je pre každú trajektóriu predikovaná nová poloha sledovaného objektu na základe jeho doterajšieho pohybu.

- **Asociácia detekcií do trajektórií** - Detegované objekty sú priradené do trajektórií na základe korelácie s predikovanými polohami v predchádzajúcom kroku. Predikované polohy sú upravené podľa skutočných detekcií.

  Na rozdiel od originálnej implementácie algoritmu *SORT*, je pôvodná metrika určujúca hodnotu korelácie *IoU* (*Intersection over Union*) nahradená euklidovskou vzdialenosťou stredov ohraničujúcich obdĺžnikov objektov. Táto zmena vyplýva predovšetkým z analýzy trénovacích dát, konkrétne z priemernej veľkosti anotovanej lopty a rýchlosti jej pohybu medzi jednotlivými snímkami, viď 3.1.

- **Inicializácia nových trajektórií** - Detegované objekty, ktoré neboli asociované so žiadnou aktívnou trajektóriou, reprezentujú nový objekt v scéne. Pre všetky takéto objekty sú inicializované nové trajektórie. Trajektória je však potvrdená až po určitom počte asociovaných detekcií v nasledujúcich snímkach.

- **Odstránenie neaktívnych trajektórií** - V prípade, že k aktívnej trajektórií nie je v danej snímke priradená žiadna detekcia, je dekrementovaný čítač TTL (*time to live*). Polohu sledovaného objektu je v tomto prípade určená iba pomocou predikcie získanej v prvom kroku. V prípade, že čítač TTL dosiahne záporných čísel, trajektória je vyradená zo zoznamu aktívnych trajektórií.

Niektoré vlastnosti sledovacieho algoritmu, akými sú napríklad hodnota čítača TTL (*time to live*), počet potvrdení trajektórie, prah vzdialenosti detekcií a predikcií a iné je vhodné nastavovať parametricky a vyhodnotiť ich vplyv na výslednú úspešnosť detekcie. Rovnako je potrebné vyhodnotiť vplyv periódy spracovávania detektoru, teda prípady v ktorých sú detekcie dostupné iba každú n-tú snímku, kedy je predpoklad urýchlenia celkového spracovania video záznamov, avšak za cenu nižšej úspešnosti.

Názov siete	Počet parametrov	Veľkosť parametrov (MB)	Veľkosť modelu (GB)	Čas spracovania (ms)
ResNet50	35.66M	136.04	6.1	819.17
ResNet18	22.64M	86.37	8.28	456.96
MobileNet	11.22M	42.8	14.75	467.47
MobileNet-0.25	10.7M	40.81	6.99	357.71
FD-MobileNet	11.23M	42.83	6.5	351.14
FD-MobileNet-0.25	10.7M	40.82	4.13	320.25
ShuffleNet	12.22M	46.61	6.9	375.75
ShuffleNet-0.5	11.46M	43.73	4.99	339.36
LH-ResNet50	23.8M	90.8	8.59	513.92
LH-ResNet18	11.38M	43.43	5.7	160.97
LH-MobileNet	0.72M	2.73	12.19	174.54
LH-MobileNet-0.25	0.22M	0.82	4.43	68.55
LH-FD-MobileNet	0.72M	2.73	3.93	61.70
LH-FD-MobileNet-0.25	0.22M	0.82	1.56	32.9
LH-ShuffleNet	0.98M	3.74	4.33	81.34
LH-ShuffleNet-0.5	0.33M	1.27	2.41	48.31

Tabuľka 4.4: Sumarizácia parametrov navrhnutých sietí pri spracovávaní troj-kanálového tenzoru veľkosti 2048×2048 px. Všetky modely majú štruktúru založenú na FPN a dvoch podsieťach, pre skrátenie označenia je však uvedený iba typ základnej siete. Označenie LH (*LightHead*) pred názvom modelu označuje variantu s odľahčenými podsieťami pre klasifikáciu a regresiu kandidátnych objektov. Hodnoty boli merané na prenosnom počítači s procesorom Intel Core i7 s frekvenciou 2.8Ghz grafickou kartou NVIDIA GeForce GTX 1050Ti, Python3.6 s knižnicou Pytorch 1.0.1 a CUDA 9.1

4.3.1 Sledovanie objektov detegovaných vo výrezoch obrazu

V prípade úspešnej detekcie je na základe analýzy datovej sady možné predpokladať, že v nasledujúcej snímke pohybujúci objekt zmení svoju polohu iba do určitej maximálnej vzdialenosti. Pre detekciu tohto objektu nie je preto nutné opäť spracovávať celú snímku, ale použiť iba výrez obrazu určený polohou predchádzajúcej detekcie, prípadne predikovanou hodnotou trajektórie. Týmto spôsobom by teda malo byť možné čiastočne urýchliť celkový výpočet. Problém však v tomto prípade spôsobia nedetegované objekty, akými sú napríklad objekty v zákryte, ku ktorým v nasledujúcich snímkach bude chýbať odpovedajúci výrez obrazu a preto ani v týchto snímkach nebudú detegované čo povedie až k zlyhaniu sledovacieho algoritmu. Detekcie získané z celých vstupných snímok videí je preto potrebné počítať s určitou periódou a v ostatných snímkach používať pre detekciou objektov vyššie popísané menšie výrezy obrazu určené na základe predchádzajúcich detekcií.

4.3.2 Ohodnotenie trajektórií

Vzhľadom k detekcii viacerých kandidátnych objektov a zostrojeniu ich trajektórií pohybu je potrebné tieto trajektórie ohodnotiť, aby mohla byť jednoducho vybraná práve jedna najvyššie hodnotená trajektória, určujúca polohu lopty vo video zázname. Ohodnotenie musí brať do úvahy jednak istotu predikcie samotnej detekčnej siete, ale aj samotný po-

hyb sledovaného objektu. Počas hry sa predpokladá, že hracia lopta sa bude väčšinu času pohybovať rôznou rýchlosťou a náhle meniť smer pohybu. Naopak, u prípadných chybných detekcií, alebo lôpt nachádzajúcich sa mimo hracej plochy je skôr predpoklad, že sa bude jednať o statické objekty, preto je žiadúce takéto trajektórie ohodnotiť nižším skóre.

Výsledné ohodnotenie trajektórie sa určí ako

$$\sigma = \left(\frac{\sum_{i=0}^{N} s_i}{N} \right)^{10} * \frac{\sum_{i=0}^{M-1} \sqrt{(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2}}{M - 1} \tag{4.1}$$

kde N je celkový počet asociovaných detekcií, s je hodnota istoty detekcie i, x a y reprezentujú súradnice stredu objektu a M označuje celkovú dĺžku trajektórie, teda súčet predikovaných a asociovaných polôh objektu. Výsledné ohodnotenie trajektórie teda kombinuje priemerné skóre asociovaných detekcií objektov a priemernú prejdenú vzdialenosť. Umocnením priemerného skóre detekcií sa zníži vplyv menej istých pohybujúcich sa detekcií objektov na úkor nepohybujúceho sa objektu, ktorý má výrazne vyššiu hodnotu istoty detekcie a teda väčší predpoklad, že reprezentuje hraciu loptu.

Kapitola 5

Implementácia systému na sledovanie lopty

5.1 Použité technológie

Pre implementáciu systému pre detekciu futbalovej lopty v obraze bol zvolený jazyk python vo verzii 3.6 s využitím knižnice *PyTorch 1.0.1*[1]. *PyTorch* je voľne dostupná knižnica navrhnutá pre hlboké učenie s využitím výpočetnej sily grafických procesorov. Poskytuje vysoko abstraktné rozhranie pre efektívnu implementáciu algoritmov na GPU, operáciami nad n-rozmernými maticami, jednoduchú prácu s datasetom, umožňuje stiahnutie natrénovaných modelov sietí z tzv. model zoo a množstvo ďalších vecí. Počas trénovania siete boli využité ďalšie doplnky, predovšetkým *Ignite* a *Visdom*. *Ignite*[2] je nadstavba nad knižnicou *PyTorch* poskytujúca efektívnu implementáciu trénovacieho cyklu, metrík, logovania alebo napríklad ukladania kontrolných bodov modelu počas trénovania. *Visdom*[3] je nástroj, ktorý poskytuje jednoduchú vizualizáciu a export dát alebo štatistík trénovania v reálnom čase. Pre definovanie a jednoduché inštalovanie všetkých závislostí bolo vytvorené virtuálne python prostredie pomocou správcu prostredia *Conda*[4]. Všetky závislostí python skriptov je preto možné nájsť v súbore `enviroment.yml`. Pre využitie GPU akcelerácie výpočtov, musí byť v systéme dostupná platforma CUDA toolkit[5] verzie 9.1.

Ako súčasť práce bol vyvinutý aj C++ projekt pod Visual Studio 15 2017 Win64. Pre preklad bol použitý nástroj `cmake`. Pre prácu s videom a vizualizáciu výsledkov sledovania lopty bola použitá knižnica *OpenCV 3.4.1*. Pre akceleráciu výpočtu neurónových sietí na GPU bolo využité PyTorch C++ API, konkrétne jeho binárna distribúcia *LibTorch*[6]. Táto distribúcia zaobaľuje knižnicu *ATen* — pre tenzorové a matematické operácie, *TorchScript* rozhranie — pre *TorchScript JIT* prekladač a interpret, *C++ Frontend* podobný python rozhraniu a ďalšie rozšírenia. Pre implementáciu viac vláknovej aplikácie sa využila knižnica *TBB* (*Threading Building Blocks*)[7].

[1] https://pytorch.org/
[2] https://github.com/pytorch/ignite
[3] https://github.com/facebookresearch/visdom
[4] https://conda.io
[5] https://developer.nvidia.com/cuda-toolkit
[6] https://pytorch.org/cppdocs/
[7] https://www.threadingbuildingblocks.org/

5.2 Štruktúra riešenia

Výsledná práca pozostáva z niekoľkých python skriptov určených pre trénovanie, vyhodnotenie a testovanie navrhnutého systému a jeho častí a zo samostatného C++ projektu obsahujúceho optimalizovanú aplikáciu pre detekciu lopty v športových videách. Najdôležitejšou časťou navrhnutého systému je konvolučná neurónová sieť zodpovedná za detekciu lopty v snímke zápasu. V podkapitole 4.2 bolo navrhnutých niekoľko variantov detekčných sietí, z ktorých každý bolo potrebné natrénovať na dostupnej datovej sade popísanej v kapitole 3 a následne vyhodnotiť úspešnosť detekcií.

5.2.1 Modely neurónových sietí

Jednotlivé modely detekčných neurónových sietí sú implementované v samostatných triedach typu `nn.Module`. Modul `model_utils.py` sa stará o výber a instanciovanie rôznych druhov modelov siete a ich parametrov na základe argumentov príkazového riadku. Pre urýchlenie trénovania a lepšiu konvergenciu sa vo všetkých prípadoch používajú predtrénované váhy základných klasifikačných sietí na datasete *ImageNet*. V prípade siete *ResNet-50* a *ResNet-18* sú tieto váhy dostupné priamo v tzv. model zoo knižnice *PyTorch*. Pri *MobileNetV2* a *ShuffleNetV2* bolo potrebné najprv transformovať zverejnené váhy natrénovaných modelov do vhodného formátu, ktorý bolo je následne možné jednoducho načítať v implementovaných triedach modelov.

Všetky implementované modely sa skladajú z dvoch oddelených častí, a to základnej siete určenej pre výpočet aktivačných máp a dvoch podsietí pre klasifikáciu a regresiu boxov. Štruktúru siete je navyše možné parametrizovať a to buď definovaním použitej základnej siete alebo výberom z dvoch variantov podsietí — štandardnej a tzv. odľahčenej označovanej ako *LightHead*. Základná sieť typu *MobileNet*, resp. *ShuffleNet* je navyše parametrizovateľná argumentom `width-multiplier` s možnými hodnotami `1.0, 0.5, 0.25`, resp. `1.0, 0.5`.

5.2.2 Načítanie trénovacích dát

Vzhľadom k veľkému množstvu trénovacích dát nie je možné tieto dáta načítať naraz kvôli obmedzenému množstvu dostupnej operačnej pamäte. Knižnica *PyTorch* však poskytuje triedu `DataLoader`, ktorá implementuje tzv. generátor, vďaka ktorému je možné načítavať jednotlivé dáta z celého datasetu postupne. Načítanie vlastného datasetu je implementované v podtriede typu `Dataset`, ktorá je zodpovedná za načítavanie jednotlivých snímok a im priradeným anotáciám polohy lopty. Veľkosť anotovaných lôpt je pred ďalším spracovaním upravená na minimálnu veľkosť 32 px, čo je dané štruktúrou siete a veľkosťou navrhnutých tzv. *anchor boxov*. Snímky a ich anotácie sa ďalej augmentujú podľa pravidiel popísaných v podkapitole 3.2 a transformujú na objekty typu `tenzor`, ktoré umožňujú efektívny výpočet na grafickej karte. Navyše, kvôli použitiu predtrénovaných modelov sietí, bolo potrebné jednotlivé snímky normalizovať hodnotami priemeru a štandardnej odchýlky datasetu *ImageNet*.

Samotné trénovanie prebieha po tzv. skupinách (z angl. *batch*), ktorých veľkosť je parametrizovateľná. Typické hodnoty sú od 12 až do 48 snímok v skupine, v závislosti na veľkosti dostupnej pamäti grafickej karty. V každej skupine je potrebné zarovnať snímky na rovnaké rozmery a zakódovať anotované ohraničujúce obdĺžniky (z angl. *bounding box*) podľa tzv. *anchor boxov*. Okrem toho táto trieda datasetu implementuje zamiešanie položiek datasetu.

5.2.3 Trénovanie siete

Trénovanie modelu neurónovej siete spočíva v iteratívnom prechode trénovacou sadou snímok s anotovanými polohami hracej lopty. Tento trénovací cyklus bol implementovaný využitím vyššie spomenutej knižnice *Ignite*. Okrem toho, táto knižnica umožňuje odchytenie významných udalostí počas trénovania modelu, akými sú napríklad ukončenie iterácie či epochy a pod. Na tieto udalosti bolo naviazané logovanie, vizualizovanie štatistík, validovanie a ukladanie natrénovaného modelu. Princíp trénovacieho cyklu je stručne popísaný v podkapitole 2.1.

Ako chyba klasifikácie tzv. *anchor boxov* počas trénovania modelu bola použitá chybová funkcia *FocalLoss* bližšie popísaná v podkapitole 2.4.3. Chybu predikcie súradníc ohraničujúcich obdĺžnikov objektov je určená funkciou *SmoothL1Loss*. Súčet týchto dvoch chýb udáva celkovú chybu predikcie modelu, podľa ktorej sa následne upravujú hodnoty parametrov s využitím optimalizátoru Adam. Jednotlivé premenné trénovania, akými sú miera učenia, počet epoch a veľkosť skupiny dát (*batch size*) sú parametrizovateľné.

Na konci každej epochy sa vyhodnotí celková chyba a úspešnosť detekcie natrénovaného modelu na validačnej dátovej sade, viď rozdelenie dát v kapitole 3. Okrem toho sa uloží natrénovaný model, optimalizátor a stav trénovania. Takto uložený natrénovaný model neurónovej siete je možné následne jednoducho načítať v detektore lopty vo videu, prípadne použiť pre dotrénovanie na ďalších dátach.

5.2.4 Sledovací algoritmus

Pre sledovanie detegovaných objektov boli implementované dve varianty v samostatných triedach, popísane v podkapitole 4.3. Každá trieda je zodpovedná pre získanie predikcí zo snímok a následne využitím algoritmu *SORT* sú získané jednotlivé trajektórie pohybujúcich sa objektov. Pre implementáciu Kalmanovho filtra a tzv. Maďarského algoritmu boli použité knižnice `filterpy`, resp. *sklearn.utils.linear_assignment*. Samotná implementácia algoritmu *SORT* je založená na pôvodnom zverejnenom riešení [8].

Pomocou parametru `period` je možné urýchliť výpočet, pričom perióda i znamená, že pre získanie detekcií je použitý iba každá i-tá snímka. Parametre `max_age` a `min_hits` sú implicitne nastavené na hodnoty 10, resp. 3. Prah minimálnej vzdialenosti pre asociáciu detekcií bol zvolený na hodnotu 90 px.

5.2.5 Evaluácia

Pre potreby vyhodnotenia systému a jeho častí bolo naimplementovaných niekoľko skriptov. Video záznamy použité pre vyhodnotenie sú zaznačené v súbore, kde je uložená cesta video záznamu, odpovedajúcemu anotačnému súboru, začiatku a konca evaluácie a špeciálneho ofsetu začiatku videa voči odpovedajúcim anotáciám. Vzhľadom k tomu, že detekcia objektov vo videách s vysokým rozlíšením je pomerne časovo náročná operácia, rýchlosť spracovania je tiež výrazne ovplyvnená testovanou konvolučnou sieťou, detekcie sa ukladajú do súboru, ktorý sa používa opakovane pri ďalšom vyhodnotení. Ukladanie detekcií do súboru je výhodné pri evaluácií systému s rôznym nastavením jednotlivých parametrov, pri použití rovnakého detektoru. Samotné detekcie sú uložené vo formáte CSV.

Pri vyhodnotení samotného detektoru sa načítavajú uložené detekcie a vypočíta sa presnosť voči ručne anotovaným polohám lopty v ich odpovedajúcich snímkach, teda nepoužívajú sa interpolované anotácie počas vyhodnotenia. Počítané metriky sú popísané v podka-

[8] https://github.com/abewley/sort

pitole 6.2. Evaluátor sledovacieho algoritmu navrhnutého v podkapitole 4.3 rovnako používa uložené detekcie v súbore na základe ktorých zostrojí a vyhodnotí výsledné trajektórie. Opäť sa vyhodnocujú iba anotované snímky videí. Skript je okrem toho možné parametrizovať nastavením tzv. periódy získavania detekcií, maximálneho počtu použitých detekcií či hodnoty prahu ich skóre.

Okrem toho sú implementované ekvivalentné skripty pre tzv. online vyhodnotenie, ktoré získavajú detekcie volaním zvolenej detekčnej siete. Týmto spôsobom je možné vyhodnotiť variantu sledovacieho algoritmu navrhnutého v podkapitole 4.3.1, ktorý sleduje objekty získané vo výrezoch snímok videí.

5.2.6 Výsledný systém pre detekciu lopty

Výsledný systém na detekciu lopty v zázname futbalového stretnutia implementovaný v jazyku Python postupne spracováva obe videá jednotlivých polovíc hracej plochy. V každom kroku najprv konkatenuje oba snímky videí pre vytvorenie celého panoramatického záberu, v ktorom sa pomocou vyššie popísaného algoritmu detegujú objekty a zostroja sa odpovedajúce trajektórie. Pre každý snímok sa aktuálne dostupné trajektórie zoradia podľa skóre určeného zo skóre detekcií a prejdenej vzdialenosti sledovaných objektov. Z takto zoradených trajektórií sa vyberie najlepšia ohodnotená, na základe ktorej sa určí predikovaná poloha hracej lopty.

5.2.7 Urýchlenie spracovania implementáciou v jazyku C++

Vzhľadom k tomu, že jazyk python nie je najvhodnejšou voľbou v aplikáciách v ktorých je kladený dôraz na rýchlosť spracovania, bola implementovaný efektívnejší variant výsledného systému v jazyku C++. Model neurónovej siete implementovaný a natrénovaný využitím knižnice *PyTorch* je potrebné pred použitím v C++ projekte exportovať do reprezentácie nezávislej na jazyku Python. Pre tento účel bol vytvorený skript využívajúci modul *TorchScript*, ktorý definuje podmnožinu jazyka Python obsahujúcu všetky vstavané tenzorové operácie knižnice *PyTorch*. Pomocou neho sa daný model siete prevedie na tzv. objekt *ScriptModule* a serializuje do súboru. Takto serializovaný model natrénovanej siete je následne možné načítať a používať pomocou C++ knižnice *LibTorch*.

V rámci C++ projektu boli naimplementované celkovo dve aplikácie, z ktorých prvá je presným ekvivalentom výslednej Python aplikácie popísanej v podkapitole 5.2.6. Analýzou behu jednotlivých častí programu bola naimplementovaná druhá aplikácia, v ktorej bolo spracovanie rozdelené do samostatných vláken, čo umožnilo lepšie využiť dostupné zdroje a celkovo urýchliť celý systém. Úzkym hrdlom však stále ostáva časovo najnáročnejšie získavanie detekcií objektov v obraze. Celkový čas spracovania je preto v prípade spracovávania každej snímky videa definovaný časom potrebným na získanie výsledkov siete. V prípade nastavenia menšej frekvencie spracovania snímok neurónovou sieťou, je možné spracovanie urýchliť, avšak za cenu nižšej presnosti, viď vyhodnotenie v podkapitole 6.3.2.

Kapitola 6

Testovanie a vyhodnotenie

Testovanie a vyhodnotenie navrhnutého systému prebehlo na datovej sade popísanej v kapitole 3. Vyhodnotenie bolo rozdelené na samostatné vyhodnotenie hlbokých neurónových sietí popísaných v podkapitole 4.2 a vyhodnotenie algoritmu sledovania hracej lopty vo video zázname navrhnutým v podkapitole 4.3. Experimenty boli zamerané na zhodnotenie vplivu rôznych nastavení jednotlivých parametrov systému a použitých detekčných sietí na úspešnosť sledovania a rýchlosť spracovania snímok vo vyššom rozlíšení. Kvôli snahe o čo najrýchlejšie spracovania, ideálne spracovanie v reálnom čase, boli testované iba tzv. rýchlejšie modely navrhnutých sietí, konkrétne varianty označené prefixom LH (*LightHead*), viď tabuľku 4.4.

6.1 Spôsob trénovania modelov sietí

Navrhnuté neurónové siete boli natrénované na trénovacej datovej sade s využitím tzv. augmentácií popísaných v podkapitole 3.2. Parametre trénovania boli vo väčšine prípadov nasledovné: počet epoch bol 40, kedy jedna epocha značí prechod celou trénovacou sadou, veľkosť skupiny (*batch size*) bol v rozmedzí od 24 do 48 snímok veľkosti 512 px, miera učenia (*learning rate*) bola nastavená na hodnotu 1e-5. Na konci každej epochy bol aktuálny natrénovaný model vyhodnotený na validačnej sade, pre overenie pretrénovania siete. Pre zlepšenie úspešnosti detekcie objektov v snímkach v menšom rozlíšení boli natrénované niektoré modely sietí s identickými parametrami trénovania avšak na datovej sade v 2/3 rozlíšení. Tieto varianty sú v ďalšom texte označené *.

Trénovanie prebiehalo na stolnom počítači s procesorom Intel Core i7 s frekvenciou 3.8 GHz, operačnou pamäťou veľkosti 16 GB a grafickou kartou NVIDIA GeForce GTX 1060 s 6 GB grafickou pamäťou.

6.2 Metriky použité pri vyhodnotení

Presnosť (*precision*) udáva ako presné sú predikcie detektoru, teda je možné ju vyjadriť ako pomer počtu správnych pozitívnych predikcií voči všetkým predikciám modelu, formálne zapísané ako

$$\rho = \frac{tp}{tp + fp} \tag{6.1}$$

kde tp označuje počet správnych predikcií (*True Positive*) a fp reprezentuje chybné predikcie (*False Positive*).

Recall naopak vyjadruje pomer počtu správnych pozitívnych predikcií voči očakávaným a je možné ho vyjadriť rovnicou

$$r = \frac{tp}{tp + fn} \tag{6.2}$$

kde fn označuje počet nedetegovaných objektov (*False Negative*).

Pre určenie správnosti detekcie objektov sa typicky používa metrika IoU (*Intersection over Union*), niekedy označovaná aj ako Jaccardov index. Táto metrika určuje pomer medzi prienikom a zjednotením anotovaného a predikovaného *bounding boxu*. Formálne je vyjadrená rovnicou 6.3.

$$\phi = \frac{R_1 \cap R_2}{R_1 \cup R_2} \tag{6.3}$$

kde R_i je plocha tzv. *bounding boxu* objektu i. Predikcia je považovaná za správnu (*True positive*), ak $\phi > \tau$, kde τ je hodnota prahu najčastejšie jedna z hodnôt $\{0.1, 0.3, 0.5, 0.7, 0.9\}$. V opačnom prípade je detekcia považovaná za tzv. *False positive*.

Priemernú presnosť AP je možné definovať aj ako plochu pod tzv. *precision-recall* krivkou. V praxi sa počíta ako priemer presností definovaných pre hodnoty *recall* $r \in \langle 0, 1 \rangle$ [7], matematicky definovaná vzťahom 6.4.

$$\mathrm{AP} = \sum_{r=0}^{1} (r_{n+1} - r_n) \rho_{interp}(r_{n+1}) \tag{6.4}$$

kde $\rho_{interp}(r_{n+1})$ je

$$\rho_{interp}(r_{n+1}) = \max_{\tilde{r}:\tilde{r} \geq r_{n+1}} \rho(\tilde{r})$$

a $\rho(\tilde{r})$ je dosiahnutá presnosť pri hodnote *recall* $\tilde{r}$ [6, 7].

6.2.1 Vyhodnotenie sledovacieho algoritmu

Najčastejšie používané metriky pre vyhodnotenie sledovania viacerých objektov sú *MOTP* a *MOTA* [1]. Metrika *MOTP* (*Multiple Object Tracking Precision*) je definovaná ako priemerné prekrytie všetkých objektov vo všetkých snímkach, berúc do úvahy rôzny počet viditeľných objektov v jednotlivých snímkach. Je definovaná ako

$$\mathrm{MOTP} = \frac{\sum_{i,t} \phi_{i,t}}{\sum_t c_t} \tag{6.5}$$

kde c_t je počet pozitívnych detekcií objektov v snímke t. V prípade sledovania jedného objektu ($c_t = 1$), ako je to v tejto práci, je možné túto metriku zjednodušiť na tzv. priemerné prekrytie, definované ako

$$\overline{\phi} = \sum_{t=1}^{N} \frac{\phi_t}{N} \tag{6.6}$$

Je však dôležité podotknúť, že táto metrika vyhodnocuje presnosť lokalizácie sledovaného objektu a preto neprináša veľa informácií o kvalite sledovania ako takej.

Na vyjadrenie úspešnosti sledovania objektov sa využíva metrika *MOTA* (*Multiple Object Tracking Accuracy*), definovaná ako

$$\mathrm{MOTA} = 1 - \frac{\sum_t (fn_t + fp_t + id\_sw_t)}{\sum_t g_t} \tag{6.7}$$

kde g_t reprezentuje počet vyskytujúcich sa objektov v snímke t, fn_t, fp_t a id_sw_t označujú postupne počet tzv. *false negatives*, počet *false positives* a počet tzv. výmen identít. V prípade sledovania iba jedného objektu ($g_t = 1, id\_sw_t = 0$), ktorého poloha je určená v každej snímke ($fp_t = 0, fn_t \in \{0, 1\}$) je možné metriku vyjadriť ako pomer počtu správne sledovaných snímok voči všetkým snímkam, viď rovnicu 6.8.

$$\text{Accuracy} = 1 - \frac{\sum_{t=1}^{N} fn_t}{N} = \frac{\sum_{t=1}^{N} tp_t}{N} \tag{6.8}$$

6.3 Vyhodnotenie

Navrhnutý systém a jeho varianty boli vyhodnotené na datovej sade popísanej v kapitole 3. Vyhodnotenie prebehlo na celkovo 11 videách zachytávajúcich rôzne situácie počas hry, viď. podkapitolu 3.2. Samotné videá boli spracovávané v troch rôznych rozlíšeniach, konkrétne 4000×3000 px, 2560×1920 px a 2048×1536 px, predovšetkým pre vyhodnotenie vplyvu zmenšenia objemu dát na rýchlosť a presnosť systému.

Vyhodnotenie prebiehalo na rozdiel od trénovania modelov sietí na prenosnom počítači s procesorom Intel Core i7 s frekvenciou 2.8 GHz, operačnou pamäťou veľkosti 32 GB a grafickou kartou NVIDIA GeForce GTX 1050Ti s 4 GB grafickou pamäťou.

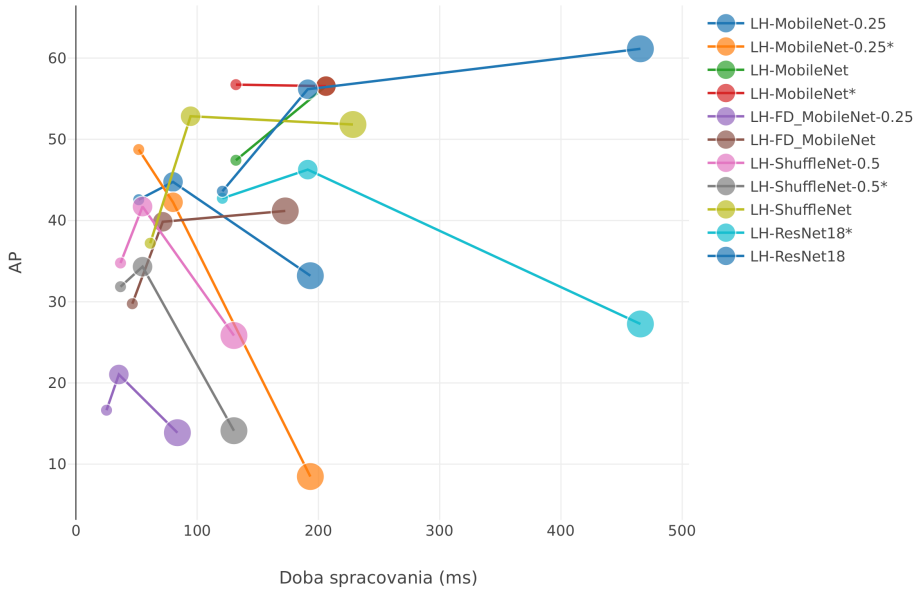
6.3.1 Vyhodnotenie natrénovaných neurónových sietí

Pre vyhodnotenie úspešnosti detekcie natrénovaných modelov neurónových sietí bola použitá metrika AP (*Average Precision*) definovaná vzorcom 6.4 v troch rôznych nastaveniach prahu prekrytia, konkrétne $\tau \in \{0.1, 0.3, 0.5\}$. Vo výsledkoch sú tieto merania označené ako AP_{10}, AP_{30}, resp. AP_{50}. Pri vyhodnotení sa do úvahy bralo iba desať detekcií s najvyšším skóre. Súhrnné výsledky je sú uvedené v tabuľke 6.1, pričom najlepšie výsledky v jednotlivých meraniach sú zvýraznené tučným písmom. Zároveň sú tieto výsledky, zvlášť pre jednotlivé metriky zobrazené v grafoch 6.2 a 6.3.

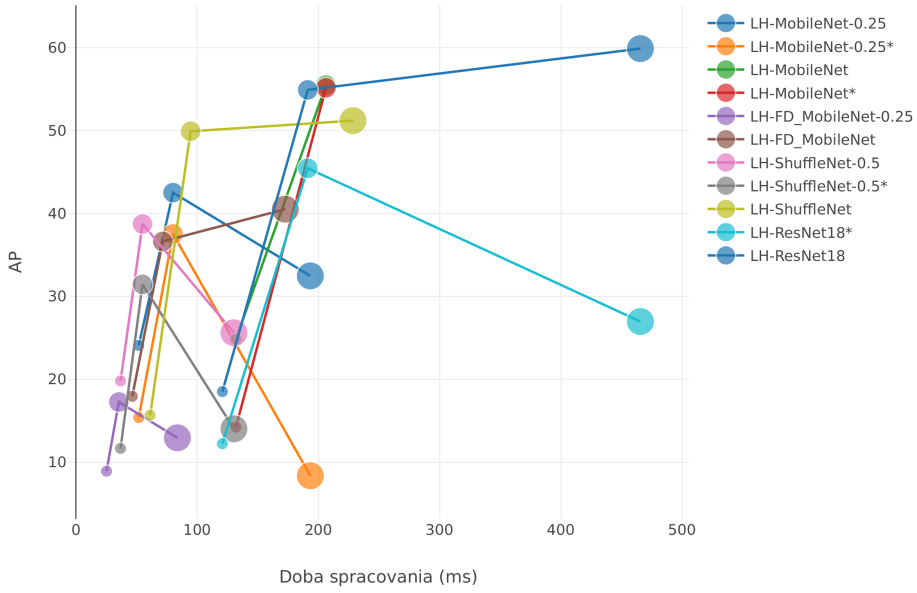
	4000×3000				2560×1920				2048×1536			
Názov siete	AP_{10}	AP_{30}	AP_{50}	Čas (ms)	AP_{10}	AP_{30}	AP_{50}	Čas (ms)	AP_{10}	AP_{30}	AP_{50}	Čas (ms)
LH-ResNet18	**61.14**	**59.88**	**53.77**	465.5	56.17	54.9	11.83	191.2	43.6	18.53	0.66	120.8
LH-ResNet18*	27.26	26.97	24.12	465.5	46.27	45.45	8.38	191.2	42.74	12.25	0.48	120.8
LH-MobileNet	–	–	–	–	**56.55**	**55.47**	14.10	206.2	47.42	**24.84**	1.09	132
LH-MobileNet*	–	–	–	–	56.54	55.15	5.99	206.2	**56.73**	14.24	0.23	132
LH-MobileNet-0.25	33.20	32.48	30.33	193.3	44.75	42.51	**15.93**	80.1	42.56	24.09	0.68	51.8
LH-MobileNet-0.25*	8.48	8.39	7.22	193.3	42.24	37.53	6.31	80.1	48.74	15.39	0.49	51.8
LH-FD-MobileNet	41.18	40.52	24.91	172.7	39.85	36.63	12.49	71.6	29.76	17.95	0.83	46.5
LH-FD-MobileNet-0.25	13.87	12.96	8.48	83.7	21.04	17.28	4.24	35.4	16.64	8.93	0.33	25.3
LH-ShuffleNet	51.82	51.19	47.75	228.5	52.84	49.91	10.48	94.6	37.21	15.68	0.29	61.3
LH-ShuffleNet-0.5	25.84	25.62	24.24	130.3	41.71	38.74	12.36	55	34.77	19.81	**1.52**	36.8
LH-ShuffleNet-0.5*	14.12	14.04	13.42	130.3	34.31	31.46	5.31	55	31.85	11.68	0.69	36.8

Tabuľka 6.1: Sumarizácia dosiahnutých výsledkov jednotlivých natrénovaných sietí na vybraných validačných video záznamoch. Modely označené symbolom '*' boli trénované na datovej sade v 2/3 rozlíšení.

Ako je možné vidieť, znižovaním rozlíšenia snímok vo väčšine prípadov klesá presnosť detekcií. Je to možné pozorovať predovšetkým pri metrike AP_{50}. Pre adresovanie tohto problému boli natrénované rovnaké modely sietí na datovej sade v 2/3 rozlíšení, označené

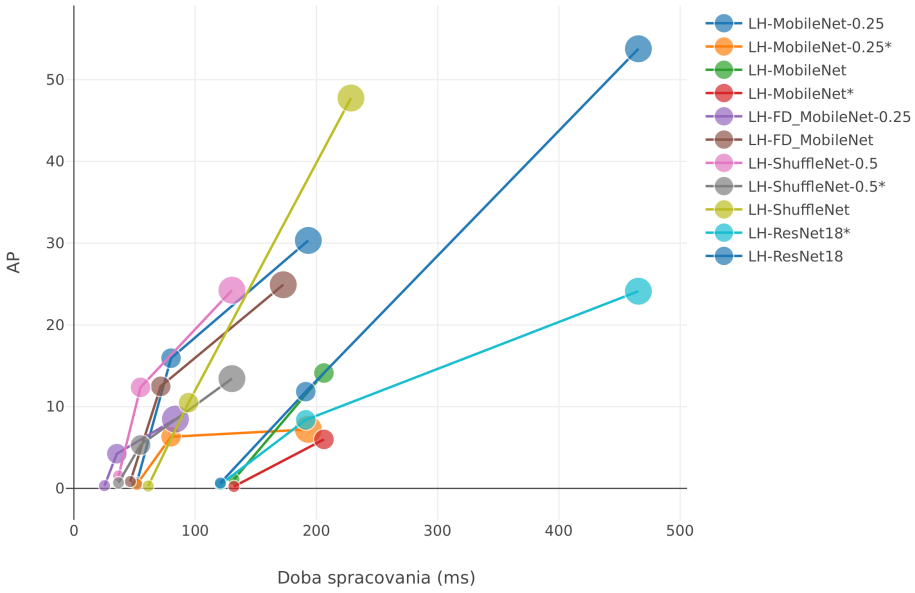


(a)



Obr. 6.1

Obr. 6.2: Porovnanie priemernej presnosti metrík (a) AP_{10}, (b) AP_{30} a rýchlosti jednotlivých neurónových sietí pri spracovávaní snímok v troch rozlíšeniach (2048:1536, 2560:1920 a 4000:3000), naznačených rôznou veľkosťou bodov. Varianty označené '*' boli trénované na datovej sade v 2/3 rozlíšení.

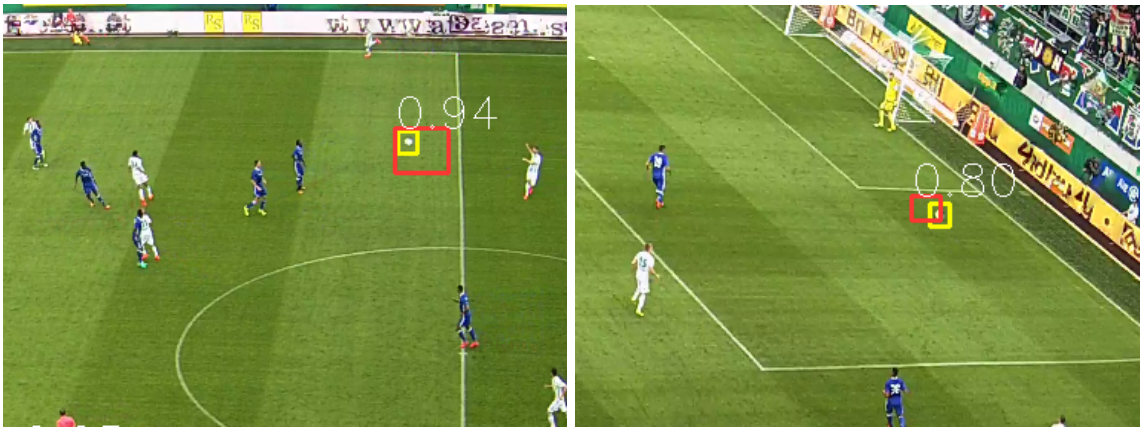


Obr. 6.3: Porovnanie priemernej presnosti metriky AP_{50} a rýchlosti jednotlivých neurónových sietí pri spracovávaní snímok v troch rozlíšeniach (2048:1536, 2560:1920 a 4000:3000), naznačených rôznou veľkosťou bodov. Varianty označené '*' boli trénované na datovej sade v 2/3 rozlíšení.

symbolom '*', ktoré v niektorých prípadoch, napríklad v prípade *LH-MobileNet* dokázali zlepšiť priemernú presnosť o približne 9 percentuálnych bodov v metrike AP_{10}. Na druhej strane však takto natrénované modely dosahujú výrazne nižšiu úspešnosť pri detekcii objektov v snímkach v plnom rozlíšení. Na základe získaných výsledkov by preto bolo vhodné rozšíriť implementované augmentácie obrazu počas trénovania siete o generovanie snímok v rôznom rozlíšení a pretrénovať navrhnuté neurónové siete.

Rovnako je možné pozorovať očakávané zníženie presnosti detekcie pri použití modelov obsahujúcich menšie množstvo parametrov, teda varianty *MobileNet-0.25* a *ShuffleNet-0.5*. Na druhej strane však tieto modely dokážu spracovať vstup za približne polovičný čas.

Obrázky 6.5 ukazujú príklady úspešnej detekcie hracej lopty počas futbalového zápasu. Ako je možné vidieť, detektor dokáže pomerne presne lokalizovať loptu rôznej veľkosti a vzdialenosti od kamery. Navyše k samotnému ohraničujúcemu obdĺžniku detegovanej lopty je uvedené skóre detekcie, ktoré je v rozmedzí od 0 do 1. V prípade dobre viditeľnej lopty je toto skóre blízke hodnote 1, čo reprezentuje vysokú istotu detekcie. V prípade detekcie v snímkach s menším rozlíšením je pomerne častým problémom nie úplne presná lokalizácia objektu. To má dopad na hodnotu prekrytia detekcie a anotácie, čo spôsobuje výrazne nižšie hodnoty priemernej presnosti AP_{50}. Príklady takýchto detekcií je možné pozorovať na obrázkoch 6.4.

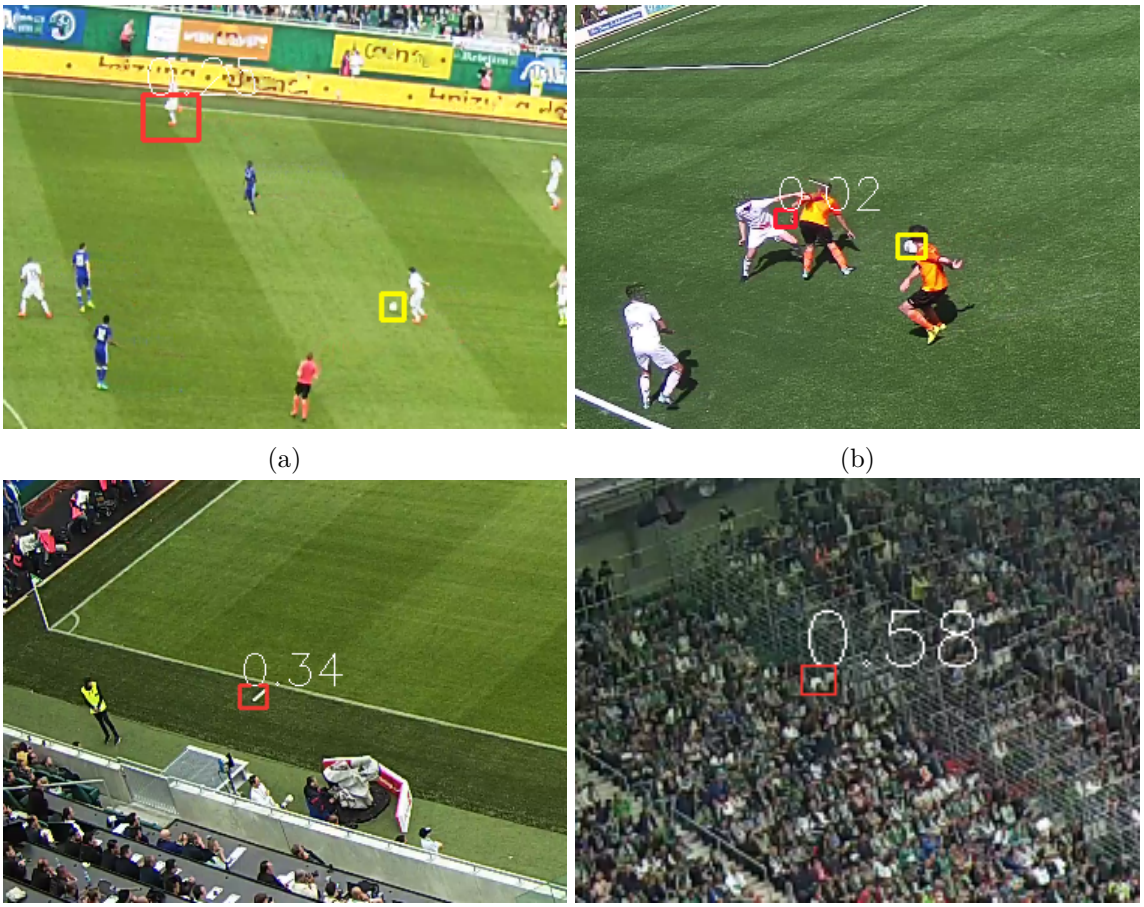


Obr. 6.4: Príklady správnych detekcií s nedokonalou lokalizáciou objektu spôsobenou typicky spracovaním snímky v menšom rozlíšení alebo menej presným detektorom.

Obr. 6.5: Príklady správnych detekcií znázornené červeným obdĺžnikom a istotou detekcie. Žltý obdĺžnik reprezentuje anotovanú polohu lopty.

Naopak prípady, kedy objekt nie je vôbec detegovaný je možné pozorovať na obrázkoch 6.6a a 6.6b. Najčastejšie sú tieto problémy spôsobené čiastočným zakrytím lopty, prípadne je lopta v obraze rozmazaná, a to buď kvôli jej rýchlemu pohybu alebo sa nachádza v okrajových častiach videí, napríklad pri rohových kopoch. Ďalším faktorom sú zhoršené svetelné podmienky, spôsobené napríklad dymom svetlíc alebo odrazom slnka. Tieto hra-

ničné prípady sa v trénovacej sade nachádzali veľmi zriedkavo, preto zlepšenie úspešnosti by mohlo byť docielené rozšírením datovej sady o podobné problematické situácie, prípadne rozšíriť použité augmentácie trénovacích dát, napríklad o umelé rozostrenie obrazu a pod.



(a) (b)

(c) (d)

Obr. 6.6: Priklady chybných detekcií objektov znázornené červeným obdĺžnikom. Žltý obdĺžnik reprezentuje anotovanú polohu lopty. Obrázky (a) a (b) ukazujú nedetegovanie lopty často spôsobené rozmazaním, prípadne zakrytom objektu. Snímky (c) a (d) naopak znázorňujú falošné detekcie vyskytujúce sa na mimo hracej plochy.

Na obrázku 6.6 sú navyše znázornené najčastejšie príklady chybnej detekcie, tzv. *False Positives*. Veľké množstvo falošných detekcií spôsobujú samotní hráči, predovšetkým v prípade nosenia bielych dresov. Ďalším často pozorovaným problémom boli rohové vlajky a reklamy okolo ihriska a tribúnach fanúšikov, prípadne malé objekty vysoko kontrastujúce s trávnikom, napríklad opadané lístie a iné. Vo väčšine prípadov, až na chybné detekcie hráčov sa však jedná o statické objekty. Po zostrojení ich trajektórie boli málo sa pohybujúce objekty penalizované nízkym ohodnotením a teda vo väčšine prípadov úspešne eliminované.

6.3.2 Vyhodnotenie sledovania lopty vo video zázname

Vyhodnotenie celého systému prebehlo na 11 testovacích video sekvenciách s vybranými detekčnými sieťami navrhnutými v podkapitole 4.2. Detekcie boli počas evaluácie prahované s hodnotou skóre 0.4. Prah minimálnej vzdialenosti pre asociovanie detekcií do trajektórií

v algoritme *SORT* bol nastavený na hodnotu 90 px. Okrem toho boli jednotlivé varianty testované pri celkovo 4 rôznych nastaveniach periódy detektoru, konkrétne 1, 2, 3 a 5, pričom perióda i znamená, že pri výpočte trajektórií bola použitá každá i-tá snímka videa.

Na jednotlivých testovacích videách boli vypočítané metriky priemerného prekrytia a úspešnosti (*Accuracy*) sledovania hracej lopty, popísané v podkapitole 6.2.1. Úspešnosť bola vypočítaná s prahom prekrytia $\tau = 0.1$. Sumarizované výsledky pre rôzne hodnoty periódy výpočtu detekcií pri spracovávaní snímok v rôznych rozlíšeniach je možné vidieť v tabuľke 6.2. Zároveň sú výsledky oboch metrík pri testovaní v troch rozlíšeniach obrazu znázornené v grafoch 6.7, 6.8 a 6.9.

Vo všeobecnosti je možné tvrdiť, že sledovanie objektov dosahovalo lepšie výsledky použitím presnejšieho detektoru. Najlepšie výsledky dosahoval systém s použitím modelu typu *LH-ResNet18*, avšak rýchlosť spracovania je v tomto prípade výrazne vyššia. Naopak najrýchlejší navrhnutý model *LH-FD-MobileNet-0.25* nedokázal úspešne detegovať väčšinu objektov, preto celková úspešnosť je výrazne nižšia v porovnaní s ostatnými detektormi.

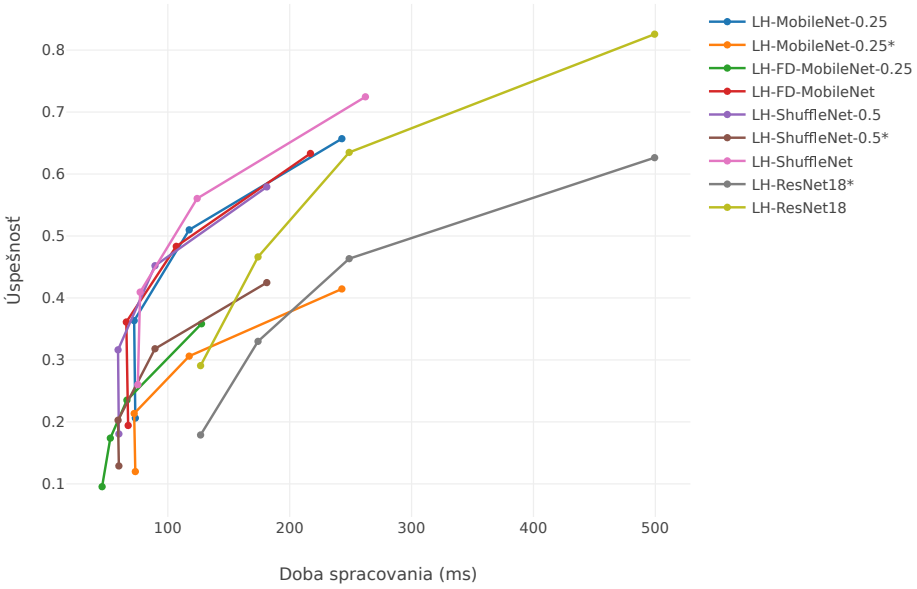
Použitie modelov natrénovaných na trénovacej sade v nižšom rozlíšení, označených symbolom '*', malo pozitívny vplyv na celkovú úspešnosť pri spracovávaní video záznamov v nižšom rozlíšení. V prípade chýbajúcich detekcií, implementovaný lineárny prediktor polohy lopty bol schopný úspešne sledovať objekt iba pomerne krátky čas. Takto jednoduchý lineárny prediktor pochopiteľne nedokáže predpovedať prudké zmeny smeru pohybu, rovnako má problém s predikciou polohy letiacej lopty, ktorá sa pohybuje nelineárne. Napriek tomu zlepšil úspešnosť systému predovšetkým v situáciách, kedy bola sledovaná lopta zakrytá relatívne krátky čas, tzn. jednotky snímok.

Znižovaním frekvencie detektoru rapídne klesá úspešnosť sledovania. S nižšou frekvenciou však rastie schopnosť systému spracovávať vstup v reálnom čase. Vo väčšine prípadov umožnilo spracovávanie každej druhej snímky takmer dvojnásobne zrýchlenie systému za cenu primeranej straty presnosti.

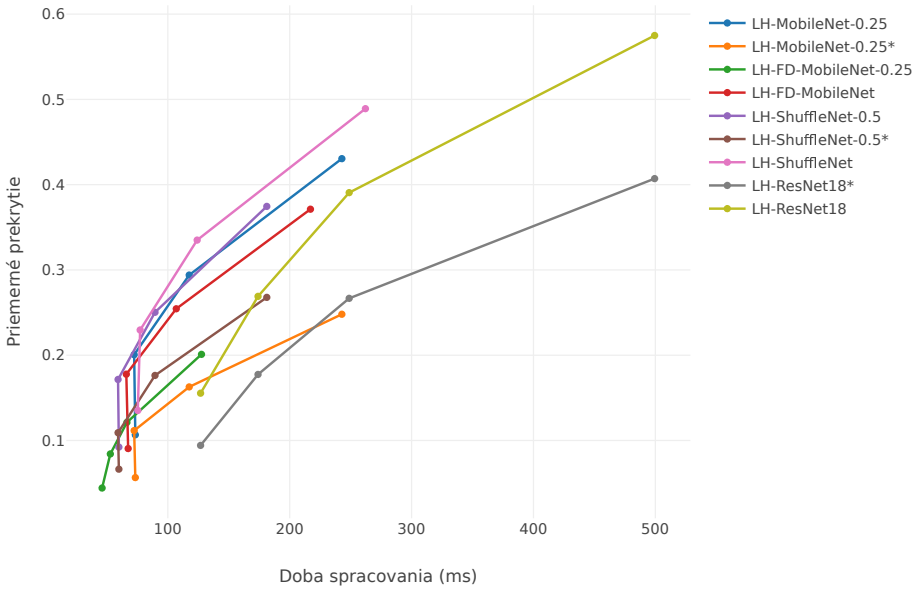
Znižovaním rozlíšenia obrazu je možné pozorovať výrazný pokles priemerného prekrytia sledovaného objektu. Na úspešnosť však tento faktor nemal až tak výrazný vplyv. Pokles priemerného prekrytia je zjavný predovšetkým pri polovičnom rozlíšení, kedy všetky detektory majú problém s presnou lokalizáciou výrazne menších objektov. Tento fakt do značnej miery ovplyvňuje návrh tzv. *anchor boxov* použitých pri dekódovaní predikcií modelov. Najmenší box má rozmery 32×32 px, čo u predikcií v polovičnom rozlíšení znamená dvojnásobnú veľkosť v pôvodom snímku.

		4000 × 3000			2560 × 1920			2048 × 1536		
Názov siete	Perióda detektoru	$\overline{\phi}$	Acc	Čas (ms)	$\overline{\phi}$	Acc	Čas (ms)	$\overline{\phi}$	Acc	Čas (ms)
LH-ResNet18	1	**57.49**	**82.57**	499.40	36.38	76.49	204.75	21.05	63.66	136.93
	2	39.06	63.49	248.78	25.59	59.97	94.43	15.66	50.94	63.79
	3	26.90	46.62	174	17.64	43.78	62.95	10.85	37.13	47.15
	5	15.55	29.07	126.8	10.71	27.94	53.9	05.68	20.44	35.66
LH-ResNet18*	1	40.70	62.63	499.40	**38.32**	77.96	204.75	**24.78**	73.48	136.93
	2	26.66	46.33	248.78	25.86	59.17	94.43	18.29	58.54	63.79
	3	17.74	32.99	174	18.34	43.83	62.95	13.66	45.93	47.15
	5	09.42	17.88	126.8	10.81	28.10	53.9	07.46	27.25	35.66
LH-MobileNet	1	–	–	–	36.82	76.32	236.30	21.62	63.71	155.14
	2	–	–	–	26.88	61.95	112.74	15.99	51.39	73.08
	3	–	–	–	17.99	44.00	71.43	10.13	34.12	45.10
	5	–	–	–	10.32	26.51	62.82	06.16	21.06	43.86
LH-MobileNet*	1	–	–	–	34.54	**79.32**	236.30	22.79	**76.88**	155.14
	2	–	–	–	26.23	66.21	112.74	17.59	63.25	73.08
	3	–	–	–	17.81	47.30	71.43	12.31	46.28	45.10
	5	–	–	–	10.30	28.84	62.82	07.17	28.33	43.86
LH-MobileNet-0.25	1	43.04	65.70	242.79	29.98	63.71	113.06	19.14	57.41	72.04
	2	29.39	50.99	117.53	21.35	50.88	52.28	13.83	44.40	37.11
	3	20.06	36.34	72.37	14.19	35.88	38.20	09.33	32.14	27.72
	5	10.66	20.61	73.33	07.56	19.81	37.77	04.82	17.43	28.95
LH-MobileNet-0.25*	1	24.80	41.45	242.79	28.70	66.55	113.06	19.86	63.88	72.04
	2	16.27	30.60	117.53	20.13	51.50	52.28	14.47	50.14	37.11
	3	11.16	21.35	72.37	14.11	38.21	38.20	10.44	37.81	27.72
	5	05.64	11.98	73.33	07.59	21.23	37.77	05.73	22.82	28.95
LH-FD-MobileNet	1	37.12	63.31	217.01	27.28	57.97	90.81	16.21	44.46	60.89
	2	25.44	48.32	106.91	18.31	43.49	43.61	11.20	34.18	30.62
	3	17.77	36.11	65.94	12.33	30.21	31.96	07.56	23.50	25.08
	5	09.05	19.42	67.44	05.97	15.84	33.35	03.67	11.98	24.12
LH-FD-MobileNet-0.25	1	20.10	35.83	127.62	14.76	34.01	63.25	09.58	27.25	43.38
	2	12.12	23.50	66.42	09.45	23.50	35.79	05.67	17.71	26.76
	3	08.42	17.37	52.81	05.65	14.82	30.31	03.36	10.84	22.89
	5	04.42	09.54	46.06	02.43	06.70	28.12	01.16	03.97	24.31
LH-ShuffleNet	1	48.91	72.45	262.12	32.97	73.19	108.36	18.64	58.31	70.78
	2	33.49	56.04	124.05	22.93	56.72	51.16	13.10	42.75	33.14
	3	22.95	40.94	77.28	15.68	40.37	33.28	08.58	29.52	27.26
	5	13.51	26.00	75.3	09.11	25.09	36.0	04.45	16.18	28.42
LH-ShuffleNet-0.5	1	37.45	57.92	181.19	27.97	59.90	88.79	17.83	52.86	58.78
	2	25.03	45.20	89.46	19.02	44.80	46.61	12.62	40.20	29.65
	3	17.15	31.62	59.16	12.44	30.66	34.14	08.92	29.35	27.95
	5	09.22	18.05	59.83	06.65	17.37	32.06	04.16	14.87	26.68
LH-ShuffleNet-0.5*	1	26.78	42.47	181.19	26.33	59.00	88.79	17.34	54.51	58.78
	2	17.62	31.80	89.46	17.41	43.21	46.61	12.41	41.62	29.65
	3	10.91	20.27	59.16	12.48	32.36	34.14	08.46	29.58	27.95
	5	06.63	12.89	59.83	06.25	16.97	32.06	04.24	15.61	26.68

Tabuľka 6.2: Sumarizácia nameraných výsledkov vyhodnotenia sledovania lopty v rôznych rozlíšeniach a periódach detektoru.

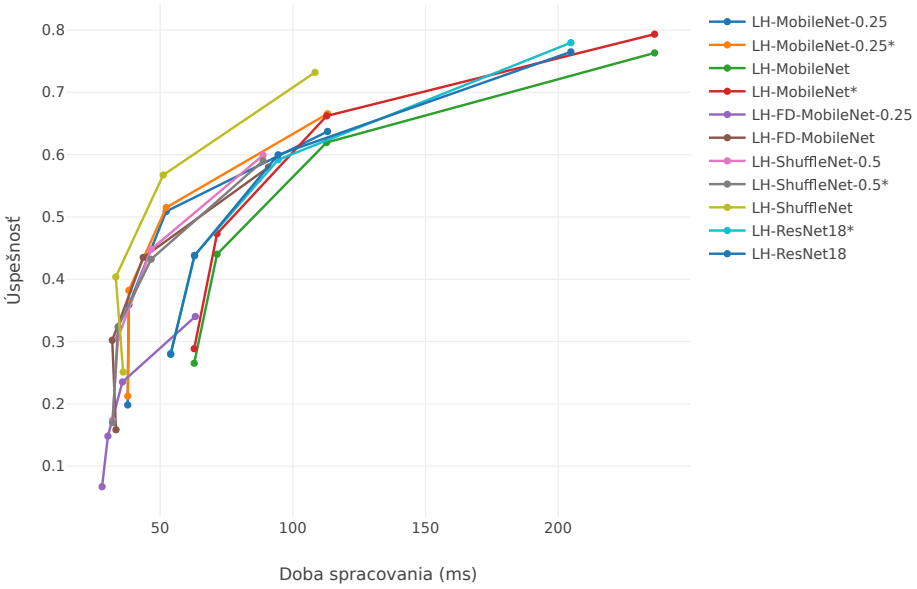


(a)

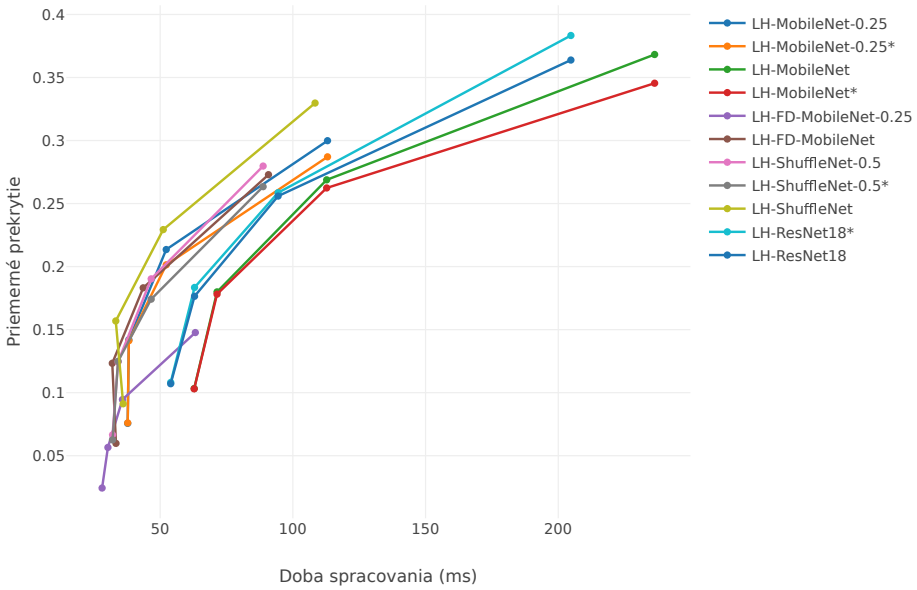


(b)

Obr. 6.7: Porovnanie (a) úspešnosti a (b) priemerného prekrytia sledovanej pohybujúcej sa lopty a rýchlosti spracovania videí v rozlíšení 4000×3000 px v závislosti od zvolenej periódy detektoru, v poradí 5, 3, 2 a 1, pričom s klesajúcou hodnotou periódy rastie úspešnosť aj doba spracovania systému.

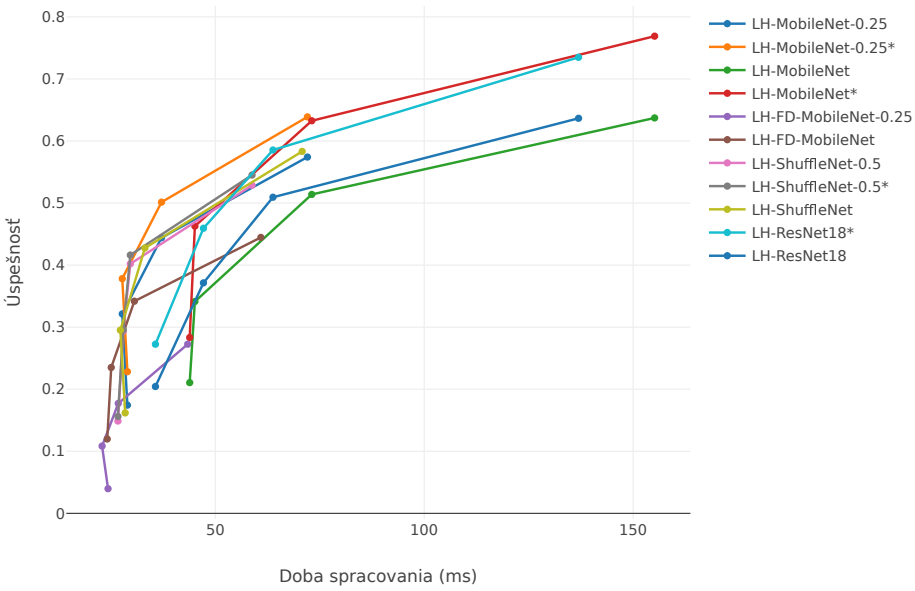


(a)

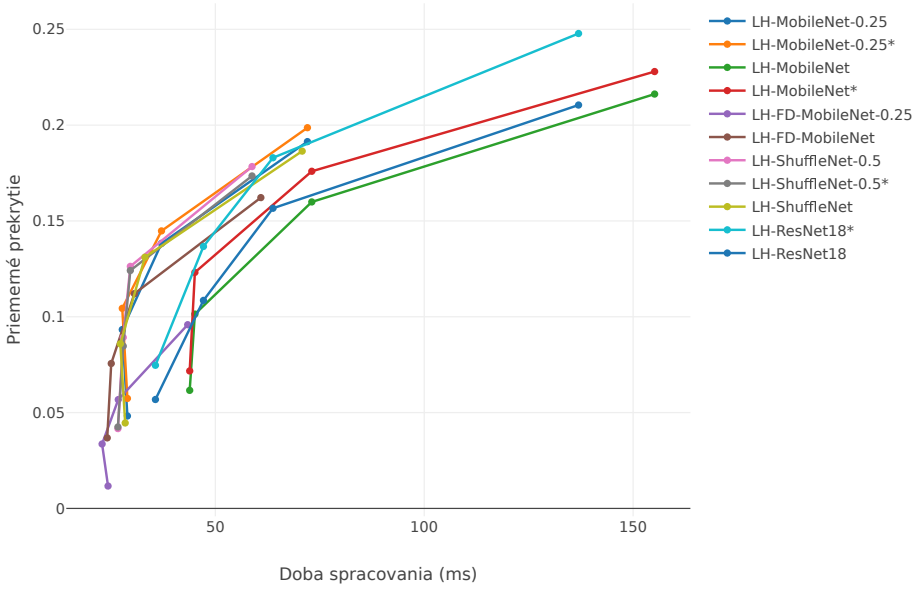


(b)

Obr. 6.8: Porovnanie (a) úspešnosti a (b) priemerného prekrytia sledovanej pohybujúcej sa lopty a rýchlosti spracovania videí v rozlíšení 2560×1920 px v závislosti od zvolenej periódy detektoru, v poradí 5, 3, 2 a 1, pričom s klesajúcou hodnotou periódy rastie úspešnosť aj doba spracovania systému.



(a)



(b)

Obr. 6.9: Porovnanie (a) úspešnosti a (b) priemerného prekrytia sledovanej pohybujúcej sa lopty a rýchlosti spracovania videí v rozlíšení 2048×1536 px v závislosti od zvolenej periódy detektoru, v poradí 5, 3, 2 a 1, pričom s klesajúcou hodnotou periódy rastie úspešnosť aj doba spracovania systému.

6.3.3 Ďalšia práca

Na základe porovnania výsledkov jednotlivých modelov a ich ekvivalentov označených symbolom '*', ktoré boli natrénované na rovnakej trénovacej sade, avšak v 2/3 rozlíšení, je možné pozorovať zlepšenie presnosti detekcie v snímkach s nižším rozlíšením. Rozšírením tzv. augmentácií dát počas trénovania o generovanie nových snímok v rôznom rozlíšení by preto mohlo mať pozitívny vplyv na schopnosti detektoru. Malé zlepšenie presností jednotlivých modelov by taktiež mohlo byť dosiahnuté úpravou samotného trénovania, a to napríklad využitím postupného znižovania miery učenia pomocou tzv. *Learning-rate scheduler*, prípadne rozšírením datovej sady o väčšie množstvo tzv. ťažkých príkladov (*hard examples*). Využitím segmentačnej masky počas trénovania pre výpočet chyby predikcie ako je to popísané v architektúre *RetinaMask* [9] by rovnako mohlo prispieť k zlepšeniu celkovej úspešnosti detekcie.

Z výsledkov je zrejmé, že postupným zmenšovaním modelov sietí dochádza k výraznému zníženiu presnosti napriek zrýchleniu spracovania vstupu. Porovnateľné možnosti zrýchlenia systému boli dosiahnuté využitím rôznej frekvencie spracovávania snímok, avšak napriek tomu, že tento prístup umožnil využitie kvalitnejších detektorov bol pozorovaný rovnaký pokles celkovej úspešnosti sledovania. Práve z tohto dôvodu bol v podkapitole 4.3.1 navrhnutý algoritmus pre sledovanie objektov vo vynechaných snímkach pomocou výrezov, ktorý by mohol zlepšiť celkovú úspešnosť systému pri zachovaní pomerne rýchleho spracovania. Je však potrebné vhodne zvoliť počet výrezov použitých pre detekciu a hodnotu frekvencie spracovávania celej snímky s ohľadom na časovú náročnosť algoritmu.

Ďalšie možnosti rozšírenia práce ponúka použitie sledovacích algoritmov založených na neurónových sieťach. Medzi ich predstaviteľov patrí GOTURN [15], ktorý je založený na architektúre *AlexNet* [16] usporiadanej v štruktúre tzv. siamských sietí. Tento model bol experimentálne natrénovaný a následne bolo uskutočnených niekoľko experimentov. Aj keď použitie tohto systému umožnilo výrazne zrýchliť samotné spracovanie, systém mal problémy so sledovaním rýchlo sa pohybujúcej lopty, prípadne lopty, ktorá je niekoľko snímok v zákryte. Tieto problémy by mohlo ísť minimalizovať dodatočným trénovaním a vyladením parametrov samotného algoritmu. V kombinácií so sledovacím algoritmom implementovaným v tejto práci je predpoklad, že vďaka rýchlosti modelu *GOTURN* by mohlo byť dosiahnuté výrazne rýchlejšie spracovanie videa za cenu menšej straty presnosti v porovnaní s implementovanými variantmi.

Kapitola 7

Záver

Cieľom tejto práce bolo navrhnúť systém pre sledovanie lopty v zázname futbalového stretnutia pomocou hlbokých konvolučných sietí v spolupráci s firmou CamVision s.r.o.. Boli preskúmané a vyhodnotené rôzne prístupy vizuálneho sledovania pohybujúcich sa objektov vo video zázname so zameraním na efektívnu detekciu predovšetkým malých, rýchlo sa pohybujúcich objektov vo vysokom rozlíšení.

Bol navrhnutý a implementovaný systém založený na princípe sledovania objektov na základe detekcií, ktorý dokáže využívať celkovo štyri architektúry detekčných sietí. Implementované konvolučné siete pre detekciu objektov futbalovej lopty sú založené na architektúre typu *RetinaNet*, v ktorej bola modifikovaná jej základná sieť určená pre extrakciu príznakov a zároveň boli zjednodušené klasifikačné a regresné podsiete. V základných sieťach boli pre získanie aktivačných máp využité efektívnejšie modely *ResNet18*, *MobileNetV2*, *ShuffleNetV2* a ich varianty, vďaka čomu bolo možné dosiahnuť výrazne zrýchlenie pri spracovávaní snímok vo vysokom rozlíšení. Natrénované modely boli vyhodnotené na anotovaných video záznamoch, samostatne v troch rôznych rozlíšeniach, hlavne pre určenie vplyvu veľkosti obrazu na rýchlosť a presnosť detekcie lopty. Okrem toho boli natrénované ekvivalentné modely sietí na datovej sade v nižšom rozlíšení pre zlepšenie výsledkov detekcie objektov vo videách v menšom rozlíšení.

Sledovací algoritmus *SORT* bol upravený na základe analýzy poskytnutých dát, pričom navrhnutý systém bol vyhodnotený s rôznym nastavením parametrov a v kombinácií so všetkými natrénovanými modelmi sietí. Bolo overené, že úspešnosť sledovania do veľkej miery závisí na kvalite získaných detekcií objektov, pričom najlepšiu úspešnosť sledovania 82.57 % dosiahol systém využitím modelu *LH-ResNet18* avšak pri rýchlosti spracovania 499.4 ms. Kvôli zvýšenej časovej náročnosti spracovania využitím týchto kvalitnejších a aj zložitejších modelov, boli otestované možnosti zrýchlenia jednotlivých variantov použitím snímok v menšom rozlíšení, ako aj dopad zvyšovania periódy získavania detekcií na celkovú úspešnosť sledovania. Navrhnutá detekčná sieť *LH-ShuffleNet* dokázala dosiahnuť úspešnosť 73.19 % pri rýchlosti spracovania 108.36 ms a model *LH-MobileNet-0.25* dokonca úspešnosť 63.88 pri rýchlosti 72.04 ms.

Pre ďalšie zlepšenie celkovej úspešnosti boli navrhnuté viaceré možnosti, zamerané jednak na skvalitnenie detekcií implementovaných modelov sietí ale aj na rozšírenie existujúceho sledovacieho algoritmu napríklad o tzv. sledovanie objektov vo výrezoch obrazu pri použití vysokej periódy detektoru, alebo využitia sledovacích algoritmov založených na neurónových sieťach.

Literatúra

[1] Bernardin, K.; Elbs, A.; Stiefelhagen, R.: *Multiple object tracking performance metrics and evaluation in a smart Room environment. Proceedings of IEEE International Workshop on Visual Surveillance*, 01 2006.

[2] Bewley, A.; Ge, Z.; Ott, L.; aj.: *Simple online and realtime tracking. 2016 IEEE International Conference on Image Processing (ICIP)*, 2016: s. 3464–3468.

[3] Chollet, F.: *Xception: Deep Learning with Depthwise Separable Convolutions. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017: s. 1800–1807.

[4] Dai, J.; Li, Y.; He, K.; aj.: *R-FCN: Object Detection via Region-based Fully Convolutional Networks. CoRR*, ročník abs/1605.06409, 2016, 1605.06409.
URL http://arxiv.org/abs/1605.06409

[5] Dai, J.; Qi, H.; Xiong, Y.; aj.: *Deformable Convolutional Networks. arXiv preprint arXiv:1703.06211*, 2017.

[6] Everingham, M.; Van Gool, L.; Williams, C. K. I.; aj.: *The Pascal Visual Object Classes (VOC) Challenge. International Journal of Computer Vision*, Jun 2010, ISSN 1573-1405.

[7] Everingham, M.; Winn, J.: *The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Development Kit.* 2012.
URL http://host.robots.ox.ac.uk/pascal/VOC/voc2012/devkit_doc.pdf

[8] Fu, C.-Y.; Liu, W.; Ranga, A.; aj.: *DSSD : Deconvolutional Single Shot Detector.* 01 2017.

[9] Fu, C.-Y.; Shvets, M.; Berg, A. C.: *RetinaMask: Learning to predict masks improves state-of-the-art single-shot detection for free. CoRR*, ročník abs/1901.03353, 2019.

[10] Girshick, R.: *Fast R-CNN. International Conference on Computer Vision (ICCV)*, 2015.

[11] Girshick, R.; Donahue, J.; Darrell, T.; aj.: *Rich feature hierarchies for accurate object detection and semantic segmentation. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 11 2013.

[12] Haykin, S.: *Neural Networks and Learning Machines: Third Edition.* Pearson, 2009, ISBN 978-0-13-147139-9.

[13] He, K.; Gkioxari, G.; Dollár, P.; aj.: *Mask R-CNN*. Oct 2017: s. 2980–2988, ISSN 2380-7504.

[14] He, K.; Zhang, X.; Ren, S.; aj.: *Deep Residual Learning for Image Recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016: s. 770–778.

[15] Held, D.; Thrun, S.; Savarese, S.: *Learning to Track at 100 FPS with Deep Regression Networks. LNCS*, ročník 9905, 10 2016: s. 749–765.

[16] Krizhevsky, A.; Sutskever, I.; E. Hinton, G.: *ImageNet Classification with Deep Convolutional Neural Networks. Neural Information Processing Systems*, ročník 25, 01 2012.

[17] Lawrence Zitnick, C.; Dollar, P.: *Edge boxes: Locating object proposals from edges.* ročník 8693, 09 2014.

[18] Li, Z.; Peng, C.; Yu, G.; aj.: *Light-Head R-CNN: In Defense of Two-Stage Object Detector*. 11 2017.

[19] Lin, T.-Y.; Dollár, P.; Girshick, R.; aj.: *Feature Pyramid Networks for Object Detection. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017: s. 936–944.

[20] Lin, T.-Y.; Goyal, P.; Girshick, R. B.; aj.: *Focal Loss for Dense Object Detection. 2017 IEEE International Conference on Computer Vision (ICCV)*, 2017: s. 2999–3007.

[21] Liu, W.; Anguelov, D.; Erhan, D.; aj.: *SSD: Single Shot MultiBox Detector. CoRR*, ročník abs/1512.02325, 2015, 1512.02325.

[22] Ma, N.; Zhang, X.; Zheng, H.-T.; aj.: *ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design*. September 2018.

[23] Qin, Z.; Zhang, Z.; Chen, X.; aj.: *Fd-Mobilenet: Improved Mobilenet with a Fast Downsampling Strategy. 2018 25th IEEE International Conference on Image Processing (ICIP)*, 2018: s. 1363–1367.

[24] Redmon, J.; Divvala, S.; Girshick, R.; aj.: *You Only Look Once: Unified, Real-Time Object Detection*. 06 2016: s. 779–788.

[25] Redmon, J.; Farhadi, A.: *YOLO9000: Better, Faster, Stronger*. 12 2016.

[26] Redmon, J.; Farhadi, A.: YOLOv3: An Incremental Improvement. *arXiv*, 2018.

[27] Ren, S.; He, K.; Girshick, R.; aj.: *Faster R-CNN: Towards real-time object detection with region proposal networks. IEEE Transactions on Pattern Analysis and Machine Intelligence*, ročník 39, č. 6, June 2017: s. 1137–1149, ISSN 0162-8828.

[28] Sandler, M. B.; Howard, A. G.; Zhu, M.; aj.: *MobileNetV2: Inverted Residuals and Linear Bottlenecks. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018: s. 4510–4520.

[29] Simonyan, K.; Zisserman, A.: *Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv 1409.1556*, 09 2014.

[30] Uijlings, J. R. R.; van de Sande, K. E. A.; Gevers, T.; aj.: *Selective search for object recognition. International Journal of Computer Vision*, ročník 104, č. 2, 2013: s. 154–171.

[31] Zhang, X.; Zhou, X.; Lin, M.; aj.: *ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices*. 07 2017.

[32] Zhao, Z.; Zheng, P.; Xu, S.; aj.: *Object Detection with Deep Learning: A Review. CoRR*, 2018, 1807.05511.

Príloha A

Obsah priloženého DVD

Priložené DVD obsahuje nasledujúce položky:

- Digitálna verzia tejto práce
- Zdrojové súbory s implementáciou navrhnutého systému
- Súbor `README.md` opisujúci kroky potrebné pre spustenie systému
- Natrénované modely neurónových sietí
- Ukážku datovej sady a testovacích videí
- Video ukážky demonštrujúce dosiahnuté výsledky.