VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

DETEKCE A ROZPOZNÁNÍ CHODCŮ VE VÍCEKAMEROVÉM SYSTÉMU

PEDESTRIAN DETECTION AND RECOGNITION IN A MULTI-CAMERA SYSTEM

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **FILIP MACÁK**
AUTHOR

VEDOUCÍ PRÁCE **Ing. TOMÁŠ GOLDMANN,**
SUPERVISOR

BRNO 2021

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav inteligentních systémů (UITS) Akademický rok 2020/2021

Zadání bakalářské práce

Student: **Macák Filip**
Program: Informační technologie
Název: **Detekce a rozpoznání chodců ve vícekamerovém systému**
Pedestrian Detection and Recognition in a Multi-Camera System
Kategorie: Umělá inteligence

Zadání:

1. Seznamte se s problematikou detekce chodců ve videozáznamu. Uveďte alespoň 5 příkladů komerčních a nekomerčních řešení pro detekci chodců a sledování jejich pohybu.
2. Sumarizujte informace o algoritmech pro detekci chodců a extrakci charakteristických příznaků ze snímků postav.
3. Navrhněte řešení, které umožní rozpoznat jednotlivé chodce napříč záznamy z kamerového systému. Předpokládejte, že všechny osoby jsou při vstupu nasnímány jednou z kamer a že všechny záznamy jsou pořízeny simultánně.
4. Navržené řešení implementujte v programovacím jazyce Python a vytvořte k němu uživatelské rozhraní.
5. Proveďte experimenty zaměřené na vyhodnocení spolehlivosti řešení.

Literatura:

- RAYCHAUDHURI, Amlan, et al. A Novel Approach for Human Silhouette Extraction from Video Data. In: Advanced Computing and Systems for Security. Springer, Singapore, 2017. p. 125-135.
- CHEN, Muchun, et al. Real-Time Multiple Pedestrians Tracking in Multi-camera System. In: International Conference on Multimedia Modeling. Springer, Cham, 2020. p. 468-479.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 a 2

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Goldmann Tomáš, Ing.**
Vedoucí ústavu: Hanáček Petr, doc. Dr. Ing.
Datum zadání: 1. listopadu 2020
Datum odevzdání: 12. května 2021
Datum schválení: 11. listopadu 2020

Abstrakt

Hlavným cieľom tejto bakalárskej práce je vytvoriť aplikáciu na detekciu a rozpoznanie osôb zachytených na záberoch v multi-kamerovom systéme. Výstupom aplikácie je video, na ktorom sú detekované osoby vyznačené a každej osobe je priradené identifikačné číslo, pomocou ktorého je možno ju rozoznať naprieč zadanými záznamami. Pri riešení práce bolo preskúmaných viacero riešení problematiky detekcie a rozpoznávania založených na neurónových sietiach a text práce slúži ako prehľad týchto problematík. Aplikácia je postavená na knižniciach PyTorch a Torchreid. Na detekciu je využitý detektor so sieťou Faster-RCNN a rozpoznávanie je založené na sieti OSNet. Súčasťou aplikácie je aj jednoduché užívateľské rozhranie pre uľahčenie práce s aplikáciou. Aplikácia slúži ako ukážka state-of-the-art riešení deteckie a rozpoznávania osôb.

Abstract

The main purpose of this bachelor's thesis is to create an application for person detection and recognition from scenes captured in a multi-camera system. The output of the application is a video on which the detected persons are highlighted and each person is assigned an identification number through which it can be recognized across the input scenes. Several solutions to the problem of person detection and recognition were examined and the text of this work serves as an overview of these problems. The application is built on PyTorch and Torchreid libraries. A detector with a Faster-RCNN network is used for detection and recognition is based on the OSNet network. The application also includes a simple user interface to facilitate work with the application. The application serves as a demonstration of the state-of-the-art for person detection and recognition.

Kľúčové slová

detekcia osôb, rozpoznávanie osôb, kamerový systém, umelá inteligencia, konvolučné neurónové siete, Python, PyTorch, torchreid, Faster-RCNN, OSNet

Keywords

person detection, person recognition, camera system, artificial intelligence, convolutional neural networks, Python, PyTorch, torchreid, Faster-RCNN, OSNet

Citácia

MACÁK, Filip. *Detekce a rozpoznání chodců ve vícekamerovém systému.* Brno, 2021. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Tomáš Goldmann,

Detekce a rozpoznání chodců ve vícekamerovém systému

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením Ing. Tomáša Goldmanna. Uviedol som všetky literárne pramene, publikácie a ďalšie zdroje, z ktorých som čerpal.

.....
Filip Macák
18. mája 2021

Poďakovanie

Týmto by som chcel poďakovať vedúcemu mojej práce pánovi Ing. Tomášovi Goldmannovi za cenné rady a odbornú pomoc, ktorú mi poskytol pri riešení tejto práce. Taktiež chcem poďakovať mojím rodičom za podporu môjho štúdia a kamarátom Radovi, Mišovi a Poizzimu za to, že sú tu pre mňa, keď ich potrebujem. V poslednom rade chcem poďakovať Eliške, Toasterovi, Vyzimu a Oloykovi za to, že mi pomohli vytvoriť videá, na ktorých bola aplikácia testovaná.

Obsah

Kapitola 1

Úvod

Detekcia chodcov je dôležitou témou v oblasti počítačového videnia, a preto aj dodnes zostava ako aktívna oblasť výskumu. Počítačové videnie je formou umelej inteligencie, kde počítač analyzuje vizuálne dáta. Dokáže na ich základe získať informácie o situácii a prostredí alebo spraviť nejaké rozhodnutie. Počítačové videnie ako také prináša už v dnešnej dobe veľa užitočných uplatnení ako napríklad: autonómne vozidla, rozpoznávanie tvári, schopnosť čítania textu z obrazu alebo sledovanie výkonu športovcov.

Samotná detekcia chodcov je uplatňovaná v širokom spektre problémov. Existuje mnoho prístupov k riešeniu detekcie chodcov. Keďže sa jedna o zložitú problematiku, v reálnom svete nie je jasný ideálny prístup k riešeniu, a teda záleží, na čo sa bude dané riešenie využívať. Riešenia musia balansovať medzi rýchlosťou detekcie a jej presnosťou. Ak má byť riešenie využité na real-time detekciu, musí byť schopné detegovať osoby veľmi rýchlo a to môže znamenať zvýšenú pravdepodobnosť nezachytenia nejakej osoby (*false negative*) alebo zachytenia osoby, ktorá sa v obraze nevyskytuje (*false positive*).

Dôležitou súčasťou detekcie chodcov (ale aj detekcie objektov všeobecne) je strojové učenie a neurónové siete. Hlavnými dôvodmi, prečo je strojové učenie tak dôležité, sú veľká rýchlosť detekcie a schopnosť prispôsobiť sa rôznym podmienkam. Detekcia osôb je veľmi náročným problémom. Bežnými problémami pre detekciu a rozpoznávanie osôb bývajú: rôzne druhy oblečenia a vzhľadu osôb, detekcia osôb, ktoré sú ďaleko od kamery, prostredie, ktoré zakrýva časť osoby, navzájom sa prekrývajúce osoby, zlé svetelné podmienky a pohyb kamery.

Kapitola 2 sa zameriava na predstavenie problematiky detekcie chodcov, predstavuje neurónové siete so zameraním na konvolučné neurónové siete, ktoré sú pre modernú detekciu chodcov kľúčové.

Kapitola 3 opisuje *state-of-the-art* algoritmy a prístupy k riešeniu detekcie chodcov. Rozoberá jednotlivé vybrané algoritmy a slúži ako sumár pre výber správnych prístupov k návrhu riešenia tohto problému v praktickej časti tejto práce a taktiež uvádza zopár súčasných prípadov využitia detekcie osôb a sledovania ich pohybu.

Kapitola 4 spracováva otázky riešenia detekcie osôb v multi-kamerových systémoch, pričom dôležitou časťou tejto problematiky je re-identifikácia osôb naprieč kamerami. Poznatky z tejto kapitoly slúžia na výber správneho prístupu k problematike, podľa toho pre aký typ multi-kamerového systému je riešenie navrhované.

Kapitola 5 obsahuje návrh výslednej aplikácie a jej implementačné detaily. Na vytvorenie návrhu a implementácie boli využité poznatky z predchádzajúcich kapitol.

Kapitola 6 uvádza niektoré z predvedených experimentov, ktoré slúžia na overenie spoľahlivosti aplikácie. Kapitola taktiež uvádza aj problémy aplikácie.

Kapitola 2

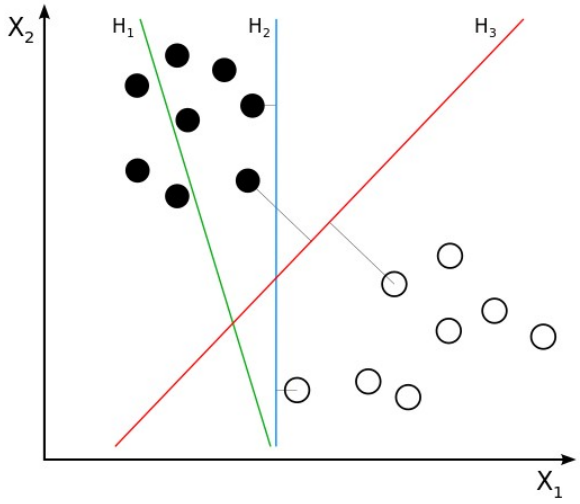
Strojové učenie a neuronové siete

V dnešnej dobe sa vo všeobecnej detekcii objektov, a teda aj v probléme detekcie chodcov, stali dominantnou silou konvolučné neurónové siete označované CNN z anglického názvu *Convolutional Neural Networks*. Cieľom tejto prace bude využiť tieto nove metódy, avšak je vhodne spomenúť aj staršie spôsoby riešenia tejto problematiky. Pred nástupom týchto neurónových sieti bolo bežné, že algoritmy tento problém riešili štýlom posúvajúceho sa okna, ktoré predstavovalo nejakú podčasť celého obrazu a v tomto okne sa hľadali vybrané príznaky objektu, ktorý sa snažíme detegovať. Takéto algoritmy boli inšpirované algoritmom Viola Jones [31]. Veľa takýchto starších algoritmov sa práve zameriavalo skôr na extrakciu príznakov a daný extrahovaný príznakový vektor bol ďalej poslaný binárnemu klasifikátoru, ktorý rozhodol, či sa v danej časti obrázka hľadaný objekt nachádza alebo nie. V niektorých prípadoch sa riešenia snažili pridať extra príznaky ako napríklad pohyb pre detekciu vo videozázname [19].

Jeden z významných posunov v obore detekcie ľudí bolo využitie HOG (*Histogram of oriented gradients*) príznakového deskriptoru, ktoré navrhli v ich práci Dalal a Triggs [8]. Základnou myšlienkou za HOG deskriptorom je, že vzhľad a tvar lokálneho objektu v obrázku môže byt opísaný rozložením svetelných gradientov alebo smerov hrán. Svetelný gradient je vektor vyjadrujúci zmenu svetelnej intenzity alebo farby pixelu. V prvom vydaní ich prace sa zameriavali iba na detekciu ľudí v statických obrázkoch, avšak časom rozšírili ich testovanie aj na videozáznamy. Pri vyhľadávaní objektu touto metódou je zaber rozdelený do niekoľko menších častí. Pre každú časť je vytvorený príznakový deskriptor HOG, ktorý slúži ako vstup pre *Support vector machine* (SVM) klasifikátor. HOG deskriptory môžu byt využité aj s inými algoritmami strojového učenia.

Support vector machine (SVM) je algoritmus strojového učenia s učiteľom. Je najčastejšie využívaný na klasifikáciu prvkov. SVM boli v minulosti veľmi využívané pre detekciu chodcov, a preto je podľa mňa dôležite ich spomenúť. Hlavným cieľom SVM je nájsť v obecnom n-dimenzionálnom priestore takú nadrovinu, ktorá rozdelí objekty trénovacej množiny na 2 pol priestory tak, že všetky objekty rovnakej triedy sa nachádzajú v tom istom pol priestore.

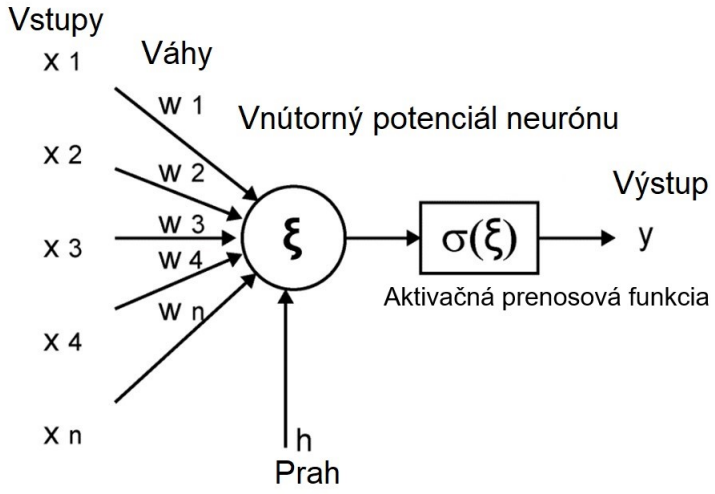
Pokiaľ sa takáto nadrovina nedá vytvoriť, jedná sa o nelineárne oddeliteľné dáta. Pokiaľ sa nám však takúto nadrovinu nájsť podarí, znamená to, že dáta sú lineárne oddeliteľné, tak hľadanie nekončí a algoritmus sa snaží nájsť nadrovinu, ktorej vzdialenosť od najbližších objektov oboch tried je čo najväčšia, takzvaný *maximum-margin hyperplane* (na obrázkú 2.1 ju predstavuje nadrovina H3)



Obr. 2.1: Na obrázku sú znázornené 3 nadroviny, ktoré rozdeľujú priestor. Nadrovina H_1 rozdeľuje priestor tak, že nie všetky prvky jednej množiny sú v jednej polrovine. Nadrovina H_2 rozdeľuje priestor na 2 polroviny, kde každá má všetky prvky jednej množiny, avšak nejedná sa o ideálne riešenie. Nadrovina H_3 je ideálnym riešením pre tento priestor [34].

2.1 Neurónové siete

Neurónová sieť [4] je matematický model inšpirovaný biologickými nervovými systémami. Neurónové siete nachádzajú uplatnenie vo veľa odvetviach a v počítačovom videní sa používajú na detekciu a rozpoznávanie objektov. Základnou časťou neurónovej siete je model neurónu, ktorý ma N vstupov a M výstupov. Popis modelu umelého neurónu je na obrázku 2.2. Neuróny sú spojené takzvanými hranami. Neuróny aj hrany typicky majú nejakú hodnotu „váhy“, ktorá sa upravuje s postupným učením siete. Neuróny v sieti sú združované do vrstiev. Každá vrstva môže nad jej vstupmi previesť rôzne transformácie. Klasická neurónová sieť sa skladá z 3 typov vrstiev: vstupná vrstva, skryté vrstvy, výstupná vrstva (obr. 2.4). Skrytých vrstiev môže byt niekoľko, ale nemusí sa v sieti nachádzať ani jedna.



Obr. 2.2: Popísaný model umelého neurónu. Znázornenie závislostí jednotlivých zložiek pre výpočet hodnoty neurónu.

Vo výpočte výstupu neurónu postupujeme nasledovne: súčet vstupných hodnôt vynásobíme ich príslušnými váhami, k tomuto pripočítame hodnotu prahu (angl. *bias*), a tento medzi výsledok je poslaný ako argument aktivačnej funkcii, ktorá vracia výstup neurónu. Matematicky je to možné vyjadriť ako [14]:

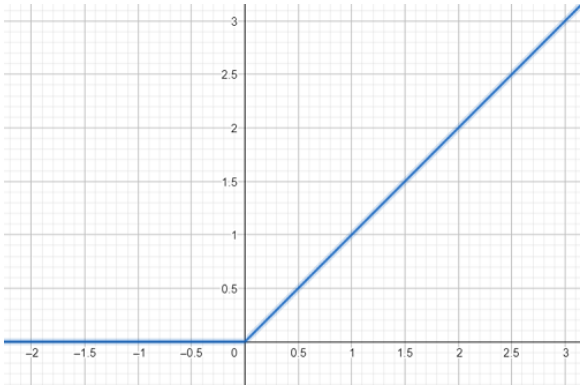
$$y = \sigma(\sum_{i=0}^{n} w_i x_i - h)$$

kde y značí výstupnú hodnotu neurónu, σ aktivačnú funkciu, x vstupy neurónu, w váhy jednotlivých vstupov a h prah neurónu.

Z praktických dôvodov sa niekedy prah neurónu nahradí nultým vstupom a nultou váhou. Kde nultý vstup má vždy hodnotu rovnú jedna a nultá váha je nastavená na hodnotu -h. V tomto prípade sa prah stane jednou z váh a taktiež v priebehu učenia podlieha prispôsobeniu hodnoty.

Aktivačná funkcia (σ) slúži na zavedenie nelinearity do výpočtu siete. Existuje viacero rôznych aktivačných funkcii. Najjednoduchší typ aktivačnej funkcie je skoková funkcia, ktorá je rovná 1 pre hodnoty väčšie alebo rovné 0, a rovná 0 v ostatných prípadoch, a predstavuje ostrú nelinearitu. Medzi ďalšie známe aktivačné funkcie patria *ReLU*, *sigmoid* a *softmax*.

ReLU (*Rectified Linear Units*) [1] je typ aktivačných funkcií, ktoré sú lineárne na pozitívnej osi a rovné nule pre negatívnu os. Je daná vzorcom $f(x) = max(0, x)$ (obr. 2.3). Jedna z výhod ReLU funkcie je jej jednoduchosť, s čím súvisí nízka záťaž na jej výpočet. Ďalšou výhodou je, že v porovnaní s funkciami ako sigmoid rieši problém strácajúceho sa gradientu, ktorý nastáva, keď je gradient na toľko malý, že efektívne zabraňuje úpravu váh. ReLU funkcia funguje skvele pre veľkú časť aplikácií, a preto je veľmi rozšírená.



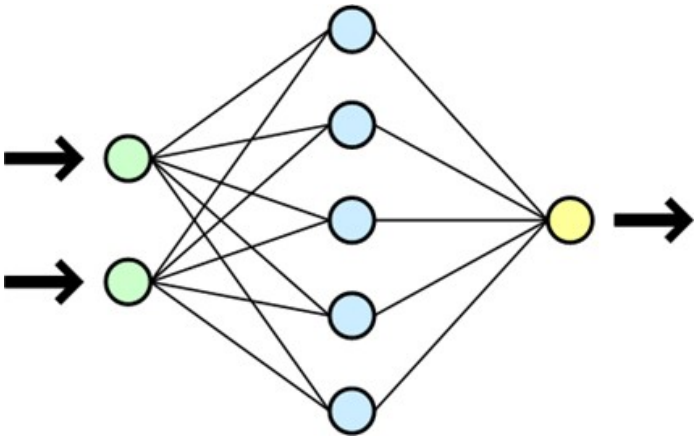
Obr. 2.3: Znázornenie ReLU aktivačnej funkcie.

Sigmoid aktivačná funkcia je daná vzorcom $f(x) = \frac{1}{1+e^{-x}}$. Táto funkcia je využívaná najmä pre modely, kde musíme predpovedať pravdepodobnosť ako výstup. Obor hodnôt funkcie je $(0, 1)$, čo odpovedá hodnotám pravdepodobnosti. Funkcia sigmoid je špeciálnym prípadom funkcie softmax pre klasifikátory s dvoma triedami.

Softmax je funkcia, ktorá prevádza vektor K reálnych hodnôt na vektor K hodnôt, ktorých súčet je jedna. Vstupné hodnoty môžu byť kladné, záporné, nulové a aj väčšie ako jedna, ale softmax ich pretransformuje na hodnoty od nula po jedna. tak aby mohli byť interpretované ako pravdepodobnosti. Veľa viac vrstvových neurónových sietí končia s výstupnými skóre, ktoré nie sú dobre naškálované, a preto môže byť práca s nimi zložitejšia. S týmto si vie funkcia softmax poradiť, a z tohto dôvodu sa táto funkcia zväčša využíva na konci siete. Softmax je daná vzorcom:

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

kde $\vec{z}$ je vstupný vektor, z_i sú jednotlivé hodnoty vstupného vektoru, K je počet tried v klasifikátore a suma v menovateli zaručuje normalizáciu hodnôt, čo znamená, že súčet hodnôt je jedna a všetky hodnoty budú na intervale $(0, 1)$. Využíva sa v klasifikátoroch s viacerými triedami.



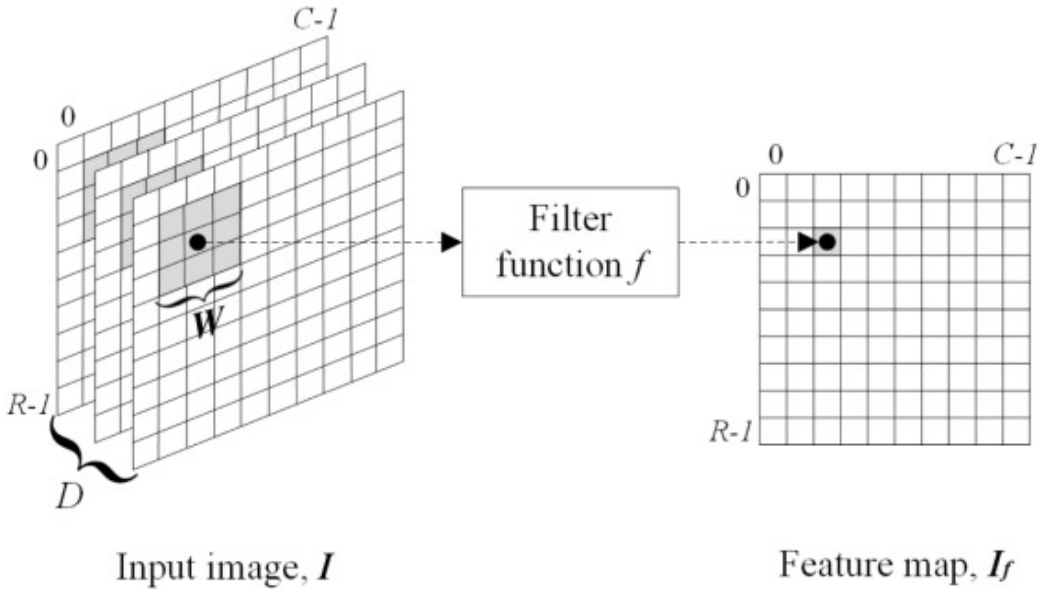
Obr. 2.4: Jednoduchý model neurónovej siete. Zelené uzly predstavujú vstupnú vrstvu, modré uzly predstavujú skrytú vrstvu a žltý uzol predstavuje vrstvu výstupnú. Jednotlivé uzly sú poprepájané hranami.

2.2 Konvolučné neurónové siete

Convolutional neural network (CNN) [30] je typ neurónovej siete, ktorý sa v dnešnej dobe veľmi často využíva na spracovanie obrazu či videa. Tak isto ako všeobecné neurónové siete aj tie konvolučné sa skladajú zo vstupnej vrstvy, skrytých vrstiev a vrstvy výstupnej, avšak skrytá časť CNN sa skladá z jednej alebo viacerých konvolučných vrstiev, medzi ktorými sa nachádzajú združovacie vrstvy, a ako posledné nasleduje jedna alebo viac plne prepojených vrstiev, ktoré spracovávajú príznaky získané z predchádzajúcich vrstiev.

Majme obrázok I skladajúci sa z R riadkov, C stĺpcov a D farebných zložiek, ktorý predstavuje vstup pre konvolučnú neurónovú sieť. Ak sa jedná o klasický RGB formát obrázku, tak bude platiť $D = 3$, kde červená, zelená a modrá sú farebné zložky. Vstup pre CNN môže byť popísaný ako 3-rozmerná funkcia $I(x, y, z)$, kde $0 \leq x < R$, $0 \leq y < C$ a $0 \leq z < D$ sú priestorové koordináty, a amplitúda I v každom bode (x, y, z) predstavuje intenzitu pixelu v tom danom bode. [30]

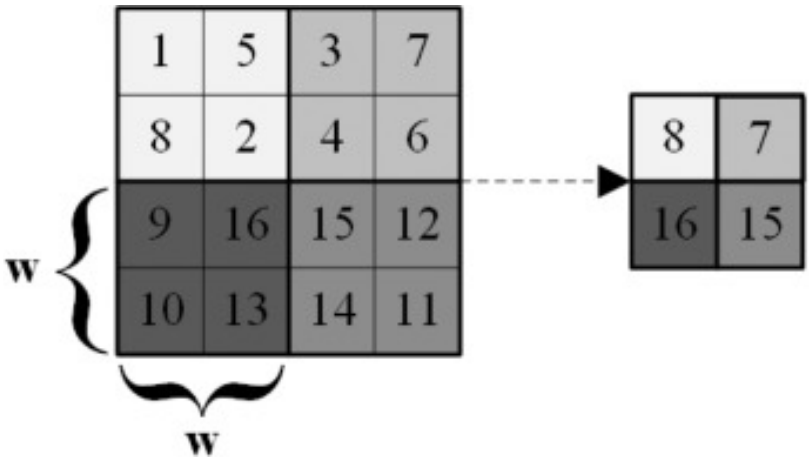
Úlohou konvolučných vrstiev je extrahovať príznaky obrazu. Konvolučné vrstvy sa skladajú z konvolučných filtrov. Filter je zvyčajne reprezentovaný ako matica, najčastejšie 3×3 alebo 5×5, ktorá prechádza všetkými kanálmi obrazu, napríklad pre obrázok, ktorý je definovaný v predchádzajúcom odstavci, by mal filter hĺbku 3. Tieto filtre sú posúvané po matici vstupu. Miesto vstupu nad ktorou sa matica filtra nachádza, sa vynásobí s filtrom a pre dané miesto sa vyprodukuje jedna hodnota. Toto je znázornené na obrázku 2.5. Po prejdení celého vstupu maticou filtra sa vyprodukuje takzvaná mapa príznakov, a keďže vrstva obsahuje viac konvolučných filtrov, tak jej výstupom je sada takýchto príznakových máp.



Obr. 2.5: Znázornenie fungovania konvolučných filtrov nad vstupným obrázkom. Filter na obrázku pracuje nad maticou $w \times w$ a zo vstupného obrázka I, ktorý má hĺbku D, vytvára príznakovú mapu I_f [30].

CNNs [30] môžu využívať veľké množstvo konvolučných filtrov. Toto vedie k zvýšeniu presnosti výsledku, avšak aj k prudkému zvýšeniu množstva dát, ktoré musí sieť spracovávať. Preto sa zaviedli združovacie vrstvy, ktorých úlohou je znížiť veľkosť príznakových máp s

tým, že sa ponechá čo najviac dôležitých informácii. Na združovanie sa môže využiť viac metód, najčastejšie sa však využíva metóda *max pooling* (príklad na obr. 2.6). Pri max poolingu sa z okna o veľkosti $w \times w$ vyberie najvyššia hodnota. Výstup týchto vrstiev je poslaný na vstup klasifikátoru.



Obr. 2.6: Príklad združovania metódou *max pooling*. Z každého regiónu vstupu o veľkosti $w \times w$ je vybraná najvyššia hodnota [30].

Plne prepojené vrstvy sa nachádzajú na konci CNN a sú zodpovedné za klasifikáciu objektov v obraze. Neuróny sú v tejto vrstve prepojené so všetkými neurónmi z vrstvy predchádzajúcej. Počet výstupov tejto vrstvy je rovný počtu tried objektov, ktoré má CNN klasifikovať.

Kapitola 3

Detekcia chodcov

Táto kapitola sa zameriava na predstavenie *state-of-the-art* detekčných algoritmov, predstavenie algoritmov na extrakciu charakteristických príznakov a sumarizovanie týchto algoritmov a datasetov. Taktiež sú v tejto kapitole uvedené niektoré z príkladov využitia detekcie chodcov v praxi. Výskum v tejto časti práce slúžil na zistenie toho, aké algoritmy budú využité v praktickej časti tejto práce. Podľa článku [9] majú *state-of-the-art* algoritmy na detekciu chodcov problém sa prispôsobovať veľkej hustote chodcov a rôznorodým podmienkam, hlavne z dôvodu trénovania a testovania na podobných datasetoch (ak nie rovnakých), ale taktiež aj z dôvodu, že samotné datasety nie sú dostatočne rôznorodé. Tieto poznatky sú dôležité pre rozhodnutie, ktorý algoritmus zvoliť pre detekciu a ako ho správne vytrénovať. *State-of-the-art* detekčné algoritmy používajú ako *backbone* (oporu) algoritmy na extrakciu príznakov zo snímkov. Táto spolupráca algoritmov dosahuje veľmi dobré výsledky. Ako posledné, ale taktiež dôležité sú v tejto kapitole spomenuté trénovacie datasety.

3.1 Príklady

V tejto sekcii sú uvedené praktické príklady využitia detekcie chodcov.

3.1.1 Detekcia chodcov pre autonómne vozidla

Jedno z najambicióznejších využití počítačového videnia v histórii. Detekcia chodcov za jazdy z vozidla predstavuje veľkú výzvu, a to hneď z niekoľkých dôvodov. Veľký problém predstavuje rýchlosť použitého algoritmu. Nie všetky detekčné a rozpoznávacie algoritmy majú dostatočnú rýchlosť na to, aby mohli byt využité pre *real-time* aplikácie. Napríklad výskum [17], ktorý skúma a porovnáva 3 riešenia strojového učenia pre detekciu objektov na základe videozáznamu, vyhodnotil, že algoritmus *Support Vector Machine* (SVM) nie je dostatočne rýchly na *real-time* detekciu, a práve preto sa celkovo skôr odporúčajú *deep learning* algoritmy. Z dôvodu, že sa jedná o systém, na ktorom závisia ľudské životy, je taktiež potrebná extrémna presnosť algoritmu, kde každý *false negative* ale aj *false positive* výsledok môže znamenať nehodu. Náročnosť problému taktiež zvyšuje nutnosť fungovania v rôznych viditeľnostných podmienkach a taktiež v rôznom prostredí. Z tohto dôvodu algoritmy využívajúce počítačové videnie nie sú jedinou zložkou detekcie okolia pre takéto vozidla. Taktiež treba podotknúť, že detekcia chodcov je len jedno s využití detekčných algoritmov vo vozidlách, pretože vozidlo musí byt schopne detegovať a aj rozpoznávať veľké množstvo objektov, napríklad ostatné vozidla, dopravne značky alebo ohraničenie cesty.

Dôležitosť dobrého detekčného algoritmu je znásobená, ak sa jedná o úplne autonómne vozidlo. Mohlo by sa teda zdať, že využitie počítačového videnia v tomto prípade nie je vhodný, avšak keďže nám ide o to zachrániť čo najviac životov a zamedziť nehodám, tak týmto systémom stačí dosiahnuť lepšie výsledky ako ľudský šoféri. Preto už v dnešnej dobe môžeme sledovať nasadenie autonómnych vozidiel na skutočné cesty. Vozidla sú schopné jazdiť, zastavovať na semaforoch, spomaliť a dokonca sa aj zaparkovať. Vozidla sú vybavené mnohými senzormi, ako napríklad ultrazvuk, *Light Detection and Ranging* (LiDAR) a kamery s 360 stupňovým výhľadom.

Jednou zo spoločností, ktorá vyvíja autonómne vozidlá je spoločnosť Waymo [33]. Projekt spoločnosti Waymo začal v roku 2009 a bol známy ako *Google Self-driving Car* a tento projekt je aj dodnes podporovaný spoločnosťou Google. Vozidla Waymo boli testované v niekoľkých amerických mestách v rôznom počasí a podmienkach, tak aby sa ich vozidla boli schopne čo najlepšie prispôsobiť realite.

Niektoré dáta z týchto testov boli vydané vo *Waymo Open Dataset* [28] na podporu výskumu v tejto oblasti. Dataset obsahuje 1150 situácií, kde každá má okolo 20 sekúnd a obsahujú dobre synchronizované a vysokej kvalite zachytené kamerové záznamy a záznamy z LiDAR senzorov. Spoločnosť uvádza, že ich vozidlá sú schopné vnímať okolie až do vzdialenosti 300 metrov na všetky strany. Okrem osobných automobilov spoločnosť vyvíja aj nákladne auta, ktorých zavedenie do prevádzky by bolo významným míľnikom pre veľkú časť priemyslu. Nákladne autá využívajú rovnaké algoritmy na detekciu okolia ako osobné automobily, avšak museli byť upravené algoritmy riadenia vozidla, ktoré sa starajú o pridávanie plynu, brzdenie alebo zatáčanie.

3.1.2 Bezpečnostné systémy

Algoritmy na rozpoznávanie ľudí sa využívajú aj pri systémoch, ktoré monitorujú perimeter. V dnešnej dobe takéto systémy fungujú tak, že v prípade nejakého ohrozenia majetku alebo inej závažnej situácii, automaticky zašlú upozornenie majiteľovi tohto systému, či už sa jedná o upozornenie pracujúceho strážnika na bezpečnostnom panely alebo majiteľa stráženého objektu priamo na jeho *smartphone*. V niektorých prípadoch môže byť nutné využiť algoritmus na rozpoznanie ľudí na rozdiel od jednoduchších pohybových senzorov, napríklad keď sa v okolí perimetru pohybuje divoká zver a nechceme zbytočné falošné poplachy. Ešte komplexnejšou situáciou môžu byt perimetre, kde sa zvyčajne pohybujú ľudia a chceme rozpoznať iba prípady, kedy nejaká osoba vykoná akciu, ktorú považujeme za ohrozenie majetku. V takomto prípade sa väčšinou zavedie druhý algoritmus, ktorý je vycvičený na rozpoznávanie ľudskej aktivity a nie jej detekciu.

Ďalšou problematickou časťou môžu byť premenlivé podmienky osvetlenia. Tento problém sa dá vyriešiť využitím algoritmov, ktoré v prípade zlých podmienok upravia videozáznam, ktorý sa následne pošle ďalej rozpoznávacím algoritmom [15]. Poprípade pre funkčnosť systému v noci sa využívajú termálne kamery.

Jednou zo spoločností, ktorá ponúka riešenia monitorovania perimetru, je spoločnosť MOBOTIX. Táto spoločnosť ponuka veľkú škálu riešení, od kamier cez rôzne iné senzory až po softvérové riešenia. Jeden z ich najzaujímavejších produktov je inteligentná detekcia pohybu *MxActivitySensor 2.0*. Prvú verziu tohto detekčného algoritmu spoločnosť vydala už v roku 2013 a od vtedy bol používaný vo väčšine ich vizuálnych senzorov. Algoritmus je zodpovedný nie len za detekciu nejakého pohybu, ale aj za rozlíšenie typu pohybu, čím sa celému systému dramaticky zníži počet falošných poplachov. Algoritmus dokáže detegovať

pohybujúce sa objekty s veľkou presnosťou, a nespustí tak poplach v prípade pohybov v zábere akými sú dážď, sneženie alebo pohyb listov.

Nová verzia zavedená v roku 2017 podľa [2] znižuje výskyt falošných poplachov ešte viac. Napríklad, ak bolo bežné, že alarm spustil nejaký lietajúci vták, verzia 2.0 ponúka možnosť určenia veľkosti objektov, na ktoré má systém reagovať a systém vie túto veľkosť upraviť na základe perspektívy. Znamená to, že aj keď je nejaký objekt bližšie ku kamere, čo teda znamená že na zábere je väčší, systém to nevyhodnotí ako hrozbu. Systému sa dá nastaviť pomyselná hranica, ktorú ak nejaký objekt prekročí, bude považovaný za potenciálnu hrozbu, avšak pre väčšie sledované perimetre dokáže systém vyhodnotiť ako hrozbu aj objekt, ktorý sa tejto hranici približuje. Systém dokáže rozoznať aj nezvyčajný smer pohybu objektov, napríklad pohyb auta v protismere alebo ak nejaký človek použije východ na vstup do objektu. Vysporiadanie s hrozbami sa môže líšiť od závažnosti a nastavuje si ich sám užívateľ.

3.1.3 Smart semafory

Smart semafory poskytujú riešenie pre zvýšenie efektivity dopravy v mestách. Vo väčšine miest v súčasnosti semafory využívajú systém, kde chodec musí stlačiť nejaké tlačidlo na semafore, aby dal signál na to, že chce prejsť cez cestu. Za využitia detekcie a rozpoznávania chodcov sa však navrhlo moderné riešenie tohto problému, kde systém využije kameru zabudovanú v semafore na detekciu chodcov. Tento návrh mal jeden kľúčový problém a to bol fakt, že systém nemôže brať každého chodca, ktorý prejde okolo semaforu ako chodca, ktorý chce prejsť na druhú stranu cesty. Riešením tohto problému bolo vycvičenie algoritmu, ktorý by bol schopný rozpoznať, či chodec chce prejsť cez cestu alebo nie.

Toto riešenie v roku 2019 zaviedlo do prevádzky mesto Viedeň v spolupráci s Technickou Univerzitou v Grazi [20], kde postupne začali vymieňať bežné semafory za semafory so vstavaným systémom na rozpoznávanie chodcov. Dokončenie výmeny všetkých semaforov sa očakáva koncom roku 2020. Toto riešenie taktiež adresuje problém, kedy cez prechod chce prejsť veľká skupina ľudí a predĺži tak zelenú pre chodcov. Naopak, ak bol detekovaný človek, ktorý chcel prejsť cez cestu, ale rozmyslel si to, a ešte predtým ako sa semafor stihol prepnúť, odíde, tak to systém zaznamená a nespomalí tak premávku áut. Kamera zaznamenáva pole s rozmerom 8×5 metrov a algoritmus vyhodnotí úmysel chodca prejsť cez cestu do 2 sekúnd, čo podľa výskumu znamená, že o úmysle chodca sa systém dozvie v priemere 3 až 4 sekundy skôr v porovnaní so semaformi s tlačidlom. So zavedením kamier sa objavili obavy o ochrane osobných údajov, avšak bolo potvrdené, že každý semafor pracuje len s lokálnymi dátami, ktoré nikde ďalej neposiela.

3.1.4 Smart prechody pre chodcov

Smart prechody pre chodcov boli vyvinuté za účelom zvýšenia bezpečnosti pre chodcov v mestách. Systém smart prechodu rozpozná chodca, ktorý chce prejsť cez cestu na prechode bez svetelného označenia, a dokáže upozorniť vodičov motorových vozidiel výstražným signálom. Jedno z takýchto riešení ponúka spoločnosť Bercman [3], kde využívajú slabú umelú inteligenciu, ktorá je schopná odhadnúť trajektórie vozidiel a chodcov a v prípade možného nebezpečenstva ich vie varovať. Na varovanie chodcov používajú zvukový signál a na varovanie vodičov signál svetelný, ktorý je umiestený na samotnej značke prechodu pre chodcov.

Toto riešenie taktiež podporuje komunikáciu *Vehicle-to-everything* (V2X) [27], čo predstavuje komunikáciu medzi vozidlom a akýmkoľvek subjektom, ktorý môže ovplyvniť vozidlo alebo môže byt ovplyvnený vozidlom, s technológiou *dedicated short-range communication*

(DSRC), ktorá umožňuje komunikáciu s nízkou odozvou a bola špeciálne navrhnutá pre časovo kritické aplikácie, a taktiež *Cellular V2X* (C-V2X), ktorá je založená na mobilných bezdrôtových štandardoch. V prípade ak vozidlo, ktoré sa ocitne v nebezpečnej situácií na smart prechode pre chodcov, má podporu pre V2X komunikáciu, tak bude môcť prijať varovanie a zobraziť ho vodičovi napríklad na displeji na palubnej doske. Takéto prechody pre chodcov sú už v dnešnej dobe v prevádzke v mestách ako Tallinn a Helsinki.

3.1.5 Ďalšie príklady

Medzi ďalšie príklady využitia detekcie chodcov patria: počítanie ľudí, sledovanie pohybu v obchode za účelom zvýšenia predaja, sledovanie pacientov, ktorý môžu odpadnúť, či už doma alebo v nemocnici. Algoritmy na počítanie pracujú nad nejakým pomyselným kontrolným bodom (checkpoint) a ich výstupy môžu mať rôzny štatistický význam. Takéto algoritmy taktiež musia byť dobre vycvičené, aby si vedeli poradiť s prípadmi, kde sa ľudia prekrývajú. Pri algoritmoch, ktoré sledujú správanie ľudí v obchode, ide hlavne zistenie, kam umiestniť produkty, tak aby si ich čo najviac ľudí všimlo, či už ide o to do ktorej uličky produkt dať alebo aj o špecifické miesto v regáli. Z týchto príkladov je zrejmé, prečo je oblasť detekcie chodcov veľmi dôležitou témou.

3.2 Detekčné algoritmy

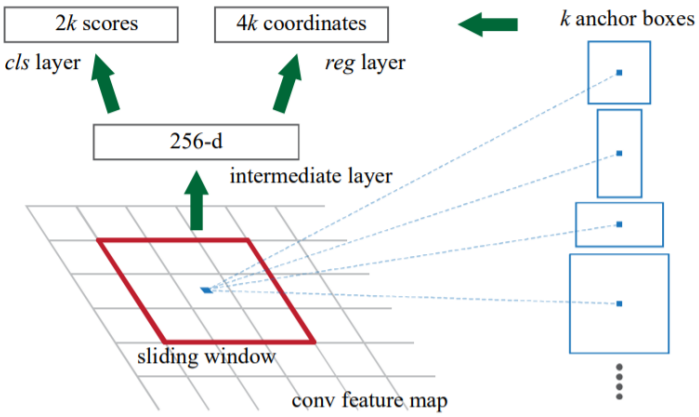
V tejto sekcii sú predstavené nasledujúce algoritmy: *Cascade Mask R-CNN* [5], *Faster R-CNN* [23], *Hybrid Task Cascade* [6], *Center and Scale Prediction* [16].

3.2.1 Faster R-CNN

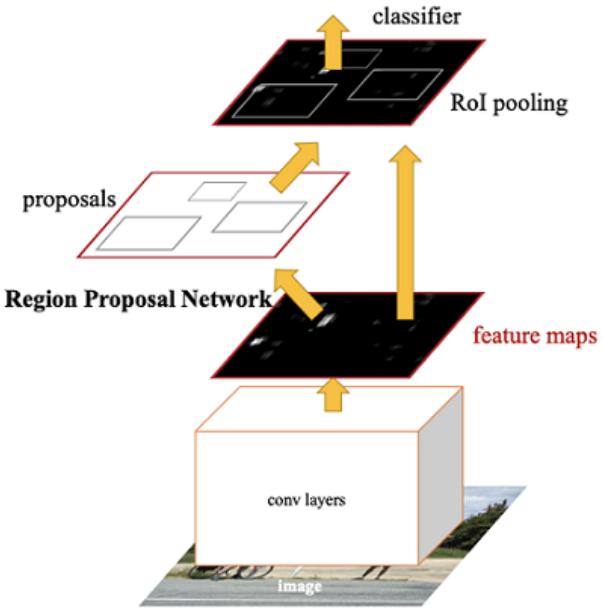
Faster R-CNN [23] je algoritmus na detekciu objektov, ktorý stavia na a rozširuje algoritmus *Fast R-CNN* využitím 2 modulov. Jedným z modulov je *Region Proposal Network* (RPN), čo je hlboká plne konvolučná neurónová sieť, a druhý je detekčný algoritmus *Fast R-CNN*. Schéma siete *Faster R-CNN* je zobrazená na obr. 3.2. Modul RPN zdieľa konvolučné príznaky zo snímky s detekčnou sieťou, čo umožňuje veľmi efektívne návrhy regiónov. RPN taktiež predpokladá hranice objektov a udáva *objectness scores* [7] (skóre objektivity) pre každú pozíciu. Skóre objektivity je definované na meranie toho, ako dobre detektor identifikuje polohy a triedy objektov počas procesu transformácie polohy objektov v 3D svete na 2D súradnice snímky. Toto skóre ovplyvňuje napríklad vzdialenosť objektu od kamery a pomáha pri rozhodovaní či ponechať triedu detekovaného objektu. RPN sieť je trénovaná *end-to-end*, čo znamená že váhy v sieti sú optimalizované zvážením vstupov a výstupov celej siete. Tento štýl trénovania uisťuje vysokú kvalitu návrhu regiónov, ktoré využíva detektor *Fast R-CNN*.

RPN pracuje na princípe posuvného okna nad mapou príznakov a jej výstupom je niekoľko možných ohraničených regiónov a ich skóre objektivity [23] (viz. obr. 3.1). Tieto regióny však môžu mať rôzne veľkosti, čo znamená rôzne veľké mapy príznakov pre CNN. Nie je jednoduché vytvoriť efektívnu štruktúru pre prácu s rôzne veľkými mapami príznakov, a preto sa využíva *Region of Interest* (ROI) pooling, tak aby redukoval mapy príznakov do rovnakej veľkosti. Narozdiel od max poolingu, ktorý má pevnú veľkosť, ROI pooling rozdeľuje vstupnú mapu príznakov na stanovené číslo (napríklad k) zhruba rovnakých regiónov, a potom aplikuje metódu max pooling na každý región zvlášť. Čo znamená, že výstup bude vždy k bez ohľadu na veľkosť vstupu. RPN a Fast R-CNN sú zlúčené do jednej siete

zdieľaním ich konvolučných príznakov. RPN časť tak udáva celkovej sieti, kde má objekty hľadať. Ako celok je táto sieť nazývaná Faster R-CNN.



Obr. 3.1: Výber oblastí (pozícia aj skóre objektivity) modulom RPN [23].



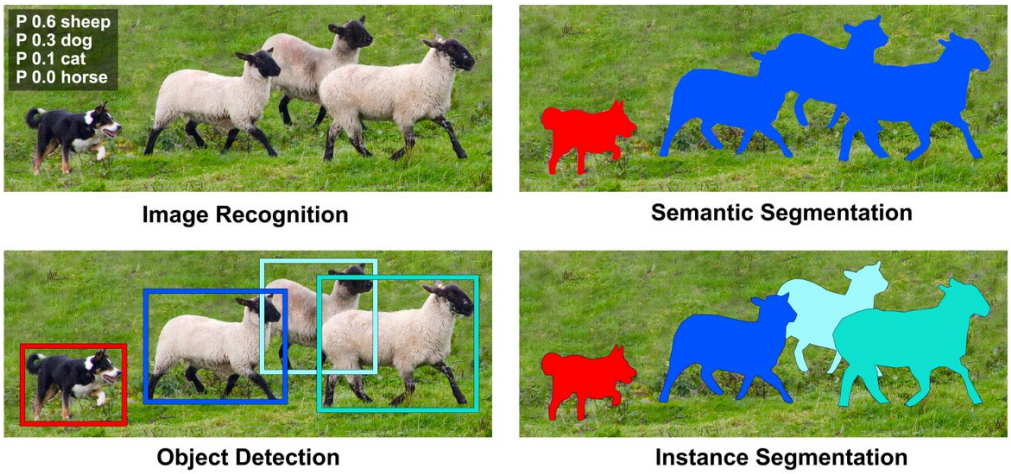
Obr. 3.2: Zjednotená sieť Faster R-CNN, kde RPN slúži na získanie navrhovaných regiónov z mapy príznakov [23].

3.2.2 Cascade Mask R-CNN

Cascade Mask R-CNN [5] rozširuje *Cascade R-CNN* [5] o segmentáciu inštancií, ktorú zavádza algoritmus *Mask R-CNN* [10]. Z tohto dôvodu je dôležité vysvetliť Mask R-CNN a Cascade R-CNN zvlášť.

Mask R-CNN [10] rozširuje Faster R-CNN (3.2.1) o riešenie úloh segmentácie inštancií. Narozdiel od sémantickej segmentácie, segmentácia inštancií nie len, že rozdeľuje detekované

objekty do príslušných tried, ale rozdeľuje aj samotné výskyty triedy (viz. obr. 3.3). Toto je dosiahnuté pridaním vetvy na predpovedanie masky objektu (tzv. segmentačná vetva) paralelne k existujúcej vetve na rozpoznávanie ohraničujúceho poľa. Jedná sa tak o intuitívne rozšírenie Faster R-CNN, ale správna konštrukcia vetvy pre masku je kritická pre dobrý výsledok. Faster R-CNN nie je navrhnutá na *pixel-to-pixel* zarovnanie medzi vstupmi a výstupmi v sieti. Toto východza z faktu, že Faster R-CNN využíva ROI pooling. Na opravu zarovnania Mask R-CNN využíva vrstvu bez kvantovania, ktorá sa nazýva ROIAlign. Táto vrstva tak verne zachováva presné priestorové lokácie. Mask R-CNN oddeľuje predikciu masky a triedy. To znamená, že predpovedá binárnu masku pre každú triedu nezávisle, bez konkurencie medzi triedami a pre predikciu kategórie sa spolieha na vetvu klasifikácie.



Obr. 3.3: Vľavo rozdiel medzi rozpoznávaním obrazu a detekciou objektov. Vpravo rozdiel sémantickej segmentácie a segmentácie inštancií [24].

Cascade R-CNN [5] je algoritmus pre detekciu objektov, ktorý sa snaží riešiť problém so znížením výkonu pri zvýšení prahových hodnôt *Intersection over Union* (IoU). IoU je využívané na meranie presnosti detekčných algoritmov nad konkrétnym datasetom. Na výpočet presnosti týmto spôsobom potrebujeme dve sady ohraničujúcich boxov. Jedna sada sú boxy, ktoré reprezentujú, kde na snímke sa objekt skutočne nachádza, a druhá sada sú boxy, ktoré sú výstupom nášho modulu. Presnosť sa potom rovná:

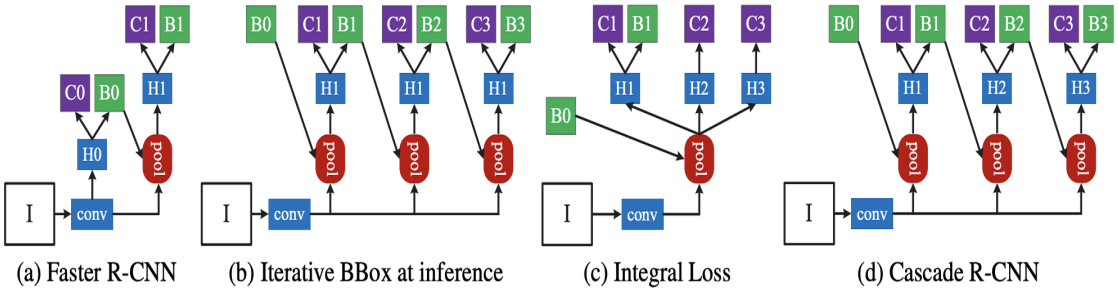
$$IoU = \frac{\text{plocha prieniku boxov}}{\text{plocha zjednotenia boxov}}$$

Čím bližšie je výsledok k jednotke, tým väčšia je presnosť algoritmu pre daný príklad (obr. 3.4). Problémy so zvyšovaním prahových hodnôt IoU môžu byť pretrénovanie siete (overfitting), kedy sieť sa až príliš prispôsobí na dataset, na ktorom je trénovaná. Cascade R-CNN je viacstupňové rozšírenie R-CNN, kde detekčné stupne hlbšie v kaskáde sú sekvenčne viac selektívne proti malým false positive chybám. Cascade R-CNN stupne sú trénované sekvenčne, kde výstup jedného stupňa sa využíva na trénovanie ďalšieho stupňa. Toto je motivované pozorovaním, kde IoU výstupu predchádzajúceho stupňa je takmer vždy lepšie ako IoU vstupu.

Cascade R-CNN [5] sa nezameriava na veľké negatívy. Namiesto toho sa snaží malou úpravou ohraničujúcich boxov v každom stupni nájsť dobrú množinu false positive prípadov pre trénovanie ďalšieho stupňa. Týmto spôsobom sekvencia detektorov adaptovaná k

Obr. 3.4: Ukážka presnosti detekcie metódou IoU. Červený štvorec predstavuje ohraničujúci box, v ktorom sa detekovaný objekt skutočne nachádza, a zelený štvorec predstavuje box, ktorý je výstupom detekčného algoritmu. Na príklade vľavo je vidieť, že táto metóda si poradí aj s prípadom, kedy výsledné boxy síce obsahujú hľadané objekty, ale sú príliš veľké.

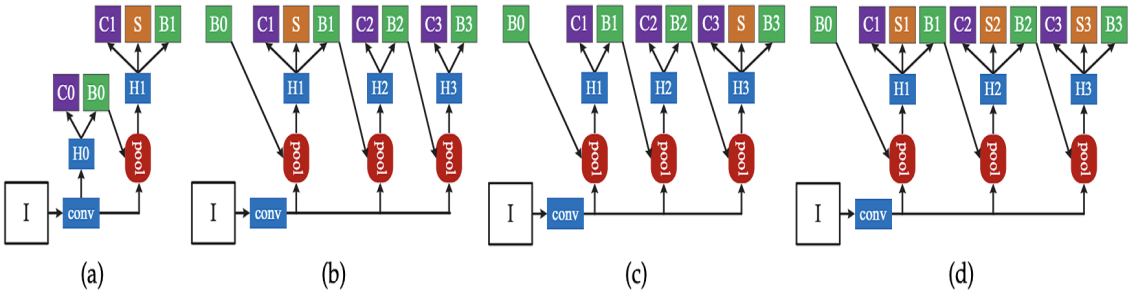
postupne sa zvyšujúcim prahovým hodnotám IoU dokáže poraziť problém s overfittingom a môže tak byť efektívne vytrénovaná. Progresívne zlepšované hypotézy sa lepšie prispôsobujú zvyšujúcej sa kvalite detektora v každom stupni.



Obr. 3.5: Ukážka rôznych architektúr. I = vstupný obrázok, conv = backbone algoritmus (CNN), pool = regionálne extrakcie príznakov, H = hlava siete, B = ohraničujúci box, C = klasifikácia a B0 = navrhované regióny [5].

Cascade Mask R-CNN [5] stavia na spojení Cascade R-CNN a Mask R-CNN. Mask R-CNN vkladá segmentačnú vetvu paralelne k detekčnej vetve. Avšak, Cascade Mask R-CNN má viac detekčných vetiev. Toto tak vytvára dve dôležité otázky: 1) kam sa má segmentačná vetva pridať a 2) koľko segmentačných vetiev sa má pridať. V článku [5] autori navrhujú tri stratégie pre riešenie tohto problému. Prvé dve stratégie adresujú otázku číslo jedna. Jedná sa o pridanie jedinej hlavy na predpoveď masky objektu, buď na prvý alebo posledný stupeň Cascade R-CNN. Keďže inštancie používané na trénovanie segmentačnej vetvy sú pozitíva z detekčnej vetvy, tieto dve stratégie sa značne líšia v ich počte. Vloženie hlavy na predpoveď masky na koniec kaskády vedie k vyššiemu počtu príkladov. Keďže pri segmentácii sa jedná o operáciu na pixeloch, veľké množstvo prekrývajúcich sa inštancií nie je nutne až tak nápomocné ako pri detekcii objektov. Tretia stratégia adresuje otázku číslo dva. Autori navrhujú, že je možné pridať segmentačnú vetvu do každého stupňa kaskády. Toto maxima-

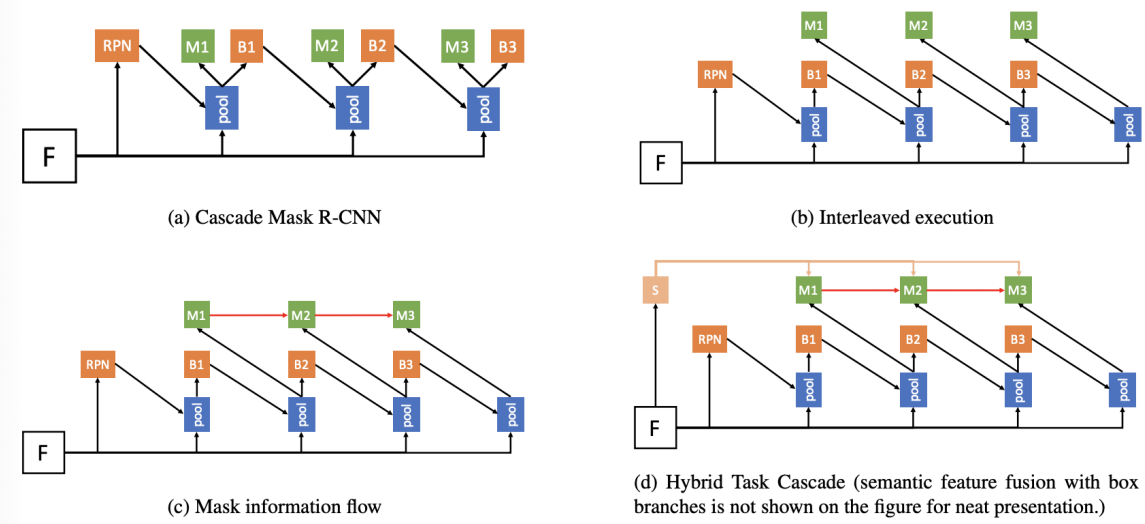
lizuje rôznorodosť inštancií použitých pre trénovanie vetvy na predpoveď masky objektu. Schémy týchto stratégií sú znázornené na obrázku 3.6. Všetky tri stratégie predpovedajú masku na objekte produkovanom na poslednej detekčnej vetve bez ohľadu na to, kde je segmentačná vetva umiestnená a koľko segmentačných vetiev kaskáda obsahuje.



Obr. 3.6: Ukážka prístupov umiestenia segmentačnej vetvy. (a) schéma základnej Mask R-CNN, (b) stratégia umiestnenia segmentačnej vetvy do prvého stupňa kaskády, (c) umiestnenie do posledného stupňa kaskády, (d) umiestnenie do každého stupňa. Označenia častí v schéme je rovnaké ako v obr. 3.5 až na označenie S = segmentačná vetva. Schémy (b)-(d) predstavujú Cascade Mask R-CNN [5].

3.2.3 Hybrid Task Cascade

Hybrid Task Cascade (HTC) [6] je framework, ktorý sa tiež snaží riešiť otázku segmentácie inštancií pre kaskádové riešenia. Autori HTC uvádzajú, že Cascade Mask R-CNN je síce jednoduchou kombináciou už zavedených algoritmov, avšak táto kombinácia prináša iba obmedzené zlepšenie. Podľa ich výskumu je kľúčom k úspešnej segmentácii inštancií v kaskádach, plné využitie recipročného vzťahu medzi detekciou a segmentáciou. HTC sa od Cascade Mask R-CNN líši v dvoch hlavných aspektoch: 1) namiesto vykonávania kaskádového zlepšenia detekcie a segmentácie zvlášť, prelína ich na spoločné viac-stupňové spracovanie, 2) zavedenie plne konvolučnej vetvy na poskytnutie priestorového kontextu, čo pomáha rozlíšiť popredie snímky od preplneného pozadia. Tieto zmeny znamenajú nie len zvýšenie efektivity toku informácii medzi stupňami v kaskáde, ale aj pre jednotlivé úlohy. HTC je taktiež trénované end-to-end spôsobom. Autorom tohto frameworku sa už podarilo dosiahnuť na niektorých datasetoch lepšie výsledky ako Cascade Mask R-CNN. Na obr. 3.7 sú zobrazené architektúry kaskádových sietí a postup vývoja od Cascade Mask R-CNN po Hybrid Task Cascade.

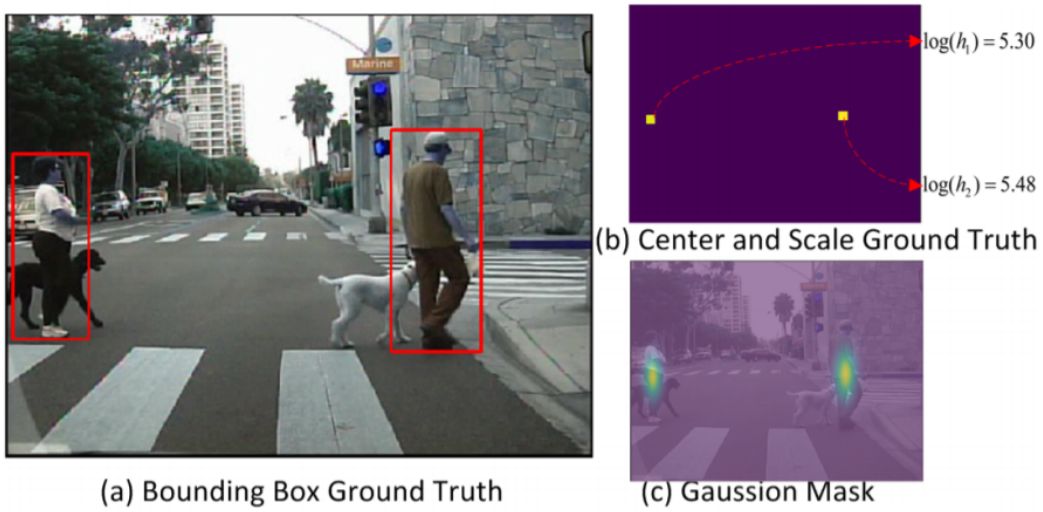


Obr. 3.7: Ukážka vývoja architektúry od Cascade Mask R-CNN po Hybrid Task Cascade [6].

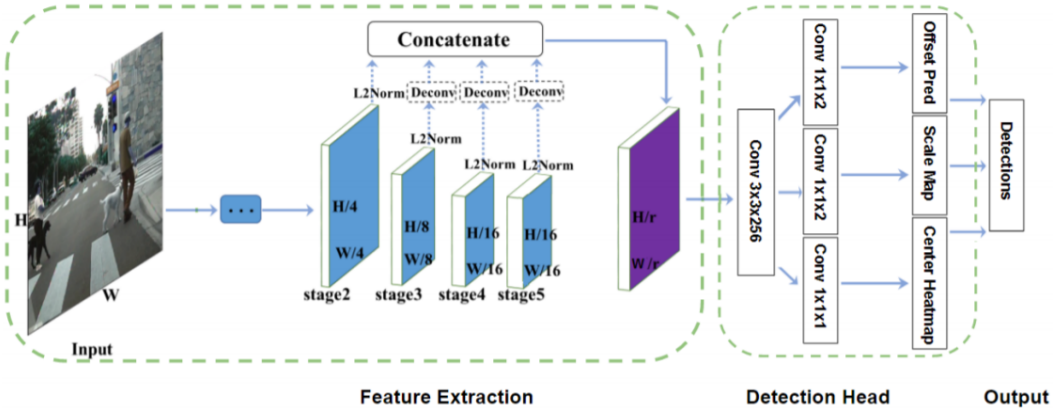
3.2.4 Center and Scale Prediction

Center and Scale Prediction (CSP) [16] je detekčný algoritmus, ktorý sa pokúša o tzv. *box-free* detekciu, čo znamená detekciu bez využitia posuvného okna alebo anchor box predikcií (napr. RPN). Tento spôsob tak zavádza do otázky riešenia detekcie objektov novú perspektívu, kde detekcia je motivovaná ako high-level úloha na detekciu semantických príznakov. Navrhovaný detektor hľadá príznaky po celej snímke, tak ako tomu je aj pri ostatných moderných detektoroch, a teda využíva konvolúciu. Avšak, narozdiel od týchto detektorov, ktoré sa zameriavajú na low-level príznaky, sa CSP snaží o vyššiu mieru abstrakcie. CSP sa snaží nájsť stredové body objektov (viz. obr. 3.8), kde moderné modely už sú schopné si poradiť aj s takouto vysokou mierou abstrakcie. Dá sa teda povedať, že táto metóda zjednodušuje úlohu detekcie chodcov na úlohu predpovede stredu objektu a váhy danej predpovede cez konvolúciu. Aj napriek tomu, že sa jedná o štrukturálne jednoduchý model, CSP dosahuje vysokú presnosť aj v porovnaní s ostatnými state-of-the-art algoritmami.

Celkovo sa CSP skladá z časti na extrakciu príznakov a detekčnej hlavy (obr. 3.9) [16]. Na extrakciu príznakov využíva, tak ako aj ostatné moderné detekčné algoritmy, backbone algoritmus. Ak uvažujeme vstupný snímok o veľkosti $H \times W$, tak výsledná mapa príznakov, ktorá je vytvorená spojením všetkých jednotlivých máp, má veľkosť $H/r \times W/r$, kde r predstavuje *downsampling factor*. Experimentami bola zistená optimálna hodnota $r = 4$. Pre vyššie hodnoty r je výsledná mapa príznakov v príliš malej kvalite na to, aby bolo možné spraviť presné predpovede a zas pre menšie hodnoty sa značne zvyšuje výpočtová záťaž. Detekčná hlava potom pracuje nad touto výslednou mapou a jej výstupom sú výsledky detekcie. V článku [16] je navrhnuté využitie jednej 3x3 konvolučnej vrstvy na redukciu počtu kanálov, a potom tri súrodenecké 1x1 konvolučné vrstvy na získanie predikcií o offsete, pozícii stredu a odpovedajúcej váhe.



Obr. 3.8: Celkový prehľad pipeline pre CSP detektor. Finálna konvolúcia má dva kanály. Horná *center heatmap* indikuje pozície stredov a dolná *scale map* slúži na predpoveď váhy pre jednotlivé stredy. V časti *detections* sú stredy vyznačené červenými bodmi a váhy bodkovanými oranžovými čiarami [16].



Obr. 3.9: Celková architektúra CSP. Skladá sa z dvoch hlavných zložiek: extrakcia príznakov a detekčnej hlavy. Modul na extrakciu príznakov spája mapy príznakov s rôznymi rozlíšeniami do jednej. Detekčná hlava obsahuje 3x3 konvolučnú vrstvu, po ktorej nasledujú 3 predikčné vrstvy: predpoveď offsetu, pozícia stredu a odpovedajúca váha [16].

3.3 Algoritmy na extrakciu príznakov

V tejto sekcii sú predstavené nasledujúce state-of-the-art algoritmy, ktoré sa využívjú ako backbone pre detekčné algoritmy predstavené v predchádzajúcej sekcii: *ResNeXt* [35], *HRNet* [32], *EfficientNet* [29].

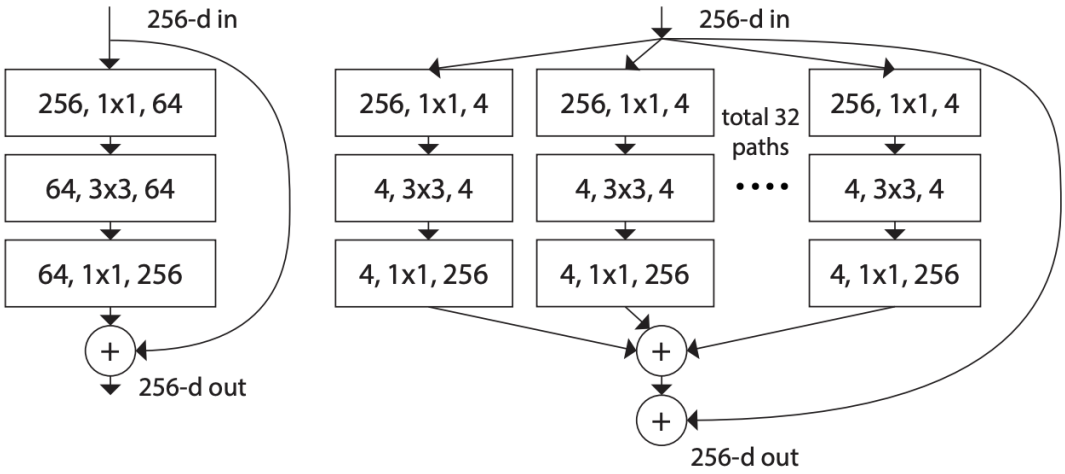
3.3.1 ResNeXt

Reziduálne (Zvyškové) siete označované *ResNets* [11] sa učia zvyškové funkcie s referenciou na vstupnú vrstvu, namiesto učenia sa nereferencovaných funkií. Siete sú tak oveľa jednoduchšie na optimalizáciu. Tieto siete zavádzajú reziduálny blok a bottleneck blok s komponentom *shortcut* (skratka). Reziduálny blok sa skladá z dvoch 3x3 konvolučných vrstiev, aktivačnej funkcie ReLU a zo skratky. Bottleneck blok má namiesto dvoch konvolučných vrstiev tri vrstvy, pričom dve z nich sú o veľkosti 1x1 a jedna 3x3. Schéma bottleneck bloku na obr. 3.10. ResNets tieto bloky ukladajú na seba a formujú tak sieť. Skratka slúži na preskočenie jednej alebo viacerých vrstiev a rozdeľuje sa na dva typy, a to identita a projekcia. Identita nepridáva žiadny nový parameter a používa sa, keď majú vstup aj výstup rovnaku dimenziu. Vzorec na výpočet bloku, kde x je vstup, y výstup a $\mathcal{F}(x, \{W_i\})$ je funkcia predstavujúca reziduálne mapovanie, ktoré sa má učiť:

$$y = \mathcal{F}(x, \{W_i\}) + x$$

Projekcia sa využíva, keď dimenzie vstupu a výstupu nie sú rovnaké. Pridáva parameter W_s, ktorý sa využíva na prispôsobenie dimenzií pomocou 1x1 konvolúcie. Vzorec výpočtu sa tak zmení na:

$$y = \mathcal{F}(x, \{W_i\}) + W_s x$$



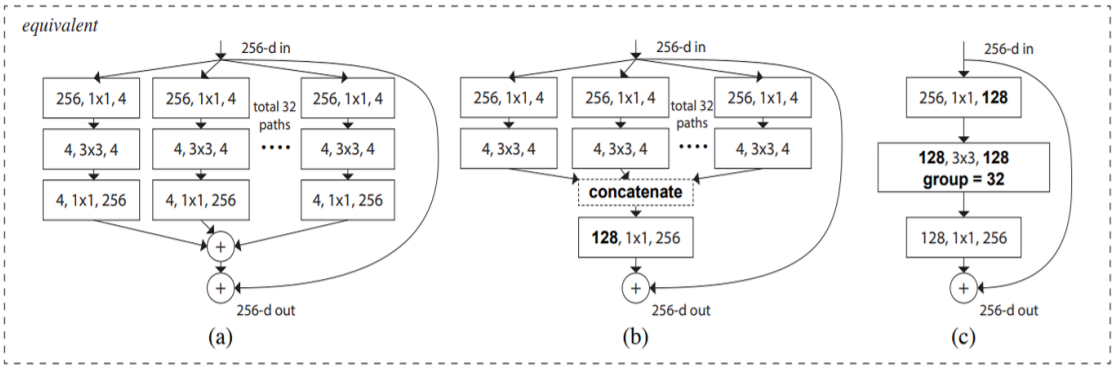
Obr. 3.10: Schéma blokov v reziduálnych sieťach. Naľavo: blok v sieti ResNet, napravo: blok v sieti ResNeXt s kardinalitou 32 so zhruba rovnakou komplexitou. Značenie čísiel = vstupné kanály, veľkosť filtra, výstupné kanály [35].

Sieť ResNeXt [35] opakuje stavebný blok, ktorý agreguje množinu transformácií s rovnakou topológiou. V porovnaní s klasickou sieťou ResNet vystavuje novú dimenziu nazývanú kardinalita (veľkosť množiny transformácií) ako zásadný faktor k dimenziám hĺbky a šírky.

Porovnanie blokov je znázornené na obrázku 3.10. Formálne môže byť množina súhrnných transformácii reprezentovaná ako:

$$\mathcal{F}(x) = \sum_{i=1}^{C} \mathcal{T}_i(x)$$

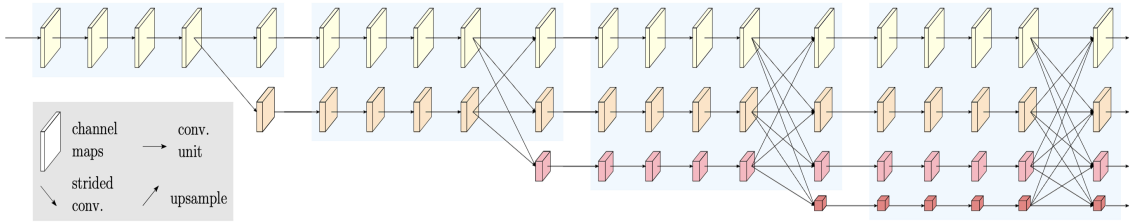
, kde $\mathcal{T}_i(x)$ môže byť ľubovoľná funkcia. Analogicky k jednoduchému neurónu, by mala $\mathcal{T}_i$ premietnuť x do vnorenia (embedding), a potom ho transformovať.



Obr. 3.11: Ukážka ekvivalentných stavebných blokov siete ResNeXt. (a) Agregované reziduálne transformácie, rovnaké ako na obrázku 3.10. (b) Ekvivalent k (a) implementovaný ako skoré zreťazenie (concatenation). (c) Ekvivalent k (a) a (b) implementovaný ako združené konvolúcie. Značenie čísiel = vstupné kanály, veľkosť filtra, výstupné kanály [35].

3.3.2 HRNet

HRNet (High-Resolution Net) [32] konvolučná neurónová sieť pre všeobecné účely, akými sú sémantická segmentácia, odhad ľudskej pózi, detekcia objektov a klasifikácia snímkov. Je schopná udržať vysoké rozlíšenie počas celého jej procesu. Začína sa jedným konvolučným prúdom s vysokým rozlíšením a postupne po jednom pridáva konvolučné prúdy s *high-to-low* rozlíšením a paralelne pripája *multi-resolution* prúdy. Výsledná sieť tak pozostáva z n stupňov, kde každý n-tý stupeň obsahuje n prúdov zodpovedajúcich n rozlíšeniam. V článku [32] používajú autori $n = 4$, čo je zobrazené aj na obr. 3.12. K zachovaniu vysokej kvality taktiež prispieva opakovaná výmena informácií medzi rôznymi rozlíšeniami. Hlavnou výhodou teda je, že výsledná reprezentácia sémanticky bohatšia a priestorovo precíznejšia. Keďže HRNet ukazuje sľubné výsledky v širokej škále problémov, dá sa považovať za silný backbone algoritmus pre problémy počítačového videnia.

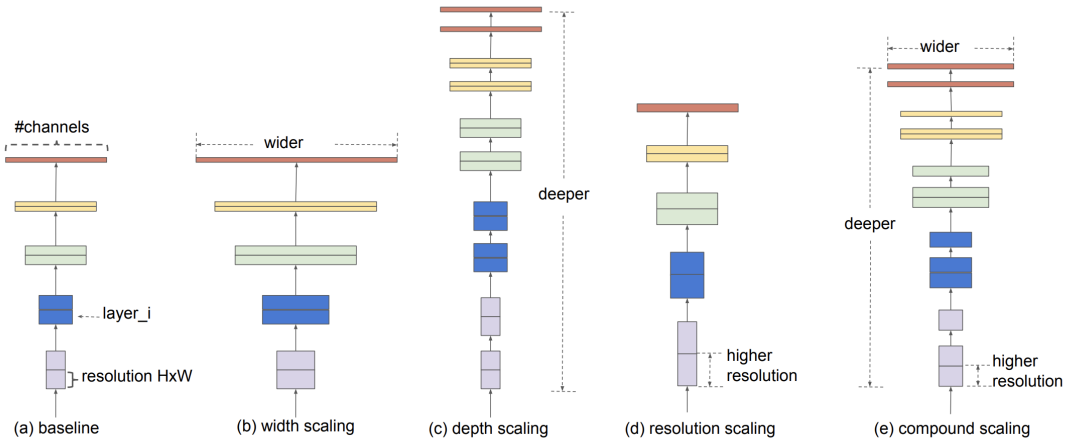


Obr. 3.12: Príklad HRNet siete. Ilustrované je iba hlavné telo siete. Sieť pozostáva zo 4 stupňov. Prvý stupeň je zložený z konvolúcií s vysokým rozlíšením. Ostatné n-té stupne opakujú bloky s n rozlíšeniami [32].

3.3.3 EfficientNet

EfficientNet [29] je architektúra konvolučnej neurónovej siete a škálovacia metóda, ktorá jednotne škáluje všetky dimenzie: hĺbka, šírka a rozlíšenie. Na toto používa tzv. *compound coefficient* (zložený koeficient). Narozdiel od tradičných prístupov, ktoré tieto dimenzie arbitrárne škálujú, EfficientNet škáluje všetky dimenzie jednotne a naraz za použitia koeficientov. Napríklad, ak chceme využiť o 2^N viac počítačových zdrojov, môžeme zväčšiť hĺbku siete o α^N, šírku siete o β^N a rozlíšenie siete o γ^N, kde α, β, γ sú konštantné koeficienty, ktoré sa dajú určiť pomocou pôvodného modelu siete. Toto škálovanie je zobrazené na obr. 3.13. Zložený koeficient, ktorý tento model využíva na jednotné škálovanie, sa označuje ϕ. Metóda zloženého škálovania vychádza z intuície, že ak je vstupný snímok väčší, tak potom sieť potrebuje viac vrstiev na zväčšenie vnímaného poľa a viac kanálov na zachytenie jemnejších vzorov vo väčšej snímke.

Autori EfficientNet [29] taktiež navrhli vlastnú architektúru siete, keďže podľa ich práce je pre efektívne škálovanie potrebné mať aj dobrú základnú sieť. Na tejto sieti tak ukázali väčší potenciál ich škálovania, ale takisto ukázali potenciál škálovania iných sietí ako napríklad ResNets alebo MobileNets.



Obr. 3.13: Ukážka škálovania siete pomocou EfficientNet. (a) Pôvodný model siete pred škálovaním. (b)-(d) klasický spôsob škálovania jednej z dimenzií siete. (e) Škálovanie pomocou EfficientNet, ktorý jednotne škáloju všetky tri dimenzie siete [29].

3.4 Datasety

V tejto sekcii je uvedených niekoľko datasetov, ktoré sa používajú na trénovanie sietí pre detekciu chodcov. Uvedené datasety: CrowdHuman [26], WiderPerson [37], CityPersons [36].

3.4.1 CrowdHuman

Dataset CrowdHuman [26] je jeden z datasetov, ktoré obsahujú v priemere najviac ľudí na jeden snímok datasetu. Dataset obsahuje 15000 snímkov a cez 330000 ľudí, čo v priemere znamená viac ako 22 osôb na jednej snímke. Tento set je tak dobrý na vycvičenie inštančnej segmentácie. Veľká hustota ľudí v snímkoch a veľká diverzita prostredí adresuje problémy, ktoré popisuje článok [9] o moderných prístupoch k trénovaniu algoritmov. Set tak poskytuje základ pre dobrú generalizáciu siete pre detekciu ľudí.

3.4.2 WiderPerson

WiderPerson [37] sa snaží adresovať dva problémy súčasných datasetov pre detekciu chodcov: 1) veľká časť súčasných datasetov obsahuje len zábery z áut, čo vedie k nedostatočnej diverzite, 2) súčasné sety obsahujú málo príkladov, kde dochádza k veľkému prekrývaniu ľudí v zábere. Set obsahuje 13382 snímkov a takmer 400000 anotácií k snímkom. Príklady obrázkov z tohto datasetu sú zobrazené na obr. 3.14.

Obr. 3.14: Prehľad snímkov datasetu WiderPerson [37].

3.4.3 CityPersons

CityPersons [36] je postavený na datasete Cityscapes, špeciálne vytvorený pre trénovanie sietí na detekciu chodcov napríklad pre autonómne vozidlá. Set obsahuje 5000 snímkov s detailne prepracovanými anotáciami. Všetky dáta boli nasnímané za jazdy z auta. Jazdou v mestách ako Hamburg a Frankfurt boli obdržané aj snímky s veľkým počtom ľudí na jednom mieste a teda vysokou mierou prekrývania.

Kapitola 4

Multi-kamerové systémy a rozpoznávanie osôb

4.1 Detekcia chodcov v Multi-kamerových systémoch

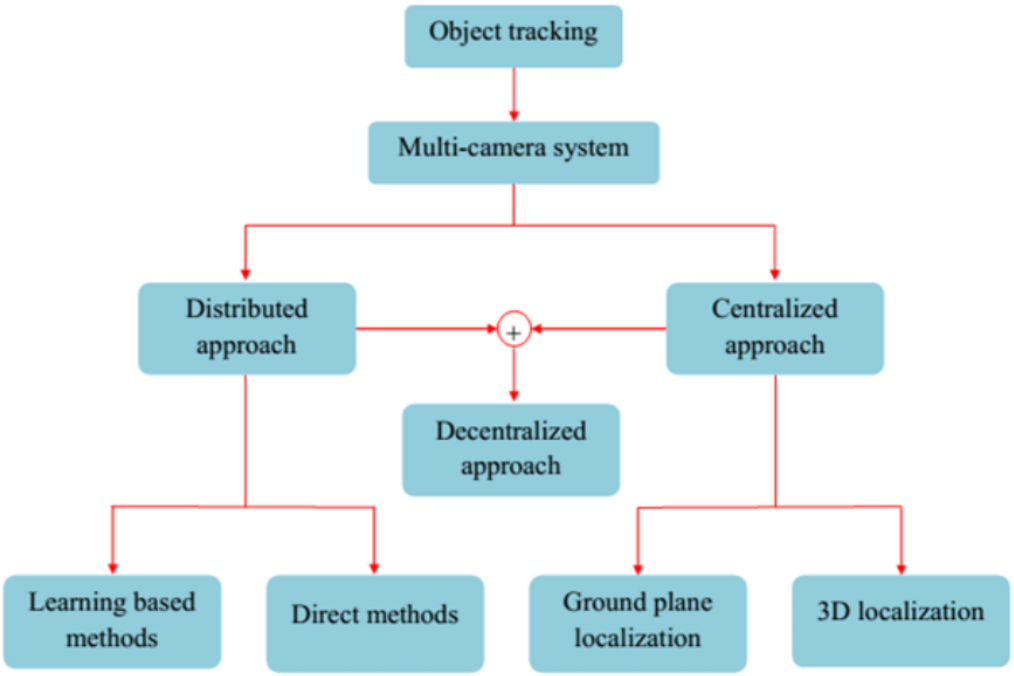
Multi-kamerové systémy môžu zaviesť do riešení rôzne výhody ako napríklad lepšie pokrytie pre zabezpečovacie systémy alebo väčšia presnosť detekcie, kde každá kamera poskytuje rôzny uhol pohľadu zaznamenávanej plochy. V dnešnej dobe je od kvalitných inteligentných zabezpečovacích systémov očakávané využitie viacerých kamier a ich vzájomné prepojenie. Podľa článku [13] existujú 2 hlavné prístupy k riešeniu detekcie v multi-kamerových systémoch: centralizované a distribuované prístupy. Toto rozdelenie je zobrazené na obr. 4.1. Dôležitou otázkou pri týchto systémoch je, či sa pohľady kamier navzájom prekrývajú alebo nie. Ak sa neprekrývajú, k problému samotnej detekcie sa pristupuje rovnako ako pri jednoduchých systémoch s jednou kamerou. Ak sa však pohľady prekrývajú, otvára to viac možností riešenia problému. Hlavným problémom pre multi-kamerovú detekciu je teda prenos a spracovanie dát z rôznych kamier.

Centralizované prístupy zväčša pozostávajú z kombinácie dát z rôznych kamier. Takéto prístupy prinášajú výhody hlavne pre systémy s prekrývajúcimi sa pohľadmi kamier. Záznamy z kamier by tak mali byť synchronizované, aby nedochádzalo k chybám. Kombináciou dát môžme nájsť 3D lokalizáciu pre jednotlivé osoby alebo môžu byť jednotlivé detekcie premietnuté na rovinu, ktorá predstavuje plochu sledovanej scény. Takáto kombinácia dát očakáva, že systém pozná vzájomnú polohu kamier.

Distribuované prístupy sú využívané hlavne v systémoch, kde sa zábery kamier neprekrývajú. Takéto systémy sú využité napríklad na pokrytie veľkých oblastí. Ďalším dôvodom pre využitie distribuovaných prístupov môže byť fakt, že nepoznáme vzájomnú geometrickú polohu kamier, čo prináša istú výhodu oproti prístupom centralizovaným, keďže ich využitie je všeobecnejšie. Ďalšou dôležitou charakteristikou týchto prístupov je fakt, že ani samotné zábery nemusia byť synchronizované a systém sa s tým vie vysporiadať.

Posledným typom prístupov podľa článku [13] sú takzvané decentralizované prístupy. Tieto prístupy sa využívajú v prípadoch, kedy je sieť kamier príliš veľká a teda využitie centralizovaných prístupov by viedla k nadmernému prenosu dát, pričom využitie distribuovaných prístupov by vôbec nevyužilo potenciál takéhoto systému. Decentralizované prístupy sú akýmsi hybridným riešením, kde kamery sú zhromažďované do zhlukov a každý zhluk má nejaký lokálny centrálny bod, ktorý zbiera a spája dáta z jednotlivých kamier. V zhlukoch sa tak k problému pristupuje rovnako ako pri centralizovaných prístupoch. Ak sa jedná o

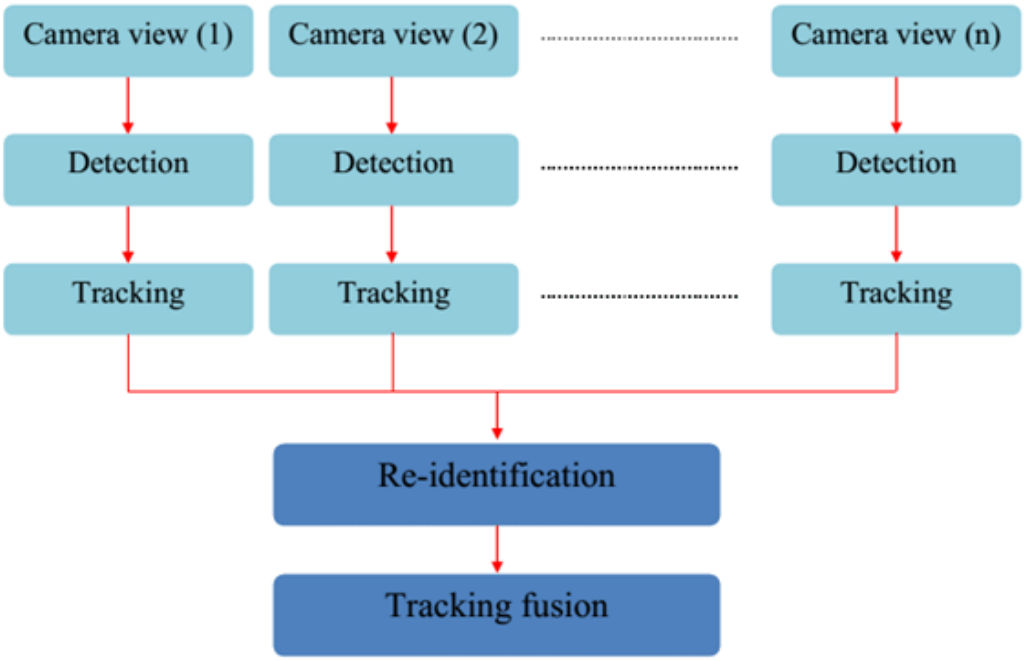
systém kde chceme zaviesť aj rozpoznávanie osôb, tak sa dáta potrebné na re-identifikáciu musia posielať aj medzi centrálnymi bodmi všetkých zhlukov.



Obr. 4.1: Rozdelenie prístupov k riešeniu detekcie a sledovania objektov v multi-kamerových systémoch [13].

4.1.1 Distribuované prístupy

Metódy založené na distribuovaných prístupoch sa nespoliehajú na veľký prenos dát medzi jednotlivými kamerami. Podľa článku [13] tento fakt tak znamená, že vo všeobecnosti sa tieto metódy skladajú z troch jednoduchých krokov: detekcia, tracking (sledovanie), re-identifikácia. Tieto kroky zaručia plnohodnotné sledovanie osôb v sledovanej oblasti. Metódy začínajú detekciou a sledovaním osôb v každej kamere zvlášť, tak ako keby sa jednalo o detekciu v jedno kamerovom systéme. Následne sa informácie o sledovaných osobách spoja do centrálneho bodu (napr. server), kde sa následne nad týmito dátami prevedie re-identifikácia osôb a po tomto kroku sa dá určiť pohyb osoby v celom sledovanom systéme. Tento postup je znázornený na obr. 4.2. Re-identifikácia osôb sa tak stáva najdôležitejšou otázkou pre tieto prístupy.



Obr. 4.2: Jednoduchá schéma znázorňujúca kroky pre metódy založené na distribuovaných prístupoch [13].

4.2 Re-identifikácia osôb

Re-identifikácia osôb je jeden zo základných problémov v otázke sledovania osôb v distribuovaných kamerových systémoch. Cieľom re-identifikácie je nájsť zhodujúce sa osoby naprieč zábermi z rôznych kamier, ktorých zorné polia sa neprekrývajú. Podľa článku [40] čelí re-identifikácia osôb dvom hlavným problémom. Prvým z týchto problémov je záber osoby z iných uhlov, kde jedna kamera zachytí osobu spredu a druhá zozadu. Druhým veľkým problémom je fakt, že pri veľkom množstve sledovaných osôb je vysoká šanca, že osoby majú podobné oblečenie a celkový vzhľad. Tento problém je ešte znásobený faktom, že osoby sa môžu nachádzať ďaleko od kamery a teda nie sú zachytené vo vysokom rozlíšení a miera detailov tak výrazne klesá. Tieto problémy sú ukázané na obr. 4.3.

Základom pre dobrú re-identifikáciu tak je zistiť, ktoré príznaky sú dôležité a ako ich správne extrahovať. Článok [40] uvádza nový prístup k riešeniu tohto problému. Podľa tohto výskumu sa dosiahne lepších výsledkov, ak sú príznaky osôb takzvane *omni-scale*, čo je definované ako kombinácia rôznych homogénnych a heterogénnych škál, kde každá sa skladá z kombinácie viacerých škál. Ako príklad týchto rôznych príznakov môže byť tričko, ktoré ma osoba oblečené. Homogénne škály môžu niesť informáciu o farbe daného trička, čo je veľmi dôležitá informácia na rozpoznanie osoby a výrazne pomôže znížiť počet potenciálnych cieľov. Problémom však je, ak má viac osôb tričko rovnakej farby, čo je v reálnych príkladoch bežné. Musia byť tak využité komplikovanejšie a na detaily bohatšie heterogénne škály, ktoré môžu niesť informáciu napríklad o tom, či sa na danom tričku nenachádza nejaké logo, čo by pomohlo identifikovať osobu zo zostávajúcich cieľov. Je tak jasné, že pre presnú identifikáciu sú príznaky z lokálnych častí (logo na tričku, farba topánok) rovnako dôležité ako globálne príznaky (pohlavie, typ postavy). Príklad účinnosti tohto riešenia môže byť ukázaný na obr. 4.3. Zoberme si obr. 4.3 (b) (vľavo) kde pomocou globálnych príznakov zistíme, že sa

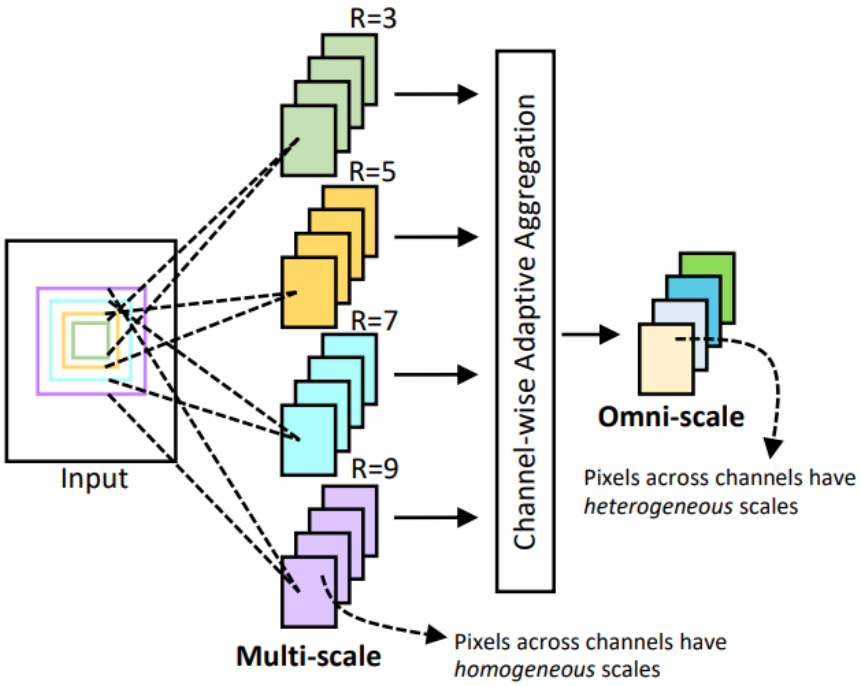
jedná o mladého muža s červeným tričkom a čiernymi šortkami. Tieto globálne príznaky nám teda nepomôžu rozlíšiť medzi skutočnou zhodou (stredný obrázok) a falošnou zhodou (obrázok vpravo). Avšak pomocou lokálnych príznakov ako farba a typ topánok, či farba ruksaku, vieme osobu rozoznať. Ako príklad využitia heterogénnych škál môže byť obr. 4.3 (d). V tomto prípade sa zjavne zhodujú informácie o veku, pohlaví osoby a aj o farbe oblečenia, avšak pomocou tohto prístupu nám heterogénne škály zaručia rozoznanie osoby na základe loga na tričku. Dôležitým faktom je to, že príznaky tohto typu, ako je napríklad logo na tričku, nie sú sami o sebe rozhodujúcim faktorom a závisia na celom kontexte, čo znamená, že ak máme dve osoby s rôznou farbou trička tak fakt, že na jednom z tých tričiek sa nachádza nejaké logo, nezohráva žiadnu rolu.

Obr. 4.3: Znázornenie problémov, ktoré môžu nastať pri re-identifikácii. Každá trojica obrázkov predstavuje zľava doprava: osobu ktorú chceme identifikovať, skutočnú zhodu, falošnú zhodu. Trojice (a) a (b) ukazujú problém rôznych uhlov, kde fakt, že osoba má na sebe ruksak, môže spôsobiť veľký problém pre jej identifikáciu. Trojice (c) a (d) ukazujú problém podobnosti osôb a problém malej úrovne detailov na snímke [40].

4.2.1 OSNet

Potenciál *omni-scale* príznakov je tak veľký, avšak väčšina súčasných re-identifikačných riešení, ktoré sú postavené na využití konvolučných neurónových sietí, využívajú siete, ktoré boli navrhnuté hlavne na rozpoznávanie objektov a nie na rozpoznávanie inštancií v triedach. Toto zahŕňa siete popísané v sekcii 3.3. Tieto dve úlohy sú od seba veľmi odlišné, a aj keď siete ako ResNet[11] alebo HRNet[32] dosahujú po správnom vytrénovaní slušné výsledky, ich celkový potenciál je o dosť menší, hlavne ak sa jedná o krajné prípady, ktoré sú spomenuté vyššie. Výskum [40] tak prichádza s novou sieťou pomenovanou OSNet, ktorá má CNN architektúru a je navrhnutá špeciálne na získavanie a reprezentáciu *omni-scale*

príznakov. Základný stavebný blok siete OSNet pozostáva z niekoľkých konvolučných prúdov s rôznymi veľkosťami škály. Schéma tohto bloku je zobrazená na obr. 4.4. Výsledné príznakové mapy s viacerými škálami sú následne dynamicky spájané pomocou váh generovaných agregačnou bránou (*aggregation gate*). Agregačná brána je malá sieť zdieľajúca parametre naprieč všetkými konvolučnými prúdmi s množstvom požadovaných vlastností pre efektívne trénovanie modelu. Samotná agregačná brána je tak trénovaná a generované váhy sa stanú závislé na vstupe, preto je potrebné dynamické spájanie rôznych škál. Práva táto agregačná brána je to, čo dovoľuje sieti reprezentovať príznaky ako *omni-scale*, čo znamená, že v závislosti na vstupnom obrázku sa brána môže zamerať na jednu zo škál a priradiť jej dominantné váhy alebo naopak si ich môže vybrať niekoľko a vytvoriť tak heterogénne škály.



Obr. 4.4: Schéma základného stavebného bloku siete OSNet. Znázornenie niekoľko multi-škálových konvolučných prúdov zo vstupu a ich spojonie pomocou agregačnej brány na vytvorenie *omni-scale* príznakových máp [40].

Ďalšou kľúčovou charakteristikou siete OSNet [40] je jej prenosnosť a veľkosť. Tento fakt prináša niekoľko výhod. Za prvé re-identifikačné datasety sú pomerne menšie oproti datasetom používaným na trénovanie napríklad detekčných sietí a teda menšia sieť s menším počtom parametrov má lepšiu šancu sa vyhnúť nechcenému pretrénovaniu (*overfitting*) siete. Za druhé pre už spomínané distribuované multi-kamerové systémy je najpraktickejším spôsobom prevádzať extrakciu príznakov na samotnej kamere, čo znamená, že sa na centrálny server, ktorý zodpovedá za re-identifikáciu, nemusia za týmto účelom posielať zábery z kamier ale iba samotné extrahované príznaky. Celkovo by teda sieť mala dosahovať lepší výkon bez toho, aby sa stratila jej presnosť, čo je ukázané tým, že dosahuje state-of-the-art výsledky na šiestich najznámejších re-identifikačných datasetoch.

Detail architektúry siete OSNet [40] je popísaný v tabuľke 4.1. V zmysle prístupu k viacerým konvolučným prúdom je sieť podobná k sieti ResNeXt [35], ktorá je popísaná v sekcii 3.3.1. Základným blokom architektúry je reziduálny bottleneck pozostávajúci z upravených 3×3 vrstiev. Podrobnejšia schéma bloku je na obr. Ak máme vstup x, cieľom bottleneck-u je naučiť sa zvyšok (*residual*) $\widetilde{x}$ s funkciou mapovania F, čo je vyjadrené ako:

$$\widetilde{x} = F(x) \tag{4.1}$$

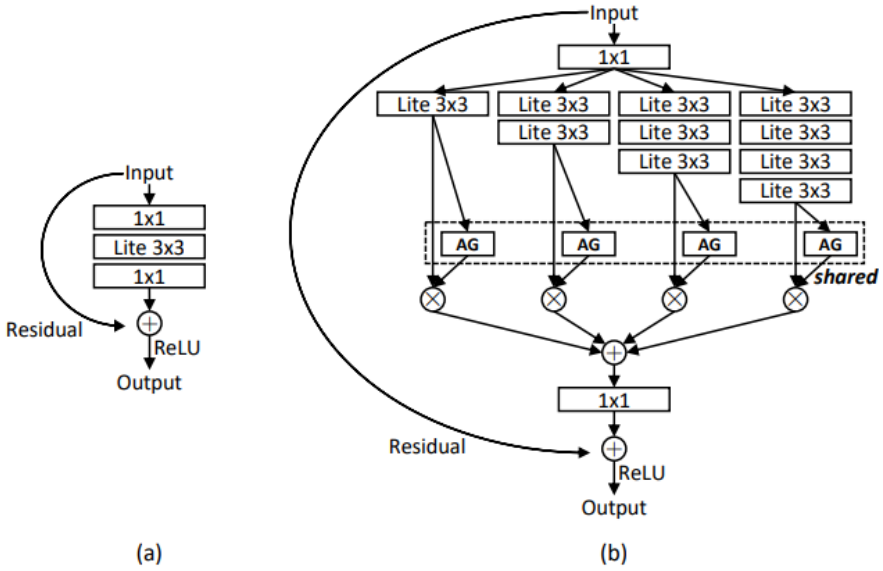
kde F predstavuje 3×3 vrstvu, ktorá sa učí jedno-škálové príznaky. Pre dosiahnutie učenia viac-škálových príznakov je potrebné rozšíriť zvyškovú funkciu F o exponent t, ktorý predstavuje škálu príznaku. Majme tak F^t, kde $t > 1$, v praxi to znamená, že sa v sieti za seba uloží t 3×3 vrstiev, a z toho vyplýva veľkosť škály $(2t+1) \times (2t+1)$. Zvyšok $\widetilde{x}$, ktorý sa treba naučiť, je tak suma prírastkových škál až po T vyjadrené ako:

$$\widetilde{x} = \sum_{t=1}^{T} F^t(x), \text{ kde } T \geq 1 \tag{4.2}$$

Ak je $T = 1$, tak sa rovnica 4.2 redukuje na rovnicu 4.1. V článku [40] je sieť OSNet navrhnutá s $T = 4$, čo znamená, že najväčšia škála ma veľkosť 9×9. Využitím skratky sieť zachováva príznaky menších škál naučených v predchádzajúcich vrstvách, čo umožňuje výsledným príznakom zachytiť celý rozsah škál. Takýmto spôsobom tak získame príznaky špecifických škál, ktoré sú teda homogénne. Na získanie *omni-scale* príznakov je navrhnutá agregačná brána, ktorá kombinuje výstupy prúdov dynamicky v závislosti na vstupnom obrázku. Agregačná brána je malá neurónová sieť. Nech x^t označuje $F^t(x)$, *omni-scale* zvyšok $\widetilde{x}$ získame ako:

$$\widetilde{x} = \sum_{t=1}^{T} G(x^t) \odot x^t \tag{4.3}$$

kde $G(x^t)$ je vektor s dĺžkou celej kanálovej dimenzie x^t a $\odot$ značí Hadamardov súčin. G je implementované ako malá sieť skladajúca sa s bez parametrickej globálnej združovacej vrstvy a s viac vrstvového perceptronu s jednou ReLU aktivovanou vrstvou nasledovanou sigmoid aktiváciou.



Obr. 4.5: Podrobná schéma reziduálneho bottleneck-u používaného v sieti OSNet. Vľavo (a) základný bottleneck blok vpravo (b) navrhovaný bottleneck s využitím agregačnej brány (AG). Prvá a posledná 1×1 vrstva je využitá na redukciu respektíve obnovenie príznakových dimenzií [40].

stage	output	OSNet
conv1	128×64, 64 64×32, 64	7×7 conv, stride 2 3×3 max pool, stride 2
conv2	64×32, 256	bottleneck × 2
transition	64×32, 256 32×16, 256	1×1 conv 2×2 average pool, stride 2
conv3	32×16, 384	bottleneck × 2
transition	32×16, 384 16×8, 384	1×1 conv 2×2 average pool, stride 2
conv4	16×8, 512	bottleneck × 2
conv5	16×8, 512	1×1 conv
gap	1×1, 512	global average pool
fc	1×1, 512	fc
# params		2.2M
Mult-Adds		978.9M

Tabuľka 4.1: Architektúra siete OSNet pre vstupný obrázok o veľkosti 256×128 pixelov [40].

Sieť má tri-krát menej parametrov a tri-krát menej operácii násobenia a sčítania oproti bežným sietiam tejto veľkosti. Toto bolo dosiahnuté využitím upraveného odľahčeného dizajnu 3×3 konvolučných vrstiev. Architektúra sa dá jednoducho zväčšiť alebo zmenšiť

pomocou multiplikátora šírky, na dosiahnutie dobrého balansu medzi výkonom a presnosťou podľa potreby užívateľa. Pre identifikáciu osôb sa vypočíta euklidovská vzdialenosť príznakových vektorov, ktoré sú výstupom poslednej plne prepojenej vrstvy ("fc" v tabuľke 4.1). Výstupné príznakové vektory sú 512-D.

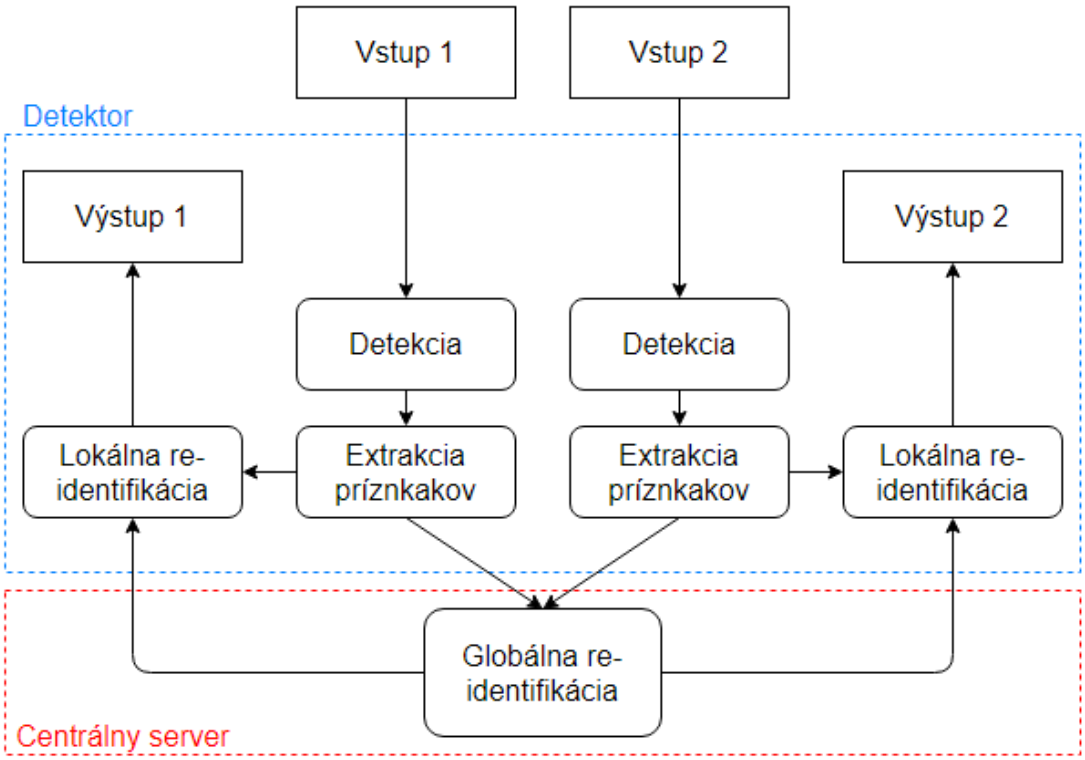
Kapitola 5

Návrh a implementácia

5.1 Návrh

Výsledná aplikácia bola navrhnutá na základe výskumu z predchádzajúcich kapitol. Na riešenie problému multi-kamerového systému bol zvolený distribuovaný prístup, ktorý je popísaný v sekcii 4.1. Aplikácia neočakáva spracovávanie vstupu so systému s prekrývajúcimi sa zábermi. Toto znamená rozdelenie funkčných častí na lokálny detektor a centrálny server. Úlohou detektoru je nájsť v zábere osoby a previesť extrakciu príznakov. Príznaky sú následne odoslané na server, kde sú spracované a porovnávajú sa s príznakmi zo zvyšných záberov. Výsledky porovnávania sú posielané späť na každý detektor, ktorý ich následne zobrazí na výstup. Toto je v podstate základná funkcionalita aplikácie. Schéma základného návrhu sa nachádza na obr. 5.1. V jednej časti sa však návrh od klasických distribuovaných systémov líši a to v časti spracovávania vstupu. Pre distribuované systémy je bežné, že detekcia a extrakcia príznakov prebieha na každej kamere zvlášť. Za účelom jednoduchšieho a praktickejšieho využitia tento návrh berie do úvahy spúšťanie detektoru na jednom stroji s viacerými vstupmi. Toto tak znamená ľahšie pracovanie s aplikáciou, kde užívateľovi stačia uložené zábery z kamier, ktoré chce spracovať. Návrh však taktiež umožňuje využitie klasického prístupu k použitiu detektoru, avšak táto časť nebola podnetom práce. S týmto prístupom však nastáva jeden problém, ktorý sa týka výstupu aplikácie. Návrh tak počíta s dvoma spôsobmi zobrazovania výsledkov: zobrazenie v okne počas priebehu detekcie, kde sa vytvorí jedno okno pre každý spracovávaný vstup alebo uloženie výsledku do videa, kde každé video korešponduje k jednému vstupu. Logika vytvárania a zobrazovania výsledkov je tak zabudovaná v samotnom detektore.

Aplikácia podľa návrhu zobrazenom na obr. 5.1 pracuje následovne. Nad každým vstupným záberom sa spraví detekcia osôb. Pre každú detekovanú osobu sa spraví extrakcia príznakov. Extrahované príznakové vektory sa uložia na lokálnom detektore a ďalej sa posielajú na spracovanie centrálnemu serveru. Server príznakové vektory spracuje, porovná s ostatnými, ktoré už má uložené a výsledky porovnávania posiela spätne detektoru, ktorý si podľa nich synchronizuje lokálne dáta. Podľa týchto dát je v každom detektore vytvorený výstup. Príznakové vektory sa na server posielajú v dvoch prípadoch: ak bola detekovaná nová osoba v lokálnom detektore a v intervale x snímkov, kedy sa odošlú lokálne dáta zo všetkých detektorov. Toto zaručuje synchronizáciu dát v oboch smeroch, čo zaručuje využitie skutočnosti, že aplikácia pracuje s videom a vie sa tak lepšie prispôsobiť aj meniacim sa podmienkam v záberoch. Medzi intervalom synchronizácie dát tak každý detektor pracuje iba lokálne, využitím podobných postupov aké sú použité pre globálnu re-identifikáciu.



Obr. 5.1: Schéma návrhu aplikácie. Modrá časť zobrazuje funkcionality detektoru, červená časť funkcionalitu serveru. Schéma zobrazuje prácu s dvoma vstupmi avšak obdobne by sa k problému pristupovalo aj pre viac vstupov.

K aplikácii bolo navrhnuté aj jednoduché užívateľské rozhranie, ktoré slúži na jednoduchú prácu s aplikáciou. Súčasťou rozhrania musí byť spôsob zadávania súborov, ktoré obsahujú zábery z kamier, a keďže aplikácia ako bolo spomenuté v tejto sekcii má viac možností výstupov tak aj nejaký spôsob ako vybrať formát výstupu.

5.2 Implementácia

Implementácia aplikácie je rozdelená do troch hlavných častí: detektor, server, užívateľské rozhranie.

5.2.1 Detektor

Detektor prijíma vstupy pomocou argumentov z príkazového riadku. Na spracovanie argumentov je využitá vstavaná Python knižnica *argparse*. Argument *input* (-i) môže prijať viac ako jednu hodnotu, týmto spôsobom je tak možné predať detektoru viac vstupných videí. Argumenty *display* a *save* sa týkajú výstupu detektoru, podľa toho či chce užívateľ výsledok zobraziť respektíve uložiť do video-súboru. Posledný argument *port* súvisí s komunikáciou so serverom a jeho východzia hodnota je 10000, čo je východzia hodnota portu, na ktorom server bude očakávať príznakové vektory, ktoré sú výsledkom detekcie a extrakcie príznakov.

Na spracovanie vstupov a vytvorenie výsledného výstupu je použitá knižnica *OpenCV* [18]. Táto knižnica bola použitá na otvorenie vstupných video prúdov do formy, aby sa dalo pristupovať k informáciám o danom videu a taktiež k jednotlivým snímkov videa pomocou fukncie *VideoCapture*. Funkcia môže buď otvoriť video zadané cestou k súboru, ale môžu ňou byť otvorené aj aktívne nahrávacie zariadenia, ktoré sú rozpoznané systémom ako napríklad web-kamera. Ak užívateľ zadal, že chce výstup detektoru uložiť do súboru, tak sa pomocou funkcie *VideoWriter* vytvoria prúdy s rovnakými parametrami aké má vstupné video a môžu sa tak postupne vkladať výsledné snímky priamo do súboru. Pokiaľ užívateľ chce výsledok len zobraziť, tak sa pre každý vstup vytvorí jedno okno, v ktorom sa bude zobrazovať daný výstup. Posledná funkcia, na ktorú je táto knižnica využitá, je vykreslenie dát ,ako sú ohraničujúce boxy a identifikačné čísla osôb.

Na vytvorenie detekcie a ukladanie príznakových vektorov je využitý balíček *PyTorch* [22]. Balíček uľahčuje prácu s neurónovými sieťami. *PyTorch* ponúka dve verzie, jedna ktorá využíva na výpočty GPU pomocou CUDA a druhá využíva CPU. Aplikácia bola navrhnutá tak, aby bolo možno použiť obe verzie pre vytvorenie detekcií. Toto bolo dosiahnuté prístupom, kde aj pri viacerých vstupoch sa postupuje snímok za snímkom a nevyužíva sa tak viac vlákien, čo by pri využití GPU nebolo možné. Na detekciu je využitý model z model zoo tohto balíčka. Jedná sa o sieť *Faster R-CNN* [23], ktorá je viac popísaná v sekcii 3.2.1 a model má backbone so sieťou *MobileNetV3* [12], ktorý bol vybraný vďaka poskytnutiu dobrého balansu medzi presnosťou a rýchlosťou siete hlavne kvôli poskytnutiu podpory pre CPU detekciu, kde model s využitím *ResNet* dosahuje slabý výkon. Použitá sieť má deväťnásť miliónov parametrov a jej hodnotenie box AP (*Average precision*), čo je metrika často používaná na porovnanie presnosti modelov, je rovná 32,8. Prah prípustných detekcií bol stanovený na 98% a to z dôvodu potreby kvalitných detekcii pre lepší výsledok rozpoznávania. Balíček taktiež obsahuje spôsob na transformáciu príznakových vektorov na formát, ktorý je možné odoslať na server pomocou TCP protokolu.

Na lokálnu re-identifikáciu v rámci jedného video-prúdu (kamery) je použitá knižnica *Torchreid* [39], ktorá je bližšie popísaná v sekcii 5.2.1. Na extrakciu príznakov detekovaných osôb je použitá sieť *OSNet* [40]. Re-identifikácia je robená výpočtom euklidovskej vzdialenosti príznakových vektorov. Za týmto účelom bolo potrebné stanoviť hraničnú hodnotu, ktorá bude slúžiť k rozpoznaniu osôb. Experimenty na MARS [38] (*Motion Analysis and Re-identification Set*) datasete, ktorý slúži na trénovanie re-identifikačných sietí, sú uvedené v sekcii 6.1.1. Z experimentov vyšlo, že priemerná hodnota euklidovskej vzdialenosti príznakových vektorov pre zábery z jednej kamery, je rovná 300,252 a hraničná hodnota tak bola pre výslednú aplikáciu nastavená na 300. Pre globálnu re-identifikáciu bola táto hodnota rovná 489,121 a hraničná hodnota tak pre globálnu re-identifikáciu bola nastavená na 500. Na výpočet euklidovskej vzdialenosti je využitá funkcia *compute_distance_matrix* z knižnice *Torchreid*. Táto funkcia poskytuje možnosť predania 2 polí obsahujúcich príznakové vektory. Výstupom funkcie je dištančná matica obsahujúca porovnania medzi každou dvojicou zo zadaných vstupných polí. Toto tak umožňuje jednoducho a efektívne vypočítať euklidovské vzdialenosti aj pre veľké množstvo detekovaných ľudí. S maticou sa dá jednoducho pracovať pomocou knižnice *NumPy*, čo je bežná knižnica pre prácu s dátami v jazyku Python, nie len pre účeli umelej inteligencie. Knižnica poskytuje jednoduchý spôsob vyhľadania hodnôt v matici podľa zadanej podmienky ako aj nájdenie minima pre daný riadok alebo stĺpec.

Postup práce implementovaného detektoru je tak následovný. Pre každý vstupný video súbor zadaný pomocou argumentov sa vytvorí vstupný video prúd, z ktorého sa berú postupne snímky. Pre každý snímok sa vytvorí detekcia osôb, kde pre každú detekovanú

osobu sa spraví výrez s hranicami ohraničujúceho boxu pomocou knižnice *OpenCV*. Výrezy sú uložené a následne odoslané na vstup sieti *OSNet*, ktorej výstupom sú príznakové vektory prislúchajúce zadaným výrezom. Ak pre daný lokálny detektor ešte neboli uložené žiadne dáta, všetky zachytené príznakové vektory sa odosielajú na server, kde je ku každému priradené identifikačné číslo. Ak má detektor uložené lokálne dáta, porovnáva získané príznakové vektory iba lokálne. Osoby, ktoré nie je možné priradiť k žiadnemu lokálnemu príznaku, sú v intervale piatich snímkov posielané na server, aby im bolo priradené ID, ktoré sa následne uloží do lokálneho slovníka. Nové osoby sú posielané v intervaloch piatich snímkov za účelom zníženia prenosu dát, ale taktiež z 2 dôvodov týkajúcich sa rozpoznávania. Rozpoznávanie môže mať problémy, ak nie je v zábere celá osoba ako napr. prekrývanie osoby. Ďalším problémom je, že rozpoznávanie nie je perfektné, a preto aj keď sa jedná o tú istú osobu, ktorú máme uloženú v pamäti, nedá sa očakávať, že bude rozpoznaná na každej zadanej snímke. Čakanie na interval piatich snímkov oba tieto problémy zmierňuje. Komunikácia so serverom prebieha pomocou TCP protokolu za využitia vstavanej knižnice *socket*. Každých päť snímkov dochádza aj k aktualizácii dát na servery, kedy každý detektor odošle na server všetky príznaky, ktoré drží v pamäti tak aby si mohol server aktualizovať príznaky, ktoré používa na získavanie identifikačných čísiel. Na spracovávanej snímke sú tak podľa všetkých získaných informácii vykreslené ohraničujúce boxy a identifikačné čísla ku každej osobe pomocou knižnice *OpenCV*. Výsledný spracovaný snímok je na konci zobrazený do okna alebo uložený do výstupu alebo oboje podľa toho aký formát výstupu si užívateľ zvolil.

Torchreid

Torchreid [39] je softvérová knižnica postavená na PyTorch, ktorá poskytuje jednoduché rozhranie pre prácu, vývoj, trénovanie a vyhodnocovanie re-identifikačných sietí. Knižnica bola vyvinutá za účelom podpory re-identifikačných výskumov a projektov. Knižnica je napísaná v jazyku Python, kde nejaké časti kódu sú založené na Cython-e z dôvodu lepšej optimalizácie a výkonu. Knižnica taktiež poskytuje model zoo s viacerými state-of-the-art pred-trénovanými CNN modelmi. Knižnica sa skladá zo šiestich modulov, z čoho hlavné sú tri: *Data*, *Engine* a *Models*. Modul *Data* poskytuje nástroje na uľahčenie prípravy dát z datasetov určených na trénovanie modelov a to pre datasety založené na obrázkoch a aj videách. Modul *Engine* poskytuje zefektívnenú pipeline na trénovanie a vyhodnocovanie modelov. Obsahuje naimplmentované 2 paradigmy učenia pre CNN modely a to klasifikácia pomocou *softmax loss* a učenie podobnosti (*metric learning*) pomocou *triplet loss*. Tieto dve paradigmy sú využívané vo väčšine súčasných state-of-the-art re-identifikačných riešeniach. Modul taktiež obsahuje sadu nástrojov na vizualizáciu výsledkov trénovania. Príkladmi sú funkcia *visrank*, ktorá pre vybraný dataset a otázny obrázok zobrazí k najviac pravdepodobných výsledkov, alebo funkcia *visactmap*, ktorá zobrazuje výsledné aktivačné mapy. Posledný z hlavných modulov je *Models*. Tento model ponúka sadu implementácií CNN architektúr a to nie len pre modely, ktoré boli navrhnuté pre re-identifikačné účely. Medzi príklady patria ResNet[11], ResNeXt[35], MobileNetV2[25] a hlavne špecificky dizajnovaný OSNet[40]. K týmto modelom sú taktiež poskytované váhy a autori projektu sa snažia udržať zoznamy modelov aktuálny voči novým výskumom v tejto problematike.

5.2.2 Server

Server bol implementovaný za účelom ukážky komunikácie pre distribuované multi-kamerové systémy. Server prijíma jeden nepovinný argument *port* s východzou hodnotou 10000. Ser-

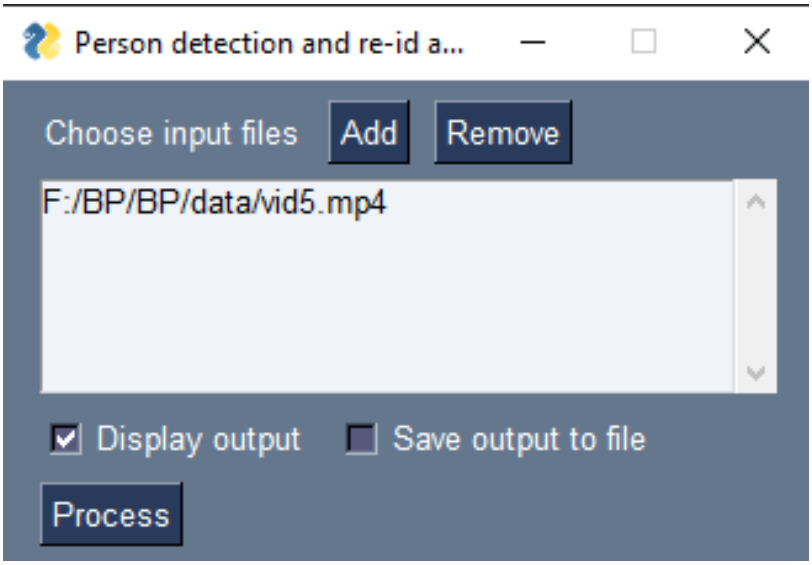
ver na komunikáciu podobne ako detektor využíva vstavanú knižnicu *socket*. Pomocou tejto knižnice sa tak vytvorí jednoduchý server, ktorý očakáva jedno spojenie na zadanom porte a správy odoslané pomocou TCP protokolu. Pre možnú dĺžku a komplexnosť prijímaných dát obsahuje server pomocnú funkciu *recvall*, ktorá načíta časti správy po 4kiB častiach a spája ich do jedného dátového bloku. Pre odoslanie a prijímanie dát je použitá aj vstavaná knižnica *pickle* na zmenu formátu dát.

Server očakáva štyri možné formáty prijatých dát. Každý z týchto formátov je typu slovník, kde kľúče tohto slovníka určujú akú akciu detektor od serveru očakáva. Prvý formát obsahuje kľúč „*end*", čo značí, že detektor ukončil detekciu a je to signál pre to, aby sa ukončil aj server. Ďalší formát obsahujúci kľúč „*get_ids*" určuje, že detektor potrebuje rozpoznať nové osoby, keďže jedine server môže priradiť osobám nové ID. Toto je hlavná rozpoznávacia funkcionalita serveru, kde získavanie dištančnej matice a jej využitie na porovnávanie príznakov je dosť podobné tomu ako je to na detektore. Výsledky rozpoznávania sú uložené do pola polí obsahujúcich priradené ID a daný príznakový vektor a sú odoslané ako odpoveď naspäť na detektor. Ďalšou akciou, ktorú server vykonáva, je aktualizácia serverových hodnôt určená kľúčom „*update*". Server príjme celé lokálne slovníky z detektoru a aktualizuje si podľa nich svoje uložené hodnoty. Posledným typom správy je správa s kľúčom „*taken_ids*", ktorá informuje server o tom, ktoré ID už boli priradené, tak aby s nimi server nemusel pracovať, keďže aplikácia pracuje nad predpokladom, že zábery kamier sa neprekrývajú a teda každé ID môže byť zobrazené len raz.

5.2.3 Užívateľské rozhranie

Užívateľské rozhranie (UI) bolo implementované pomocou balíčka *PySimpleGUI* [21]. Tento balíček umožňuje jednoduchý prístup k tvoreniu UI, či už po stránke vzhľadu ale aj funkcionality. Obsahuje veľké množstvo objektov, ktoré môže užívateľ využiť a k väčšine problémov poskytuje jednoduché Python riešenia. Na vytvorenie kostry UI sa využíva takzvaný *layout*, ktorý je reprezentovaný polom polí objektov tohto balíčka (napr. tlačítka, textové polia atď.). Toto pole je potom pre-transformované do výsledného okna, ktoré sa zobrazí užívateľovi. Ďalšiu dôležitú časť, ktorú balíček obsahuje, je implementácia udalostí. Medzi tieto patrí napr. stisk tlačítka. Udalosti sa dajú jednoducho z okna vyčítať pomocou funkcie *read* nad objektom *window*, ktorý predstavuje zobrazované okno.

Výsledné UI je zobrazené na obr. 5.2. Užívateľské rozhranie dovoľuje užívateľovi pridávať a odoberať vstupné video-súbory. Aktuálny list vybraných súborov je vždy zobrazený v strede UI. V spodnej časti UI si užívateľ môže zvoliť formu výstupu, akú chce dostávať od detektoru. A stlačením tlačítka *Process* sú zavolané skripty serveru a detektoru pomocou vstavanej knižnice *subprocess*. Detektor je zavolaný s príslušnými hodnotami podľa toho, čo si užívateľ zvolil v aplikácii. Súčasťou UI je taktiež okno zobrazujúce postup práce, ktorý je získaný ako počet spracovaných snímkov z celkového počtu snímkov. Z tohto okna sa dá spracovanie videí zrušiť. Ak užívateľ zvolil, že chce výstup uložiť do súboru a spracovávanie preruší v jeho priebehu, videá budú uložené do bodu do ktorého boli spracované.



Obr. 5.2: Implementované užívateľské rozhranie pomocou knižnice *PySimpleGUI*.

Kapitola 6

Experimenty

V tejto kapitole sú popísané niektoré z experimentov, ktoré slúžili k otestovaniu správnosti aplikácie. Dataset na experimentovanie bol vytvorený pre túto prácu a je k práci priložený. Dataset vychádza z osemnástich videí v kvalite Full HD (1080p) s počtom snímkov za sekundu (fps) rovnému 60. Táto kvalita záznamu je najmenšia odporúčaná kvalita na dosiahnutie kvalitných výsledkov. Aplikácia samozrejme umožňuje použiť na vstupe aj videá s nižšou kvalitou, ale môže to do značnej miery ovplyvniť kvalitu detekcie a aj rozpoznávania osôb. Zábery obsahujú najviac tri osoby a dataset simuluje prípady použitia pre kamerové systémy s jednou, dvomi a tromi kamerami. Dataset taktiež obsahuje príklad problematických scenárov pre detekciu ako napr. prekrývanie osôb a pre rozpoznávanie napr. zmena osvetlenia. Celkovo dataset obsahuje 11 súborov na experimentovanie.

6.1 Experimenty na sieti OSNet

Sekcia uvádza niektoré z experimentov, ktoré slúžili, na nastavenie parametrov re-identifikácie pre výslednú aplikáciu. Experimenty boli zamerané, na zistenie možností praktického využitia modelu siete OSNet [40], ktorý je bližšie popísaný v sekcii 4.2.1. Na tieto experimenty bol taktiež využitý dataset MARS [38], ktorý sa radí medzi popredné re-identifikačné datasety.

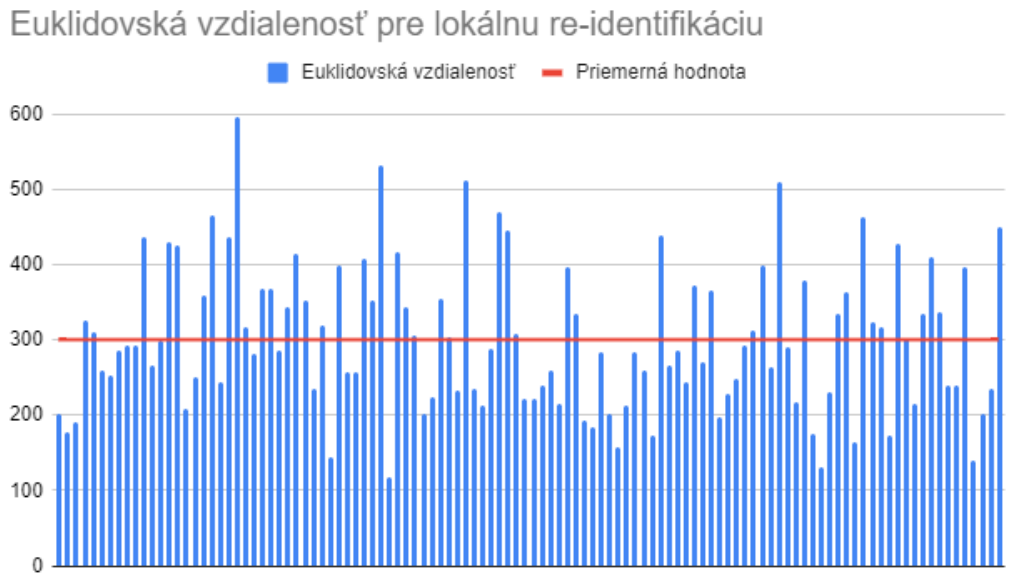
6.1.1 Experiment 1 - hranica euklidovskej vzdialenosti

Tento experiment slúžil na zistenie hraničných hodnôt euklidovskej vzdialenosti medzi príznakovými vektormi pre re-identifikáciu. Experiment bol rozdelený do dvoch častí, jedna mala za úlohu zistiť hraničnú hodnotu pre lokálnu re-identifikáciu a druhá pre tú globálnu. Na experimentovanie bol využitý dataset MARS [38] z toho dôvodu, že poskytuje veľké množstvo rôznorodých dát. Trénovací dataset poskytuje cez 600 osôb, kde ku každej sú v priemere stovky záberov z rôznych kamier. Dataset v názve každého záberu uvádza identifikačné číslo kamery, na ktorej bol daný záber zachytený. Táto skutočnosť bola využitá pre určenie hraničnej hodnoty pre lokálnu re-identifikáciu, kde má zmysel brať iba hodnoty zo snímkou z tej istej kamery. Experiment bol čiastočne obmedzený hardwarom, na ktorom boli experimenty prevádzané, z dôvodu, že dochádzalo k preplneniu pamäte GPU pre priečinky s veľkým množstvom záberov. Boli tak na experiment využité len priečinky obsahujúce menej ako 300 záberov, avšak tento fakt by nemal mať na dosiahnuté výsledky veľký vplyv, keďže experiment zahrňoval 116 priečinkov s viac ako 20000 zábermi.

Experimenty prebiehali tak, že sa z každého priečinka zobral úvodný záber, získal sa jeho príznakový vektor, a ten bol následne porovnaný so získanými príznakovými vektormi všet-

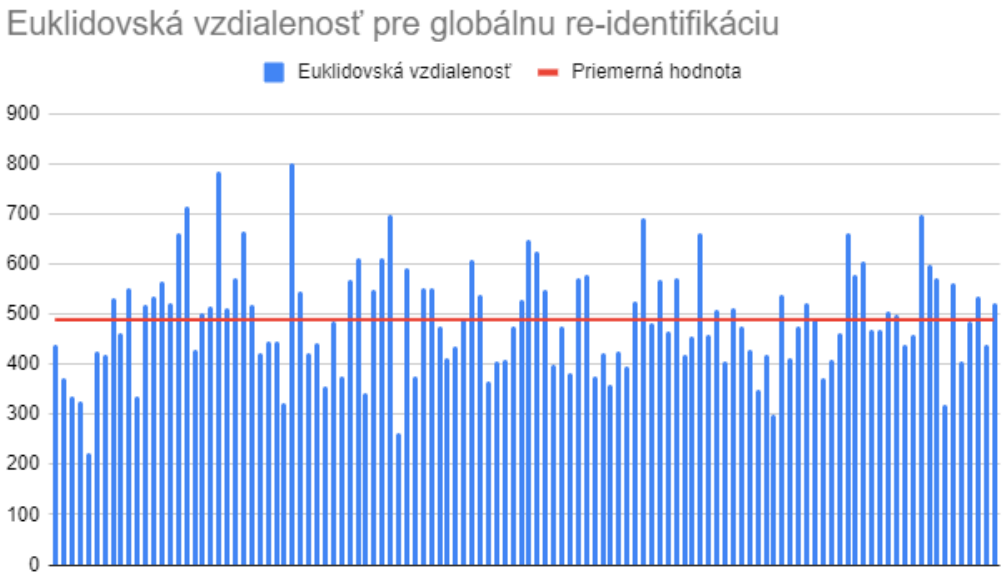
kých ostatných záberov v danom priečinku. Pre experiment pre lokálnu hraničnú hodnotu boli príznaky porovnávané len medzi zábermi z tej istej kamery, na ktorej bol zachytený prvý záber priečinku.

Graf výsledkov experimentu pre určenie hraničnej hodnoty pre lokálnu re-identifikáciu je zobrazený na obr. 6.1. Priemerná hodnota euklidovskej vzdialenosti medzi príznakovým vektorom prvého záberu kamery v porovnaní zo všetkými ostatnými zábermi z tej istej kamery bola rovná 300,252. Maximálna priemerná hodnota pre jeden celý priečinok bola rovná 596,283 a minimálna priemerná hodnota bola 118,11. Pre tento experiment bol počet vstupných priečinkov rovný 112.



Obr. 6.1: Na grafe sú zobrazené výsledky experimentu zamerané na určenie hraničnej hodnoty euklidovskej vzdialenosti pre lokálnu re-identifikáciu. Každý jednotlivý stĺpec predstavuje priemernú hodnotu jedného priečinku, na ktorom bol experiment uskutočnený. Experimenty pre lokálnu re-identifikáciu boli prevádzané vždy len na záberoch z tej istej kamery. Červená čiara predstavuje priemernú hodnotu všetkých získaných hodnôt.

Graf pre hodnoty experimentu globálnej re-identifikácie sa nachádza na obrázku. 6.2. Priemerná hodnota v tomto prípade bola rovná 489,121. Maximálna priemerná hodnota pre jeden priečinok bola 801,258 a minimálna 223,322. Experiment zahrňoval 116 priečinkov. V tomto experimente boli do úvahy brané všetky zábery v danom priečinku.



Obr. 6.2: Na grafe sú zobrazené výsledky experimentu zamerané na určenie hraničnej hodnoty euklidovskej vzdialenosti pre globálnu re-identifikáciu. Každý jednotlivý stĺpec predstavuje priemernú hodnotu jedného priečinku, na ktorom bol experiment uskutočnený. Experimenty pre globálnu re-identifikáciu boli robené nad všetkými zábermi v jednotlivých priečinkoch. Červená čiara predstavuje priemernú hodnotu všetkých získaných hodnôt.

Výsledky týchto dvoch experimentov boli použité na určenie hraničných hodnôt pre výslednú aplikáciu, kde hodnota pre lokálnu re-identifikáciu bola stanovená na 300 a pre globálnu na 500.

6.2 Experimenty na výslednej aplikácii

6.2.1 Experiment 1 - jedna osoba na jednom zázname

Tento experiment bol zameraný na zistenie toho, ako si vie aplikácia poradiť s detekciou a rozpoznaním jednej osoby v jednom zázname. Táto funkcionalita je základným stavebným blokom pre rozpoznávanie naprieč rôznymi záznamami. Predpokladom pre tento experiment je jeden záber, v ktorom sa jedna osoba voľne pohybuje. Osoba môže zo záberu odísť a vrátiť sa naspäť. Skúmané boli skutočnosti, či dochádza ku chybám detekcie (neprimerané ohraničujúce boxy, nedetekovanie osoby na zázname) a či je systém schopný rozoznať osobu s viacerých profilov.

Na obr. 6.3 sú zobrazené tri kľúčové rozpoznania osoby z rôznych profilov. Na obr. 6.4 sú zobrazené niektoré snímky z experimentu, ktoré znázorňujú funkcionalitu lokálneho rozpoznávania. Samotné detekcie v experimente dosiahli pomerne dobré výsledky. Jediným nedostatkom je malá nepresnosť ohraničujúcich boxov, kde sa môže stať, že malá časť osoby vyčnieva mimo tento box ako je možné vidieť aj na obr. 6.3 a 6.4.

Obr. 6.3: Tri rozoznania tej istej osoby z rôznych profilov vstupujúcej do záberu kamery. V každom príklade bolo priradené ID = 0.

Obr. 6.4: Ukážka lokálneho rozpoznávania v jednom zábere kamery.

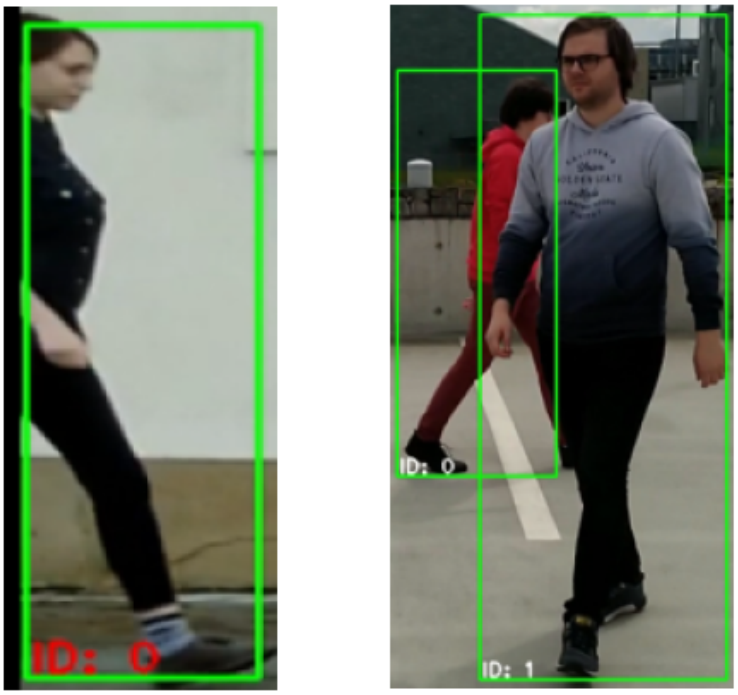
Experiment avšak odhalil aj problém s re-identifikáciou. Na prvom testovanom videu re-identifikácia prešla v poriadku, avšak na druhom videu sa nepodarilo v jednom prípade rozoznať osobu vstupujúcu späť do záberu kamery a bolo jej tak chybne priradené nové ID, ako je možné vidieť na obr. 6.5. Euklidovská vzdialenosť príznakov pri re-identifikácii bola rovná 555,9, čo prekračuje limit stanovený pre minimálnu identifikáciu na servery.

Obr. 6.5: Chybná re-identifikácia pri vstupe do záberu. Obrázok vľavo má priradené ID = 0 a obrázok vpravo nebol re-identifikovaný a teda sa mu priradilo ID = 1.

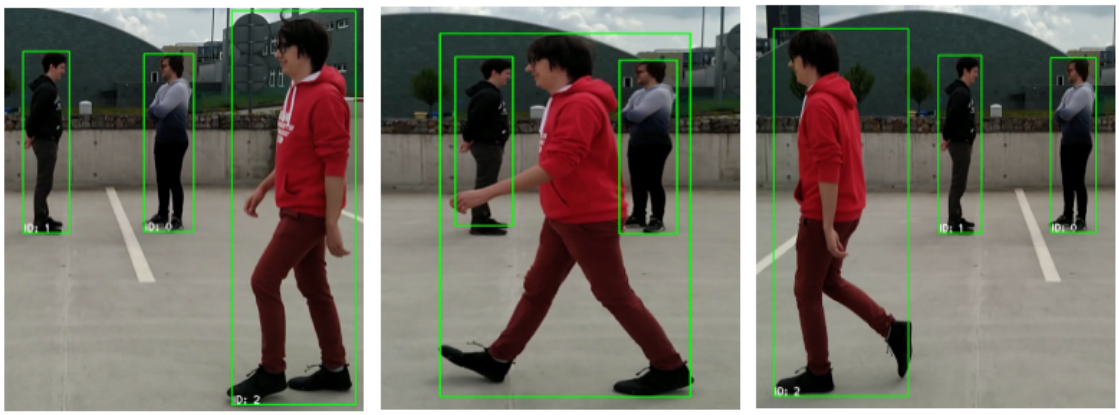
6.2.2 Experiment 2 - prekrytie osoby

Tento experiment skúmal vplyv prekrytia osoby na detekciu a re-identifikáciu. Pôvodný návrh aplikácie nezahrňoval funkcie, ktoré by pomohli s problémom neúplnej detekcie osoby, čo výrazne zvyšuje nepresnosť rozpoznávania. Už s prvých experimentov boli zjavné dva bežné scenáre, kedy by k problému s rozpoznávaním dochádzalo. Jedným z týchto scenárov je vstup osoby do záberu kamery, kde detektor osobu zachytí skôr ako prejde do záberu celá. Druhým je prekrytie osoby inou osobou, čo môže spôsobiť ešte väčší problém, kedy sa príznaky jednej osoby zanesú do príznakového vektoru druhej osoby. Problematické scenáre sú zobrazené na obr. 6.7.

Experimentovaním k návrhu aplikácie pribudli dve podmienky, ktoré musia byť splnené, aby dochádzalo k rozpoznávaniu. Jednou z podmienok je, že osoba sa musí nachádzať aspoň 20 pixelov od hrany záznamu a druhou podmienkou je, že ohraničujúci box rozpoznávanej osoby sa nesmie prekrývať s iným ohraničujúcim boxom. Prvá podmienka vedie k tomu, že osoba bude rozpoznaná až chvíľu po tom, čo do záberu skutočne vstúpila a druhá podmienka k tomu, že ak sa počas spracovávania snímky dve osoby prekryjú, nebudú rozpoznávané ani lokálne, takže dochádza k malému výpadku v rozpoznávaní za cenu lepšej aj keď nie perfektnej stability systému. Výsledky zmien pre prvú podmienku je možné vidieť v experimente 6.2.1 na obr. 6.3. Príklad podmienky pre prekrývanie je zobrazený na obr. 6.7.



Obr. 6.6: Vľavo: osoba vstupujúca do záznamu. Polovica tela je mimo záznamu (čierny rám na kraji predstavuje hranu záznamu), čo znemožňuje kvalitnú re-identifikáciu. Vpravo: jedna z detekovaných osôb je schovaná za druhou, čo môže spôsobiť chybné rozpoznanie a v horšom prípade aj narušenie uložených príznakov.



Obr. 6.7: Vľavo: systém pred prekrytím osôb, V strede: prekrytie osôb a vynechanie rozpoznávania, Vpravo: systém po prekrytí a obnove rozpoznávania.

6.2.3 Experiment 3 - jedna osoba na dvoch záznamoch

Tento experiment sa zameral na re-identifikáciu jednej osoby naprieč viacerými záznamami. S pohľadu samotnej re-identifikácie je tento experiment podobný prípadu z experimentu 6.2.1, kedy osoba vystúpila a opäť vstúpila do záznamu. Experiment tak skôr testoval správnosť komunikácie viacerých prúdov s centrálnym serverom.

Experiment bol úspešný, čo je zobrazené na obr. 6.8. Potvrdila sa tak správnosť komunikácie so serverom a to, že funkcionality rozpoznávania a re-identifikácie fungujú aj medzi viacerými záznamami.

Obr. 6.8: Vľavo: detekovaná a rozpoznaná osoba na zázname číslo 1. Vpravo: detekovaná a správne re-identifikovaná osoba na zázname číslo 2.

6.2.4 Experiment 4 - viac osôb, tri záznamy

Tento experiment vychádzal z najkomplexnejšieho súboru v pripravenom datasete. Jedná sa o súbor, ktorý obsahuje tri záznamy s celkovo troma osobami, kde dve osoby prejdú cez všetky záznamy a jedna prejde cez dva z troch záznamov. V záznamoch taktiež dochádza k prekrytiu osôb. Cieľom experimentu bolo ukázať potenciál aplikácia rozoznávať osoby naprieč viacerými záznamami.

Z obr. 6.9 môžme vidieť, že experiment bol úspešný, kde všetky osoby boli správne re-identifikované v každom zázname, do ktorého vstúpili a aplikácia si taktiež poradila z prekrytím osôb v zázname číslo 2.

Obr. 6.9: Vľavo hore: osoba s ID = 0 naprieč záznamami 1, 2 a 3. Vľavo dole: osoba s ID = 2 naprieč záznamami 1, 2 a 3. Vpravo: osoba s ID = 1 naprieč záznamami 1 a 2.

6.3 Problémy aplikácie

Experimentovaním boli odhalené problémy s aplikáciou, ktoré sa pri vývoji nepodarilo odstrániť. Jeden z týchto problémov bol spomenutý v experimente 6.2.2 a je ním prekrývanie osôb. Podmienka, ktorá bola záverom tohto experimentu, do značnej miery s týmto problémom pomáha, avšak neodstraňuje ho úplne. Stále môžu nastať scenáre, kedy vďaka prekrývaniu osôb dôjde k prepisu niektorých príznakových vektorov a k takzvanému "ukradnutiu identity", kde človek v popredí obrázku môže spôsobiť, že osobe v pozadí už nebude možné priradiť príznakový vektor s jej pôvodným ID. Bohužiaľ sieť OSNet si s takýmito prípadmi sama o sebe poradiť nevie a súčasný návrh aplikácie taktiež nie.

Druhým a o niečo podstatnejším problémom sú chyby re-identifikácie osoby po vstupe do záberu, ako je napríklad zobrazené na obr. 6.5. Podobný problém, ako je zobrazený na tomto obrázku, nastal aj pri jednom z experimentov re-identifikácie dvoch osôb na dvoch záznamoch. Sieť OSNet v niektorých prípadoch vracia príznakové vektory s veľmi odlišnými hodnotami aj na základe zmeny pozadia alebo malej zmeny osvetlenia záberu. V tomto prípade ani nebolo pri vývoji jasné, ako by sa dal problém riešiť, keďže v spomínanom experimente mal nesprávny príznakový vektor k tomu hľadanému výrazne nižšiu euklidovskú vzdialenosť ako ten správny.

Kapitola 7

Záver

Hlavným cieľom tejto bakalárskej práce bolo vytvoriť aplikáciu na detekciu a rozpoznávanie osôb naprieč zábermi multi-kamerového systému. Predpokladaný multi-kamerový systém má kamery s neprekrývajúcimi sa zornými poliami a zábery z tohto systému sú zachytené simultánne. V rámci práce sú predstavené prístupy k riešeniu jednotlivých problémov a je uvedený návrh a implementácia, ktorá tieto prístupy spája. Aplikácia bola implementovaná v programovacom jazyku Python za pomoci frameworkov PyTorch a Torchreid. Na detekciu osôb bola využitá sieť Faster-RCNN s backbone MobileNetV3. Na rozpoznávanie a získanie príznakových vektorov osôb bola využitá sieť OSNet. Aplikácia má jednoduché užívateľské rozhranie, ktoré uľahčuje prácu s na-implementovanými časťami. Výstupom aplikácie je video s vyznačenými osobami a priradenými identifikačnými číslami.

Na testovanie aplikácie bol vytvorený dataset, ktorý je k práci priložený a obsahuje osemnásť videí a jedenásť testovacích príkladov. Aplikácia na experimentoch ukázala potenciál a veľkú presnosť súčasných state-of-the-art detekčných modelov, avšak aplikácia má aj problémy týkajúce sa rozpoznávacej siete OSNet, ktorá napriek tomu, že patrí na vrchol moderných riešení rozpoznávania osôb, zaviedla do implementácie problémy s re-identifikáciou, ktoré sa počas vývoja nepodarilo odstrániť. Táto aplikácia tak slúži ako dobrá praktická ukážka toho, že oblasť re-identifikácie osôb pomocou konvolučných neurónových sietí má stále veľký priestor na zlepšenie a samotná sieť OSNet, ktorá prišla s novým prístupom k tejto problematike, má veľký potenciál.

Možnosti pre pokračovanie v tejto práci určite sú. Ďalšia práca by sa mohla zamerať na praktickú ukážku rôznych iných modelov alebo na možné nasadenie aplikácie pre real-time použitie na viacerých kamerách, ktoré je lákavou témou problematiky multi-kamerových systémov.

Literatúra

[1] Agarap, A. F. *Deep Learning using Rectified Linear Units (ReLU)*. 2019.

[2] BenchmarkMagazine. *Mobotix MxActivitySensor 2.0* [http://benchmarkmagazine.com/mobotix-mxactivitysensor-2-0-2/]. [Online; accessed 23-November-2020].

[3] Bercman. *Bercman* [https://www.bercman.com/]. [Online; accessed 23-November-2020].

[4] Bhadeshia, H. K. D. H. Neural Networks in Materials Science. *ISIJ International*. 1999, zv. 39, č. 10, s. 966–979. DOI: 10.2355/isijinternational.39.966.

[5] Cai, Z. a Vasconcelos, N. *Cascade R-CNN: High Quality Object Detection and Instance Segmentation*. 2019.

[6] Chen, K., Ouyang, W., Loy, C. C., Lin, D., Pang, J. et al. Hybrid Task Cascade for Instance Segmentation. In:. Jún 2019, s. 4969–4978. DOI: 10.1109/CVPR.2019.00511.

[7] Choi, H., Kang, M., Kwon, Y. a Yoon, S. eui. *An Objectness Score for Accurate and Fast Detection during Navigation*. 2019.

[8] Dalal, N. a Triggs, B. Histograms of oriented gradients for human detection. In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. 2005, sv. 1, s. 886–893 vol. 1. DOI: 10.1109/CVPR.2005.177.

[9] Hasan, I., Liao, S., Li, J., Akram, S. U. a Shao, L. *Generalizable Pedestrian Detection: The Elephant In The Room*. 2020.

[10] He, K., Gkioxari, G., Dollár, P. a Girshick, R. *Mask R-CNN*. 2018.

[11] He, K., Zhang, X., Ren, S. a Sun, J. *Deep Residual Learning for Image Recognition*. 2015.

[12] Howard, A., Sandler, M., Chu, G., Chen, L.-C., Chen, B. et al. *Searching for MobileNetV3*. 2019.

[13] Iguernaissi, R., Merad, D., Aziz, K. a Drap, P. People tracking in multi-camera systems: a review. *Multimedia Tools and Applications*. Apríl 2019, zv. 78. DOI: 10.1007/s11042-018-6638-5.

[14] Ing. Milan Blaha, P. *Matematický model a aktivní dynamika neuronu* [https://portal.matematickabiologie.cz/index.php?pg=analyza-a-hodnoceni-biologickych-dat--umela-inteligence--neuronove-site-jednotlivy-neuron--jednotlivy-neuron--matematicky-model-a-aktivni-dynamika-neuronu]. [Online; accessed 23-November-2020].

[15] Kumar, H., Bhattacharya, S., Thomas, S. S., Gupta, S. a Venkatesh, K. S. Design of smart video surveillance system for indoor and outdoor scenes. In: *2017 22nd International Conference on Digital Signal Processing (DSP)*. 2017, s. 1–5. DOI: 10.1109/ICDSP.2017.8096120.

[16] Liu, W., Hasan, I. a Liao, S. *Center and Scale Prediction: A Box-free Approach for Pedestrian and Face Detection.* 2020.

[17] Masmoudi, M., Ghazzai, H., Frikha, M. a Massoud, Y. Object Detection Learning Techniques for Autonomous Vehicle Applications. In: *2019 IEEE International Conference on Vehicular Electronics and Safety (ICVES)*. 2019, s. 1–5. DOI: 10.1109/ICVES.2019.8906437.

[18] OpenCV. *OpenCV github repository* [https://github.com/opencv/opencv]. [Online; accessed 03-May-2021].

[19] Park, D., Zitnick, C., Ramanan, D. a Dollar, P. Exploring Weak Stabilization for Motion Feature Extraction. In:. Jún 2013, s. 2882–2889. DOI: 10.1109/CVPR.2013.371.

[20] Pelzl, C. *Intelligent pedestrian traffic lights: New system by TU Graz automatically recognises pedestrians' intent to cross the road* [https://www.tugraz.at/en/tu-graz/services/news-stories/media-service/singleview/article/denkende-fussgaengerampeln-neues-system-der-tu-graz-erkennt-kreuzungswunsch-automatisch0/]. 2019. [Online; accessed 23-November-2020].

[21] PySimpleGUI. *PySimpleGUI github repository* [https://github.com/PySimpleGUI/PySimpleGUI]. [Online; accessed 03-May-2021].

[22] PyTorch. *PyTorch github repository* [https://github.com/pytorch/pytorch]. [Online; accessed 03-May-2021].

[23] Ren, S., He, K., Girshick, R. a Sun, J. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.* 2016.

[24] Rieke, C. *Deep Learning for Instance Segmentation of Agricultural Fields.* Dizertačná práca.

[25] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A. a Chen, L.-C. *MobileNetV2: Inverted Residuals and Linear Bottlenecks.* 2019.

[26] Shao, S., Zhao, Z., Li, B., Xiao, T., Yu, G. et al. CrowdHuman: A Benchmark for Detecting Human in a Crowd. *ArXiv preprint arXiv:1805.00123*. 2018.

[27] Sinha, N. Emerging Technology Trends in Vehicle-to-Everything Connectivity. In: *2019 Wireless Telecommunications Symposium (WTS)*. 2019, s. 1–12. DOI: 10.1109/WTS.2019.8715535.

[28] Sun, P., Kretzschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V. et al. *Scalability in Perception for Autonomous Driving: An Open Dataset Benchmark*. December 2019.

[29] Tan, M. a Le, Q. V. *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*. 2020.

[30] Valueva, M., Nagornov, N., Lyakhov, P., Valuev, G. a Chervyakov, N. Application of the residue number system to reduce hardware costs of the convolutional neural network implementation. *Mathematics and Computers in Simulation*. 2020, zv. 177, s. 232 – 243. DOI: https://doi.org/10.1016/j.matcom.2020.04.031. ISSN 0378-4754. Dostupné z: http://www.sciencedirect.com/science/article/pii/S0378475420301580.

[31] Viola, P. a Jones, M. Robust Real-Time Face Detection. *International Journal of Computer Vision*. Máj 2004, zv. 57, s. 137–154. DOI: 10.1023/B:VISI.0000013087.49260.fb.

[32] Wang, J., Sun, K., Cheng, T., Jiang, B., Deng, C. et al. *Deep High-Resolution Representation Learning for Visual Recognition*. 2020.

[33] Waymo. *Waymo Technology* [https://waymo.com/]. [Online; accessed 23-November-2020].

[34] Wikipedia. *Artificial neural network — Wikipedia, The Free Encyclopedia* [https://en.wikipedia.org/wiki/Support-vector_machine]. 2020. [Online; accessed 09-January-2021].

[35] Xie, S., Girshick, R., Dollár, P., Tu, Z. a He, K. *Aggregated Residual Transformations for Deep Neural Networks*. 2017.

[36] Zhang, S., Benenson, R. a Schiele, B. CityPersons: A Diverse Dataset for Pedestrian Detection. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, s. 4457–4465. DOI: 10.1109/CVPR.2017.474.

[37] Zhang, S., Xie, Y., Wan, J., Xia, H., Li, S. Z. et al. WiderPerson: A Diverse Dataset for Dense Pedestrian Detection in the Wild. *IEEE Transactions on Multimedia (TMM)*. 2019.

[38] Zheng, L., Bie, Z., Sun, Y., Wang, J., Su, C. et al. *MARS: A Video Benchmark for Large-Scale Person Re-identification*. 2016.

[39] Zhou, K. a Xiang, T. *Torchreid: A Library for Deep Learning Person Re-Identification in Pytorch*. 2019.

[40] Zhou, K., Yang, Y., Cavallaro, A. a Xiang, T. *Omni-Scale Feature Learning for Person Re-Identification*. 2019.