VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

DETEKCE OBLIČEJE V NEKVALITNÍCH VIDEOZÁZNAMECH

FACE DETECTION IN POOR QUALITY VIDEOS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **MICHAL KOVAL**
AUTHOR

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

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav inteligentních systémů (UITS) Akademický rok 2020/2021

Zadání bakalářské práce

Student: **Koval Michal**
Program: Informační technologie
Název: **Detekce obličeje v nekvalitních videozáznamech**
Face Detection in Poor Quality Videos
Kategorie: Umělá inteligence

Zadání:

1. Seznamte se s dostupnými kamerovými systémy pro monitorování veřejného prostranství. Zjistěte, jaké jsou nedostatky těchto systémů při rozpoznávání obličeje.
2. Sumarizujte dostupné algoritmy strojového učení pro detekci obličeje. Zaměřte se na řešení detekce obličeje v nekvalitních videozáznamech (nízké rozlišení, špatné světelné podmínky).
3. Navrhněte řešení pro detekci obličeje v záznamech z kamer. Detektor zaměřte především na situace, kdy je obličej částečně zakrytý.
4. Navržený algoritmus implementujte v libovolném programovacím jazyce. Vytvořte k němu jednoduché uživatelské prostředí. Výsledná aplikace bude určená pro operační systém Linux.
5. Proveďte experimenty s cílem porovnat výkonnost vašeho detektoru s existujícími detektory.

Literatura:

- Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press Ltd, 2016, ISBN 978-02-620-3561-3
- Faen Zhang, Xinyu Fan, Guo Ai, Jianfei Song, Yongqiang Qin, Jiahong Wu. 2019. Accurate Face Detection for High Performance.

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

Táto bakalárska práca sa venuje problematike detekcie tvárí v nekvalitných videozáznamoch, pričom sa konkrétne zameriava na prekryté tváre. Popisuje základné princípy algoritmov strojového učenia a ich métody, ktoré sú často využívané v oblasti počítačového videnia. Z nich sú bližšie priblížené konvolučné neurónové siete a ich state of the art modely zamerané detekciu tvárí. V praktickej časti boli vytvorené a natrénované modely na detekciu tvárí inšpirované známym state of the art modelom RetinaFace. Najlepšia varianta z nich dosauje na WIDER Face HARD testovacom sete 85,5% priemernú presnosť a na testovacom sete zameranom na prekryté tváre 90,9%. Súčasťou práce je aj program s grafickým uživateľským rozhraním, ktorý poskytuje nástroje na použitie vytrénovaných modelov na videu a obrázkoch.

Abstract

This bachelor thesis deals with face detection in low quality videos, while mainly focusing on occluded faces. It describes elementary priciples of machine learning algorithms and their methods, which are often used in the field of computer vision. Out of them are more closely described convolutional neural networks and their state of the art models focused on face detection. Out of those, convolutional neural networks and state of the art models for face detection are more closely described. For the practical part face detection models inspired by state of the art model RetinaFace were implemented and trained. The best performing model achieves 85.5% average precision on WIDER Face HARD testing dataset and 90.9% on dataset focused on occluded faces. Part of this thesis is also a program with graphical user interfaces which provides tools to use developed models on videos and pictures.

Kľúčové slová

detekcia tváry, kamerové systémy, nekvalitné videozáznamy, neurónové siete, strojové učenie, konvolučné neurónové siete, Keras, RetinaFace, WIDER Face, prekryté tváre

Keywords

face detection, closed circuit cameras, poor quality videos, neural networks, machine learning, convolutional neural networks, Keras, RetinaFace, WIDER Face, occluded faces

Citácia

KOVAL, Michal. *Detekce obličeje v nekvalitních videozáznamech.* Brno, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Tomáš Goldmann, Ing.

Detekce obličeje v nekvalitních videozáznamech

Prehlásenie

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

.....
Michal Koval
11. mája 2021

Poďakovanie

Týmto by som chcel poďakovať pánovi Ing. Tomášovi Goldmannovi za jeho čas a odbornú pomoc pri vypracovávaní tejto bakalárskej práce. Taktiež chcem poďakovať svojím rodičom, rodine a priateľom za ich podporu počas celého štúdia.

Obsah

Kapitola 1

Úvod

Táto práca práca sa zaoberá problematikou detekcie tvárí v nekvalitných videozáznamoch. So zvyšovaním výpočetnej sily počítačov a pokrokoch v oblasti strojového učenia sa rôzne aplikácie počítačového videnia stávajú každodennou záležitosťou, ako napríklad odomknutie telefónu snímkom tváre alebo autonómne autá. Cieľom detekcie tváre je nájsť polohu tváre (napr. označiť rámčekom) v obraze. Tento krok je kľučový aj pri rozpoznávaní tvárí, pretože najskôr treba vedieť, kde sú aby sa dali identifikovať. Významnou aplikáciou tíchto technológií sú kamerové systémy. Využitím počítačového videnia sa dajú kamerové systémy výrazne zefektívniť. Aplikácie môžu byť rôzne, identifikácia hľadaných osôb, detekcie zločinu alebo dozor nad lockdownom počas pandemie COVID-19. Pre realizáciu týchto metód sa v praxi veľmi často používa implementácia pomocou rôznych typov neurónových sietí, ktoré sú vhodné najmä vďaka ich architektúre, schopnosti abstrakcie a učenia sa.

Na začiatku tejto práce je čitateľ oboznámený s oblasťou kamerových systémov. Sú popísané ich technické parametre, využitie a načrtnutá významnosť ich použitia spolu s algoritmami strojového učenia.

Ďalšia časť pokračuje popisom strojového učenia a metód na detekciu tvárí v obraze. Sú v nej vysvetlené základné pojmy umelej inteligencie a strojového učenia. Následne neurónové siete a konvolučné neurónové siete, ktoré sú základom mnohých algoritmov pre detekciu tvárí. Je popísaný princíp učenia neurónových sietí a ich súčasti, ako sú vrstvy a aktivačné funkcie. Pri konvolučných neurónových vrstvách je popísaná aj operácia diskrétnej dvojrozmernej konvolúcie, ktorá je základom konvolučných neurónových vrstiev. Na záver kapitoly sú popísané samotné algoritmy a modely strojového učenia pre detekciu tvárí.

V rámci praktickej časti bol navrhnutý a implementovaný detektor tvári so zameraním na prekryté tváre. Ten bol potom použitý v programe s grafickým uživateľským rozhraním, ktorý umožňuje jednoduché používanie detektoru a demonštruje jeho funkčnosť. Architektúra detektora vychádza zo *state of the art* metódy RetinaFace. Využíva modetné techniky ako sú pyramída príznakov, kontextové moduly a *multi-task* stratovú funkciu. Z tejto architektúry bolo implementovaných a vytrénovaných 6 rôznych modelov. Každý využíva inú takzvanú „chrbotvú“ sieť, ktoré slúžia na extraciu príznakov. Dektory boli vyhodnocované v 3 kategóriách. V priemernej presnosti pri všeobecnej detekcii tvárí, kde najlepší model dosiahol 85,5%. V priemernej presnosti pri detekcii prekrytých tvárí, kde najlepší výsledok bol 90,9%. Nakoniec po výkonnostnej stránke pri použití na videu, kde najvýkonnejší model dosiahol od 2 po 8,5 snímkov za sekundu, v závislosti od náročnosti videa.

Kapitola 2

Kamerové systémy pre monitorovanie verejného priestranstva

Priemyselné televízie alebo uzavreté kamerové systémy (CCTV - *closed-circuit television*) sú v dnešnej dobe veľmi bežné a stále rozširujúce sa prostriedky na monitorovanie verejného priestranstva. Fungujú na princípe podobnom televíznemu prenosu, avšak signál nie je verejne prístupný. Dnes sa už na prenos kamerového signálu používa aj internet. Podkategória kamier využívajúcich internet sa nazíva IP kamery [50], pretože využívajú *Internet Protocol* pre prenos signálu. Záznamy dokážu odosielať na vzdialené úložiská a je ich možno sledovať aj v priamom prenose.

V tejto kapitole sú opísané prvky kamery, ktoré hrajú hlavnú úlohu v kvalite snímku. Budú tu v krátkosti zhrnuté niektoré významnejšie príklady využitia tíchto systémov. Záver kapitoly sa venuje problematike efektívneho využívania kamerových systémov a načrtáva ponteciál aplikácie metód strojového učenia v tejto oblasti.

2.1 Prvky kamery

U každej kamery, či už sa jedná o bezpečnostnú kameru alebo digitálny fotoaparát, majú na výslednú kvalitu obrazu najväčší vplyv tieto parametre:

Optická sústava

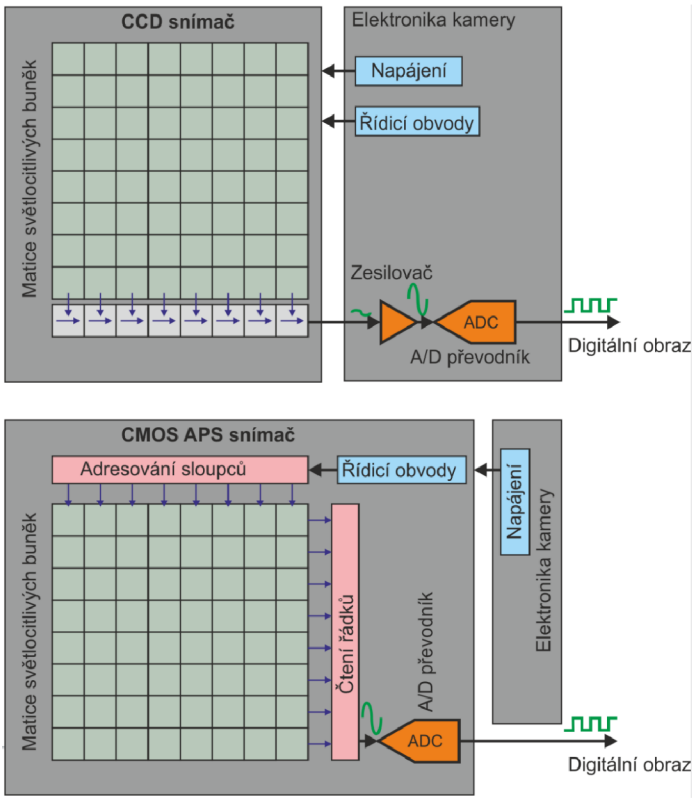
Optická sústava je oko kamery. Informácie sú prebraté z [68]. Účelom optickej sústavy je sústreďovať svetlo na senzor kamery. Zvyčajne sa skladá z niekoľkych vrstiev šošoviek. Snímané svetlo prechádza postupne každou vrstvou šošoviek, pričom každá má niejaký účel (filtrácia, rozptýlenie, spojenie). Šošovky sú vyrábané zo skla alebo plastu. Kľúčovou vlastnosťou materiálu je priepustnosť svetla, ktorú majú sklenené šošovky lepšiu a preto je aj ich cena vyššia. Ďalšou dôležitou súčasťou v optickej sústave je apertúra. Apertúra je zväzok tenkých kovových lamiel, ktoré vytvárajú kruhový otvor, pomocou ktorého uzatvárania a otvárania sa dá regulovať množstvo svetla prenikajúceho do objektívu. Nastavenia apertúry majú dopad na hĺbku ostrosti a celkovú svetlosť výsledného obrazu.

Snímač obrazu - senzor

Snímač je kľúčový prvok kamier. Konvertuje optický obraz z optickej sústavy na digitálny signál. Skladá sa zo sústavy jednoduchých fotosenzorov. Dnes sa používajú v zásade dva druhy obrazových senzorov, typu *Complementary Metal Oxide Semiconductor* alebo *Charge Coupled Device*.

- CCD (*Charge Coupled Device*) - Informácie prevzaté z [62]. Každá jednotlivá fotobunka CCD senzoru je analógová – pasívna. Elektrický náboj sa vygeneruje až po dopade svetla na ňu, a je reprezentovaný zmenou napätia na kontaktoch fotobunky. Toto napätie je merané ovládacím obvodom a ďalej spracovávané. Výsledkom je digitálna informácia a elektronická podoba obrazu, ktorý dopadol na plochu celej sústavy fotobuniek senzora. CCD senzor je pomerne jednoduchý a lacný na výrobu, má však niektoré obmedzenia, najmä z hľadiska rýchlosti.

- CMOS (*Complementary Metal Oxide Semiconductor*) - Informácie prevzaté z [62]. Obrazový čip typu CMOS má jednotlivé fotobunky aktívne, vyrobené polovodičovou technológiou CMOS. Každá fotobunka je vlastne fotodiódou, cez ktorú periodicky prechádza elektrický impulz. Zmena v jeho veľkosti či inom parametre nastáva v momente, keď na fotodiódu dopadne svetlo. Následne je táto zmena vyhodnocovaná a ďalej spracovávaná podobne, ako v prípade CCD senzora. Zložitejšia štruktúra samotného obrazového senzora znamená, že jeho výroba je náročnejšia, a teda aj drahšia. Na druhej strane znamená podstatne rýchlejšiu prácu pri snímaní obrazu.



Obr. 2.1: Schéma CMOS a CCD snímačov (zdroj [26]).

Obrazový procesor

Obrazový procesor je mozog kamery. Následujúce informácie sú prevzaté z [45]. Procesor spracuváva digitálny signál prichádzajúci zo senzoru. S týmto spracovaním sa spája aj mnoho parametrov, ktoré sa dajú na kamerách nastaviť, ako napríklad kontrast, saturácia, ostrosť a redukcia šumu. Z hľadiska výsledneho obrazu sa teda procesor stará o vyváženie bielych farieb, farebný priestor (sRGB, Adobe RGB), ostrosť, kontrast, saturáciu, farebný posun a kompresiu. Kvalita obrazového procesuru je najviac viditeľná práve na farbách snímky a redukcii šumu. Medzi jeho ďalšie funcionality patria aj automatické zaostrovanie, automatická expozícia, komunikácia s pamäťou a ukladanie snímky. Taktiež sa od neho odvíja aj spotreba energie kamerového zariadenia a jej optimalizáciou môžu výrobcovia digitálnych fotoapáratov zvyšovať počet snímkov na jedno nabitie.

2.2 Záznamové zariadenia

Podľa rebríčkou na portále heuréka[1] je najpredávanejšia bežná spotrebiteľská IP kamera Xiaomi Mi Home Security Camera 360° [30]. Je to kamera určená primárne na monitorovanie interiérov. Umožňuje otáčanie objektívu v rozsahu 360°, pričom jej zorné pole je 100°. V predaji sú varianty, ktoré snímajú obraz v HD (720p) a aj Full HD (1080p) kvalite pri 15 snímkoch za sekundu. Záznam môže ukladať na microSD alebo na cloudové úložisko. Je vybavená technológiami umelej inteligencie na detekciu pohybu a aj IR prísvitu s *Low-light true color* technológiou, čo jej umožňuje snímať aj v tme.

Druhou najpredávanejšou celkovo a prvou najpredávanejšou kamerou určenou pre exteriéry je Xiaomi IMILAB Outdoor Security EC2 [24]. Táto kamera tiež disponuje už zmienenými vlastnosťami ako predošlá, Full HD kvalita záznamu, nočné videnie, ukladanie na microSD alebo cloud a detekcia pohybu a aj ľudí. Jej zorné pole má 120°. Je prispôsobená na vonkajšie použitie tým, že je vodeodolná, zachytáva wifi signál až na vzdialenosť 100 metrov pri použití vhodného routru a disponuje 5100mAh batériou. Výdrž je ďalej optimalizovaná tým, že snímanie zapína len vtedy, keď detekuje pohyb, inač je v úspornom režime.

Obr. 2.2: Vľavo interiérová Xiaomi Mi Home Security Camera 360°, vpravo exteriérová Xiaomi IMILAB Outdoor Security EC (prevzaté z [25] [24]).

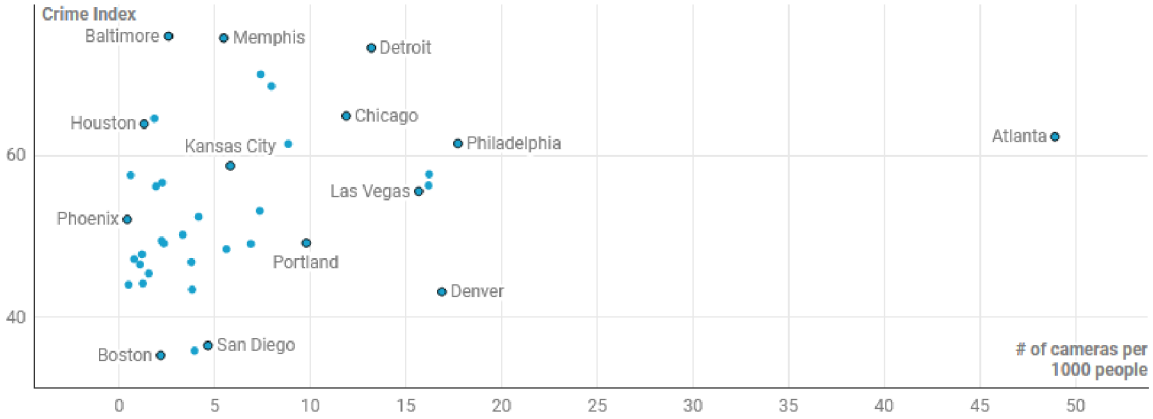
[1] https://ip-kamery.heureka.sk/

2.3 Príklady praktického nasadenia

Podľa odhadov z prieskumu trhu [44] bolo v roku 2020 vo svete 700 miliónov kamerových systémov pre monitorovanie verejného priestranstva. Prieskum zároveň predpovedá, že tento počet stúpne v roku 2021 na 1 bilión.

USA

Najväčšie mestá v USA sú pod veľkým dohľadom kamier. Zdrojom informácií bol [6]. V priemere má jedno mesto 6 kamier na 1000 obyvateľov, pričom mesto s najväčším osadením kamier Atlanta, má takmer 50 kamier na 1000 obyvateľov. V USA sú kamerové systémy široko rozsiahlé a používané od sledovania premávky, prevenciu zločinu na ulicia až po súkromnú ochranu majetku. Miami Dade County začalo počas COVID-19 pandémie používať verejné kamerové systémy s využitím umelej inteligencie k sledovaniu, či ľudia dodržiavajú regulácie sociálneho odstupu pre zamädzenie šírenia vírusu [29].



Obr. 2.3: Z grafu znázorňujúceho kriminalitu a osadenie kamier v mestách USA sa nedá vyčítať jasná korelácia medzi počtom kamier a kriminalitou (zdroj [6]).

Čínsky systém sociálneho kreditu

Zdrojom informácii je článok [13]. Čína je domovom pre 9 z 10 najviac sledovaných miest a je v nej viac ako polovica 54% všetkých všetkých bezpečnostných kamier na svete [31]. Hlavným motivátorom inštalovania toľkých kamier bol ich systém sociálneho kreditu. Jeho účelom je analyzovať sociálne správanie ľudí a na základe toho im prideľovať sociálny kredit, na základe ktorého môžu mať rôzne výhody alebo aj prekážky v bežňom živote. To uskutočnili vďaka ich veľkému počtu kamier, kontroly nad internetom, systému rozpoznávania tvárí a umelej inteligencii. Ich systém dokáže identifikovať aj vyhľadávané osoby, predikovať zločiny alebo sledovať podozrivé správanie.

COVID-19 lockdown v Moskve

Informácie prevzaté z [51]. V Moskve je umiestnených zhruba 170 tisíc bezpečnostných kamier nainštalovaných na uliciach a v metre. 100 tisíc z nich boli využité počas prvej vlny pandémie COVID-19 na monitorovanie ľudí, ktorí mali byť v karanténe. Ich systém

rozoznávania tvárí založený na neurónových sieťach má 95% úspešnosť a dokáže detekovať aj dodržiavanie odstupov medzi ľuďmi, či väčšie skupiny ľudí pohromade.

2.4 Problémy a efektivita kamerových systémov

Je mnoho faktorov a nedostatkov, ktoré hrajú rolu v efektívnom nasadení, týchto systémov. Môže sa jednať o nedostatočnú technickú výbavu týchto systémov, softwárové spracovanie alebo aj problémy spojené s ľuďmi, ktorí pracujú s kamerovými systémami.

2.4.1 Technické spracovanie

Ukladanie záznamov

Informácie v tejto sekcii sú prevzaté z [33]. Dnes už máme bezpečnostné kamery, ktoré dokážu zaznamenávať video vo vysokej kvalite. Problém ale nastáva s ukladaním videa. Kamery bežia stále po celý deň, každý deň v týždni. Jeden kamerový systém môže a aj zvykne mať niekoľko kamier súčastne aktívnych. Zákony sú všade iné, ale väčšina krajín vyžaduje aby sa kamerové záznamy ukladali po dobu niekoľkých dní až týždňov. Jedna kamera snímajúca v HD[2] kvalite na 30fps[3] dokáže za deň vyprodukovať aj 6TB dát. Preto administrátory kamerových systémov optimalizujú ukladanie dát. Komprimujú video záznamy alebo aj znižuju kvalitu nahrávania na kamerách. Čím dochádza k znižovaniu veľkosti videozáznamov. Ďalší spôsob je snímať menej snímkov za sekundu. Pri 15 snímkoch už ale videozáznam stráca plynulosť a môže tak nastať prípad, že sa vynechajú snímky dôležité pre identifikáciu páchateľa alebo iný účel kamerového systému.

2.4.2 Manuálne spracovanie

Výstupom kamerových systémov je obrovské množstvo vizuálnych informácii. To teda vytvára problém, ako ich efektívne spracovávať. S tým sa úzko spája ľudský faktor operátorov, ktorí videa z kamier sledujú v reálnom čase. Sledovanie a detekcia sledovaných javov z kamerových záznamov vyžaduje neustálu pozornosť operátorov týchto systémov. Takáto práca je mentálne namáhavá a stresujúca pre ľudí [67], čo vedie k zníženiu ich pozornosti, zhoršeniu ich pracovného nasadenia a v konečnom dôsledku k zníženiu efektívneho nasadenia týchto sledovacích systémov. Vo výskume [15] zameranom na proaktívne sledovanie na jednom monitore sa ukázalo, že tretina všetkých účastníkov, zameraná na všeobecné sledovanie odpútala pozornosť v priebehu prvých 30 minút sledovania. Tretina účastníkov, zameraná na detekciu špecifických javov dokázala udržať pozornosť až po dobu 90 minút. Túto prácu ešte viac komplikuje fakt, že v praxi je jednému operátorovi priradená viac ako jedna kamera a monitor.

Ďalší problém ľudských operátor predstavuje neúmyselná alebo percepčná slepota. Je to jav, kedy človek nevníma neočakávaný podnet, v dôsledku nedostatočnej pozornosti a zamerania sa iba na konkrétny sledovaný podnet. Výskum zameraný na neúmyselnú slepotu [47], pozostávajúci z laikov ale aj skúsených operátorov, vykonávaný v realistických pracovných podmienkach, ukázal že celkovo 66% sledovateľov nezaregistrovalo vyčnievajúci jav v simulovanom video zázname, piráta. Ďalej sa zistilo, že neočakávaný podnet nedôležitý pre

[2] *high definition* - 1280×720 rozlíšenie
[3] *frames per second* - snímky za sekundu

Obr. 2.4: Príklad kontrolnej miestnosti kamerových systémov (zdroj [58]).

účel sledovania má omnoho väčšiu šancu ísť bez povšimnutia 79% ako neočakávaný dôležitý podnet 55%.

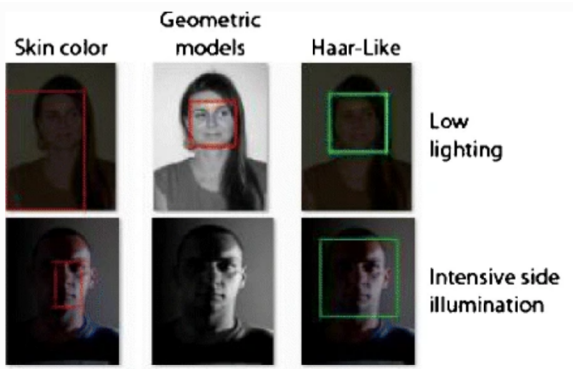
Práve kvôli týmto problémom s manuálnym spracovaním ľudskými operátorormi má zmysel vyvíjať a nasádzať inteligentné systémy pre efektívne používanie kamerových systémov. S tým však vznikajú ďalšie problémy a výzvy spájajúce sa so softwarovým spracovávaním obrazu z kamerových záznamov.

2.4.3 Umiestnenie kamery

Následujúce paragrafy čerpajú informácie z [33] [55]. Umiestnenie kamier je účelové. Je nutné ich inštalovať na miesta, ktoré je potrebné monitorovať. Z toho však vyplývajú komplikácie kvality obrazu spojené s prostredím kamery.

Svetelné podmienky

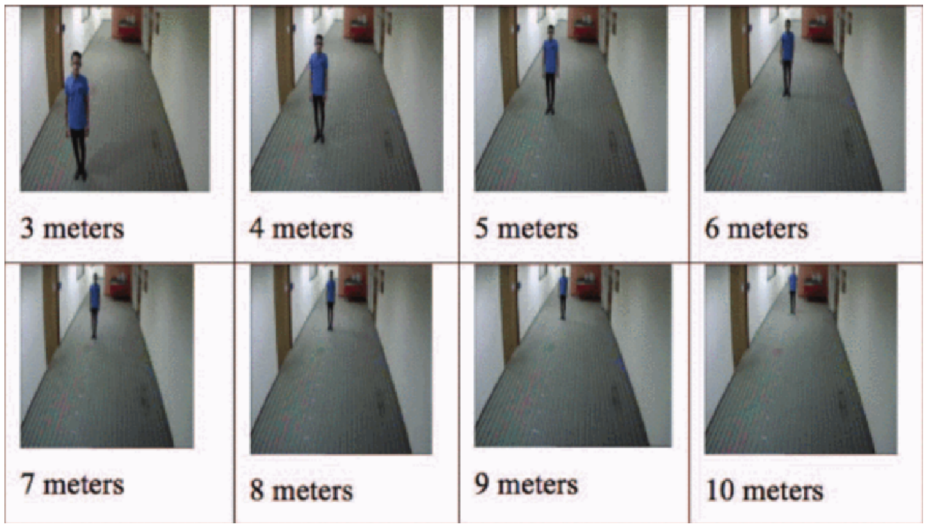
Kamerové systémy robia záznamy vkuse, v noci, cez deň, v každom ročnom období. V takom dynamickom prostredí sa svetelné podmienky neustále menia. Aj keď bola kamera nainštalovaná optimálne v lete, je veľmi možné, že bude mať nevhodné svetelné podmienky v zime, kedy je iné počasie a slnko zvykne zapadať skôr. Sú kamery, ktoré majú aj senzory na infračervené svetlo, pre nahrávanie aj v úplnej tme.



Obr. 2.5: Príklady detekcie tvári za zlých svetelných podmienok s rôznymi metódami (prevzaté z [36]).

Rozlíšenie a vzdialenosť objektu

Výskum zameraný na vzťah medzi vzdialenosťou sledovaných objektov a rozlíšenia kamier [55] ukázal, že jedna kamera s jedným rozlíšením pre dokonalé rozpoznanie tváre nestačí. Experiment bol prevedený s 3 rozličnými rozlíšeniami kamier a na vdialenostiach sledovaného subjektu od 3 až do 10 metrov. Výsledky sa líšili pre rôzne kamery na jednotlivých vdialenostiach. Najlepšie skóre dosiahla kamera s rozlíšením 1280×720 na vzdialenosť 3 metrov. Kamera s najväčším rozlíšením 1920×1080 dokázala rozonať subjekty s akceptovateľnými výsledkami až po vdialenosť 10 metrov. Ostatné kamery dokázili roroznať subjekt na maximálnu vzdialenosť 7 metrov. Výsledky ukazujú, že väčšie rozlíšenie kamery neprináša nutne väčšiu pravdepodobnosť presného rozoznania ale umožňujú rozoznávať aj subjekty vzdialenejšie od kamery, keďže dochádza k menšej strate kvalite pri orezaní a priblížení na subjekt. Zároveň pri tomto probléme treba brať do úvahy aj to, že v reálnych podmienkach sa budú so vzdialenosťou od kamery meniť aj svetelné podmienky a aj uhol, pod ktorým kamera sleduje subjekt. Limitáciu umiestnenia kamier treba riešiť advekvátnym umiestnením dostatočného počtu kamier vzhľadom k tomu, akú plochu chceme pokryť. Pri verejných priestransvách je odporúčané používať kamery s čo najväčšou kvalitou obrazu a rozlíšením, pretože pokrývajú veľké zorné pole.



Obr. 2.6: Extrahované snímky osoby zo záznamu podľa vzdialenosti od kamery (zdroj [55]).

Kapitola 3

Algoritmy strojového učenia pre detekciu tvári

Cieľom tejto kapitoly je uviesť čitateľa do problematiky strojového učenia a detekcie tváre v obraze. Na to je potrebné vysvetliť kľúčové pojmy z oblasti umelej inteligencie, strojového učenia a hlbokého učenia a ich vzájomný vzťah. Ďalej bude prebraná problematika neurónových sietí a konvolučných neurónových sietí, ktoré sú často využívané v oblasti počítačového videnia. Po tíchto potrebných základoch sú predstavené algoritmy a modely pre detekciu tvárí.

3.1 Umelá inteligencia, strojové učenie a hlboké učenie

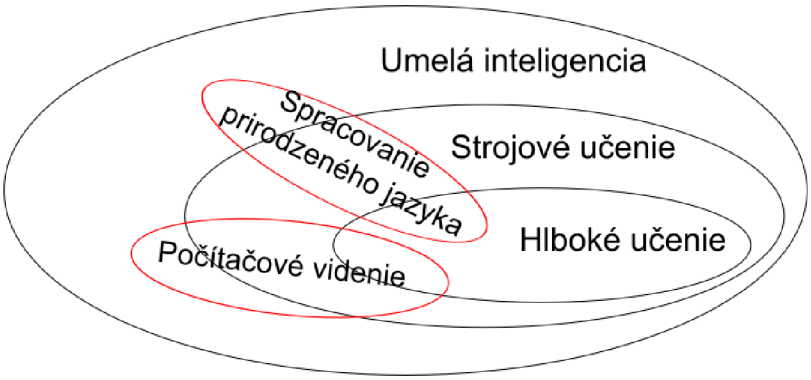
Mnohé problémy sú pre ľudí ľahko riešiteľné ale zároveň sú náročné na napísanie algoritmu, aby ich mohol riešiť počítač. Medzi ne patria aj napríklad rozpoznávanie obrazu či reči, ktoré sú pre nás prirodzené vďaka dlhým rokom evolúcie. Takéto problémy sú vhodné na riešenie metódami strojového učenia. Princíp strojového učenia spočíva v tom, že necháme počítač naučiť sa riešiť daný problém namiesto toho, aby sme mu dali explicitný postup riešenia. S rastúcim výpočetným výkonom a pokrokoch v oblasti strojového učenia v posledných rokoch sa stretávame s rôznymi formami automatizácie komplexných činností pomocou tíchto metód aj v bežnom živote. Od robotických vysávačov v domácnostiach až po autónmne (*self-driving*) autá.

3.1.1 Umelá inteligencia

Táto časť je prebratá z [9]. Myšlienka umelej inteligencie (AI - *Artificial Intelligence*) sa prvykrát zrodila v päťdesiatych rokoch, kedy sa hŕstka priekopníkov, z vtedy ešte iba rozvíjajúceho sa oboru počítačovej vedy, začala zaujímať, či by sme dokázali počítače prinútiť „myslieť“. Na túto otázku ešte nemáme odpoveď. Definícia umelej inteligencie sa postupom času ustálila na oblasť počítačovej vedy, ktorej účelom je automatizovať intelektuálne úlohy, ktoré sú normálne vykonávané ľuďmi. AI je obecný obor, ktorý zahŕňa strojové učenie, hlboké učenie ale aj ďalšie prístupy bez učenia, ako napríklad šachové algoritmy, ktoré fungujú na princípe explicitných pravidiel, ale nezahŕňajú žiadne učenie.

Dlho sa odborníci domnievali, že umelá inteligencia na úrovni človeka by mohla byť dosiahnutá tým, že by programátori mali dostatočne rozsiahlu sadu explicitných pravidiel pre manipuláciu s poznatkami. Táta idea umelej inteligencie je známa ako symbolická AI. Síze sa ukázalo, že symbolická AI je vhodná pre riešenie dobre definovaných logických prob-

lémov (hranie šachu), ale na druhú stranu sa postupom času prišlo aj na to, že vymyslieť explicitné pravidlá pre riešenie nejasnejších problémov, ako je klasifikácia obrazu či rozpoznávanie reči, je veľmi zložité. Reakciou na to bol vznik nového prístupu, ktorý nahradil symbolický AI: strojové učenie.



Obr. 3.1: AI podmnožiny.

3.1.2 Strojové učenie

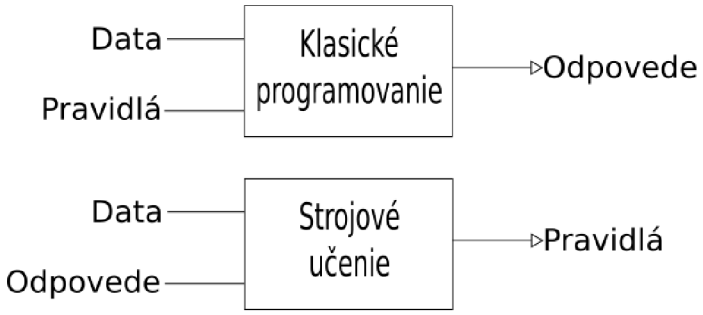
V klasickom programovaní je paradigma symbolickej AI také, že ľudia zadávajú pravidlá a vstupné dáta, ktoré budú programom spracované na základe pravidiel. Výstupom programu sú niejaké výsledky, odpovede, hodnoty. V strojovom učení sa mení táto postupnosť vstupov a výstupov. Ľudia vkladajú ako vstup data a očakávané výsledky spracovania tíchto dát. Výstupom sú pravidlá, ktoré potom môžu byť aplikované na novej sade dát k získaniu novej originálnej odpovede. Strojový učiaci sa systém je skôr natrénovaný ako deterministicky naprogramovaný. Je postupne zoznamovaní s relevantnými príkladmi pre daný účel a v tíchto príkladoch nachádza štatistickú štruktúru, pomocou ktorej nájde pravidlá pre automatizáciu úlohy. [9]

Metody strojového učenia vyžadujú k nájdeniu pravidiel tri veci:

- Vstupné datové body - V prípade detekcie tvárí z obrázkou, je potrebné dodať vstupnú sadu obrázkou, na ktorých sa bude trénovať.
- Očakávané výstupy - V obrazovej úlohe na rospoznávanie objektorv by to mohli byť označenia ako „auto“, „vlak“ a podobne.
- Metódu na hodnotenie úspešnosti algoritmu - Slúži na ohodnotenie správnosti výstupov algoritmu, používa sa aj ako spätná väzba pre úpravu spôsobu učenia.

3.1.3 Hlboké učenie

Zdrojom informácii pre túto sekciu boli [9][12][28]. Hlboké učenie je špecifická podmnožina strojového učenia. Princíp trénovania je na vyššej abstrakčnej úrovni. Jedná sa o viacstupňový hierachický spôsob, ako sa naučiť dátové reprezentácie. Hĺbka v hlbokom učení nenaznačuje hlbšie porozumenie problému dosiahnuté tímto prístupom, ale reprezentuje myšlienku postupných vrstiev reprezentácií. Počet vrstiev modelu sa nazýva práve hĺbka



Obr. 3.2: Strojové učenie ako nové programové paradigma.

modelu. Tieto vrstevné reprezentácie sú takmer vždy naučené použitím modelov nazývaných neurónové siete. Hierarchická architektúra moderných modelov hlbokého učenia často zahŕňa desiatky až stovky vrstiev reprezentácií, pričom sa všetky pri trénovaní učia.

Hlboké učenie je stále v rozvoji ale už bolo široko aplikované v tradičných oblastiach umelej inteligencie ako sú sémantická analýza, spracovávanie prirodzenej reči, počítačové videnie a mnoho ďalších. Rozvoju tejto oblasti najviac pomohli tri faktory: drastický nárast výpočtovej sily pri spracovávaní dat (GPU - nVidia CUDA [39]), výrazne nižšia cena hardwaru a najmä významné pokroky v oblasti strojového učenia. Pre rýchlejší rozvoj oblasti a jednoduhšiu prácu boli vyvinuté aj niekoľké softwárové frameworky zamerané na implementáciu metód hlbokého učenia. Medzi najpoužívanešie patria Keras a PyTorch pre jazyk Python, Deeplearning4j pre Javu alebo multyplatformové TensorFlow, MXNet a Caffe.

3.2 Umelé neurónové siete

Informácie vychádzajú z [38]. Umelé neurónové siete boli inšpirované nervovými sústavami živých organizmov, konkrétne mozgom, a tým ako pracuje s informáciami. Neurón, či už umelý alebo biologický v mozgu, je základná výpočtová jednotka. V mozgoch sú neuróny poprepajáné dokopy, čím vytvárajú siete na spracovanie dát. Neurónové siete v mozgoch sú dynamické štruktúry. Vstupy aj výstupy jednotlivých neurónov sa v čase menia. Tie sú v mozgu realizované ako jemné elektrické signály. Rovnako sa mení štruktúra siete zmenami prepojenia medzi neurónmi.

Umelé neurónové siete fungujú na rovnakom princípe, avšak značne zjednodušenom. Zmeny prepojení neurónov v mozgu sú simulované zmenami váhových koeficientov vstupov. Vyslanie elektrického signálu ďalším neurónom v mozgu je nahradené prahovou hodnotou, ktorá rozhodne či umelý neurón aktivuje výstup.

Umelé neurónové siete boli úspešne aplikované oblastiach počítačového videnia, spracovávania prirodzenej reči, autonómych áut, marketingu a mnohých ďalších.

Vrstvy siete

Umelé neurónové siete typicky pozostávajú zo stoviek umelých neurónov. Tie sú medzi sebou poprepájané a usporiadané do rôznych vrstiev. Výstupy n-tej vrstvy sú prevedené na vstupy n+1 vrstvy. Prvá vrstva sa nazýva vstupná a jej účelom je prijímať hodnoty z vonku pre spracovanie a priviesť ich na vstupy následujúcej vrstvy. Posledná sa nazýva výstupná a hodnoty na jej výstupe sú výsledkom spracovania vstupných hodnôt celým systémom.

Vnútorné vrstvy sa nazývajú skryté vrstvy. Ich počet sa odvíja od zložitosti úlohy, ktorú ma neurónová sieť vykonávať a na zvolenom type siete.

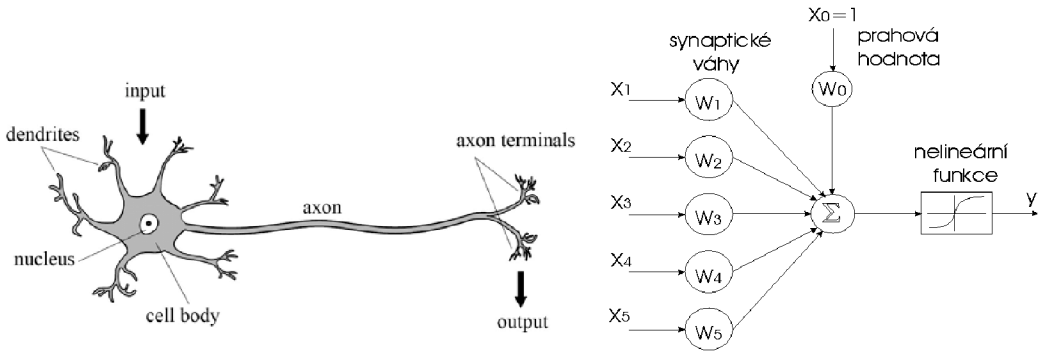
Umelý neurón

Prvý matematický model neuronu vytvorili McCulloch a Pitts v roku 1943 [37]. Tento model je používaný dodnes. Skladá sa z troch hlavných častí: vstupná, výstupná a funkčná. Vstupná časť sa skladá zo vstupov a z ich priradených a nastaviteľných váh. Pomocou váhových koeficientov sa dajú jednotlivé vstupy zvýhodňovať či potlačovať. Výkonná jednotka spracuje informácie zo vstupu a vyráta výstup. Výstupná časť spočíva v tom, že privádza výstupné informácie na vstupy iných neurónov.

Matematicky sa fungovanie umelého neurónu dá popísať vzorcom [60]:

$$y = \phi\left(\sum_{i=0}^{n} x_i w_i\right) \tag{3.1}$$

kde vstupné hodnoty x_i sú vynásobené ich príslušnými váhovými koeficientmi w_i a potom sú sčítané. Na sumu sa aplikuje aktivačná funkcia ϕ (spravidla nelineárna) a jej výsledok je výstupom neurónu, ktorý putuje na vstupy následujúcich neurónov.



Obr. 3.3: Podobnosť medzi biologickým neurónom a umelým neurónom v neurónových sieťach (zdroj [46] [41])

Na obrázku 3.4 je zobrazená prahová hodnota w_0, ktorá nie je pripojená k žiadnemu vstupu. Ak suma váženeho súčtu nie je väčšia ako prahová hodnota, tak sa neurón neaktivuje a jeho výstup sa nezmení.

Aktivačná funckia

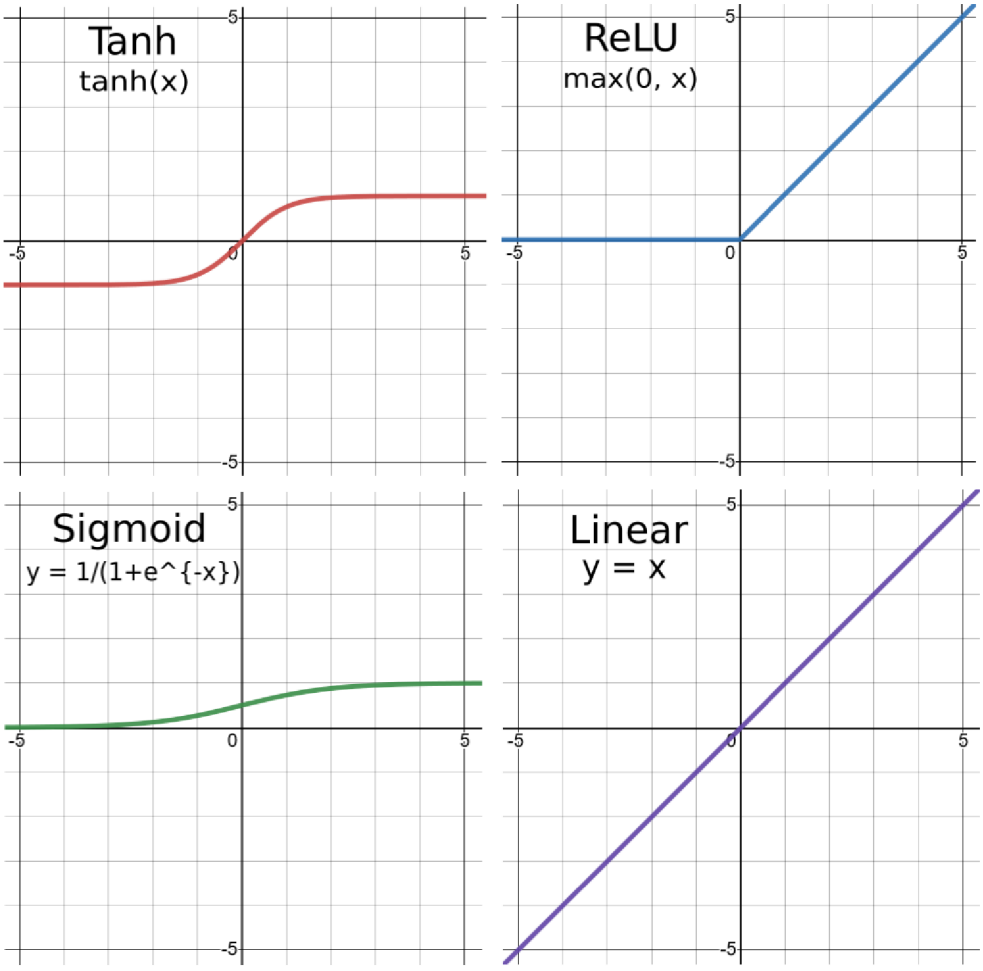
Informácie prevzaté z [60]. U väčšiny neurónových sietí prechádzajú výstupy z vrstiev cez prenosové (aktivačné) funkcie. Vnútorná suma vstupov je vyhodnotená aktivačnou funkciou, ktorá rozhodne o finálnej hodnote na výstupe neurónu. Ich zmyslom použitia je modifikácia úrovne výstupu do normalizovaných hodnôt. Bez použitia týchto funkcií by vstupná hodnota mohla dosiahnuť vysokých hodnôt, čo by mohlo spôsobovať problémy najmä u viacvrstvových neurónových sietí. Rôzne typy týchto funkcií majú rôzny vplyv na presnosť siete. Najviac používané sú nelineárne funkcie. Sú vhodnejšie pre zložitejšie dátové štruktúry, pretože sa pomocou nich model ľahšie generalizuje alebo adaptuje na rôznorodejšie dáta. Nasledujúce funkcie [34] patria medzi najpoužívanejšie.

Sigmoid graf má tvar prevráteného 'S'. Predpis funkcie je $A(x) = 1/(1+e^{-x})$. Zvyčajne je použitá vo výstupných vrstvách binárnej klasifikácie. Keďže je je obor hodnôt $< 0, 1 >$, výsledok môže byť odhadnutý ako 1 ak je hodnota vyššia ako 0,5 alebo 0 v opačnom prípade.

Hyperbolický tangens je matematicky posunutá verzia funkcie sigmoid. Sú si podobné a môžu byť od seba navzájom odvodené. Jej predpis funkcie je $tanh(x) = 2/(1 + e^{-2x}) - 1$ (s použitím sigmoidu môže byť aj $tanh(x) = 2 * sigmoid(2x) - 1$). Zvyčajne je používana v skrytých vsrtvách, vzhľadom k tomu že jej obor je $< -1, 1 >$. Vďaka tomu sa priemerná hodnota skrytých vrstiev pohybuje v okolí 0. Toto napomáha učeniu v následujúcich vrstvách.

ReLU (*Rectified Linear Unit*) je veľmi často používaná v konvolučných neurónových sieťach a v neurónoch skrytých vrstiev. Jej predpis je $A(x) = max(0, x)$. Záporné vstupy zmení na 0 a nezáporné hodnoty nechá bez zmeny, jej obor hodnôt je tak $< 0, \infty >$. ReLU je menej výpočtovo náročná ako tanh a sigmoid, pretože vyžaduje jednoduchšie matematické operácie (porovnanie). Tým že aktivuje následujúce neuróny len niekedy robí sieť riedšou a teda menej náročnou na výpočty. Inými slovami ReLU urýchľuje trénovanie siete.

Softmax je tiež podobná sigmoid funcii a rovnako je používaná v poslednej výstupnej vrstve v klasifikačných problémoch. V binárnej klasifikácii sú použiteľné obe, pri viac triednej klasifikácii sa spravidla používa softmax spolu s funkciou straty krížovou entropiou.



Obr. 3.4: Znázornenie vybraných aktivačných funkcií.

3.2.1 Trénovanie neurónových sietí

Informácie v tejte sekcii sú čerpané z [54]. Kĺúčovou vlastnosťou neurónových sietí je to, že programátor nešpecifikuje všetky parametre potrebné pre výpočet. Namiesto toho je neurónová sieť trénovaná tým, že je vystavená tisícom vhodných príkladov pre úlohu, pomocou ktorých si upravuje parametre tak, aby maximalizovala úspešnosť riešenia úlohy. Učia sa teda na základe „skúsenosti“. Úspešne vytrénovaná sieť dokáže vraciať správne odpovede, bez toho aby sme explicitne vedeli ako ich vyrátala. V stručnosti je trénovanie siete optimalizačná úloha, ktorá spočíva v upravení prepojení (váh) v sieti tak, aby sieť pri danom vstupe podala požadovaný výstup. Aktuálne sa na to používajú gradientové metódy, ktoré fungujú na princípe derivácií.

Stratová funkcia

Stratová funkcia (*loss function*) vyjadruje celkovú veľkosť chyby, ktorej sa dopúšťa predikčný systém. Čím horšie výsledky predikcií, tým viac rastie hodnota stratovej funkcie. Správne odpovede poznáme, keďže sa jedná o učenie s učiteľom a sú obsiahnuté v trénovacích dátach. Počas trénovania sa paremetre siete menia na základe chybovej funkcie, tak aby sa hodnota chybovej funkcie počas trénovania znižovala. Po zrátaní chybovej funkcie sa jej hodnota propaguje späť medzi jednotlivé neuróny, smerom od výstupnej vrstvy po vstupnú. Do každého neurónu v skrytých vrstvách sa dostane rôzna hodnota v závislosti od jeho podielu na celkovej hodnote stratovej funkcie. Takto sa šíri informácia o chybe cez celú sieť a každý neurón vie, do akej miery sa podielal na celkovej chybe.

Algoritmus spätného šírenia chýb

Algoritmus spätného šírenia chýb (*backpropagation*) je najpoužívanejším algoritmom učenia neurónovej siete. Je metódou spätnej propagácie chyby s následnou úpravou váh spojení umelých neurónov. Vykonáva postupne kroky optimalizačnej metódy zostup po gradiente (*gradient descent*) [53]. Mení parametre v neurónovej sieti opačným smerom, než je gradient chybovej funkcie. Tímto približuje chybu globálnemu minimu, ktoré je teoretickým riešením reprezentujúcim najmenšiu možnú chybu. Táto úprava parametrov sa vykonáva pri každej podanej dávke dát, pri každej iterácii trénovania neurónovej siete. Každá dávka môže posúvať smer gradientu a teda aj viesť k zmene parametrov siete.

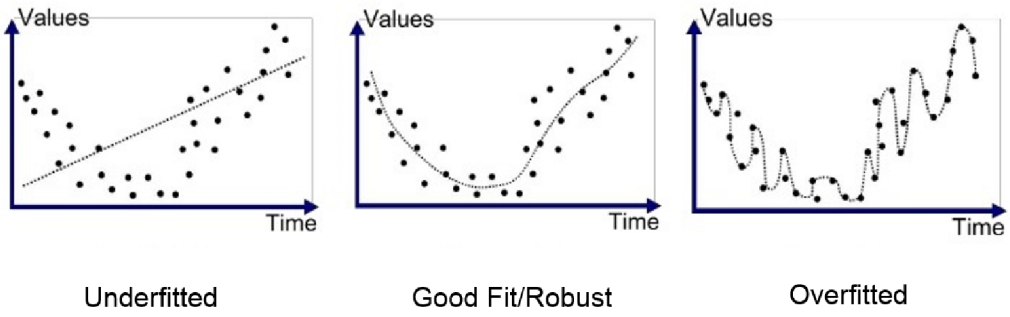
Pretrénovanie

Čím má sieť viac vrstiev a neurónov tým teoreticky lepšie dokáže aproximovať nejakú funkciu. Zvýšením počtu vrstiev a neurónov však často nastáva problém pretrénovania (*overfitting*). Pretrénovanie sa vyznačuje tým, že model má veľmi dobrú úspešnosť a nízku hodnotu stratovej funkcie na trénovacích dátach. Ale v reálnom použití a pri validačných testoch zlyháva. Pretrénovaním stráca model schopnosť generalizácie. Zabrániť pretrénovaniu sa dá zjednodušením modelu, rozšírením testovacích dát, augmentáciou dát, *droupout* metódou [59], regularizáciou alebo jednoducho znížením počtu iterácii, do takej miery kym ešte nedochádza k pretrénovaniu.

Podtrénovanie

Podtrénovanie (*underfitting*) je opakom pretrénovania. Model strojohého učenia je podtrénovaný, keď nedokáže aproximovať trend vychádzajúci z dát. Podtrénovanie znižuje presnosť

modelu, inými slovami model „nesedí“ (*fit*) dátam. Zvyčajne to nastáva pri nedostatku trénovacích dát alebo aj pri trénovaní lineárneho modelu pomocou nelineárnych dát. V takých prípadoch sú pravidlá modelu príliš jednoduché a flexibilné aby mohli byť aplikovateľné na malé dáta. Podtrénovanie môže byť napravené zvýšením komplexity modelu, zvýšením počtu príznakov, zvýšením doby trénovania a počtu epóch alebo skvalitnením trénovacej sady dát (rozšírenie, redukcia šumu).



Obr. 3.5: Ukážka aproximácie funkcie pri podtrénovaní, vhodnej miere trénovania a pretrénovaní (zdroj [5]).

Parametre trénovania neurónovej siete

Okrem koeficientov, ktoré sa upravujú počas trénovania podľa optimalizačnej metódy, tak aby sa hodnota stratovej funkcie znižovala, majú neurónové siete aj hyperparametre. Tie si vyžadujú manuálny zásah a úvahu programátora modelu, k tomu aby boli vhodne nastavené. Medzi tieto parametre patria:

- **Rýchlosť učenia** (*learning rate*) zadáva veľkosť kroku pri algoritme spätného šírenia chyby 3.2.1, keď sú parametre aktualizované podľa optimalizačnej funkcie. Reprezentuje ako dôležitá je zmena na váhach. Všeobecne je dobré tréning začínať s väčšou rýchlosťou učenia pre rýchlejšie štart učenia a postupne ju znižovať pre jemnejšie doladenie váh a zvýšenie presnosti.

- **Dávka** (batch) obsahuje trénovacie príklady, ktoré sú naraz dané neurónovej sieti počas trénovania. Veľkosť dávky vyjadruje počet trénovacích príkladov v jednej dávke. Obvyklá veľkosť dávky je 32, pričom sa môže meniť vzhľadom k pamäťovej náročnosti dát alebo rýchlosti trénovania siete. Priveľká dávka vedie k strate schopnosti modelu generalizovať.

- **Iterácia** (iteration/step) je jeden krok trénovania neurónovej siete, na ktorého konci sa prevedie úprava váh neurónov podľa metódy zostupu po gradiente [53]. Počet iterácii vyjadruje počet potrebných dáviek k spracovaniu jednej epochy. Teda platí vzťah:

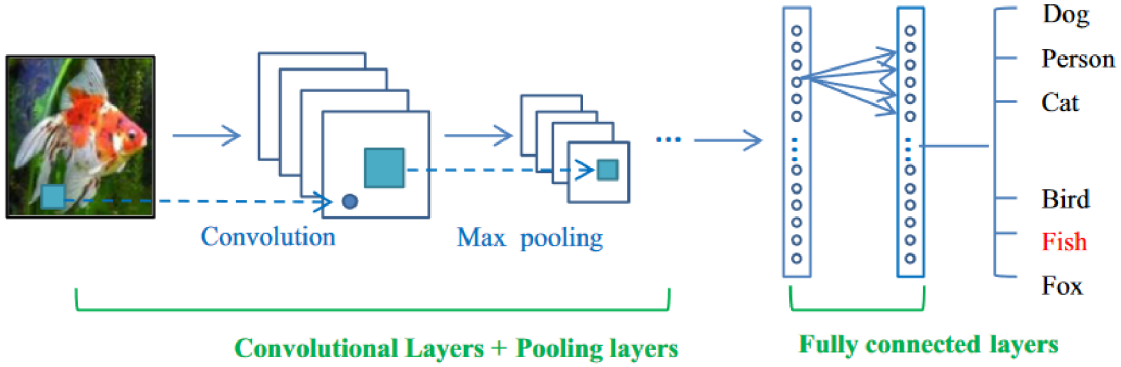
$$iterations * batch\,size = epoch \tag{3.2}$$

- **Epocha** je keď je spracovaná celá trénovacia dátová sada modelom raz. Celkový počet epóch teda vyjadruje koľkokrát spracuje algoritmus trénovací dataset. Počet epóch sa odvíja od používaného datasetu a náročnosti trénovanej úlohy.

3.2.2 Konvolučné neurónove siete

Zdrojom informácií v tejto sekcii boli [3][70][42]. Konvolučná neurónová sieť (CNN) je jedna z najvýznamnejších metód hlbokého učenia. Boli úspešne aplikované v širokej škále oblasti počítačového videnia.

Štruktúra CNN je znázornená na obrázku 3.6. Na ich vstupe je obrázok a na ich výstupe je predikcia triedy. Vstupný vektor vstupuje najskôr do konvolučnej vrstvy, kde sa vykonáva 2D konvolúcia. Výsledok konvolúcie je ďalej spracovaný aktivačnou funkciou (typicky ReLu 3.2) a jej výsledok putuje do *pooling* vrstvy. V nej prebieha transformácia obrazu pre zníženie dimenzií. Tieto vrstvy sa zvyčajne aj niekoľkokrát za sebou opakujú. Za poslednou konvolučnou vrtvou je tzv. *flatten* vrstva a v nej sa data prevedú do jedno dimenzionálneho vektoru. Tento vektor je vstupom plne prepojenej vrstvy. Tíchto vrstiev môže byť viac za sebou. Posledná vrstva zvykne mať aktivačnú funkciu Sigmoid 3.2 a toľko výstupov (neurónov), koľko je klasifikačných tried v úlohe.



Obr. 3.6: Znázornenie zreťazenie vrstiev vo všeobecnej CNN architektúre (zdroj [21]).

Konvolučná vrstva

Účelom konvolučnej vrstvy je počítať operáciu konvolúcie. Pri spracovávaní obrázkov sa používa diskrétna dvojdimenzionálna konvolúcia [3]:

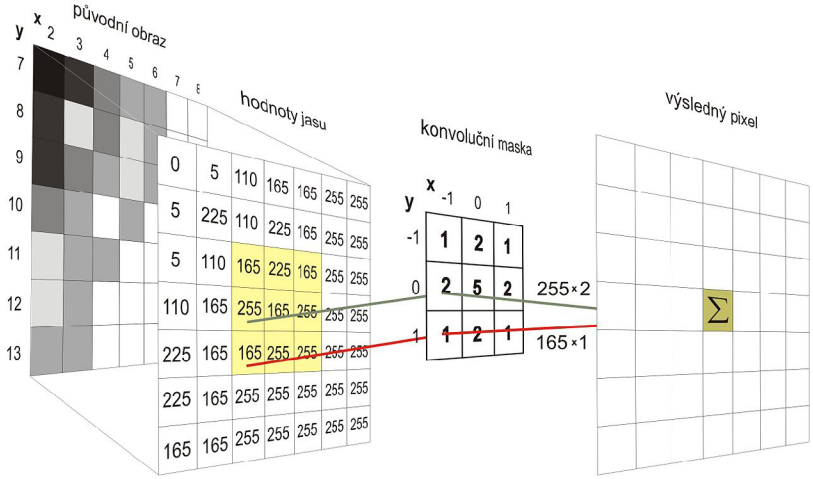
$$(f * h)(x, y) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} f(x - i, y - j)h(i, j) \tag{3.3}$$

kde f je vstupný obrázok, h je konvolučná maska a x, y sú súradnice riadkov a stĺpcov vo výslednej matici.

V diskrétnej konvolúcii sa dá jadro (*kernel*) vnímať ako tabuľka (konvolučná maska), ktorú pokládame na príslušné miesto obrázku. Každý pixel prekrytý tabuľkou sa vynásobí koeficientom v príslušnej bunke a následne sa spraví suma tíchto hodnôt. Suma reprezentuje hodnotu nového pixelu. V prípade masky o rozmeroch 3×3 budú prekryté 9 pixely. Koeficienty masky budú mať hodnotu 1/9. Nový pixel vyrátaný aplikáciou konvolúcie na jedno miesto v pôvodnom obraze bude priemerom deviatich okolitích pixelov. Koeficienty v konvolučnej maske udávaju vplyv hodnôt pixelov pod nimi. Ich správnym nastavením sa dá nadefinovať veľké množstvo operácií (derivácia obrazu, zvýraznenie hrán a ďalej). V rámci jednej konvolučnej vrstvy je aplikovaných niekoľko filtrov. Platí, že koľko filtrov, toľko vý-

stupných príznakových máp. Práve hodnoty týchto filtrov sú parametre, ktoré sa sieť musí naučiť pri tréningu.

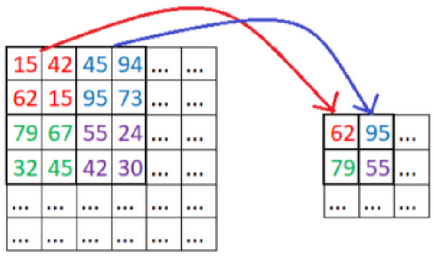
Na každú vytvorenú mapu príznakov je následne aplikovaná nelineárna funkcia, pričom medzi najpoužívanejšie patrí ReLU. Jej predpis je $f(x) = max(0, x)$, čiže len zmení záporné hodnoty na 0 a kladné ponechá bez zmeny.



Obr. 3.7: Princíp diskrétnej dvojrozmernej konvolúcie (zdroj [70]).

Pooling vrstva

Všeobecne sa *pooling* vrstvy (*pooling layer*) nachádzajú za konvolučnými vrstvami a sú používané na redukciu dimenzií mapy príznakov a parametrov siete, ale tak, aby sa zachovali najrelevantnejšie získané informácie. Pri operácii poolingu sa mapa príznakov rozdelí na niekoľko pod-okien veľkosti R×R s nulovým prienikom akýchkoľvek 2 okien. Z každého okna sa potom ponechá len jeho maximálna hodnota. Výstupy z jednotlivých okien sa spoja a vytvoria novú zmenšenú príznakovú mapu. Keď toto aplikujeme na všetky mapy, vznikne nová množina príznakových máp, ktorá poputuje ako vstup do ďalšej konvolučnej vrstvy, kde sa celý tento postup zopakuje. Inou technikou na zmenšenie dimenzionality výstupnej mapy je preskakovanie pixelov (*stride*). Pri konvolúcii používame pri spracovaní veľkej vstupnej mapy posuvné okno o veľkosti filtra. Zníženie veľkosti výstupnej mapy teda môžeme dosiahnuť tak, že posuvné okno po vstupe nebudeme posúvať o jeden, ale o viac pixelov. Posun však môže byť maximálne o veľkosti filtra, aby sme na vstupe nenechali nespracované pixely.



Obr. 3.8: Ukážka *Max Poolingu*, vyberá najväčšie hodnoty príznakov (zdroj [42]).

Plne prepojená vrstva

Za poslednou pooling vrstvou je niekoľko plne prepojených vrstiev (*fully connected layers*). Tieto vrstvy sú podobné vrstvám v tradičných architektúrach neurónových sietí. V prvnej z nich dôjde k spojeniu príznakových máp do jednej spoločnej vrstvy. Umožňujú nám využívať kladnú spätnú väzbu. Slúžia na vykonanie záverečnej klasifikácie, výstup tejto vrstvy môže byť využitý k získaniu predikcí, či určitá vlastnosť patrí do danej kategórie, alebo sa dá využiť ako vstupný vektor funkcie pre ďalšie spracovanie.

3.3 Algoritmy detekcie tvári

Identifikácia objektov z obrázkov je v dnešnej dobe veľmi rozšírená, či už sa jedná o detekciu a rozpoznanie ľudskej tváre, identifikáciu ručne písaných znakov alebo iných objektov priamo z obrázku. Detekcia tvárí je konkrétnym prípadom obecnej detekcie objektov, ktorá je podkategóriu počítačového videnia. Je to triviálny problém pre ľudí, ale náročný pre počítače. Pre realizáciu týchto identifikácií sa v praxi veľmi často používa implementácia pomocou rôznych typov neurónových sietí, ktoré sú vhodné najmä vďaka ich architektúre, schopnosti abstrakcie, generalizácie a učenia sa.

3.3.1 Viola-Jones algoritmus

Najvýznamnejší milník v oblasti detekcie tvári bol Viola-Jones detektor [66] publikovaný v roku 2001. Bol nazvaný po svojich autoroch, ktorí sa ním zapísali do histórie tejto vedeckej oblasti. Dokázal v detekovať tváre vo videozáznamoch v reálnom čase, čím dosiahol desať až stonásobnú rýchlosť v porovnaní s vtedy existujúcimi algoritmami.

Algoritmus vďačí za svoj úspech trom hlavným metódam:

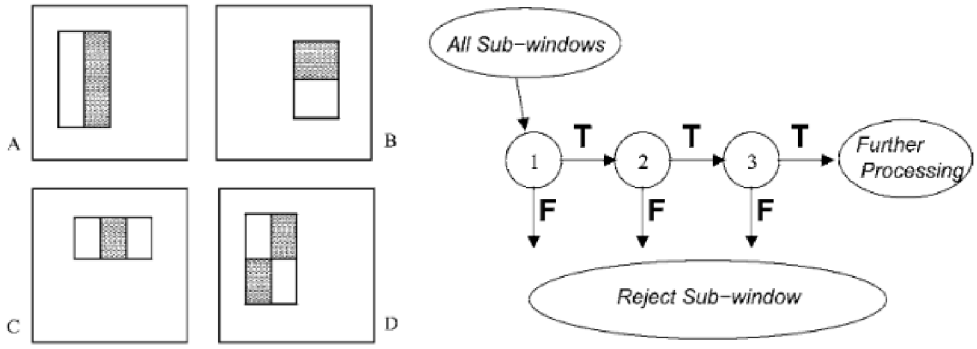
- Haar kaskádové príznaky, vyobrazené na obrázku 3.9. Dajú sa reprezentovať ako čierne alebo biele obdlžníky s ľubovoľnou veľkosťou, ktoré sa voľne pohybujú po obraze. Haarov príznak je číselná hodnota v stupňoch sivej farby vyrátaná ako rozdiel súčtu pixelov pod bielym obdlžníkom a súčtu pixelov pod čiernym obdlžníkom.

- Adaboost [14] - Detektor využíva rozhodovací strom na báze upraveného algoritmu Adaboost (obrázok 3.9), ktorý je optimalizovaný tak, že vždy vyberá čo najmenšiu sadu funkcií klasifikátorov tváre. Klasifikátory rozdeľujú príznaky do niekoľkých skupín, pričom sa začína vždy od všeobecných príznakov a pri prvom zlyhanom teste sa podobraz vylučuje a nepokračuje ďalej. Týmto sa výrazne redukuje počet iterácii klasifikátorov nad podobrazmi.

- *Integral Image* [4][57] - Technika, ktorá prevádza vstupný obraz do štruktúry integrálneho obrazu. Výpočty nad integrálnym obrazom vyžadajú výrazne menší počet aritmetických operácii pre extrakciu Haarovych príznakov.

$$ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y') \tag{3.4}$$

  Kde $ii(x, y)$ je integralny obraz, $i(x', y')$ je originalny obraz a x, y, x, y' sú súradnice pixelov.



Obr. 3.9: Haarove kaskádové príznaky a znázornenie rozhodovavieho stromu Adaboost (prevzaté z [66]).

3.3.2 Detekcia objektu

Informácie prevzaté z [56]. Moderné metódy detekcie objektov v obraze sú postavené na hlbokých konvolučných neurónových sieťach. Obrázok je daný na vstup siete a následne prechádza cez rôzne konvolučné a *pooling* vrstvy. Výstupom je vo formáte triedy najdeného objektu. Obecný postup pri detekcii objektov sa skladá z následujúcich krokov:

- Obrázok je na vstupe siete. Pred týmto krokom je možné ešte aplikovať metódy pred spracovania dát. V rámci obrazových dát sa často používajú jednoduché úpravy obrázkov ako rotácia, posuny, zmeny veľkosti a podobne. Týmto krokom sa obohacuje dátová sada, čo môže viesť v konečnom dôsledku k presnejšiemu modelu s lepšou schopnosťou generalizovať.

- Obrázok je rozdelený do niekoľkych rôznych regiónov. Tieto regony sú nazývané aj „*anchor boxes*“ alebo „*prior boxes*“. Sú rôzne metodiky na generáciu tíchto regiónov 3.3.2, pri čom cieľom je ich generovať vhodne množstvo a vo vyhovujúcich veľkostiach. Problém s týmito faktormi vzniká v tom, že ich môže byť vygenerovaných priveľa a drasticky sa tým zvýši výpočtová doba. V opačnom prípade by to zas viedlo k vynechaniu potencionálnych oblastí kde by sa mohli detekované objekty nachádzať a teda k zníženiu sensitivity detektoru.

- Každý z tíchto regiónov je považovaný za samostatný obrázok ktorý je daný konvolučnej neurónovej sieti, z ktorej získame jeho klasifikáciu. Či sa v regióne nachádza objekt, sa určí podľa Intersection over Union (IoU) metriky, bližšie popísanej v 5.1.

- Regióny a ich klasifikácie sú skombinované späť do dokopy, čím získaváme originálny obrázok s detekovanými objektmi.

Jednoštádiové a dvojštádiové detektory

Informácie prevzaté z [28]. Moderné detektory sa všeobecne delia do dvoch tried: jednoštádiové (*single-stage*) a dvojštádiové (*two-stage*) detektory.

Medzi moderné dvojštádiové detektory patria Faster R-CNN (*Faster Region-based Convolutional Neural Networks*) a Mask R-CNN. Dvojštádiové detektory rozdeľujú proces na 2 kroky: V prvej fáze používajú *Region Proposal Network* (RPN) k vygenerovaniu regiónov (nazývané aj kotvy), ktoré by mohli potencionálne obsahovať hľadaný objekt. Z na-

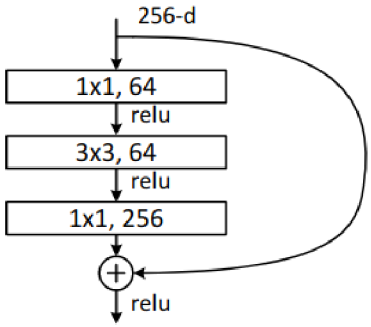
vrhnutých regiónov sú v druhej fáze extrahované príznaky operáciou *RoI Pooling*[1], ktoré sú následne použité ku klasifikácii a regresii ohraničujúcih oblastí objektov. Dvojštádiové detektory dosahujú najvyššie presnosti detekcií, avšak za cenu nízkeho výkonu.

Medzi jednoštádiové detektory patria YOLO *(You Only Look Once)* a SSD (*Singe Shot MultiBox Detector*). Jednoštádiové detektory riešia problém detekcie objektov ako jednu regresnú úlohu, tým že berú na vstupe obrázok a naučia sa z neho ohraničujúce oblasti a ich klasifikácie v rámci jedného priechodu modelom (*forward pass*). YOLO to dosahuje sofistikovanou stratovou funkciou v kombinácii s postprocesingom, ktorý sa zbavuje väčšiny oblastí. Tieto modelu majú nižšiu presnosť, ale sú výrazne rýchlejšie ako dvojštádiové modely a dosahujú až *real-time* výkon.

ResNet

Chrbtové siete (*backbone networks*) slúžia ako základné extraktory príznakov pre detekciu objektov. Vstupom sú pre nich obrázky a výstupom sú mapy vlastností vstupných obrázkov. Ako jeden z najúspejších backbone networkov za posledné roky sa ukázal ResNet, pretože je často používaný vo veľmi úspešných modeloch [49]. Bol súčasťou architektúry aj ďalej spomínaných state of the art modelov 3.3.3 a 3.12.

Sieť ResNet [23] bola vytvorená v roku 2015 výskumnou skupinou Microsoft Research. Je niekoľko typov tejto siete záležiac od počtu vrstiev, pričom najviac úspešne aplikované sú ResNet50 a ResNet152. V tej dobe prekvapil svojou hĺbkou, ale najinovatívnejší prínos bolo zavedenie štruktúry *residual block* 3.10. V tradičných sieťach každá vrstva posiela jej výsledok do následujúcej vrstvy. Residuálne bloky umožňojú ten istý výsledok posielať do následujúcej vrstvy a aj do vzdialenejšej vrstvy. ResNet je tiež jedna z prvých chrbtových sietí využívakúcich techniku dávkovej normalizácie (*batch normalization*) [7].



Obr. 3.10: Znázornenie štruktúry residual block v ResNet-50/101/152 (zdroj [23]).

MobileNet

MobileNet [27] je rodina výkonných chrbtových sietí, ktoré boli primárne určené na riešenie obrazových úloh na mobilných a vstavaných zariadeniach. Sú založené na efektívnej architektúre do hĺbky smerujúcich (*depth-wise*) oddeliteľných konvolúcií. Vďaka tomu sú to „ľahké“ siete, ktoré obsahujú málo parametrov, ale vysokú presnosť predikcií. Všetky vrstvy hĺbkovej konvolúcie sú následované vrstvou dávkovej normalizácie (*batch normalization*) [7]

[1]RoI - Region of Interest - región záujmu

a ReLU 3.2 aktivačnou funkciou, aby bola zaistená nelineárnosť. Posledná vrstva je plne prepojená vrstva 3.2.2, ktorá využíva Softmax 3.2 funkciu pre klasifikáciu.

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

Obr. 3.11: Architektúra chrbtovej siete MobileNet (zdroj [27]).

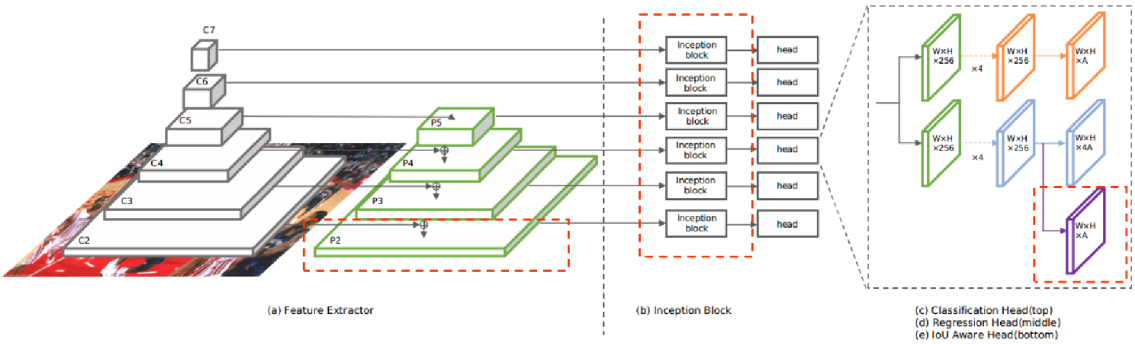
3.3.3 Moderné algoritmy

V tejto časti si predstavíme aktuálne *state of the art* metódy detekcie tvárí v obraze.

Úspešnosť metód je porovnávaná na základe benchmarku WIDER FACE [72] a rankingu na stránke Papers With Code [49]. WIDER FACE je aktuálne jeden z najrozsiahlejších a najnáročnejších dátových sád pre detekciu tvárí. Obsahuje 32 203 obrázkov a 393 703 anotovaných tvárí, z čoho je 40% trénovacích, 10% validačných a 50% testovacích. Dáta sú rozdelené do 3 sád podľa obtiažnosti detekcie do ľahkých, stredných a ťažkých. Obrázky v dátovej sade sú náročné na detekciu a približujú sa obtiažnosti aplikácie systému detekcie tvárí v skutočnom prostredí. Dátová sada obsahuje obrázky so širokou variáciou faktorov ovplyňujúcich obtiažnosť detekcie, ako sú rôzne veľkosti tváre, rôzne uhly fotky, rôzne svetelné podmienky, rozličné pozície tváre, výrazy tváre, rozličné stupne prekrytia tváre a makeup.

TinaFace

TinaFace [73] bola publikovaná koncom roku 2020 a v dobe písania tejto práce je to detektor tvári s najväčšou priemernou presnosťou 92,4%. Metóda poukazuje na to, že detekcia tvári je jednou z tried generickej detekcie objektov a dá sa riešiť technikami používanými v generickej detekcii objektov.



Obr. 3.12: Architektúra modelu TinaFace (zdroj [73]).

Na obrázku 3.12 je znázornená architektúra modelu TinaFace, ktorá sa skladá z nasledujúcich častí. (a) Extraktor príznakov ResNet-50 [23] a 6 levelová *Feature Pyramid Network* [35] pre extrakciu viacúrovňových príznakov vstupného obrazu. (b) *Inception block* pre zvýraznenie vnemového poľa. (c) Klasifikačná hlavica: 5 vrstiev FCN pre klasifikáciu základných príznakov. (d) Regresná hlavica: 5 vrstiev FCN pre regresiu základných príznakov. (e) *IoU Aware Head*: jedna konvolučná vrstva pre IoU 5.1 predikciu .

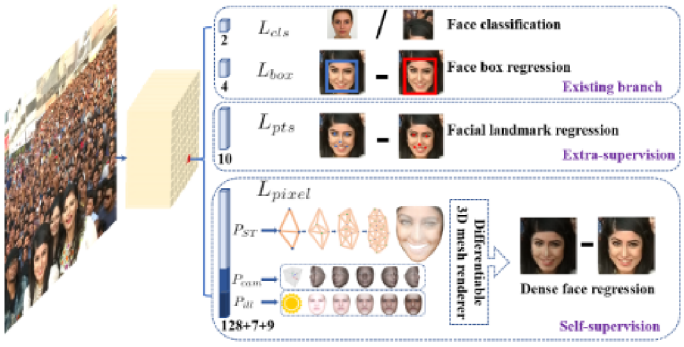
RetinaFace

RetinaFace [11] priniesol robustný jedno štádiový (*single-stage*) 3.3.2 detektor tvárí, ktorý vykonáva lokalizáciu tvári po pixeloch v rôznych škálach. Medzi jeho hlavné prínosy patria anotácie piatich poznávacích znakov tváre v WIDER Face datasete [72] a samokontrolovanú *mesh decoder* vetvu pre predikciu 3D informácií tvárí na pixelovej úrovni paralelne s už existujúcimi kontrolovanými vetvami. Metoda bola publikovaná v roku 2019 a v tej dobe dosiahla najlepšie skóre na hard WIDER Face 91.4%. Architektúra modelu pozostáva z pyramídovej architektúry extrakcie príznakov postavenej na ResNet-152 [23] chrbtovej sieti. Z jej jednotlivých levelov pyramídy sú výstupy napojené na samostatné kontextové moduly inšpirované z SSH [43] a Pyramid-Box [61]. Multi-task loss je vyrátaný z architektúry znázornenej na 3.13 následovným vzorcom:

$$L = L_{cls}(p_i, p_i^*) + \lambda_1 p_i^* L_{box}(t_i, t_i^*) + \lambda_2 p_i^* L_{pts}(l_i, l_i^*) + \lambda_3 p_i^* L_{pixel} \tag{3.5}$$

kde $L_{cls}(p_i, p_i^*)$ je klasifikačná strata, kde platí že: p_i pravdepodobnosť, že predikovaná oblasť i je tvár a p_i^* je 1 pre pozitívnu oblasť a 0 pre negatívnu. L_{cls} je *softmax* stratová funkcia pre binárne triedy (tvár/nie tvár). $L_{box}(t_i, t_i^*)$ je regresná strata detekovaných oblastí, kde platí že: t_i sú koordináty predikovanej oblasti a t_i^* sú koordináty skutočnej oblasti z anotácií. Koordinácie sú v normalizovanom stredovom formáte (stredový bod, šírka, výška). $L_{box}(t_i, t_i^*) = R(t_i - t_i^*)$, kde R je robustná stratová funkcia *smooth-L_1* zadefinovaná v [17]. $L_{pts}(l_i, l_i^*)$ je regresná strata tvárových znakov, pre ktorú platí že: l_i sú predikované koordináty znakov, l_i^* sú pravdivé anotované koordináty znakov a $L_{pts}(l_i, l_i^*)$ je *smooth-L_1* stratová

funkcia rovnako ako pri regresnej strate detekovaných oblastí. L_{pixel} je hustá regresná strata z analýzy pixelov *mesh* dekóderom bližie opísaná v [11]. $\lambda_1 - \lambda_3$ sú vyvážujúce parametre majúce hodnoty 0,25, 0,1 a 0,01, ktoré v celkovej strate dávajú väčší význam regresnej strate oblastí potom regresnej strate lokácií tvárovych znakov a najmenšiu hustej regresnej strate.



Obr. 3.13: Znázornenie nezávyslých vyhodnocovacích vetiev modelu RetinaFace (zdroj [11]).

Kapitola 4

Návrh a implementácia detektoru tvárí

Táto kapitola sa venuje praktickej časti bakalárskej práce. V tejto kapitole je popísaný návrh a realizácia modelu konvolučnej neurónovej siete určenej na detekciu tvárí inšpirovanej RetinaFace 3.3.3 modelom. Mimo samotnej implementácie modelu boli vytvorené trénovacie, testovacie skripty a skripty na prípravu súborov dát. Následne boli vytrénované modely použité v implementácii program s grafickým uživateľskym rozhraním, ktorý umožňuje modely používať a demonštruje ich funkčnosť a výkon. Na evaluáciu modelov sa používa metrika AP (*average precision*) 5.1.

4.1 Použité nástroje a technológie

V tejto časti sú popísané nástroje a technológie použité pri implementácii praktickej časti bakalárskej práce. Sú tu zhrnuté hlavné softwarové vývojé nástroje (jazyky, knižnice, frameworky) použité na implementáciu práce.

4.1.1 Python

Python [2] je multi platformový skriptovací programovací jazyk, ktorý ponúka vysokú úroveň abstrakcie. Podporuje rôzne programovacie paradigma, vrátane objektovo orientovaného, imperatívneho, procedurálneho alebo aj funkčného. Je vyvíjaný ako open source projekt, a v súčasnosti je pri verzii 3.9. Python poskytuje mnoho knižníc a má široký záber využitia ako napríklad tvorba webových aplikácií, *web scraping*, vizualizácia a spracovanie dát atď. Pre strojové učenie podporuje knižnice Keras, Tensorflow, MXNet, PyTorch a ďalšie.

Modely, trénovacie skripty a aj program s GUI boli implementované v jazyku Python 3.8.5.

4.1.2 Jupyter Notebook

Jupyter Notebook [16] (niekdajší názov IPython Notebooks) je webové interaktívne prostredie pre spúšťanie kódu. Jupyter Notebook dokument je dokument formátu JSON so skratkou `.ipynb`, ktorý môže obsahovať spustiteľné bunky kódu (Python), text (s použitím Markdown formátovania), grafy, obrázky a aj videa. Podporuje spúšťanie mnohých jazykov, vrátane Pythonu, R a Haskellu. Vďaka jeho možnosti postupného spúšťania kódu,

pekných vysvetľujúcich textov a vizualizácii výsledkov je Jupyter Notebook obľúbený v oblasti dátovej vedy. V rámci práce bol používaný na spúšťanie trénovacích a vyhodnocovacích skriptov.

4.1.3 Tensorflow

Tensorflow [64] [63] je zdarma *open source* softwarová knižnica určená pre strojové učenie. Bola vyvinutá Google Brain tímom pre interné používanie v Googli, ale bola neskôr vydaná pod Apache 2.0 licenciou v roku 2015. Je používaná v Googli v oblastiach produkcie, ale aj výskumu.

Knižnica je multiplatformová pričom je podporovaná na Linux, macOS, Windows, Android a dokonca aj na webových prehliadačoch s použitím JavaScript knižnice Tensorflow.js [20]. Hardwarovo podporuje CPU, GPU (CUDA), mobilné a vstavané zariadenia a Tensor Processing Unit (TPU). Tensorflow sa primárne dá používať v Jazykoch Python a C++, jeho vytrénované modely sa dajú exportovať do rôznych formátov a použiť aj s inými knižnicami a jazykmi.

Môže byť použitá na rôzne výpočetné účely ale jej hlavný cieľ je trénovanie a odvodenie hlbokých neurónových sietí. Tensorflow výpočty sú abstraktne vyjadrené ako grafy dátových tokov, kde uzly grafu predstavujú matematické operácie a hrany predstavujú data (tenzory). Vykonávanie numerických výpočtov teda prebieha tým, že tenzory prúdia cez graf.

Ako reakciu na pokles používania TensorFlowu vo výskumných prácach v prospech konkurenčného frameworku PyTorch [22] bola v roku 2019 vydaná verzia TensorFlow 2.0. Priniesla mnoho zmien, spomedzi nich jedna z najvýznamnejších bola zmena automatickej diferenciálnej schémy zo statického výpočetného grafu na „Define-by-run“ [10] schému. Ďalej odstránili staré knižnice, zaistili kompatibilitu medzi vytrénovanými modelmi na rôznych verziách TensorFlowu a vylepšili výkon akcelerácie s GPU. Kód napísaný v starších verziách musí byť prepísaný do Tensorflow 2.0, aby boli naplno využité nové funkcie a vylepšenia.

Tensorflow bol využitý ako *back-end* pre primárnu knižnicu strojového učenia používanú v bakalárskej časti, Keras.

```
model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10)
])
model.compile(optimizer='adam',
              loss=loss_fn,
              metrics=['accuracy'])
```

Výpis 4.1: Príklad vytvorenia jednoduchého modelu neurónovej siete s využitím Tensorflowu a Keras knižníc (prevzaté z [63]).

4.1.4 Keras

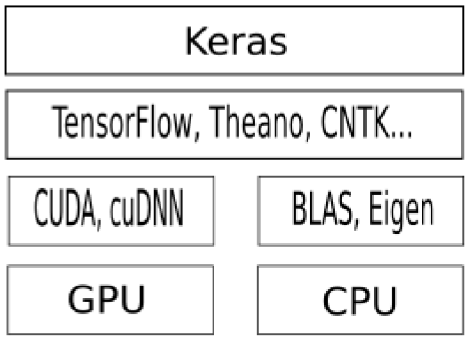
Keras [1] [9] je *open source* framework pre hlboké učenie v Pythone. Poskytuje pohodlný spôsob, ako definovať a cvičiť takmer akýkoľvek druh modelu hlbokého učenia. Pôvodne bol vyvinutý pre výzkumníkov s cieľom umožniť im rýchlejšie experimentovanie. Ponúka vysokú úroveň abstrakcie, pričom sám nerieši nízkoúrovňové operácie (manipulácia s tenzormi,

derivácie), ktoré prenecháva *back-end* tenzorovým knižniciam (TensorFlow, Theano, CNTK, MXNet, PlaidML). Keras má následujúce kľúčové vlastnosti:

- Umožňuje, aby rovnaký kód bežal na CPU, GPU alebo TPU.
- Má uživateľsky prívetivé API, ktoré uľahčuje rýchlo vytvárať prototypy modelov s hlbokým učením.
- Má vstavanú podporu pre konvolučné neurónové siete (využívané v počítačovom videní), rekurentné siete (využívané v spracovávaní sekvencií) a ich akúkoľvek kombináciu.
- Podporuje ľubovoľné sieťové architektúry: modely s viacerými vstupmi alebo výstupmi, zdielanie vrstiev, zdielanie modelov a podobne. Vďaka tomu je Keras vhodný pre budovanie akéhokoľvek modelu hlbokého učenia.

Keras je distribuovaný pod MIT licenciou, čo znamená, že môže byť používaný aj v komerčných projektoch. Má viac než 200 000 používateľov a je používaný známymi firmami ako Google, Netflix, CERN a obľúbený a úspešný na webovej stránke Kaggle, ktorá usporiadáva súťaže v hlbokom učení. Je kompatibilný s ľubovoľnou verziou Pythonu od verzie 2,7.

V Kerase boli implementované a natrénované modely na detekciu tvárí. Je tiež použitý vo finálnom programe na odvodzovanie z modelov. Táto knižnica bola zvolená kvôli tomu, že je vysokoúrovňová a jednoduchá na používanie s dobrou dokumentáciou, vďaka čomu je vhodná pre nováčikov v strojovom učení.



Obr. 4.1: Znázornenie tech-stacku strojového učenia frameworku Keras.

4.1.5 PyQt

PyQt [69] je väzba multiplatformového GUI toolkitu Qt pre programovací jazyk Python. Je vyvíjaný firmou Riverbank Computing a distribuovaný pod GNU/GPL-2 licenciou, čo znamená, že pri komerčnom použití je potrebné si zakúpiť licenciu. PyQt implementuje 440 tried a cez 6000 funkcíi a metód. V nich sú zahrnuté mnoho GUI prvky (*widget*), triedy pre prístup k SQL databázam, XML parser, SVG podpora a ďalšie. Iné alternatívy pre tvorbu GUI v Pythone sú PySide (ďalšia Qt väzba), PyGTK, wxPython a Tkinter (natívna GUI knižnica pre Python).

V práci bol použitý pre implementáciu grafického uživateľského rozhrania vo finálnom programe určenom na detekciu tvári. Táto grafická knižnica bola zvolená z dôvodu, že je multiplatformová a mal som predošlé skúsenosti Qt toolkitom, ale chcel som vyskúšať jeho Python obdobu.

4.1.6 OpenCV

OpenCV (*Open Source Computer Vision Library*) [48] je *open source* softwarová knižnica zameraná na počítačové videnie a strojové učenie. OpenCV bola vytvorená za účelom poskytnúť spoločnú infraštruktúru pre počítačové videnie a k urýchleniu nasadenia tejto technológie v komerčnej sfére. To je umožnené vďaka tomu, že je vydaná pod BSD licenciou, ktorá dovoľuje firmám používať a ďalej upravovať zdrojové kódy. Knižnica obsahuje viac ako 2500 klasických ale aj *state of the art* optimalizovaných algoritmov počítačového videnia a strojového učenia. Knižnica je široko používaná vo výskumných skupinách, firmách a štátnych podnikoch. Počet stiahnutí tejto knižnice prekračuje 18 miliónov. OpenCV je napísaná v C++ a poskytuje rozhrania pre C++, Python, Javu a MATLAB a je podporovaný na operačných systémoch Windows, Linux, Android a Mac OS. Táto knižnica je využívaná v rámci implementácie modelu ale aj vo výslednej aplikácii prevažne na prácu s obrázkami a videom

4.2 Použitý výpočetný hardware

Trénovanie neurónových sietí je výpočetne náročný a časovo veľmi zdĺhavý proces, preto je optimálne ich trénovať na čo najvhodnejšom hardwari. Teoreticky by sa dali trénovať na akomkoľvek počítači s dostatočne veľkou pamäťou pre model a dáta, ale aj na tích najvýkonnejších grafických kartách trénovanie trvá aj niekoľko dní. Trénovanie na vlastnom hardwari neprichádzalo do úvahy, kedže vlastním notebook iba s integrovanou GPU, ktorá výkonovo nie je dostačujúca. Kvôli tomu som sa rozhodol využiť cloudové výpočetné služby, ako sú Metacentrum a Google Colab, pričom som sa nakoniec rozhodol pre Google Colab, kvôli jeho jednoduchosti používania a natívnom spôsobe spúšťania programov pomocou Jupyter Notebook.

4.2.1 Metacentrum

Metacentrum [40] ponúka zdroje pre takzvané grifové výpočty. Grid je v podstate sieť mnohých prepojených počítačov, ktoré môžu byť umiestnené na rôznych miestach (mestách, inštitúciach) a ich vlastnosti (RAM, CPU, GPU) sa môžu líšiť. Metacentrum slúži na akademické účely a prístupu je nutná registrácia vyžadúújúca preukázanie príslušnosti k akademickej inštitúcii. Plánovací systém *Portable Batch System* (PBS) si udržuje prehľad o zdrojoch gridu (pamäť, CPU čas, miesto na disku) a necháva výpočetné úlohy čakať vo frontách, kym sa neuvoľní dostatok zdrojov pre ich beh. Uživatelia pripravujú a posielajú svoje úlohy z takzvaných čelných uzlov, ktoré sú rezervované pre aktivity užviateľov. Ostatné zdroje sú výpočetné uzly, ktoré vykonávajú samotné výpočty. Tento systém umožňuje spúšťať 2 typy úloh:

- **Interaktívne úlohy** umožňujú po pridelení zdrojov pracovať s terminálom interaktívne. Tento režim je optimálne využívať primárne pri ladení a testovaní, keďže je prehľadnejší. Nie je však efektívny z hľadiska využívania zdrojov, keďže sa čaká na vstupy od uživateľa. Ďalšou nevýhodou je, že uživateľ musí byť stále pripojený a pri zavretí terminálu alebo aj výpadku internetu sa úloha ukončí.
- **Dávkové úlohy** sú spúšťané ako *shell* skripty, ktoré obsahujú informácie pre PBS ako sú nároky na potrebné zdroje, pokyny k vykonávaniu úloh a samotné výpočetné úlohy. Keď sa dostane úloha na rad k výpočtu, začne sa vykonávať daný script. Dáv-

kové úlohy umožňujú automatizáciu, spúšťanie bez nutnosti uživateľa byť prítomný a plánovanie viacerých úloh naraz.

```
#!/bin/bash

# setting the PBS options
#PBS -N myFirstJob
#PBS -l select=1:ncpus=4:mem=4gb:scratch_local=10gb
#PBS -l walltime=1:00:00
#PBS -q gpu
#PBS -m ae

# running the tasks
task.sh
python script.py
maple < task.mpl
```

Výpis 4.2: Príklad dávkového skriptu.

4.2.2 Google Colab

Colaboratory (skratene aj Colab) [19] [18] je zdarma poskytované Googlom Jupyter Notebook prostredie, ktoré beží na cloude a ukladá dokumenty na Google Drive. Umožňuje komukoľvek spúšťať Python kód cez webový prehliadač bez nutnosti inštalácií a nastavovania a je obzvlášť vhodný pre dátovú vedu, strojové učenie a edukáciu. Na colaboratory sú predinštalované mnohé knižnice zaoberajúce sa strojovým učením a prácou s dátami. Na moje potreby nebolo nutné nič inštalovať manuálne, v prípade potreby je to možné použitím bežných *bash* príkazov, napr. pip pre inštalovanie Python knižníc.

Colab poskytuje uživateľom na výpočty využívať zdarma, ale limitovane GPU (NVIDIA Tesla K80, Tesla P4, Tesla P100 a Tesla T4) a TPU (špecializované procesory na tenzorové výpočty) pre akceleráciu trénovania neurónových sietí. Avšak nie je možné si zvoliť konkrétny typ karty, ktorá bude pridelená virtuálnemu stroju. Maximálna doba behu jedného virtuálneho stroju je 12 hodín pre zdarma uživateľov a 24 hodín pre PRO uživateľov, pričom PRO predplatné stojí 10$ na mesiac. Po uplnynutí tej doby, alebo pri nevyužívaní, sa virtuálny stroj vypne, aby sa neplýtvalo zdrojmy. Google takiež sleduje mieru používania hardwaru uživateľov a blokuje ich od prehnaného užívania, aby bol výpočetný hardware prístupný aj pre ostatných uživateľov. Tieto limity však nikde otvorene neuvádzajú, ale Google tvrdí, že PRO uživatelia môžu využívať zdroje viac [18].

Dáta uložené na disku virtuálneho stroja nie su perzistentné, takže treba pri každom spúšťaní uploadovať dátovú sadu a po ukončení sťahovať výsledky. Tento problém sa však dá jednoducho vyriešiť vďaka tomu, že je možné virtuálny stroj napojiť na Google Drive úložisko, a načítať z neho nie len Jypyter dokumenty ale aj dáta a ukladať tam výsledky.

Colaboratory bolo používané pretože je veľmi intuitívne a príjemné na používanie, pri čom vďaka jeho interaktivite je vhodné na ladenie ale aj na dlhodobé trénovanie. Je však nutné nechať zapnutý počítač a webový prehliadač počas trénovania. Trénovanie som spúšťal na noc a potom cez deň som reštartoval virtuálny stroj aby som resetoval časovač na využívanie GPU. Miestami som mal zapnutích aj viac virtuálnych strojov naraz, ale takým užívaním som rýchlo vyčerpal limity na využívanie hardwaru čo viedlo k tomu, že som musel

čakať aj deň, aby som mohol zase používať GPU na trénovanie. Kvôli tomu som si zakúpil PRO verziu, pri ktorej som už nenarazil na limitáciu pri rovnakom využívaní hardwaru, ako pred tým.

4.3 Návrh detektoru

Cieľom práce je navrhnúť riešenie pre detekciu tvárí vo videozáznamoch, so zameraním na prekryté tváre. Na základe poznatkov prebratých v rešeršnej časti práce navrhujem model určený na detekciu tvárí s následujúcimi vlastnosťami:

- Moderný detektor **založený na CNN architektúre** 3.2.2 a 3.3.2. Dnes sú konvolučné neurónové siete využívané vo všetkých aplikáciách počítačového videnia a zatiaľ nepoznáme lepšiu paradigmu, ktorá by ich v oblasti videnia pokorila CNN. Klasické metódy založené na kaskádových príznakoch a rozhodovacích stromoch 3.3.1 síce sú výpočetne nenáročné a schopné chodu aj na slabých zariadeniach a na vhodných dátach dokážu detekovať dobre. Problém u nich je pri komplikovaných problémoch ako sú práve prekryté tváre, kde metódy založené na CNN dokážu vďaka komplexnejšej arhitektúre lepšie aproximovať riešenia pre tieto komplexnejšie problémy.

- **Jednoštádiový detektor** 3.3.2 z toho dôvodu, že detektor je navrhovaný s primárnym využitím na video záznamoch, čo teda výkonnostne znamená, že by mal byť schopný spracovať obraz v rýchlosti 25 snímkov za sekundu, pre plynulý obraz. V prípade aplikácie so CCTV kamerami 2.2, by stačili aj 15 snímky za sekundu, keďže tie snímajú s redukovanou frekvenciou, kvôli šetreniu úložiska. Jednoštádiová architektúra detektoru je preto jasná voľba, keďže sú výrazne výkonnejšie ako dvojštádiové detektory.

- Pre zvýšenie presnosti na prekrytých tvárach navrhujem použiť pri trénovaní nie len štandardné anotácie obsahujúce ohraničujúce oblasti tvári ale aj anotácie **tvárových bodov** (oči, ústa, nos ...), podobne ako to bolo aplikované v 3.3.3. Moja úvaha je nad tým taká, že na prekrytej tvári, ktorej je vidno napr. len pravé oko, dokáže model naučený rozpoznávať tieto orientačné body tváre lepšie detekovať prekrytú tvár, ako model, ktorý sa učil bez nich.

- Ako ďalšie vylepšenie presnosti na prekrytých tvárach navrhujem použiť techniku **jemného ladenia** na dátovej sade obsahujúcej prekryté tváre, po tom čo bol prvotne vytrénovaný robustný model na všeobecnej trénovacej sade.

- Ďalej navrhujem skúsiť rôzne **varianty chrbtových sietí** pre extrakciu príznakov ako sú napr. 3.3.2 a 3.3.2 pre ich induviduálne vlastnoti. Malé siete by mohli poskytnúť výslednému riešeniu väčší výkon (potrebné pri videozáznamoch) a tie väčšie by mohli dosahovať väčšiu presnosť (vhodné na prekryté tváre). Cieľom je experimentovať a nájsť varianty, ktoré ponúkajú to najlepšie z oboch svetov.

4.4 Výber dátovej sady

Úspešnosť modelov neurónových sietí sa z veľkej časti odvíja od kvality dát, ktorými je sieť trénovaná. Dátová sada sa delí na trénovaciu a testovaciu, pričom môže byť využitá ešte aj

validačná, ale tá nebola v práci použitá. Na trénovacej sade sa model trénuje a na testovacej sa overuje schopnosť modelu generalizovať a správnej fungovať na doteraz nevidených dátach. Pre obe sady sa vyžaduje aby boli čo najviac podobné dátam z aplikácie modelu v reálnom svete. Pokiaľ by bol detektor používaný na viacerých miestach a situáciách, je potrebné dataset rozšíriť o o snímky daných scien a kontextov. Dataset sa skladá z samotných dát (obrázkov) a ich anotácií.

Na trénovanie a testovanie bol použitý WIDER Face [72] dataset. Je to verejne prístupná dátová sada zameraná na úlohu detekcie tvárí. Je to jeden z najrozsiahlejších datasetov pre tento účel, obsahuje 32 203 obrázkov a 393 703 anotovaných tvárí. Obsahuje obrázky s rôznou kombináciou náročných prvkov, ako sú rôzna veľkosť tvárí, pozícia, prekrytie, výraz tváre či osvetlenie. Mnoho *state of the art* modelov ho tiež používa, vrátane RetinaFace, z ktorého vychádza model tejto práce. RetinaFace ale pridal k totmuto datasetu dotatočné anotácie obsahujúce 5 znakov tváre, oči, nos a kútiky úst. Tie boli použité tiež. Formát anotácií pre trénovaciu sadu vyzerá následovne:

```
# Názov obrázku 1
xmin, ymin, xmax, ymax, ldm0x, ldm0y, ldm0v, ldm1x, ldm1y, ldm1v, ldm2x,
ldm2y, ldm2v, ldm3x, ldm3y, ldm3v, ldm4x, ldm4y, ldm4v, blur (pre tvár 1)
...
xmin, ymin, xmax, ymax, ldm0x, ldm0y, ldm0v, ldm1x, ldm1y, ldm1v, ldm2x,
ldm2y, ldm2v, ldm3x, ldm3y, ldm3v, ldm4x, ldm4y, ldm4v, blur (pre tvár N)
# Názov obrázku 2
...
```

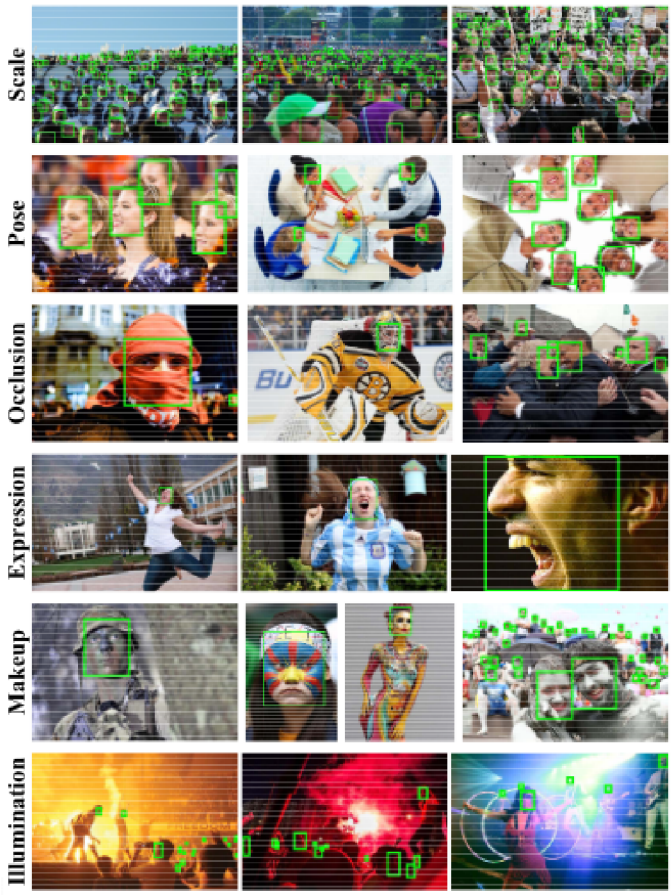
Výpis 4.3: Formát anotácí pre trénovaciu sadu.

kde skratka ldm reprezentuje *landmark* (tvárový znak). LdmNv je validačná hodnota znaku, ktorá môže nadobúdať hodnoty -1 pre nevalidný znak, 0 pre nespochybniteľný znak a 1 pre označiteľný znak. Rozmazanie obrázka (*blur*) nadobúda hodnoty na intervale $< 0, 1 >$.

WIDER Face bol použitý pri primárnom trénovaní modelov na vytrénovanie robustného všeobecného detektoru a zároveň bol z celého datasetu extrahovaný menší dataset obsahujúci iba obrázky s prekrytými tvárami. Táto dátova sada pozostáva z 5447 obrázkov a bola použitá na finetuning modelov, aby boli presnejšie na prekrytých tvárach. Ďalej bol pokus použiť ešte WFLW [71] dataset na obohatenie sady prekrytích tvárí, keďže jeho anotácie tiež obsahujú aj značenie prekrytých tvárí. Avšak prišiel som na to, že obsahuje takmer rovnaké obrázky ako WIDER Face, rozdiel medzi dátovými sadami je v anotáciach a množstve obrázkov. Nakoniec nebol zakomponovaný do sady prekrytých tvári, ale z jeho validačnej sady bola extrahovanáná pod sada obrázkov, obsahujuca prekryté tváre, ktorá pozostáva zo 668 obrázkov. Táto sada bola použitá na evaluáciu modelov detekovať prekryté tváre. Formát anotácií pre testovacie sady je:

```
# Názov obrázku 1 (Pre WIDER Face stredový formát boxov)
xmin, ymin, xmax, ymax pre tvár 1
...
xmin, ymin, xmax, ymax pre tvár N
# Názov obrázku 2 (Pre WFLW rohový formát boxov)
stred_x, stred_y, šírka, výška pre tvár 1
...
stred_x, stred_y, šírka, výška pre tvár N
```

Výpis 4.4: Formát anotácí pre testovaciu sadu.



Obr. 4.2: Ukážka dátovej sady WIDER Face (zdroj [72]).

4.5 Model konvolučnej neurónovej siete

Zo začiatku boli pokusy vytvoriť vlastný model ale detekcia tvári je príliš komplexný problém, preto som sa rozhodol prevziať a upraviť existujúce riešenie. Model zvolený na použitie v práci bol RetinaFace [11] už spomínaný v 3.3.3. Tento model bol prevzatý a upravený, pretože vyhovuje požiadavkám z návrhu 4.3. Je to *single-stage* detektor, čo je dobré pre výkonnosť, ktorá je dôležitá pri detekcii na videu. *Two-stage* architektúry rozdeľujú proces detekcie na fázu návrhu regiónov a klasifikačnú fázu. *Single-stage* priamo odvodzujú detekované oblasti. Použitie tvárových orientačných bodov bolo priamo inšpirované práve RetiFace metódou. Taktiež je táto metóda celkom známa a existuje niekoľko jej implementácii, pričom táto práca prevzala a upravila kódy z [52].

Na rozdiel od oficiálnej RetinaFace metódy v mojej nebol použitý *mesh-decoder* v implementácií modelu. Tento prvok je príliš zložitý a jedinečný pre RetinaFace metódu pričom nebola robená žiadna rešerša z oblasti grafových konvolúcií a 3D rekonštrukcie tváre a teda je vynechaný v rámci tejto práce. V stratovej funkcii *Multi-task loss* 3.3.3 je teda vypustený 4. prvku celkového súčtu, hustá regresná strata.

Ako chrbtové (*backbone*) siete boli použité ResNet-152 3.3.2, pretože bola použitá aj v oficiálnej implementácii RetinaFace a zároveň pretože je to rozsiahla sieť a teda predpokladám, že bude dosahovať vysokú presnosť. Ďalej ResNet-50 pretože je menšia a teda

očakávam od nej vyšší výkon a zároveň bola použitá v iných úspešných metódach, ako napríklad TinaFace [73]. Ako posledná bola zvolená MobileNetV2 3.3.2, ktorá je najmä navrhnutá na výkon, tak aby dokázala fungovať aj na mobiloch. Zo zvolených je to najmenšia sieť, pre porovnanie obsahuje 3 538 984 parametrov a ResNet-152 obsahuje 60 419 944 parametrov.

Medzi ďalšiu zmenu a pokus o optimalizáciu bola úprava pyramídy príznakov [35]. Oficiálna implementácia RetinaFace má výtupy z pyramídy príznakov P_2 až P_6. V takzvaných „lite“ verziách modelov boli vypustené P_2 (160×160×256) a P_6 (10×10×256) mapy príznakov a teda ostali P_3, P_4 a P_5 a ich jednotlivé kontextové moduly.

Kombináciou tíchto úprav vzniká dokopy 6 modelov, ktoré budú vytrénované a porovnané.

Layer (type)	Output Shape	Param #	Connected to
input_image (InputLayer)	[(None, 640, 640, 3)	0	
tf.math.truediv_49 (TFOpLambda)	(None, 640, 640, 3)	0	input_image[0][0]
tf.math.subtract_5 (TFOpLambda)	(None, 640, 640, 3)	0	tf.math.truediv_49[0][0]
MobileNetV2_backbone (Functiona	((None, 160, 160, 24	1364864	tf.math.subtract_5[0][0]
FeaturePyramid (FeaturePyramid)	((None, 80, 80, 256)	1849344	MobileNetV2_backbone[0][0] MobileNetV2_backbone[0][1] MobileNetV2_backbone[0][2] MobileNetV2_backbone[0][3]
SSH0 (SSH)	(None, 80, 80, 64)	117888	FeaturePyramid[0][0]
SSH1 (SSH)	(None, 40, 40, 64)	117888	FeaturePyramid[0][1]
SSH2 (SSH)	(None, 20, 20, 64)	117888	FeaturePyramid[0][2]
ClassHead_0 (ClassHead)	(None, None, 2)	260	SSH0[0][0]
ClassHead_1 (ClassHead)	(None, None, 2)	260	SSH1[0][0]
ClassHead_2 (ClassHead)	(None, None, 2)	260	SSH2[0][0]
BboxHead_0 (BboxHead)	(None, None, 4)	520	SSH0[0][0]
BboxHead_1 (BboxHead)	(None, None, 4)	520	SSH1[0][0]
BboxHead_2 (BboxHead)	(None, None, 4)	520	SSH2[0][0]
LandmarkHead_0 (LandmarkHead)	(None, None, 10)	1300	SSH0[0][0]
LandmarkHead_1 (LandmarkHead)	(None, None, 10)	1300	SSH1[0][0]
LandmarkHead_2 (LandmarkHead)	(None, None, 10)	1300	SSH2[0][0]
tf.concat_37 (TFOpLambda)	(None, None, 2)	0	ClassHead_0[0][0] ClassHead_1[0][0] ClassHead_2[0][0]
tf.concat_35 (TFOpLambda)	(None, None, 4)	0	BboxHead_0[0][0] BboxHead_1[0][0] BboxHead_2[0][0]
tf.concat_36 (TFOpLambda)	(None, None, 10)	0	LandmarkHead_0[0][0] LandmarkHead_1[0][0] LandmarkHead_2[0][0]
softmax_5 (Softmax)	(None, None, 2)	0	tf.concat_37[0][0]

Total params: 3,574,112
Trainable params: 3,543,392
Non-trainable params: 30,720

Obr. 4.3: Textový výpis architektúry MobileNet lite modelu.

4.6 Príprava dát

Pre prvý krok trénovania je potrebné vytvoriť z trénovacieho datasetu *TfRecord* súbory. *TfRecord* je formát binárneho súboru, ktorý ukladá sekvencie binárnych reťazcov. Je to efektívny spôsob seriálizácie štruktúrovaných dát, ako sú práve obrázky a im príslušné anotácie. Pri trénovaní s veľkým objemom dát je odporúčane používať tento binárny formát pre datasetm pretože to môže výrazne zvýšiť výkon načítacej *pipeline* a v konečnom dôsledku zníži trénovaciu dobu. Binárne dáta zaberajú menej miesta na disku, sú rýchlejšie skopírované a rýchlejšie čítané z disku. Mimo to, tým že *TfRecord* je štandardný formát Tensorflowu, je pre neho dobre optimalizovaný. Na vytvorenie *TfRecord* trénovacieho súboru bol použitý skript `create_train_record.py`, prevzaný a upravený z [52].

Príprava dát pre finetuning na prekytých tvárach pozostávala z výberu vhodných obrázkov z celého WIDER Face datasetu. Bol na to napísaný skript `generate_occlu_labels.py`, ktorý vyberá obrázky, ktoré obsahujú niejaké mierne alebo silne prekryté tváre (*occlusion* hodnoty rovné 1 alebo 2 v oficiálnom anotačnom súbore `wider_face_train_bbx_gt.txt`). Tento script vygeneruje nový textový súbor s anotáciami, s ktorým sa pomocou už spomínaného `create_train_record.py` vytvorí TfRecord súbor pre trénovanie modelov.

4.7 Konfiguračné súbory

Konfiguračné súbory obsahujú mnohé nastavenia modelu, trénovania, testovania cesty v rámci adresárovej štruktúry a podobne. Výhodou používania konfiguračných súborov je, že umožňujú jednoduchú modularitu, stačí zmeniť parameter skriptu a automaticky sa načítajú a odovzdajú nastavenia z celého konfiguračného súboru. Každý model tejto práce má vlastný konfiguračný súbor s nastaveniami. V priebehu práce sa začínalo s jedným konfiguračným súborom, ktorý bol kopírovaný a pozmenený podľa potrieb, preto sa môže stať, že nie všetky majú rovnaké možnosti.

Súbory boli napísane vo formáte YAML pre ich jednuduchú editáciu, prehľadnosť a parsovanie s použitím jazyka Python. Schéma názvov konfiguračných súborv je odvodená od použitej chrbtovej siete a počtu výstupov z pyramídy príznakov daného modelu. Napríklad `retinaface_resnet50x3.yaml`. Podobné názvy nesú aj pod priečinky jednotlivých modelov v adresárovej štruktúre.

4.8 Trénovanie

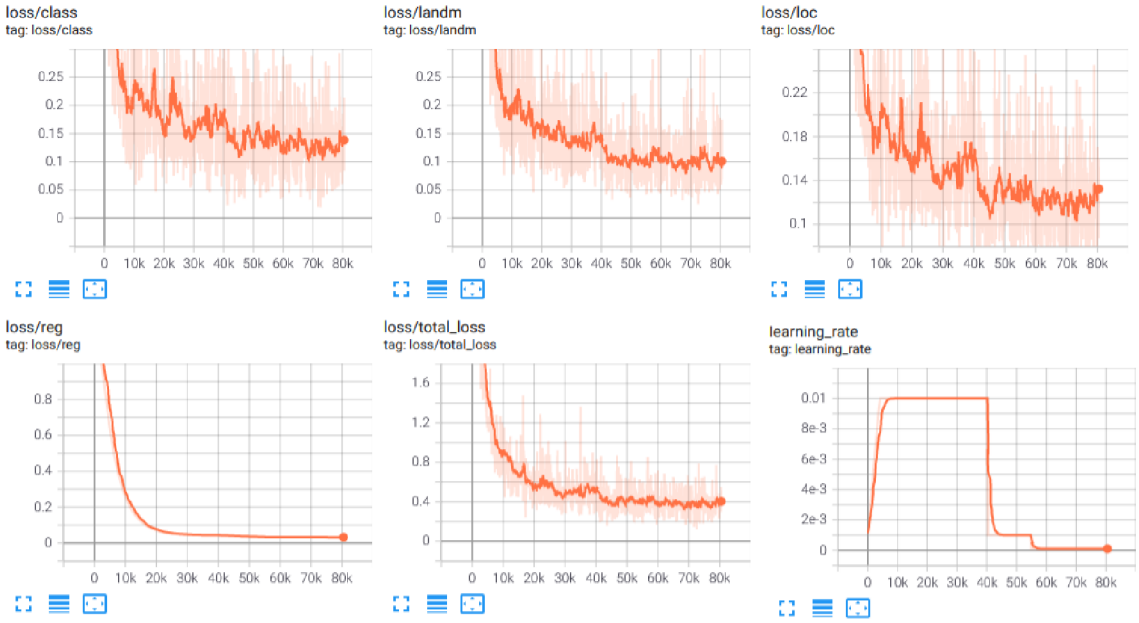
Trénovanie sietí je implementované v trénovacom skripte `train.py`, ktorý najskôr načíta konfiguráciu, dataset a model, priraví optimalizér a stratovú funkciu, v prípade pokračovania v predošlom trénovaní načíta uložené váhy a spustí trénovaciu slučku.

Doba trvania trénovania modelov bola rozdielna od jednotlivých modelov pričom trvala 1 až 2 dni. Rýchlosť učenia *learning rate* 3.2.1 sa začínala na 0,01 pričom minimálna hodnota bola nastavená na 0,001 Maximálny počet epóch bol nastavený na 100, pričom trénovania boli ukončené v rozmedzí 60. až 80. epochy. Spúšťanie evaluácie modelov prebiehalo manuálne, kedy v tíchto neskorších epochách hodnota stratovej funkcie výrazne neklesala aj po niekoľkých epochách. Evaluácia bola spustená po každom uložení váh, čo nastane v závyslosti na modely každých 3000 až 10 000 krokov. Pri ResNet-152 bolo potrebné nastaviť ukladanie na každých 10 000 krokov, pretože uložený model má zhruba 500MB a pri častom ukladaní sa rýchlo zapĺnila pamäť. V momente, kedy sa výsledok evaluácie nezlepšil s

následujúcimi 3 uloženými váhami, sa trénovanie ukončilo a za výsledný model boli zobraté váhy najpresnejšieho checkpointu z posledných troch.

Po tomto primárnom trénovaní na celej WIDER Face sade následovalo jemné ladenie (*finetuning*) na sade obsahujúcej prekryté tváre. Tuna bola použitá rýchlosť učenia 0,0001 pre jemnejšie doladenie váh na prekrytých tvárach a navýšená maximálna epocha na 300. Na testovacej sade na prekryté tváre extrahovanej z WFLW datasetu sa nepodarilo posunúť najlepšie skóre 90.9% AP a na WIDER Face testovacom sete to viedlo k miernemu poklesu presnosti u niektorých modelov. Pri pokračovaní v jemnom ladení už začínalo dochádzať k pretrénovaniu a aj modely, ktoré pred tým dosiahly 90.9% AP na prekrytých tvárach začínali klesať v presnosti. Varianta modelu ResNet-50 s piatimi levelmi pyramídy príznakov, však stúpla z predošlej presnosti 83,37% na 85,5% vo všeobecnej detekcii. Viac o výsledkoch je v 5.3. Jemné ladenie je implementované v skripte `fine_tune_wider_occ_all.py`.

Ako stratová funkcia pri trénovaní bol použitá modifikovaná *Multi-task* stratovná funckia z RetinaFace 3.3.3, s tým, že na klasifikačnú stratu bola použitá *categorical crossentropy* funkcia, váhové parametre boli vypustené a 4. časť súčtu celkovej straty, regresná strata pixelového dekóderu, bola vynechaná, keďže *mesh* dekóder nebol implementovaný. Stratová funkcia je implementovaná v súbore `losses.py`.



Obr. 4.4: Priebeh trénovania plnej varianty Mobilenet modelu. Výsledky sú vyhladené hodnotou 0,9 v v programe TensorBoard.

4.9 Aplikácia s GUI

Súčasťou zadania bolo vytvoriť aplikáciu s grafickým uživateľským rozhraním demonštrujúcu riešenie detekcie tvárí pre operačný systém Linux. Výsledny program bol vytvorený s použitím vytrénovaných modelov. Na tvorbu grafického rozhrania bolo použité PyQt. Program bol implementovaný čo najviac multiplatformovo, bol vyvíjaný na Linuxe, kde aj dobre funguje a bol testovaný aj na Windowse kde tiež funguje. Program sa spúšta skrip-

tom `face_detect_app.py`, ktorý obsahuje implementáciu GUI programu. Ikonky použité v programe boli prevzaté z voľne prístupnej databázy ikoniek flaticon.com.

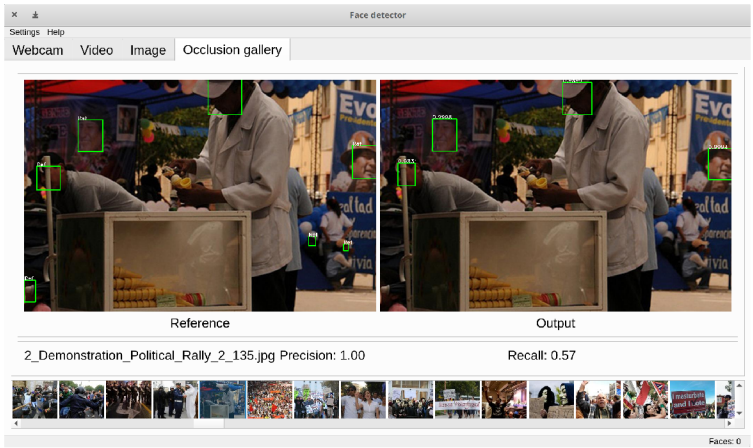
Program na detekciu tvárí umožňuje prepínať medzi vytrénovanými modelmi, pre ich jednotlivé otestovanie a porovnanie výkonnosti. Zároveň ponúka 4 nástroje na demonštráciu vytrénovaných modelov: detekcia z videokamery, z nahraného videa, z nahraného obrázku a nakoniec galériu prekrytých tvári. V každom režime sú dva bodové tlačítka (*radio button*), ktoré umožňujú aktivovať a deaktivať zobrazovanie ohraničujúcich obdĺžnikov a piatich tvárových znakov. V režimoch, kde je možnosť ukladania spracovaných obrazov, budú mať uložené obrazy vykreslené tieto možnosti podľa aktuálne zaškrnutých tlačítiek.

V režime snímania z webkamery je možné nahrávať video a aj spraviť fotku z aktuálneho snímku. Rýchlosť prehrávania záznamu z kamery závysí na výkonnosti počítača a použitom modeli.

V režime prehrávania videa sa zvolené video najskôr po načítaní spracuje modelom, čo trvá dlhšie pri dlhých videách, aby potom bolo možné ho plynule prehrávať. Režim ponúka štandarné vlastnosti video prehrávača ako je pretáčanie zastavovanie a podobne. Ďalej je možné uložiť kópiu videa s vykreslenými oblasťami alebo aj znakmi. Video je prehrávané bez zvuku, kedže ten nie je pri aplikácii detekcie tvári potrebný. Nahrávanie videa je možné vykonať aj pomocout tzv. *drag and drop* funkcionality. Video súbor je potrebné uchopiť a premiestniť myškou a pustiť nad video plochou.

Obrazový režim umožňuje nahrať a vykresliť do obrázku detekované tváre. Upravený obrázok sa dá uložiť. Podobne ako vo video režime, aj tu sa dá použiť *drag and drop* na otvorenie obrázku v programe.

V režime galérie prekrytých tvári je možné si vyberať pomocou posúvneho okna obrázky, na ktorých sa potom vykreslia referenčné ohraničenia tvári a výstupy z aktuálne zvoleného modelu vedľa seba pre jednoduché porovnanie. Pod obrázkami sa zobrazuje aj názov obrázku a vypočítaná *precision* (presnosť) a *recall* (sensitivita) 5.1. Obrázky pre galériu boli vybrané z validačnej sady WIDER Face datasetu pomocou skriptu `extract_occ_images.py`. Skript vyberá iba obrázky ktoré obsahujú aspoň jednu silne prekrytú tvár, teda v anotácia na pozícii „*occlusion*“ je hodnota 2. Výstupom je textový súbor obsahujúci vybrané názvy obrázkov a ich anotácie a zároveň vytvorí pre dané obrázky ikonky, ktoré sú zobrazené v slidery galérie. Po skripte boli obrázky ešte manuálne vybrané tak, aby galéria obsahovala obrázky, na ktorých si jednoducho všimneme niejaké prekryté tváre aj ľudským okom pre jednoduché porovnanie.



Obr. 4.5: Ukážka programu v režime galérie prekrytých tvári.

Kapitola 5

Experimenty a dosiahnuté výsledky

V tejto kapitole sú zhrnuté dosiahnuté výsledky trénovania modelov. Sú porovnané vo všeobecnej detekcie tvárí s modernými metódami na WIDER Face testovacom sete. Následne je vyhodnotená ich schopnosť detekovať prekryté tváre z WFLW dátovej sady. Na záver je porovnaný ich výkon.

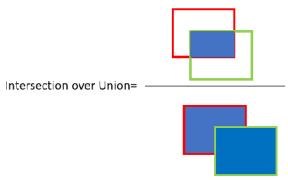
5.1 Metrika vyhodnotenia

Zdroj informácii v tejto časti bol [32]. Na zhodnotenie úspešnosti modelov bola použitá metrika priemerná presnosť (*average precision*), ktorá je používaná na hodnotenie WIDER Face dátovej sady a aj v ich oficiálnych hodnotiacich nástrojoch. Je vyrátaná ako hodnota presnosť(*precision*)/sensitivita(*recall*), kde tieto hodnoty sú vyrátané následovne:

$$Precision = \frac{true\ positive}{true\ positive + false\ positive} \tag{5.1}$$

$$Recall = \frac{true\ positive}{true\ positive + false\ negative} \tag{5.2}$$

Detekcie sú vyhodnotené ako skutočne pozitívne, falošne pozitívne a falošne negatívne podľa metriky *Intersection over Unio* (IoU). Ako skutočne pozitívne sú vyhodnotené tie detekcie, ktoré majú podiel prieniku a zjednotenia anotovaných a predikovaných oblastí väčší než je prahová hodnota (bola nastavená na 0,5). Falošne pozitívne detekcie sú také, ktoré detektor označí za objekt, aj keď sa v tej oblasti žiaden objekt podľa anotácií nenachádza. Prípad keď je v anotáciach objekt, no žiadna z nájdených predikcií s ním nemá prienik, inač povedané nebol vôbec nájdený, sa považuje falošne negatívny.



Obr. 5.1: Grafické znázornenie IoU. Červená označuje anotovanú oblasť a zelená detekovaný oblasť (zdroj [32]).

5.2 Porovnanie s existujúcimi detektormi

Na porovnanie úspešnosti modelov s existujúcimi riešeniami na WIDER Face datasete bol použitý nástroj prevzatý z [8]. Je to neoficiálny evaluačný nástroj kompatibilný s oficiálnymi hodnotiacimi kritériami WIDER Face. Pre hodnotenie tohto datasetu boli vytvorené 3 súbory s anotáciami podľa obtiažnosti obrázkov. Na vyhodnotenie úspešnosti sa používa metrika priemerne presnosti (AP).

Tabuľka 5.1: WIDER Face evaluácia.

	WIDER Face		
Moje modely	**Easy**	**Medium**	**Hard**
ResNet-50	94,4%	93,1%	**85,5**%
ResNet-50 lite	94,8%	93,6%	83,6%
ResNet-152	94,2%	92,8%	83,6%
ResNet-152 lite	**95**%	**93,9**%	83,7%
MobileNetV2	93,4%	91,6%	83%
MobileNetV2 lite	93,7%	92,7%	81,3%
State of the art			
TinaFace	95,8%	95,3%	**92,4**%
RetinaFace	**96,3**%	95,6%	91,4%
AInnoFace	96,5%	**95,7**%	91,2%

V tabuľke 5.1 sú uvedené najlepšie hodnotenia dosihanuté pre každý vytrénovaný model a pre porovnanie 3 najlepšie hodnotené *state of the art* riešenia. Najlepšie hodnotenia v každej kategórii sú zvýraznené hrubým písmom. V ľahkej a strednej kategórii sa moje modely približujú k hodnoteniam *state of the art* modelov. V ťažkej kategórii sú už väčšie odstupy v presnosti 7% až 11%, preto sa bude považovať ťažká kategória za tú najdôležitejšiu. Najlepšie skóre dosiahla plná varianta ResNet-50 modelu, čo celkom prekvapilo, keďže som očakával že najpresnješie budú ResNet-152 varianty, kvôli ich výrazne väčšej veľkosti a počtu parametrov. MobileNet varity pozitívne prekvapili tiež, aj keď ich lite varianta dosiahla najnižšie hodnotenie, nie je výrazne nižšie ako hodnotenie ResNet-152 modelov, pričom MobileNet je výrazne menšia (0,05%) ako ResNet-152 a teda aj výrazne rýchlejšia, čo je preukázané v 5.4.

5.3 Zhodnotenie detekcie prekrytých tvári

Podobný ale upravený evaluačný skript `eval_.py` bol použitý aj na zhodnotenie úspešnosti na extrahovaných prekrytých tvárach z WFLW datasetu. Tuna ale neboli rozdelené anotácie podľa obtiažnosti, takže sa hodnotila iba priemerná presnosť a porovnáva sa iba hodnotenie vlastných modelov, keďže toto nie je štandardný dataset, na ktorom sú ohodnotené existujúce *state of the art* modely.

Vyhodnotenie presnosti modelov na prekrytých tvárach bolo spravené dvakrát. Po prvom primárnom trénovaní na celej WIDER Face dátovej sade pre získanie referencie pre neskoršie porovnanie s presnoťami po jemnom ladení na prekrytých tvárach z WIDER Face dátovej sady. Tento postup bol zvolený pre zhodnotenie úspešnosti jemného ladenia.

Tabuľka 5.2: Porovnanie detekcie prekrytých tvárí pred a po jedmnom ladení na prekrytých tvárach.

Model	**AP Pred**	**AP Po**
ResNet-50	90,9%	90,9%
ResNet-50 lite	85,48%	90,9%
ResNet-152	90,9%	90,9%
ResNet-152 lite	30,3%	90,9%
MobileNetV2	25,97%	45,45%
MobileNetV2 lite	87,6%	90,9%

Z výsledkov sa nedá jasne vyčítať najlepší model, keďže najvyšiu priemernú presnosť 90,9% dosiali viaceré modely. Predpokladám, že toto je zapríčinené prítomnosťou niekoľkých veľmi náročných anotovaných tvárí, ktoré nedokázal žiaden z modelov detekovať. Tento limit sa nepodarilo preraziť ani po opakovanom jemnom ladení na prekrytých tvárach, kedy už začalo dochádzať k pretrénovaniu a presnosť začínala klesať.

ResNet-152 lite a MobileNetV2 varianty dosiahli príliš nízke základné hodnotenia. Predpokladám, že ich nízke skóre je dôsledkom pretrénovania na všeobecnej WIDER Face sade, pretože v jej vyhodnotení dosahujú porovnateľné výsledky s ostatnými modelmi. Jemným ladením sa podarilo dostať ResNet-152 lite až na 90,9% AP, čo je navyššie dosiahnuté vylepšenie po jemnom ladení. MobileNetV2 získal jemným ladením iba zanedbateľné vylepšie avšak pri ďalšom trénovaní už začínala presnosť klesať. Modely ResNet-50, ResNet-152, ktoré boli na 90,9% sa nepodarilo posunúť a so začínajúcim pretrénovaním začínala klesať presnosť. MobileNetV2 lite a ResNet-50 lite boli už po prvom trénovaní blízko stropu. Jemným ladením sa ich podarilo vylepšiť na 90,9% a ďalej podobne ako u predošlých modelov začínala presnosť klesať.

Celkovo sa dá jemné ladenie vyhodnotiť ako úspešné. Modely, ktoré nedosiahli strop 90,9% AP už po prvotnom trénovaní dosiahli vyššiu presnoť na prekrytých tvárach. Zlyhaním ale je, že sa nepodarilo žiadnym spôsobom preraziť 90,9% presnosť na prekrytých tvárach.

5.4 Výkonnosť modelov

Testovanie výkonu tiež prebiehalo na Google Colab výpočetných serveroch, kde sú používané NVIDIA Tesla K80 grafické karty, ktoré majú 4992 NVIDIA CUDA jadier a dosahujú 2.91 teraflopov pri výpočtoch v dvojitou presnosťou. Metrika vyhodnotenia bude počet snímiek za sekundu FPS, pretože modely boli primárne spravené pre dektciu na videu.

Tabuľka 5.3: Výkonnosť na prvom testovacom videu.

Model	**FPS**	**Celkový čas**
ResNet-50	1,17	224s
ResNet-50 lite	1,51	174s
ResNet-152	0,767	344s
ResNet-152 lite	0,96	274s
MobileNetV2	1,60	164s
MobileNetV2 lite	2,08	126s

Výkonnosť modelov bola testovaná použitím modelov na zámerne náročnom videu. Počet tvári na smíku sa pohyboval v rozmedzi 40 až 80. Dĺžka videa trvá 11 sekúnd pri 23 snímkov za sekundu. Dokopy má 264 snímok. Video je v 4K kvalite, konkrétne rozlíšenie má 3840x2160 pixelov. Video bolo natočené na Shibuya ulici v Tokiu, v podmienkach podobných, v akých by mohla byť zasadená kamera pre monitorovanie verejného priestranstva. Video je verejne prístupné a prevzaté z [65].

Obr. 5.2: Snímok z video z ulice Tokia použitého pri meraní výkonu modelov.

Na zmeranie výkonu bolo použité ešte jedno video, ktoré som natočil na webkamere na svojom notebooku. Je na ňom len jedna tvár, ktorá je v pohybe, v rôznych uhlov a aj prekrytá. Video tiež trvá 11s pri 30 snímkach za sekundu a dokopy má 349 snímkov. Rozlíšenie je 640x480 pixelov.

Tabuľka 5.4: Výkon na zázname z webkamery.

Model	**FPS**	**Celkový čas**
ResNet-50	6,02	57s
ResNet-50 lite	8,02	43s
ResNet-152	4,18	83s
ResNet-152 lite	4,99	69s
MobileNetV2	6,5	53s
MobileNetV2 lite	8,58	40s

Na tomto menšom a teda výpočtovo menej náročnom videu, modely už stíhajú spracovať video síce v trhavej ale pozerateľnej rýchlosti. Na záznamy zo CCTV kamerových systémov, ktoré zámerne snímajú v nízkom nastavený snímkov za sekundu, kvôli šetreniu úložiska, by tieto modely aj stačili. Avšak výkonnostne aj v tejto malej kvalite sú modely ďaleko od *real-time* výkonu (aspoň 25 snímkov za sekundu) pre prehrávanie videí. Pričom treba brať do úbahy, že meranie prebiehalo s použitím výkonného hardwaru, ktorý je vysoko nad mainstreamovým štandardom počítačov. Po výkonnostnej stránke modely zlyhali. V prípadoch, kde nie sú nároky na fungovanie v reálnom čase a je možné použiť predspracovanie detektorom a až potom spustenie videa (tak ako je to implementované aj vo finálnom programe) sú tieto modely dostatočné.

Kapitola 6

Záver

Táto práca bola venovaná oblasti detekcie tvárí v obraze so zameraním na nekvalitné videozáznami. V prvej kapitole bol úvod do problematiky kamerových systémov a ich využitia s algoritmami pre rozpoznávanie obrazu. Sú tam opísané ich technické parametre ovplyvňujúce kvalitu záznamov a vysvetlené dôvody, prečo má zmysel používať tieto technológie spolu s algoritmamy strojového učenia.

V následujúcej kapitole sa práca venovala tématike strojového učenia s bližším zameraním na neuronové siete a konvolučné neurónové siete. Boli vysvetlené ich základné princípy, učenie, vrstvy, funkcie a parametre. Následne boli uvedené konkrétne príklady metód využiatia strojového učenia na riešenie detekcie tvári v obraze.

Štvrtá kapitola sa venuje praktickej časti práce a opisuje návrh detektoru tvárí, implementáciu modelov, výber dátová sady, trénovanie modelov a použité technológie. V praktickej časti boli implementované a vytrénované 6 modely na detekciu tvári vychádzajúce z metódy RetinaFace. Ako dátová sada pre trénovanie bola použitá WIDER Face, ktorá obsahuje 32 203 obrázkov. Na implementáciu bol použitý *framework* Keras a trénovanie bolo uskutočnené použitím Google Colab cloudovej služby. Na validáciu bola použitá metrika priemernej presnosti, kde najlepší model dosiahol v ťažkom WIDER Face teste 85,5%. Pre porovnanie aktuálna *state of the art* metóda dosahuje 92,4%. Rozhodol som sa detekciu zamerať najmä na prekryté tváre. Pre zvýšenie presnosti na prekrytých tvárach boli modely ďalej trénované procesom nazývaným „*finetuning*“, kde boli využité iba obrázky obsahujúce prekryté tváre z WIDER Face dátovej sady. Na overenie úspešnosti na prekrytých tvárach bol použitý WFLW dataset, ktorého boli extrahované iba obrázky s prekrytými tvárami. Najlepšie modely dosiali na tomto datasete 90,9% priemernú presnosť.

V rámci praktickej časti bol implementovaný aj program na detekciu tvári. V programe boli použité vybrané modely, ktoré dosahovali najlepšie výsledky na prekrytých tvárach. Program ponúka grafické uživateľské rozhranie vytvorené s PyQt. Je možné v ňom využívať vytrénované modely a porovnávať ich výkonnosť. Umožňuje detekovať na zázname z webkamery, videu a obrázkoch.

Možnosti budúceho vývoja zahŕňajú rozšírenie dátovej sady prekrytých tvári. Optimalizácia výkonnosti modelov zmenami v architektúrach a skúšaním ďalších malých chrbtových sietí, keďže menšie siete sa viac osvedčili. Potencionálne by bolo zaujímavé skúsiť využiť WFLW dátovú sadu na trénovanie, keďže tá obsahuje až 98 tvárových znakov. Zakomponovanie grafovej konvolúcie ako v oficiálnej RetinaFace implementácii pre 3D rekonštrukciu tvárí je ďalšia možnosť vývoju, ktorá by modely práce posunula bližšie k originálnemu modelu a teda možno aj zvýšila presnoť.

Literatúra

[1] *About Keras* [online]. [cit. 11. mája 2021]. Dostupné z: https://keras.io/about/.

[2] *About Python* [online]. [cit. 11. mája 2021]. Dostupné z: https://www.python.org/about/.

[3] Ahn, S. H. *Convolution* [online]. [cit. 11. mája 2021]. Dostupné z: http://www.songho.ca/dsp/convolution/convolution.html#convolution_2d.

[4] Alto, V. *Haar Cascade: Integral Image* [online]. Data Science Chalk Talk, júl 2019 [cit. 11. mája 2021]. Dostupné z: https://datasciencechalktalk.com/2019/07/16/haar-cascade-integral-image/.

[5] Balaji, S. *Overfitting* [online]. Júl 2019 [cit. 11. mája 2021]. Dostupné z: https://medium.com/@cs.sabaribalaji/overfitting-6c1cd9af589.

[6] Bischoff, P. *Why Are CCTV Footages Always So Blurry And Low Quality? We Find Out!* [online]. Comparitech, február 2021 [cit. 11. mája 2021]. Dostupné z: https://www.comparitech.com/blog/vpn-privacy/us-surveillance-camera-statistics/.

[7] Brownlee, J. *A Gentle Introduction to Batch Normalization for Deep Neural Networks* [online]. Machine Learning Mastery, január 2019 [cit. 11. mája 2021]. Dostupné z: https://machinelearningmastery.com/batch-normalization-for-training-of-deep-neural-networks/.

[8] Cheng, T. *WiderFace-Evaluation* [online]. August 2018 [cit. 11. mája 2021]. Dostupné z: https://github.com/wondervictor/WiderFace-Evaluation.

[9] Chollet, F. *Deep learning v jazyku Python*. 1. vyd. GRADA, 2019 [cit. 11. mája 2021]. ISBN 978-80-247-3100-1.

[10] *Define-by-Run* [online]. Chainer, 2015 [cit. 11. mája 2021]. Dostupné z: https://docs.chainer.org/en/stable/guides/define_by_run.html.

[11] Deng, J., Guo, J., Zhou, Y., Yu, J., Kotsia, I. et al. *RetinaFace: Single-stage Dense Face Localisation in the Wild*. 2019.

[12] Deng, L. A tutorial survey of architectures, algorithms, and applications for deep learning. *APSIPA Transactions on Signal and Information Processing*. Cambridge University Press. 2014, zv. 3, s. e2, [cit. 11. mája 2021]. DOI: 10.1017/atsip.2013.9.

[13] Denyer, S. *Beijing bets on facial recognition in a big drive for total surveillance* [online]. The Washington Post, január 2018 [cit. 11. mája 2021]. Dostupné z:

https://www.washingtonpost.com/news/world/wp/2018/01/07/feature/in-china-facial-recognition-is-sharp-end-of-a-drive-for-total-surveillance/.

[14] Desarda, A. *Understanding AdaBoost* [online]. Towards Data Science, január 2019 [cit. 11. mája 2021]. Dostupné z: https://towardsdatascience.com/understanding-adaboost-2f94f22d5bfe.

[15] Donald, F., Donald, C. a Thatcher, A. Work exposure and vigilance decrements in closed circuit television surveillance. *Applied ergonomics* [online]. Marec 2015, zv. 47, s. 220–228, [cit. 11. mája 2021]. DOI: 10.1016/j.apergo.2014.10.001. Dostupné z: https://www.researchgate.net/publication/267335410_Work_exposure_and_vigilance_decrements_in_closed_circuit_television_surveillance.

[16] Driscoll, M. *Jupyter Notebook: An Introduction* [online]. Real Python [cit. 11. mája 2021]. Dostupné z: https://realpython.com/jupyter-notebook-introduction/.

[17] Girshick, R. *Fast R-CNN*. 2015.

[18] Google. *Colaboratory Frequently Asked Questions* [online]. Google Research [cit. 11. mája 2021]. Dostupné z: https://research.google.com/colaboratory/faq.html.

[19] Google. *What is Colaboratory?* [online]. Google Research [cit. 11. mája 2021]. Dostupné z: https://colab.research.google.com/notebooks/intro.ipynb.

[20] Josh Gordon, S. R. *Introducing TensorFlow.js: Machine Learning in Javascript* [online]. Medium, marec 2018 [cit. 11. mája 2021]. Dostupné z: https://medium.com/tensorflow/introducing-tensorflow-js-machine-learning-in-javascript-bf3eab376db.

[21] Guo, Y., Liu, Y., Oerlemans, A., Lao, S., Wu, S. et al. Deep learning for visual understanding: A review. *Neurocomputing*. 2016, zv. 187, s. 27–48.

[22] He, H. *The State of Machine Learning Frameworks in 2019* [online]. The Gradient, október 2019 [cit. 11. mája 2021]. Dostupné z: https://thegradient.pub/state-of-ml-frameworks-2019-pytorch-dominates-research-tensorflow-dominates-industry/.

[23] He, K., Zhang, X., Ren, S. a Sun, J. Deep Residual Learning for Image Recognition. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, s. 770–778. DOI: 10.1109/CVPR.2016.90.

[24] Heureka. *Xiaomi Imilab EC2*. [cit. 11. mája 2021]. Dostupné z: https://ip-kamery.heureka.sk/xiaomi-imilab-ec2/#specifikacia.

[25] Heureka. *Xiaomi Mi Home Security Camera 360°*. [cit. 11. mája 2021]. Dostupné z: https://ip-kamery.heureka.sk/xiaomi-mi-home-security-camera-360-1080p/#specifikacia.

[26] Hotař, V. *Senzorika pro vizualizaci technologické scény* [online]. Technická univerzita v Liberci, 2019 [cit. 11. mája 2021]. Dostupné z: http://www.ksr.tul.cz/ksr/podklady/ARS-8.Kamery-3.pdf.

[27] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W. et al. *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. 2017.

[28] Jiao, L., Zhang, F., Liu, F., Yang, S., Li, L. et al. A Survey of Deep Learning-Based Object Detection. *IEEE Access*. Institute of Electrical and Electronics Engineers (IEEE). 2019, zv. 7, s. 128837–128868. DOI: 10.1109/access.2019.2939201. ISSN 2169-3536. Dostupné z: http://dx.doi.org/10.1109/ACCESS.2019.2939201.

[29] Johnston, R. *Miami-Dade transit using AI to monitor passengers for social distancing* [online]. StateScoop, október 2020 [cit. 11. mája 2021]. Dostupné z: https://statescoop.com/miami-dade-transit-using-ai-to-monitor-passengers-for-social-distancing/.

[30] Kadlec, R. *Recenzia Xiaomi Mi Home Security Camera 360°: Domáca bezpečnostná kamera za skvelú cenu* [online]. TouchIT, júl 2018 [cit. 11. mája 2021]. Dostupné z: https://touchit.sk/recenzia-xiaomi-mi-home-security-camera-360-domaca-bezpecnostna-kamera-za-skvelu-cenu/177499.

[31] Keegan, M. *The Most Surveilled Cities in the World* [online]. U.S. News, august 2020 [cit. 11. mája 2021]. Dostupné z: https://www.usnews.com/news/cities/articles/2020-08-14/the-top-10-most-surveilled-cities-in-the-world.

[32] Khandelwal, R. *Evaluating performance of an object detection model* [online]. Towards Data Science, január 2020 [cit. 11. mája 2021]. Dostupné z: https://towardsdatascience.com/evaluating-performance-of-an-object-detection-model-137a349c517b.

[33] Kua, R. *Why Are CCTV Footages Always So Blurry And Low Quality? We Find Out!* [online]. Rojak Daily, marec 2017 [cit. 11. mája 2021]. Dostupné z: https://rojakdaily.com/news/article/2053/why-are-cctv-footages-always-so-blurry-and-low-quality-we-find-out.

[34] Kumawat, D. *7 Types of Activation Functions in Neural Network* [online]. Analytics Steps Infomedia, august 2019 [cit. 11. mája 2021]. Dostupné z: https://www.analyticssteps.com/blogs/7-types-activation-functions-neural-network.

[35] Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B. et al. *Feature Pyramid Networks for Object Detection*. 2017.

[36] Marciniak, T., Chmielewska, A., Weychan, R., Parzych, M. a Dabrowski, A. Influence of low resolution of images on reliability of face detection and recognition. *Multimedia Tools and Applications*. Jún 2013, zv. 74. DOI: 10.1007/s11042-013-1568-8.

[37] McCulloch, W. a Pitts, W. A Logical Calculus of the Idea Immanent in Nervous Activity. *Bulletin of Mathematical Biology*. December 1943, zv. 5, s. 115–133. DOI: 10.1007/BF02478259.

[38] Mehlig, B. *Artificial Neural Networks*. Január 2019.

[39] Mesquita, D. *How to use NVIDIA GPUs for Machine Learning with the new Data Science PC from Maingear* [online]. Towards Data Science, október 2019 [cit. 11. mája 2021]. Dostupné z: https://towardsdatascience.com/how-to-use-gpus-for-machine-learning-with-the-new-nvidia-data-science-workstation-64ef37460fa0.

[40] METACENTRUM. *Metacentrum Wiki - Průvodce pro začátečníky* [online]. Marec 2021 [cit. 11. mája 2021]. Dostupné z: https://wiki.metacentrum.cz/wiki/Pruvodce_pro_zacatecniky.

[41] MOLNÁR, K. *Úvod do problematiky umělých neuronových sítí* [online]. Elektrorevue [cit. 11. mája 2021]. Dostupné z: http://www.elektrorevue.cz/clanky/00013/index.html.

[42] MURÁŇ, J. *Úvod do konvolučných neurónových sietí* [online]. Data Science Chalk Talk, november 2019 [cit. 11. mája 2021]. Dostupné z: https://umelainteligencia.sk/uvod-do-konvolucnych-neuronovych-sieti/.

[43] NAJIBI, M., SAMANGOUEI, P., CHELLAPPA, R. a DAVIS, L. *SSH: Single Stage Headless Face Detector*. 2017.

[44] NASH, J. *Global sales of video surveillance equipment projected to surpass $20 billion this year* [online]. Biometric Update, január 2020 [cit. 11. mája 2021]. Dostupné z: https://www.biometricupdate.com/202001/global-sales-of-video-surveillance-equipment-projected-to-surpass-20-billion-this-year.

[45] A, N. *Camera Tuning : Understanding the Image Signal Processor and ISP Tuning* [online]. PathPartner Technology, marec 2020 [cit. 11. mája 2021]. Dostupné z: https://www.biometricupdate.com/202001/global-sales-of-video-surveillance-equipment-projected-to-surpass-20-billion-this-year.

[46] NEVES, A. C., GONZALEZ, I., LEANDER, J. a KAROUMI, R. *A New Approach to Damage Detection in Bridges Using Machine Learning*. Január 2018. 73-84 s. ISBN 978-3-319-67442-1.

[47] NÄSHOLM, E., ROHLFING, S. a SAUER, J. D. Pirate Stealth or Inattentional Blindness? The Effects of Target Relevance and Sustained Attention on Security Monitoring for Experienced and Naïve Operators. *PLOS ONE* [online]. Public Library of Science. Január 2014, zv. 9, č. 1, s. 1–8, [cit. 11. mája 2021]. DOI: 10.1371/journal.pone.0086157. Dostupné z: https://doi.org/10.1371/journal.pone.0086157.

[48] *OpenCV About* [online]. OpenCV team, 2021 [cit. 11. mája 2021]. Dostupné z: https://opencv.org/about/.

[49] *Papers With Code* [online]. [cit. 11. mája 2021]. Dostupné z: https://paperswithcode.com/task/face-detection.

[50] LUDWIG, S. *Pro's and Cons for IP vs. Analog Video Surveillance* [online]. Securiy Magazine, apríl 2018 [cit. 11. mája 2021]. Dostupné z: https://www.securitymagazine.com/articles/88854-pros-and-cons-for-ip-vs-analog-video-surveillance.

[51] REEVELL, P. *How Russia is using facial recognition to police its coronavirus lockdown* [online]. ABC News, apríl 2020 [cit. 11. mája 2021]. Dostupné z: https://abcnews.go.com/International/russia-facial-recognition-police-coronavirus-lockdown/story?id=70299736.

[52] Huang, K.-Y. *RetinaFace Unofficial Implementation* [online]. Jún 2020 [cit. 11. mája 2021]. Dostupné z: https://github.com/peteryuX/retinaface-tf2.

[53] Ruder, S. *An overview of gradient descent optimization algorithms*. 2017.

[54] Sarvepalli, S. K. Deep Learning in Neural Networks: The science behind an Artificial Brain. Október 2015, [cit. 11. mája 2021]. DOI: 10.13140/RG.2.2.22512.71682.

[55] Senan, M. F. E. M., Abdullah, S. N. H. S., Kharudin, W. M. a Saupi, N. A. M. CCTV quality assessment for forensics facial recognition analysis. In: *2017 7th International Conference on Cloud Computing, Data Science Engineering - Confluence*. 2017, s. 649–655 [cit. 11. mája 2021]. DOI: 10.1109/CONFLUENCE.2017.7943232.

[56] Sharma, P. *A Step-by-Step Introduction to the Basic Object Detection Algorithms (Part 1)* [online]. Analytics Vidhya, október 2018 [cit. 11. mája 2021]. Dostupné z: https://www.analyticsvidhya.com/blog/2018/10/a-step-by-step-introduction-to-the-basic-object-detection-algorithms-part-1/.

[57] Sonka, M., Hlavac, V. a Boyle, R. *Image Processing, Analysis, andMachine Vision*. 4. vyd. Cengage Learning, 2015. ISBN 978-1-133-59360-7.

[58] SourceSecurity. *Openview Security Solutions installs control room for Lancashire Hub CCTV Centralisation Project* [online]. SourceSecurity [cit. 11. mája 2021]. Dostupné z: https://www.sourcesecurity.com/news/openview-security-solutions-installs-control-room-lancashire-hub-cctv-centralisation-project-co-3425-ga-co-12068-ga.23705.html?utm_source=SIc&utm_medium=Redirect&utm_campaign=Int%20Redirect%20Popup.

[59] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. a Salakhutdinov, R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *J. Mach. Learn. Res.* JMLR.org. január 2014, zv. 15, č. 1, s. 1929–1958. ISSN 1532-4435.

[60] Szabłowski, B. *What is an artificial neuron and why does it need an activation function?* [online]. Towards Data Science, október 2020 [cit. 11. mája 2021]. Dostupné z: https://towardsdatascience.com/what-is-an-artificial-neuron-and-why-does-it-need-an-activation-function-5b4c1e971d80.

[61] Tang, X., Du, D. K., He, Z. a Liu, J. *PyramidBox: A Context-assisted Single Shot Face Detector*. 2018.

[62] *CCD vs. CMOS* [online]. Teledyne DALSA [cit. 11. mája 2021]. Dostupné z: http://www.teledynedalsa.com/en/learn/knowledge-center/ccd-vs-cmos/.

[63] *TensorFlow 2 quickstart for beginners* [online]. [cit. 11. mája 2021]. Dostupné z: https://www.tensorflow.org/tutorials/quickstart/beginner.

[64] *TensorFlow 2.0 is now available!* [online]. TensorFlow Blog, september 2019 [cit. 11. mája 2021]. Dostupné z: https://blog.tensorflow.org/2019/09/tensorflow-20-is-now-available.html.

[65] Videezy. *Tokyo Japan - Shibuya area video* [online]. Videezy [cit. 11. mája 2021]. Dostupné z: https://www.videezy.com/abstract/40906-tokyo-japan-shibuya-area.

[66] Viola, P. a Jones, M. J. Robust Real-Time Face Detection. *Int. J. Comput. Vision.* USA: Kluwer Academic Publishers. máj 2004, zv. 57, č. 2, s. 137–154. DOI: 10.1023/B:VISI.0000013087.49260.fb. ISSN 0920-5691. Dostupné z: https://doi.org/10.1023/B:VISI.0000013087.49260.fb.

[67] Warm, J., Parasuraman, R. a Matthews, G. Vigilance Requires Hard Mental Work and Is Stressful. *Human factors* [online]. Júl 2008, zv. 50, s. 433–41, [cit. 11. mája 2021]. DOI: 10.1518/001872008X312152. Dostupné z: https://www.researchgate.net/publication/23157806_Vigilance_Requires_Hard_Mental_Work_and_Is_Stressful.

[68] *What are the Basic Camera Parts and How Do Cameras Work?* [online]. PhotographyTalk [cit. 11. mája 2021]. Dostupné z: https://www.photographytalk.com/digital-camera-parts.

[69] *What is PyQt?* [online]. Riverbank Computing [cit. 11. mája 2021]. Dostupné z: https://riverbankcomputing.com/software/pyqt.

[70] Wikipedia. *Konvoluce* [online]. Wikipedia [cit. 11. mája 2021]. Dostupné z: https://cs.wikipedia.org/wiki/Konvoluce.

[71] Wu, W., Qian, C., Yang, S., Wang, Q., Cai, Y. et al. Look at Boundary: A Boundary-Aware Face Alignment Algorithm. In: *CVPR*. 2018.

[72] Yang, S., Luo, P., Loy, C. C. a Tang, X. WIDER FACE: A Face Detection Benchmark. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, s. 5525–5533. DOI: 10.1109/CVPR.2016.596.

[73] Zhu, Y., Cai, H., Zhang, S., Wang, C. a Xiong, Y. *TinaFace: Strong but Simple Baseline for Face Detection*. 2020.

Príloha A

Obsah přiloženého paměťového média

Obsah hlavného pracovného priečinku je rozdelený na 2 DVD z kapacitných dôvodov. Pri používaní je potrebné zlúčiť `data` priečinky z obocch diskov do jedného.
DVD 1:

- `configs` - priečinok obsahujúci konfiguračné súbory
- `data` - priečinok obsahujúci datátové sady a k nim príslušné skripty a z nich vytvorené `TfRecord` záznamy, na tomto disku je len WFLW dataset
- `checkpoints` - priečinok obsahujúci finálne uložené váhy modelov
- `icons` - priečinok obsahuje ikonky použité v programe s GUI
- `jupyter_notebooks` - priečinok obsahujúci skripty, ktoré boli používané na Google Colab k trénovaniu a evaluácii
- `logs` - priečinok obsahujúci logy z trénovania
- `models` - priečinok obsahujúci konfiguračné súbory
- `performance_testing` - priečinok obsahuje videa a skript, ktoré boli použité na vyhodnotenie výkonu modelov
- `widerface_evaluate` - priečinok obsahujúci súbory slúžiace na vyhodnotenie presnosti modelov
- `app_img_list.txt` - textový súbor obsahujúci anotácie obrázkov použitých v programe s GUI v galérii prekrytých tvárí
- `dataset.py` - súbor implementujúci funkcie používané pri načítaní dát
- `face_detect_app.py` - súbor finálneho programu, implementuje GUI, logiku a spúšťa program
- `fine_tune_wider_occ_all.py` - implementácia jemného ladenia modelov
- `save_weights.py` - skript slúžiaci na uloženie posledných váh modelu

- `test_widerface.py` - skript slúžiaci na spracovanie WIDER Face testovacej sady modelom pre jej následovnú evaluáciu
- `train.py` - implementácia primárneho trénovania modelov
- `utils.py` - súbor implementujúci rôzne užitočné funkcie používané vo viacerých iných skriptoch, preto je umiestnený v hlavnom priečinku
- `manual.pdf` - manuál popisujúci postup spúšťania a používania programu, trénovania a testovania modelov
- `README` - základná dokumentácia poskytujúca informácie programe, použité knižnice, verzie, inštálciu atď

DVD 2:

`data` - priečinok obsahujúci datátové sady a k nim príslušné skripty a z nich vytvorené *TfRecord* záznamy, na tomto disku je WIDER Face dataset