VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

SLEDOVÁNÍ OBLIČEJE V REÁLNÉM ČASE

REAL-TIME FACE TRACKING

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — ALEKSEI ERMAK
AUTHOR

VEDOUCÍ PRÁCE — Ing. MICHAL HRADIŠ, Ph.D.
SUPERVISOR

BRNO 2019

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

Zadání bakalářské práce

22023

Student: **Ermak Aleksei**
Program: Informační technologie
Název: **Sledování obličeje v reálném čase**
Real-Time Face Tracking
Kategorie: Zpracování obrazu
Zadání:

1. Prostudujte základy konvolučních neuronových sítí a jejich učení.
2. Vytvořte si přehled o současných metodách detekce a sledování objektů.
3. Vyberte konkrétní metody využitelné pro velmi efektivní detekci a sledování obličejů ve videu.
4. Obstarejte si databázi vhodnou pro experimenty.
5. Implementujte navrženou metodu a proveďte experimenty nad datovou sadou.
6. Porovnejte dosažené výsledky a diskutujte možnosti budoucího vývoje.
7. Vytvořte stručný plakát prezentující vaši práci, její cíle a výsledky.

Literatura:

- Zhang et al.: Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks. IEEE Signal Processing Letters, 2016.
- Huang et al.: Speed/Accuracy Trade-Offs for Modern Convolutional Object Detectors. CVPR 2017.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3.

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

Vedoucí práce: **Hradiš Michal, Ing., Ph.D.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 15. května 2019
Datum schválení: 1. listopadu 2018

Abstrakt

Práce se věnuje problematice sledování obličeje v reálném čase. Na začátku jsou popsány existující metody sledování objektů a detekce obličejů. V další části se tato práce věnuje návrhu, implementaci a trénování konvoluční neuronové sítě, která se ukázala efektivní pro řešení problematiky sledování obličejů. Dále se implementovaná síť porovnává s existujícími metodami. Poslední část práce popisuje optimalizaci vytvořené sítě pomocí sady nástrojů společnosti Intel zvané OpenVINO.

Abstract

This bachelor thesis focuses on the issue of face tracking in real time. In the beginning, this work describes the existing methods of object tracking and face detection. The following part of this thesis concentrates on the design, implementation and testing of the convolutional neural network, which was proved as the effective solution for the face tracking issue. In addition to this, the implemented network is compared to those existing methods. The last part of the thesis describes the optimization of the designed network using OpenVINO toolkit provided by Intel.

Klíčová slova

sledování obličeje, detekce obličeje, sledování objektů, konvoluční neuronové sítě

Keywords

face tracking, face detection, object tracking, convolutional neural networks

Citace

ERMAK, Aleksei. *Sledování obličeje v reálném čase.* Brno, 2019. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Michal Hradiš, Ph.D.

Sledování obličeje v reálném čase

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Michala Hradiše, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

. . . . . .
Aleksei Ermak
14. května 2019

Poděkování

Tímto bych chtěl velmi poděkovat svému vedoucímu práce Ing. Michalu Hradišovi, Ph.D. za cenné rady, pomoc, ochotu a čas, který věnoval mi při odborném vedení této práce. Dále bych chtěl poděkovat Silvie Mišákové za obrovskou pomoc při gramatické kontrole práce a rodičům a přátelům za morální podporu.

Obsah

1 Úvod **2**

2 Detekce a sledování objektů **3**
2.1 Problematika sledování . . . 3
2.2 Klasifikace existujících metod sledování objektů . . . 4
2.3 Sledovací algoritmy založené na korelačních filtrech . . . 5
2.4 Sledovací algoritmy založené na hlubokém učení . . . 7
2.5 Detektory obličejů . . . 11

3 Návrh a implementace sledovací sítě **14**
3.1 Sledovací neuronová síť . . . 14
3.2 Vytvoření trénovacího datasetu . . . 17
3.3 Učení modelu . . . 18
3.4 Vývojové nástroje . . . 20

4 Testování a porovnání **22**
4.1 Příprava testovacích dat . . . 22
4.2 Průběh testování a kalibrace výstupů . . . 22
4.3 Metriky pro porovnání metod . . . 23
4.4 Analýza výsledků . . . 25

5 Optimalizace sítě a její vyhodnocení **28**
5.1 Optimalizace pomocí OpenVINO . . . 28
5.2 Kvantizace neuronové sítě . . . 30
5.3 Vyhodnocení optimalizované sítě . . . 31

6 Závěr **33**

Literatura **34**

A Obsah přiloženého paměťového média **42**

Kapitola 1

Úvod

Úlohy zpracování obrazů a získávání užitečných dat z nich jsou v moderním světě aktuálnější, než nikdy předtím. Detekce, klasifikace a sledování pohybujících se objektů jsou jen některé z úkolů, které se řeší v současné době. V posledních letech, díky rychlému rozvoji v oblasti strojového učení, pokročilým výpočetním technologiím a práce mnoha talentovaných odborníků, byl ve všech těchto úkolech dosažen obrovský pokrok. I přesto stále existuje velké množství problémů a jeden z nich se řeší právě v tomto díle.

Tato práce se zabývá detekcí a sledováním lidských obličejů v reálném čase. Praktickou částí je implementace konvoluční neuronové sítě, která dokáže sledovat lidské obličeje v sekvenci obrázků. Implementovaná síť funguje ve vysoké rychlosti a je schopná běžet na zařízeních s nízkým výkonem.

Další kapitola je věnována problematice sledování. Tato kapitola také popisuje principy činností algoritmů pro sledování objektů, anglicky zvaných *trackery* a přehled existujících metod detekce obličejů. Kapitola č. 3 popisuje návrh, implementaci a trénování konvoluční neuronové sítě pro sledování obličejů v reálném čase. Zde je také popsáno vytvoření trénovacího datasetu a nástroje, které byly použity při implementaci. Čtvrtá kapitola obsahuje popis testování vytvořené sítě a porovnání s existujícími algoritmy sledování. V této kapitole čtenář najde popis testovacího datasetu a metrik, pomocí kterých byla sledovací síť otestována. Poslední kapitola je věnována optimalizaci a zrychlení vytvořené neuronové sítě pomocí sady nástrojů společnosti Intel zvané OpenVINO.

Kapitola 2

Detekce a sledování objektů

V této kapitole se popisuje problém, který se řeší v dané práci, a to sledování objektů. Zde je popsáno, co to je sledování objektů a proč se zabývat problémem sledování. V rámci teto kapitoly jsou zmíněny metody vizuálního sledování objektů existující v současném světe a jejich klasifikace. Na konci kapitoly jsou popsány aktuální řešení v oblasti detekce obličejů.

2.1 Problematika sledování

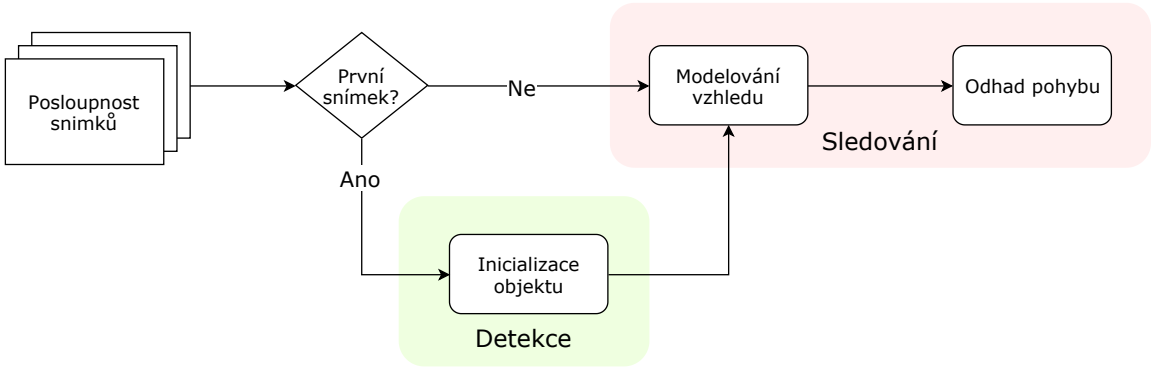
Detekce objektů je jednou z úloh počítačového vidění, která se zabývá lokalizací polohy objektů v obraze. V současném světe existuje obrovské množství případů, kde se využívá detekce objektů: detekce chodců [4], značek [70] a jiných detailů prostředí u samořídicích aut, detekce lidí u bezpečnostních systémů [69], detekce nemocí u rostlin [54] a na medicínských skenech [12] a mnoho dalších. Některé úlohy, jako například dohled nad pohybem osob pomocí bezpečnostních systémů nebo sledování silničního provozu, vyžadují efektivní a rychlou detekci v reálném čase. Protože detekce objektů je časově a výpočetně náročná operace vzhledem k velkému množství dat, která obsahují videa, práce v reálném čase musí běžet na výkonných zařízeních. Řešením úloh takového typu je sledování objektů.

Vizuální sledování objektů [64] je další úloha z oblasti počítačového vidění. Sledovací algoritmy jsou často postaveny na spolupráci s detekcí a klasifikací a navrženy tak, aby extrahovaly informace o objektu (nebo více objektech) a na jejích základě prováděly detekci v dalších snímcích. Algoritmy sledování objektů se aplikují u monitorování dopravního toku, bezpečnostních a zabezpečovacích systémech, při rozpoznávání aktivity, autonomních vozidlech a v rozhraních člověk-stroj.

Obtížnost vizuálního sledování spočívá v problémech jako jsou okluze, variace osvětlení, nízké rozlišení, deformace a otáčení cíle sledování, reidentifikace cíle, rychlý nebo rozmazaný pohyb a sledování cíle v přítomnosti hluků.

Jednou z výhod sledování oproti detekci je skutečnost, že nemusíme aplikovat časově a výpočetně náročnou detekci v každém snímků. Sledování objektů pomocí detekce v každém snímku je možné, avšak podobné sledování musí běžet na vysokovýkonných strojích, aby splňovalo jeden z hlavních požadavků - rychlost.

Typický proces sledování objektů se skládá ze čtyři modulů [47, 20]: inicializace cíle sledovaní (object initialization), modelování vzhledu objektu (appearance modelling), odhad pohybu (motion estimation) a lokalizace objektu (object localization). Proces může být



Obrázek 2.1: Zjednodušený proces sledování objektů principem *tracking-by-detection*.

zjednodušen omezením na etapy: inicializace objektu, modelování jeho vzhledu a odhad jeho pohybu.

Inicializace objektu je proces lokalizace polohy objektu v prvním snímku s některou z následujících reprezentací: kostra objektu, ohraničující rámeček (ohraničení, *bounding box*), elipsa, centroid nebo silueta objektu. Inicializace polohy objektu může být zadána manuálně (člověkem) nebo automaticky, pomocí detektoru objektů. Modelování vzhledu objektu se skládá z identifikace vizuálních objektových vlastností pro lepší reprezentaci zájmové oblasti a efektivity konstrukce matematických modelů pro detekci objektů pomocí technik učení. Odhad pohybu je proces odhadu pozice objektu v následujícím snímku.

Obecně sledování funguje tak, že pomocí detektoru nebo manuálně definujeme prvotní polohu objektu v prvním snímku. V následujících snímcích polohu objektu a jeho pohyb určuje sledovací algoritmus. Podobný princip sledování objektů se anglicky nazývá *tracking-by-detection*.

Tato práce se zabývá detekcí a sledováním obličejů v reálném čase a pro řešení daného problému je potřeba mít přehled o současném stavu rozvoje v této oblasti. Proto je zbytek kapitoly věnován existujícím metodám sledování objektů a detektorům obličejů, které mohou být potenciálně využity v řešení problematiky této práce.

2.2 Klasifikace existujících metod sledování objektů

V literatuře [20] je možné se setkat s rozdelením sledovacích algoritmů podle jejích obecných vlastností na *single-object* a *mutli-object* algoritmy, generativní a diskriminativní algoritmy, online a offline algoritmy, *online-learning* a *offline-learning* algoritmy.

Single-object algoritmy umí sledovat jenom jeden objekt ve scéně. Multi-object algoritmy jsou schopné současně sledovat víc objektů. Generativní a diskriminativní algoritmy [75] si „pamatují“ charakteristické rysy objektů, ale liší se podle principů vyhledávání objektů ve snímcích. Generativní algoritmy hledají nejvhodnější regiony se sledovanými cíli z množiny regionů. Diskriminativní se snaží odlišit cílové objekty od pozadí.

Online algoritmy [53] zpracovávají video záznamy snímek po snímku a pro sledování využívají informace získané z předchozích snímků. Offline algoritmy mají k dispozici celé video a mohou využívat informace jak z předchozích, tak i z následujících snímků. Vzhledem k problému sledování v reálném čase, offline algoritmy nejsou vhodné pro řešení problematiky teto práce.

Offline learning algoritmy [79] se učí před použitím v praxi. Obvykle se jedna o algoritmus naučeny sledovat specificky objekt. Online-learning algoritmy se učí v průběhu sledování a „pamatuji si“ jak vypadá objekt. Online-learning algoritmy jsou univerzální a používají se pro sledování libovolných objektů, avšak učení v průběhu sledování vyžaduje doplňkové výpočetní náklady. V této práce se řeší úloha sledování obličejů, konkretního objektu, proto je vhodné využit offline-learning sledovací algoritmus.

Podle způsobu zpracovávání a reprezentace dat, modelovaní vzhledu a odhadu pohybu objektů se mohou sledovací algoritmy dále dělit na velký počet tříd. Mezi starší významnější algoritmy sledování patří například Kalmanův filtr [41], algoritmy využívající optický tok [51], částicové filtry [1], a slabé klasifikátory [62].

Algoritmus **Medianflow** [39] měří trajektorie pohybu objektů mezi předchozími a aktuálními záznamy. Obraz je rozdělen na malé oblasti, pro které je použit algoritmus vyhledávání optického toku. Medián posunutí všech bodů objektu je považován za posunutí jeho středu. Tento algoritmus dobře určuje selhání procesu sledování a funguje v případech hladkého pohybu objektu. Špatně zpracovává případy úplného a parciálního zakrytí objektu.

Diskriminativní algoritmus **Boosting** [23] vytváří slabé klasifikátory, které používají Haarovy příznaky. Klasifikátor se učí v reálném čase na základě oblasti určené uživatelem (nebo detektorem). Celá vybraná oblast je považována za pozitivní vzorek a oblastí kolem pozitivního vzorku jsou považovány za negativní. V dalším snímku klasifikátor zpracovává všechny pixely v oblasti předchozí pozice sledovaného objektu a přiděluje jim váhy. Oblast s největší vahou je pak novou pozicí objektu. Tato oblast se stává dalším pozitivním vzorkem a oblasti kolem jsou přidány do seznamu negativních a algoritmus tak neustále pokračuje v učení.

Dobrý výkon vykazuje algoritmus **TLD** (*Tracking-Learning-Detection*) [40]. Přístup použitý v tomto algoritmu se nazývá Sledování-učení-detekce a skládá se ze tří modulů: krátkodobého sledování, učení a detekce objektů. Sledovací modul provádí sledování objektů ve snímcích. Detekční modul lokalizuje polohu objektu, který byl sledován poslední dobu a v případě potřeby upravuje sledovací modul. Modul učení vyhodnocuje chyby detekčního modulu a aktualizuje ho tak, aby se jim v budoucnu vyhnul. Tento algoritmus je výkonný a odolný vůči částečnému a úplnému zmizení objektu, stejně jako k různým změnám vzhledu objektu.

V současné době největší výkon vykazují sledovací algoritmy používající korelační filtry (*correlation filter trackers*) a modely hlubokého učení (*deep trackers*). Toto rozdělení se popisuje například v [49], a tímto algoritmům budou věnovaný následující dvě podkapitoly.

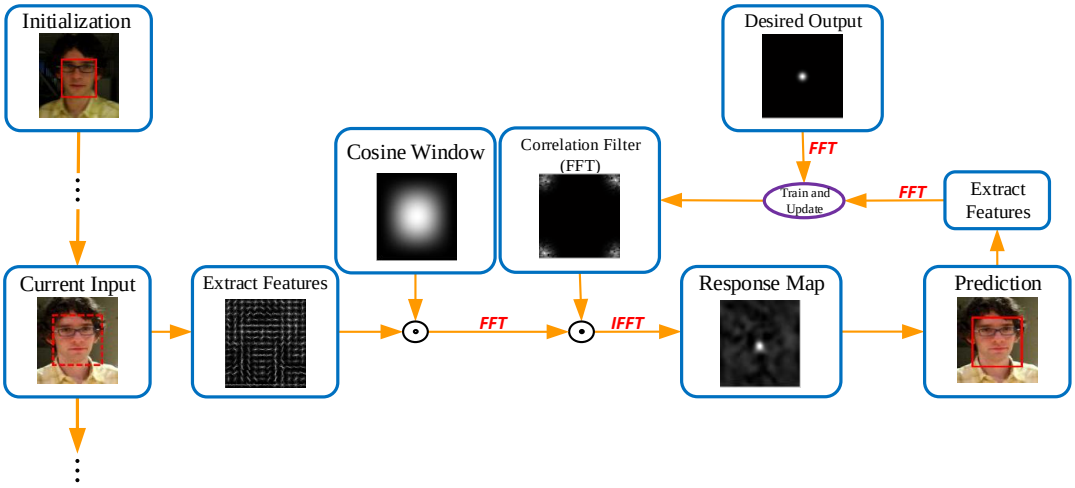
2.3 Sledovací algoritmy založené na korelačních filtrech

Princip sledování pomocí algoritmů založených na korelačních filtrech [7] je možné formulovat jako hledání korelace mezi úspěšnými snímky. To znamená, že umístění objektu v aktuálním snímku je oblast s maximální korelací se sledovaným objektem v předchozím snímku. Pro modelování vzhledu objektu tyto algoritmy používají korelační filtry, které se trénují na vzorcích. Cíl na začátku sledování je vybrán pomocí malého sledovacího okna v prvním snímku a od tohoto momentu sledování cíle a trénování filtrů fungují společně. Dále je cíl sledován korelací filtrů se sledovacím oknem v dalším snímku. Umístění odpovídající maximální výstupní korelaci označuje novou pozici cíle, po čemž se aktualizují korelační filtry.

Hlavní výhodou algoritmů založených na korelačních filtrech je efektivita výpočtů. Důvodem je to, že výpočet může být efektivně prováděn ve Fourierove doméně, což umožňuje výrazné zrychlení celého výpočtu díky možnosti využít rychlou Fourierovu transformaci (*Fast Fourier Transform, FFT*). Na začátku se počítá Fourierova transformace vstupního obrázku $F = \mathcal{F}(f)$ a korelačního filtru $H = \mathcal{F}(h)$ ve dvourozměrném prostoru (2D). Konvoluční věta uvádí, že korelace se stává *element-wise násobením* (*Hadamardův součin*) ve Fourierově doméně. Po označení element-wise násobení symbolem $\odot$ a komplexní konjugaci symbolem $*$, operace korelace má tvar:

$$G = F \odot H^* \tag{2.1}$$

Korelační výstup se transformuje zpět do prostorové domény pomocí inverzní rychlé Fourierově transformace (*Inverse Fast Fourier Transform, IFFT*). Úzkým hrdlem ve výpočtu dopředné a inverzní rychlé Fourierově transformace je, že celý proces má časovou složitost $O(P \log P)$, kde P je počet pixelů ve sledovacím okně.



Obrázek 2.2: Ilustrace procesu sledování pomocí algoritmů založených na korelačních filtrech. Na každém snímku se po inicializaci obrazová data v předchozí odhadované poloze ořezují jako aktuální vstup. Následně mohou být extrahovány vizuální příznaky pro lepší popis vstupu a je obvykle použito kosinové okno pro vyhlazení hranic vstupního obrázku. Pak se korelace mezi aktuálním vstupem a naučeným filtrem počítá ve frekvenční doméně na základě konvoluční věty. Symbol $\odot$ na obrázku označuje element-wise násobení a FFT znamená rychlou Fourierovu transformaci. Po korelaci se získá mapa prostorové spolehlivosti (angl. *spatial confidence map*) inverzní FFT (IFFT), jejíž pík lze předpovědět jako novou pozici cíle. Konečný vzhled v nově odhadnuté pozici je extrahován pro trénování a aktualizaci korelačního filtru. Obrázek a popis jsou převzatý z [8].

Třída algoritmů založených na korelačních filtrech obsahuje mnoho různých implementací sledovacích metod, mezi nimiž jsou takové významné algoritmy jako MIL, KCF, MOSSE, DCF-CSR(CSRT).

Sledovací algoritmus **MIL** (*Multiple Instance Learning*) [2] používá učení na pozitivních vzorcích, kterými jsou prvotní poloha sledovaného objektu a jeho blízké okolí. Jako negativní vzorky je bráno okolí příliš vzdálené od objektu. Algoritmus porovnává každý další snímek

s předchozím a určuje nejvhodnější posunutí sledovacího okna. Algoritmus sleduje objekty i při parciálním zakrytí, ale není stabilní při rychlém pohybu objektu.

Podobně jako MIL, **KCF** tracker (*Kernelized Correlation Filter*) [27] používá množství pozitivních vzorků, ale ve větším počtu a s menší vzdáleností od původní polohy sledovaného objektu. Získaná data se liší velmi málo, a proto je možné pro ně použít matematické metody, což nakonec snižuje výpočetní složitost algoritmu a zvyšuje přesnost sledování díky velkému počtu vzorků. Algoritmus hledá nejvhodnější filtr pro korelaci, který by minimalizoval chybu pro odhad následující pozice objektu. KCF je rychlý algoritmus, ale není schopný sledovat rychle pohybující se objekty.

Algoritmus **MOSSE** (*Minimum Output Sum of Squared Error*) [7] používá adaptivní korelaci pro sledování objektů a produkuje stabilní korelační filtry při inicializaci pomocí jediného snímku. MOSSE tracker je odolný vůči změnám osvětlení, škálování a deformacím vzhledu objektu. MOSSE tracker také pracuje s vyšším FPS (450 a více). Algoritmus je snadno implementovatelný a velmi rychlý, avšak není přesný jako ostatní algoritmy.

DCF-CSR algoritmus (*Discriminative Correlation Filter with Channel and Spatial Reliability*, CSRT) [52] používá mapy prostorové spolehlivosti (angl. spatial reliability map) pro úpravu a podporu korelačních filtrů aplikovaných na sledovanou oblast. Toto zajišťuje lepší sledování objektů ve vybrané oblasti libovolné velikosti. Algoritmus pracuje při poměrně nižším FPS (25 FPS), ale poskytuje vyšší přesnost sledování objektů.

2.4 Sledovací algoritmy založené na hlubokém učení

Deep learning-based trackery [86], neboli trackery založené na hlubokém učení, využívají možnosti neuronových sítí pro vizuální sledování objektů.

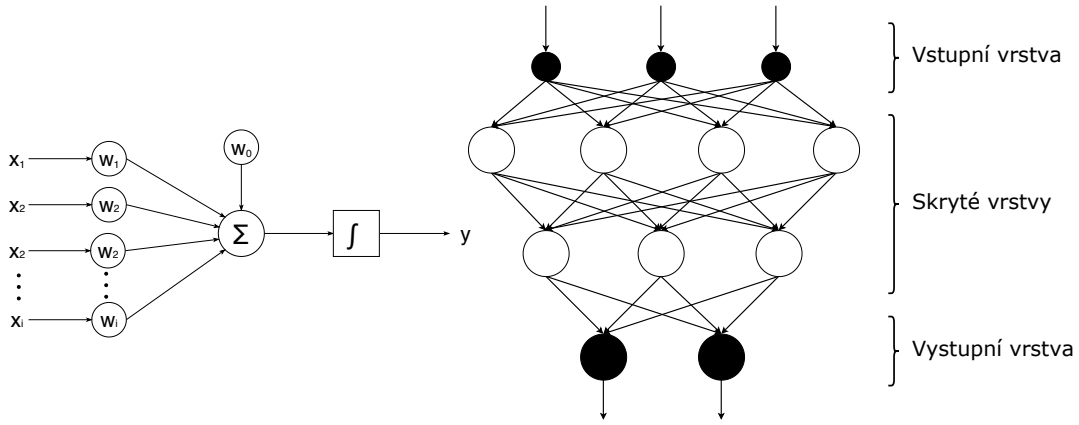
Neuronové sítě

Neuronová síť je matematický model, který má strukturu a funguje na principu sítí nervových buněk biologického organismu [18]. Díky tomuto má model schopnost analyzovat, a dokonce zapamatovat si různé informace. Základní prvek sítě – uměly neuron, nebo perceptron je definován váhami w_i, prahovou hodnotou w_0 a aktivační funkcí f.

$$y = f\left(\sum_{i=1}^{n} x_i w_i + w_0\right) \tag{2.2}$$

Váhy představují vektor hodnot, kterým přenásobíme vektor vstupních hodnot x_i. Na základě těchto vah mohou být vstupní hodnoty posilovány nebo potlačeny. Po vynásobení váhami se vstupy sečtou a k tomu součtu se přičte prahová hodnota, též zvána biasem. Tento součet definuje potenciál neuronu, na který se aplikuje aktivační funkce. Funkce aktivace je tady způsob, jak normalizovat vstupní data a omezit maximální odchylku výstupu neuronu [59]. To znamená, že pokud máme na vstupu velké číslo, předáním funkce aktivace získáme výstup v menším rozsahu, který potřebujeme. Obvykle jde o sigmoidální funkci, hyperbolicky tangens nebo ReLU (*Rectified Linear Unit*). Výsledek funkce je pak výstupem celého neuronu.

Sám o sobě je umělý neuron jenom slabá základní jednotka, která není schopná vykonávat složité operace a dosahovat potřebných výsledků. Proto jsou neurony spojeny do vrstev a tím vytváří vícevrstvé neuronové sítě [60]. Obecné vícevrstvé neuronové sítě se skládají ze tří typů vrstev: vstupních, skrytých a výstupních. Externí vstupní signály jsou přivedeny



Obrázek 2.3: Model umělého neuronu (zleva) a příklad architektury vícevrstvé neuronové sítě (zprava).

na vstupy neuronů vstupní vrstvy a výstupy jsou výstupní signály poslední vrstvy. Vstupní vrstva nedělá žádné výpočty, ale distribuuje přicházející signály na další vrstvy. Neurony jedné vrstvy nejsou navzájem spojeny, ale každý neuron jedné vrstvy je spojen s každým neuronem sousední vrstvy.

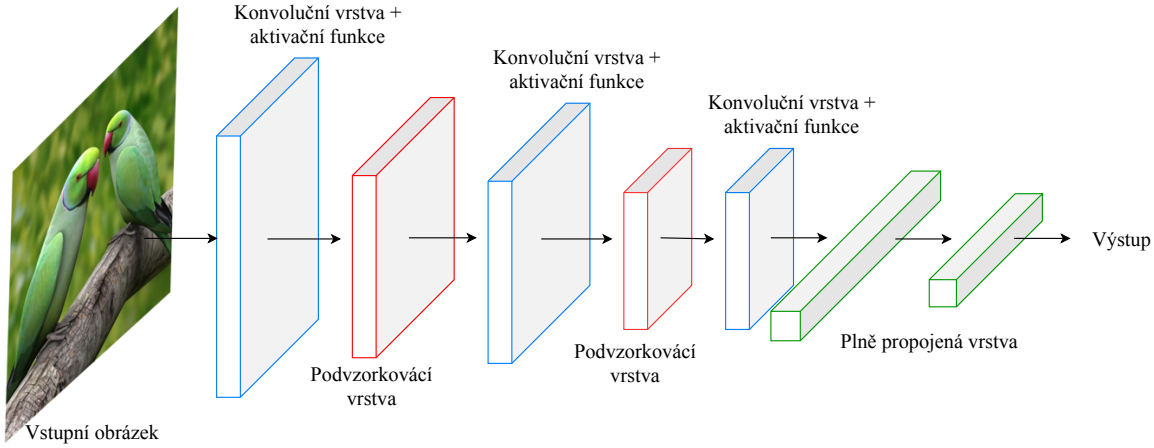
Neuronové sítě mají mnoho důležitých vlastností, ale klíčová je schopnost učit se. Učení neuronových sítí spočívá ve změně a adaptaci vah neuronů v sítě tak, aby celá síť produkovala požadované výstupy. Učení sítí může být provedeno v souladu s různými učebními algoritmy. Nejpoužívanějším je algoritmus učení s učitelem (*supervised learning*) [42]. Adaptace vah neuronů se provádí pomocí algoritmu zpětného šíření chyb (*backpropagation*) [10], který iterativně adaptuje každou váhu na základě parciální derivace rozdílu mezi očekávanými a výstupními hodnotami sítě.

Konvoluční neuronové sítě

Jedním z typů neuronových sítí klasicky používaných pro práci s obrazovými daty jsou konvoluční neuronové sítě. Výhoda tohoto typy je v tom, že dané sítě jsou schopné za pomocí konvolučních filtrů extrahovat charakteristické příznaky objektů z obrázku, například hrany nebo barvy. Na obrázku 2.4 je příklad architektury konvoluční neuronové sítě [61], která se skládá z několika základních druhů vrstev: konvoluční (*convolution*), aktivační, podvzorkovací (*pooling*) a plně propojená (*fully connected*).

Název „konvoluční“ pochází z matematické operace konvoluce, což je druh lineární operace, který konvoluční vrstvy sítě používají k výpočtům. Aplikací teto operace na dvě funkce generuje třetí funkci, kterou lze někdy považovat za modifikovanou verzi jedné z původních funkcí. Konvoluci ve vrstvě si můžeme jednoduše představit jako posouvání konvolučního filtru po matici vstupního obrázku a násobení překrývajících se hodnot. Filtry konvoluční vrstvy jsou v podstatě maticemi neuronů, které se také učí algoritmem zpětného šíření chyb, avšak tyto neurony nemají svou vlastní aktivační funkci. Těmto filtrům se také říká recepční pole nebo jádro a jejich hlavním cílem je vytvoření takzvaných příznakových map (*feature maps*), obsahujících charakteristické příznaky z obrázků. Definice této operace bude vypadat následovně:

$$g[x, y] = i[x, y] * k = \sum_a \sum_b k[a, b] i[x - a, y - b], \tag{2.3}$$



Obrázek 2.4: Příklad architektury konvoluční neuronové sítě.

kde $[x, y]$ jsou souřadnice pixelu na obrázku i, k je konvoluční filtr o velikosti $a \times b$ a $g[x, y]$ je výstupní hodnota bodu $[x, y]$.

Aktivační vrstva hraje roli aktivační funkce pro hodnoty příznakových map z konvoluční vrstvy. Po extrakci příznaků konvoluční vrstvou a aplikaci aktivační funkce na něj, za účelem minimalizace časové a paměťové náročnosti při výpočtech, se používá podvzorkovácí vrstva. Tato vrstva redukuje šířku a výšku příznakových map, a to tak, že agreguje několik hodnot do jedné. Plně propojená vrstva se obvykle umisťuje na konec sítě. Jedná se o klasickou neuronovou síť bez skrytých vrstev, ve které každý neuron vstupní vrstvy je propojen se všemi neurony výstupní vrstvy. Výstup plně propojené vrstvy je pak výstupem celé konvoluční sítě.

Díky své struktuře a vlastnostem jsou konvoluční neuronové sítě schopny identifikovat objekty bez ohledu na různé deformace a posunutí, což řeší hodně problémů v úloze sledování objektů.

Obecně existují dva různé způsoby využití hlubokých neuronových sítí pro sledování [86]: použití konvolučních vrstev sítě jako extraktoru příznaků v kombinaci s korelačními filtry a použití celé sítě pro modelování vzhledu objektu a odhadu jeho pohybu.

Podle [9] je možné dal rozdělit sledovací algoritmy hlubokého učení na algoritmy používající jednu neuronovou síť (*single CNN-based tracking*), používající siamské sítě (*siamese CNN-based tracking*), založené na rekurentních neuronových sítích (*RNN-based tracking*) a na sítě využívající posilované učení (*RL-based tracking*). Pro konečnou predikci polohy objektů na obrázcích tyto algoritmy většinou používají regrese souřadnic ohraničení. Jednou z technik pro lokalizaci objektů, která byla převzatá z oblasti detekce objektů a používá se pro sledování, je metoda *bounding box regression* [11] a je popsána na konci kapitoly.

Single CNN-based tracking

Tyto algoritmy sledování využívají jednu konvoluční neuronovou síť pro sledování objektů. Většina těchto algoritmů jako vstupní data používá výřez z obrázku na pozici předchozí polohy sledované cíle, ze kterého se konvolučními vrstvami extrahují příznaky. Za pomoci plně propojených vrstev takové sítě predikují souřadnice nového ohraničení regresí. Existuje

velké množství vizuálních trackerů daného typu. Významnými představiteli jsou MDNet a FCNT.

MDNet tracker [57] pro sledování využívá neuronovou síť, která se skládá ze tří konvolučních vrstev a dvou plně propojených vrstev. Tuto síť ukončuje sada plně propojených vrstev. Během trénování je aktivní jen jedna poslední vrstva a učí se za pomoci stochastického gradientního sestupu. Trénování je zaměřeno na učení počátečních konvolučních vrstev sítě jako extraktoru příznaků. Při použití v praxi se sada plně propojených vrstev vyměňuje za jednu, která se učí sledovat konkrétní cíl za běhu. Sledovaní probíhá zpracováním oblastí kolem předchozí pozice cíle a regresí souřadnic ohraničení.

FCNT tracker [76] se skládá z několika částí. Každá z nich obsahuje primárně konvoluční vrstvy. Začátkem neuronové sítě je část VGG Net sítě [72], jejíž výstupní příznakové mapy jsou zpracovány dvěma malými sítěmi. Jedna klasifikuje objekt a druha specifikuje jeho polohu pomocí regrese takzvané teplotní mapy (*heat map regression*) cíle.

Siamese CNN-based tracking

Siamské neuronové sítě se skládají ze dvou obvykle identických sítí, jejichž výstupy jsou pak spojeny. Poslední část sítě produkuje jediný výstup celé sítě. Trackery založené na siamských sítích prokázaly vynikající výkon na VOT[1] soutěží a OTB datasetu [3] hlavně díky jejich schopnosti modelovat podobnosti mezi objekty. Vstupy trackerů jsou obvykle aktuální a předchozí snímky, nebo výřezy z aktuálního a předchozího snímků odpovídající poloze cíle a jeho okolí z předchozího snímku. Každý ze vstupních obrázků je zpracován jednou ze sítí. Obě sítě získávají příznaky ze vstupních dat, na jejichž základě se pak počítá posunutí cíle ve scéně.

V literatuře [63] se můžeme setkat s rozdělením těchto trackerů na *Two Channel Siamese*, *Pseudo Siamese*, *Siamese*, *Two-Stream Siamese* a *Recurrent Siamese* sítě. Některými z těchto trackerů jsou SINT tracker[74], SiamFC tracker [5], skládající se z konvolučních vrstev jako FCNT, a DCFNet [77] fungující na spolupráci siamských sítí a korelačních filtrů.

RNN-based tracking

Rekurentní neuronová síť (*Recurrent Neural Networks, RNN*) [71] je typem neuronových sítí, jejíž výstup není dán pouze aktuálním vstupem ale i vnitřním stavem sítě, který kóduje historii vstupů, které už byly zpracovány. Toto jim umožňuje učit se různým reprezentacím a závislostem mezi vstupy napříč časem. Rekurentní neuronové sítě se používají pro predikci časových řad, rozpoznávání řeči, ručně psaných textů a skládání hudby.

Také byly pokusy použit tento druh sítě ve vizuálním sledování objektů. Rekurentní neuronové sítě takových trackerů přijímají na vstup příznakové mapy vzniklé zpracováním vstupních obrázků konvolučními vrstvami. V následujících snímcích tyto sítě efektivně predikují pohyb cíle na základě jeho pohybu v předchozích snímcích. Sítě krátkodobě chrání informaci o vizuálních vlastnostech sledovaných cílů. Díky tomuto jsou algoritmy založené na nich velmi robustní a odolné proti různorodým okluzím a překrytím. Metodou *bounding box regression* pak tyto trackery definují ohraničení pro pohybující se objekt. Významnými trackery jsou SANet tracker [16], ve kterém se střídají vrstvy konvoluční a rekurentní sítí a tracker ROLO [58], který je postaven na spolupráci detektoru YOLO (*You Look Only Once*) [66] a LSTM (*Long Short-Term Memory*) rekurentní neuronové sítě.

[1] http://www.votchallenge.net/

RL-based tracking

Při posilovaném učení (*Reinforcement Learning*) [48] neuronová síť interaguje s vnějším prostředím a učí se volit činnost tak, aby dosáhla nějakého určitého cíle, obecně nějaké odměny. Při rozhodování síť zkoumá nové taktiky a rozhodnutí, která mohou vést k vyšším ziskům a snaží se maximalizovat celkovou odměnu. Možnosti posilovaného učení byly také využity pro řešení úlohy sledování objektů. ADNet tracker [85] je navržen z konvoluční neuronové sítě, která se za běhu sledování učí vybírat optimální akce pro sledování cíle (například posunutí nebo škálování ohraničení) z jeho aktuální pozice. EAST tracker [28], podobně jako ADNet tracker, se skládá z konvoluční sítě, která produkuje optimální akce pro bounding box. Tento tracker je ale schopný pomocí natrénovaného agentu řídit jakými vrstvami sítě bude zpracován vstupní obrázek, a to na základě jeho složitosti. Akce predikované těmito trackery se používají k regresi souřadnic ohraničení cílů.

2.5 Detektory obličejů

V současnosti existuje velké množství algoritmů detekce objektů, avšak v rámci teto práce je pozornost věnovaná těm, které jsou zaměřeny na detekci obličejů. Podle [17] metody detekce lidských tváři je možné rozdělit do tří skupin:

- detektory založené na kaskádových lineárních klasifikátorech (angl. *cascade-based*)
- detektory založené na strukturních klasifikátorech (angl. *DPM-based*)
- detektory založené na neuronových sítích

Základní myšlenkou detekce obličeje na základě kaskádové strategie je rychle vyloučit většinu oblastí pozadí v rané fázi, aby bylo snížené množství výpočtů v pozdější fázi [82]. Velmi významnými detektory této kategorie jsou starší Viola-Jones a AdaBoost detektory.

DPM-based metody (*Deformable Part Model*) [19] jsou založeny na technice hledání deformovatelných částí modelů, kde je obličej definován jako soubor jeho částí. Klasifikátor SVM (*Support Vector Machine*) [15], často používaný u těchto metod, je natrénován k nalezení těchto částí a jejich geometrického vztahu. Tyto detektory jsou odolné částečné okluzi, protože mohou detekovat obličeje i pokud některé části nejsou přítomny.

Detektory založené na neuronových sítích jsou velmi přesné [24]. Oproti výše uvedeným algoritmům, dané detektory jsou schopny efektivně detekovat obličeje v úlohách reálného světa s většími vizuálními variacemi lidských tváří. I když takové detektory jsou přesné a velmi odolné vůči okluzím a deformacím, jejich výpočetní rychlost není vysoká.

Existuje množství implementací detektorů obličejů založených na neuronových sítích. Některé z nich jsou rozšířenými verzemi univerzálních detektorů objektů. Například, pro účely detekce obličejů mohou být modifikovány takové detektory jako R-CNN (*Regions with CNN features*) [22], Fast R-CNN [21], Faster R-CNN [67] nebo YOLO. Metody založené na R-CNN modelu jsou odolné vůči velké variaci vzhledů obličeje a demonstrují *state-of-the-art* výkon. Zde se popisují několik takových detektorů.

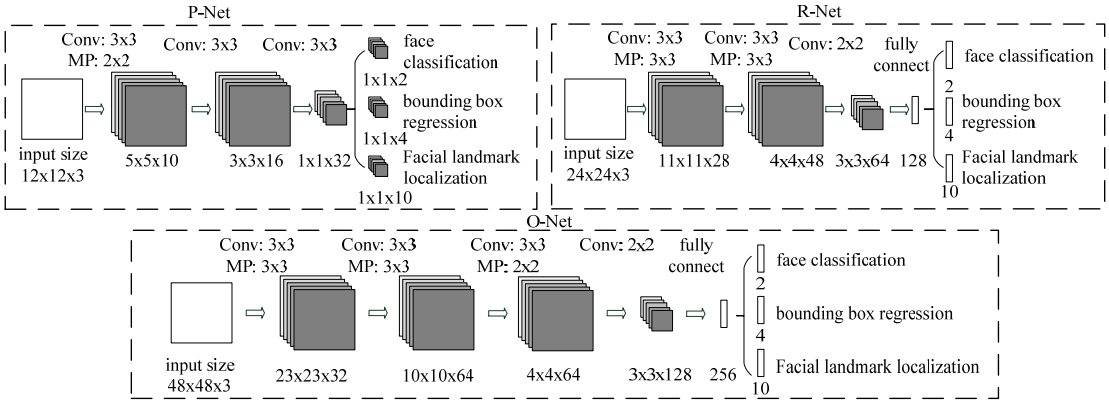
MTCNN

Detektor MTCNN [87] je rychlý detektor obličejů, jehož neoptimalizovaná MATLAB implementace dosahuje rychlosti 99 FPS na GPU. Tento kaskádní detektor se fakticky skládá

ze 3 neuronových sítí: P-Net (*Proposal Net*), R-Net (*Refine Net*) a O-Net (*Output Net*). Architektura detektoru je znázorněna na obrázku 2.5.

Ze vstupního obrázku se nejdříve vytváří sada kopií vstupního obrázku různých velikostí (*image pyramid*). P-Net síť získává z obrázků bounding boxy, které potenciálně mají v sobě obličeje (*stage 1*). Oblasti odpovídající získaným bounding boxům pak zpracovává složitější R-Net síť, která odstraňuje všechny kandidáty bez lidských obličejů (*stage 2*). Zbylé oblasti zpracovává O-Net síť, která vykonává několik úloh: definuje jsou-li v oblastech lidské obličeje a jak se mají posunout bounding boxy, aby co nejlépe obklopovaly obličeje a lokalizuje obličejové body (*stage 3*).

Během zpracování každé z těchto neuronových sítí, na ohraničení, které se překrývají, aplikuje operace *non-maximum suppression*, která slučuje dané boxy dohromady. Každá neuronová síť detektoru zpřesňuje ohraničení za pomocí metody *bounding-box regression*.



Obrázek 2.5: Architektura detektoru MTCNN. Převzato z [87].

CMS-RCNN

Detektor CMS-RCNN (*Contextual Multi-Scale Region-based CNN*) [89] je schopný detekce obličejů při velmi velkých okluzích a deformacích, různých rozlišeních a osvětlení. Architektura neuronové sítě tohoto detektoru umožňuje současně zpracovávat nejenom oblasti lidských obličejů, ale i kolem nich, například oblasti potenciálně obsahující tělo. Toto dovoluje detektoru lokalizovat obličeje, které například nejsou skoro úplně vidět.

Jak píšou autoři, detektor je založen na modelu VGG-16 [72] a skládá se ze dvou sítí. MS-RPN síť (*Multi-Scale Region Proposal Network*), obsahující 5 konvolučních vrstev, extrahuje příznaky z oblastí vstupního obrázku, které jsou pak zpracovány pomocí CMS-CNN sítě (*Contextual MultiScale Convolution Neural Network*). Tato síť na základě získaných charakteristických příznaků obličejů a těl pak produkuje pravděpodobnost výskytu obličejů a aplikuje *bounding box regression* metodu pro jejích lokalizaci.

HyperFace

HyperFace detektor [65] je schopný nejenom detekce obličejů, ale i obličejových bodů, odhadu póz hlavy a pohlaví persony. Tento detektor se skládá ze tří modulů. První modul získává ze vstupního obrázku oblasti, které potenciálně obsahují obličeje a mění jejích velikost na 227 × 227 pixelu. Druhý modul používá konvoluční vrstvy AlexNet sítě [43], jimiž

klasifikuje oblasti zpracované prvním modulem a určuje jsou-li na nich obličeje. Pokud oblast obsahuje obličej, tento modul lokalizuje pro něj obličejové body, odhaduje pózu hlavy a pohlaví osoby. Třetí modul slouží pro následné zpracování pozitivních oblastí, konečnou predikci ohraničujících rámečku a zvyšování přesnosti výstupů z druhého modulu.

Významnou technikou pro predikci a upřesňování ohraničení pro objekty na obrazcích, kterou také využívají popsané detektory a nejenom v detekci obličejů, je již zmíněná metoda *bounding box regression*.

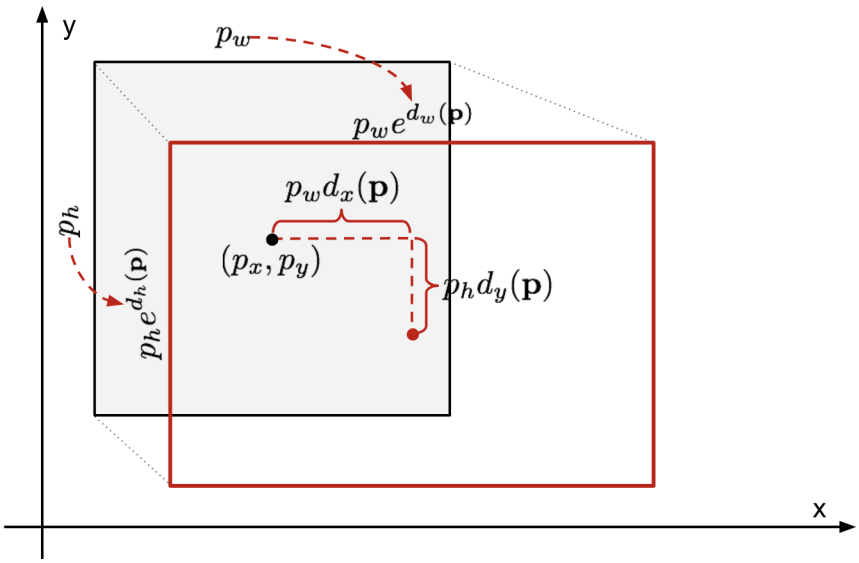
Bounding box regression

Cílem dané metody [11] je pomocí regrese predikovat posun souřadnic ohraničení relativně k souřadnicím zkoumané oblasti. V případě neuronových sítí, pomocí extrahovaných příznaků objektu, se určují parametry transformace ohraničovacích rámečků, které jsou pak využity k definování ohraničení objektu.

Pro navrhnutý bounding box $P = (P_x, Py, P_w, P_h)$, kde P_x a P_y jsou souřadnice středu a P_w a P_h jsou šířka a výška ohraničení, neuronová síť produkuje parametry transformace c_x, c_y, c_w, c_h. Pomocí těchto parametrů se pak počítá výsledné ohraničení G pro objekt a to následovně:

$$\begin{aligned} G_x &= P_w c_x + P_x \\ G_y &= P_h c_y + P_y \\ G_w &= P_w exp(c_w) \\ G_h &= P_h exp(c_h), \end{aligned} \tag{2.4}$$

kde G_x a G_y jsou souřadnice středu a G_w a G_h jsou šířka a výška ohraničení G.



Obrázek 2.6: Transformace ohraničení pomocí metody *bounding box regression*. Převzato z [78].

Kapitola 3

Návrh a implementace sledovací sítě

Algoritmy sledování založené na hlubokém učení jsou odolnější k okluzím a deformacím sledovaných objektů, něž algoritmy používající korelační filtry, a také jsou velmi efektivní pro sledování objektů v reálném prostředí. V případě modelů hlubokého učení je také možné využit *offline-learning* přístup a natrénovat model speciálně pro úlohu sledování obličejů, kterou řeší tato práce.

Z těchto důvodu jsem se rozhodl použit algoritmus založený na hlubokém učení pro sledování obličejů ve snímcích. Natrénování modelů hlubokého učení pro sledování jednoho konkrétního objektu by eliminovalo potřebu modelu učit se za běhu sledování, kterou mají algoritmy používající korelační filtry. Tudíž by to zmenšilo výpočetní náročnost, což je důležité pro sledování v reálném čase. Také vzhledem ke specifice sledování objektů jednoho druhu, použití složitých modelů, jako siamské nebo rekurentní sítě, je zbytečné.

Proto v této práci jsem využil jednu natrénovanou konvoluční neuronovou síť jako řešení modelování vzhledu a odhadu pohybu běhen sledování obličejů. Tato kapitola popisuje algoritmus, podle kterého funguje síť a návrh architektury a trénování dané neuronové sítě.

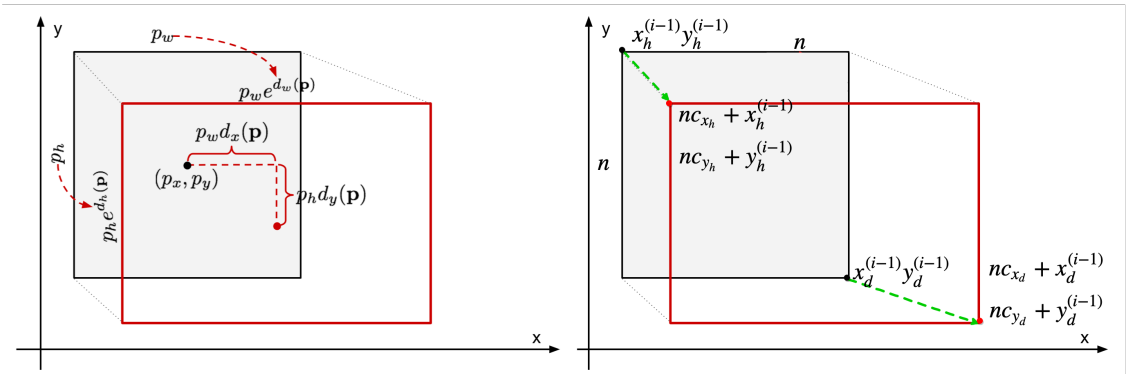
3.1 Sledovací neuronová síť

Myšlenka sledování detekovaných tváří v následujících snímcích spočívá v posunutí ohraničujících rámečků tak, aby přesně obklopovaly lidské tváře. Pro dané účely dále navrhuji sledovací neuronovou síť, která implementuje algoritmus *bounding box regression* a za pomoci svých vlastností říká, jak se mají posouvat ohraničující rámečky.

V této práci je sledování obličejů v sekvenci obrázků pomocí dané metody implementováno tak, že oblast ohraničení tváře z předchozího snímku slouží jako vstupní data pro neuronovou síť. Předpokládejme, že se v následujícím snímku s pohybem osoby sledovaný obličej posunul. Cílem neuronové sítě je odhadnout jeho novou polohu.

Namísto regrese souřadnic středu a poměru ohraničení jsem se rozhodl natrénovat regrese souřadnic horního levého a dolního pravého rohů bounding boxu [26]. Bounding box ze snímku $i-1$ má souřadnice $x_h^{(i-1)}$, $y_h^{(i-1)}$, $x_d^{(i-1)}$, $x_d^{(i-1)}$, které odpovídají souřadnicím levého horního a pravého dolního rohu. Šířka a výška boxu jsou pak:

$$\begin{aligned} w^{(i-1)} &= x_d^{(i-1)} - x_h^{(i-1)} \\ h^{(i-1)} &= y_d^{(i-1)} - y_h^{(i-1)}. \end{aligned} \tag{3.1}$$



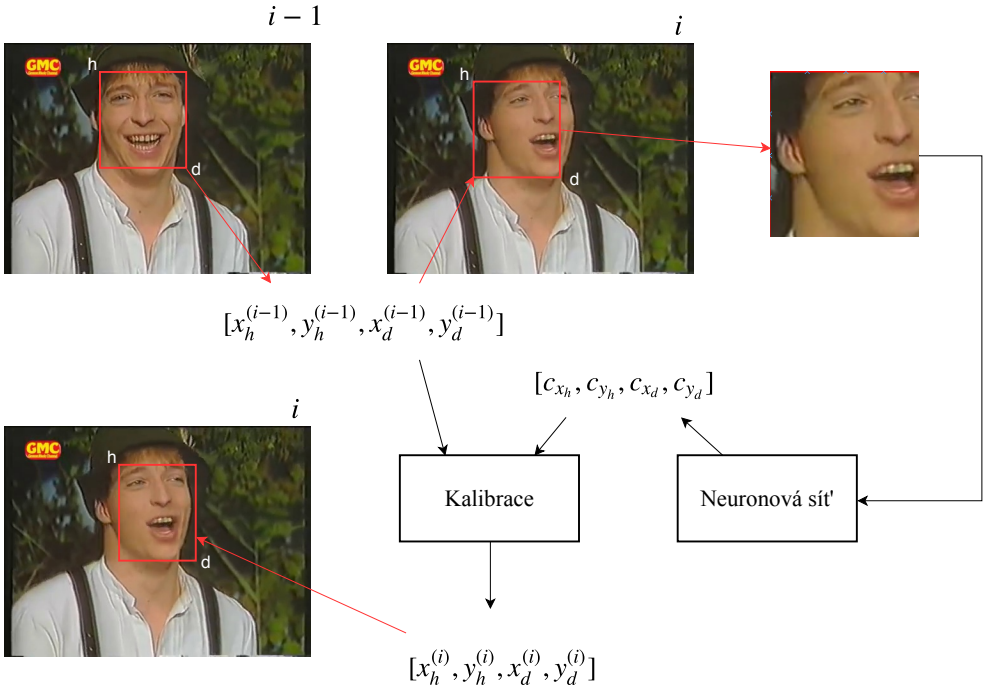
Obrázek 3.1: Transformace ohraničení originálním algoritmem bounding box regression (zleva) a algoritmem použitým v práci (zprava).

Podle těchto souřadnic ze snímku i se vyřezává oblast obsahující posunutou lidskou tvář velikosti $n \times n$, kde $n = max(w^{(i-1)}, h^{(i-1)})$. Tento výřez je pak předzpracováván a je podán na vstup neuronové sítě. Síť extrahuje charakteristické rysy obličeje, určuje posun tváře na výřezu a vrátí parametry transformace c_{x_h}, c_{y_h}, c_{x_d}, c_{y_d} pro souřadnice ohraničení. Po vynásobení stran výřezu a těchto parametrů získáváme offsety pro souřadnice bounding boxu ze snímku $i-1$. Přičtením offsetu k $x_h^{(i-1)}$, $y_h^{(i-1)}$, $x_d^{(i-1)}$, $y_d^{(i-1)}$ získáme nové souřadnice boxu ve snímku i:

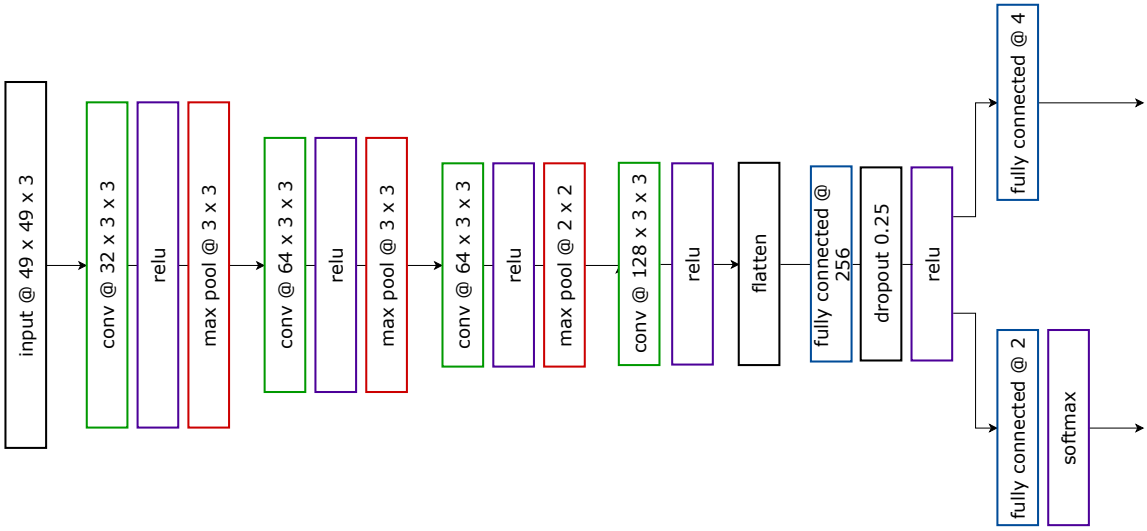
$$\begin{aligned} x_h^{(i)} &= nc_{x_h} + x_h^{(i-1)} \\ y_h^{(i)} &= nc_{y_h} + y_h^{(i-1)} \\ x_d^{(i)} &= nc_{x_d} + x_d^{(i-1)} \\ y_d^{(i)} &= nc_{y_d} + y_d^{(i-1)}. \end{aligned} \tag{3.2}$$

Pro řešení problému výskytu objektu a odhadu pohybu daná neuronová síť využívá *multi-task learning* [88], tedy vykonává několik úloh současně. Jedním z výstupů této sítě bude informace potřebná ke změně bounding boxu, která je popsána výše. Dalším výstupem bude míra pravděpodobnosti, že v okně je lidský obličej. Tato informace slouží například k odstranění zbytečných sledovacích oken, která nesledují žádnou tvář, například když sledovaný obličej zmizí ze scény.

Na obrázku 3.3 je navrhnuta architektura sledovací sítě. Síť se skládá ze čtyř konvolučních a tří podvzorkovacích vrstev. Jako funkce aktivace je zde použita funkce ReLU. Vrstva *flatten* transformuje vícedimenzionální výstup konvolučních vrstev na jednodimenzionální vektor, který je pak vstupem plně propojených vrstev. Pro ošetření stavu „přetrénování“, při němž síť dobře funguje s trénovacími vzorci, které si „zapamatovala“, ale nedokáže se vypořádat s reálnými daty, během učení je zde použita drop-out vrstva [73], která nuluje výstupní data s pravděpodobnosti 0.25. Danou síť ukončují dvě separátní, plně propojené vrstvy, jejichž výstupem jsou pravděpodobnost výskytu obličejů a parametry transformace. Na výstup plně propojené vrstvy, zodpovědné za výskyt tváři, je aplikována aktivační funkce softmax [59]. Tato funkce se často používá pro úlohu klasifikace a normalizuje výstup předchozí vrstvy v rozsahu [0, 1].



Obrázek 3.2: Ilustrace procesu zpracování dvou po sobě jdoucích snímků neuronovou sítí.



Obrázek 3.3: Architektura sledovací neuronové sítě.

3.2 Vytvoření trénovacího datasetu

Navržená sledovací síť se učí metodou učení s učitelem. Při učení s učitelem [42] je síť trénována pomocí správných odpovědí (výstupy) odpovídajících počátečním datům (vstupům). V procesu supervizovaného učení se váhy sítě upravují tak, aby síť generovala odpovědi, které jsou nejblíže těm, které jsou správné. Trénovací data jsou vybrána tak, aby plně popsala všechny vlastnosti množiny, které jsou důležité pro danou úlohu. Aby se síť naučila efektivně pracovat s daty, jak trénovacími, tak i testovacími, je potřeba mít velkou sadu trénovacích vzorků. Obvykle se během učení síť mnohokrát trénuje na všech trénovacích obrazcích, které mohou být i promíchány v náhodném pořadí. Čím vetší je počet vzorků a čím jsou mnohotvárnější, tím přesněji funguje neuronová síť.

Navržená sledovací síť vykonává dvě činnosti: určuje, zda je ve vstupním obrázku obličej a jak se má posunout bounding box. Proto pro klasifikaci musí trénovací dataset obsahovat dva druhy vzorků, s obličejem a bez něj. Dalším požadavkem bude posunutí těchto obličejů o nějakou náhodnou veličinu pro učení regrese. Pro účely trénování sítě se mohou využít libovolné hotové datasety obličejů s definovanými bounding boxy.

WIDER FACE [83] dataset obsahuje 32,203 obrázků s 393,703 anotovanými bounding boxy tváří. Tento dataset má vysoký stupeň variability v měřítku, póze, okluzi, expresi a osvětlení. Obrázky v tomto datasetu jsou rozděleny podle akcí, které vykonávají osoby na nich. Celkem jde o 60 akcí. Pro každou akci v datasetu je od 1000 do 3000 snímků. Všechny obrázky v datasetu byly získány pomocí vyhledávacích programů jako Google a Bing. Z těchto obrázků byly manuálně odstraněný všechny, které neobsahují žádnou lidskou tvář. Zbývající obrázky jsou anotovaný také manuálním způsobem a zkontrolovány různými lidmi.

AFW (*Annotated Faces in the Wild*) [44] dataset je vytvořen z fotografií z hostingu Flickr. Tento dataset obsahuje 205 obrázků s 473 anotovanými tvářemi. Anotace pro každý obličej se skládá z bounding boxu, šesti obličejových bodů a úhlu otáčení osoby.

FDDB (*Face Detection Data Set and Benchmark*) [38] dataset obsahuje 2,845 obrázků a 5,171 anotovaných obličejů. Obrázky pro tento dataset jsou získány z novinových článků webu Yahoo!. Charakteristickou vlastností daného datasetu jsou anotace obličejů, které, na rozdíl od ostatních datasetů, mají tvar elipsy.

PASCAL FACE [80] obsahuje 851 obrázků a 1,341 anotovaných lidských tváří. Dataset je vytvořen z testovacích vzorků datasetu PASCAL obsahujících obličeje. Dataset MALF [81] obsahuje 5,250 obrázků a 11,931 anotovaných obličejů.

Vzhledem k tomu, že neuronové sítě obvykle vyžadují velké množství dat pro supervizované trénování, z výšeuvedených datasetů byl zvolen dataset WIDER FACE. Obrázková data z datasetu WIDER FACE jsem zkombinoval s daty z datasetu CelebA. Dataset CelebA [50] obsahuje 202,599 obrázků s anotovanými tvářemi a obličejovými body. Z těchto dvou datasetu jsem vytvořil dataset pro trénování neuronové sítě.

Z každého obrázku z datasetů WIDER FACE a CelebA jsem vybral obličeje, jejichž velikost ohraničení je vetší než 40×40 pixelů. Dále jsem z těchto obrázků vyřezal oblasti odpovídající souřadnicím ground truth ohraničení zvětšené 1.5-krát. Tyto regiony jsou uloženy do .jpg souborů a označeny jako pozitivní vzorky. Jako negativní vzorky jsem zvolil náhodné oblasti na obrázku, které neobsahují žádný obličej. Toto rozdělení na pozitivní a negativní vzorky umožňuje trénovat klasifikační část neuronové sítě.

Pro učení regrese se ohraničení obličejů na pozitivních vzorcích transformují tak, aby naučená síť byla schopná nejen sledovat obličeje statické velikosti ale i škálovat ohraničující

Obrázek 3.4: Ukázka obrázků z datasetu WIDER FACE. Převzato z [83].

Obrázek 3.5: Ukázka obrázků z datasetu CelebA. Převzato z [50].

rámečky pro zvětšující se nebo zmenšující se obličeje, například když se osoba přibližuje nebo vzdaluje od kamery. Proto jsou v průběhu učení bounding boxy obličejů náhodně škálovány a pak posunuty o náhodné veličiny: velikost transformovaného bounding boxu může být minimálně 75% a maximálně 1.25% oproti původní velikosti a může být posunut různými směry maximálně o 40%.

Celkově z datasetů WIDER FACE a CelebA bylo vygenerováno 446000 obrázků v poměru 50% pozitivních a 50% negativních. Trénovací dataset během učení byl rozdělen na trénovací, validační a testovací sady. Trénovací sada se skládá ze 70% obrázků nového datasetu, na nichž se neuronová síť bezprostředně učí. Účelem validační sady, která obsahuje 10% obrázků a neúčastní se procesu učení, je pravidelné testování neuronové sítě během učení, s cílem detekovat stav přetrénovaní. Testovací sada, obsahující 20% obrázků, je nutná pro testování finální verze natrénované sítě.

3.3 Učení modelu

Učení sítě probíhá s použitím takzvaných *hyperparametrů* [45]. Hyperparametry pro učení neuronových sítí nejsou předem známe, nastavují se experimentálně a závisí na typu úlohy, kterou řeší neuronová síť. Existují jen univerzální doporučení ohledně stanovení optimálních hyperparametrů.

Jedním z hyperparametrů je počet epoch učení. Počet epoch říká, kolik krát projdou všechna trénovací data neuronovou sítí během učení. Příliš velký počet epoch vede k přetrénování sítě. Pro učení sledovací sítě byl zvolen počet epoch rovny 100.

Dalším důležitým hyperparametrem je hodnota *learning rate* [90] nebo koeficient učení. Tato hodnota je krok, se kterým se mění váhy sítě. Koeficient učení se nastavuje v rozmezí od 0 do 1. Bude-li tento koeficient příliš malý, trénovaní sítě bude pomalé. Pokud learning rate naopak příliš velký, nastane případ, kdy se váhy sítě přestanou učit vhodným způsobem. Při učení sledovací sítě jsem nastavil koeficient učení na hodnotu 0.001. Každých 20 iterací učení jsem zmenšoval koeficient desetkrát.

Nezbytným parametrem učení sítě je chybová funkce, která určuje, jak se liší výstup sítě od požadovaných hodnot. Vzhledem k tomu, že síť se bude učit více úloh současně (*multi-task learning* [88]), potřebujeme několik chybových funkcí pro různé typy úloh. V této práci jsem použil chybovou funkci *cross entropy loss* [25] pro trénování klasifikace. V rovnici 3.3 chybové funkce t_{in} a s_{in} značí korektní a predikovanou třídu objektů pro obrázek n z množiny vstupních obrázků N. C značí počet tříd – v našem případě 2 třídy, „obličej“ a „není obličej“.

$$CE = -\frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{C'=2} t_{in} log(s_{in}) \tag{3.3}$$

I přesto, že chybové funkce l_2 *loss* a *smooth* l_1 *loss* se široce používají pro učení regrese ohraničení, ve své práci jsem se rozhodl použit chybovou funkce překrytí ohraničení (3.4) (*Intersection Over Union loss*). Podle autorů této chybové funkce [84], parametry transformace ohraničovacích rámečků (*cx*, *cy*, *cw*, *ch*) jsou optimalizovány nezávisle při použití l_2 *loss* a *smooth* l_1 *loss* [46], i když se ve skutečnosti vysoce vzájemně ovlivňují. Takové učení má za následek řadu případů selhání, kdy jedna nebo dvě strany předpověděného ohraničení natrénovanou sítí jsou velmi blízko k správnému ohraničovacímu rámečku, ale celé ohraničení není akceptovatelné. Z experimentů autoři také zjistili, že natrénovaná síť se více zaměřuje na větší objekty, zatímco ignoruje menší objekty.

Z těchto důvodů jsem se rozhodl využit navrženou chybovou funkce *IoU Loss*, která nejenom zabraňuje výše uvedeným problémům, ale i dokonce stabilizuje učení a zlepšuje přesnost regrese. V algoritmu 1 je popis výpočtu chyby pro predikované a správné ohraničení. Rovnice 3.4 je výpočet chyby L_{IoU} přes všechny predikované a správné ohraničení z množiny N.

$$L_{IoU} = \frac{1}{N}\sum_{n=1}^{N} IoULoss(g_n, p_n) \tag{3.4}$$

Celková chyba sítě L (3.5) je pak součtem chyb dvou chybových funkcí:

$$L = CE + L_{IoU}. \tag{3.5}$$

Důležité je také počáteční nastavení vah neuronové sítě [13]. Pokud budou váhy inicializovaný stejnou hodnotou, neurony budou mít stejný gradient při zpětném šíření chyb. To znamená že se nebudou učit a během učení budou mít stejné charakteristiky. Váhy se obvykle nastavují na náhodná malá čísla. Existují také různé heuristiky počátečního nastavení vah, jako například *xavier* a *he* inicializace [14]. V této práci jsem pro počáteční nastavení vah sítě využil heuristiku *xavier*.

Algorithm 1 Chybová funkce IoU loss [84, 46]

Vstup: ground truth ohraničení $g = [x_{1g}, y_{1g}, x_{2g}, x_{2g}]$
Vstup: ohraničení predikované neuronovou sítí $p = [x_{1p}, y_{1p}, x_{2p}, x_{2p}]$
Výstup: chyba překrytí ohraničení $\mathcal{L}$

1: **Funkce** $IoULoss(g, p)$:
2: $w_g = x_{2g} - x_{1g}$
3: $h_g = y_{2g} - y_{1g}$
4: $w_p = x_{2p} - x_{1p}$
5: $h_p = y_{2p} - y_{1p}$
6: $A_g = w_g \cdot h_g$
7: $A_p = w_p \cdot h_p$
8: $I_w = min(x_{2g}, x_{2p}) - max(x_{1g}, x_{1p})$
9: $I_h = min(y_{2g}, y_{2p}) - max(y_{1g}, y_{1p})$
10: $I_w = max(I_w, 0)$
11: $I_c = max(I_c, 0)$
12: $I = I_w \cdot I_h$
13: $U = A_g + A_p - I$
14: $IoU = \frac{I}{U}$
15: $\mathcal{L} = -ln(IoU)$
16: **return** $\mathcal{L}$

Trénování sledovací sítě probíhá následovně. Pomocí OpenCV knihovny se načítají obrázky, jejichž počet definuje parametr *batch size*. Parametr batch size určuje, kolik obrázků zpracuje síť při přímém průchodu a je nastavený na 32.

Z textového souboru se načítají příslušné ground truth ohraničení obličeje na obrázcích. V dalším kroku tyto bounding boxy náhodně posouvají a škálují, jak je to popsáno v podkapitole 3.2.

Z načtených obrázků se pak podle souřadnic transformovaných boxů vyřezávají příslušné oblasti. Tyto výřezy se přizpůsobují velikosti vstupní vrstvy sítě (49×49) a normalizují se na rozsah [-1, 1]. Také se k těmto výřezům náhodně přidávají šumy a mění osvětlení [6]. Po přímém průchodu, pomocí získaných koeficientů regrese od sítě, se vypočítávají predikované ohraničení. Porovnáním se správnými ohraničujícími rámečky se počítá chyba sítě a pomocí optimalizátoru *SGD* (*Stochastic Gradient Descent*) se mění váhy neuronové sítě algoritmem zpětného šíření chyb. Tento proces se opakuje pro všechna data v trénovací sadě a to tolikrát, kolik definuje parametr epoch učení.

3.4 Vývojové nástroje

Ve svojí práci jsem využil skriptovací jazyk Python[1], který podporuje objektově orientované, funkční a imperativní styly programování. Python je univerzální jazyk, ve kterém je možné úspěšně vyvíjet systémové aplikace s grafickým rozhraním, nástroje příkazového řádku, vědecké aplikace, hry, webové aplikace a mnoho dalších.

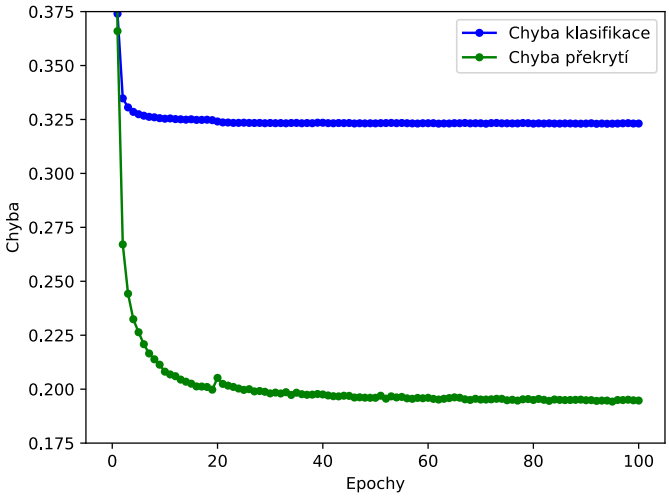
Pro práci s obrázky jsem využil knihovnu OpenCV[2]. OpenCV je knihovna s otevřeným zdrojovým kódem a je zaměřená na úlohy počítačového vidění a strojového učení. Pomocí

[1] https://www.python.org/
[2] https://opencv.org/

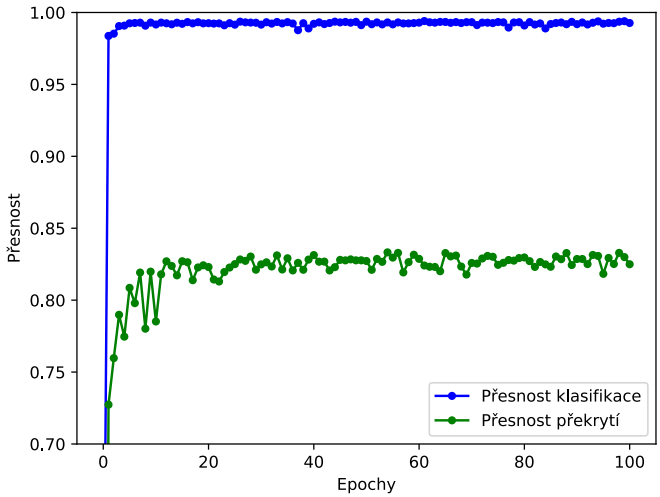
této knihovny lze načítat a ukládat obrázky a provádět různé operace jako normalizaci a změny velikosti obrázků.

Pro implementaci a učení sítě jsem využil framework PyTorch[3]. PyTorch je moderní knihovna hlubokého učení, rozvíjející se pod vedením Facebooku. Tato knihovna má rozsáhlé API pro modelování, implementaci a učení neuronových sítí různých druhů.

Trénování sítě proběhlo na 8 jádrovém procesoru *Intel(R) Core(TM) i7-4790 CPU @ 3.60GHz*. Graf na obrázku 3.6a zobrazuje chybu klasifikace a překrytí ohraničení během trénování sítě. Graf na obrázku 3.6b zobrazuje přesnost klasifikace a překrytí ohraničení na validačním datasetu.



(a) Graf chyb klasifikace a překrytí sítě během trénování.



(b) Graf přesnosti klasifikace a překrytí sítě na validačním datasetu.

[3] https://pytorch.org/

Kapitola 4

Testování a porovnání

V této kapitole se popisuje testování vlastního řešení a porovnání s existujícími metodami na vytvořeném testovacím datasetu. Zde také jsou popsány metriky porovnání. Na závěru kapitoly je krátká analýza výsledků testování.

4.1 Příprava testovacích dat

Pro účely testování sledovacího programu byl vytvořen vlastní testovací dataset. Tento dataset se skládá ze 28 obrázkových sekvencí. 16 sekvencí jsou složeny z krátkých úseků videí z největšího internetového serveru pro sdílení videosouborů Youtube. 5 sekvencí bylo natočeno autorem této práce. Anotace pro výše uvedené testovací data byly vytvořeny pomocí detektoru obličejů MTCNN. Po anotování detektorem, byly všechny snímky zkontrolovány manuálně a špatně anotovaná data byla opravena.

7 sekvencí jsou původem z datasetů pro sledování objektů, ze kterého byly vybrány jenom ty, které obsahují lidskou tvář. Tyto obrázkové sekvence poskytla firma Innovatrics[1], která se zabývá biometrickými technologiemi správy identit. Tyto sekvence již obsahují anotace, ale i přesto byly zkontrolovány.

Obrázkové sekvence obsahují 1 až 3 obličeje ve scénách. 18 sekvencí obsahuje jeden obličej, 5 sekvencí obsahují 2 obličeje a 5 sekvencí obsahují 3 obličeje.

4.2 Průběh testování a kalibrace výstupů

V této práci se provádí porovnání s existujícími trackery popsanými v podkapitole 2.2. To jsou trackery MIL, KCF, MOSSE, Boosting, Medianflow a DCF-CSR(CSRT). Všechny tyto algoritmy sledování jsou součástí knihovny OpenCV.

Pro dodržení stejných podmínek během testování, pro všechny uvedené sledovací algoritmy i vlastní sledovací síť byla lokalizace obličeje prováděna pomocí detektoru MTCNN. Tento detektor definoval sledovací okna pro obličeje na prvních snímcích obrázkových sekvencí. Každých 100 snímků tento detektor lokalizoval nové a ztracené obličeje. Po ukončení zpracování jedné sekvence, než se začínalo testovaní na další, byly sledovací algoritmy vráceny do původního stavu – mazala se informace o oblastech, které sledovaly.

Při porovnání algoritmů sledování je nutné brát v úvahu to, že obrázkové sekvence jsou anotovány různými způsoby. Jak již bylo popsáno v podkapitole 4.1, část sekvencí je anotována detektorem MTCNN a ostatní jsou z různých datasetů pro sledování. Tyto

[1] https://www.innovatrics.com/

datasety mají odlišné politiky anotování a proto například v některých sekvencích bounding boxy obklopují jenom obličeje a v některých jsou anotovaný nejenom obličeje ale i celé hlavy. Rozdíl je také v minimální velikosti anotovaných bounding boxu v sekvencích. Různé algoritmy nemusí stejně přesně sledovat malé obličeje.

Tyto odlišností ovlivňuji výsledky vyhodnocování, proto sledovací algoritmy naučené specifické politice sledování nemohou byt správně vyhodnocené na různých datasetech bez předzpracování. Metody sledování, se kterými se porovnává vytvořená síť, dokonce ani nejsou zaměřené na sledování obličejů. Ve své práci [56] Markus Mathias, Rodrigo Benenson, Marco Pedersoli a Luc Van Gool poskytují řešení problému pomocí předzpracování anotací datasetů a globálních rigidních transformací výstupů sledovacích metod.

Předzpracování anotací testovacích datasetů zahrnuje kontrolu anotací a „ignorování“ malých bounding boxu a obličejů, které se těžce detekují a sledují. Při kontrole datasetových anotací se opravují anotační chyby, jako například špatná forma ohraničení, nebo se označují obličeje, které předtím nebyly anotovány.

„Ignorováním“ malých ohraničení se myslí speciální označení bounding boxu flagem, který dovoluje metodám během testování nesledovat určitý obličej, pokud je příliš malý nebo se špatně lokalizuje. To že metoda nedokáže sledovat takový obličej nebude považováno jako chyba při testech, avšak metoda také nebude „potrestaná“ pokud je schopná tento obličej sledovat.

Rozhodování, jaky bounding box se bude „ignorovat“, se provádí následujícím způsobem. V této práci je během testování minimální velikost stran bounding boxu obličejů 30 pixelů. To je práh α. Naivním přístupem by bylo označit všechny boxy menší něž α za malé, ale v případě velikosti například $\alpha - 1$, by bylo „ignorování“ tohoto boxu chybou. Pro vyřešení tohoto problému autoři práce zavedli další práh β, který se počítá:

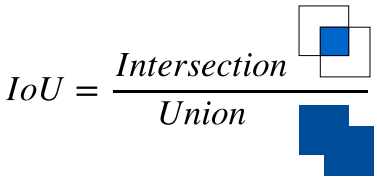
$$\beta = \sqrt{overlap \cdot \alpha^2}, \tag{4.1}$$

kde *overlap* je stupeň překrytí ground truth ohraničení a predikovaného bounding boxu. Pro kalibraci výstupu sledovacích metod, hodnota *overlap* v této práci byla nastavena na 0.5 a proto podle β všechny bounding boxy, které mely strany menší něž 21 pixel, byly označeny flagem „ignorování“.

Za účelem maximalizace překrytí anotovaných a predikovaných ohraničení, které jsou výstupem metod sledování, se přes každou obrázkovou sekvenci počítají globální rigidní transformace zvlášť pro každou sledovací metodu. Pomocí těchto transformací se pak provádí translace a škálování bounding boxu jednotlivých metod. Vzhledem k tomu, že parametry bounding boxu ovlivňují sebe navzájem, kalibrace se provádí několikrát. V této práci se výstupy sledovacích metod kalibrují pětkrát. Kvůli transformacím nemá žádná metoda výhodu oproti ostatním metodám a lze porovnávat vytvořený tracker s metodami, které nemusí být zaměřené na sledování obličejů.

4.3 Metriky pro porovnání metod

Pro odhad přesnosti a spolehlivosti sledovacích metod [55] porovnáváme ground truth ohraničení ve snímcích sekvencí s oblastmi, které metody sledují. Při tomto porovnání se zajímáme o to, jak moc se tyto oblasti překrývají. Míra tohoto překrytí se nazývá Intersection Over Union (IoU), což je podíl průniku a spojení oblastí vymezených predikovanými a skutečnými hraničními oblastmi.



Obrázek 4.1: Intersection Over Union.

Hodnoty IoU jsou v rozsahu od 0.0 do 1.0. IoU rovny 0.0 znamená, že se predikovaná oblast nepřekrývá se skutečnou. IoU rovny 1.0 znamená plné překrytí těchto oblastí a nejpřesnější detekci. Ve skutečností metody zřídka dosahují úplného překrytí oblastí. Proto se používá práh, který udává minimální hodnotu IoU, aby se detekce považovala za úspěšnou. Při testování metod sledování byl tento práh nastaven na hodnotu 0.6.

Pokud se oblasti překrývají vice jak 60% (IoU>=0.6), detekce je považovaná za skutečné pozitivní (*True Positive, TP*). Falešně positivní (*False Positive, FP*) detekce jsou případy, kdy za hledanou oblast je označena nějaká jiná, než která opravdu obsahuje objekt (IoU<0.6). Není-li anotovaný objekt nalezen, detekce je označena jako falešně negativní (*False Negative, FN*). V případě, kdy nebyl detekován žádný objekt a ani ve skutečností není žádný anotován, jedná se o skutečně negativní detekci (*True Negative, TN*).

Pro vyhodnocení úspěšnosti metod detekce a sledování se často používá *precision/recall* křivka. Hodnota *precision* vyjadřuje podíl skutečně pozitivních detekcí mezi všemi, které metoda označila jako pozitivní.

$$precision = \frac{TP}{TP + FP} \tag{4.2}$$

Hodnota *recall* ukazuje podíl nalezených detekcí pozitivní třídy mezi všemi skutečně pozitivními detekcemi. Počítá se jako podíl počtu skutečně pozitivních a součtu skutečně pozitivních a falešně negativních detekcí.

$$recall = \frac{TP}{TP + FN} \tag{4.3}$$

Jedna z populárních metrik pro měření přesnosti detekce objektů je hodnota *average precision, AP* [29], která vyjadřuje průměr hodnot *precision* pro hodnoty *recall* v rozsahu od 0 do 1. Hodnota AP ukazuje obsah plochy pod *precision/recall* křivkou.

$$AP = \int_0^1 p(r)dr \tag{4.4}$$

Obsah plochy pod křivkou se také označuje jako *AUC* (*Area Under Curve*) [29] a počítá se jako:

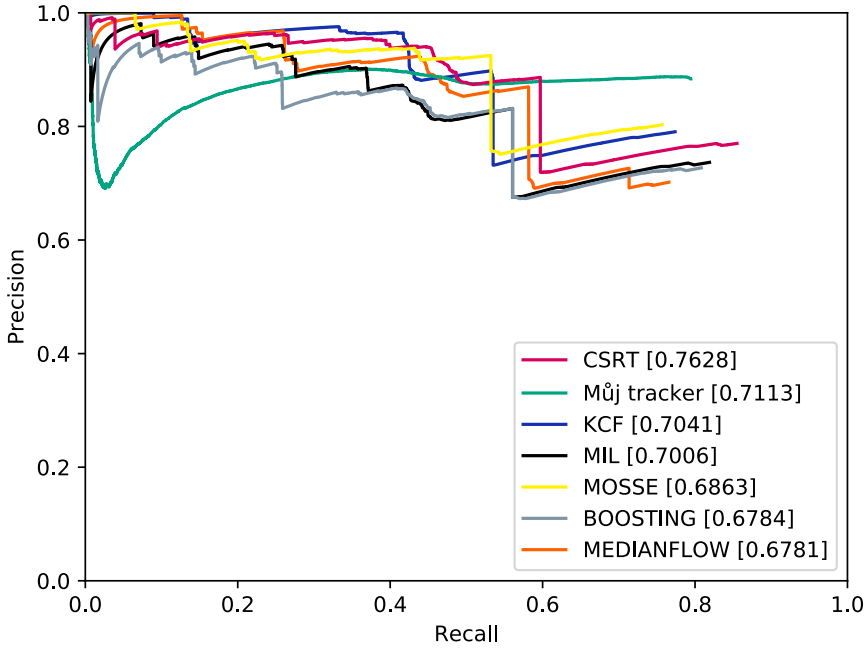
$$AUC = \sum (r_{n+1} - r_n) p_{interp}(r_{n+1}), \tag{4.5}$$

kde $p_{interp}(r_{n+1})$ je interpolace hodnoty *precision* odpovídající hodnotě *recall*. $p_{interp}(r_{n+1})$ se počítá jako nejvyšší hodnota *precision* naměřena pro hodnotu $\tilde{r}$, která je vyšší než hodnota *recall*:

$$p_{interp}(r_{n+1}) = \max_{\tilde{r} \geq r_{n+1}} p(\tilde{r}). \tag{4.6}$$

4.4 Analýza výsledků

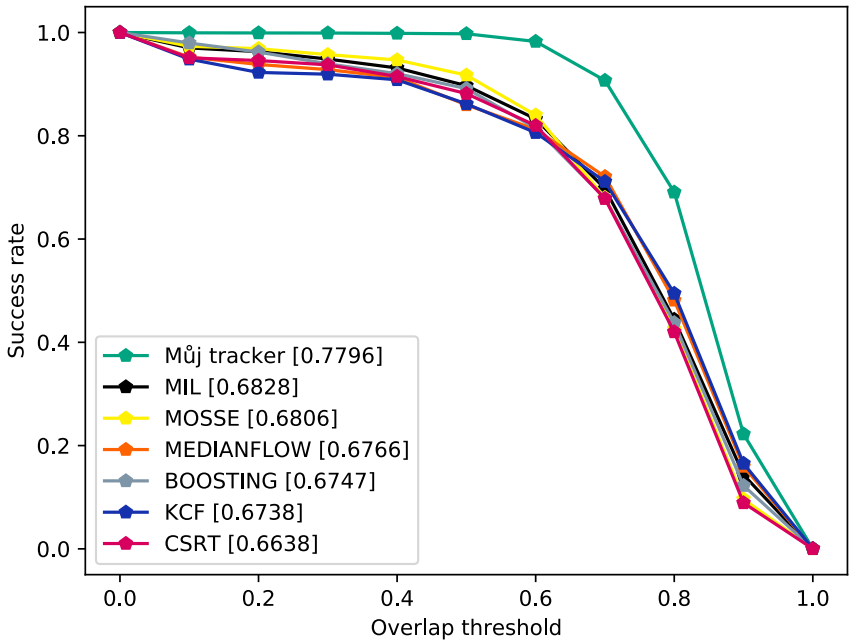
Testovaní sledovacích metod probíhalo na 2 jádrovém procesoru *Intel(R) Core(TM) i5-4790 CPU @ 1.4GHz*. Výstupy testování jsou pak seznamy ohraničení pro snímky z testovací sady a naměřené hodnoty *FPS* (*Frame Per Second*) udávající rychlost zpracování snímků jednotlivými metodami. Z těchto výstupů pak byly vytvořeny grafy, jejichž cílem je vizuálně zobrazit efektivitu, rychlost a přesnost vytvořené v rámci této práce sledovací sítí a porovnat ji s existujícími řešeními.



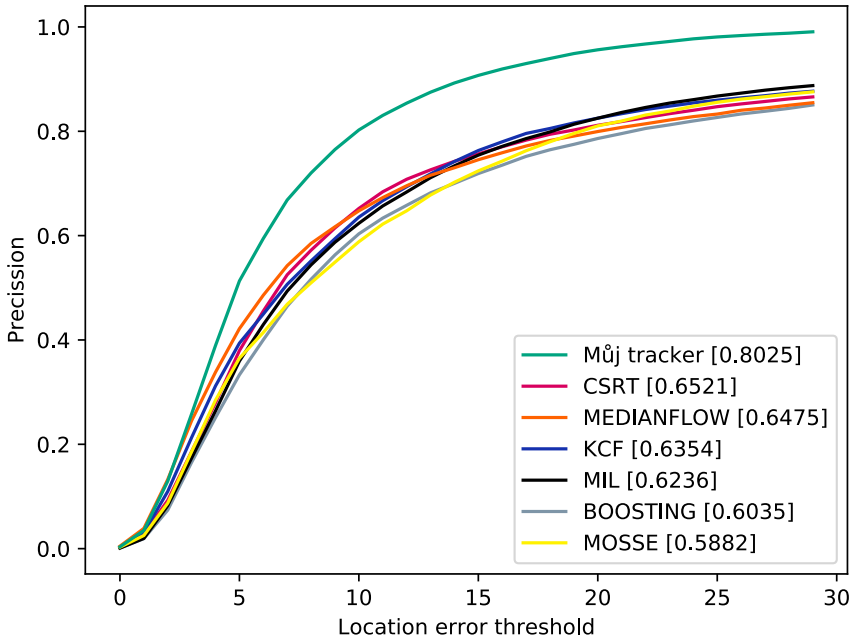
Obrázek 4.2: Precision/recall křivka pro metody sledování.

Na precision/recall křivce v grafu na obrázku 4.2 efektivita jednotlivých trackeru je vyjádřena pomocí obsahu plochy pod křivkou (AUC). Z grafu můžeme vidět, že vytvořená síť dosáhla hodnoty 0.7113 AUC na testovacím datasetu. Nejlepší se stal CSRT tracker, který zasledoval vice ground truth ohraničení a dosahl hodnoty 0.7628 AUC.

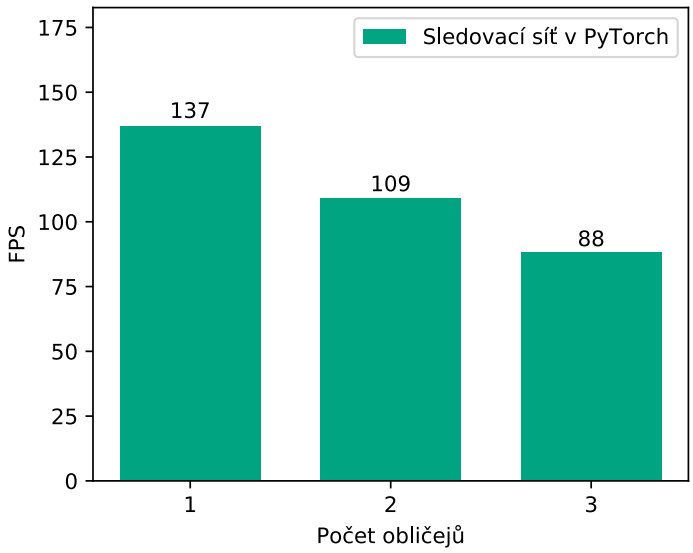
Další graf na obrázku 4.3, vyjadřující přesnost metod, je graf úspěšnosti (*success plot*). Graf úspěšnosti ukazuje procento ohraničujících boxů (*success rate*), jejichž překrytí s odpovídajícími ground truth ohraničovacími rámečky je větší, než daný práh (*overlap threshold*). Metody v grafu úspěšnosti jsou seřazeny podle hodnoty jejich *success score*, která je také definována jako plocha pod křivkou (AUC). Graf úspěšnosti ukazuje, že vytvořená sít má success score 0.7796, což je největší dosaženy výsledek na testovacím datasetu mezi všemi testovanými metodami. Na tomtéž grafu můžeme také vidět, že přibližně 70% všech pozitivních detekcí sledovací sítě má na testovacím datasetu překrytí 0.8.



Obrázek 4.3: Success plot pro metody sledování.



Obrázek 4.4: Precision plot pro metody sledování.



Obrázek 4.5: Graf hodnot FPS pro sledovací neuronovou sít v závislosti na počtu obličejů.

Graf přesnosti na obrázku 4.4 představuje procento ohraničujících boxů, jejichž souřadnice středů jsou v dané vzdálenosti (0 až 30 pixelů) ke korektním ohraničením. Metody v grafu přesnosti jsou seřazeny podle jejich *precision score*, což je procento ohraničujících rámečků ve vzdálenosti 10 pixelů. Z grafu můžeme vidět, že 80% správných ohraničení navržená síť sledovala z maximální vzdálenosti 10 pixelu od středů ground truth ohraničení.

Pro sledovací neuronovou síť jsem také naměřil rychlost zpracování snímků při různém počtu sledovaných obličejů. Graf na obrázku 4.5 ukazuje, že rychlost sledovací neuronové sítě při sledování jednoho obličeje je na testovacím počítače 113 FPS. Na tomtéž grafu můžeme vidět, jak se zvyšujícím se počtem lidských obličejů rychlost sítě klesá o přibližně 15-17%.

Pro algoritmy sledování jsem také naměřil rychlost zpracování snímků na celém testovacím datasetu a dobu sledování obličejů jednotlivými metodami, kterou jsem vyjádřil jako průměr počtu snímku nepřerušovaného sledování obličejů. Z tabulky 4.1 vyplývá, že řešení natrhnuté v této práci není nejrychlejším na testovacím datasetu, ale lokalizované obliceje sleduje nejdéle.

	KCF	MIL	MOSSE	Boosting	Medianflow	CSRT	Můj tracker
FPS	26	7	**491**	15	268	12	126
Počet snímků nepřerušovaného sledování	79	97	91	69	76	83	**141**

Tabulka 4.1: Tabulka hodnot rychlosti zpracování snímků a doby sledování obličejů jednotlivými metodami na testovacím datasetu.

Kapitola 5

Optimalizace sítě a její vyhodnocení

S cílem zrychlit vytvořený tracker jsem optimalizoval model sledovací neuronové sítě pomocí nástrojů ze sady OpenVINO (*Open Visual Inferencing and Neural Network Optimization*) společnosti Intel.

5.1 Optimalizace pomocí OpenVINO

OpenVINO toolkit[1] je sada nástrojů pro optimalizaci programu pracující v oblasti počítačového vidění a využívající hluboké učení. Cílem OpenVINO je zrychlení procesu vývoje systémů počítačového vidění a také optimalizace kódu těchto systémů, které vyžadují velké výpočetní náklady pro různé hardwarové platformy (CPU, GPU, FPGA) společnosti Intel. OpenVINO je speciálně navržen pro zrychlení neuronových síti používané v úlohách počítačového vidění jako je klasifikace obrazu nebo detekce objektů. Všechny komponenty OpenVINO jsou poskytovány zdarma.

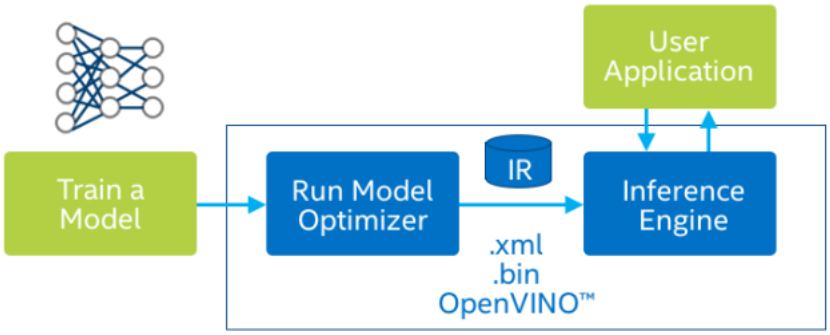
Pro frameworky Caffe, TensorFlow, MXNet a Kaldi je možné přímé konvertování na OpenVINO modely. Převod neuronových sítí z ostatních frameworku, včetně PyTorch, je zatím možný pomocí konvertování modelu do formátu ONNX a až potom do OpenVINO.

ONNX[2] (*Open Neural Network Exchange*) je otevřený formát reprezentace modelů hlubokého učení, podporovány velkými technologickými giganty jako například Microsoft, Facebook a Amazon. Pomocí tohoto formátu mohou být neuronové sítě exportovány z jednoho frameworku do jiného. Vývojáři mohou také díky ONNX převést modely vytvořené a natrénované v PyTorch, Caffe2 a dalších frameworcich a pak je integrovat do svých aplikací.

Framework PyTorch má vlastní rozhraní umožňující převod natrénovaných modelů do formátu ONNX. Po převodu do ONNX je možné optimalizovat model pro rychlý běh na platformách Intel.

Optimalizace neuronových sítí v OpenVINO probíhá za pomoci cross-platformního nástroje, který se nazývá *optimalizátor modelů* (*Model Optimizer*) [35]. Optimalizátor modelů provádí analýzu statických modelů hlubokého učení a upravuje tyto modely pro optimální vykonávání operací na cílových zařízeních. Po zpracování modelu optimalizátor produkuje XML soubor obsahující topologie neuronové sítě a binární soubor obsahující předtrénované

[1] https://software.intel.com/en-us/openvino-toolkit
[2] https://onnx.ai/



Obrázek 5.1: Proces optimalizace modelů a jeho následujícího použití pomocí OpenVINO. Převzato z [30].

parametry sítě. Tyto soubory tvoří takzvaný optimalizovaný *Intermediate Representation* (IR) model.

Běh IR modelů na cílových zařízeních zajišťuje další komponenta sady OpenVINO zvaná *Inference Engine* [31]. *Inference Engine* je C++ knihovnou obsahující C++ třídy nutné pro spouštění a běh IR modelů.

Proces řízení optimalizovaných modelů pomocí *Inference Engine* je zobrazen na obrázku 5.2 a skládá se z následujících kroků.

1. Načtení pluginu, který specifikuje cílové zařízení, na nimž bude fungovat model (CPU, GPU atd.) a další parametry.

2. Načtení IR modelu neuronové sítě, konkrétně souborů s topologií a předtrénovanými parametry.

3. Konfigurace přesností (FP32 pro CPU, FP16 a FP32 pro GPU) a velikosti vstupů a výstupů modelu. V tomto kroku se konfiguruje *batch size* nebo počet obrázků, které bude zpracovávat model za běhu.

4. Načtení modelu se specifikovaným pluginem a vytvoření spustitelné sítě. Spustitelná síť je od tohoto okamžiku nastavená pro konkrétní zařízení a již není možné toto zařízení měnit.

5. Vytváření požadavků na zpracování (*request*). Jeden model může mít libovolné množství požadavků pro zpracování dat.

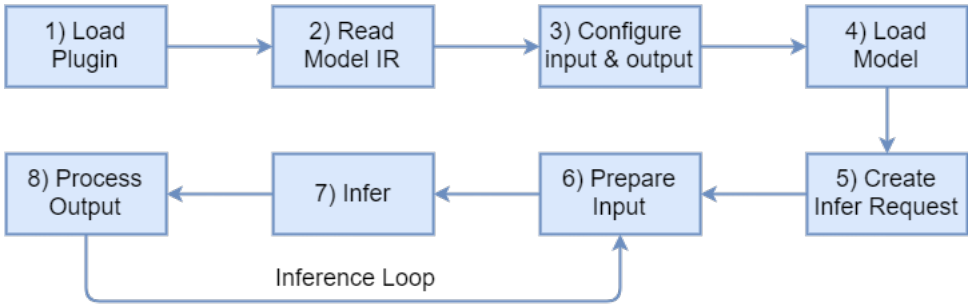
6. Příprava vstupních dat.

7. Zpracování vstupních dat. OpenVINO umožňuje jak synchronní zpracování, tak i asynchronní zpracování vstupních dat.

8. Zpracování výstupů modelu.

Vytvořenou a natrénovanou sledovací síť jsem zoptimalizoval pomocí optimalizátoru modelů, jak je to popsáno výše. Inference Engine poskytuje také Python API [36], ve kterém je možné naprogramovat běh IR modelu v jazyce Python.

Při testování IR modelu sledovací sítě jsem se setkal s problémem, kdy optimalizovaná síť není schopná zpracovávat proměnný počet obličejů ve videích, kvůli přesné nastavenému



Obrázek 5.2: Etapy inicializace optimalizovaných modelů a jejich běhu. Převzato z [32].

batch size v kroku 3. OpenVINO poskytuje podporu pro zpracování proměnného počtu vstupních obrázků (*dynamic batch size*) [37], avšak taková podpora je možná jen pro sítě, které nemohou libovolně měnit dimense vstupních dat. Sledovací síť, vytvořena v rámci této práce, obsahuje *flatten* vrstvu, která transformuje vícedimensionální vektor vstupních dat na jednodimensionální vektor, a proto *dynamic batch size* není pro danou síť možný.

Jako jedna z variant řešení daného problému mě napadlo mít sadu nakonfigurovaných sítí pro různý počet obličejů, kdy v jednom okamžiku je aktivní jen jedna síť, jejíž batch size je rovný počtu vstupních obrázků. Další variantou je zpracovávat obličeje jednou sítí pomocí asynchronního volání množství requestů. Dále tato síť se bude nazývat jako asynchronní síť.

5.2 Kvantizace neuronové sítě

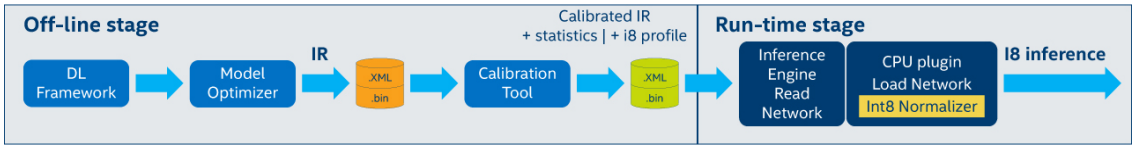
Optimalizovaná síť pro operace s čísly používá datový typ float point s přesností 32 bitu (dále označuje jako FP32) při spouštění na procesorech a datové typy float point s přesností 32 nebo 16 bit (dále označuje jako FP16) při spouštění na grafických kartách [68]. OpenVINO umožňuje *kvantizaci* optimalizovaných sítí pro použití datového typu integer s přesností 8 bit (dále označuje jako I8) na CPU jednotkách [34]. 8-bitové výpočty nabízejí lepší výkon ve srovnání s výpočty vyšší přesnosti (FP16 a FP32), protože umožňují načíst více dat do jedné instrukce procesoru a tím zvětšují počet operací za sekundu prováděných CPU (*Operation Per Second, OPS*). Rychlost výpočtů závisí i na šířce pásma operační paměti a snížení přesnosti umožňuje lepší využití vyrovnávací paměti. Data lze tedy rychleji přesouvat hierarchií paměti tak, aby se maximalizovalo využití výpočetních zdrojů.

Kvantizace optimalizované sítě probíhá za pomoci kalibračního nástrojů, který je součástí OpenVINO, a kalibračního datasetu, který může být malou částí validačního datasetu.

Prvním krokem kvantizace sítě je sbírání statistik přesnosti a maximálních a minimálních hodnot vrstev optimalizované sítě. Dále tento nástroj vyhodnocuje přesnost různých konfigurací celé sítě, v nichž kalibrační nástroj postupně mění přesnost každé vrstvy, podporující 8-bitové výpočty, na I8. Ty vrstvy, jejichž kvantizace vážně snížila přesnost sítě, kalibrační nástroj nechává v její původní přesnosti FP32.

Na konci kvantizace, nástroj ukládá statistiky a informace o kvantizovaných vrstvách do XML souboru optimalizované sítě. Ve fáze načtení sítě, *Inference Engine* používá vygenerované statistiky k normalizaci vah a biasů kvantizovaných vrstev sítě pro 8-bitové výpočty.

U optimalizované sledovací sítě se podařilo kvantizovat všechny konvoluční vrstvy. Snížení výpočetní přesnosti vrstev však může ovlivnit efektivitu sítě. Stejně jako u optimali-



Obrázek 5.3: Kvantizace sítě (*offline stage*) a její následné spouštění a běh (*run-time stage*). Převzato z [33].

zované sítě, implementoval jsem dva způsoby zpracování proměnného počtu obrázků kvantizovanou sítí: použití sady sítí a jedné asynchronní sítě fungujících s přesností 8 bit.

V další podkapitole je popsáno testování optimalizované a kvantizované verzí mého řešení a jejích porovnání s původní implementací ve frameworku PyTorch.

5.3 Vyhodnocení optimalizované sítě

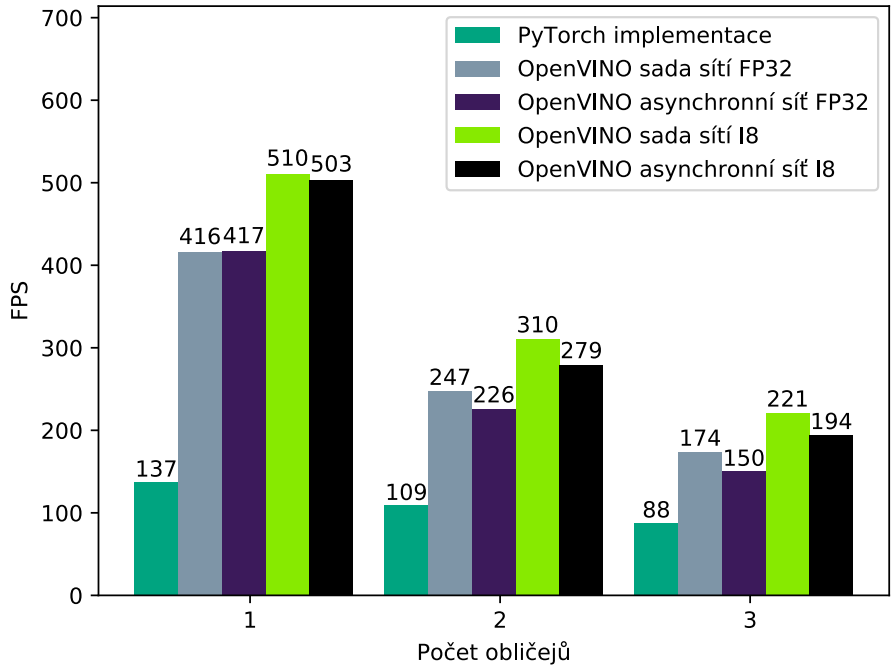
Graf na obrázku 5.4 zobrazuje rychlost zpracování proměnného počtu obličejů původní implementací sítě ve frameworku PyTorch, optimalizovanou a kvantizovanou asynchronními sítěmi a sadami optimalizovaných a kvantizovaných sítí s rozdílným batch size. Z tohoto grafu vyplývá, že při požití sady sítí nebo jedné asynchronní sítě, se rozdíl rychlosti projevuje jen s rostoucím počtem obličejů ve scéně. Bez ohledu na použitou konfiguraci, graf jasně zobrazuje výrazné zrychlení optimalizované sítě oproti její původní implementaci v PyTorch: optimalizace zrychlila sledovací neuronovou síť 2 až 3 krát.

Optimalizovaná síť po kvantizaci ukazuje ještě větší zrychlení zpracování vstupních dat: kvantizace konvolučních vrstev zrychlila optimalizovanou síť o 23% až 27% a kvantizovaná síť je rychlejší původní implementaci 2 až 4-krát.

Z tabulky 5.1, která obsahuje hodnoty AUC implementací sledovacích sítí na *precision/recall*, *succes plot* a *precision plot* grafech, lze vidět, že přesnost sady optimalizovaných a kvantizovaných sítí a sítí s asynchronním zpracováním s různou přesnosti konvolučních vrstev se oproti své implementaci v PyTorch frameworku nezměnila. Naopak, rychlost optimalizovaných a kvantizovaných implementací na celém testovacím datasetu se podle tabulky 5.2 výrazně zvětšila.

Sada optimalizovaných sítí je skoro 3-krát rychlejší něž původní síť a sada kvantizovaných sítí zrychlila zpracování testovacího datasetu o 251 procent. Doba sledování se nepatrně snížila: snížení pro optimalizované sítě je jen pár snímků, zatímco u kvantizovaných jde o deset.

Z těchto grafů a tabulky také vyplývá, že oba způsoby zpracování, jak sada sítí, tak i jedna asynchronní síť, mají skoro identickou přesnost a rychlost, proto je jedno jaký ze způsobů používat pro sledování. Ale vzhledem k tomu, že sada sítí vyžaduje více paměti pro načtení a asynchronní síť spotřebovává více CPU zdrojů, v praxi je potřeba rozhodnout na základě specifiky zařízení pro spouštění sledování.



Obrázek 5.4: Graf naměřených hodnot FPS sledovacích sítí v závislosti na počtu obličejů.

Graf	PyTorch	OpenVINO AS FP32	OpenVINO SS FP32	OpenVINO AS I8	OpenVINO SS I8
Precision/recall	0.7113	0.7158	0.7158	**0.7166**	**0.7166**
Success plot	0.7796	**0.7812**	**0.7812**	0.778	0.778
Precision plot	0.8025	**0.8042**	**0.8042**	0.8029	0.8029

Tabulka 5.1: Tabulka hodnot AUC v grafech měření efektivity a přesnosti sledování obličejů různými implementacemi sledovací sítě. AS znamená asynchronní síť a SS znamená sada sítí.

	Pytorch	OpenVINO AS FP32	OpenVINO SS FP32	OpenVINO AS I8	OpenVINO SS I8
FPS	126	354	359	429	**442**
Počet snímků nepřerušovaného sledování	**141**	139	139	131	131

Tabulka 5.2: Tabulka hodnot rychlosti zpracování snímků a doby sledování obličejů různými implementacemi sledovací sítě na testovacím datasetu. AS znamená asynchronní síť a SS znamená sada sítí.

Kapitola 6

Závěr

Cílem této práce bylo implementovat metodu sledování obličejů schopnou pracovat v reálném čase. Po nastudování příslušné literatury a současných metod sledování objektů byla vytvořena sledovací konvoluční neuronová síť. Tato síť byla implementována v jazyce Python s využitím frameworku PyTorch pro modelování architektury a následného trénování.

Vytvořená síť je schopná sledovat proměnný počet lidských obličejů. Testování sítě na vytvořeném testovacím datasetu ukázalo, že má vysokou přesnost a efektivitu. Síť funguje rychlostí přibližně 126 FPS při testování na datasetu, ve kterém vyskytuje od 1 do 3 lidských obličeje.

V závěru této práce, za pomoci sady nástrojů pro optimalizaci modelů hlubokého učení společnosti Intel, která se nazývá OpenVINO, byla vytvořena optimalizovaná verze sledovací neuronové sítě. Rychlost optimalizované sítě na testovacím datasetu se při sledování jednoho obličeje zvětšila z 137 FPS na 416 FPS. Rychlost sítě na celém testovacím datasetu se zvětšila ze 126 FPS na 359 FPS. Následná kvantizace konvolučních vrstev pomocí OpenVINO zvětšila rychlost sledování optimalizovanou sítí o 23% až 27%.

Vytvořená síť může být také rozšířena o rozpoznávání pohlaví, věku nebo emocí lidí. Síť je možné také zlepšit o sledování rychle pohybujících se obličejů. Rozšíření sítě o práci s 3D prostorem umožňuje například mapování animovaných avatarů na obličej uživatele a také dovoluje usnadnit tvorbu animací mimiky lidského obličeje při vytváření animovaných filmů.

Neuronová sít pro sledování obličejů může najít uplatnění v různých oblastech jako například robotechnika, bezpečnostní systémy nebo rozhraní člověk-počítač. Může být také využita v mobilních aplikacích typu Snapchat nebo v aplikacích pro natáčení vídeí. V posledních letech sledování obličejů nachází uplatnění i v aplikacích virtuální a rozšířené reality.

Literatura

[1] Almeida, A.; Almeida, J.; Rui Araujo: Real-Time Tracking of Moving Objects Using Particle Filters. In *Proceedings of the IEEE International Symposium on Industrial Electronics, 2005. ISIE 2005.*, ročník 4, June 2005, ISSN 2163-5137, s. 1327–1332, doi:10.1109/ISIE.2005.1529124.

[2] Babenko, B.; Yang, M.; Belongie, S.: Visual tracking with online Multiple Instance Learning. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, June 2009, ISSN 1063-6919, s. 983–990.

[3] Bei, S.; Zhen, Z.; Wusheng, L.; aj.: Visual object tracking challenges revisited: VOT vs. OTB. *PLOS ONE*, ročník 13, č. 9, 09 2018: s. 1–12, doi:10.1371/journal.pone.0203188.
URL https://doi.org/10.1371/journal.pone.0203188

[4] Benenson, R.; Omran, M.; Hosang, J. H.; aj.: Ten Years of Pedestrian Detection, What Have We Learned? *CoRR*, ročník abs/1411.4304, 2014, 1411.4304.
URL http://arxiv.org/abs/1411.4304

[5] Bertinetto, L.; Valmadre, J.; Henriques, J. F.; aj.: Fully-Convolutional Siamese Networks for Object Tracking. *CoRR*, ročník abs/1606.09549, 2016, 1606.09549.
URL http://arxiv.org/abs/1606.09549

[6] Bhat, G.; Johnander, J.; Danelljan, M.; aj.: Unveiling the Power of Deep Tracking. *CoRR*, ročník abs/1804.06833, 2018, 1804.06833.
URL http://arxiv.org/abs/1804.06833

[7] Bolme, D. S.; Beveridge, J. R.; Draper, B. A.; aj.: Visual object tracking using adaptive correlation filters. *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2010: s. 2544–2550.

[8] Chen, Z.; Hong, Z.; Tao, D.: An Experimental Survey on Correlation Filter-based Tracking. *CoRR*, ročník abs/1509.05520, 2015, 1509.05520.
URL http://arxiv.org/abs/1509.05520

[9] Cheng, S.; Lin, M. P. Y.; Shen, J.: Mobile Face Tracking: A Survey and Benchmark. *CoRR*, ročník abs/1805.09749, 2018, 1805.09749.
URL http://arxiv.org/abs/1805.09749v1

[10] Cilimkovic, M.: Neural Networks and Back Propagation Algorithm. Technická zpráva, Institute of Technology Blanchardstown, Dublin, Ireland, 2010.

[11] Dickerson, N. L.: *Refining Bounding-Box Regression for Object Localization.* Diplomová práce, Portland State University. Department of Computer Science, 2017.

[12] DinuA., J.; Joseph, F. O. M.: A study on Deep Machine Learning Algorithms for diagnosis of diseases. *International Journal of Applied Engineering Research*, ročník 12, č. 17, 2017: s. 6338–6346.

[13] Doshi, N.: Deep Learning Best Practices (1) — Weight Initialization. 2018, [online; navštíveno 21.10.2018].
URL https://medium.com/usf-msds/deep-learning-best-practices-1-weight-initialization-14e5c0295b94

[14] Doshi, N.: Xavier and He Normal (He-et-al) Initialization. 2018, [online; navštíveno 21.10.2018].
URL https://medium.com/@prateekvishnu/xavier-and-he-normal-he-et-al-initialization-8e3d7a087528

[15] Evgeniou, T.; Pontil, M.: Support Vector Machines: Theory and Applications. In *Machine Learning and Its Applications: Advanced Lectures*, editace G. Paliouras; V. Karkaletsis; C. D. Spyropoulos, Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, ISBN 978-3-540-44673-6, s. 249–257, doi:10.1007/3-540-44673-7_12.
URL https://doi.org/10.1007/3-540-44673-7_12

[16] Fan, H.; Ling, H.: SANet: Structure-Aware Network for Visual Tracking. *CoRR*, ročník abs/1611.06878, 2016, 1611.06878.
URL http://arxiv.org/abs/1611.06878

[17] Farfade, S. S.; Saberian, M. J.; Li, L.: Multi-view Face Detection Using Deep Convolutional Neural Networks. *CoRR*, ročník abs/1502.02766, 2015, 1502.02766.
URL http://arxiv.org/abs/1502.02766

[18] Fasel, B.: An introduction to bio-inspired artificial neural network architectures. *Acta neurologica Belgica*, ročník 103, 04 2003: s. 6–12.

[19] Felzenszwalb, P. F.; Girshick, R. B.; McAllester, D.; aj.: Object Detection with Discriminatively Trained Part-Based Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, ročník 32, č. 9, Sep. 2010: s. 1627–1645, ISSN 0162-8828, doi:10.1109/TPAMI.2009.167.

[20] Fiaz, M.; Mahmood, A.; Jung, S. K.: Tracking Noisy Targets: A Review of Recent Object Tracking Approaches. *CoRR*, ročník abs/1802.03098, 2018, 1802.03098.
URL http://arxiv.org/abs/1802.03098

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

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

[23] Grabner, H.; Grabner, M.; Bischof, H.: Real-Time Tracking via On-line Boosting. *Proceedings of British Machine Vision Conference (BMVC)*, ročník 1, 01 2006: s. 47–56, doi:10.5244/C.20.6.

[24] Guo, G.; Wang, H.; Yan, Y.; aj.: A Fast Face Detection Method via Convolutional Neural Network. *CoRR*, ročník abs/1803.10103, 2018, 1803.10103.
URL http://arxiv.org/abs/1803.10103

[25] Gómez, R.: Understanding Categorical Cross-Entropy Loss, Binary Cross-Entropy Loss, Softmax Loss, Logistic Loss, Focal Loss and all those confusing names. 2018, [online; navštíveno 03.11.2018].
URL https://gombru.github.io/2018/05/23/cross_entropy_loss/

[26] He, Y.; Zhang, X.; Savvides, M.; aj.: Softer-NMS: Rethinking Bounding Box Regression for Accurate Object Detection. *CoRR*, ročník abs/1809.08545, 2018, 1809.08545.
URL http://arxiv.org/abs/1809.08545

[27] Henriques, J. F.; Caseiro, R.; Martins, P.; aj.: High-Speed Tracking with Kernelized Correlation Filters. *CoRR*, ročník abs/1404.7584, 2014, 1404.7584.
URL http://arxiv.org/abs/1404.7584

[28] Huang, C.; Lucey, S.; Ramanan, D.: Learning Policies for Adaptive Tracking with Deep Feature Cascades. *CoRR*, ročník abs/1708.02973, 2017, 1708.02973.
URL http://arxiv.org/abs/1708.02973

[29] Hui, J.: mAP (mean Average Precision) for Object Detection. 2018, [online; navštíveno 23.02.2019].
URL https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173

[30] Intel Corporation: Deep Learning For Computer Vision. [online; navštíveno 17.04.2019].
URL https://software.intel.com/en-us/openvino-toolkit/deep-learning-cv

[31] Intel Corporation: Inference Engine Developer Guide. 2019, [online; navštíveno 28.02.2019].
URL https://docs.openvinotoolkit.org/2019_R1/_docs_IE_DG_Deep_Learning_Inference_Engine_DevGuide.html

[32] Intel Corporation: Integrate the Inference Engine New Request API with Your Application. 2019, [online; navštíveno 17.04.2019].
URL https://docs.openvinotoolkit.org/latest/_docs_IE_DG_Integrate_with_customer_application_new_API.html

[33] Intel Corporation: Introducing int8 quantization for fast CPU inference using OpenVINO. 2019, [online; navštíveno 28.02.2019].
URL https://www.intel.ai/introducing-int8-quantization-for-fast-cpu-inference-using-openvino/#gs.a936t0

[34] Intel Corporation: Low-Precision 8-bit Integer Inference. 2019, [online; navštíveno 28.02.2019].

URL
https://docs.openvinotoolkit.org/latest/_docs_IE_DG_Int8Inference.html

[35] Intel Corporation: Model Optimizer Developer Guide. 2019, [online; navštíveno 28.02.2019].
URL https://docs.openvinotoolkit.org/latest/_docs_MO_DG_Deep_Learning_Model_Optimizer_DevGuide.html

[36] Intel Corporation: Overview of Inference Engine Python API. 2019, [online; navštíveno 28.02.2019].
URL https://docs.openvinotoolkit.org/latest/_inference_engine_ie_bridges_python_docs_api_overview.html

[37] Intel Corporation: Using Dynamic Batching. 2019, [online; navštíveno 28.02.2019].
URL http://docs.openvinotoolkit.org/2019_R1.01/_docs_IE_DG_DynamicBatching.html

[38] Jain, V.; Learned-Miller, E.: FDDB: A Benchmark for Face Detection in Unconstrained Settings. Technická Zpráva UM-CS-2010-009, University of Massachusetts, Amherst, 2010.

[39] Kalal, Z.; Mikolajczyk, K.; Matas, J.: Forward-Backward Error: Automatic Detection of Tracking Failures. In *2010 20th International Conference on Pattern Recognition*, Aug 2010, ISSN 1051-4651, s. 2756–2759, doi:10.1109/ICPR.2010.675.

[40] Kalal, Z.; Mikolajczyk, K.; Matas, J.: Tracking-Learning-Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, ročník 34, č. 7, July 2012: s. 1409–1422, ISSN 0162-8828, doi:10.1109/TPAMI.2011.239.

[41] Kalman, R.: A New Approach To Linear Filtering and Prediction Problems. *Journal of Basic Engineering (ASME)*, ročník 82D, 01 1960: s. 35–45, doi:10.1115/1.3662552.

[42] Kotsiantis, S.: Supervised Machine Learning: A Review of Classification Techniques. *Informatica (Slovenia)*, ročník 31, 01 2007: s. 249–268.

[43] Krizhevsky, A.; Sutskever, I.; E. Hinton, G.: ImageNet Classification with Deep Convolutional Neural Networks. *Neural Information Processing Systems*, ročník 25, 01 2012, doi:10.1145/3065386.

[44] Köstinger, M.; Wohlhart, P.; Roth, P. M.; aj.: Annotated Facial Landmarks in the Wild: A large-scale, real-world database for facial landmark localization. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, Nov 2011, s. 2144–2151, doi:10.1109/ICCVW.2011.6130513.

[45] Lau, S.: A Walkthrough of Convolutional Neural Network — Hyperparameter Tuning. 2017, [online; navštíveno 03.11.2018].
URL https://towardsdatascience.com/a-walkthrough-of-convolutional-neural-network-7f474f91d7bd

[46] Lee, S.; Kwak, S.; Cho, M.: Universal Bounding Box Regression and Its Applications. *CoRR*, ročník abs/1904.06805, 2019, 1904.06805.
URL http://arxiv.org/abs/1904.06805

[47] Li, X.; Hu, W.; Shen, C.; aj.: A Survey of Appearance Models in Visual Object Tracking. *CoRR*, ročník abs/1303.4803, 2013, 1303.4803.
URL http://arxiv.org/abs/1303.4803

[48] Li, Y.: Deep Reinforcement Learning: An Overview. *CoRR*, ročník abs/1701.07274, 2017, 1701.07274.
URL http://arxiv.org/abs/1701.07274

[49] Lin, Y.; Cheng, S.; Shen, J.; aj.: MobiFace: A Novel Dataset for Mobile Face Tracking in the Wild. *CoRR*, ročník abs/1805.09749, 2019, 1805.09749.
URL http://arxiv.org/abs/1805.09749

[50] Liu, Z.; Luo, P.; Wang, X.; aj.: Deep Learning Face Attributes in the Wild. *CoRR*, ročník abs/1411.7766, 2014, 1411.7766.
URL http://arxiv.org/abs/1411.7766

[51] Lucena, M.; Fuertes, J. M.; de la Blanca, N. P.: Using Optical Flow for Tracking. In *Progress in Pattern Recognition, Speech and Image Analysis*, editace A. Sanfeliu; J. Ruiz-Shulcloper, Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, ISBN 978-3-540-24586-5, s. 87–94.

[52] Lukezic, A.; Vojír, T.; Cehovin, L.; aj.: Discriminative Correlation Filter with Channel and Spatial Reliability. *CoRR*, ročník abs/1611.08461, 2016, 1611.08461.
URL http://arxiv.org/abs/1611.08461

[53] Luo, W.; Zhao, X.; Kim, T.: Multiple Object Tracking: A Review. *CoRR*, ročník abs/1409.7618, 2014, 1409.7618.
URL http://arxiv.org/abs/1409.7618

[54] Maniyath, S. R.; VinodP, V.; Niveditha, M.; aj.: Plant Disease Detection Using Machine Learning. *2018 International Conference on Design Innovations for 3Cs Compute Communicate Control (ICDI3C)*, 2018: s. 41–45.

[55] Manohar, V.; Soundararajan, P.; Raju, H.; aj.: Performance Evaluation of Object Detection and Tracking in Video. In *Computer Vision – ACCV 2006*, editace P. J. Narayanan; S. K. Nayar; H.-Y. Shum, Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, ISBN 978-3-540-32432-4, s. 151–161.

[56] Mathias, M.; Benenson, R.; Pedersoli, M.; aj.: Face Detection without Bells and Whistles. *LNCS*, ročník 8692, 09 2014, doi:10.1007/978-3-319-10593-2_47.

[57] Nam, H.; Han, B.: Learning Multi-Domain Convolutional Neural Networks for Visual Tracking. *CoRR*, ročník abs/1510.07945, 2015, 1510.07945.
URL http://arxiv.org/abs/1510.07945

[58] Ning, G.; Zhang, Z.; Huang, C.; aj.: Spatially Supervised Recurrent Convolutional Neural Networks for Visual Object Tracking. *CoRR*, ročník abs/1607.05781, 2016, 1607.05781.
URL http://arxiv.org/abs/1607.05781

[59] Nwankpa, C.; Ijomah, W.; Gachagan, A.; aj.: Activation Functions: Comparison of trends in Practice and Research for Deep Learning. *CoRR*, ročník abs/1811.03378,

2018, 1811.03378.
URL http://arxiv.org/abs/1811.03378

[60] Oken, A.: An Introduction To and Applications of Neural Networks. 2017.

[61] O'Shea, K.; Nash, R.: An Introduction to Convolutional Neural Networks. *CoRR*, ročník abs/1511.08458, 2015, 1511.08458.
URL http://arxiv.org/abs/1511.08458

[62] Parag, T.; Porikli, F.; Elgammal, A.: Boosting adaptive linear weak classifiers for online learning and tracking. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, June 2008, ISSN 1063-6919, s. 1–8, doi:10.1109/CVPR.2008.4587556.

[63] Pflugfelder, R. P.: Siamese Learning Visual Tracking: A Survey. *CoRR*, ročník abs/1707.00569, 2017, 1707.00569.
URL http://arxiv.org/abs/1707.00569

[64] Ramalakshmi, V.; Germanus Alex, M.: A Survey on Visual Object Tracking: Datasets, Methods and Metrics. *International Research Journal of Engineering and Technology*, 2016.

[65] Ranjan, R.; Patel, V. M.; Chellappa, R.: HyperFace: A Deep Multi-task Learning Framework for Face Detection, Landmark Localization, Pose Estimation, and Gender Recognition. *CoRR*, ročník abs/1603.01249, 2016, 1603.01249.
URL http://arxiv.org/abs/1603.01249

[66] Redmon, J.; Farhadi, A.: YOLOv3: An Incremental Improvement. *CoRR*, ročník abs/1804.02767, 2018, 1804.02767.
URL http://arxiv.org/abs/1804.02767

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

[68] Rodriguez, A.; Segal, E.; Meiri, E.; aj.: Lower Numerical Precision Deep Learning Inference and Training. Technická zpráva, Intel Corporation, 2018.

[69] S, P.; U M, M.; Michahial, S.; aj.: Human Motion Detection and Tracking for Real-Time Security System. *IJARCCE*, ročník 5, 12 2016: s. 203–207, doi:10.17148/IJARCCE.2016.51245.

[70] Saadna, Y.; Behloul, A.: An overview of traffic sign detection and classification methods. *International Journal of Multimedia Information Retrieval*, ročník 6, č. 3, Sep 2017: s. 193–210, ISSN 2192-662X, doi:10.1007/s13735-017-0129-8.
URL https://doi.org/10.1007/s13735-017-0129-8

[71] Sherstinsky, A.: Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network. *CoRR*, ročník abs/1808.03314, 2018, 1808.03314.
URL http://arxiv.org/abs/1808.03314

[72] Simonyan, K.; Zisserman, A.: Very Deep Convolutional Networks for Large-Scale Image Recognition. *CoRR*, ročník abs/1409.1556, 2014.

[73] Srivastava, N.; Hinton, G.; Krizhevsky, A.; aj.: Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, ročník 15, 06 2014: s. 1929–1958.

[74] Tao, R.; Gavves, E.; Smeulders, A. W. M.: Siamese Instance Search for Tracking. *CoRR*, ročník abs/1605.05863, 2016, 1605.05863.
URL http://arxiv.org/abs/1605.05863

[75] Ulusoy, I.; Bishop, C. M.: Comparison of Generative and Discriminative Techniques for Object Detection and Classification. In *Toward Category-Level Object Recognition*, editace J. Ponce; M. Hebert; C. Schmid; A. Zisserman, Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, ISBN 978-3-540-68795-5, s. 173–195, doi:10.1007/11957959_9.
URL https://doi.org/10.1007/11957959_9

[76] Wang, L.; Ouyang, W.; Wang, X.; aj.: Visual Tracking with Fully Convolutional Networks. In *2015 IEEE International Conference on Computer Vision (ICCV)*, Dec 2015, ISSN 2380-7504, s. 3119–3127, doi:10.1109/ICCV.2015.357.

[77] Wang, Q.; Gao, J.; Xing, J.; aj.: DCFNet: Discriminant Correlation Filters Network for Visual Tracking. *CoRR*, ročník abs/1704.04057, 2017, 1704.04057.
URL http://arxiv.org/abs/1704.04057

[78] Weng, L.: Object Detection for Dummies Part 3: R-CNN Family. 2017, [online; navštíveno 07.04.2019].
URL https://lilianweng.github.io/lil-log/2017/12/31/object-recognition-for-dummies-part-3.html

[79] Xiang, Y.; Alahi, A.; Savarese, S.: Learning to Track: Online Multi-object Tracking by Decision Making. In *2015 IEEE International Conference on Computer Vision (ICCV)*, Dec 2015, ISSN 2380-7504, s. 4705–4713, doi:10.1109/ICCV.2015.534.

[80] Yan, J.; Zhang, X.; Lei, Z.; aj.: Face Detection by Structural Models. *Image and Vision Computing*, ročník 32, 01 2013, doi:10.1016/j.imavis.2013.12.004.

[81] Yang, B.; Yan, J.; Lei, Z.; aj.: Fine-grained Evaluation on Face Detection in the Wild. In *Automatic Face and Gesture Recognition (FG), 11th IEEE International Conference on*, IEEE, 2015.

[82] Yang, H.; Wang, X. A.: Cascade classifier for face detection. *Journal of Algorithms & Computational Technology*, ročník 10, č. 3, 2016: s. 187–197.

[83] Yang, S.; Luo, P.; Loy, C. C.; aj.: WIDER FACE: A Face Detection Benchmark. *CoRR*, ročník abs/1511.06523, 2015, 1511.06523.
URL http://arxiv.org/abs/1511.06523

[84] Yu, J.; Jiang, Y.; Wang, Z.; aj.: UnitBox: An Advanced Object Detection Network. *CoRR*, ročník abs/1608.01471, 2016, 1608.01471.
URL http://arxiv.org/abs/1608.01471

[85] Yun, S.; Choi, J.; Yoo, Y.; aj.: Action-Decision Networks for Visual Tracking with Deep Reinforcement Learning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017, ISSN 1063-6919, s. 1349–1358, doi:10.1109/CVPR.2017.148.

[86] Zhai, M.; Roshtkhari, M. J.; Mori, G.: Deep Learning of Appearance Models for Online Object Tracking. *CoRR*, ročník abs/1607.02568, 2016, 1607.02568.
URL http://arxiv.org/abs/1607.02568

[87] Zhang, K.; Zhang, Z.; Li, Z.; aj.: Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks. *CoRR*, ročník abs/1604.02878, 2016, 1604.02878.
URL http://arxiv.org/abs/1604.02878

[88] Zhang, Y.; Yang, Q.: A Survey on Multi-Task Learning. *CoRR*, ročník abs/1707.08114, 2017, 1707.08114.
URL http://arxiv.org/abs/1707.08114

[89] Zhu, C.; Zheng, Y.; Luu, K.; aj.: CMS-RCNN: Contextual Multi-Scale Region-based CNN for Unconstrained Face Detection. *CoRR*, ročník abs/1606.05413, 2016, 1606.05413.
URL http://arxiv.org/abs/1606.05413

[90] Zulkifli, H.: Understanding Learning Rates and How It Improves Performance in Deep Learning. 2018, [online; navštíveno 15.03.2019].
URL https://towardsdatascience.com/understanding-learning-rates-and-how-it-improves-performance-in-deep-learning-d0d4059c1c10

Příloha A

Obsah přiloženého paměťového média

dataset obsahuje testovací dataset

prepared obsahuje soubory s předtrénovaným, optimalizovaným a kvantizovaným modely sítí a výstupy trackerů, které byly využity k vygenerování výsledných grafů

doc obsahuje zdrojové soubory k technické dokumentaci bakalářské práce a již přeložené PDF

video obsahuje video, prezentující cíle a výsledky práce

rtft obsahuje zdrojové soubory, které byly použity k přípravě dat, trénování, optimalizaci a testování neuronových sítí

README.md soubor s podrobnějším popisem jednotlivých souborů na disku DVD a návod k použití skriptů projektu

reqs.txt seznam knihoven a balíčků potřebných k přípravě dat, operacím s neuronovými sítěmi a generování grafů

run.sh skript pro ovládaní a spouštění skriptů pro přípravu dat, operace s neuronovými sítěmi a generování grafů