



**VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ**  
BRNO UNIVERSITY OF TECHNOLOGY

**FAKULTA INFORMAČNÍCH TECHNOLOGIÍ**  
FACULTY OF INFORMATION TECHNOLOGY

**ÚSTAV INTELIGENTNÍCH SYSTÉMŮ**  
DEPARTMENT OF INTELLIGENT SYSTEMS

**AKCELERACE NEURONOVÉ SÍTĚ PRO DETEKCII OB-  
LIČEJE VE ZHORŠENÝCH SVĚTELNÝCH PODMÍNKÁCH**  
ACCELERATION OF A NEURAL NETWORK FOR FACE DETECTION IN LOW LIGHT CONDITI-  
ONS

**BAKALÁŘSKÁ PRÁCE**  
BACHELOR'S THESIS

**AUTOR PRÁCE**  
AUTHOR

**VOJTECH ORAVA**

**VEDOUCÍ PRÁCE**  
SUPERVISOR

**Ing. TOMÁŠ GOLDMANN**

**BRNO 2023**

## Zadání bakalářské práce



147660

Ústav: Ústav inteligentních systémů (UITS)  
Student: **Orava Vojtěch**  
Program: Informační technologie  
Specializace: Informační technologie  
Název: **Akcelerace neuronové sítě pro detekci obličeje ve zhoršených světelných podmínkách**  
Kategorie: Umělá inteligence  
Akademický rok: 2022/23

### Zadání:

1. Seznamte se s problematikou detekce obličeje v reálných podmínkách, především se zaměřte na problémy spojené se špatnými světelnými podmínkami.
2. Sumarujte informace o detektorech obličeje založených na neuronových sítích a zjistěte možnosti jejich akcelerace. Především se zaměřte na akcelerační platformu Intel Compute Neural Stick 2.
3. Navrhněte řešení s využitím vlastní neuronové sítě pro detekci obličeje ve videu, které je pořizováno za špatných světelných podmínek.
4. Navržený algoritmus implementujte v programovacím jazyce Python a provedete jeho akceleraci pomocí Intel Compute Neural Stick 2.
5. Provedete experimenty zaměřené na porovnání vašeho detektoru s již existujícími detektory. Zhodnotte rozdíl v rychlosti mezi akcelEROvaným a neakcelEROvaným řešením.

### Literatura:

- MASI, Iacopo, et al. Deep face recognition: A survey. In: *2018 31st SIBGRAPI conference on graphics, patterns and images (SIBGRAPI)*. IEEE, 2018. p. 471-478.
- TRIGUEROS, Daniel Sáez; MENG, Li; HARTNETT, Margaret. Face recognition: From traditional to deep learning methods. *arXiv preprint arXiv:1811.00116*, 2018.
- DI NARDO, E.; PETROSINO, A.; SANTOPIETRO, V. Embedded Deep Learning for Face Detection and Emotion Recognition with Intel Movidius (TM) Neural Compute Stick. 2018.

Při obhajobě semestrální části projektu je požadováno:

Body 1 a 2.

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

Vedoucí práce: **Goldmann Tomáš, Ing.**

Vedoucí ústavu: Hanáček Petr, doc. Dr. Ing.

Datum zadání: 1.11.2022

Termín pro odevzdání: 10.5.2023

Datum schválení: 3.11.2022

## Abstrakt

Cílem této práce je vytvořit neuronovou síť pro detekci obličejů ve zhoršených světelných podmínkách, tuto síť akcelerovat a porovnat s existujícími řešeními. Problém detekce je řešen konvoluční neuronovou sítí natrénovanou pomocí datasetů WIDER FACE a DARK FACE. Síť je akcelerována zařízením Intel Neural Compute Stick 2. Práce obsahuje sumarizaci dosavadních řešení a algoritmů detekce (jak klasických metod, tak těch využívajících neuronové sítě) a poskytuje porovnání těchto řešení s nově vytvořenými detektory. Ty jsou založeny na TensorFlow Object Detection API. Nejlepší varianta dosahuje průměrné přesnosti 47,1 % na upraveném validačním datasetu WIDER FACE (YOLOv7-face 42,8 %). Měřena byla také rychlosť a vliv akcelerace a kvantování modelů. Kvantováním bylo dosaženo i více než 3násobného zrychlení některých modelů. Součástí práce je také grafická aplikace pro detekci ve videích s vytvořenými modely a některými existujícími řešeními.

## Abstract

The goal of this paper is to build neural network for face detection in low light conditions, accelerate this network and compare it with some other existing networks. Detection problem is solved with convolution neural network (CNN), which is trained on datas from face datasets. This CNN is accelerated by device Intel Neural Compute Stick 2. This work also summarise existing approaches in face detection (classic and neural networks based) and compares this approaches.

## Klíčová slova

Detekce obličeje, akcelerace neuronových sítí, NCS 2, detekce ve zhoršených světelných podmínkách, neuronové sítě, Python, počítačové vidění, WIDER FACE, DARK FACE, strojové učení

## Keywords

Face detection, Neural Networks acceleration, NCS 2, Detection in Low Light Conditions, Neural Networks, Python, Computer Vision, WIDER FACE, DARK FACE, Machine Learning

## Citace

ORAVA, Vojtěch. *AKCELERACE NEURONOVÉ SÍTĚ PRO DETEKCI OBLIČEJE VE ZHORŠENÝCH SVĚTELNÝCH PODMÍNKÁCH*. Brno, 2023. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Tomáš Goldmann

# AKCELERACE NEURONOVÉ SÍTĚ PRO DETEKCI OBLIČEJE VE ZHORŠENÝCH SVĚTELNÝCH PODMÍNKÁCH

## Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Tomáše Goldmanna. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....  
Vojtěch Orava  
18. dubna 2023

## Poděkování

Chtěl bych poděkovat panu Ing. Tomáši Goldmannovi za vedení práce, poskytnuté konzultace a cenné rady. Také chci poděkovat rodině a přátelům za podporu při studiu.

# Obsah

<b>1</b>	<b>Úvod</b>	<b>3</b>
<b>2</b>	<b>Detekce obličeje v reálných podmírkách</b>	<b>5</b>
2.1	Detekce obličeje . . . . .	6
2.2	Problémy a omezení . . . . .	7
2.3	Algoritmy detekce obličeje . . . . .	9
<b>3</b>	<b>Neuronové sítě pro detekci obličeje</b>	<b>14</b>
3.1	Neuronové sítě . . . . .	14
3.2	Datasetsy . . . . .	17
3.3	Detektory obličeje . . . . .	18
3.4	Frameworky pro neuronové sítě . . . . .	24
<b>4</b>	<b>Systémy pro detekci obličejů a akcelerace detekce</b>	<b>25</b>
4.1	Kamery . . . . .	25
4.2	Dostupná řešení . . . . .	25
4.3	Akcelerace detekčních algoritmů . . . . .	26
<b>5</b>	<b>Návrh řešení</b>	<b>29</b>
5.1	Použité nástroje . . . . .	29
5.2	Data . . . . .	30
5.3	Detekční neuronová síť . . . . .	31
5.4	Uživatelské rozhraní . . . . .	33
<b>6</b>	<b>Implementace</b>	<b>35</b>
6.1	Příprava dat . . . . .	35
6.2	Detektor obličejů . . . . .	38
6.3	Trénování . . . . .	40
6.4	Akcelerace detekce . . . . .	42
6.5	Grafické uživatelské rozhraní . . . . .	43
6.6	Nástroje pro experimenty . . . . .	46
<b>7</b>	<b>Experimenty</b>	<b>50</b>
7.1	Podklady . . . . .	50
7.2	Porovnání akcelEROVANÉHO a normálního řešení . . . . .	52
7.3	Porovnání s YOLOv7-face . . . . .	60
7.4	Porovnání s Viola-Jones . . . . .	61
7.5	Porovnání s MTCNN . . . . .	61

7.6	Detekce s metodou Mirnet . . . . .	63
7.7	Zhodnocení experimentů . . . . .	64
<b>8</b>	<b>Závěr</b>	<b>66</b>
<b>A</b>	<b>Obsah přiloženého datového média</b>	<b>67</b>

# Kapitola 1

## Úvod

Neuronové sítě (anglicky *neural networks*) mají v dnešním světě mnoho využití. Jelikož se jedná o jednu z aplikací umělé inteligence (anglicky *artificial intelligence*), lze neuronové sítě použít například k rozpoznávání řeči, zpracování přirozeného jazyka či k detekci obličejů. Tyto činnosti jsou pro běžného člověka poměrně snadné, avšak pro počítače znamenají relativně náročnou činnost.

Lidé jsou schopni velmi dobře rozeznat obličeje jiných lidí, bez ohledu na světlé podmínky, úhel natočení či částečné zakrytí tváře. Počítačové algoritmy a neuronové sítě zaměřující se na detekci obličejů v těchto neideálních podmínkách musí být těmto jevům přizpůsobeny.

Aby byly počítače schopné tyto akce vykonávat v rozumném čase (případně v reálném čase), je potřeba, aby neuronové sítě byly dostatečně rychlé. Zrychlení neuronové sítě lze dosáhnout buď optimalizací kódu, vylepšením procesu trénování neuronové sítě nebo také využitím speciálních hardwarových zařízení.

Tato práce se zabývá problematikou výše zmíněných fenoménů, konkrétně optimalizací a zrychlením neuronových sítí pro detekci obličejů v neoptimálních světlých podmínkách. Z oblasti neuronových sítí byly použity konvoluční neuronové sítě, k jejichž trénování je využito jak dat z datasetu WIDER FACE, tak dat ze specializovaného datasetu DARK FACE.

V kapitole 2 je popsána problematika detekce obličeje v reálných podmínkách, včetně problémů, které detekci ztěžují či přímo znemožňují. Tato část také popisuje klasické přístupy k detekci obličejů jako jsou algoritmy Viola-Jones nebo Local Binary Patterns.

Kapitola 3 se věnuje neuronovým sítím jak obecně (perceptron, aktivační funkce, učení), tak konkrétně konvolučním neuronovým sítím. Dále jsou v kapitole popsány datasety pro učení neuronových sítí a existující řešení detekce obličeje založené na neuronových sítích (YOLO, MTCNN, SSD), včetně jejich porovnání a porovnání algoritmů zaměřených na detekci ve špatných světlých podmínkách.

Popis používaných řešení a systémů k detekci s popisem nástrojů a přístupů k akceleraci detekčních algoritmů tvoří obsah kapitoly 4. Konkrétně je zde popsáno zařízení Intel Neural Compute Stick 2 a optimalizační knihovna OpenVINO.

S přechodem do praktické části práce začíná kapitola 5. Ta charakterizuje využité nástroje k tvorbě, trénování a akceleraci modelů. Následuje návrh aplikace s grafickým uživatelským rozhraním pro testování detektorů.

Implementací se zabývající kapitola 6 je věnována přípravě dat (augmentace), výběru a trénování detekčních modelů. Modely vznikly během práce celkem 4, z nichž byly později vytvořeny akcelEROVANÉ a kvANTOVANÉ verze. Část kapitoly je věnována praktickému využití

akcelerace (knihovna OpenVINO a kvantování modelů). Popisu aplikace pro detekci a konstrukci nástrojů pro provádění experimentů je věnován závěr této kapitoly. Je zde i část věnovaná vyhodnocovacím metrikám a výpočtům spojených s testováním.

Na vyhodnocení úspěšnosti detekce ve 2 kategoriích (fotky a videa) je zaměřena kapitola 7. Z výsledků vyplývá, že nejlepší nově natrénovaný model dosahuje přesnosti detekce vyšší než existující řešení jako jsou YOLOv7-face nebo MTCNN. V rámci testů se také projevil vliv akcelerace, zejména pak kvantování. Některé modely byly zrychleny  $3\times$  až  $4\times$  při minimálních ztrátách přesnosti.

## Kapitola 2

# Detekce obličeje v reálných podmírkách

Detekce obličeje (anglicky *face detection*) [?, ?] je technologie, která umožňuje v digitálním obrázku lokalizovat lidský obličej. Detekce obličeje patří do skupiny technologií HCI (Human – Computer interaction). Detektovat obličeje je sandná pro člověka, ale zároveň se jedná o relativně náročný úkol pro počítač. Detekce obličeje je výchozím bodem pro další algoritmy analyzující lidský obličej, jako jsou například (v závorce za pojmem následuje anglický výraz):

- rozpoznávání obličeje (*face recognition*),
- zarovnání obličeje (*face alignment*),
- autentizace pomocí obličeje (*face verification/authentication*),
- sledování pohybu hlavy (*head pose tracking*),
- určování věku nebo pohlaví (*age/gender recognition*),

a mnoho dalších.

Samotná detekce obličeje se v realním prostředí využívá například v oblasti fotografování (automatické ostření na tvář), marketingu (zjišťování zájmu zákazníku o produkty podle počtu výskytu obličejů) nebo bezpečnosti (bezpečnostní kamery a systémy).

Následující podkapitoly se zabývají principem fungování detekce obličeje v reálných podmírkách a problémy a omezeními, které se v běžném světě vyskytují a detekce by si s nimi měla umět poradit (špatné světelné podmínky, příliš členité pozadí, přílišný počet obličejů v obrázku, barva kůže, nízké rozlišení atd.). Na konci této kapitoly se nachází souhrn klasických algoritmů a detektorů obličejů.



Obrázek 2.1: Příklad zvýraznění detekce obličeje. Obrázek převzat z datasetu WIDER FACE.

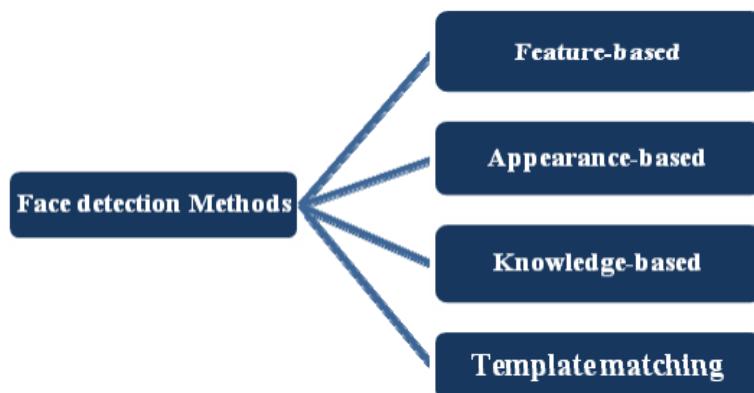
## 2.1 Detekce obličeje

Způsoby detekce obličeje lze rozdělit do několika skupin, vědecké práce na toto téma se liší a nelze jasně říci zda to či ono dělení je jediné korektní. Často bývají detekované obličeje ohrazeny obdélníkem jako na obrázku 2.1.

Dle [?] existují 2 různé přístupy k hledání tváří v obrázcích, a to **přístup založený na vlastnostech** (anglicky feature based approach) a **přístup založený na obrázku** (anglicky image based approach). **Přístup založený na vlastnostech** nepoužívá přímo k detekci obličeje neuronové sítě. Využívá vlastnosti obličeje jako takového (rysy, pozice očí, uší, obočí, barva kůže...). Efektivita tohoto přístupu se snižuje s výskyty problémů popsaných v sekci 2.2, protože může docházet například k zakrytí nebo špatné viditelnosti některých vlastností obličeje.

Naproti tomu **obrazový přístup** uplatňuje schopnosti neuronových sítí a umělé inteligence k natrénování modelu neuronové sítě a následné přímé detekci pomocí tohoto modelu.

Podle [?] lze rozdělit metody detekce obličeje do 4 základních kategorií (viz obrázek 2.2) a 2 zvláštních kategorií (Haarovy vlastnosti a umělá inteligence).



Obrázek 2.2: Dělení metod detekce obličeje dle [?].

**Feature-based methods** (metody založené na vlastnostech) opět pracují s vlastnostmi obličeje. Vyhledávají v obraze rysy obličeje. Detekci může výrazně ztížit či znemožnit neviditelnost některých rysů. Výhodou těchto metod je rychlosť v porovnání s ostatními metodami.

**Appearance-based methods** (metody založené na vzhledu/obrázku) využívají klasifikace vlastností tváře do 2 tříd v podobrázku celého obrázku. Klasifikátory podle nichž se daná metoda rozhoduje, zda se jedná o tvář či nikoli, mají různé váhy, metoda postupuje od slabých klasifikátorů k silnějším.

Metody založené na znalostech (**Knowledge-based methods**) se uplatňují při detekci obličeje v obrázku s členitým/komplexním pozadím. Znalosti, které k detekci pomáhají jsou například: obličej má 2 uši, jeden nos, jedny ústa nebo známé vzdálenosti mezi jednotlivými rysy obličeje.

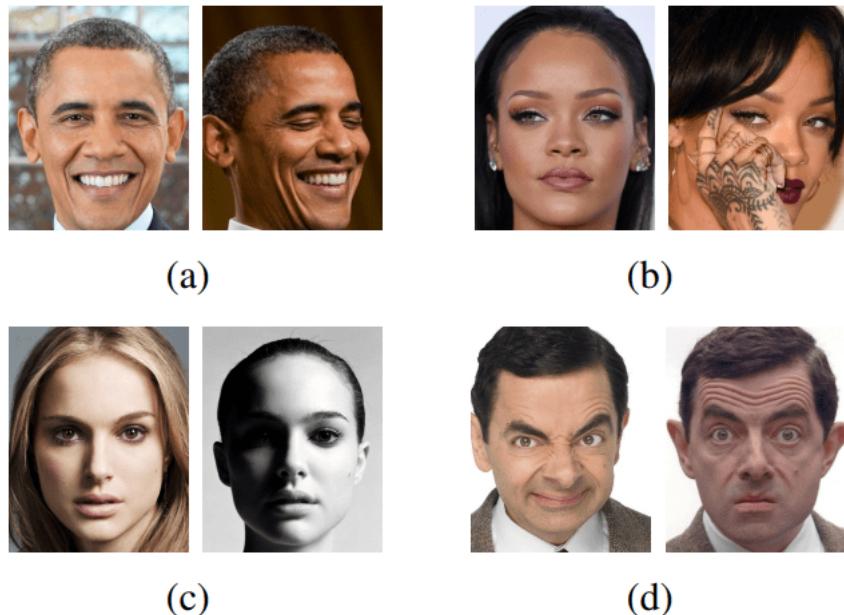
**Templatematching** (šablonování, maskování) aplikuje na obrázek předem danou masku obličeje a snaží se detektovat obličeje pomocí postupného maskování. Tato metoda je snadná na implementaci, jejím nedostatkem je však závislost na přímém pohledu tváře na obrázku.

Haarovými vlastnostmi a neuronovými sítěmi se zabývají sekce 2.3, respektive 3.1.

## 2.2 Problémy a omezení

Algoritmy pro detekci obličeji čelí několika výzvám a omezením spojených s ne vždy perfektním zobrazením obličeje v obrázku. Lidské obličeje na fotografiích a obrázcích mohou být částečně zakryté (např. sluneční brýle), fotografie mohou být pořízené za nevhodných světelných podmínek (např. zastínění části tváře) nebo mohou obrázky nabývat nedostatečné kvality (nízké rozlišení).

Jelikož tedy vstupní obrázek detekce obličeje nemusí být vždy ideální, nemusí být obličeji vždy správně detektován. V této sekci jsou popsány některé problémy [?, ?], které mohou bránit v úspěšné detekci. Minimalizace dopadu těchto jevů na detekci je klíčem k navýšení uspěšnosti detekce. Mezi problémy a omezení (viz obrázek 2.3) pro detekci patří pozice hlavy, zakrytí části/částí obličeje, špatně osvětlená scéna nebo výraz tváře.



Obrázek 2.3: Některé problémy vyskytující se při detekci obličejů. Převzato z [?]. (a) Pozice hlavy; (b) Zakrytí části obličeje; (c) Špatné světelné podmínky; (d) Výraz tváře.

### Pozice hlavy

Hlava může být na fotografii různě natočena, takže obličej nemusí být zachycen v přímém pohledu do kamery, ale může být zaznamenán z profilu nebo ze šikma (poloprofil) jako na obrázku 2.3 část (a).

### Zakrytí části obličeje

Výsledek detekce může být ovlivněn i zakrytím části obličeje (rukou, brýlemi, vlasy, šátkem apod.).

### Výraz tváře

Výraz lidského obličeje mohou ovlivnit emoce a nálady. Detektor [?] zabývající se detekcí a rozpoznáváním emocí dosahuje přesnosti 96 %.

### Orientace obrázku

Problémem pro detekci může být různá orientace obrázku (vzhůru nohama, zrcadlově otočený, natočený do strany apod.). Na obrázek je tak nutno aplikovat některé transformační operace pro zarovnání.

### Nedostatečně výkonná detekce

Velmi důležitým faktorem při detekci obličejů, zvláště v real-timových aplikacích, je rychlosť detekce. Pokud má algoritmu vysokou přesnost, ale je příliš pomalý pro vybranou aplikaci, stává se nepoužitelným. Akcelerací detekčních algoritmů se mj. zabývá i tato práce.

## Příliš členité pozadí

Pokud se v obrázku nachází příliš mnoho objektů, může dojít ke snížení přesnosti a rychlosti detekce.

## Přílišný počet obličejů v obrázku

Výskyt velkého počtu obličejů v jednom obrázku, často překrývajících se, může představovat problém pro detekční algoritmus.

## Nízké rozlišení

Obrázky a fotografie s nízkým rozlišením nemusejí obsahovat dostatek informace nutné ke správnemu detekování tváře.

## Špatné světelné podmínky

Na detekci mohou mít vliv světelné podmínky panující při pořizování zkoumané fotografie či videa. Aspekty jež světlo ovlivňuje jsou mj. jas, kontrast, barvy, stíny, ostrost. Tato práce se zabývá detekcí obličejů v záznamech, v nichž některý z těchto faktorů omezuje detekci.



Obrázek 2.4: Příklad stejného obličeje vyfoceného při různých světelných podmínkách. Převzato z [?].

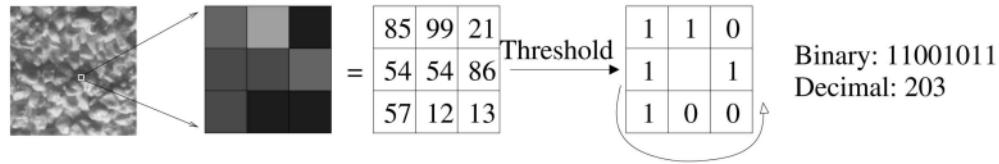
## 2.3 Algoritmy detekce obličeje

Existuje několik různých přístupů k detekci obličeje. Tato sekce se zaměřuje na detekci s využitím detektorů založených primárně ne na neuronových sítích (neuronová síť není použita vůbec, nebo není použita k přímé detekci). Detektory obličeje využívající principy neuronových sítí k přímé detekci jsou popsány v sekci 3.3.

### Local Binary Patterns

Algoritmus využívající **lokální binární vzory** (anglicky *Local Binary Patterns* – dále jen LBP) pro popis struktury/textury obrázku má mnoho aplikací [?]. Jedná se o jeden z nejvýkonnějších algoritmů pro popis textur v obrázcích. LBP algoritmus má vysokou účinnost detekce (89 %) [?] a nízkou výpočetní náročnost. LBP pracuje s černobílými (anglicky *gray-scale*) obrázky a v původní verzi funguje tak, že každému pixelu přiřadí binární číselnou hodnotu, vypočítanou dle hodnot pixelů v  $3 \times 3$  okolí daného pixelu. Každý takovýto pixel

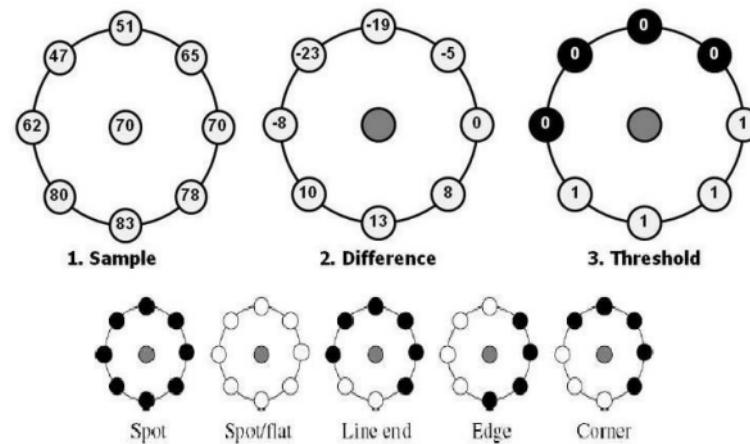
v okolí je ohodnocen hodnotou buď 1 nebo 0 v závislosti na tom, zda jeho hodnota překročila stanovený práh (anglicky *threshold*), který odpovídá hodnotě prostředního pixelu (viz obrázek 2.5).



Obrázek 2.5: Ukázka ohodnocení pixelu algoritmem LBP. Jako práh je využita hodnota středního pixelu. Převzato z [?].

Algoritmus byl později vylepšen tak, aby uměl pracovat s různě velkým okolím, které navíc nemusí mít čtvercový tvar. Používané okolí ve tvaru kružnice je popsáno dvojicí  $(P, R)$ , kde  $P$  je počet vzorkovacích bodů a  $R$  je poloměr kružnice. Další vylepšení LBP algoritmu se týkalo definování tzv. uniformních vzorů (anglicky uniform patterns). Tyto vzory jsou ty vzory v nichž se vyskytuje nejvýše 2 přechody z 1 na 0 a opačně. Příkladem uniformního vzoru na okolí  $(8, 2)$  je 11100000 (1 přechod), či 00111000 (2 přechody).

Lokální primitiva (obrázek 2.6) a histogramy vytvořené z takto získaných hodnot se využívají mj. k detekci obličejů.

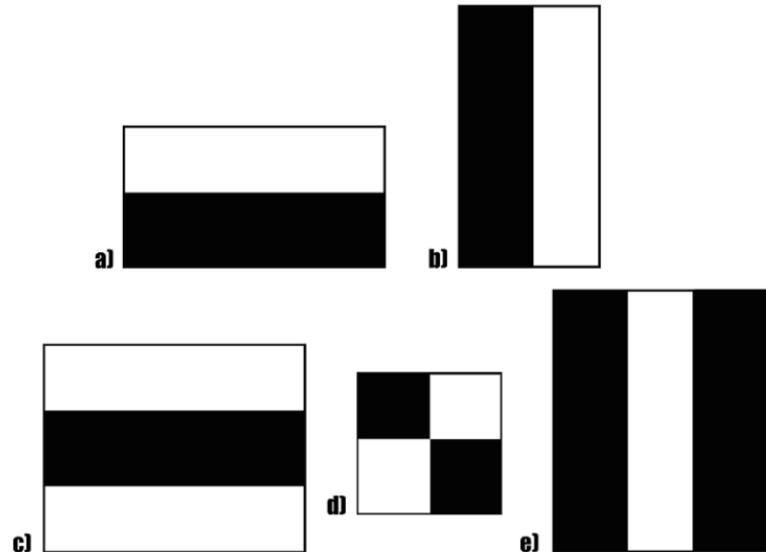


Obrázek 2.6: Lokální primitiva detekovaná algoritmem LBP s využitím okolí ve tvaru kružnice  $(P, R)$ . Převzato z [?].

### Viola-Jones algoritmus

Algoritmus **Viola-Jones** [?, ?], někdy také nazývaný **Haar Cascades**, je obecně technika pro detekci objektů, vytvořená před používáním metod hlubokého učení. Používá se k detekci obličejů, částí těla, očí, úst atd. V určitých aplikacích dosahuje přesnosti detekce kolem 90 % [?].

Princip fungování algoritmu spočívá v detekci hran a čár (obecně vlastností) v černobílém obrázku (hodnoty pixelů jsou na intervalu  $< 0; 1 >$ ). Vybere se jedna z vlastností (některé z nich zobrazuje obrázek 2.7) a vypočítá se průměrná hodnota pixelů ve všech obdélnících (obdélníky jsou dva, tři nebo čtyři). Rozdíl těchto hodnot pak určuje zda se jedná o hranu, čáru tzn. zda je daná vlastnost detekována. Pokud například budeme hledat vlastnost b) z obrázku 2.7 a vypočtený rozdíl hodnot je blízký 1, detekce byla úspěšná (viz ukázka v obrázku 2.8).



Obrázek 2.7: Některé z Haarových vlastností [?] a) horizontální hrana; b) horizontální hrana; c) horizontální čára d) diagonála e) vertikální čára.

0.1	0.2	0.2	0.4	0.2	0.4
0.1	0.8	0.7	0	0.1	1
0.3	1	0.6	0.2	0	0.4
0.5	0.7	0.9	0.1	0	0.5
0.7	1	1	0.1	0.3	0.6
0.2	0.3	0.7	0.4	0.5	1

$$\text{Suma pixelů v černém obdélníku} = \\ 0.8 + 0.7 + 1 + 0.6 + 0.7 + 0.9 + 1 + 1 \\ = 6.7$$

$$\text{Suma pixelů v červeném obdélníku} = \\ 0 + 0.1 + 0.2 + 0 + 0.1 + 0 + 0.1 + 0.3 \\ = 0.8$$

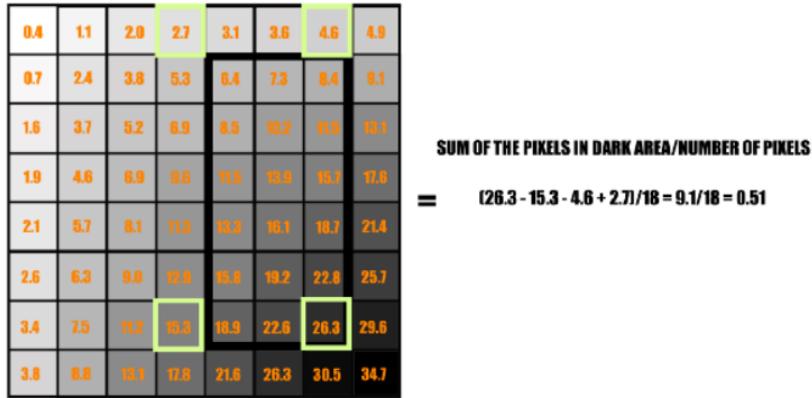
$$\text{Průměrná hodnota v černém obdélníku} = \\ 6.7 / 8 = 0.8375$$

$$\text{Průměrná hodnota v červeném obdélníku} = \\ 0.8 / 8 = 0.1$$

$$0.8375 - 0.1 = 0.7375 \\ \text{DETEKOVÁNA HRANA}$$

Obrázek 2.8: Příklad výpočtu detekce hrany dle vlastnosti b) z obrázku 2.7. Bílý obdélník je nahrazen červeným pro lepší ilustraci.

Algoritmus postupně prochází celý obrázek a hledá výskyt některé z vlastností. Toto procházení u obrázků s velkým počtem pixelů znamená enormní výpočetní nároky, protože je potřeba vždy počítat s hodnotami všech dotčených pixelů. Proto Viola a Jones [?] navrli vylepšení nazvané anglicky **Integral Image**. To spočívá v tom, že všechny pixely stačí projít pouze 1x, a každý tento pixel lze ohodnotit sumou hodnot pixelů směrem nalevo a nahoru (důsledkem je, že pixel s nejnižším ohodnocením se nachází v levém horním rohu a pixel s nejvyšším ohodnocením v pravém dolním rohu). Následný výpočet průměrné hodnoty ukazuje obrázek 2.9.



Obrázek 2.9: Ukázka výpočtu hodnoty obdélníku v Haarově vlastnosti za pomocí vylepšení **Integral Image**. Převzato z [?].

Jelikož Haarových vlastností je velké množství, bylo vybráno 6000 nejvíce vyhovujících, které se k detekci obličejů používají. Detekce je rozdělena na několik etap, v každé etapě je vyhledáván v části obrázku výskyt několika vlastností (počet s každou etapou roste), pokud se vyhledání nezdáří, není již dále daná část obrázku prohledávána.

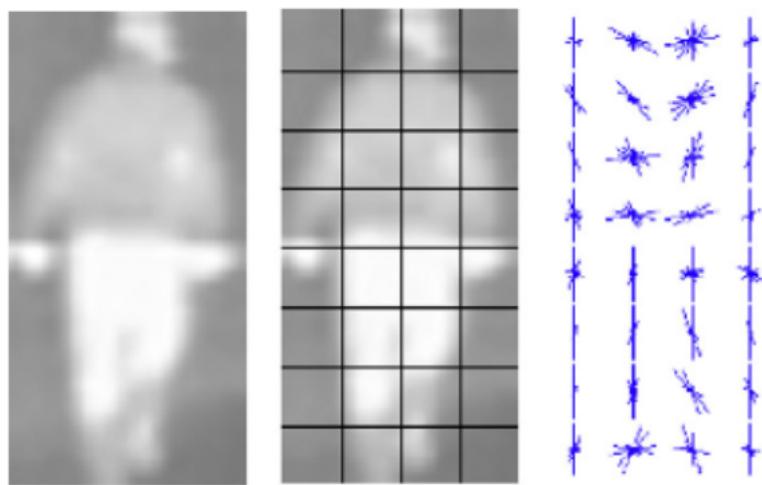
### Histogram orientací gradientů

Metoda HOG (anglicky Histograms of Oriented Gradients) [?, ?] používá k detekci obličejů či postav histogramy orientovaných přechodů v obrázku, rozděleném dle mřížky na několik bloků (často například  $8 \times 8$  nebo  $4 \times 4$  pixelů). Pro každý pixel v takovémto bloku jsou vypočítány hodnoty gradientů – velikost (magnitude) a směr (direction). Tyto hodnoty jsou pak přiřazeny do jednoho či více sloupců v histogramu popisujícím celý blok.

Tento histogram bývá rozdělen na 9 sloupců (rozmezí – anglicky *bin*) vyjadřujících směr gradientu v úhlu na škále od  $0^\circ$  do  $180^\circ$ , kdy každý sloupec odpovídá intervalu  $20^\circ$ . Výslednému vektoru, který udává velikosti gradientů v jednotlivých úhlech se říká HOG deskriptor.

HOG využívá pro zlepšení detekce při zhoršených světelných podmínkách normalizaci HOG deskriptorů. Tato normalizace je prováděna na vektorech 4 sousedících bloků. Pokud je tedy vektor složen z 9 hodnot, vypočítá se normalizační vektor z  $9 \times 4 = 36$  hodnot.

Výsledkem metody je obrázek (viz obrázek 2.10) reprezentovaný orientovanými gradienty, který se využívá k detekci osoby nebo obličeje.



Obrázek 2.10: Zleva: výřez originálního obrázku převedený do šedotónové reprezentace, obrázek rozdělený na bloky, výstup metody HOG. Převzato z [?].

## Kapitola 3

# Neuronové sítě pro detekci obličeje

Tato kapitola popisuje neuronové sítě a jejich využití pro detekci obličejů. Sekce 3.1 se věnuje popisu neuronových sítí obecně, v sekci 3.2 jsou popsány datasety a jejich využití k trénování a validaci neuronových sítí. Sekce 3.3 se zabývá konkrétními detektory obličejů s využitím neuronových sítí. Konkrétní programovací frameworky pro práci s neuronovými sítěmi vyobrazuje sekce 3.4.

### 3.1 Neuronové sítě

Neuronové sítě [?, ?] umožňují nalezení neznámého řešení problému pomocí naučení se z podobných problémů u nichž známe řešení. Tyto umělé sítě jsou inspirovány biologickou nervovou soustavou lidí (lidským mozkem). Síla neuronových sítí se projevuje v úlohách zaměřených na detekci, rozpoznávání (objektů, lidí, obličejů, obecně vzorů – anglicky *patterns*) a zpracování dat. Existuje řada druhů neuronových sítí (konvoluční, hluboké, dopředné, rekurentní), pro detekci obličejů v obrázcích se nejčastěji používají konvoluční neuronové sítě (anglicky *Convolutional Neural Networks* – CNN).

Na princip fungování neuronových sítí může být nahlíženo jako na nelineární matematickou funkci, která převádí  $X$  vstupů na  $Y$  výstupů. Proces transformace vstupních informací na výstup je ovlivňován váhováním hodnot vně sítě. Tyto hodnoty vah jsou určovány při tzv. učení/trénování neuronových sítí. Pro správné natrénování neuronové sítě je potřeba dostatečného počtu vstupních trénovacích dat (obrázků, textů, hodnot) a dostatečně výkonný hardware.

#### Biologický neuron

Jak již bylo zmíněno, inspirací neuronových sítí je biologická nervová soustava. V lidském mozku se nachází okolo  $10^{11}$  neuronů (elektricky aktivních buněk spracovávajících signály). Vstupem těchto neuronů jsou tzv. dendrity, výstupy pak nazýváme axony. Jednotlivé neurony jsou navzájem propojeny tisíci spojí pojmenovanými synapse. Synapse zajišťují komunikaci mezi neurony.

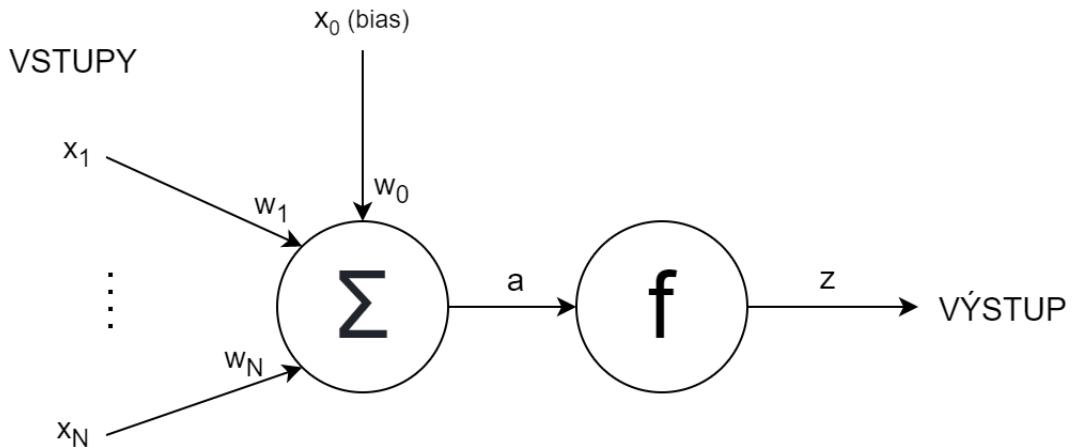
Takto vytvořený paralelismus poskytuje mozku schopnost rychle zpracovávat informace. Neurony biologické i umělé pracují s váhovanými vstupy. Po překročení určitého prahu na vstupech je adekvátně upraven výstup neuronu. Klíčovou vlastností potom je způsobilost měnit hodnoty jednotlivých vah na základě externích vlivů. Tím dochází k učení sítě neuronů.

## Perceptron

Nejjednoduším modelem umělého neuronu je perceptron. Perceptron má  $N$  vstupů, jejichž hodnoty  $x$  jsou vynásobeny váhmi  $w$  a sesumovány dle vzorce 3.1.

$$a = \sum_{i=1}^N (w_i * x_i) + w_0 * x_0 \quad (3.1)$$

Hodnota  $x_0$  se nazývá bias a jedná se o neměnnou vstupní hodnotu (často má hodnotu +1). Váhy a vstupy (včetně biasu) mohou být jak kladné, tak i záporné. Hodnota  $a$  je po vypočtení předána tzv. **aktivační funkci**.



Obrázek 3.1: Perceptron s  $N$  vstupy, biasem  $x_0$  a aktivační funkcí  $f$ .

## Aktivační funkce

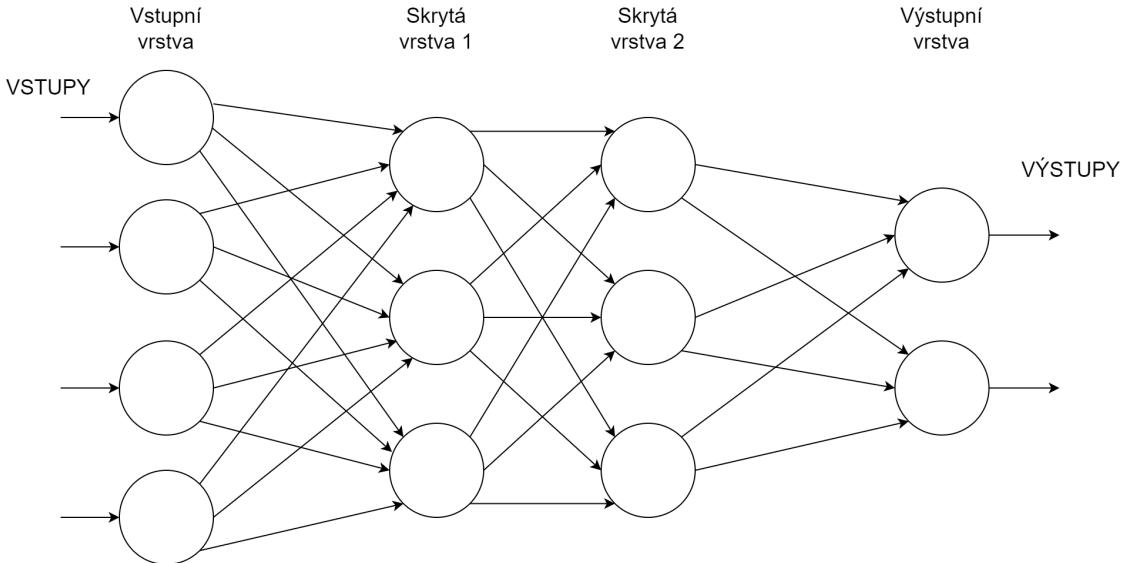
Aktivační funkce  $f$  je matematická funkce, která určuje výstup neuronu (vzorec 3.2). Na výběru vhodné aktivační funkce závisí přesnost neuronové sítě. Funkce může být různá, pro příklad je zde uveden výstup rozlišující zda se jedná o kladné či záporné číslo nebo nulu (viz vzorec 3.3). V praxi se často používá aktivační funkce nazvaná **rectified linear unit** (ReLU) [?] definovaná jako  $f(a) = \max\{0, a\}$ .

$$z = f(a) \quad (3.2)$$

$$f(a) = \begin{cases} 0 & \text{pro } a = 0 \\ 1 & \text{pro } a > 0 \\ -1 & \text{pro } a < 0 \end{cases} \quad (3.3)$$

## Spojování neuronů

Spojením několika neuronů lze vytvořit tzv. vrstvu (anglicky *layer*) perceptronů. Tyto vrstvy se dělí na vstupní (anglicky *input*), výstupní (anglicky *output*) a skryté (anglicky *hidden*). Konkatenací vstupní, několika skrytých a výstupní vrstvy je možno vytvořit neuronovou síť připravenou k trénování.



Obrázek 3.2: Příklad propojení vrstev neuronů do neuronové sítě.

## Učení

Aby neuronová síť mohla fungovat, musí se natrénoval (tzn. nastavit co nejpřesněji váhy na vstupech neuronů). K trénování se používají data z datasetů (viz sekce 3.2). Trénování neuronových sítí lze rozdělit do 3 kategorií [?]:

- **Učení bez učitele** (anglicky *unsupervised learning*) – tyto algoritmy procházejí data z datasetu a provádějí nad nimi shlukování do tzv. clusterů.
- **Učení s učitelem** (anglicky *supervised learning*) – každým datům z datasetu je přiřazena informace o požadovaném výstupu. Vstupy jsou zpracovány neuronovou sítí a podle chyby (rozdílu vstup/výstup) jsou upraveny parametry v neuronové síti.
- **Posilované učení** (anglicky *reinforcement learning*) – tento druh učení není vázán pouze na data z datasetu, ale navíc interaguje s prostředím.

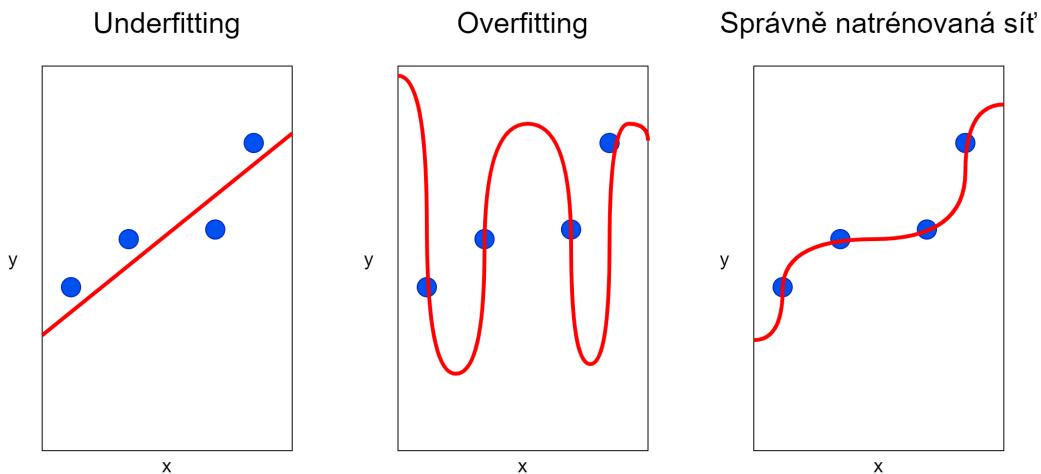
Trénování neuronových sítí je rozděleno do tzv. **epoch**, kdy jedna epocha znamená, že všechna trénovací data byla zpracována neuronovou sítí právě jedenkrát. S pojmem epocha souvisí pojem **batch**, který udává počet trénovacích dat, které sít zpracuje, než dojde k aktualizaci jejich vnitřních stavů. Běžně používané hodnoty batch jsou mocniny 2 (2, 4, 8, 16, 32, 64, ...).

## Chyby

Abychom byli schopní aktualizovat váhy vstupů jednotlivých neuronů, je potřeba měřit chybovost výstupu neuronové sítě. Jednou z možností měření chyby (využitelné například při lineární regresi) je *mean squared error*. Dle vztahu 3.4 je vypočítána chybovost a jsou upraveny váhy. Snahou je chybu co nejvíce zmenšit.

$$MSE = \frac{1}{N} \sum_i^N (vystupNS - spravnyVystup)^2 \quad (3.4)$$

Při trénování může dojít k situaci, kdy neuronová síť zvládá zpracovávat trénovací data s velmi vysokou přesností, ale při použití testovacích (validačních) dat se chybovost zvyšuje. V takovémto případě mluvíme o **přetrénování** (anglicky *overfitting*). Druhým nežádoucím jevem, který může nastat je **nedotrénovanost** (anglicky *underfitting*) – síť není dostatečně natrénovaná a dochází tak k vysoké chybovosti (příčinou je například málo trénovacích dat).



Obrázek 3.3: Ukázka výstupu neuronové sítě při nedoučení (*underfit*), přeучení (*overfitting*) a dostačeně dobrém množství dat pro učení. Obrázek vychází z [?].

## 3.2 Datasety

Datasetem rozumíme soubor podobných dat (například obrázků obličejů, číslic, předmětů nebo textů). Pro detekci obličejů se využívají datasety obsahující fotografie a videa lidí z veřejně dostupných zdrojů (internet, televize) nebo jsou datasety přímo účelně vytvářeny (fotografování lidí) a následně je možné si je kupit nebo stáhnout. Data v datasetech je nutné tzv. anotovat (označit co a kde na daném obrázku je). V případě tváří se může jednat například o věk osoby, pohlaví, rasu, aby bylo možné určit zda po klasifikaci neuronovou sítí výstup odpovídá požadovanému výsledku. Následující podsekce popisují některé datasety používané k učení neuronových sítí pro detekci nebo rozpoznávání obličejů.

### Datasety lidských obličejů

Dataset **WIDER FACE**<sup>1</sup> je dataset snímků obličejů obsahující 32203 obrázků ve skupinách trénovací, validační a testovací s poměrovým rozdělením 40%/10%/50%. Dataset je volně přístupný a autoři poskytují k trénovacím a validačním obrázkům anotační soubory s pozicemi obličejů. Fotografie spadají do různých skupin jako jsou různé pozice hlavy, různé zakrytí obličeje nebo špatné světelné podmínky. Fotografie jsou navíc rozděleny do 61 kategorií podle událostí, za kterých byly pořízeny (například demonstrace nebo volby).

<sup>1</sup><http://shuoyang1213.me/WIDERFACE/>



Obrázek 3.4: Příklady obličejů z datasetu WIDER FACE.

Dataset **DigiFace1M** [?] se skládá z 1 milionu snímků digitálně vytvořených obličejů (viz obrázek 3.5), čímž se vyhýbá případným právním a etnickým problémům, které mohou být spojeny s využití tváří reálných osob. Při použití tohoto datasetu umělých tváří společně s 200 až 2000 fotografiemi tváří reálných lidí pro učení, lze dosáhnout podobných výsledků detekce jako s modely trénovanými na datasetech s čistě reálnými obličeji.



Obrázek 3.5: Příklady vygenerovaných obličejů pod různými úhly a různým osvětlením z datasetu DigiFace1M [?].

**Multi-PIE** [?] je dataset zaměřený na fotografie obličejů pořízených za různých světelních podmínek a pod různými úhly. Obsahuje přes 750 000 snímků, které byly vytvořeny fotografováním 337 lidí po dobu několika měsíců. Tento dataset je placený.

Dataset **UTKFace** [?] obsahuje přes 20 000 fotek obličejů lidí různých věků, pohlaví a ras z různých úhlů a za rozličných světelních podmínek. Dataset je volně dostupný pro nekomerční použití. Data jsou podrobně anotovány dle vzorce `[věk]_[pohlaví]_[rasa]_[datum a čas].jpg`, kde věk je v rozmezí 0–116 let, pohlaví muž/žena, rasa je jedna z výčtu běloch, černoch, asiat, ind, ostatní a datum a čas uchovává informaci o okamžiku zařazení fotografie do datasetu.

Dataset **DARK FACE** [?] obsahuje 6000 fotografií různých prostředí s lidmi, pořízených za špatných světelních podmínek. Používá se mj. při hodnocení účinnosti detektorů obličejů. Dataset obsahuje jak anotované tak neanotované obrázky.

### 3.3 Detektory obličeje

Jak již bylo zmíněno, detekce obličeje je jak samostatnou disciplínou, tak i vstupním bodem několika dalších analyzačních úkonů (rozpoznávání tváře, ověřování pomocí obličejů nebo identifikace obličejů). Zpracovávaný obrázek bývá rozdělen na několik menších ob-

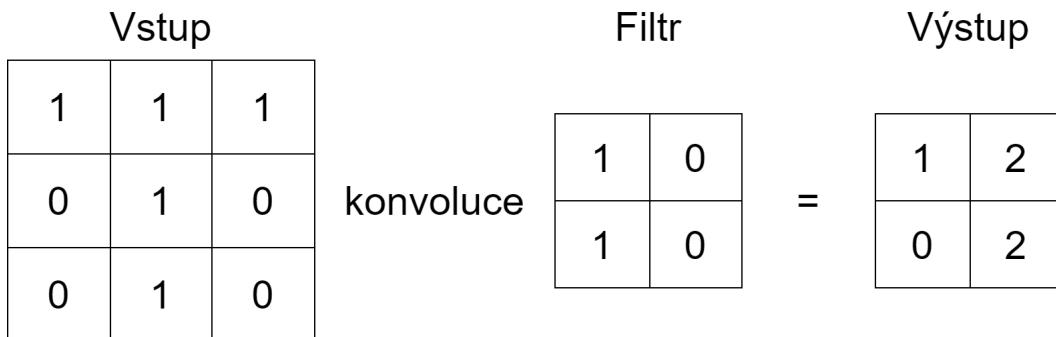
lastí (výřezů) a tyto výřezy bývají zpracovávány jednotlivě. Existuje řada různých druhů a architektur neuronových sítí využívaných k detekci obličejů [?]. Patří mezi ně například:

- **Rotation invariant neural network** (RINN, rotačně invariantní neuronová síť) – dokáže detektovat obličeje bez ohledu na úhel natočení tváře (není nutná normalizace v preprocessingu). Systém tohoto typu neuronové sítě obsahuje několik sítí, s tím že první se nazývá *router network*. Tato síť určuje orientaci (natočení) výřezu obrázku a připravuje (normalizuje) tak výřez pro 1 či více detekčních sítí.
- **Fast neural network** (FNN, rychlá neuronová síť) – obrázek je rozdelen na malé podobrázky a každý podobrázek je zvlášť otestován na výskyt tváře rychlou neuronovou sítí. Cílem je snížení výpočetního času nutného pro nalezení obličeje.
- **Polynomial neural network** (PNN, polynomiální neuronová síť) – oblasti v obrázku jsou klasifikovány jako tvář/ne-tvář pomocí binomické projekce této oblasti do prostoru vlastností obličeje naučeného tzv. základní analýzou komponent (anglicky principal component analysis, PCA). Zkoumáním vlivu PCA na vzorky lze detektovat zda se jedná o obličej.
- **Convolutional neural network** (CNN, konvoluční neuronové sítě) – samostatně popsány v další podsekci.

## Konvoluční neuronové sítě

Konvoluční neuronové sítě (CNN) [?, ?, ?] jsou použity ve většině systémů v oblasti počítačového vidění. Jelikož můžeme hodnoty pixelů vyjádřit číslami (používá se jak šedotónová varianta obrázku → při 8 bitové barevné hloubce může pixel nabývat hodnot 0–255, tak i běžná RGB varianta), lze tyto čísla použít jako vstupy neuronové sítě. Při vysokém rozlišení obrázku však nastává problém. Pokud bychom chtěli každou hodnotu pixelu použít na vstupu neuronové sítě pro celý obrázek naráz, měla by neuronová síť již u obrázku s rozmiřem  $500 \times 500$  pixelů celkem 25 000 vstupních neuronů, což by vedlo k extrémnímu množství neuronů v dalších vrstvách. Proto se využívá konvoluce.

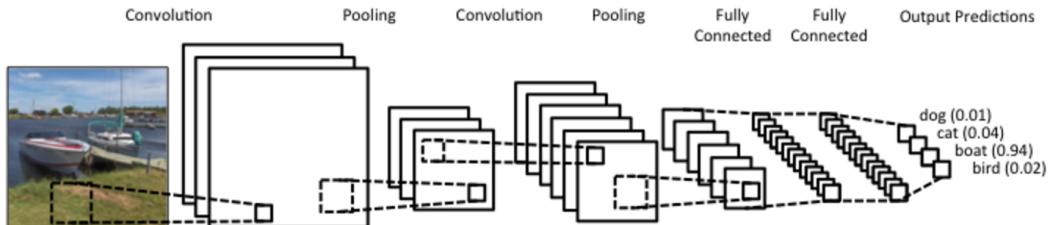
**Konvoluce** je okenní matematická funkce, při níž je na každou část obrázku (okno, *window*) násobením aplikována matici (filtr, kernel, konvoluční jádro). Princip konvoluce zachycuje následující obrázek.



Obrázek 3.6: Princip konvoluce 2D obrázku s rozmiřem  $3 \times 3$  a filtru  $2 \times 2$ .

Konvoluce dovoluje zásadně snížit počet nutných vstupů neuronové sítě. Navíc jsou konvoluční operace rychlé, protože tato funkce je hardwarově implementována na grafické kartě.

Jak lze vidět na obrázku 3.7 konvoluční neuronová síť se skládá z několika vrstev.



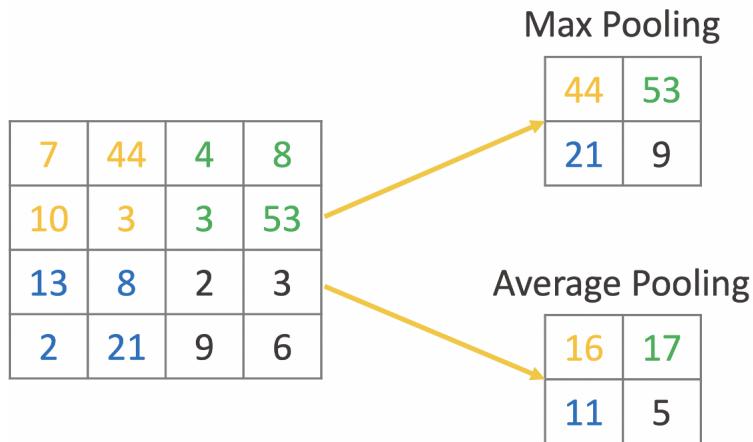
Obrázek 3.7: Řetězec vrstev konvoluční neuronové sítě. Převzato z [?].

**Konvoluční vrstva** provádí výše popsanou konvoluci. Otázkou zůstává jak zpracovávat okraje obrázku. Existují dva přístupy:

- *Wide convolution* – přidání nul kolem dokola obrázku a provedení konvoluce krajních hodnot (výstup bude stejně velký jako vstup).
- *Narrow convolution* – provedení konvoluce jako na obrázku 3.6. Dojde ke zmenšení velikosti na výstupu.

Dalším parametrem konvolučních neuronových sítí je tzv. **stride** velikost. Určuje o kolik hodnot se konvoluční jádro posune při každém kroku. Na obrázku 3.6 je použit stride 1. Větší hodnota stride vede k menšímu výstupu. Běžně se používají hodnoty 1 nebo 2.

**Poolingová vrstva** (anglicky pooling layer) je vrstva aplikovaná po konvoluční vrstvě. Používá se ke snížení výpočetní náročnosti. Tím, že je aplikována statistická funkce (maximum, průměr) na výstup konvoluční vrstvy (většinou filtrem o velikosti  $2 \times 2$  se stride = 2), dojde ke zredukování počtu hodnot výsledné matice (viz obrázek 3.8).



Obrázek 3.8: Příklad pooling vrstvy s max poolingem a průměrovacím poolingem. Převzato z [?].

V reálné aplikaci jsou konvoluční a poolingové vrstvy poskládány za sebe. Na konci řetězce (obrázek 3.7) se pak nachází část plně propojených vrstev neuronů (*fully-connected layers*).

## Detektory

Tato část popisuje vybrané existující detektory obličejů založené na neuronových sítích a porovnání úspěšnosti detekce těchto detektorů s detektory popsanými v sekci 2.3.

**MTCNN** (Multi-Task Cascaded Convolutional Network) [?, ?] provádí detekci obličejů pomocí kaskády konvoluční neuronových sítí. Algoritmus má dvě části. První část vytváří několik verzí vstupních obrázků, každa taková verze má jiné rozlišení. Druhá část složená z kaskády neuronových sítí se stará o detekci obličeje v různých verzích obrázku. Použití více verzí obrázku umožňuje zvýšit efektivitu detekce.

Single shot detector **SSD** [?] je konvoluční neuronová síť opět složená ze 2 částí, sloužící obecně k detekci objektů. Tyto části jsou: a) natrénovaný klasifikační model (VGG–16 nebo jiný) a b) část SSD – další část, která rozděluje obrázek pomocí mřížky (*grid*) s rozměry  $4 \times 4$  nebo  $8 \times 8$ . V každém takto vytvořeném bloku je pak hledán objekt (obličej).

Detekční model **RetinaFace** [?] používá multitasking pro učení. Detekce probíhá určením pozice obličeje podle vyhledání pixelů, které tvář obsahuje. Model má 2 struktury: *Multi-task loss* – minimalizace chyb modelu a *Dense Regression Branch* – renderer, který filtrouje obličeje a získává pixely z vyrenderovaných obličejů.

**YOLO** (You Only Look Once) [?] je stejně jako SSD tzv. single shot detektor – rozděluje vstupní obrázek do mřížky a provádí detekci případně rekognici nad jednotlivými bloky mřížky. Pro každý blok je určena hodnota jistoty výskytu objektu. Na vstupu sítě YOLO jsou barevné obrázky o velikosti  $448 \times 448$  pixelů. Architektura neuronové sítě je složena ze 7 konvolučních vrstev následovaných pooling vrstvami a 3 plně propojenými vrstvami na výstupu. YOLO má několik verzí, které se liší složením neuronové sítě a úspěšností detekce.

## Porovnání výkonnosti detektorů

Práce [?] se zabývá porovnáním detektorů Viola-Jones, Histogram of Oriented Gradients ze sekce 2.3 a algoritmů založených na konvolučních neuronových sítích MTCNN a SSD (klasifikátor VGG-16 je nahrazen klasifikátorem MobileNet) ze sekce 3.3. Testování bylo prováděno na datasetech AFW a WIDER FACE. Co se týká úspěšnosti detekce, nejlépe vyšel z testu detektor MTCNN, nejrychlejší pak byl detektor SSD (viz obrázek 3.9). Z detektorů založených na klasických metodách měl vyšší úspěšnost HOG, rychlejší však byl detektor Viola-Jones. Hodnota AP (*average precision*) udává průměrnou přesnost detekce a hodnota FPS (*frames per second*) počet snímků zpracovaných za sekundu.

Algorithm	AFW		WIDER FACE	
	AP	FPS	AP	FPS
Viola-Jones	40.80%	4.16	9.09%	7.52
HOG	69.88%	0.33	18.05%	0.71
MTCNN	<b>89.62%</b>	0.75	<b>44.66%</b>	1.22
MobileNet-SSD	89.28%	<b>9.46</b>	25.36%	<b>10.89</b>

Obrázek 3.9: Výsledek testování detekčních algoritmu [?].

## Detektory zaměřující se na špatné světelné podmínky

Detektory zaměřující se na detekci ve špatných světelných podmínkách jsou buď specializované detektory, kdy je neuronová síť detektoru učena na datasetu vhodných obrázků (obrázky pořízené za špatných světelných podmínek) nebo je použit detektor obličeje na trénovaný na obyčejných snímcích. V obou případech je nutno nejprve provést *low-light image enhancement* (vylepšení obrázku) a až poté přejít k detekci.

Pro vylepšení obrázku existují různé metody [?]:

- Mirnet,
  - Adaptive Gamma Correction (AGC),
  - Retinex,
  - RetinexNet,
- a další.

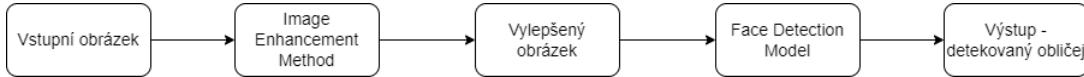
**Mirnet** je metoda postavená na konvolučních neuronových sítích a jejím cílem je zvýšení kvality obrazu pomocí několika úprav v konvolučních vrstvách. Je použita i v praktické části této práce, viz kapitola 5.

**AGC** je metodou využívající gamma korekci k vylepšení nasvícení scény (obrázku). Metoda navíc rozděluje obrázky podle intenzity gamma na tmavé (hodnota pod 0,5) a světlé (hodnota nad 0,5).

Koncept **Retinex** se snaží upravit kvalitu obrazu tak, aby odpovídala kvalitě lidského vidění. Dosaženo toho je tak, že je prováděno odmazávání odrazů světla z obrázku.

**RetinexNet** vychází z Retinexu a provádí dekompozici obrázku tak, aby došlo k oddělení odrazů nezávislých na osvětlení a osvětlení se známou strukturou. Metoda také celkově upravuje nasvícení pro dosažení co největší konzistence světla v obrázku.

Pro porovnání účinnosti *image enhancement* metod lze použít 2 veličiny. *Peak signal-noise ratio* (PSNR) – poměr užitečného signálu k šumu a *Structural similarity* (SSIM) – index podobnosti 2 obrazů.



Obrázek 3.10: Řetězec postupu detekce obličeje ve zhoršených světelných podmírkách. Vychází z [?].

Obrázek 3.10, který vychází ze článku [?] popisuje potřebný postup pro úpravu obrázku (*preprocessing*) a následné kroky vedoucí k detekci obličeje. V rámci tohoto článku jsou zpracovány výsledky detekce pomocí výše zmíněných metod vylepšení obrazu a detektoru RetinaFace. Jako testovací dataset byl použit dataset DARK FACE, popsaný v sekci 3.2. V rámci testování preprocessingových metod vylepšení obrazu bylo zjištěno, že nejlepší hodnoty PSNR (19,48 dB) a SSIM (0,74) dosahuje metoda Mirnet (hodnoty dalších metod viz obrázek 3.11).

Model	PSNR (dB)	SSIM
Raw	7.83	0.20185421
AGC	13.60	0.25519422
Retinex	13.40	0.47176218
RetinexNet	16.77	0.41909298
MirNet	19.48	0.7411703

Obrázek 3.11: Hodnoty PSNR a SSIM pro metody vylepšení obrazu [?].

Výsledky detekce však ukázaly, že v detekci obličejů si nejlépe vedl Retinex s 0,43 % mAP (*mean average precision* – průměrná přesnost detekce). Z toho vyplývá, že lepší preprocessingová metoda nutně neznamená úspěšnější detektor.

Detekcí ve špatných světelných podmírkách se zabývá také článek [?]. V rámci tohoto článku byl vytvořen samoučící se (self-supervised) detektor dosahující 44,4 % mAP také na datech z datasetu DARK FACE.

Práce [?] se rovněž zabývá detekcí nad datasetem DARK FACE při špatném osvětlení. Navržený algoritmus dosahuje 82,3 % *mAP*. Algoritmus využívá 3 hlavní části umožňující dosáhnutí takto vysoké přesnosti detekce. Jsou to: *Multi-scale retinex with color restoration* – metoda podobná Retinexu, ale zachovává konzistenci barev, architektura konvolučních vrstev *PyramidBox* a *Multi-scale test module* – úpravy velikosti vstupního obrázku pro lepší detekci.

Velké rozdíly v hodnotách mAP jsou způsobeny tím, že neuronové sítě z prací [?, ?] jsou trénovány přímo na datasetu obličejů ve zhoršených pozorovacích podmínkách (specifikované detektory), zatímco [?] používá model učený na běžných snímcích tváří.

### 3.4 Frameworky pro neuronové sítě

Pro programování neuronových sítí nebo obecně aplikací strojového učení se využívají k tomu určené frameworky. Každý framework je trochu jiný a používá se k jiným účelům. Všechny zde zmíněné frameworky podporují programovací jazyk Python. Mezi nejpoužívanější frameworky patří: TensorFlow, PyTorch, Keras, ONNX, Caffe [?].

**TensorFlow** je framework pro jazyky R, C++, Python a další. Má několik modulů s rozličnými funkcionalitami. Je velmi rozšířený a implementuje jej například Google překladač. Dokumentace je velmi podrobná a framework podporuje paralelismus napříč GPU.

**PyTorch** je vědecký framework použitelný v jazyce Python, vhodný na prototypování. Navíc podporuje GPU paralelismus.

Dalším frameworkem je **Keras**, v němž se programují konvoluční a rekurentní neuronové sítě. Je minimalistický a snadno se integruje spolu s TensorFlow.

Framework **ONNX** je open-source framework pro hluboké učení. Modely vytvořené v ONNX lze konvertovat do jiných frameworků (TensorFlow, Keras).

**Caffe** je framework podporovaný napříč různými programovacími jazyky (C, C++, Python, MATLAB) a je vhodný pro konvoluční neuronové sítě. Umožňuje nastavit parametry sítě a natrénovat síť, není nutné ji přímo vytvářet.

V rámci této práce jsou použity frameworky TensorFlow a Keras.

## Kapitola 4

# Systémy pro detekci obličejů a akcelerace detekce

Detekci obličejů s využitím počítačových programů lze provádět nad snímkem (fotografií, obrázkem), sadou snímků, videozáznamem nebo tzv. real-timově pomocí kamer a kamero-vých systémů. V této kapitole jsou popsány aktuálně využívané prostředky pro vytváření obrázků k detekci obličejů (kamery) a také dostupná řešení zabývající se detekcí, a to jak placené komerční, tak neplacené open-source systémy.

### 4.1 Kamery

Nezbytnou součástí oboru detekce obličejů jsou kamery a kamerové systémy. Existují kamery specializované k detekci či rozpoznávání obličejů a kamery obyčejné, které pouze zprostředkovávají obraz dále ke zpracování.

Specializované kamery se používají například k zabezpečení objektů nebo jako domovní videozvonky, kdy kamera (respektive její software) v zachyceném obrazu detekuje a rozpozná obličej, a následně může vykonat přiřazenou akci (spustit alarm, poslat notifikaci, umožnit osobě vstup,...) [?].

Další oblastí, kde se kamery s detekcí a rozpoznáváním obličejů uplatní je bezesporu dohled ve veřejných prostorách (anglicky *CCTV surveillance*). Detekce obličeje může sloužit k hledání podezřelých osob v záznamech z dohledových kamer. Tyto záznamy jsou shromážďovány na serveru, pomocí detekce jsou z obrazu vyřezány fragmenty s obličeji lidí a poté jsou tyto fragmenty porovnány rozpoznávacím algoritmem s obličeji hledaných osob. Při shodě dochází k informování administrátora systému, který podnikne další kroky [?].

Kamery tedy mají v doméně detekce obličejů nezastupitelnou roli, na jejich praktické využití v systémech a řešení pro detekci se zaměřují následující odstavce.

### 4.2 Dostupná řešení

Řešení umožňující detekci (a často i rekognici) obličejů lze rozdělit do dvou kategorií: komerční (placené, profesionální) a nekomerční (zdarma, open-source, amatérské). V následujících dvou podsekčích jsou popsány konkrétní systémy umožňující detekci obličejů.

## Komerční

Komerčně využívaná zařízení pro detekci, případně rekognici lze běžně zakoupit a používat ve firemním nebo domácím prostředí. Mezi zástupce těchto zařízení patří například produkty firem Netatmo, Google, Nest [?].



Obrázek 4.1: Videozvonek Nest Hello [?]. Příklad komerčního využití rozpoznávání obličejů v praxi.

## Nekomerční

Nekomerční řešení pro detekci obličejů zahrnují frameworky s otevřeným zdrojovým kódem (open-source). Tyto frameworky a řešení využívají neuronové sítě, které jsou trénovány pomocí datasetů a následně jsou využity k detekci obličeje [?]. Mezi open-source řešení patří například TinaFace<sup>1</sup> nebo MTCNN<sup>2</sup>, další volně použitelné frameworky popisuje sekce 3.4.

### 4.3 Akcelerace detekčních algoritmů

S rozmachem využívání strojového učení a neuronových sítí se objevila potřeba zrychlení těchto sítí. Proto byly vytvořeny specializované zařízení s cílem urychlit výpočty a snížit energetickou náročnost výpočtů ve srovnání s běžnými (univerzálními) výpočetními prostředky. Vznikly tak hardwarové akcelerátory neuronových sítí. Příkladem jsou Intel Neural Compute Stick 2 [?], Coral<sup>3</sup> nebo NVIDIA Jetson<sup>4</sup>.

Existují různé typy architektur akcelerátorů [?], mezi něž patří architektury popsané v tomto odstavci. **NPU** (*neural processing unit*) – k provádění matematických operací využívají speciální NPU obsahující PE (*processing engines*). NPU používá hardwarové verze MLP (*multi-layer perceptron*), pro dosažení zrychlení obecného výpočtu (nejen neuronových sítí). Pokud je část programu určená ke zrychlení spouštěna často a jsou dobře známy vstupy a výstupy, lze tuto část programu zrychlit pomocí NPU. Příkladem algoritmu pro zrychlení je rychlá Fourierova transformace (FFT).

Architektura **RENO** využívá memristory (ReRAM) – speciální druh paměti jejíž struktura umožňuje urychlit maticové a vektorové násobení.

<sup>1</sup><https://github.com/Media-Smart/vedadet/tree/main/configs/trainval/tinafac>

<sup>2</sup><https://github.com/ipazc/mtcnn>

<sup>3</sup><https://coral.ai>

<sup>4</sup><https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/>

Mezi další akcelerátory patří série akcelerátorů **DianNao** využívaná na akademické půdě. Tyto akcelerátory obsahují NFU (*neural functional unit*), vstupní, výstupní a synaptický buffer a kontrolní procesor.

Průmyslovým akcelerátorem od Google je **TPU** (*tensor processing unit*). Je použitelný i přes cloud.

Kromě ReRAM jsou v akcelerátor používány i paměti typu HMC (*hybrid memory cube*). Obě paměti (ReRAM a HMC) umožňují tzv. *processing-in-memory*, takže snižují čas výpočtu tím, že není nutno data přesouvat mezi procesorem a pamětí. Příklady akcelerátorů postavených na HMC jsou Neurocube [?] nebo Tetris [?].

Většina akcelerátorů je zaměřena na výpočty s již natrénovanou neuronovou sítí, jen pár jich podporuje učení neuronových sítí. V některých případech (*edge computing*) se ke zrychlení neuronové sítě používají cloulové služby – v datacentrech provádějí náročné výpočty grafické karty a výsledky spolu s nenáročnými výpočty jsou zpracovány například na IoT (*Internet of Things* – internet věcí) nebo mobilních zařízeních.

## Intel Neural Compute Stick 2

Intel Neural Compute Stick 2 (NCS2) [?] (obrázek 4.2) je akcelerátor neuronových sítí zaměřený na počítačové vidění. Obsahuje VPU (*vision processing unit*) Intel Movidius X. Připojuje se k počítači přes rozhraní USB 3.0 a umožňuje urychlení neuronové sítě bez použití cloudu a s nízkými energetickými nároky (použití například v kombinaci s Raspberry Pi). Podporuje mj. frameworky TensorFlow, Caffe, PyTorch, Keras popsané v sekci 3.4. Pro práci s NCS2 se používá knihovna/framework OpenVINO [?].



Obrázek 4.2: Intel Neural Compute Stick 2 [?]

Experimenty provedené v [?] porovnávají výkonnost akcelerátorů NVIDIA Jetson Nano, NVIDIA Jetson Xavier NX a Raspberry Pi 4B s NCS2. K porovnání posloužily frameworky YOLOv3 – popsaný v sekci 3.3 a YOLOv3-tiny. Experimenty byly založeny na detekci objektů ve dvou videích:

- *Video1* – rozlišení  $768 \times 436$  pixelů, celkem 1596 framů
- *Video2* – rozlišení  $1920 \times 1080$  pixelů, celkem 960 framů

Naměřené výsledky zobrazuje tabulka v obrázku 4.3.

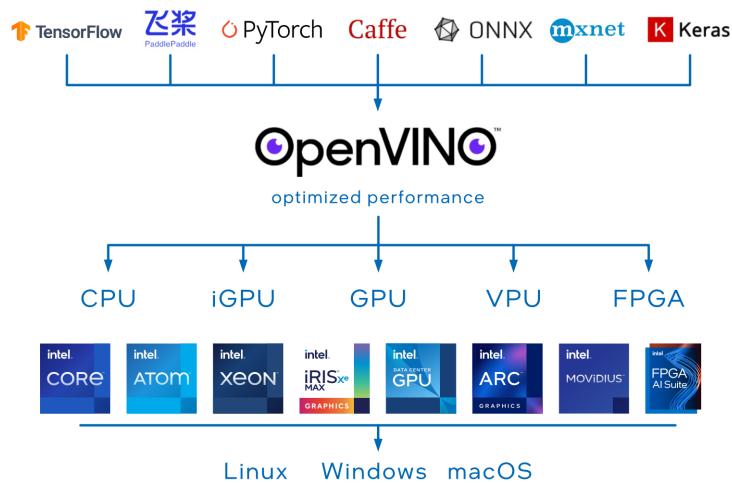
	Models	Accelerator -based SBCs	Mean Confidence(%)	FPS	CPU Usage %	Memory Usage (GB)	Power (W)	Time (s)
Video1	YOLOv3	RPI+NCS2	99.3	2.5	4.3	0.33	6.0	690
		Nano	99.7	1.7	26.5	1.21	7.9	967
		NX	99.7	6.1	22.5	1.51	15.2	256
	YOLOv3-tiny	RPI+NCS2	0	18.8	15.5	0.11	6.5	121
		Nano	57.9	6.8	28.8	1.00	7.2	236
		NX	57.9	41.1	30.5	1.33	13.5	46
Video2	YOLOv3	RPI+NCS2	85.8	2.5	9.8	0.41	6.2	496
		Nano	71.5	1.6	28.8	1.36	8.0	587
		NX	71.5	5.9	26.8	1.69	15.2	162
	YOLOv3-tiny	RPI+NCS2	61.5	19.0	24.8	0.18	6.8	162
		Nano	54.1	6.6	37.3	1.16	7.4	152
		NX	54.1	35.6	55.5	1.47	13.2	31

Obrázek 4.3: Výsledky testu akcelerátorů neuronových sítí s modelem YOLOv3 a YOLOv3-tiny [?].

Z tabulky v obrázku 4.3 můžeme vidět, že NCS2 dosahuje podobných výsledků detekce ve Video1 jako Jetson Nano a zároveň nejméně zatěžuje procesor. U Video2 má NCS2 nejlepší výsledky detekce. Využití paměti s větším vstupem roste, stále je však mezi konkurenčními nejnižší, to samé lze říci o spotřebě energie. S dostatečně dobrým modelem neuronové sítě je tak NCS2 úsporným akcelerátorem s vysokou úspěšností detekce.

## OpenVINO

Framework/knihovna OpenVINO se stará o konverzi a optimalizaci natrénovaného modelu neuronové sítě v jednom z podporovaných frameworků (viz obrázek 4.4) pro použití na NCS2 nebo jiném zařízení (procesory, grafické karty). OpenVINO se nezaměřuje pouze na oblast počítačového vidění, ale na neuronové sítě obecně. Knihovna a její použití je detailněji popsáno v dalších kapitolách.



Obrázek 4.4: Možnosti použití frameworku OpenVINO [?].

# Kapitola 5

## Návrh řešení

Tato kapitola popisuje jak nástroje a postupy použité při realizaci řešení, tak také způsob získávání dat pro natrénování detekční neuronové sítě. Dále se kapitola zabývá návrhem této konvoluční neuronové sítě a návrhem způsobu získávání výsledků detekčních experimentů. V kapitole také nalezneme popis funkcí uživatelského rozhraní detektora a přiblížení metody Mirnet.

### 5.1 Použité nástroje

K programování, akceleraci a vizualizaci výsledků detekce s využitím neuronových sítí je potřeba zvolit vhodný programovací jazyk, vhodné knihovny a frameworky. Mezi nejpoužívanější jazyky k tomuto účelu patří C++ a Python. Ze zadání byl vybrán jazyk **Python**. Vytváření neuronových sítí je usnadněno, pomocí k tomu určených knihoven jako jsou TensorFlow, PyTorch nebo ONNX. Akceleraci neuronových sítí na zařízeních značky Intel (tedy i Neural Compute Stick 2 – NCS2) se věnuje knihovna **OpenVINO**. Aby bylo možné ověřit funkčnost detektoru, je potřeba grafické rozhraní pro zobrazování detekce. K tomuto účelu lze využít knihoven pro grafická uživatelská rozhraní jako jsou Tkinter nebo **PyQt**. Tato část vystihuje soubor zvolených nástrojů k vytvoření detektoru obličejů a možnostem jeho použití.

#### TensorFlow

Framework **TensorFlow**, částečně popsáný v sekci 3.4, slouží k programování aplikací strojového učení. Využívá knihovnu **Keras** a umožňuje vytvářet nebo používat existující modely neuronových sítí. Tvorba neuronových sítí je realizována spojováním různých vrstev sítě a nastavováním aktivačních funkcí, filtrů apod. TensorFlow navíc poskytuje API pro různé aplikace umělé inteligence, jedno z těchto API se věnuje detekci objektů.<sup>1</sup>

V rámci frameworku TensorFlow je integrován program TensorBoard umožňující vizualizovat výsledky a průběh trénování a výsledky evaluace TensorFlow modelů.

#### OpenVINO

Teoreticky je tato knihovna popsána v předcházející kapitole. Praktické použití knihovny spočívá ve využití části nazvané **Model Optimizer** sloužící k optimalizování modelu neu-

---

<sup>1</sup>[https://github.com/tensorflow/models/blob/master/research/object\\_detection/g3doc/tf2\\_detection\\_zoo.md](https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md)

ronové sítě pro dosažení nižší latence při zpracování obrázku při detekci. Model může být dále akcelerován pomocí kvantování hodnot vah jednotlivých vrstev neuronů. Knihovna zároveň funguje jako programové rozhraní pro práci s Intel Neural Compute Stick 2 a dalšími akcelerátory značky Intel.

## PyQt6

Knihovna PyQt6<sup>2</sup> je Python knihovna vycházející z C++ implementace grafické knihovy Qt6. Používá se k návrhu grafického uživatelského rozhraní aplikací napsaných v jazyce Python, pracuje se signály a sloty (události a jejich obsluha). Knihovna poskytuje rozhraní a třídy pro práci s více vlákny, umožňuje používat dialogová okna a je snadná na používání.

## 5.2 Data

Pro trénování neuronových sítí je potřeba mít dostatek trénovacích dat. Detektory obličejů jsou trénovány na obrázcích z datasetů, které musejí mít přidruženy anotační soubory se souřadnicemi poloh obličejů. V této sekci je popsán návrh vhodného datasetu a možnosti jeho rozšíření pomocí augmentace.

### Datasetsy

Mezi existujícími datasety obličejů lze v jednoduchosti nalézt dva druhy dat. Jedním druhem jsou takové datasety, které obsahují fotografie obličejů v poměru jeden obličej na jednu fotografiu. Příkladem takového datasetu mohou být datasety **DigiFace1M** nebo **UTKFace** popsané v sekci 3.2. Tyto datasety mohou sloužit jako datasety k trénování detektorů obličejů, ale jejich použití je vhodné spíše na klasifikaci/rozpoznávání. Anotační soubory buď existují a obsahují souřadnice hranic obličeje na snímku, nebo úplně chybí a obrázek tak obsahuje pouze obličej – souřadnice odpovídají rozměrům obrázku.

Druhou možností jsou datasety, kde je na jedné fotografii větší množství obličejů (případně je zde jen jeden obličej, ale nezabírá celou plochu obrázku). Tyto datasety mívají k daným obrázkům přidruženy anotační soubory, obsahující minimálně koordináty tváří, v případě některých (například **WIDER FACE**, popsaný v sekci 3.2) jsou doplněny informace o rozmazání, póze, zastínění anebo zakrytí daného obličeje. Takovéto datasety jsou dobré pro trénování detekčních neuronových sítí.

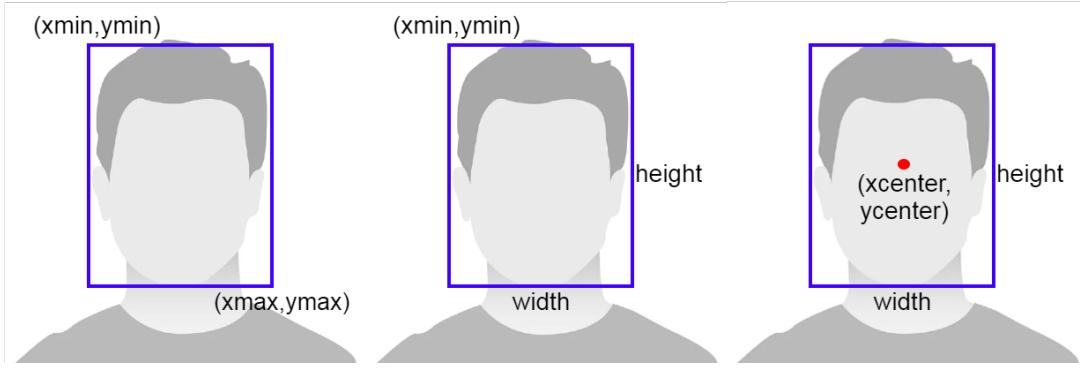
Body, udávající takzvané ohraňující boxy (anglicky *bounding boxes*) v nichž se vyskytují tváře je možné zapsat v různých tvarech. Základní dělení je na tři typy zápisu těchto souřadnic: **Pascal VOC**, **COCO** a **YOLO** [?].

**PascalVOC** anotuje obličeje ve tvaru  $x_{min}, y_{min}, x_{max}, y_{max}$ , kde  $x_{min}, y_{min}$  jsou souřadnice levého horního rohu a  $x_{max}, y_{max}$  jsou souřadnice pravého dolního rohu ohraňování.

Formát **COCO** využívá stejně jako PascalVOC hodnoty  $x_{min}, y_{min}$  a přidává k nim údaje *width* a *height*, odpovídající šířce a výšce hraničního obdélníku.

Třetí z nejpoužívanějších stylů značení, **YOLO**, nahrazuje hodnoty  $x_{min}, y_{min}$  hodnotami  $x_{center}, y_{center}$ , které značí střed ohraňování. Tento formát navíc stejně jako COCO poskytuje hodnoty *width* a *height*, aby bylo možné určit celou plochu ohraňování. Graficky tyto formáty zobrazuje obrázek 5.1.

<sup>2</sup><https://pypi.org/project/PyQt6/>



Obrázek 5.1: Anotace souřadnic obličejů pomocí různých anotačních formátů. Zleva: PascalVOC, COCO a YOLO.

## Augmentace

Dat z datasetů nemusí být vždy tolik, kolik je zapotřebí k natrénování dostatečně přesné neuronové sítě, nebo tyto data nemusejí přesně odpovídat zvolenému účelu a vytváření zcela nových dat by bylo neefektivní. V těchto případech lze tento problém řešit augmentací.

Augmentace (s daty, jimiž jsou obrázky) spočívá v kopírování existujících dat s provedením různých úprav (jas, vystrížení části obrázku, otočení, kontrast, ISO, gamma, rozmazání, záměna barevných kanálů apod.). Takto nově vzniklá data pak lze použít jako dodatečná data k trénovacím obrázkům. Augmentaci lze provádět ručně, ale existují i specializované knihovny zajišťující, že k nově vzniklému obrázku budou automaticky vytvořeny anotační soubory se správně pozměněnými pozicemi obličejů (pokud při augmentaci došlo k natočení, oříznutí nebo jiné podobné transformaci).

## 5.3 Detekční neuronová síť

Neuronová síť pro detekci obličejů je, jak už bylo zmíněno v kapitole 2, konvoluční neuronová síť, složená z několika různých vrstev (konvoluční vrstva, poolingová vrstva a další). Jelikož je detekční model často složen z velmi mnoha částí (např. generátor kotevních bodů/boxů – anglicky *anchor boxes*) a je náročné jej naprogramovat, poskytují frameworky pro práci s umělou inteligencí možnost využití API s předchystanými modely. Tyto modely pak je možno buď trénovat tzv. „od nuly“ nebo lze začít s trénováním částečně předtrénované sítě.

Při detekci ve zhoršených světelných podmínkách je vhodné využít některou z metod vylepšení obrázku (viz metody v sekci 3.3). Jednou z nejlepších takovýchto metod je metoda Mirnet, tvořená natrénovanou neuronovou sítí, převádějící tmavé a špatně osvětlené obrázky na obrázky s lepší viditelností.

Tato sekce popisuje jak API pro detekci objektů z knihovny TensorFlow, tak detailně také metodu Mirnet.

### TensorFlow 2 Object Detection API

Framework TensorFlow poskytuje open-source API pro práci s modely pro detekci objektů<sup>3</sup> nebo například sledování (tracking) objektů. Toto API je podporováno TensorFlow verzí

<sup>3</sup>[https://github.com/tensorflow/models/tree/master/research/object\\_detection](https://github.com/tensorflow/models/tree/master/research/object_detection)

1 i verzí 2 a umožňuje jednoduchým způsobem trénovat, validovat a testovat předpřipravené modely. Modely, které TensorFlow 2 Object Detection API (dále jen **TF OD API**) poskytuje jsou dostupné v tzv. TensorFlow 2 Detection Model Zoo. Tyto modely jsou předtrénovány na datasetu COCO 2017 a lze je tak využít na detekování jakýchkoliv objektů. Mezi nejznámější modely poskytované v rámci Detection Model Zoo patří **SSD ResNet50 V1 FPN** nebo **Faster R-CNN ResNet101**. Modely se liší mj. liší rozlišeními, s nimiž pracují, od  $512 \times 512$  pixelů až po  $1536 \times 1536$  pixelů.

TF OD API ke každému předtrénovanému modelu poskytuje základní informace o rychlosti detekce, o přesnosti změrené na validačních datech z výše zmíněného COCO datasetu a také informaci o tvaru výstupu neuronové sítě (souřadnice/klíčové body). Souřadnicový systém ohraničujících obdélníků odpovídá formátu PascalVOC.

S modely se pracuje pomocí skriptů v jazyce Python (trénování, evaluace, export), nastavování se provádí skrze editaci konfiguračních souborů, které jsou poskytovány pro každý model zvláště. Při trénování pak dochází k vytváření tzv. *checkpointů*, vytvoření souborů s údaji o vahách neuronů. Trénování neprobíhá v epochách jak je tomu běžné, ale v krocích, kdy jeden krok znamená, že síť zpracuje počet obrázků daných velikostí **batch**. Z toho vyplývá nasledující vztah:

$$krokuZaEpochu = \frac{početObrázku}{batch} \quad [kroky] \quad (5.1)$$

Celkový počet proběhlých epoch pak lze vypočítat jako:

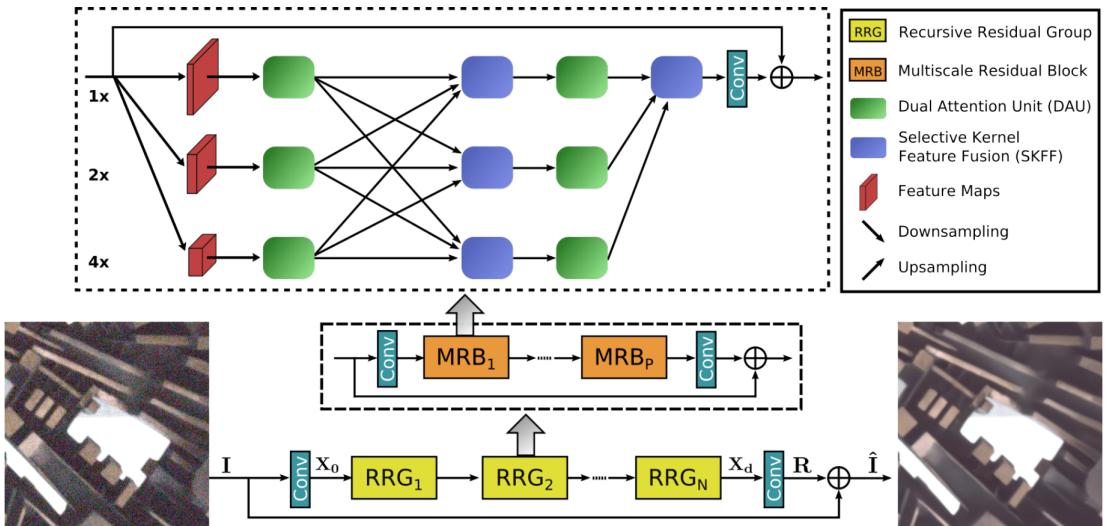
$$celkemEpoch = \frac{početKroku}{krokuZaEpochu} \quad (5.2)$$

## Mirnet

Metoda Mirnet[?] je sofistikovaná metoda pro zlepšování kvality špatně osvětlených obrázků pomocí neuronové sítě. Metoda je tvořena několika bloky (Feature Extractor, Dual Attention Unit, Selective Kernel Feature Fusion a další), které postupně zpracovávají vstupní obrázek a snaží se jej zkvalitnit (zesvětlit, zvýraznit detaily). Architekturu Mirnetu zachycuje obrázek 5.2. Jelikož je Mirnet tvořen neuronovou sítí, je třeba jej před použitím natrénovat. K tomuto účelu je zapotřebí dataset s párovými obrázky, kdy oba obrázky zachycují stejný fenomén, ale na jednom z nich je snížená/zhoršená viditelnost (například tmou). Ideálním kandidátem na trénování, který splňuje tyto požadavky je dataset LoL Dataset<sup>4</sup> (ukázka viz obrázek 5.3). Dostatečně natrénovaná síť pak dokáže vylepšit vizuální kvalitu jakéhokoliv obrázku, což je možno využít při detekci ve zhoršených světelných podmírkách.

---

<sup>4</sup><https://daoshee.github.io/BMVC2018website/>



Obrázek 5.2: Architektura neuronové sítě metody pro vylepšení kvality obrázků Mirnet [?].



Obrázek 5.3: Ukázka párového snímku z datasetu LoL (vlevo normální snímek, vpravo stejný snímek se zhoršenou viditelností).

## 5.4 Uživatelské rozhraní

Výkonnost a úspěšnost detektorů může být vyhodnocována jednak veličinami přesnosti a rychlosti, ale také lidským pozorováním. Proto je vhodné detektory objektů používat ve spojení s grafickým uživatelským rozhraním, kdy jsou do promítaného videa nebo obrázku explicitně zakresleny lokátory detekovaných objektů, v tomto případě konkrétně tváří. Uživatelské rozhraní by ovšem mělo nejen umět přehrávat snímky a vykreslovat ohraňující obdélníky, ale také nabízet možnost kontroly přehrávání a možnost výběru vstupních dat. Nutnou součástí každého rozhraní pracujícího s videem by navíc měla být informace týkající se přehrávání – aktuální čas a celková délka videa.

K návrhu grafického prostředí lze využít různorodé postupy, programy a knihovny. V rámci tvorby uživatelského rozhraní k této práci byla využita knihovna PyQt6 popsaná v sekci 5.1.

Aplikace navržená během tvorby této práce se však nestará jen o vyobrazení detekčních schopností navržené sítě, nýbrž také o přípravu dat pro vyhodnocení úspěšnosti a rychlosti detekce. Tato data (v případě úspěšnosti se jedná o procenta, v případě rychlosti pak o hodnotu FPS – *frames per second*, snímky za sekundu) finálně vyhodnocuje separátní program.

# Kapitola 6

# Implementace

Předchozí kapitola pojednávala o návrhu nástrojů a postupů pro vytvoření detektoru obličejů ve zhoršených světelných podmínkách. Tato kapitola navazuje líčením o implementaci těchto postupů. Na začátku implementace detekční sítě stojí příprava trénovacích dat (obrázků a anotačních souborů), popsaná v sekci 6.1. Následně je nutné vybrat a implementovat neuronovou síť detektoru a natrénovat ji (viz sekce 6.2). Pro zrychlení detektace je poté využito akceleračních nástrojů popsaných v předchozích kapitolách, v následující kapitole se tomuto věnuje sekce 6.4. K experimentům je nezbytné implementovat grafické uživatelské rozhraní (sekce 6.5) a příslušné vyhodnocovací nástroje a programy (sekce 6.6).

## 6.1 Příprava dat

Tématem této části je příprava fotografií z datasetů k trénování neuronové sítě. Proces předchystání dat spočívá ve zvolení vhodného datasetu pro detekci obličejů, anotováním obličejů ve fotografiích (pokud toto není dostupné přímo s datasetem) a provedením augmentace za účelem znásobení počtu dat a provedení případných úprav pro konkrétní zadání (v tomto případě mj. ztmavení). Konečným bodem přípravy je převod dat do TensorFlow formátu **TFRecords**. Takto nachystané prostředky pak lze použít k trénování TensorFlow modelů.

### Použité datasety

Pro detekci obličejů bylo zásadní zvolit vhodný dataset. V průběhu práce byly zvažovány a zkoušeny různé datasety a nakonec bylo rozhodnuto, že budou použity 2 datasety. Konkrétně dataset WIDER FACE a dataset DARK FACE, oba uvedené v sekci 3.2.

WIDER FACE poskytuje k trénovacím a validačním obrázkům anotační soubory v anotačním formátu COCO (viz sekce 5.2). Trénovacích snímků je dostupných celkem 12 880, z nichž byly odebrány ty, na nichž se nevyskytují žádné obličeje. Z takto filtrovaného výběru potom bylo vzato prvních 12 000 snímků a byla provedena jejich augmentace.



Obrázek 6.1: Ukázka snímku z trénovací části datasetu WIDER FACE.

DARK FACE dataset dává k dispozici 6000 anotovaných obrázků vyfocených v noci v městském prostředí (viz ukázka na obrázku 6.2). Z těchto 6000 fotek bylo prvních 4800 vybráno jako trénovací data (zbytek byl využit jako validační dataset). Následně byla provedena augmentace, podobně jako s datasetem WIDER FACE.



Obrázek 6.2: Ukázka snímku z datasetu DARK FACE.

## Augmentace fotografií WIDER FACE

Obecně byla augmentace rozebrána v sekci 5.2, konkrétně se v práci k provedení augmentace používá Python knihovna **albumentations**<sup>1</sup>. Proces augmentace sestává z několika kroků. Prvním z nich je zisk anotačních dat z anotačního souboru. První údaj obsahuje název fotografie, za ním následuje číslo udávající počet obličejů v obrázku a poté hraniční body všech obličejů v daném snímku.

Poté následuje načtení originální fotografie z datasetu WIDER FACE do augmentačního programu. Na načtené fotografii je provedena augmentace – je vytvořena kopie, v níž jsou s určitou pravděpodobností (viz číslo v závorce) provedeny tyto úpravy:

- horizontální převrácení (0.5),
- vertikální převrácení (0.5),
- náhodná změna jasu a kontrastu směrem ke ztmavení (0.7),
- náhodná změna gammy (0.1),
- přidání Gaussova šumu (0.2),
- přidání ISO šumu (0.4).

Pro vytvořený augmentovaný obrázek (ukázka na obrázku 6.3, originál viz obrázek 6.1) přepočítá knihovna albumentations nové souřadnice obličejů. Obrázek je pak uložen a anotační data skládající se z **názvu obrázku, souřadnic obličejů ve formátu PascalVOC a klasifikačních tříd** (všechny objekty patří do třídy *face*), jsou uložena ve formátu JSON. Podobný anotační JSON soubor se vytvoří i pro originální fotografii. Názvy obrázků jsou při tomto procesu nastaveny na číselné hodnoty 1.jpg až 12000.jpg pro originální snímky a na hodnoty 1.0.jpg až 12000.0.jpg pro augmentované obrázky. Celkem je tak k trénování připraveno 24 000 obrázků.



Obrázek 6.3: Ukázka augmentovaného snímku z datasetu WIDER FACE. Fotka je určena k trénování detektoru obličejů. Augmentace byla provedena s využitím knihovny albumentations.

<sup>1</sup><https://albumentations.ai>

## Augmentace fotografií DARK FACE

Augmentace obrázků z datasetu DARK FACE je velmi podobná augmentaci fotografií z datasetu WIDER FACE. Anotační soubor zda má mírně odlišnou syntaxi a výčet úprav je ochuzen o Gaussův šum a snížení pravděpodobnosti změny jasu a kontrastu na 0.5. Výstup augmentačního skriptu ovšem zachovává stejný formát. S využitím augmentace je tak vytvořeno 4800 nových snímků (1.0.png až 4800.0.png). Celkem tak trénovací set tvoří 9600 tmavých obrázků.

### TFRecord

Vstupy pro trénink TensorFlow modelů mohou být dvojice dat vstup  $\times$  výstup nebo se může jednat o serializovanou strukturu dat formátu **tfrecord**<sup>2</sup>. Jedná se o binární sekvenci dat, mající určitou organizaci a řád. Výhodou použití tfrecord namísto klasických obrázků a anotačních souborů je to, že odpadá část předzpracování dat před spuštěním trénování sítě. Skript starající se o transformaci originálních obrázků, augmentovaných obrázků a popisných souborů do formátu tfrecord pracuje tak, že načte obrázek a anotační soubor. Pak provede kontrolu, že žádný ohraničující box nezasahuje mimo rozměry obrázku. Dále vytvoří proměnnou typu **tf.train.Example**, která tvoří základ jednoho tfrecordu. Do proměnné se zapisují bytová data dle následujícího řádu:

- data obrázku,
- výška,
- šířka,
- název obrázku,
- formát obrázku,
- ID obrázku (odpovídá názvu),
- PascalVOC formát pozice obličeje,
- název a číslo třídy objektu (1 a „face“).

Z důvodu snazší manipulace s daty ve formátu tfrecord byly obrázky z datasetu WIDER FACE rozděleny do skupin po 2000 jednotkách. Celkově tak bylo vytvořeno 12 trénovacích tfrecord souborů (12 000 originálních a 12 000 augmentovaných obrázků po dvoutísícových skupinách). Pro druhý dataset byly vytvořeny skupiny po 200 snímcích a vzniklo tak 48 tfrecord souborů.

## 6.2 Detektor obličejů

Tato část pojednává o implementaci použitých detektorů obličejů. Zabývá se návrhem a tvorbou původního modelu pro detekci a následně i použitím konkrétních zvolených existujících modelů. V původním plánu byla snaha o vytvoření plně vlastního detektoru obličejů, od čehož bylo upuštěno pro příliš vysokou náročnost. Tomuto prvnímu záměru se věnuje první část této sekce. Přirozeně pak následujícím krokem byl přechod na modely existující.

---

<sup>2</sup>[https://www.tensorflow.org/tutorials/load\\_data/tfrecord](https://www.tensorflow.org/tutorials/load_data/tfrecord)

K tomuto účelu bylo vybráno **TensorFlow Object Detection API** (dále jako TF OD API) popsané v sekci 5.3. Výsledný detektor, respektive detektory tváří tak stojí na základech detektorů objektů. Explicitně využité modely neuronových sítí popisují další pasáže této sekce.

## Původní záměr

Jak již bylo avizováno výše, původní záměr spočíval ve vytvoření vlastní detekční sítě s využitím frameworků TensorFlow a Keras. Po několika pokusech se podařilo vytvořit a natrénuvat model podobný neuronové síti VGG-16. Tento model zvládal detekovat pouze 1 obličej, protože se spíše než o detektor jednalo o klasifikátor (který klasifikoval v obrázku obličej a dokázal vrátit souřadnice ohrazení tváře). Následným testováním na obrázcích s více obličeji, kdy bylo vyzkoušeno oblast detekovaného obličeje v obrázku začernit a pokusit se o detekci znova, bylo zjištěno, že metoda je neefektivní a nepříliš funkční.

Novým způsobem, který bylo možné zkuskit zrealizovat byla celková implementace SSD detektoru. Poměrně složité matematické operace s jednotlivými vrstvami výstupů detekční sítě se ukázaly být příliš náročné na implementaci. Bylo tak rozhodnuto, o využití existujícího řešení s implementací vlastní augmentace vstupních dat a vlastního natrénuování. Při trénování muselo být využito dostupného předtrénovaného stavu sítě, protože kompletní trénování sítě tzv. „od nuly“ by zabralo neúnosně mnoho výpočetního i reálného času.

Použité modely tak vycházejí z detektorů objektů a jsou přetrénovány tak, aby detekovaly jen 1 třídu objektů (obličeje), místo výchozích 90 tříd. V tom se také odlišují od konkrétně zaměřených detektorů tváří jako je například RetinaFace, která byla trénována přímo pro detekci obličejů.

## SSD ResNet50 V1 FPN

Prvním zvoleným modelem z TensorFlow 2 Detection Model Zoo<sup>3</sup> (TF DMZ) byl model **SSD ResNet50 V1 FPN 640x640 (RetinaNet50)**. Jak název napovídá, jedná se o neuronovou síť ResNet50 spojenou s kaskádou SSD. Model má na vstupu preprocessingovou část, která se stará o změnu rozlišení vstupního obrázku na  $640 \times 640$  pixelů. Dle stránek TF DMZ dosahuje ve výchozím předtrénovaném stavu model rychlosti detekce 46 ms/snímek a má průměrnou přesnost 34,3 % na datasetu COCO 2017.

SSD část tohoto modelu je popsána v sekci 3.3. Síť ResNet50 (Residual Network) [?] je konvoluční neuronová síť patřící mezi skupinu ResNet modelů, mající 50 vrstev. Vychází z původního 18, respektive 34vrstvého ResNetu. Každá vrstva obsahuje konvoluční filtr o velikosti  $3 \times 3$ . Konkurentem ResNetu je například síť VGG-16. Další verze ResNetu mají 101 nebo 152 konvolučních vrstev.

Tento model byl natrénuován na datasech popsaných výše. Samotné trénování popisuje sekce 6.3. Po natrénuování byl model vyzkoušen při detekci. Bez konkrétního měření přesnosti (konkrétní hodnoty byly změřeny později a jsou zaznamenány v experimentech v kapitole 7) dosahoval model obecných výsledků detekce obličejů, ovšem ukázalo se, že detekce je pomalá a nelze s ním detekovat tváře ve videu v reálném čase. Proto byl hledán jiný, rychlejší model. Nakonec byl vybrán model **SSD MobileNet V2 FPNLite**.

---

<sup>3</sup>[https://github.com/tensorflow/models/blob/master/research/object\\_detection/g3doc/tf2\\_detection\\_zoo.md](https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md)

## SSD MobileNet V2 FPNLite

SSD MobileNet V2 FPNLite  $640 \times 640$  je model složený z SSD části a neuronové sítě MobileNet. MobileNet [?] je pro svou efektivitu a rychlosť určen prevažne pre mobilné zariadenia (napríklad v Android telefonech je lze používať s TensorFlow Lite), ale môže poslúžiť i ako real-time detektor na bežnom počítači. MobileNet používa stejnú ako ostatné detekčné sítě konvolučné vrstvy, ale konvoluce v nej sú rozdelené na tzv. *depthwise* a *pointwise* časti. Depthwise konvoluce je aplikovaná na každý vstupný kanál a pointwise konvoluce pak na tento výstup aplikuje konvoluci s filtrom o velikosti  $1 \times 1$ . Veľkosť sítě v počte vrstiev je polovičná oproti ResNet50. Veľkosť na disku je v prípade MobileNetu  $10\times$  nižšia než v prípade ResNet50.

MobileNet podľa dát z TF DMZ dosahuje rýchlosť zpracovania 1 obrázku za 39 ms a průměrná úspešnosť na datasetu COCO 2017 je 28,2 %. Model stejnú ako predchádzajúci ResNet50 pracuje s rozlišením  $640 \times 640$  pixelov. V práci boli nakonec použité oba modely a během experimentů byly porovnány mezi sebou i se *state-of-the-art* řešeniami.

## 6.3 Trénování

Následující blok je věnován problematice trénování vybraných neuronových sítí z předcházející sekce. Neboť je trénování detekčních neuronových sítí velmi výpočetně a časově náročná akce, je potřeba mít výkonnou grafickou kartu nebo využít trénování v cloudu (například Google Colab). Autor textu nejprve zkoušel trénovat síť na vlastní grafické kartě NVIDIA GTX 1070 Ti s 8 GB video paměti RAM. To se ukázalo jako příliš pomalé a tak bylo trénování přesunuto na servery služby Metacentrum. Podrobněji Metacentrum popisuje text níže.

### Metacentrum

Metacentrum<sup>4</sup> je virtuální organizace spravovaná organizací Cesnet, která poskytuje zdarma akademickým pracovníkům a studentům výpočetní zdroje na několika serverech po celé ČR. Spouštění úloh je realizováno interaktivě nebo dávkově. Pro potřeby trénování bylo využito dávkových úloh, které spočívají v přípravě bash skriptu a jeho následném předání plánovači úloh. Skripty dávkových úloh mají danou počáteční syntaxi dle pravidel Metacentra (nastavení požadovaných zdrojů, požadovaná délka běhu atd.). Po těchto pravidlech následuje klasický kód v bashi.

Aby mohlo trénování běžet, bylo potřeba předpřipravit prostředí na serverech Metacentra. Příprava sestávala z vytvoření virtuálního prostředí (pomocí aplikace micromamba<sup>5</sup>) pro běh TF OD API, samotné instalace TF OD API a instalace potřebných knihoven pro práci s grafickými kartami NVIDIA (cuDNN 8.1.0 a CUDA 11.2). Pro trénování pak byl vybrán téměř vždy server Galdor s využitím dvou grafických karet NVIDIA A40 (48 GB video paměti).

Trénink modelu z TF OD API se spouští speciálním Python skriptem, který je obsažen v repozitáři TF OD API. Skript ke spuštění potřebuje znát místo uložení trénovaného modelu a také konfigurační soubor `pipeline.config`. V tomto konfiguračním souboru se nastavují parametry trénování (optimizér, počet kroků, batch velikost, míra učení – *learning rate*, maximální počet detekcí v obrázku, počet detekovaných tříd, druhy a váhy jednotlivých tříd).

<sup>4</sup><https://metavo.metacentrum.cz>

<sup>5</sup>[https://mamba.readthedocs.io/en/latest/user\\_guide/micromamba.html](https://mamba.readthedocs.io/en/latest/user_guide/micromamba.html)

vých chybových funkcí a další parametry) a také místo uložení trénovacích a evaluačních TFRecord souborů a lokace výchozího předtrénovaného bodu modelu.

Skript pro učení modelu automaticky každých 1000 provedených kroků ukládá aktuální stav modelu do tzv. checkpointů. Tyto checkpointy je pak možno převést na klasický TensorFlow *saved model* formát a následně použít k detekci. Výstupní hodnoty jednotlivých chybových funkcí jsou při běhu trénování vypisovány na standardní chybový výstup ve formátu, který zachycuje obrázek 6.4.

```
I0326 18:59:42.492782 22633028699968 model_lib_v2.py:705] Step 361100 per-step time 0.332s
INFO:tensorflow:{'Loss/classification_loss': 0.12352989,
'Loss/localization_loss': 0.09105951,
'Loss/regularization_loss': 0.04382626,
'Loss/total_loss': 0.25841564,
'learning_rate': 0.01434427}
```

Obrázek 6.4: Výstupní informace trénovacího skriptu během trénování modelu z TensorFlow Object Detection API.

## Natrénované modely

Celkově byly natrénovány 4 modely. Oba modely popsané v sekci 6.2 s oběma datasety ze sekce 6.1. Jednotlivé parametry trénovaných modelů zachycuje tabulka 6.1. Zkratky DF a WF v názvech modelů odpovídají názvu datasetu, který byl použit při tréninku (DF = DARK FACE, WF = WIDER FACE). Počet kroků je metrika, kterou používá k počítání průběhu trénování TF OD API. Počet epoch je pak vypočítán dle vzorce 5.2. Velikost batche byla nastavena na hodnotu 32 z toho důvodu, aby nedocházelo při trénování k problémům s nedostatkem video paměti. Počty trénovacích obrázků odpovídají velikostem augmentovaných datasetů, které byly popsány výše.

Model	Kroků	Epoch	Doba trénování	Batch	Počet tr. obrázků
DF Mobilenet	100 000	~333	5,5 h	32	9 600
DF Resnet50	500 000	~1667	4 dny	32	9 600
WF Mobilenet	362 000	~483	40 h	32	24 000
WF Resnet50	500 000	~667	3,5 dnů	32	24 000

Tabulka 6.1: Parametry modelů natrénovaných v rámci práce

Checkpointy, které trénováním vznikly pak byly pomocí TF OD API převedeny exportérem na tzv *saved model* formát. Tento formát je klasický TensorFlow formát modelů, jenž se dá v dalších aplikacích načíst a použít k detekci.

## Mirnet

Poněvadž je práce zaměřena na detekci ve zhoršených světelných podmínkách a využívá metodu Mirnet, která je sama neuronovou sítí, bylo třeba natrénovat i ji. K tomuto účelu postačila vlastní grafická karta autora a nebylo třeba využívat služeb Metacentra. Kód neuronové sítě Mirnetu byl převzat z [?]. Trénování sítě bylo spuštěno po dobu 70 epoch a byl použit dataset LoL. 350 obrázků z tohoto datasetu bylo využito k trénování a zbylých 135 pak k validaci modelu. Vzhledem k vysoké náročnosti na video paměť, byl batch nastaven na velikost 1.

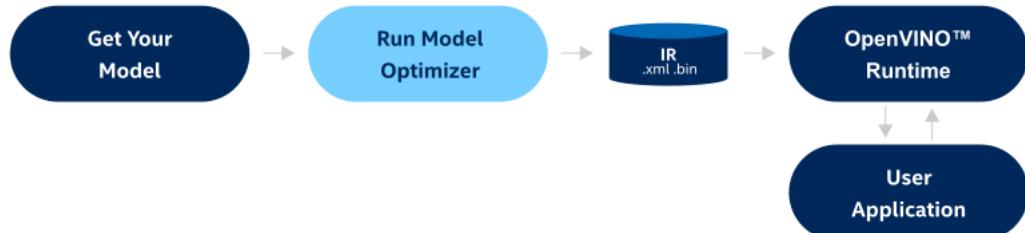
## 6.4 Akcelerace detekce

Teorie věnující se akceleraci neuronových sítí a akcelerační knihovně OpenVINO byla shrnutá v sekci 4.3. Tato část je zaměřena na konkrétní implementaci a použití knihovny OpenVINO během práce na detektoru obličejů. Samotné detekční modely vytvořené s pomocí frameworku TensorFlow, které byly popsány výše, nedosahují na málo výkonnému hardwaru požadované rychlosti zpracování vstupního videa. Řešením je použití akcelerační knihovny, která inferenci modelu výrazně urychlí, lze ji však využít jen na hardwaru značky Intel.

Akcelerace provedená v rámci této práce stojí na dvou pilířích z knihovny OpenVINO – Model Optimizer a kvantování. **Model Optimizer** je vstupním bodem v práci s knihovnou OpenVINO. Stará se o převedení uloženého TensorFlow modelu do formátů XML a BIN. Další možností ještě vyšší akcelerace je tzv. **kvantování**, které nemusí být dostupné a funkční na každém Intel zařízení. Kvantování však umožňuje razantní nárůst rychlosti se zanedbatelným vlivem na přesnost detekce. Obě tyto činnosti jsou shrnuty na následujících řádcích.

### Model Optimizer

Utilita **Model Optimizer** [?] se stará o převod a přizpůsobení modelu pro práci s možnostmi knihovny OpenVINO. Optimizer je součástí knihovního balíčku a spouští se jako samostatný program v příkazovém řádku. Typický pracovní postup Model Optimizeru zachycuje obrázek 6.5. Jak již bylo zmíněno, program vytváří 2 soubory – XML a BIN. XML soubor uchovává topologii neuronové sítě a v BIN souboru jsou uloženy váhy a biasy natřenované sítě.



Obrázek 6.5: Postup použití (workflow) modulu Model Optimizer z knihovny OpenVINO ke zrychlení modelu neuronové sítě. Převzato z [?].

Při převodu obyčejného TensorFlow nebo PyTorch modelu stačí programu předat cestu k uloženému modelu. Transformace modelu z TensorFlow Object Detection API (TF OD API) je nicméně o něco složitější. Model Optimizeru musí být předána cesta k uloženému modelu (parametr `saved_model_dir`), dále pak cesta ke konfiguračnímu souboru modelu (`tensorflow_object_detection_api_pipeline_config`) a také lokace tzv. transformačního konfiguračního souboru (`transformations_config`), který je specifický pro modely z TF OD API.

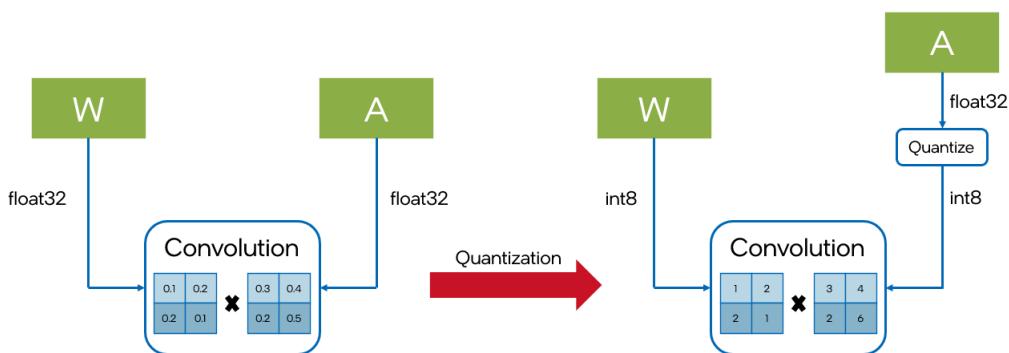
Aby byla co nejvíce navýšena rychlosť detekcie na Intel Neural Compute Stick 2 (NCS2), byl navíc přidán parametr `data_type` s hodnotou FP16. Ten zajistí, že model bude po převodu používat jen 16bitový float systém, což by mělo vést k celkovému zvýšení rychlosti inference. Posledním parametrem použitým při převodu je `input_shape`, nastavující vstup detektoru na  $640 \times 640$  pixelů.

Transformovaný model pak dosahuje značného zrychlení jak ukazují experimenty v kapitole 7. Akceleraci však lze ještě vylepšit pomocí kvantování.

## Kvantování

Kvantování [?] pomocí knihovny OpenVINO je proces, při němž dochází k transformaci modelu z *floating-point* aritmetiky na 8bitovou celočíselnou aritmetiku. Model se tak stává méně náročným pro hardware a může tak běžet rychleji. Zároveň nedochází k rapidnímu snížení výkonnosti detekce.

Základní princip při kvantování zobrazuje obrázek 6.6. Model neuronové sítě převedený OpenVINO Model Optimizerem, který pracuje s 32bitovými nebo 16bitovými floaty je kvantován, tak aby využíval 8 bitová velá čísla na interní výpočty modelu. Na výstupu jsou pak celá čísla převedena zpět na floaty.



Obrázek 6.6: Ukázka principu kvantování neuronových modelů [?]

Skript starající se o kvantování modelů používaných v rámci této práce, potřebuje na vstupu kromě XML a BIN souborů modelu také kalibrační dataset. Dle dokumentace by tento dataset měl být tvořen snímky z trénovacího nebo validačního balíčku a obsahovat alespoň 300 fotografií. Kvantování pak probíhá za spolupráce knihoven **nncf**<sup>6</sup> a OpenVINO, výstupní model je uložen opět ve formátu XML a BIN souborů. Vliv kvantování na rychlosť lze posoudit z grafů (TODO čísla grafů) v kapitole 7. Nevýhodou kvantování je mj. to, že takto pozměněný model nelze spustit na NCS2, které s celými čísly neumí pracovat. Kvantování již z logiky věci zmenšuje celkovou velikost modelu.

## 6.5 Grafické uživatelské rozhraní

Grafické uživatelské rozhraní (GUI) aplikace vytvořené k praktickému zkoušení modelů bylo naprogramováno tak, aby nejen zobrazovalo video a detekované obličeje, ale také tak, aby vytvářelo soubory s naměřenými metrikami pro další zpracování. Funkce a schopnosti GUI, způsoby měření, provádění detekce a organizaci výstupů jsou shrnutý v následujícím oddílu.

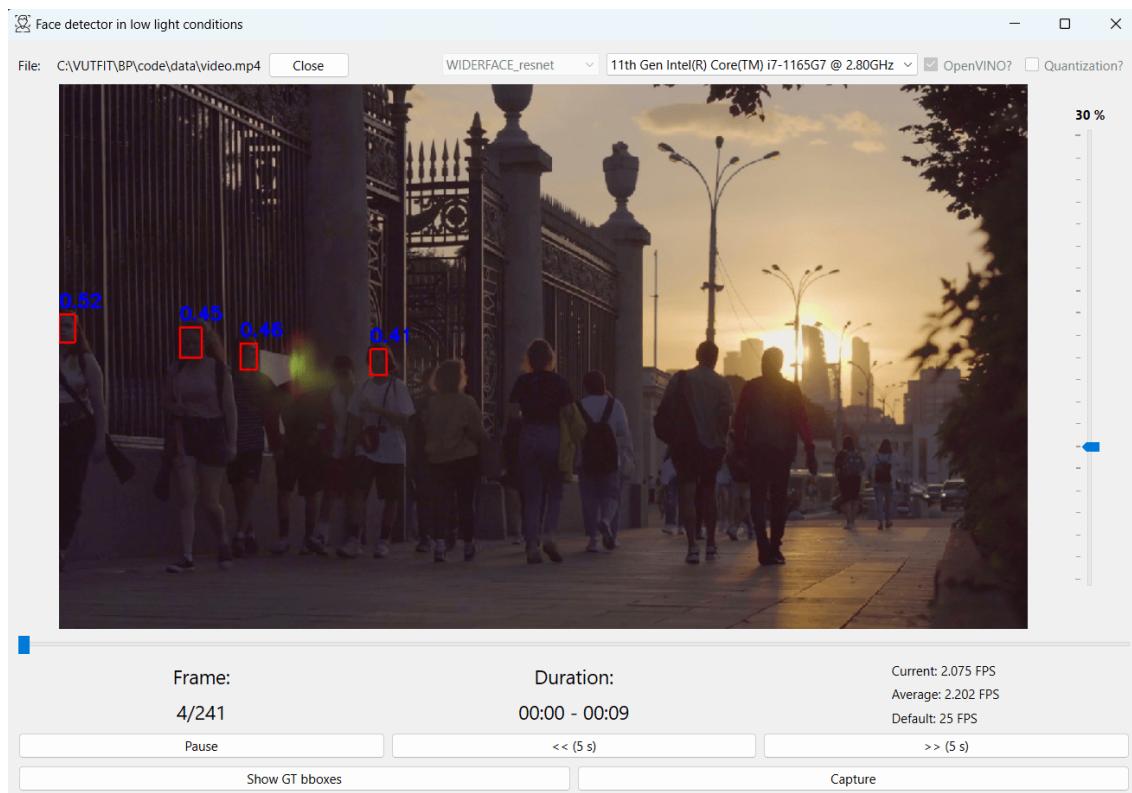
<sup>6</sup><https://github.com/openvinotoolkit/nncf>

## Rozhraní

Kostra aplikace je postavena na knihovně PyQt6 a jejích signálech a slotech (události a reakce na ně). Rozhraní (viz obrázek 6.7) se skládá z několika sekcí. Horní část slouží k výběru přehrávaného videa ve formátu mp4 a výběru modelu pro detekci (jeden ze čtyř natrénovaných z předcházející části + Viola-Jones + MTCNN). Dále lze pak vybrat zařízení na němž detekce poběží v případě spuštění akcelerované OpenVINO verze modelu (NCS2, CPU, GPU), a nakonec je možno určit zda se má použít akcelerovaný model (poličko OpenVINO?) a pokud ano, tak zda jeho kvantovaná verze (poličko Quantization?).

Následuje plocha s promítaným načteným videem, v němž jsou modře ohraničeny detekované obličeje. Na každém takovém boxem je – v případě detekce na novém modelu – červeným písmem napsána jistota (*confidence*) detekce. Vedle videa se nachází posuvník s možností nastavení prahu jistoty (*confidence threshold*) detekce – výchozí hodnota odpovídá hodnotě 0,3.

Ve spodní části je umístěn posuvník pro pohyb ve videu, dále pak ukazatele statistických informací (číslo aktuálního snímku/počet snímků celkem, uběhlý čas/celková délka videa a aktuální, průměrné a výchozí FPS). Na samém spodním kraji se nachází tlačítka ovládání přehrávání, tlačítko pro zobrazení správných (*ground truth*) boxů (pokud je anotační soubor videa k dispozici) a tlačítko pro uložení aktuálního snímku.



Obrázek 6.7: Rozhraní vytvořené aplikace pro detekci obličejů ve zhoršených světelných podmínkách.

## Měřené metriky

Při přehrávání videa nedochází pouze k detekci tváří, ale jsou zaznamenávány i metriky detekce. Mezi tyto metriky patří:

- aktuální FPS,
- počet detekcí v aktuálním snímku,
- počet a polohy detekovaných obličejů ve formátu COCO.

**Aktuální FPS** jsou vypočítány pomocí doby uběhlé mezi začátkem detekce a zakreslením hraničních boxů kolem detekovaných tváří do snímku. **Počet detekcí** aplikace zaznamenává pro každý snímek zvlášť. Tyto dvě metriky jsou exportovány v `csv` souborech k dalšímu zpracování (viz sekce [6.6](#)).

Ke každému snímku je do souboru `name_video_output.txt` zapsán **počet detekovaných obličejů a jejich souřadnice** v anotačním formátu COCO. Textový soubor odpovídá formátu, který používá dataset WIDER FACE. Formát COCO byl zvolen z toho důvodu, že stejný formát mají anotační soubory testovacích videí a je tak usnadněna práce s vyhodnocováním přesnosti detekce napříč snímkem videa. Zpracování těchto výstupů se věnuje oddíl [6.6](#).

## Detekce

Detekce, správa ovládání a zápis statistických informací probíhá v aplikaci v separátním vláknu. Vlákno je vytvořeno pomocí PyQt6 knihovních tříd `QThreadPool` a `QRunnable`. Pomocí PyQt6 signálů jsou mezi hlavním vlákнем aplikace a detekčním vlákнем předávány informace o snímkové frekvenci a aktuálním snímku s využitím slotových funkcí v rámci hlavního aplikačního vlákna. Další funkctionalita detekčního vlákna je zápis výstupních informací do `csv` souborů a předávání snímku s detekovanými obličeji pro vykreslení v hlavním vláknu.

Nastavení druhu používaného modelu, výběr zda využít jeho akcelerovanou nebo případně kvantovanou verzi je v gesci hlavního vlákna. Modely se nastavují jako globální proměnné, takže jsou dostupné ve všech funkcích a třídách aplikace.

O detekci se starají dvě hlavní funkce, obě jsou globální. Funkce `detect_faces` je funkce pro detekci OpenVINO a kvantovaných verzí modelu, podobně pojmenovaná funkce `detect_faces_tf` pak spouští detekci na TensorFlow variantě modelu. Která z funkcí bude využita definuje uživatelem dané nastavení v horním ovládacím rádku aplikace. Toto nastavení nelze již během zpracovávání videa měnit. Pro nastavení jiného modelu je třeba video uzavřít, změnit nastavení a video soubor znova načíst.

Detekované tváře, respektive jejich ohraničující body a důvěra modelu ve správnost detekce (*confidence*), jsou hlavním výstupem obou výše zmíněných funkcí. Na vykreslení obdélníků do snímku dbají funkce `apply_bboxes` a `apply_bboxes_tf`, jenž zároveň vrací souřadnice v COCO formátu pro uložení do výstupního souboru k dalšímu zpracování.

Pokud je k pouštěnému videu dostupný anotační soubor (pojmenovaný ve tvaru `name_video_gt.txt`), umí program ukazovat zelenou barvou *ground truth* boxy. Tato utilita se zapíná tlačítkem „Show GT boxes“ a může posloužit k vizuální kontrole detekce uživatelem. Podrobnějšímu popisu aplikace a GUI se věnuje soubor `gui_manual.pdf` v přiložených datových souborech.

## 6.6 Nástroje pro experimenty

Nástroje pro porovnávání modelů a experimenty s nimi jsou potřeba proto, aby bylo možné zhodnotit rozdíly mezi akcelerovaným a neakcelerovaným řešením a porovnat implementované detektory s existujícími. Tato část popisuje metody a postupy využité k evaluaci modelů a způsoby jejich porovnávání pro možnosti experimentů v další kapitole. Je zde vykázáno jak se která měřená metrika vyhodnocuje a jak vypadají výstupy experimentačních programů.

### Evaluace

Modely jsou v této práci porovnávány ve dvou kategoriích: **detekce ve videu** a **detekce v obrázcích z validačního datasetu**. K validaci je nutno stanovit některé pojmy a rovnice podle nichž dochází k vyhodnocování. Tyto měřící metody vycházejí z [?]. Základním prvkem je výpočet **IoU** (průnik nad sjednocením, anglicky *Intersection over Union*). IoU udává poměr průniku plochy detekovaného obličeje s plochou *ground truth* ohraničujícího boxu a sjednocení těchto ploch. Ukázka výpočtu viz obrázek 6.8.

$$\text{IoU} = \frac{\text{průnik ploch}}{\text{sjednocení ploch}} = \frac{\text{[blue square]}}{\text{[green square + red square]}}$$

Obrázek 6.8: Výpočet IoU. Obrázek vychází z [?].

Hodnota IoU pak slouží k určení, zda byl překonán práh  $t$  pro to, aby se dala detekce považovat za úspěšnou ( $\text{IoU} \geq t$ ). Detekce tak může dopadnout jedním z následujících 4 způsobů:

- True Positive (TP) – správná detekce *ground truth* boxu,
- False Positive (FP) – detekce objektu tam, kde ve skutečnosti není,
- False Negative (FN) – objekt nebyl detektorem detekován, přestože se zde nachází,
- True Negative (TN) – objekt správně nebyl detekován.

Z těchto dat pak vychází 2 metriky, které se používají k měření úspěšnosti detekce: **přesnost detekce** – poměr správných detekcí ke všem detekcím (dále jako *precision*) a **poměr správných detekcí ke všem *ground truth* boxům** (dále jako *recall*). Tyto dvě čísla jsou vypočítána pomocí následujících rovnic:

$$precision = \frac{TP}{TP + FP} \quad (6.1)$$

$$recall = \frac{TP}{TP + FN} \quad (6.2)$$

Hodnoty obou metrik se nejčastěji vyjadřují v procentech.

## Experimenty s videii

Pro detekci ve videu slouží vytvořená GUI aplikace zaznamenávající data do csv souborů. Mezi hodnocené metriky patří:

- průměrné FPS,
- průměrná přesnost detekce (*precision*) v napříč snímky (*frames*),
- průměrná hodnota IoU mezi snímky.

Pro měření přesnosti a hodnoty IoU slouží skript zpracovávající výstupní soubor se souřadnicemi detekovaných obličejů. Průměrná hodnota FPS je pak vyčtena z rozhraní aplikace.

Výpočet průměrné přesnosti je zpracován jako 11bodová interpolace [?]  $precision \times recall$  křivky (PR křivka). Tato křivka je tvorena z hodnot *recall* na ose x a hodnot *precision* na ose y. Hodnoty jsou zde vypočítány pomocí kumulativního součtu jednotlivých detekcí obličejů v obrázku, viz následující rovnice:

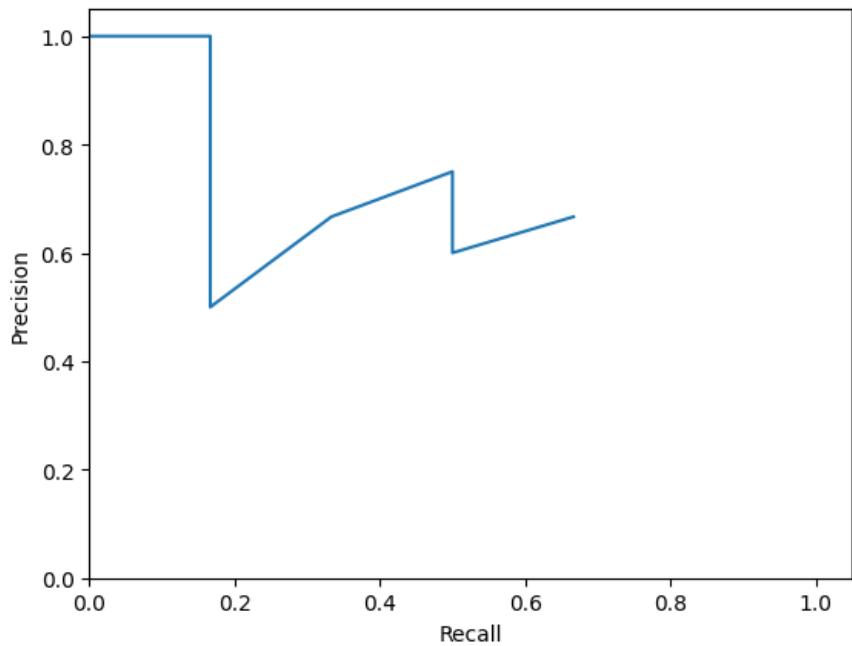
$$precision_k = \frac{\sum_{i=0}^k TP}{\sum_{i=0}^k TP + \sum_{i=0}^k FP} \quad (6.3)$$

$$recall_k = \frac{\sum_{i=0}^k TP}{\sum_{i=0}^k TP + \sum_{i=0}^k FN} \quad (6.4)$$

Tabulka 6.2 ukazuje příklad výpočtu hodnot *precision* a *recall* pro vytvoření křivky na obrázku s 6 obličeji (celkový počet *ground truth* boxů je tak roven 6). Na obrázku 6.9 jsou tyto hodnoty přeneseny do grafu a je vykreslena spojovací křivka. Do pole hodnot pro vykreslení se navíc přidává bod na souřadnicích [0, 1] aby byla křivka kompletní pro další výpočet.

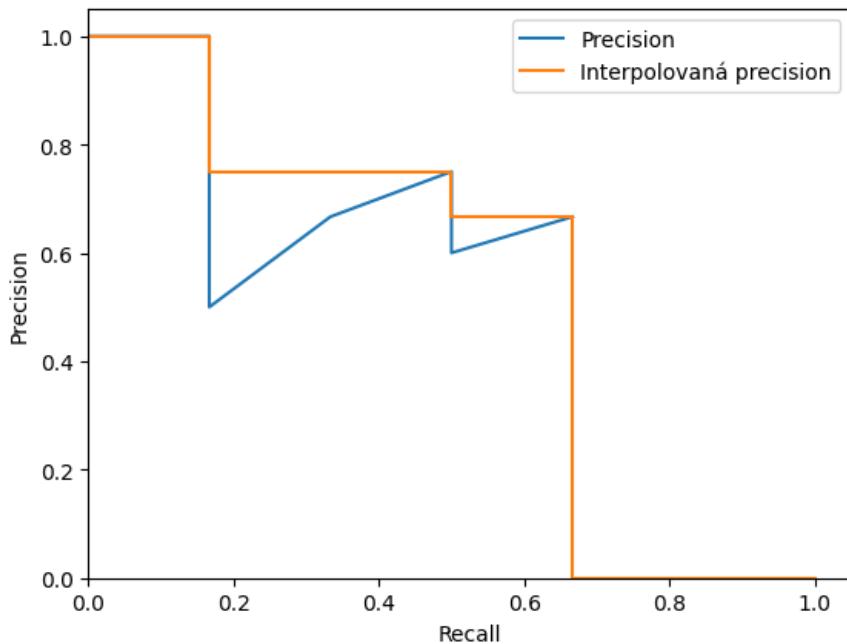
Výsledek detekce	K. součet TP	K. součet FP	Precision	Recall
TP	1	0	1/(1+0) = 1	1/6 = 0,166
FP	1	1	1/(1+1) = 0,5	1/6 = 0,166
TP	2	1	2/(2+1) = 0,66	2/6 = 0,33
TP	3	1	3/(3+1) = 0,75	3/6 = 0,5
FP	3	2	3/(3+2) = 0,6	3/6 = 0,5
TP	4	2	4/(4+2) = 0,66	4/6 = 0,66

Tabulka 6.2: Příklad výpočtu *precision* a *recall* pro 11bodovou interpolaci. V obrázku je celkem 6 obličejů, správně byly detekovány 4.



Obrázek 6.9: Graf  $precision \times recall$  křivky z příkladu z tabulky 6.2.

Výsledká průměrná přesnost je vypočítána jako plocha pod křivkou. Tuto plochu můžeme vypočítat pomocí zmíněné 11bodové interpolace. Tato metoda byla představena v roce 2007 v rámci detekční soutěže PASCAL VOC. Interpolace je prováděna na ose x ( $recall$ ) v bodech 0 až 1 s krokem 0,1 a jsou interpolovány hodnoty  $precision$ , které v daném bodě odpovídají **nejvyšší hodnotě napravo**. Pro nás příklad by 11 interpolace vypadala tak, jak ukazuje graf na obrázku 6.10.



Obrázek 6.10: Graf interpolované přesnosti z příkladu z tabulky 6.2 a z grafu 6.9.

Finální výpočet přesnosti pak vypadá následovně [?, ?] ( $P = \text{precision}$ ,  $R = \text{recall}$ ):

$$AP_{11} = \frac{1}{11} * \sum_{R \in \{0;0,1;...;0,9;1\}} P_{\text{interpolované}}(R) \quad (6.5)$$

Pro názorný příklad jsou interpolované hodnoty  $H$  rovny  $[1, 1, 0,75, 0,75, 0,75, 0,6666, 0,6666, 0, 0]$ . Výsledná přesnost tak je  $AP = \frac{1}{11} * \sum H$  což se rovná cca 50,756 %.

## Experimenty s datasetem

K měření úspěšnosti detekce v datasetu je využit validační dataset popsaný dále v sekci 7.1. Skripty měřící *precision* a *recall* umožňují nastavit model a zařízení na nichž se bude měřit. Evaluace je pak prováděna pomocí COCO evaluace<sup>7</sup>.

COCO evaluace detekčních modelů počítá přesnost detekce s různými prahy  $t$  pro hodnotu IoU. V experimentech v kapitole 7 jsou naměřené hodnoty *precision* pro tyto prahy:

1. průměr mezi 0,5 až 0,95 s krokem 0,05
2. 0,5
3. 0,75

Hodnota *recall* se pak počítá pro:

1. 1 detekci na 1 obrázek,
2. 10 detekcí na 1 obrázek,
3. 100 detekcí na 1 obrázek.

<sup>7</sup><https://cocodataset.org/#detection-eval>

# Kapitola 7

# Experimenty

Kapitola experimenty je zaměřena na otestování nově vzniklých detektorů obličejů ve zhoršených světelných podmírkách a jejich praktickému porovnání s existujícími detektory. Provedené testy využívají skripty a aplikace navržené a implementované v předchozích kapitolách. Výsledky testů umožňují identifikovat problémy jednotlivých detektorů a mohou určit, který z detektorů je vhodnější k řešení daného konkrétního problému.

## 7.1 Podklady

Jak již bylo zmíněno, testování je prováděno na validačním datasetu a na dvou testovacích oanotovaných videích. Tato část je věnována popisu jejich přípravy.

### Příprava validačního datasetu

Mezi data, která jsou hodnocena za účelem srovnání detekčních sítí, patří nejen data získaná z GUI aplikace, ale také výsledky evaluace modelů na validačním datasetu. Jako validační dataset byla vybrána část WIDER FACE datasetu určená k validaci s některými úpravami:

- odebrání snímků s více než 100 obličeji,
- augmentace snímků, tak aby spadaly do kategorie zhoršených světelných podmínek.

Vyběr obrázků s  $\leq 100$  obličeji byl zvolen z toho důvodu, že evaluační příkazy validátoru COCO ve výchozím nastavení detekují pouze maximálně 100 obličejů v jednom snímku. Vznikl tak validační dataset s 3167 fotografiemi. Typický snímek z datasetu viz obrázek 7.1.



Obrázek 7.1: Snímek z augmentovaného validačního datasetu WIDER FACE.

COCO validátor bere na vstupu anotační data ve speciálním formátu (**Data Format**) v podobě JSON souboru. Byl vytvořen skript převádějící *ground truth* souřadnice obličejů na tento formát. Dále pak validátor potřebuje ke svému fungování výstupní data detektoru (souřadnice obličejů a číslo obrázku ke kterému se informace vztahují) opět ve specifickém formátu **Result Format**. Skripty pro spuštění detekce na každém z vytvořených detektorů i na každém použitém existujícím řešení, vytvářejí přímo tento výsledkový soubor. Ukázky obsahují data v obou zmíněných formátech jsou obsaženy na přiloženém datovém médiu.

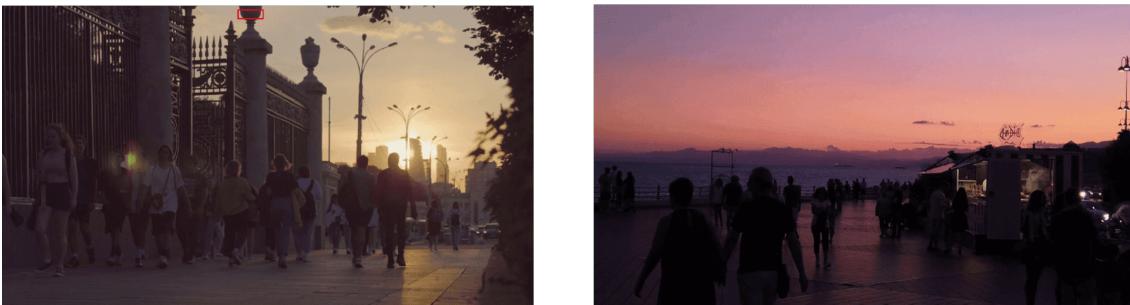
Naprogramovaný skript `eval.py` zpracuje výstupy (JSON result soubor) všech detektorů a zapíše do souboru statistiky vypočítané COCO evaluátorem. Tyto statistiky pak posloužily jako data do tabulek v následujících testech. Pokud není v textu řečeno jinak, byly detektory spuštěny vždy s prahem jistoty detekce (*confidence*)  $c = 0.25$ .

## Příprava videí

K otestování schopností detektorů byly vybrány 2 testovací videa. Obě videa bylo třeba oanotovat, k čemuž byl použit webový nástroj CVAT<sup>1</sup> (Computer Vision Annotator Tool). Tento program umožňuje ve zpracovávaném videu projít každý jeden snímek (*frame*) a nastavit v něm ohraňující boxy jednotlivých tváří. Tento úkon bylo nutno udělat ručně a výstup slouží jak *ground truth* údaje pro validaci a porovnávání modelů mezi sebou. Formát výstupu je možné zvolit v několika tvarech, pro zpracovaná videa byl zvolen výstup podobný anotaci WIDER FACE datasetu (číslo snímku, počet obličejů ve snímku, COCO souřadnice). Anotační soubory videí lze použít jak k hodnocení přesnosti detekce, tak při přehrávání pro zobrazování *ground truth* boxů.

První video (na obrázku 7.2 vlevo) je 9 sekund dlouhé, skládá se z 241 snímků o snímkové frekvenci 25 FPS. Zachycuje městskou ulici v podvečer a obsahuje cca 10 obličejů. Druhé video (na obrázku 7.2 vpravo) trvá 23 sekund, je natočeno ve 29 FPS a obsahuje 700 framů. Jeho meritem je průchod většího počtu lidí po pobřežním molu večer.

<sup>1</sup><https://www.cvcat.ai>



Obrázek 7.2: Snímky z obou testovacích videí.

Původním záměrem bylo real-timové využití Mirnetu pro detektory, které nebyly trénovány na datasetu DARK FACE. Tento úmysl se ovšem ukázal jako nerealizovatelný kvůli vysoké výpočetní a paměťové náročnosti převodu framu videa sítí Mirnet. Nebylo by tak možné dosáhnout únosné snímkové frekvence. Proto byly obě videa převedeny Mirnetem (výstupy viz obrázek 7.3) separátně od GUI aplikace. Lze je samozřejmě spustit v aplikaci stejným způsobem jako originály, čímž je simulována detekce s použitím Mirnetu.



Obrázek 7.3: Snímky z obou videí po úpravě Mirnetem

## 7.2 Porovnání akcelerovaného a normálního řešení

V prvním experimentu byly poměřeny natrénované TensorFlow modely s jejich akcelerovanými verzemi. První část testování proběhla na validačním datasetu, druhá na obou testovacích videích ve verzi bez Mirnet vylepšení.

### Testování na datasetu

Tabulky 7.1, 7.2, 7.3 a 7.4 ukazují přesnost (AP) a poměr správných detekcí (*recall*) pro jednotlivé modely a jejich akcelerované varianty. Měření optimalizovaných modelů pomocí knihovny OpenVINO probíhalo na 3 různých zařízeních: **CPU** – Intel Core i7-1165G7, **integrované GPU** – Intel Iris Xe a **VPU** – Intel Neural Compute Stick 2. Výkony výchozích TensorFlow detektorů byly v tomto testu měřeny na grafické kartě (**GPU**) NVIDIA MX 450 2 GB. Hodnoty v tabulkách jsou udány v procentech.

<b>Model</b>	<b>Zař.</b>	<b>AP</b>	<b>AP@0.5</b>	<b>AP@0.75</b>	<b>Recall1</b>	<b>Recall10</b>	<b>Recall100</b>
TensorFlow	GPU	3,0	9,3	1,0	1,9	5,5	6,1
OpenVINO	iGPU	3,1	9,2	1,0	1,8	5,5	5,9
OpenVINO	CPU	3,0	9,2	1,0	1,8	5,5	5,9
OpenVINO	VPU	2,9	8,6	1,1	1,7	5,0	5,4
Kvantovaný	iGPU	2,9	8,7	0,9	1,8	5,4	5,8
Kvantovaný	CPU	2,9	8,7	1,0	1,8	5,3	5,8

Tabulka 7.1: Model DARK FACE Mobilenet – výsledky detekce na validačním datasetu

Z tabulky 7.1 je patrné, že model SSD MobileNet natrénovaný na datasetu DARK FACE naprosto nezvládá detektovat obličeje z jiného datasetu. Maximální naměřená přesnost při  $IoU = 0,50$  nedosahuje ani 10 %. Rozdíly mezi akcelerovanou a TensorFlow verzí jsou minimální.

<b>Model</b>	<b>Zař.</b>	<b>AP</b>	<b>AP@0.5</b>	<b>AP@0.75</b>	<b>Recall1</b>	<b>Recall10</b>	<b>Recall100</b>
TensorFlow	GPU	1,6	4,7	0,5	1,0	2,6	2,8
OpenVINO	iGPU	1,8	5,0	0,5	1,0	2,6	2,8
OpenVINO	CPU	1,7	5,0	0,5	1,0	2,6	2,7
OpenVINO	VPU	1,7	5,0	0,5	1,0	2,6	2,7
Kvantovaný	iGPU	1,6	4,4	0,5	1,0	2,5	2,6
Kvantovaný	CPU	1,6	4,4	0,5	1,0	2,5	2,6

Tabulka 7.2: Model DARK FACE ResNet50 – výsledky detekce na validačním datasetu

Při záměně MobileNetu za ResNet50 nedošlo k žádnému zlepšení, jak vyplývá z tabulky 7.2. Oba modely trénované na datech datasetu DARK FACE jsou nevhodné pro detekci ve validačním datasetu.

<b>Model</b>	<b>Zař.</b>	<b>AP</b>	<b>AP@0.5</b>	<b>AP@0.75</b>	<b>Recall1</b>	<b>Recall10</b>	<b>Recall100</b>
TensorFlow	GPU	16,2	35,3	13,0	7,9	25,7	34,7
OpenVINO	iGPU	16,2	34,8	13,3	7,9	25,4	32,3
OpenVINO	CPU	16,2	34,8	13,3	7,9	25,4	32,3
OpenVINO	VPU	17,4	36,3	15,1	7,8	25,2	31,6
Kvantovaný	iGPU	17,8	37,5	15,1	7,8	25,3	31,5
Kvantovaný	CPU	17,7	37,4	15,1	7,8	25,2	31,6

Tabulka 7.3: Model WIDER FACE MobileNet – výsledky detekce na validačním datasetu

Změna trénovacích dat přinesla kýžený efekt v podobě výrazného zlepšení detekčních výsledků. Z tabulky 7.3 je patrné 350 % zlepšení přesnosti detekce v kategorii AP@0.5. Trochu překvapivě dosáhl nejvyšších výsledků přesnosti kvantovaný model, konkrétně 37,5 %. Tyto hodnoty nemusejí vypadat dobře, ale v porovnání s existujícími modely (viz sekce 7.3, 7.4 a 7.5) nejsou výsledky vůbec špatné. Model obsahující MobileNet je navíc rychlý, což ukazují měření v další části této sekce.

<b>Model</b>	<b>Zař.</b>	<b>AP</b>	<b>AP@0.5</b>	<b>AP@0.75</b>	<b>Recall1</b>	<b>Recall10</b>	<b>Recall100</b>
TensorFlow	GPU	23,2	47,1	20,4	8,3	28,2	38,1
OpenVINO	iGPU	22,2	44,8	19,8	8,3	27,7	35,2
OpenVINO	CPU	22,1	44,8	19,7	8,3	27,7	35,3
OpenVINO	VPU	22,5	45,3	20,1	8,3	27,8	35,4
Kvantovaný	iGPU	22,3	44,7	20,1	8,3	27,6	35,0
Kvantovaný	CPU	22,4	44,8	20,2	8,3	27,6	35,0

Tabulka 7.4: Model WIDER FACE ResNet50 – výsledky detekce na validačním datasetu

Nahrazení MobileNetu ResNetem50 přineslo zvýšení přesnosti o 10 procentních bodů. Hodnota 47,1 % AP@0.5 je vůbec nejvyšší naměřená přesnost ze všech měření na validačním datasetu. S akcelerací modelu a kvantováním, se rapidně snížila velikost modelu, ale došlo jen k nepatrnému snížení přesnosti a hodnoty *recall* v řádech procent.

Nejlepším natrénovaným modelem se tak ukázal být model SSD ResNet50 ve verzi WIDER FACE, který byl dále porovnán s existujícími řešeními v rámci testů na validačním datasetu.

Z validačního datasetu byly vybrány 2 snímky jako ukázkové výstupy detekce. Na obrázcích 7.4 a 7.5 jsou tyto snímky znázorněny (včetně vyznačení obličejí dle anotačních souborů).

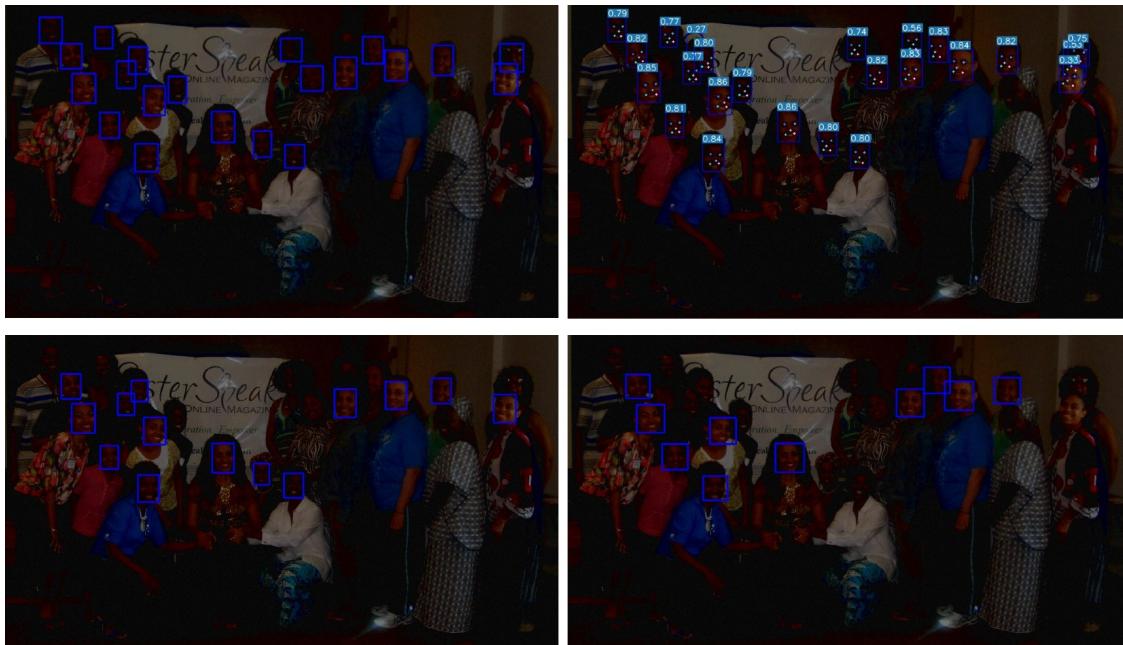


Obrázek 7.4: První vybraný ukázkový snímek z validačního datasetu. Vlevo je augmentovaný originál, vpravo jsou zeleně zobrazeny hraniční obdélníky jednotlivých obličejů.



Obrázek 7.5: Druhý vybraný ukázkový snímek z validačního datasetu. Vlevo je augmentovaný originál, vpravo jsou zeleně zobrazeny hraniční obdélníky jednotlivých obličejů.

První ukázka na obrázku 7.6 obsahuje výstupy 4 různých detektorů (výše zmíněný, nově vytvořený WIDER FACE ResNet50, YOLOv7-face, MTCNN a Viola-Jones), které jsou podrobnejší srovnány v sekcích 7.3, 7.4 a 7.5. Z výsledků je zřejmé, že nový detektor nedokázal detektovat 2 obličeje, zatímco YOLOv7-face pouze jeden. YOLOv7-face ovšem detekovalo i jeden obličeji nesprávně. Obě detekční sítě však zcela jasné překonaly jak MTCNN, tak i Viola-Jones detektor.



Obrázek 7.6: Výsledky detekce ve snímku číslo 1. Detektory nahoře zleva: nově natrénovaný WIDER FACE ResNet50, YOLOv7-face. Detektory dole zleva: MTCNN, Viola-Jones.

Na druhém snímku (obrázek 7.7) je vidět, problém detektorů MTCNN a Viola-Jones detekovat tváře ve špatně osvětleném prostředí (levý horní roh fotografií). Naměřené výsledky detektorů YOLOv7-face i WIDER FACE ResNet50 jsou téměř stejné. Ačkoliv tedy

výstupy COCO metriky pro hodnoty *precision* a *recall* z tabulek v této sekci nedosahují hodnot obvyklých v jiných pracích (80+ %), je zřejmé, že nový detektor je schopen detektovat obličeje poměrně dobře. Výsledná hodnota AP@0.50 = 47,1 % tak nutně neznamená špatný detektor.



Obrázek 7.7: Výsledky detekce ve snímku číslo 2. Detektory nahoře zleva: nově natrénovaný WIDER FACE ResNet50, YOLOv7-face. Detektory dole zleva: MTCNN, Viola-Jones.

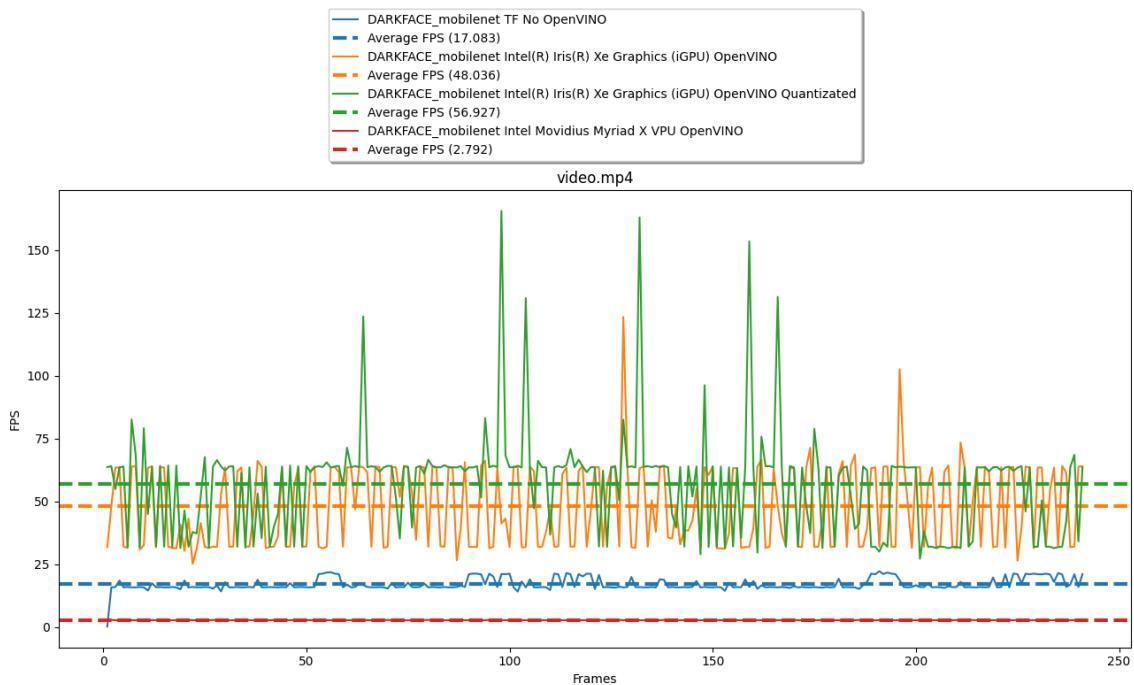
## Testování ve videích

K experimentům a měření metrik ve videích bylo využito vytvořené GUI rozhraní. Jak již bylo zmíněno v sekci 6.6, posuzovány byly tyto ukazatele: **průměrné FPS**, **průměrná přesnost mezi snímky – APF** a **průměrné IoU mezi snímky**. Následující tabulky obsahují naměřené informace. Jednotkou každého čísla u APF a IoU je %, u FPS je to počet snímků za sekundu. Měření probíhalo s prahem jistoty detekce (*confidence*)  $c = 0, 3$ . Pro hodnotu FPS je určující čas mezi začátkem detekce a vykreslením hraničních obdélníků. Tento čas je počítán i s vykreslením, aby simuloval detektor v reálném použití. Měření probíhalo na stejných zařízeních jako měření na validačním datasetu. Aplikace byla jediným spuštěným programem. K měření byl využit operační systém Windows 11. Notebook, na němž měření probíhalo, disponuje 16 GB RAM.

Model	Zař.	Video 1			Video 2		
		ø FPS	ø APF	ø IoU	ø FPS	ø APF	ø IoU
TensorFlow	GPU	17,083	14,64	36,24	17,171	5,52	22,44
OpenVINO	iGPU	48,036	13,34	33,89	41,996	5,16	22,46
OpenVINO	CPU	22,775	13,31	33,73	17,754	5,09	22,47
OpenVINO	VPU	2,792	12,87	34,42	2,786	4,70	23,51
Kvantovaný	iGPU	56,927	12,89	34,80	43,308	5,15	21,73
Kvantovaný	CPU	48,774	12,60	33,70	49,732	5,25	21,45

Tabulka 7.5: Model DARK FACE MobileNet – výsledky detekce na testovacích videích

Dle předpokladů se ukázala rychlosť detektoru MobileNet jak je vidno z tabulky 7.5. Kvantovaný model dosahuje rychlosti téměř 57 FPS u prvního videa, což z něj dělá nejrychlejší model v testování. Přesnost napříč snímky v tomto videu se pohybuje okolo 13 %. Graf na obrázku 7.8 ukazuje průběh zaznamenaných hodnot FPS napříč snímky videa číslo 1. K zobrazení byly vybrány některé detektory z tabulky 7.5. Jde vidět výrazné zrychlení mezi základní TensorFlow verzí a OpenVINO modelem, potažmo kvantovaným modelem. Rychlosť detekce pomocí NCS2 je sice pomalá, ale alespoň má téměř konstantní průběh. Grafy k dalším modelům a videím jsou uloženy na přiloženém datovém médiu.



Obrázek 7.8: Graf průběžných hodnot FPS v testovacím videu číslo 1.

Experimenty provedené ve druhém z testovacích videí ukazují na to, že video je z hlediska detektoru náročnější (APF cca 5 %). Nejrychlejší modelem je zde překvapivě kvantovaná verze běžící na procesoru. Průměrná hodnota IoU se v obou případech pohybuje kolem 35 %, respektive 22 %.

Model	Zař.	Video 1			Video 2		
		ø FPS	ø APF	ø IoU	ø FPS	ø APF	ø IoU
TensorFlow	GPU	4,767	0	4,3	4,773	5,19	42,04
OpenVINO	iGPU	7,644	0	0	7,888	3,24	37,16
OpenVINO	CPU	1,516	0	0	1,446	3,03	36,76
OpenVINO	VPU	0,992	0	0	0,992	3,68	36,15
Kvantovaný	iGPU	14,082	0	0	13,657	1,98	34,26
Kvantovaný	CPU	5,381	0	0	5,413	2,37	33,26

Tabulka 7.6: Model DARK FACE Resnet50 – výsledky detekce na testovacích videích

Model založený na ResNet50 (viz tabulka 7.6), který byl trénován datasetem DARK FACE nedokázal v prvním videu detekovat nic. Tento jev byl pravděpodobně dán možným přetrénováním modelu (*overfitting*), kdy model dokáže detekovat obličeje v datasetu DARK FACE, ale jinde jsou jeho výsledky špatné.

V případě druhého videa se přesnost detekce zásadně neliší od ostatních modelů, alespoň u TensorFlow verze. Maximální rychlosť detekce pak byla naměřena modelu, který byl kvantovaný a běžel na integrované grafické kartě Intel (14,082 FPS).

Model	Zař.	Video 1			Video 2		
		ø FPS	ø APF	ø IoU	ø FPS	ø APF	ø IoU
TensorFlow	GPU	17,327	20,03	36,34	16,855	6,5	22,14
OpenVINO	iGPU	35,649	20,26	35,69	40,890	6,96	22,76
OpenVINO	CPU	19,192	20,73	35,95	17,234	6,97	22,67
OpenVINO	VPU	2,782	22,47	38,11	2,797	8,22	26,08
Kvantovaný	iGPU	47,944	19,79	43,08	44,808	8,97	30,82
Kvantovaný	CPU	45,272	19,73	42,25	42,600	8,78	30,58

Tabulka 7.7: Model WIDER FACE MobileNet – výsledky detekce na testovacích videích

Se změnou trénovacích dat přišlo, podobně jako v případě detekce na validačním datasetu, navýšení přesnosti. Podle naměřených hodnot z tabulky 7.7 je v případě prvního videa APF nejvyšší při spuštění optimalizovaného modelu na NCS2 (22,47 %). Nejvyššího poměru průniku detekčních obdélníků k jejich sjednocení (IoU) dosáhl kvantovaný model detekující na iGPU (43,08 %). Ten je zároveň i nejrychlejším modelem pro obě videa (47,944 a 44,808 FPS).

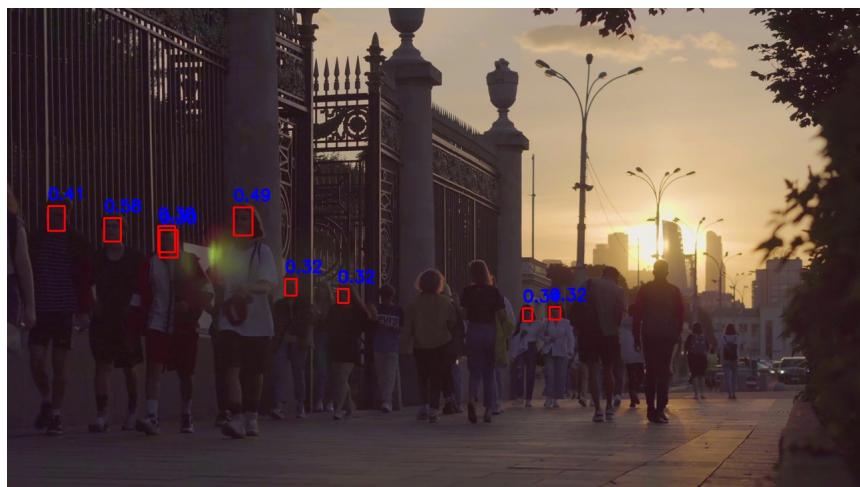
Model	Zař.	Video 1			Video 2		
		ø FPS	ø APF	ø IoU	ø FPS	ø APF	ø IoU
TensorFlow	GPU	4,773	28,53	51,55	4,772	5,10	32,88
OpenVINO	iGPU	7,547	29,01	51,65	7,686	5,07	32,73
OpenVINO	CPU	1,462	29,03	51,82	1,438	5,01	32,65
OpenVINO	VPU	0,991	28,69	52,00	0,991	4,95	33,09
Kvantovaný	iGPU	13,924	28,18	51,52	13,727	4,68	32,39
Kvantovaný	CPU	5,465	27,32	52,11	5,279	4,70	32,56

Tabulka 7.8: Model WIDER FACE Resnet50 – výsledky detekce na testovacích videích

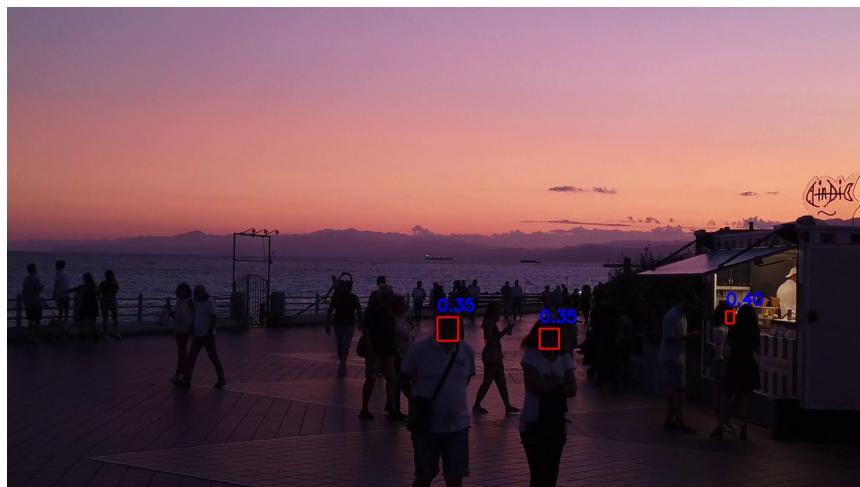
Model WIDER FACE ResNet50 (jehož výsledky shrnuje tabulka 7.8) je jediným na-trénovaným modelem, jehož průměrná hodnota IoU u prvního z experimentačních videí neklesla pod 50 %. Dosahuje také nejlepších výsledků přesnosti detekce v prvním videu.

V situaci kdy porovnáme výsledky přesnosti mezi modely u druhého testovacího videa, je nejlepší modelem MobileNet, který dosahuje 8,97 % APF. Nízké hodnoty přesnosti jsou dány charakterem videa číslo 2, které je velmi tmavé a obsahuje i velmi malé obličeje.

Na obrázcích 7.9 a 7.10 jsou zobrazeny detekce v náhodných snímcích z videí 1 a 2. Detekce probíhala detektorem WIDER FACE ResNet50.



Obrázek 7.9: Snímek z detekce v testovacím videu číslo 1 s detektorem WIDER FACE ResNet50.



Obrázek 7.10: Snímek z detekce v testovacím videu číslo 2 s detektorem WIDER FACE ResNet50.

### 7.3 Porovnání s YOLOv7-face

Prvním existujícím řešením, s nímž byly nově vytvořené modely srovnány je YOLOv7-face<sup>2</sup>. YOLOv7-face vychází z detektoru YOLOv7 [?]. YOLOv7 je nový detektor objektů dosahující přesnosti 51,4 % na COCO datasetu. Jeho přetrénovaním na detekci obličejů vznikl detektor YOLOv7-face. Detektor je dostupný v podobě GitHub repozitáře a je postaven na frameworku PyTorch.

#### Testování na datasetu

Repozitář obsahuje skript `detect.py` s jehož pomocí je možné spustit detekci v obrázcích nebo ve videu. Skript tak byl spuštěn nad validačním datasetem a výstupní souřadnice obličejů byly převedeny do výsledného COCO formátu. Následně bylo provedeno vyhodnocení jako u všech ostatních modelů a byly naměřeny výsledky, které ukazuje tabulka 7.9 (všechny čísla mají jako jednotku %, práh jistoty  $c = 0,25$ ). K měření posloužila stejná grafická karta NVIDIA MX 450 2GB.

Model	AP	AP@0.5	AP@0.75	Recall1	Recall10	Recall100
YOLOv7-face	21,4	42,8	19,6	5,6	20,8	30,7

Tabulka 7.9: Model YOLOv7-face – výsledky detekce na validačním datasetu.

Z výsledků lze vypozorovat, že průměrná přesnost je nižší než u nejlepšího natrénovaného modelu (SSD ResNet50 s přesností AP@0.50 47,1 % – viz tabulka 7.4). O něco hůře vychází ze srovnání YOLOv7-face také v případě hodnoty *recall*, která je nižší o 2,7 % ( $5,6\% \times 8,7\%$ ) až 7,4 % ( $30,7\% \times 38,1\%$ ).

V porovnání s architekturou MobileNet + SSD (tabulka 7.3 – verze akcelerovaná na NCS2) je YOLOv7-face detektor o 6,5 procentního bodu přesnější (v kategorii AP@0.50).

#### Testování ve videích

Model YOLOv7-face nebylo možné importovat do GUI aplikace z důvodu nekompatibility některých knihoven mezi PyTorch a TF OD API. Proto byla spuštěna detekce v obou testovacích videích pomocí skriptu `detect.py`. Skript na konci běhu vypisuje dobu zpracovávání detekce. Tento čas byl u videa 1 roven 42,789 s, u videa 2 pak 115,998 s. Ze známosti času a frekvence videí (25 FPS a 29 FPS) bylo vypočteno průměrné FPS, které je zapsáno v tabulce 7.10.

Z dalších naměřených výsledků je patrné, že přesnost detekce v prvním videu je podobná přesnosti modelu WIDER FACE ResNet50 (tabulka 7.8). U druhého videa pak YOLOv7-face předčilo v přesnosti detekce (11,76 %) všechny testované existující i nová řešení. Rychlosť okolo 6 FPS v obou nahrávkách je zhruba poloviční ve srovnání s kvantovaným modelem WIDER FACE ResNet50.

Model	Video 1			Video 2		
	ø FPS	ø APF	ø IoU	ø FPS	ø APF	ø IoU
YOLOv7-face	5,632	28,46	43,68	6,034	11,76	39,29

Tabulka 7.10: Model YOLOv7-face – výsledky detekce na testovacích videích.

<sup>2</sup><https://github.com/derronqi/yolov7-face>

## 7.4 Porovnání s Viola-Jones

Další zvoleným existujícím řešením je klasický algoritmus **Viola-Jones/Haar Cascades**. Tato část se zaměřuje na porovnání vytvořených detektorů s tímto detektorem. Detektor Viola-Jones, který byl použit k testování je obsažen v knihovně OpenCV<sup>3</sup>. Pomocí příkazu `cv2.CascadeClassifier()` je možné vytvořit objekt detektoru. Detektor ke správnému fungování potřebuje speciální XML soubor<sup>4</sup>. Po načtení souboru je možné funkci `detectMultiScale()` detekovat obličeje v černobílé verzi vstupního obrázku.

Detekce byla opět provedena jak na validačním datasetu, tak ve videích, kdy byl detektor Viola-Jones přímo zabudován do GUI aplikace. Lze tak v libovolném mp4 videu spustit detekci tímto detektorem. Následující oddíly se věnují pokusům s detektorem.

### Testování na datasetu

Výsledky měření (viz tabulka 7.11) ukazují velmi nízkou úspěšnost detekce. Ta je dána jak zastaralostí a nedokonalostí této technologie, tak také omezením detekce podle specifického XML souboru. Použitý soubor (*haarcascade\_frontalface\_alt.xml*) se zaměřuje na detekci tváří, které jsou v přímém pohledu do kamery/fotoaparátu, což je značně limitní.

Přesnost detekce i poměr správných detekcí zhruba odpovídají vytvořenému detektoru SSD MobileNet, který byl natrénován na datasetu DARK FACE (tabulka 7.1). Nejlepší natrénovaný model SSD Resnet50 má cca  $5 \times$  vyšší přesnost detekce AP@0.50.

Model	AP	AP@0.5	AP@0.75	Recall1	Recall10	Recall100
Viola-Jones	3,8	9,8	0,9	1,6	4,3	5,2

Tabulka 7.11: Model Viola-Jones – výsledky detekce na validačním datasetu.

### Testování ve videích

Detektor překvapil vysokou úspěšností detekce v 1. testovacím videu (22,54 % průměrná přesnost mezi snímky). Rychlostně byl detektor pomalejší než výše zmíněné YOLOv7-face i většina natrénovaných detektorů. Ve druhé nahrávce nedetekoval model téměř nic a ukázal se tak velmi nevhodný pro detekci ve zhoršených světelných podmínkách. Všechny naměřené metriky v obou záznamech zobrazuje tabulka 7.12.

Model	Video 1			Video 2		
	$\phi$ FPS	$\phi$ APF	$\phi$ IoU	$\phi$ FPS	$\phi$ APF	$\phi$ IoU
Viola-Jones	3,141	22,54	42,95	5,182	0,23	1,44

Tabulka 7.12: Model Viola-Jones – výsledky detekce na testovacích videích.

## 7.5 Porovnání s MTCNN

Detektor MTCNN<sup>5</sup> (Multi-Task Cascaded Convolutional Network) [?] byl vybrán jako poslední existující řešení ke srovnání úspěšnosti detekce. Jedná se o kaskádu neuronových sítí,

<sup>3</sup>[https://docs.opencv.org/3.4/db/d28/tutorial\\_cascade\\_classifier.html](https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html)

<sup>4</sup><https://github.com/opencv/opencv/blob/master/data/haarcascades/>

*haarcascade\_frontalface\_alt.xml*

<sup>5</sup><https://github.com/ipazc/mtcnn>

která se používá k detekci a zarovnání obličejů. Není příliš rychlá, ovšem dle [?] dosahuje dobrých výsledků detekce v základním neupraveném datasetu WIDER FACE.

V práci byla síť testována opět jak na validačním datasetu, tak v testovacích videích. Díky tomu, že lze síť stáhnout jako Python balíček v prostředí pip, bylo možné ji implementovat do GUI aplikace. Měření ve videích tak probíhá ve stejném prostředí jako v případě natrénovaných modelů a detektoru Viola-Jones. Pro provedení detekce v datasetu byl vytvořen speciální Python skript.

### Testování na datasetu

Proběhlé měření (tabulka 7.13) ukázalo výsledky přesnosti téměř poloviční oproti detektoru WIDER FACE ResNet50. Hodnota *recall* dopadla pro MTCNN výrazně hůře ( $14,8\% \times 38,1\%$ ). Výsledky jsou horší i v případě srovnání s YOLOv7-face, ze srovnání se starým algoritmem Viola-Jones pak vychází MTCNN o poznání lépe. Největším problémem MTCNN je však rychlosť zpracování obrázku, což demonstруje následující text.

Model	AP	AP@0.5	AP@0.75	Recall1	Recall10	Recall100
MTCNN	11,7	23,1	9,9	4,4	12,2	14,8

Tabulka 7.13: Model MTCNN – výsledky detekce na validačním datasetu.

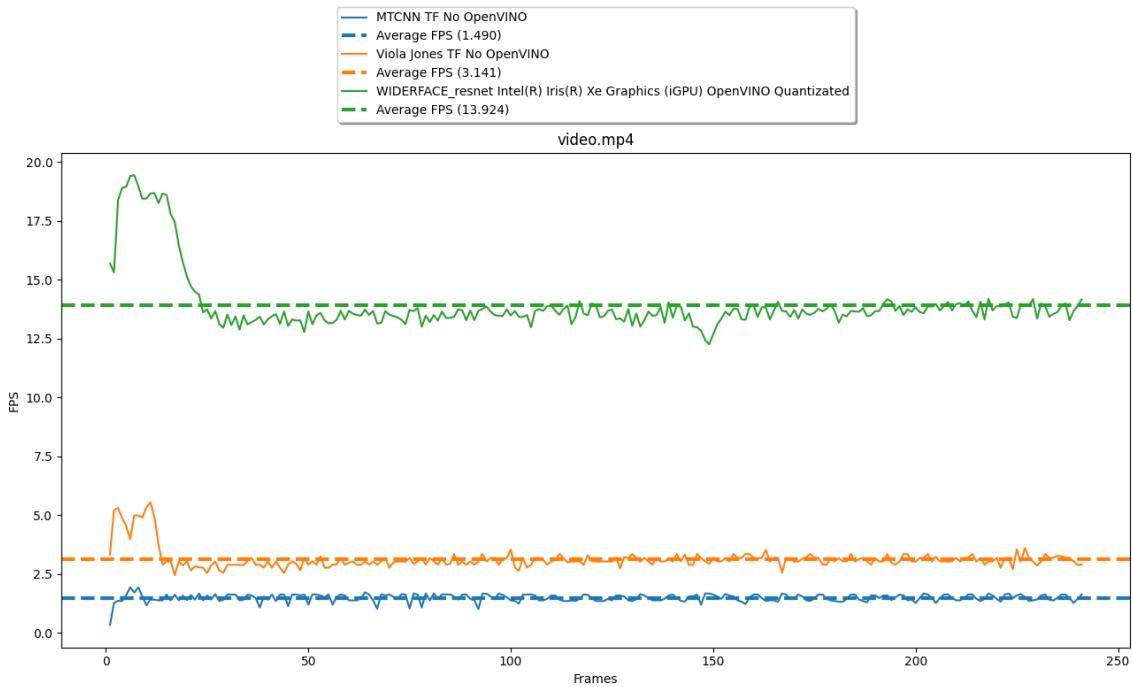
### Testování ve videích

MTCNN z měření vyšlo jako jeden z nejpomalejších detektorů, kdy naměřené hodnoty průměrné snímkové frekvence nepřekročily ani v jednom případě hranici 1,5 FPS. Podobně jako detektor Viola-Jones, byl model schopen detektovat obličeje jen v 1. záznamu (APF 15,40 %). Výsledek je podobný jako v případě detektoru DARK FACE MobileNet (tabulka 7.5), který je však díky architektuře MobileNetu výrazně rychlejší.

Model	Video 1			Video 2		
	$\varnothing$ FPS	$\varnothing$ APF	$\varnothing$ IoU	$\varnothing$ FPS	$\varnothing$ APF	$\varnothing$ IoU
MTCNN	1,490	15,40	53,79	1,497	0	11,08

Tabulka 7.14: Model MTCNN – výsledky detekce na testovacích videích.

Následující graf (na obrázku 7.11) porovnává rychlosť zpracování snímku v nejlepším novém detektoru a dvou existujících řešeních (Viola-Jones a MTCNN). Je vidět jasná převaha kvantované verze, což spolu s výšší přesností detekce dělá z nového řešení vítěze srovnání.



Obrázek 7.11: Graf průběhu hodnot FPS v testovacím videu 1 za použití detektorů WIDER FACE Resnet50 v kvantované verzi na iGPU, detektoru Viola-Jones a sítě MTCNN.

## 7.6 Detekce s metodou Mirnet

V tomto testu byly vybrané modely otestovány na testovacích videích, které byly upraveny metodou Mirnet. Výsledky měření přináší tabulka 7.15. Všechny měření kromě YOLOv7-face byly opět provedeny pomocí GUI aplikace. Pro model YOLOv7-face byl výpočet FPS proveden stejným způsobem, který byl popsán sekci 7.3 (časy jednotlivých detekcí jsou 46,008 s a 128,909 s).

Model	Mirnet Video 1			Mirnet Video 2		
	$\phi$ FPS	$\phi$ APF	$\phi$ IoU	$\phi$ FPS	$\phi$ APF	$\phi$ IoU
WF MobileNet (TF+GPU)	16,844	14,32	28,24	16,922	9,38	24,07
WF ResNet50 (TF+GPU)	4,760	27,19	49,00	4,649	11,80	40,04
WF MobileNet (kv.+iGPU)	46,193	14,51	33,40	46,570	9,54	31,70
WF ResNet50 (kv.+iGPU)	13,157	27,06	48,18	13,74	11,77	40,95
Viola-Jones	3,156	27,15	59,38	3,973	9,57	54,07
MTCNN	1,332	11,75	4788	1,317	9,85	45,61
YOLOv7-face	5,238	23,86	35,90	5,430	13,66	37,22

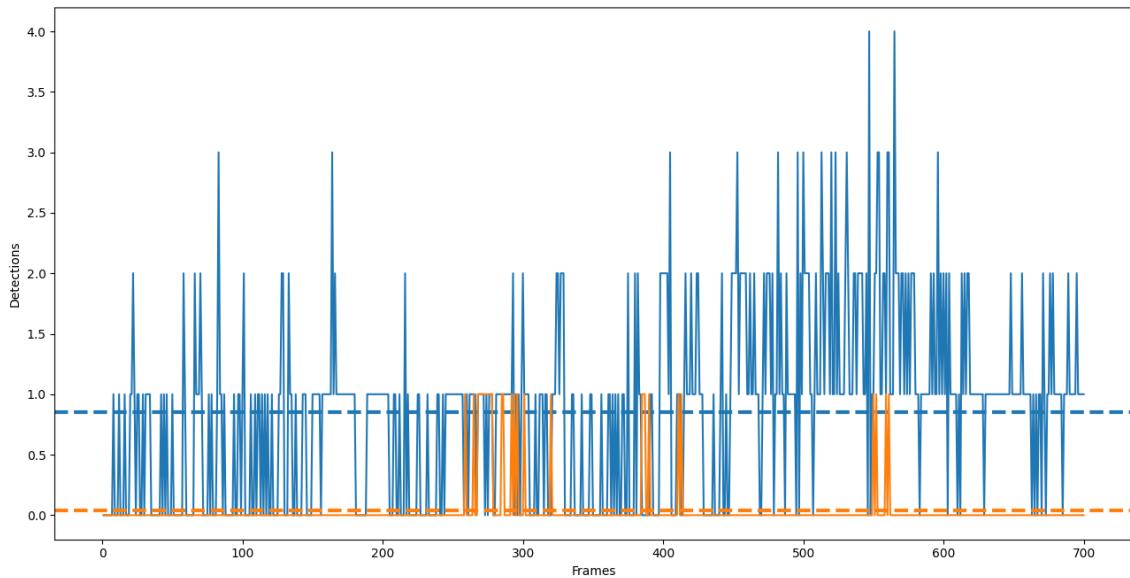
Tabulka 7.15: Mirnet – výsledky detekce na testovacích videích.

K měření byly vybrány všechny existující řešení a modely natrénované na datasetu WIDER FACE (základní TF verze a kvantovaná verze). Změřené rychlosti detekce se příliš neliší od hodnot naměřených na originálních videích.

Výsledky přesnosti u videa číslo 1 jsou, až na jednu výjimku (Viola-Jones), horší než při měření originálu. To je dáno tím, že toto video není dostatečně špatně osvětlené, tak aby se projevil účinek sítě Mirnet.

V případě druhého videa, je patrné zlepšení přesnosti všech modelů. Model MobileNet dokázal detektovat obličeje s průměrnou přesností 6,5 % bez použití Mirnetu, s Mirnetem pak s přesností 9,38 % v TensorFlow verzi. Kvantovaná verze dosahuje téměř stejného výsledku (8,97 %  $\times$  9,54 %). Dvojnásobný nárušt APF zaznamenalo použití Mirnetu u detektoru SSD + ResNet50, kdy bez použití metody byla APF 5,10 % pro TensorFlow verzi a 4,68 % pro kvantovanou verzi modelu. S Mirnetem se průměrná přesnost TensorFlow varianty zvedla na 11,80 % a na 11,77 % v případě kvantované verze.

Pro detektor YOLOv7-face pak došlo k nárustu o cca 2 procentní body (11,76 %  $\times$  13,66 %). Viola-Jones i MTCNN zaznamenaly APF na hranici 10 %, přičemž bez použití Mirnetu měly oba detektory (téměř) nulovou průměrnou přesnost. Nejvíce je efekt použití Mirnetu vidět na grafu v obrázku 7.12. V originálním videu dokázala síť MTCNN detektovat průměrně 0,0043 obličejů/snímek, při využití metody se tato hodnota zvedla na 0,856 obličejů/snímek.



Obrázek 7.12: Graf počtu detekcí napříč snímkami testovacího videa 2 s řešením MTCNN. Oranžově při detekci bez Mirnetu, modře při detekci ve videu upraveném Mirnetem.

## 7.7 Zhodnocení experimentů

Byly provedeny experimenty, s cílem porovnat reálné schopnosti detektorů obličejů ve zhoršených světelných podmínkách. Z nových řešení vyšel nejlépe model WIDER FACE ResNet50, který byl nejlepším modelem v kategorii přesnosti při detekci obličejů ve snímcích z validačního datasetu (47,1 % AP@0.50 – TensorFlow verze). Při detekci ve videu číslo

1 byl tento model opět nejlepší ze všech (APF 29,03 % – optimalizovaná verze běžící na CPU).

U detekce ve druhém testovacím videu bylo nejlepší nové řešení to s modelem WIDER FACE MobileNet (APF 8,97 % – kvantovaná verze spuštěná na integrované grafické kartě). Nejlepšího výsledku ovšem dosáhl detektor YOLOv7-face, a to jak ve variantě bez Mirnetu, tak i s touto metodou, který tak prokázal všeobecnost detekce (APF 11,76 % a AP 13,66 %).

V rychlosti byly bezkonkurenční modely založené na architektuře MobileNet, jejichž kvantované verze vždy překonaly hranici 40 FPS při průměrné snímkové frekvenci. Pro co nejlepší detekci je tak vhodný model WIDER FACE ResNet50 či existující model YOLOv7-face, při požadavku na real-timovou detekci je relevantní modelem WIDER FACE MobileNet.

# Kapitola 8

## Závěr

V rámci této práce byla představena problematika detekce obličejů v reálných podmínkách, dále byly popsány problémy omezující detekci a klasické algoritmy detekce obličejů. V samostatných kapitolách a sekcích byly popsány neuronové sítě (obecně, konvoluční) a algoritmy pro detekci založené na neuronových sítích. Dále pak existující komerční a nekomerční řešení, systémy detekce a možnosti akcelerace neuronových sítí specializovaným hardwarem (Intel Neural Compute Stick 2).

Následovaly kapitoly zaměřené na návrh a implementaci detekčních modelů pro detekci obličejů ve zhoršených světelných podmínkách, návrh grafické aplikace pro uživatele a nástrojů k vyhodnocování experimentů. V těchto kapitolách bylo představeno TensorFlow Object Detection API, na němž jsou nově vzniklé modely postaveny. Poté byly prezentovány postupy trénování, tvorby datasetu, byly zvoleny vyhodnocovací metriky a modely byly akcelEROVány pomocí knihovny OpenVINO. Jako jedna z možností zlepšení detekce byla představena a použita metoda pro vylepšení kvality obrázku Mirnet.

Nakonec byly provedeny testy za účelem porovnání výkonnosti (rychlosti) a přesnosti detektorů ve zhoršených světelných podmínkách. Testy byly složeny ze dvou částí: detekce ve fotografiích z datasetu a detekce v testovacích videích.

Podařilo se natrénovat detektor dosahující v některých aplikacích vyšší přesnosti detekce než *state-of-the-art* řešení YOLOv7-face. Proces kvantování modelů pomocí knihovny OpenVINO dokázal výrazně zrychlit detekci při zachování téměř stejné přesnosti modelů. Důležitým výsledkem je naměřená rychlosť modelů využívajících MobileNet, které dosahovaly při měření stabilně rychlosti přes 40 FPS a jsou tak vhodné pro real-timeovou detekci. Byla také vytvořena grafická aplikace umožňující vyzkoušet detekci v libovolném mp4 videu.

Jelikož je detekce obličejů vstupním bodem pro mnoho dalších aplikací, mohla by práce být dále rozšířena o rozpoznávání tváří, sledování (trackování) osob nebo vytváření teplotních map podle výskytu obličejů.

## Příloha A

# Obsah přiloženého datového média

Na přiložené SD kartě se nacházejí všechny soubory využité k trénování, všechny vytvořené skripty (Python, Jupyter Notebooks) a textové návody ke GUI aplikaci a k dalším náležitostem. Následující seznam obsahuje popis hierarchie uložených modelů:

- A
-