Aplikace pro klasifikování vybraných charakteristik lidského obličeje s využitím umělé inteligence

Martin Sedláček

Bakalářská práce
2024

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav informatiky a umělé inteligence

Akademický rok: 2023/2024

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Martin Sedláček**
Osobní číslo:	**A21275**
Studijní program:	**B0613A140020 Softwarové inženýrství**
Forma studia:	**Prezenční**
Téma práce:	**Aplikace pro klasifikování vybraných charakteristik lidského obličeje s využitím umělé inteligence**
Téma práce anglicky:	**Application for Classification of Selected Human Facial Characteristics Using Artificial Intelligence**

Zásady pro vypracování

1. Proveďte literární rešerši metod v oblasti segmentace lidského obličeje.
2. Zvolte, vytvořte nebo doplňte vhodný dataset pro segmentaci lidského obličeje s následnou klasifikací vybraných charakteristik.
3. Dotrénujte vhodný předtrénovaný model, např. YOLOv8.
4. Vyhodnoťte výsledný model dle standardně používaných metrik.
5. Nejlepší model konvertujte do formátu ONNX pro další využití.

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. ROSEBROCK, Adrian. Deep learning for computer vision with Python: Practitioner Bundle. PyImageSearch, 2017. ISBN 978-1-986723817.
2. ROSEBROCK, Adrian. Deep learning for computer vision with Python: Starter bundle. PyImageSearch, 2017. ISBN 9781986538138.
3. ROSEBROCK, Adrian. Deep learning for computer vision with Python: ImageNet bundle. PyImageSearch, 2017. ISBN 978-1-986723848.
4. GOODFELLOW, Ian, Yoshua BENGIO a Aaron COURVILLE. Deep learning: ImageNet bundle. Adaptive computation and machine learning. Cambridge, Massachussetts ; London: The MIT Press, 2016. ISBN 9780262035613.
5. TERVEN, Juan; CORDOVA-ESPARZA, Diana. A comprehensive review of YOLO: From YOLOv1 to YOLOv8 and beyond. *arXiv preprint arXiv:2304.00501*, 2023.
6. ZHAO, Xu, et al. Fast Segment Anything. *arXiv preprint arXiv:2306.12156*, 2023.

Vedoucí bakalářské práce: **prof. Ing. Zuzana Komínková Oplatková, Ph.D.**
Ústav informatiky a umělé inteligence

Datum zadání bakalářské práce: **5. listopadu 2023**
Termín odevzdání bakalářské práce: **13. května 2024**

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
DĚKAN
L.S.

doc. Ing. Jiří Vojtěšek, Ph.D. v.r.
děkan

prof. Mgr. Roman Jašek, Ph.D., DBA v.r.
ředitel ústavu

Ve Zlíně dne 5. ledna 2024

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne 9.5.2024 Martin Sedláček, v. r.

ABSTRAKT

Tato práce se zabývá úvodní fází vývoje systému, který získává informace o osobách, které se nacházejí v blízkosti reklamních panelů. Práce se zaměřuje tvorbou a vyhodnocením modelů umělé inteligence pro detekci osob a segmentaci lidského obličeje. Dále obsahuje analýzu metod v oblasti počítačového vidění. V rámci práce byly vytvořeny modely, které jsou základem pro samostatný open-source systém využitelný v oblasti marketingu pro tvorbu statistiky o pohybu osob ve vybraných prostorech.

Klíčová slova: umělá inteligence, umělé neuronové sítě, detekce objektů, segmentace lidského obličeje, YOLOv8.

ABSTRACT

This paper deals with the initial phase of the development of a system that obtains information about people who are in the vicinity of advertising panels. The work focuses on the creation and evaluation of artificial intelligence models for person detection and human face segmentation. It also includes an analysis of methods in the field of computer vision. Within the framework of the thesis, models have been created, which are the basis for a stand-alone open-source system usable in the field of marketing for the creation of statistics on the movement of people in selected spaces.

Keywords: artificial intelligence, artificial neural networks, object detection, human face segmentation, YOLOv8.

Chtěl bych poděkovat své vedoucí práce paní prof. Ing. Zuzaně Komínkové Oplatkové, Ph.D. za ochotu, trpělivost a odborné vedení během mého psaní bakalářské práce. Dále bych chtěl poděkovat svým nejbližším za podporu během celého studia.

Prohlašuji, že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD

Důvodem pro zvolení tohoto tématu byla spolupráce s firmou, která instaluje reklamní panely do prostorů jako je čekárna u lékaře, pošta nebo další veřejné prostory. Hlavní myšlenkou bylo vytvořit open-source projekt, který zpracovává obraz z kamery nacházející se poblíž reklamního panelu a dokáže vyhodnotit počet lidí, kteří se v těchto prostorech nacházejí. Dále se taky zohledňovala možnost, aby systém dokázal vytvořit takový výstup, aby ho bylo v budoucnu možné použít pro další modely, které budou dle konkrétního případu použití vyhodnocovat specifické charakteristiky jednotlivých osob.

Tato práce by měla přispět ve vývoji tohoto systému tvorbou modelů pro detekci jednotlivých osob a segmentaci obličeje těchto osob, čímž bude získávat informace, které budou určeny k dalšímu zpracování. Mezi další zpracování může patřit vyhodnocení pózy hlavy, výrazu v obličeji, případně určení věku či pohlaví. Tato práce by měla analyzovat existující metody v oblasti zpracování lidského obličeje tak, aby mohla nabídnout způsob, který bude sloužit jako odrazový můstek pro získávání nejrůznějších charakteristik, tedy bude dostatečně multifunkční.

Práce musí brát v potaz, že řešení by mělo být v rámci možností nenáročné na výkon, jelikož by mělo být použitelné i na zařízeních, které nejsou určené čistě pro práci s velkými modely umělé inteligence. Zároveň by měly být modely ve formátu, který umožňuje budoucí konvertování pro využití ve speciálních zařízeních jako je TPU (Tensor Processing Unit). Dalším požadavkem je, aby systém pracoval i bez připojení k internetu kvůli tomu, aby ho bylo snadné nasadit.

I. TEORETICKÁ ČÁST

1 UMĚLÁ INTELIGENCE

Umělá inteligence je v dnešní době často zmiňovaný pojem. V poslední době můžou za rozšíření povědomí o tomto fenoménu generativní modely jako například ChatGPT spadající pod velké jazykové modely využívány pro generování textu, nebo modely pro generování obrázků jako je DALL-E nebo Midjourney. Samotný pojem AI (artificial intelligence, umělá inteligence) byl poprvé použit už v padesátých letech minulého století.

Toto odvětví cílí na tvorbu počítačových programů řešících úkony, které jsou pro člověka velmi běžné a zvládá je intuitivně, ovšem jako program jsou velmi obtížně řešitelné při využití standartních metod programování. Skvělými příklady takových úkonů jsou mimo jiné: počítačové vidění (computer vision) zabývající se rozeznáváním objektů na obrázku, porozumění kontextu obrázků či textu nebo již zmíněné generativní programy. Konkrétněji AI definoval John McCarthy jako: „*Konstrukce počítačových programů, které se zabývají úkoly, které jsou v současné době uspokojivěji vykonávány lidskými bytostmi, protože vyžadují mentální procesy na vysoké úrovni*.“

Jako AI můžou být považovány jak relativně jednoduché struktury na zpracování ručně vybraných charakteristik, tak i velmi komplexní architektury inspirovány principem lidského mozku.

1.1 Strojové učení

Strojové učení je velmi rozšířená podoblast AI zabývající se řešením aplikací s komplexními problémy. Přesněji se zabývá způsoby pro automatizování hledání správného řešení problémů, u kterých je obtížné definovat jednoznačný postup. Obvykle se systémy využívající strojové učení snaží přiblížit optimálnímu výsledku využitím nabytých zkušeností, nejsou tedy specificky naprogramovány pro konkrétní úkon. Těmto systémům je zadáno pouze, čeho mají dosáhnout, nikoliv jak toho mají dosáhnout, čímž se liší od konvenčních technik programování [2], [3].

1.2 Hluboké učení

Hluboké učení (Deep learning) je podoblast strojového učení, která se specializuje na hledání souvislostí a rozpoznávání vzorů z poskytnutých dat. Struktura těchto algoritmů je ve svém základu inspirována principem fungování mozku. Využívají se takzvané ANNs (Artificial Neural Networks, Umělé neuronové sítě), díky kterým můžeme tyto principy simulovat.

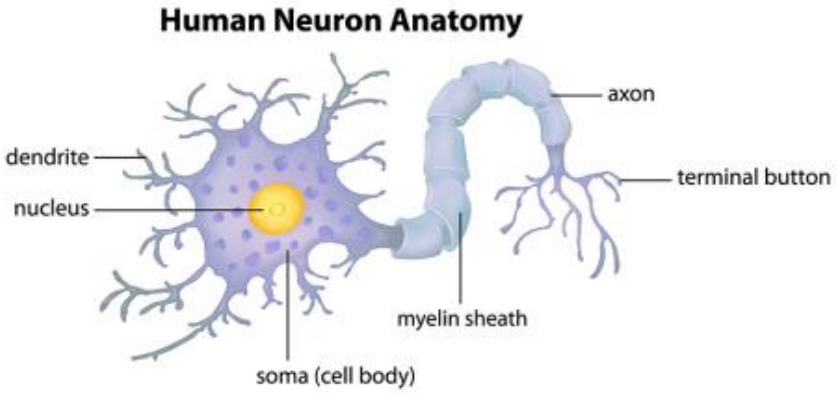
Nutno podotknout, že ANNs nejsou přímou kopií mozku a ani se o to nesnaží, přebírají pouze velmi základní logiku. Tyto sítě jsou uspořádány ve vrstvách, kde se každá vrstva skládá z jednotlivých umělých neuronů a je napojena na neurony v následující vrstvě. Ve výsledku máme tedy vrstvu vstupní a výstupní a skryté vrstvy nacházející se mezi nimi, tyto vrstvy reprezentují úrovně abstrakce (podrobněji popsáno v kapitole 1.2.1). S pojmem hluboké učení je spjata struktura využívající velkou a hlubokou neuronovou síť, kde se pojmem hluboká označuje síť s velkým počtem vrstev. Momentálně neexistuje konsensus mezi experty, který by přesně definoval počet vrstev potřebných na to, aby se program dal považovat jako hluboké učení, ale obvykle jako hluboké učení označujeme síť, která obsahuje alespoň jednu skrytou vrstvu, tedy síť obsahující více jak dvě vrstvy [1], [4], [6].

1.2.1 Základy umělých neuronových sítí

Vývoj ANNs začal již ve čtyřicátých letech minulého století. Původní modely byly lineární a relativně jednoduché. Asociovaly k výstupní hodnotě označované jako y vstupní hodnotu x. V podstatě se trénování modelu pomocí vzorků dat začalo vnímat jako utváření funkce. Standartní neuronová síť je tvořena větším počtem částí, které se označují jako neurony. Jejich úkolem je samostatně procesovat data která mají na vstupu, a v závislosti na tom se aktivovat. Tyto neurony tvoří jednotlivé vrstvy, které jsou spolu propojeny. Výstup neuronů ve vrstvě je přiveden na vstup neuronů ve vrstvě následující. Neurony vstupní vrstvy jsou aktivovány vnějším prostředím neboli daty x. Hodnoty z výstupní vrstvy jsou potom výstup y. Můžeme tedy říct, že neurony svou aktivací ovlivňují aktivace v následujících vrstvách [4], [7].

1.2.1.1 Neuron

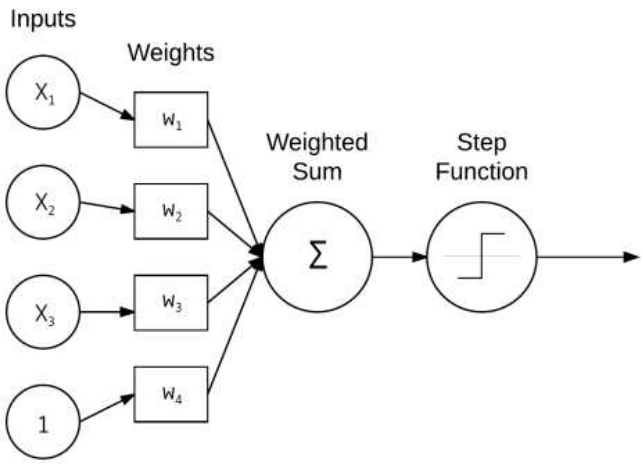
Umělé neurony, které jsou základem ANNs, byly inspirovány buňkou nervu. Každá nervová buňka přenáší signál. Dendrit slouží jako vstup přijímající signál o určité síle, tělo buňky pak jako přenosová funkce. Axon můžeme považovat jako výstup posílající signál do další nervové buňky [8]. Některá propojení jednotlivých nervů jsou silnější a signál zesilují, některá spojení naopak signál zeslabují [1].



Obrázek 1. Neuron [1]

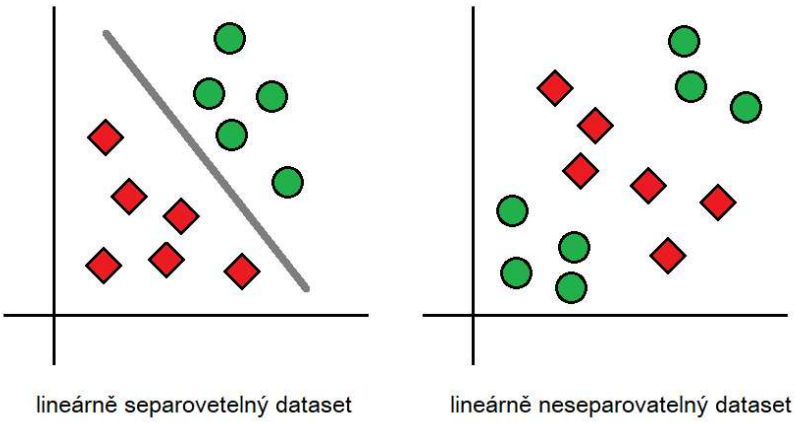
1.2.1.2 Perceptron

Jedním ze způsobů, jak napodobit princip nervu, je algoritmus perceptron, který v padesátých letech minulého století publikoval Rosenblatt [9]. Tento model se zvládal automaticky učit na vzorcích dat tím, že ladil hodnoty takzvaných vah (weights), které určují sílu jednotlivých spojení. Signály byly následně sečteny a na výslednou hodnotu je aplikována aktivační schodová funkce, která rozhoduje o výstupu [1], [8].



Obrázek 2. Perceptron [1]

Problémem preceptronu bylo, že díky lineární aktivační funkci zvládal pouze problémy, které byly lineárně spravovatelné. Tento problém byl překonán, až když se začaly používat nelineární funkce, jako je například funkce sigmoid. Z obrázku 3 jsou zřejmé rozdíly mezi lineárně separovatelnými a lineárně neseparovatelnými datasety [1], [10].



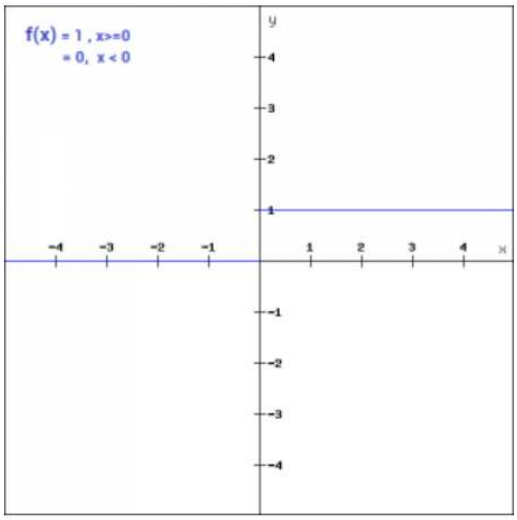
Obrázek 3. Příklad lineární oddělitelnosti datasetů

1.2.2 Aktivační funkce

Aktivační funkce se v ANNs používají pro transformaci výsledných hodnot neuronu, které následně jdou na výstup, aby byly předány neuronům v následující vrstvě. Aktivační funkce se aplikuje na součet vstupních signálů s ohledem na jejich přiřazené váhy. Některé aktivační funkce se označují jako squashing funkce. Tyto funkce mají amplitudu a tím limitují signál do určitého rozsahu [1], [11], [12].

1.2.2.1 Binární schodová funkce

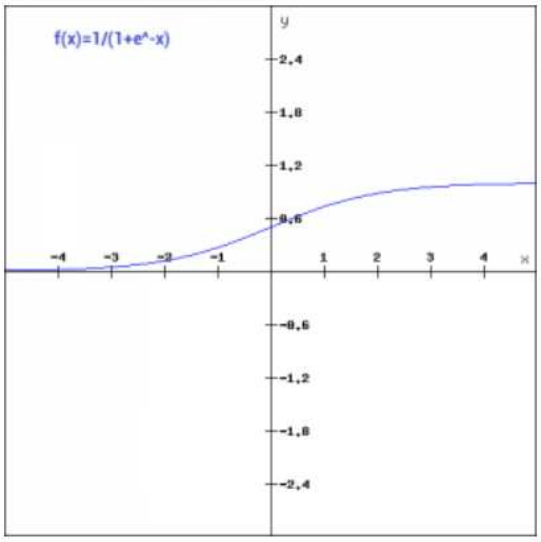
Nejjednodušší aktivační funkcí je schodová funkce. Výstup této funkce může být pouze 0 nebo 1. Je-li na vstupu funkce hodnota 0 nebo menší, výstupem je 0, je-li hodnota kladná, na výstupu je 1. V tomto případě se může brát 0 na vstupu jako prahová hodnota, která určuje, zdali je neuron aktivován, a posílá hodnotu na vstup další vrstvy. Schodová funkce má využití pouze jako binární klasifikátor. Další nevýhodou této funkce je, že má nulový gradient, a tudíž se nehodí pro trénování metodami, které jsou založeny na aproximaci minim pomocí gradientu [1], [11], [12].



Obrázek 4. Binární schodová funkce [11]

1.2.2.2 Funkce sigmoid

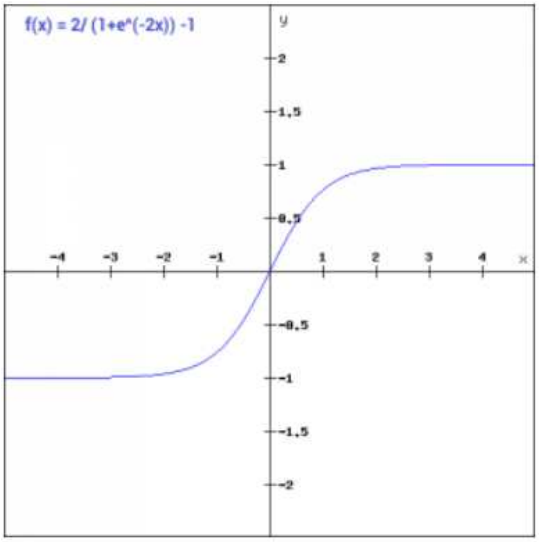
Historicky výrazně více využívaná aktivační funkce je funkce sigmoid. Výhodou této funkce je, že jsou její hodnoty jednoznačně odlišitelné. Všechny hodnoty jsou transformovány do rozsahu 0 až 1. Okolo většiny svého průběhu tato funkce saturuje. Vstupní hodnoty, které jsou velmi kladné, saturují na výstupu u hodnoty 1, a hodnoty velmi záporné saturují u hodnoty 0. Tato skutečnost zapříčiňuje, že je funkce citlivá pouze blízko nulové vstupní hodnoty. Saturované neurony mají velmi malý gradient, tudíž je trénování založené na gradientu obtížné [1], [5], [11], [12].



Obrázek 5. Funkce sigmoid [11]

1.2.2.3 Funkce hyperbolický tangens

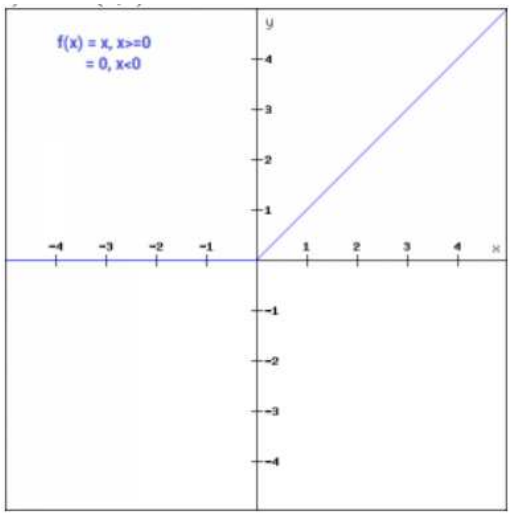
Funkce hyperbolický tangens, označovaná jako tanh (hyperbolic tangent), je podobná funkci sigmoid. Rozdílem je, že hyperbolický tangens je vystředěná na nule a transformuje hodnoty do rozsahu -1 až 1. Výhodou hyperbolického tangentu je, že oblast okolo nuly je mnohem strmější a lineárnější, nicméně hodnoty v saturaci mají stále mizivý gradient [1], [11], [12].



Obrázek 6. Funkce hyperbolický tangens [11]

1.2.2.4 Funkce ReLU

Funkce ReLU (rectified linear unit, rektifikovaná lineární jednotka) je v dnešní době spolu s jejími variacemi nejvíce využívaná aktivační funkce. Někdy se označuje jako rampová funkce kvůli jejímu průběhu, který kladné hodnoty lineárně navyšuje. Hodnoty této funkce nesaturují, tudíž se funkce hodí pro trénovací metody založené na gradientu. Základní verze ReLU transformuje negativní hodnoty na 0, čímž dociluje toho, že neurony se zápornými hodnotami nejsou aktivovány, a tudíž je výpočetně úspornější. Gradient je v tomto případě nulový. Tento problém řeší varianta ReLU označována jako Leaky ReLU, která umožňuje malý gradient v záporných hodnotách. ReLU je ve většině případů efektivnější než funkce sigmoid a tanh [1], [11].



Obrázek 7. Funkce ReLU [11]

1.2.3 Ztrátové funkce

Chceme-li natrénovat umělou neuronovou síť a vylepšit, jak si počíná ve specifickém úkonu, musíme umět vyjádřit, jak jsou momentální predikce přesné. K tomu slouží takzvaná ztrátová funkce. Jednoduše řečeno ztrátová funkce nám navrací hodnotu podle toho, jak je daná predikce „dobrá“ nebo „špatná“. Čím lépe si model počíná v budování souvislostí mezi vstupem a očekávaným výstupem, tím je hodnota ztrátové funkce nižší. Neodvádí-li model dobrou práci v predikování správných výsledků, ztrátová hodnota je vysoká. Tato informace se dále využívá pro správné nastavení hodnot vah (weights) a prahů (biases) [1].

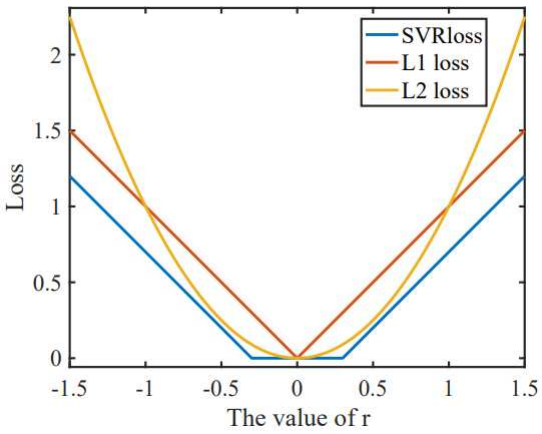
V průběhu trénování modelu by měly hodnoty ztrátové funkce postupně klesat. Ideální je, aby na konci trénování byla ztráta co možná nejnižší, je však nutno mít na paměti, že příliš malé hodnoty ztrátové funkce nemusí nutně znamenat správně natrénovaný model, ale může to být příznak takzvaného overfittingu [1].

Pro konkrétní problematiky existují specifické ztrátové funkce. Volba nesprávné funkce může výrazně negativně ovlivnit efektivnost algoritmu [13].

1.2.3.1 Ztrátové funkce regresních problémů

U regresních problémů je naším cílem vytvořit model, který predikuje nějakou reálnou hodnotu. Tyto modely se nesnaží kategorizovat vstup do nějakých předem daných tříd, namísto toho odhadují určitou hodnotu, která je spjata se vstupními daty. Dobrým příkladem regresního problému je předpověď výskytu deště, kde se snažíme získat hodnotu reprezentující pravděpodobnost z vstupních parametrů jako jsou teplota, vlhkost nebo např. tlak [15]. V takovém případě musíme umět vyjádřit míru ztráty jak u hodnot menších, než je uspokojivý výsledek, tak i u hodnot, které výsledek přesahují.

Ideální ztrátové funkce pro řešení regresních problémů jsou funkce s bilaterálním průběhem, jelikož jsou symetrické a navrací tedy stejnou ztrátovou hodnotu pro odchylky v obou směrech. Existují však i nesymetrické ztrátové funkce, které se používají v regresních problémech [14].



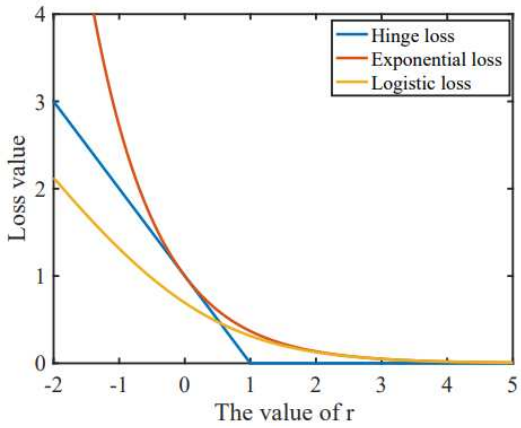
Obrázek 8. Příklad průběhů bilaterálních ztrátových funkcí [14]

Široce využívanou funkcí pro regresní problémy je střední kvadratická chyba (Mean squared error), někdy označována taky jako L2 loss. Pro výpočet střední kvadratické chyby se vezme rozdíl mezi predikcí modelu a správné hodnoty, ten se umocní a následně se zprůměruje napříč datasetem [16].

1.2.3.2 Ztrátové funkce klasifikačních problémů

Klasifikace je proces, kde je cílem zařadit data do konkrétních kategorií. U klasifikačního problému se tedy snažíme naučit model identifikovat design nebo roli vstupních dat, což nám je pomůže rozdělit do kategorických tříd. V klasifikačních problémech označovaných jako binární mají data různá označení (labels) reprezentující třídy například model rozlišující, zdali je na obrázku kočka nebo pes, případně jestli charakteristice odpovídá, tedy kočka nebo ne kočka. Dále je zde klasifikace kategorická, která rozlišuje více jak 2 třídy například, je-li na vstupu kočka, pes, myš nebo žába [15], [16].

U klasifikačních problémů se využívají ztrátové funkce s unilaterálním průběhem, což znamená, že pracuje s hodnotou vyjadřující rozdíl od požadovaného výsledku pouze v jednom směru [14].



Obrázek 9. Příklad průběhů unilaterálních ztrátových funkcí [14]

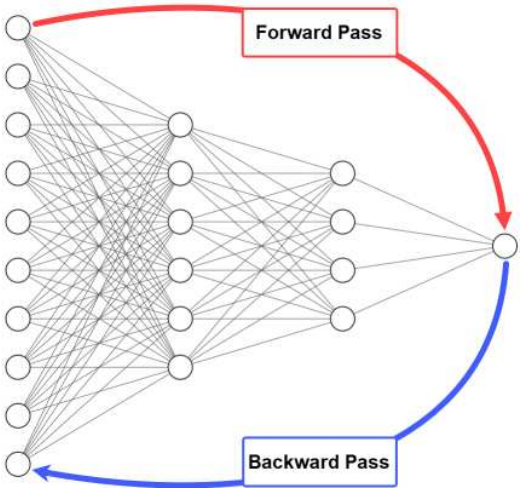
Jednou z nejvyužívanějších ztrátových funkcí pro klasifikaci je křížová entropie (cross-entropy loss). Tato funkce pracuje s pravděpodobnostmi, jak moc vstup odpovídá každé z jednotlivých kategorických tříd. Výsledná hodnota je potom součet rozdílů jednotlivých pravděpodobností oproti reálným třídám vstupních dat. Křížová entropie je mimo jiné velmi intuitivní oproti jiným klasifikačním funkcím právě díky vyjádření pravděpodobností jednotlivých tříd. Například funkce Hinge loss vyjadřuje pouze rozpětí mezi reálnou a predikovanou hodnotou [14], [16].

1.2.4 Backpropagation

Velmi důležitým algoritmem při trénování neuronových sítí je Backpropagation. Tento algoritmus je klíčový pro úpravu vah jednotlivých neuronových spojení. Bez efektivního využití Backpropagation bychom nebyli schopni natrénovat hluboké neuronové sítě s takovým počtem skrytých vrstev, s jakými se v dnešní době setkáváme. Backpropagation je označován jako jádro oblasti hlubokého učení [1].

Backpropagation můžeme rozdělit na 2 části a to na forward-pass a backward-pass. Forward-pass, také označován jako Propagation, je v podstatě proces, při kterém data na vstupu projdou každou z vrstev sítě, čímž se utvoří predikce, ze které je následně vyhodnocena ztrátová hodnota. V počáteční fázi trénování je tato predikce velmi nepřesná, jelikož na začátku jsou všechny váhy nastaveny náhodně. Rozsah hodnot, kterých můžou váhy nabýt při inicializaci, se může lišit dle typu inicializace [1].

Backward-pass je proces, při kterém se vypočítá gradient ztrátové funkce na poslední vrstvě neuronové sítě a následně se za použití tohoto gradientu upravují váhy jednotlivých spojení. Tento proces začíná od výstupní vrstvy a postupuje směrem ke vstupní vrstvě a postupně aplikuje řetízkové pravidlo [1], [17].



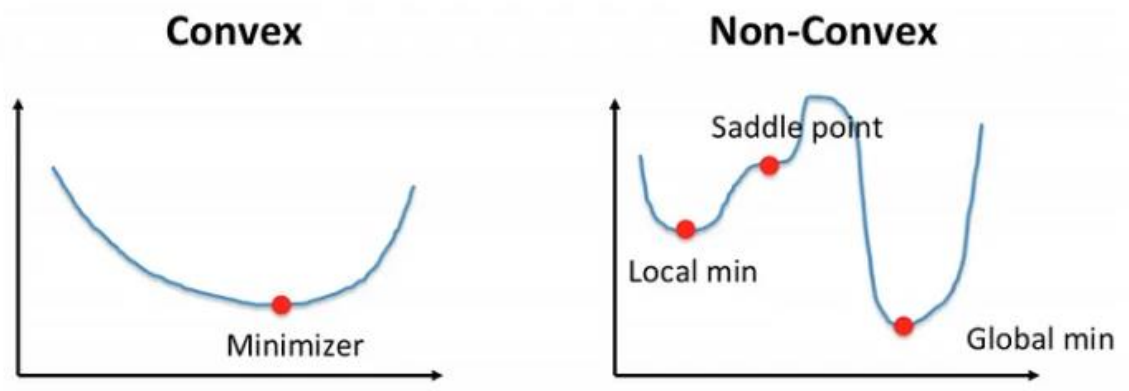
Obrázek 10. Backpropagation [17]

1.2.5 Optimalizátory

Optimalizace je nejdůležitějším aspektem v oblasti hlubokého učení a umělých neuronových sítí. Je to proces, při kterém se přiřazují váhy a biasy tak, aby síť dosahovala požadovaných predikcí. Optimalizace je považována za velice složitou a s rozsahem a hloubkou dnešních umělých neuronových sítí je nemožné po vyhodnocení výstupu tyto parametry manuálně upravovat. Z toho důvodu potřebujeme vytvořit algoritmus, který optimalizaci provede a umožní úpravu vah a biasů, čímž model získá větší přesnost. Vzhledem k tomu, že nám přesnost modelu určuje ztrátová funkce, jejíž parametry jsou váhy a biasy, můžeme zvýšit přesnost modelu přiblížením se k minimu této funkce [1], [5].

V oblasti strojového učení se v minulosti obcházela nutnost komplexní optimalizace tím, že se model specificky designoval tak, aby funkce, která se optimalizovala, byla konvexní. V takovém případě je lokální minimum automaticky i globální minimum a můžeme tedy aplikovat metody pro nalezení lokálního minima [1], [5].

Z hlediska optimalizace hlubokých neuronových sítí tyto problémy nejsou prakticky nikdy konvexní. Tyto funkce obsahují obrovské množství lokálních minim. Proto se u hlubokých neuronových sítí nesnažíme nalézt globální minimum, ale oblast, ze které dostaneme malou ztrátovou hodnotu. Tato oblast ve výsledku není ani lokální minimum, ale v praxi je dostačující. Z toho důvodu můžeme přistupovat i k problémům hlubokých neuronových sítí jako ke konvexním [1], [5].



Obrázek 11. Příklady konvexního a nekonvexního problému [18]

1.2.5.1 Gradientní sestup

Pro zvýšení přesnosti modelu je potřeba přiblížit se minimu ztrátové funkce. Jedním typem metod, které řeší tento problém, jsou metody využívající gradientního sestupu. Tyto metody jsou iterativní optimalizační algoritmy. Váhy a biasy jsou proměnnými ztrátové funkce. Vezmeme-li tyto parametry jako souřadnice v prostoru průběhu ztrátové funkce, naším cílem je nalézt bod, který je co možná nejníže, tedy má nejmenší ztrátovou hodnotu. Naneštěstí neznáme průběh této funkce, tudíž nelze automaticky nastavit parametry tak, abychom docílili nejnižší ztrátové hodnoty. Pro přiblížení se k minimu se tedy využívá gradient ztrátové funkce, který se získá pomocí parciálních derivací, které se vypočítají pro každou dimenzi. Gradientní vektor v podstatě ukazuje směrem, kde je funkce nejstrmější, respektive kde nejvíce klesá [1].

1.2.5.2 Stochastický gradientní sestup

U základního typu gradientního sestupu je ztráta počítána jako suma ztrátových hodnot na všech trénovacích datech v datasetu. Tento postup je velmi pomalý a neefektivní, obzvláště pro velké datasety, jelikož pro každou aktualizaci vah musí sítí projít všechna trénovací data [1].

Mnohem méně výpočetně náročná metoda je stochastický gradientní sestup. Tato metoda je v dnešní době mnohem častěji využívána než základní typ. Namísto vypočítávání gradientu a aktualizování vah na celém datasetu najednou tato metoda v jeden moment pracuje jen s malými částmi datasetu. Tyto části se označují jako mini-batches a kromě menší výpočetní náročnosti umožňují udělat více kroků po gradientech směrem k minimu [1].

U stochastického gradientního sestupu se často implementuje takzvané momentum, které určuje, jak velké aplikujeme kroky směřující po gradientu. Čím déle ukazuje gradientní vektor stejným směrem, tím se momentum zvyšuje. Změní-li se směr, momentum klesne [1].

1.2.5.3 Adaptivní optimalizační metody

Velmi využívanými jsou adaptivní optimalizační metody, které mimo jiné cílí na zmenšení počtu epoch (iterací skrze dataset) pro rychlejší trénování nebo na lepší uzpůsobení sítě pro větší počet hyperparametrů. Jednou z těchto metod je ADAM (Adaptive Moment Estimation). Mimo jiné je u této metody výhodou, že není třeba manuálně nastavovat hodnotu míry učení (learning rate) [19].

1.3 Konvoluční neuronové sítě

Konvoluční neuronové sítě využívají takzvané konvoluční vrstvy. Ve standartních neuronových sítích jsou vrstvy napojeny na sebe tak, že každý jednotlivý neuron je napojen na všechny neurony ve vrstvě předchozí, a tudíž i na každý neuron ve vrstvě následující. Tyto vrstvy se označují jako plně propojené vrstvy (fully connected layers). V konvolučních neuronových sítích se plně propojené vrstvy využívají většinou až na konci sítě, nicméně v závislosti na konkrétní architektuře se využití plně propojených vrstev liší. Namísto toho konvoluční sítě nahrazují tyto vrstvy za vrstvy konvoluční. Jako konvoluční se označuje síť, která má alespoň jednu konvoluční vrstvu [1].

Konvoluční vrstvy aplikují rozdílné sety filtrů, kterých jsou typicky stovky nebo tisíce. Vzhledem k tomu, jak jsou navržené plně propojené vrstvy spolu každý neuron v jednotlivé vrstvě interaguje s každým neuronem ve vrstvě následující. Oproti tomu mají konvoluční vrstvy takzvanou sparse (řídkou) interakci. Je tomu tak proto, že výstup z filtrů v konvoluční vrstvě může být menší než velikost dat na vstupu. Například při zpracovávání obrázku filtr pokrývá větší plochu, tedy větší počet pixelů než u standartních neuronových sítí, kde by jeden vstupní neuron odpovídal jednomu pixelu. Tímto dokážeme docílit výrazně menší paměťové náročnosti, jelikož nemusíme ukládat tolik parametrů, a zároveň dosáhneme výpočetně efektivnějšího modelu, protože nemusíme provádět tolik operací [5].

Další výhodou je sdílení parametrů (Parameter sharing). Tradiční modely neuronových sítí využijí při jednom úkonu každou z vah, kterými jsou v jednotlivých vrstvách propojeny neurony, přesně jednou a dále se už nevyužijí. Sdílení parametrů referuje na to, když se jeden parametr, respektive váha, využije při jednom úkonu vícekrát. Konvoluční neuronové sítě jsou tohoto schopny, jelikož stejný filtr konkrétní vrstvy, tedy stejný soubor natrénovaných vah, se použije na celou plochu předchozí vrstvy. Díky tomu velikost vstupních dat neovlivňuje počet vah ve vstupní vrstvě, jak by tomu bylo při využití plně propojené vrstvy [5].

Například v kontextu klasifikace obrázku mohou konvoluční neuronové sítě pomocí filtrů v úvodní vrstvě detekovat hrany nebo další prostá data. V následujících vrstvách pak za použití těchto hran jednoduché tvary. A v hlubších vrstvách za použití tvarů celé objekty [1].

1.3.1 Konvoluce

Klíčovým prvkem konvoluce je konvoluční filtr, často taky označován jako kernel. Rolí filtru je z jednotlivých hodnot v poli získávat novou hodnotu, která bere v potaz i okolní hodnoty. To, kolik okolních prvků pole má vliv na získanou hodnotu, záleží na velikosti filtru. Filtry, které se využívají u konvolučních sítí, mají většinou stejnou výšku i šířku, jelikož spousta knihoven využívá metody, které operují nejlépe na čtvercových polích. Jak výška, tak šířka musí být lichá, jelikož tato matice musí vždy mít středový prvek. To znamená, že se můžeme setkat s maticemi o velikosti 3x3 nebo 9x9, ale nemůžeme se setkat s rozměry filtru 4x4, jelikož středem matice by byly souřadnice x = 1,5. y = 1,5 což neukazuje na konkrétní prvek [1], [5].

Obvyklými vstupními daty pro modely, které využívají konvoluční vrstvy, jsou multidimenzionální pole například obrázky. Obrázky mají 3 dimenze, a to šířku, výšku a barevné kanály. Filtr se po vstupním poli pohybuje a vytváří pole nové obsahující hodnoty prvků vstupního pole, které jsou ovlivněny sousedními hodnotami. To, jakým způsobem jsou ovlivněny, určují váhy, které každý filtr obsahuje. Jelikož některé prvky vstupního pole sousední hodnoty nemají, aplikuje se takzvaný padding. V podstatě se v poli vytvoří okraj obsahující předem určené hodnoty. Populární je například zero-padding, kde okraj obsahuje nuly. Spousta knihoven hlubokého učení nevyužívá vyloženě konvoluci, ale zjednodušenou křížovou korelaci (cross-correlation), která je podobná, nicméně tyto termíny zaměňuje [1], [5].

Obrázek 12. Příklad konvoluce [1]

1.3.2 Typy vrstev v konvolučních neuronových sítích

Přesto, že je konvoluční vrstva klíčovým stavebním blokem u konvolučních neuronových sítí, to není zdaleka jediný typ vrstvy, který by tvořil celkovou architekturu. Konvoluční síť se obvykle skládá z více různých vrstev, které jsou napojeny za sebou tak, aby byl model co nejefektivnější. Zde jsou příklady pár široce využívaných typů vrstev [1].

1.3.2.1 Konvoluční vrstva

V konvoluční vrstvě probíhá konvoluce. Filtry s natrénovánými vahami jsou, co se týče plochy, relativně malé, ale jsou aplikovány skrze celou hloubku vstupních dat. V oblasti klasifikace obrázků je touto hloubkou v počáteční vrstvě počet kanálů. Pracujeme-li tedy s RGB obrázky, hloubka je 3, respektive jedna za každý barevný kanál. Pro vrstvy hlouběji v síti je hloubka počet aktivačních map, které jsou výstupem z předchozí vrstvy. Každá aktivační mapa je výstupem jednotlivých filtrů [1].

Důležitým hyper-parametrem u konvolučních vrstev je stride. Rovná-li se stride 1, filtr se po vstupním poli pohybuje postupně po každém pixelu, dokud nenavštíví všechny prvky tohoto pole. Je-li stride nastaven na hodnotu 2, můžeme si představit, že filtr se po vstupním poli pohybuje ob jedno, respektive každý krok je dva pixely dlouhý. Čím větší použijeme stride, tím méně se budou jednotlivé data překrývat, a tím menší bude výstup [1].

1.3.2.2 Aktivační vrstva

Aktivační vrstva je vrstva, kde se aplikuje aktivační funkce, specificky nelineární funkce, jako je například ReLU. Aktivační vrstva technicky není samostatná vrstvy, protože se v ní netrénují žádné váhy. Často se z diagramu architektury konvoluční sítě vynechává a automaticky se předpokládá, že aktivační vrstva následuje za vrstvou konvoluční [1].

1.3.2.3 Pooling vrstva

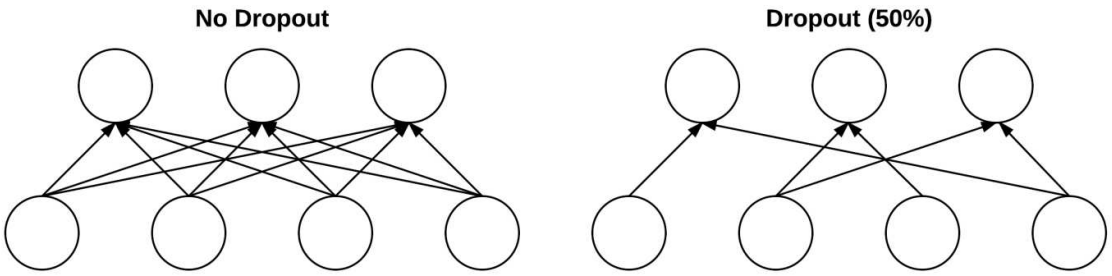
Existuje způsob jak, co se prostorové dimenze týče, zmenšit pole za pomocí vysoké hodnoty stride v konvoluční vrstvě. Nicméně se často za tímto účelem využívá specializovaný typ vrstvy, kterému se říká pooling vrstva. Pooling vrstva nezabraňuje překrývání, ani nepřeskakuje určité prvky pole, namísto toho vytváří něco jako shrnutí důležitých prvků vstupního pole. Například max-pooling rozdělí vstup do menších oblastí a následně z nich ponechá jen několik největších hodnot [1], [5].

1.3.2.4 Batch Normalization

Vrstva Batch normalization slouží k tomu, aby byly vstupní hodnoty normalizovány, než budou přivedeny na vstup další vrstvy. Některé hodnoty jsou před vstupem do další vrstvy exponenciálně vyšší než jiné, proto je těžké určit hodnotu míry učení pro trénování parametrů. Normalizace zaručuje, že tyto hodnoty budou stejnoměrně rozprostřené po určitém rozsahu a budou vystředěné od nuly. Využití Batch normalization se ukázalo jako velmi efektivní v redukování počtu epoch, které jsou potřeba k trénování neuronové sítě. Zároveň vrstva Batch normalization přidává do sítě šum, čímž generalizuje model. To znamená, že model bude přesnější na datech, na kterých není trénovaný, a nedojde k takovému overfittingu [1], [5].

1.3.2.5 Dropout

Stejně jako částečně u Batch normalization i Dropout slouží ke generalizaci. U Batch normalization to ale není hlavní důvod, proč se využívá. Dropout je forma regularizace, která zvyšuje testovací přesnost na úkor tréninkové přesnosti, čímž zabraňuje overfittingu. Droupout v podstatě odstraňuje některé ze spojení neuronů mezi plně propojenými vrstvami. Odstranění některých spojení probíhá v trénovacím procesu a způsobuje, že je zde více redundantních neuronů způsobujících aktivaci. Míra odpojení mezi vrstvami je ovládána pravděpodobností p, která určuje šanci, jestli se spojení odstraní. p = 0.5 určuje 50% šanci že je spojení odstraněno [1].



Obrázek 13. Dropout [1]

1.3.3 Příklady architektur

Důležitým faktorem, na který je třeba při vývoji konvoluční sítě dbát, je architektura modelu. Výběr architektury je kritický, chceme-li dosáhnout vysoké přesnosti u specifických aplikací. V základu se architektura konvoluční neuronové sítě skládá primárně z těchto vrstev: konvoluční vrstva (CONV), pooling vrstva (POOL), aktivační vrstva (ACT) a plně propojená vrstva (FC). Poskládáním těchto vrstev za sebe získáme architekturu modelu konvoluční neuronové sítě [1], [20], [51].

1.3.3.1 ShallowNet

Velmi základní architekturou je ShallowNet. Tato architektura obsahuje pouze malé množství vrstev, a to vstupní vrstvu následovanou konvoluční vrstvou, dále aktivační vrstvu a plně propojenou vrstvu. Diagramem by se dala vyjádřit jako: INPUT => CONV => ACT => FC. Počet vrstev se může lišit u různých ShallowNet architektur, zároveň některé architektury využívají pooling vrstvy, obecně ale platí, že ShallowNet je velmi základní architektura, která se využívá na velmi jednoduché aplikace [1].

1.3.3.2 AlexNet

AlexNet už je komplexnější architektura, která využívá i vrstvy jako Batch Normalization (BN) a Dropout. AlexNet několikrát za sebou opakuje kombinace vrstev CONC, ACT, BN, POOL, DROPOUT a následně podobně z FC namísto CONV [19].

AlexNet je známý tím, že využívá relativně velké filtry o rozměru 11x11 [20].

1.3.3.3 VGGNet

Architektura nazývána jako VGG (Visual Geometry Group, Skupina vizuální geometrie) byla vytvořena za účelem efektivního rozpoznávání obrázků. Oproti AlexNet má o 19 vrstev více. Co se filtrů týče, VGG volí přístup více malých filtrů. Napříč celým modelem mají filtry velikost 3x3. Další charakteristikou VGG je, že za sebe zařazuje více konvolučních vrstev před tím, než aplikuje pooling vrstvu. Díky tomu se může naučit rozpoznávat komplexnější rysy před zmenšením prostorové velikosti pole [1], [20], [51].

2 SEGMENTACE

Jedním z nejtěžších úkolů v oblasti počítačového vidění je segmentace obrázků. Segmentace obrázku je proces, při kterém se rozdělí digitální obrázek na podkategorie, kterým se říká segmenty. Díky tomu se sníží komplexita obrázku a je možné obrázek dále analyzovat a procesovat jednodušeji. Segmentace je v podstatě přiřazování kategorií jednotlivým pixelů, čímž je možné identifikovat jednotlivé objekty nebo různé elementy v obrázku [21], [22], [51].

2.1 Typy segmentace

2.1.1 Sémantická segmentace

Sémantická segmentace je typ segmentace, který přiřazuje pixely do takzvaných sémantických tříd. V takto segmentovaném obrázku má každý pixel přiřazenou třídu. Sémantický model dokáže segmentovat i objekty, u kterých nezná kategorii. Například tam, kde by jiný model segmentoval pouze objekty nebo elementy, u kterých zná kategorii, jako je například kůň, pes nebo motorka, sémantický model seskupí do třídy i pixely objektu bez jakéhokoliv kontextu k tomu, jestli se podobá nějaké konkrétní kategorii [21], [22], [50].

2.1.2 Segmentace instancí

Segmentace instancí je založena na vyhodnocování jednotlivých výskytů určité třídy v obrázku. Tento model nerozřazuje každý pixel do příslušné kategorie, ale zaměřuje se na rozpoznávání jednotlivých instancí konkrétní třídy. Respektive nachází-li se v obrázku kolona aut, model sémantické segmentace vyhodnotí každé jednotlivé auto jako samostatnou třídu (například Auto 1, Auto 2), nikoliv jako plochu náležící kategorii auto [22].

2.1.3 Panoptická segmentace

Panoptická segmentace se často označuje jako kombinace sémantické a instanční segmentace. Tento typ vyhodnocuje jak identitu každého objektu, tak jej odděluje od objektů náležící stejné třídě. Tato metoda se využívá v široké škále programů, které potřebují velké množství informací, například samořídící auta [22].

2.2 Metody segmentace

Existuje více různých typů segmentace, které využívají odlišných metod.

2.2.1 Tradiční metody

Přestože metody využívající hluboké učení, co se týče například přesnosti, výrazně překonaly tradiční způsoby pro segmentaci obrázků, tradiční metody zůstávají v některých ohledech užitečným nástrojem. Hlavní charakteristikou tradičních technik je vyhodnocení informací z jednotlivých pixelů, tyto informace můžou zahrnovat například jas, sytost nebo kontrasty vůči okolí, či rysy nebo texturu. Díky těmto informacím je možné shlukovat pixely do jednotlivých segmentů. Oproti metodám využívajících hluboké učení mají tradiční metody výhodu v tom, že jsou výrazně méně nákladné, co se týče výpočetní náročnosti. Nevýhodou je náchylnost na šum a jiné rušivé prvky, které se objevují v reálných podmínkách [23]. Zde je pár příkladů tradičních metod:

2.2.1.1 Detekce hran

Jednou z tradičních metod segmentace obrázků je detekce hran. Metody založené na detekci hran identifikují hranice objektů v obrázku tím, že detekují rozdíly jasu, kontrastu, textury, barvy a variací saturace. Díky tomu můžou reprezentovat hranice objektu v obrázku pomocí řetězce hran [21], [23].

2.2.1.2 Prahování

Metoda prahování (Thresholding) je jednou z nejjednodušších metod pro segmentaci. Tato metoda spočívá v klasifikaci pixelů většinou v černobílém spektru podle toho, zda je jejich hodnota nad určenou prahovou úrovní. Tato hodnota může být konstanta při segmentaci obrázku s malým šumem, nebo může být dynamická [21], [23].

2.2.1.3 Segmentace podle oblastí

Segmentace podle oblastí (Region-based segmentation) je metoda, při které se používají počáteční body. Tato metoda segmentace rozděluje obrázek na oblasti, které mají podobnou charakteristiku. Algoritmus začíná v počátečním bodě a postupně expanduje a vytváří oblasti [21], [23].

2.2.2 Moderní metody

Hlavním prvkem moderních metod je využití hlubokého učení. V dnešní době metody založené na hlubokém učení výrazně překonávají tradiční metody. Úspěch těchto metod vychází z dostupnosti velkého množství dat a výpočetních prostředků [24]. Mezi důležité segmentační metody využívající hluboké učení patří:

2.2.2.1 Plně konvoluční sítě

Standartní konvoluční neuronové sítě obsahují plně propojené vrstvy. Výstup z takovýchto sítí má fixní velikost a je vhodný pro klasifikační a regresní problémy. Namísto toho plně konvoluční sítě provádí klasifikaci na úrovni jednotlivých pixelů, tudíž je vyžadováno, aby segmentační model mohl vzít jako vstup obrázek o libovolné velikosti a výstup měl totožnou velikost. Plně konvoluční sítě využívají techniku, kdy nahrazují plně propojené vrstvy konvolučními. Díky tomu můžou tyto sítě vytvářet výstup, který jsou přiřazené kategorie ke každému pixelu v obrázku [24], [25].

2.2.2.2 U-Nets

U-Nets jsou typem metod pro segmentaci, které modifikují architekturu plně konvolučních sítí za účelem redukování ztrát důležitých dat při procesu konvoluce. U-Nets využívají takzvané skip connections. Účelem těchto spojení je přeskočit některé konvoluční vrstvy a díky tomu zachovat nejen globální kontext, ale i ten lokální. Díky tomu dokážou U-Nets mnohem přesněji segmentovat malé nebo detailní objekty. U-Nets byly vytvořeny pro segmentace v oblasti medicíny a dostaly svůj název podle tvaru, jakým se zobrazuje architektura [23], [26].

2.2.2.3 RCNNs

RCNNs jsou konvoluční neuronové sítě, které segmentují podle oblasti. Tyto metody využívají segmentaci za použití rozpoznávání objektů. Specifičtěji RCNNs provádějí sémantickou segmentaci. První se provádí detekce objektů a potom se pro každý z objektů klasifikuje oblast [22].

3 DETEKCE

Jedním ze základních odvětví v oblasti umělé inteligence a počítačového vidění je detekce objektu. Na detekci objektů se staví mnoho dalších úkolů počítačového vidění, které jsou mnohem komplexnější. Mezi oblasti, kde se využívá detekce objektů, patří například automatické řízení vozidel, analýza v medicíně, bezpečnostní kamerové systémy, kontrola výrobků a další [27], [28].

Chceme-li získat kompletnější porozumění obrázku, nestačí se zaměřovat pouze na klasifikaci objektů, ale zároveň musíme zvládnout konkrétně určit umístnění objektů v obrázku. Tento úkol se označuje jako detekce objektů. Tradiční detekci objektů můžeme rozdělit na tři části, a to výběr oblasti, výtah rysů a vlastností oblasti a klasifikace [27], [28].

3.1 Výzvy u tradičních metod detekce

Výběr oblasti je komplikovaný, jelikož se na různých pozicích může nacházet různé množství objektů, stejně tak různé objekty můžou mít různé velikosti a poměry stran. Přirozeným postupem je aplikovat na obrázek pohyblivá okénka o různých velikostí a poměrů stran a pokrýt tak všechny možné pozice, kde by se objekt mohl nacházet. Tato technika je výpočetně velmi neefektivní, jelikož díky velkému množství možných kombinací vyprodukuje spoustu redundantních případů. Zároveň pokud bychom chtěli omezit počet druhů okének na fixní množství, nemuseli bychom pokrýt všechny důležité oblasti [27], [28].

Chceme-li za použití tradičních metod rozpoznat objekt, musíme nejdříve extrahovat rysy v dané oblasti. K tomu lze využít metody jako například histograms of oriented gradients (HOG) [29], nicméně díky široké diverzitě v tom, jak konkrétní objekt může vypadat v kombinaci s faktory jako světelné podmínky, šum, nebo různá pozadí, je velmi obtížné podobné algoritmy manuálně na designovat [27].

Samotný algoritmus pro výtah rysů ovšem nestačí. Pro kompletní reprezentaci obrázku je třeba objekty dále klasifikovat. K tomu se tradičně využívají metody jako například supported vector machine (SVM) [30], [27].

Tradiční metody mají obvykle špatnou přesnost pouze v jednoduchém a velmi specifickém prostředí [28].

3.2 Moderní metody detekce

Díky rychlému rozvoji v oblasti hlubokého učení máme k dispozicí více a více efektivních nástrojů, které jsou schopny učit se složitější rysy nebo lépe zpracovávat sémantický kontext. Díky tomu můžeme adresovat a lépe řešit problémy, které byly charakteristické pro tradiční metody detekce objektů. Moderní metody mají odlišnou architekturu a chovají se jinak v kontextu strategií trénování a optimalizačních funkcí [27], [28].

3.2.1 Formy detekce

Existuje několik typů detekce, které se odlišují svým zaměřením či využitím. Některé typy mají široké využití a jiné jsou specializované pro konkrétní účely.

3.2.1.1 Generická detekce

Mezi typy detekce patří generická detekce. Jako generická detekce objektů se označuje forma detekce využívaná pro obrázky nebo videa, která však u detekce ve videu vyhodnocuje jednotlivé snímky bez návaznosti na ostatních snímcích, respektive vyhodnocuje snímky ve videu jako samostatné obrázky. Tato forma detekce je velmi multifunkční a relativně jednoduše aplikovatelná na velké množství odlišných úkolů [27].

3.2.1.2 Video detekce

Video detekce objektů má speciální využití pro detekci ve videích. Tato forma se zaměřuje na zpracování informace, která vychází ze změn mezi jednotlivými snímky. Zmíněné informace můžou hrát důležitou roli v pochopení chování a vlastností různých objektů [27].

3.2.1.3 Salientní detekce

Salientní detekce objektů se zaměřuje na vizuální význačnost objektů v obrázku. Na rozdíl od generické detekce objektů salientní detekce není založena na rozpoznání jednotlivých tříd objektů či kategorizování oblastí. Tato forma detekce identifikuje nejvíce vizuálně důležité objekty v určité oblasti. Hlavní roli hraje kontrast různých prvků v obrázku, a to bez ohledu na sémantický kontext. Salientní detekce se používá například v programech pro úpravy obrázků [27].

3.2.2 Dvoufázová detekce

Jedním ze způsobů, kterým se řeší generická detekce, jsou dvoufázové detekční metody. Celý proces detekce je v této metodě rozdělen na dvě části. První částí je takzvaná Regional proposal. V této fázi algoritmus vybírá oblasti, které slouží jako potenciální ohraničující rámečky. Regional proposal cílí na efektivní identifikaci oblastí v obrázku, u kterých je velká pravděpodobnost, že obsahují objekty [27].

Druhou fází je samotná klasifikace objektu. V této fázi jsou v oblasti vybrané v Regional proposal části předány do algoritmu zodpovědného za klasifikaci. Klasifikátor dále vyhodnocuje, zdali se v dané oblasti nachází objekt, a případně přiřazuje kategorie tříd detekovanému objektu [27].

3.2.2.1 RCNN

První model, který dvoufázovou detekci implementoval s využitím konvolučních sítí, byl R-CNN. Tento model nejprve provede selektivní výběr a extrahuje okolo 2000 oblastí, kde by se mohl nacházet objekt. Dále je každá z oblastí upravena na stejnou velikost a použije se jako vstup pro SVM klasifikátor [30]. Posledním krokem je použití regresního modelu pro vyhodnocení ohraničujících rámečků [27], [28].

3.2.2.2 SPP-Net

Nedostatkem RCNN modelů bylo, že před klasifikací musely být upraveny na stejnou velikost. Toto zapříčiňovalo, že se objekty či jejich části mohly nacházet v části, která byly ořezána z původní oblasti.

Model SPP-Net (Spatial Pyramid Pooling) řešil tento nedostatek tím, že implementoval teorii zvanou SPM (Spatial Pyramid Matching). Tento algoritmus extrahoval prvky jednotlivých oblastí poté, co originální obrázek prošel konvoluční vrstvou. Dále byla za poslední konvoluční vrstvu přidána vrstva SPP, jejíž výstup měl fixní rozměry, čímž obešel nutnost mít fixní velikost oblastí [27], [28].

3.2.2.3 Fast RCNN

Fast RCNN je vylepšená verze původní R-CNN, která nahradila SVM za klasifikátor ze softmax funkcí, dále vychází z SPP vrstvy a využívá takzvaný region of interest pooling layer, kterým nahrazuje poslední pooling vrstvu. Nakonec je síť zakončena dvěma plně propojenými vrstvami [27], [28].

3.2.2.4 Faster RCNN

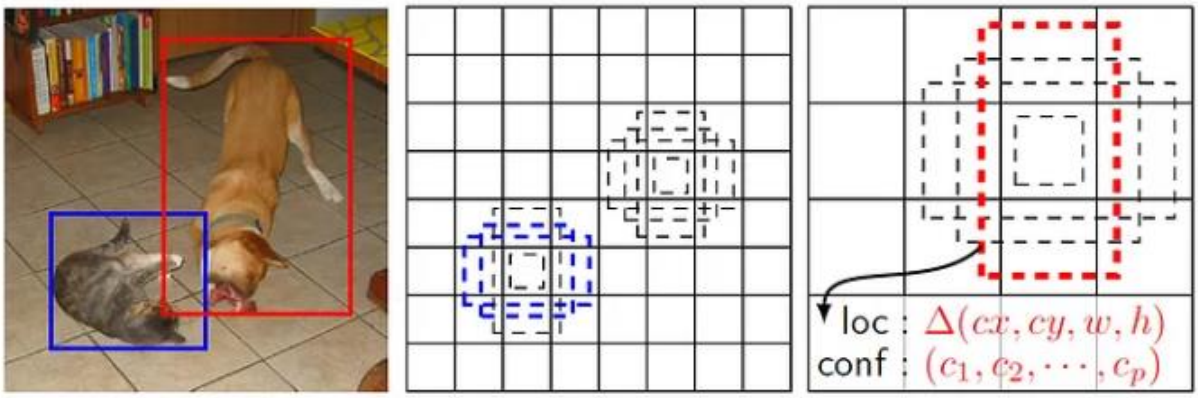
Nejzásadnějším aspektem, který ovlivňuje efektivitu dvoufázové detekce, je Region proposal fáze. Faster RCNN pro hledání oblastí, které potenciálně mohou obsahovat objekty, používá RPN (Region Proposal Network). Model je rozdělen na dvě části, první část je plně konvoluční neuronová síť, která se využívá pro generování potenciální oblasti. Druhá část je Fast RCNN detekční algoritmus. Tyto dvě části sdílí některé konvoluční vrstvy, čímž zefektivňují jejich využití v modelu. Tento model dosahuje skvělé přesnosti, ale pořád není dostatečně rychlý pro detekci v reálném čase [27], [28].

3.2.3 Jednofázová Detekce

Zatímco dvoufázové metody rozdělují proces detekce na část výběru oblastí a část klasifikace a vyhodnocování ohraničujících rámečků, jednofázová detekce spojuje tyto dvě části do jedné, respektive vyhodnotí ohraničující rámečky bez použití region proposal. Tyto metody obvykle rozdělují obrázek do mřížek, které pomáhají s lokalizací obrázku. Jednofázové metody detekce obvykle nedosahují takové přesnosti jako u dvoufázové detekce, namísto toho mají vetší rychlost, díky čemuž se více hodí pro detekci v reálném čase [30], [51].

3.2.3.1 SSD

Jako u ostatních jednofázových metod, detekce SSD (Single Shot Detector) stačí pouze jeden průchod neuronovou sítí pro vytvoření ohraničujících rámečků okolo objektů. SSD představilo způsob kde se nevyužívalo pouze fixní mřížky. Tato metoda používá rámečky různých poměrů stran a velikostí, které jsou rozpoloženy po celé ploše obrázku. Pro zpracování predikcí různých rozměrů SSD spojuje výstupy z různých map [27], [32], [51].

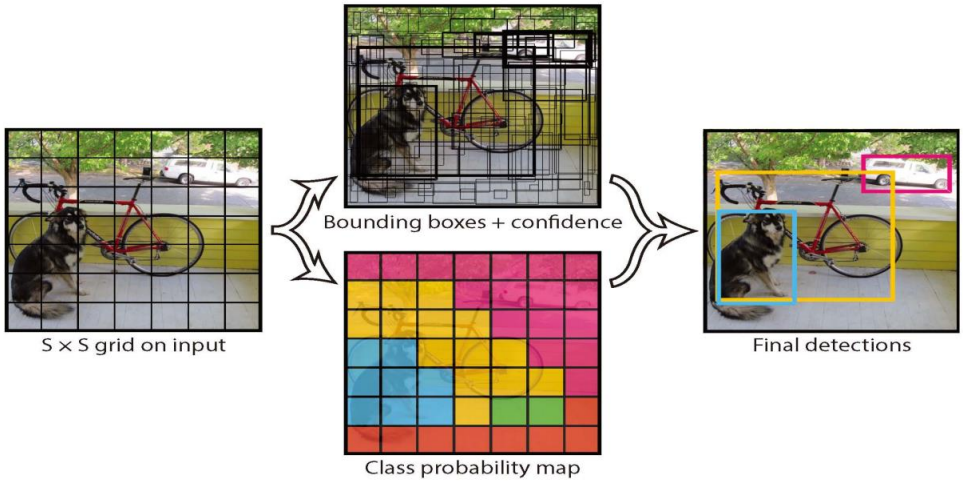


Obrázek 14. SSD [32]

3.2.3.2 YOLO

YOLO byl prakticky prvním způsobem, jak dosáhnout detekce objektů v reálném čase. YOLO znamená You only look once (Podíváš se jen jednou), čímž poukazuje na skutečnost, že provádí detekci pouze za pomocí jednoho průchodu sítí, čímž se odlišuje od ostatních metod, které buďto musely klasifikovat obrovské množství oblastí, nebo musely využívat techniku dvoufázové detekce. Základní myšlenka této metody spočívá v rozdělení vstupního obrázku na jednotlivé části pomocí mřížky. V každé z buněk mřížky se provádí predikce odhadující třídu objektu uvnitř této buňky. Zároveň se pro každou buňku odhadují ohraničující mřížky [33].

Do rodiny YOLO modelů spadá velké množství různých verzí a variací, které přibývají a zdokonalují se s postupem času. Hlouběji se na rozdíly jednotlivých verzí zaměřují tyto články [28], [33].

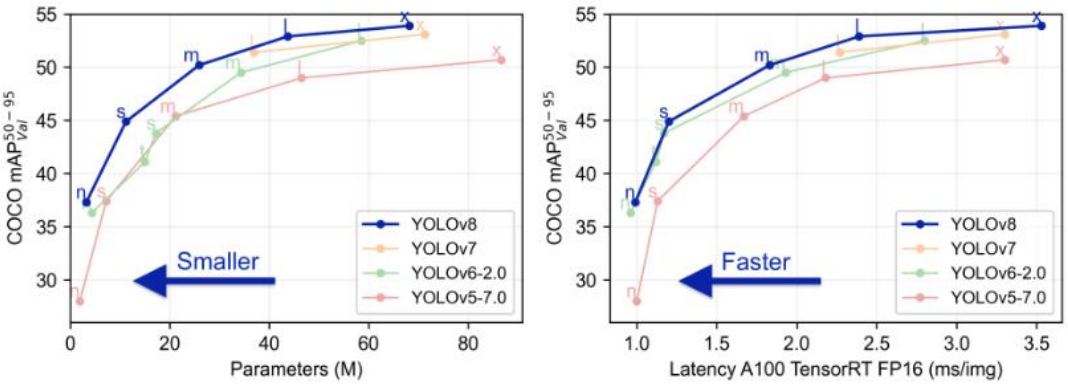


Obrázek 15. Základní myšlenka YOLO [32]

3.2.3.3 YOLOv8

YOLOv8 je nejnovější verze, kterou vydala společnost Ultralytics, která dříve vydala YOLOv5. YOLOv8 je state-of-the-art (SOTA) model pro detekci a další úkoly v reálném čase. Tento model staví na úspěchu předchozích verzí a v základu má podobnou strukturu jako YOLOv5. Novinkou, kterou YOLOv8 přidává, je model pro sémantickou segmentaci YOLOv8-seg, který zvládá segmentaci velmi přesně a udržuje si přitom rychlost [33], [44].

Velmi užitečnou možností, kterou YOLO nabízí, jsou modely různých velikostí, respektive počtů parametrů. YOLOv8 nabízí velikosti nano, small, medium, large a extra large. Tyto velikosti odpovídají písmenům na konci názvu modelu v tabulce č 1. Oproti předchozím verzím zvládají různé verze YOLOv8 být jak přesnější v poměru k rychlosti, tak v poměru k počtu parametrů [45].



Obrázek 16. Porovnání rychlosti a počtu parametrů různých YOLO verzí [45]

Tabulka 1. Porovnání modelů s různou velikostí [45]

Model	Velikost (pixely)	mAPval 50-95	Rychlost CPU ONNX (ms)	Rychlost A100 TensorRT (ms)	Parametrů (M)	FLOPs (B)
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

4 DETEKCE A SEGMENTACE LIDSKÉHO OBLIČEJE

Detekce a segmentace jsou aktivními oblastmi v kontextu získávání informací z lidského obličeje. Slouží k předzpracování (preprocessing) obrázků, na kterých se obličej nachází. Detekce a segmentace umožňují aplikovat na předzpracovaný obrázek další modely, které mají konkrétnější zaměření a získávají specifické informace [27], [34].

4.1 Detekce obličeje

Detekce obličeje je proces, která zvládá určovat polohu a rozměry lidského obličeje, který se nachází na obrázku, případně na snímku ve videu. Detekce obličeje je počátečním krokem pro spoustu technologií, které pracují s lidským obličejem, jelikož slouží jako důležitý proces předzpracovávání [27], [35].

Mezi odvětví, které využívají detekci obličeje patří například rozpoznání lidského obličeje (face recognition) podrobněji v [36], [37], [38]. Dále pak syntéza obličeje (face synthesis) [39], nebo analýza výrazu ve tváři [41].

4.2 Segmentace obličeje

Segmentace v oblasti lidského obličeje zahrnuje rozpoznání určitých důležitých charakteristik tváře. Jednou z důležitých oblastí segmentace obličeje je například Facial Landmark Extraction (Extrakce prvků obličeje), která zpracovává obličej sémantickou segmentací a přiřazuje kategorie jednotlivým oblastem. Tato technika je v kontextu segmentace obličeje dost multifunkční a dá se využívá pro různě specializované analýzy obličeje, například lze Facial Landmark Extraction využít při určování orientace obličeje, tedy natočení hlavy. Konkrétněji se na další využití a techniky segmentace zaměřují [32], [42], [43].

Segmentace lidského obličeje často nepracuje na stejné úrovni jako detekce obličeje. Technologie pracující v reálném prostředí, které využívají segmentaci obličeje, obvykle využívají první detekci a následně segmentují pouze detekovanou oblast, kde se obličej nachází.

II. PRAKTICKÁ ČÁST

5 VÝBĚR DATASETU

Prvním krokem při vytváření modelu umělé inteligence je tvorba datasetu. Dataset se skládá z jednotlivých dat a přiřazených anotací (label). V oblasti počítačového vidění jsou data obvykle obrázky. Anotace můžou být odlišné v závislosti na specifickém úkolu, na který je model zaměřený [1]. Cílem této práce je vytvořit modely pro detekci osob a segmentaci obličeje, která slouží jako příprava pro další charakterizování. Z toho důvodu je nutné vybrat datasety tak, aby co nejlépe odpovídaly případům užití. Tyto případy jsou převážně vyhodnocování obrazu kamer ve veřejných prostorech.

5.1 Dataset pro detekci hlavy

Pro efektivní segmentaci obličeje je vhodné nejprve obličej detekovat. Jedním z důležitých požadavků je, aby systém zvládal udržovat povědomí i o osobách, kterým není obličej vidět, například když se pohybují směrem od kamery. Z tohoto důvodu byla zvolena detekce hlavy namísto přímo detekce obličeje. Celkový dataset byl vytvořen kombinací několika odlišných dostupných datasetů, které byly dle potřeby upraveny.

5.1.1 PennFudan Pedestrian Dataset

První z použitých datasetů je PennFudan Pedestrian Dataset [46]. Dataset obsahuje obrázky lidí vyskytujících se v ulicích města a poblíž univerzity. Tento dataset obsahuje 170 obrázků a 345 label. Labely tohoto datsetu pro detekci jsou ohraničující rámečky okolo jednotlivých osob. Vzhledem k tomu, že je potřeba detekovat hlavy, byly vytvořeny nové anotace v online nástroji CVAT. Tyto anotace byly následně exportovány do formátu, který odpovídá labelům, které využívá YOLOv8.

Dataset je dostupný na: https://www.kaggle.com/datasets/jiweiliu/pennfudanped

Obrázek 17. Příklad obrázků z PennFundan datasetu

5.1.2 Head Detection (CCTV) - v5

Další obrázky, které jsou obsaženy ve výsledném detekčním datasetu, pochází z Head Detection (CCTV) - v5 [47]. Tento dataset je dostupný na stránce roboflow.com a obsahuje obrázky, které nejlépe odpovídají snímkům, se kterými bude systém ve styku. Dataset obsahuje 1855 obrázků, které mají ohraničující rámečky okolo hlavy, tudíž nebyla potřeba další úprava. Tyto obrázky obsahují osoby, které se z velké části nacházejí v uzavřených prostorech a vyskytují se na bezpečnostních kamerách. Ne všechny obrázky v tomto datasetu jsou zcela unikátní, jelikož pozadí je často stejné a spousta obrázků jsou pouze kopie, na kterých jsou aplikovány různé augmentace. Přesněji je to:

- Rotation mezi -23° a 23°
- Shear mezi -17° a 17°
- Brigthness mezi -44 % a 0 %

Fakt, že jsou data již předaugmentována, není ideální, jelikož při používání YOLO se při trénování udává, jak se mají data augmentovat a preprocesují se automaticky. Zvolí-li se takovýmto způsobem augmentace, má to za následek, že se ta samá data augmentují dvakrát. Data jsou zároveň všechna převedena na rozměry 640x640 a anotace jsou ve formátu YOLOv8, tudíž je není potřeba exportovat.

Dataset je dostupný na: https://universe.roboflow.com/trisha-then/head-detection-cctv

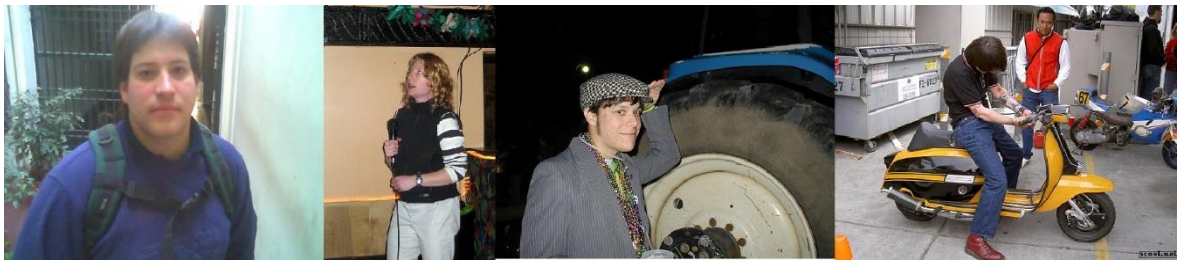
Obrázek 18. Příklad obrázků z Head Detection (CCTV)

5.1.3 Pedestrian Detection Dataset

Třetí dataset, ze kterého bylo čerpáno, je Pedestrian Detection Dataset [48]. Tento dataset obsahuje obrázky pro rozpoznávání osob od objektů, které vypadají jako osoby, například sochy. Obrázky neobsahující lidi nebyly využity, jelikož detekční model rozpoznává pouze jednu třídu. Tento dataset byl zvolen, protože obsahuje obrázky hlavy, které jsou detailnější než u předchozích zdrojů. Zároveň slouží pro dosažení větší generalizace u detekce hlavy

tím, že obsahuje odlišné typy obrázků, které nejsou z konkrétního prostředí. Tyto obrázky musely být manuálně anotovány a exportovány. Ve výsledném detekčním datasetu je použito 526 obrázků z tohoto datasetu, které byly vybrány tak aby byly co nejvíce odlišné a zlepšili tím generalizaci.

Dataset je dostupný na: https://www.kaggle.com/datasets/karthika95/pedestrian-detection



Obrázek 19. Příklad obrázků z Pedestrian Detection Dataset

5.1.4 Rozdělení detekčního datasetu

Dataset byl rozdělen na trénovací a validační data v poměru zhruba 85 % trénovací a 10 % validační tak, aby validační data obsahovaly 10 % z každého zdroje, zároveň validační data neobsahují žádné předem augmentované obrázky. Posledních 5 % jsou testovací data, která jsou pouze z Head Detection (CCTV) - v5 [47], jelikož tato data dobře odpovídají prostředím, ve kterých by mohl být model využit, protože jsou z bezpečnostních kamer, a to se podobá situaci, kdy bude kamera umístěna u vrchní části reklamního panelu, což je zamýšlené využití.

5.2 Dataset pro segmentaci obličeje

Existuje spousta možností, jakým způsobem lze segmentovat lidský obličej. Relativně multifukční metodou je přiřazení kategorie ke každému pixelu v obličeji. Díky tomuto způsobu lze získat z obličeje různé charakteristiky, jako například směřuje-li obličej ke kameře.

5.2.1 Face/Head Segmentation Dataset

Jako dataset pro segmentaci byl zvolen Face/Head Segmentation Dataset Community Edition [49]. Dostupný na: https://store.mut1ny.com/product/face-head-segmentation-dataset-community-edition

Tento dataset obsahuje obličeje lidí rozdílných národností a stáří v různých pózách. Segmentační masky obsahují kategorie pro každý jednotlivý pixel v obličeji. Jednotlivé části

obrázku mohou být rozděleny do 11 kategorií včetně pozadí. Na některé obrázky je aplikována augmentace, převážně rotation, a některé obrázky jsou černobílé.

Z datasetu bylo použito okolo 2500 obrázků a masek. Masky jsou ve formátu obrázků, tudíž je bylo potřeba konvertovat do formátu label, který podporuje YOLOv8, respektive předělat jednotlivé barevné pixely na vektory do textového dokumentu. Vzhledem k tomu, že je využíváno YOLOv8, což je relativně malý model, nebyly konvertovány všechny kategorie, ale jen oči, nos a ústa, aby se docílilo vetší přesnosti u těchto kategorií.

Obrázek 20. Porovnání původního obrázku a masek segmentace

5.2.2 Rozdělení segmentačního datasetu

Stejně jako u detekčního datasetu, jsou data rozdělena zhruba na 90 % (2248 obrázků) trénovacích dat a 10 % (252 obrázků) validačních. U segmentačního datasetu jsou však i některé z validačních dat augmentovaná, jelikož augmentovaná data v tomto datasetu nejsou pouze kopie a neexistuje tedy více upravených obrázků, které by měly stejný originální obrázek.

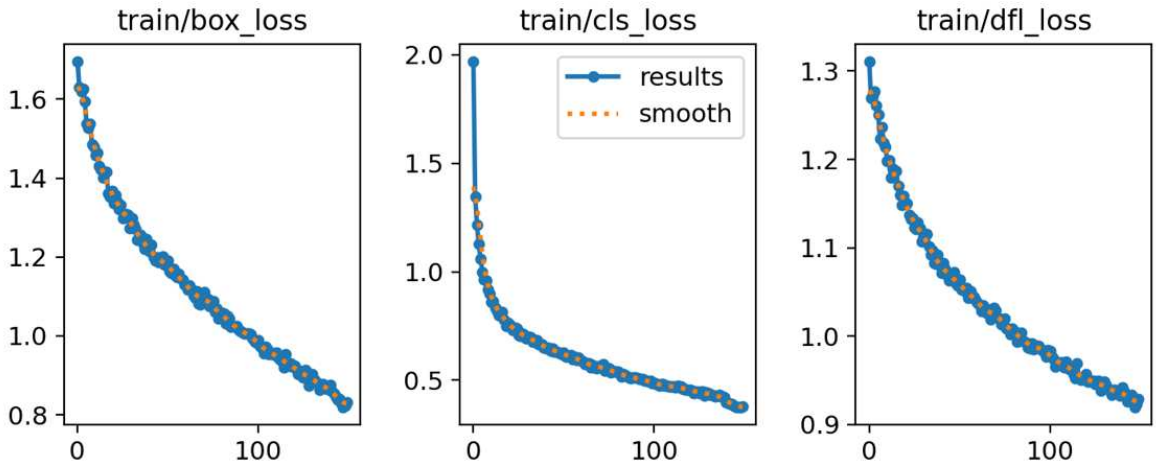
6 TRÉNOVÁNÍ MODELU

Jedním z hlavních požadavků pro navržené modely je, aby dokázaly v reálném čase pracovat na zařízeních, která nejsou sestavena přímo pro účely umělé inteligence. Proto byly pro detekci i segmentaci zvoleny modely YOLOv8 nano, které mají oproti ostatním verzím menší počet parametrů a jsou tedy rychlejší.

6.1 Trénování detekčního modelu

Při trénování YOLOv8 modelu se vyhodnocují 3 typy ztrátových hodnot. První je box_loss, tato hodnota udává, jak moc se modelem vyhodnocený ohraničující rámeček liší od anotace. Dále je zde hodnota cls_loss, která udává, zdali model vyhodnotil třídu objektu správně. Poslední ztrátovou hodnotou je dfl_loss, která adresuje vyhodnocení tříd v závislosti na tom, jak těžké je třídu detekovat, respektive na tom, jak moc je třída na obrázcích obsáhlá.

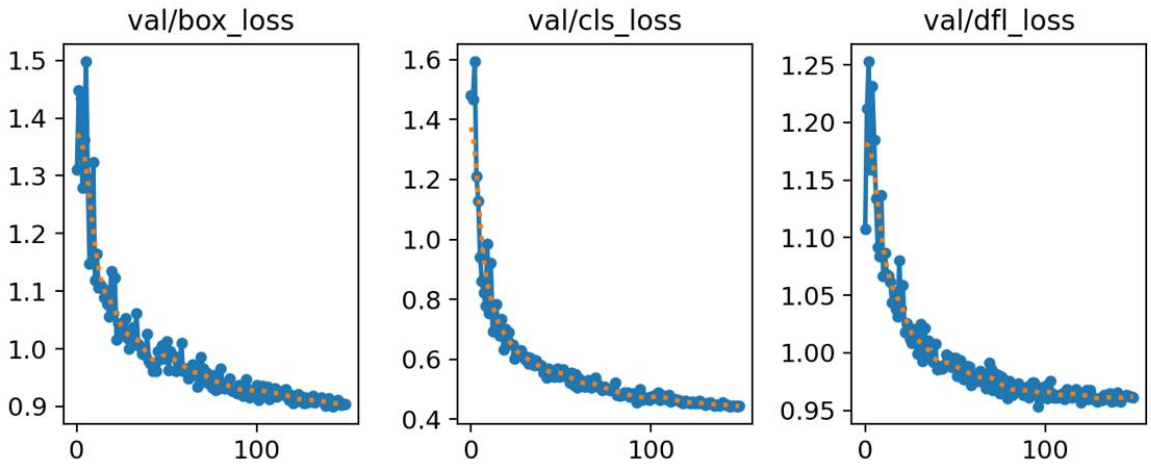
Ztrátové hodnoty v Obrázku 22. jsou trénovací hodnoty. Tyto hodnoty obvykle v průběhu trénování stále klesají. Trénovací ztrátové hodnoty ovšem nevypovídají o tom, jak si bude model počínat na datech při reálném použití, protože vyhodnocuje předpovědi pouze na obrázcích v datasetu. Pokud by se braly v potaz pouze trénovací hodnoty, mělo by to za následek, že se model naučí rozpoznávat pouze data, na kterých byl natrénován, na úkor reálných případů a nevytvořil by si pro objekty dostatečnou míru generalizace. Tomuto jevu se říká overfitting.



Obrázek 21. Trénovací ztrátové hodnoty detekčního modelu

Výrazně lepší přehled o tom, jak by si model vedl v reálném použití, podávají validační ztrátové hodnoty (viz Obrázek 23), protože validační data se nevyskytují v datasetu pro trénování.

V ideálním případě se grafy těchto hodnot rovnají, nicméně validační ztrátové hodnoty jsou v praxi skoro vždy o něco vyšší než ty trénovací.



Obrázek 22.Validační ztrátové hodnoty detekčního modelu

Po zkušebním natrénování detekčního modelu na výchozí nastavení na 100 epoch bylo zjevné, že modelu prospěje více epoch, jelikož validační ztrátová hodnota se po dokončení trénování úplně neustálila a stále klesala. Z toho důvodu byl pro další trénování zvýšen počet epoch na 150.

Zároveň trénovací ztrátové hodnoty byly znatelně níže než validační, což je příznak overfittingu. Pro zvýšení generalizace byl argument pro trénování weight_decay zvýšen na úroveň 0.0008 z původních 0.0005. Tento argument aplikuje L2 regularizaci a předchází tak overfittingu. Augmentace byla ponechána na výchozích hodnotách.

Výchozí nastavení je dohledatelné na stránce: https://docs.ultralytics.com/usage/cfg/ [44]

Z obrázků 22 a 23 je zjevné, že se jednotlivé validační hodnoty od trénovacích výrazně neliší a zároveň se ustalují, tudíž není pravděpodobné, že by vyšší počet epoch zlepšil přesnost.

6.2 Trénování segmentačního modelu

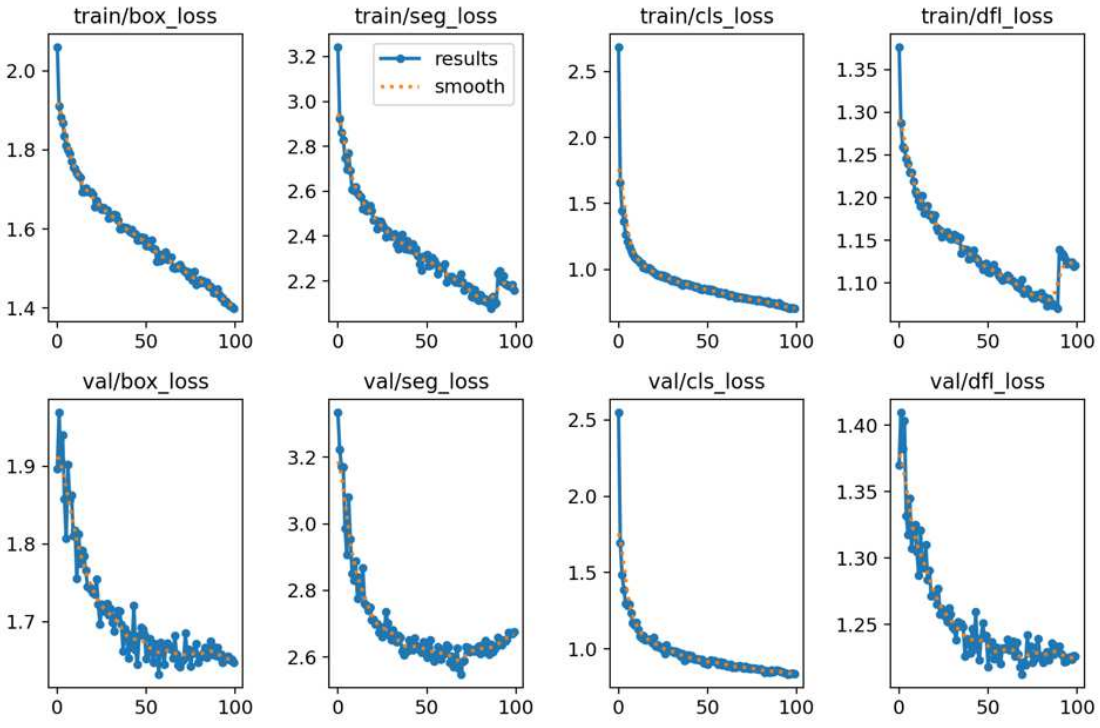
Model, který segmentuje tři třídy je výrazně těžší natrénovat než model, který detekuje pouze jednu. Proto lze očekávat, že bude mít horší výsledky než detekční model, co se ztráty a metrik týče.

Pro natrénování segmentačního modelu byl použit model pro segmentaci YOLOv8n-seg. Po úvodním natrénování na výchozí nastavení na 100 epoch se ukázalo, že model vykazuje

výrazné známky overfittingu, a to do fáze, kdy se validační ztrátová hodnota pro segmentaci začala znatelně zvyšovat okolo epochy 50. Zhoršení přesnosti modelu v průběhu trénování automaticky neznamená neúspěch, jelikož YOLO ukládá vždy model, který má nejlepší výsledky. Obvykle ale za tento fakt nemůže pouze nadměrné množství epoch a je potřeba upravit nastavení trénování. Z toho důvodu byla hodnota weight_decay zvýšena na hodnotu 0.001. Zvýšení tohoto parametru sice navýšilo ztrátové hodnoty modelů, ale zlepšilo přesnost modelu, co se týče metrik. Hodnota augmentace scale byla zvýšena na hodnotu 0.7, aby byl model lepší v segmentování vzdálených osob.

Výchozím nastavením pro optimalizaci YOLOv8 je režim auto. Tento režim pro prvních pár epoch aplikuje variaci optimalizátoru Adam a následně pokračuje s optimalizátorem SGD. Pro snížení overfittingu u epochy 50 byla míra učení pro SGD nastavena na hodnotu 0.05 z původní 0.1.

Obrázek 23 zobrazuje ztrátové hodnoty v průběhu trénování. Co se týče rozdílů mezi trénovacími a validačními ztrátovými hodnotami, jediným případem, kde není výrazný rozdíl, je klasifikační ztráta. Vzhledem k tomu, že pro model není hlavní výzvou klasifikovat jednotlivé detekované části obličeje, můžeme říct, že grafy klasifikačních ztrátových hodnot ukazují, že model zvládá rozlišovat jednotlivé části obličeje od pozadí. O poznání hůře si ale vede ve vytváření ohraničujících rámečků a segmentačních masek.



Obrázek 23. Ztrátové hodnoty při trénování segmentačního modelu

7 VYHODNOCENÍ MODELŮ

Pro vyhodnocování datasetů v oblasti počítačového vidění se používají křivky precision a recall. Hodnoty precision udávají, kolik s modelem nalezených objektů bylo vyhodnoceno správně. Je to počet všech správně vyhodnocených objektů lomeno počet všech nalezených instancí. Oproti tomu hodnoty recall udávají, kolik objektů model nalezl oproti tomu, kolik jich nalézt měl. Tato hodnota se počítá jako počet všech správně vyhodnocených objektů lomeno počet správně vyhodnocených objektů a počet objektů, které nebyly modelem zaznamenány [40].

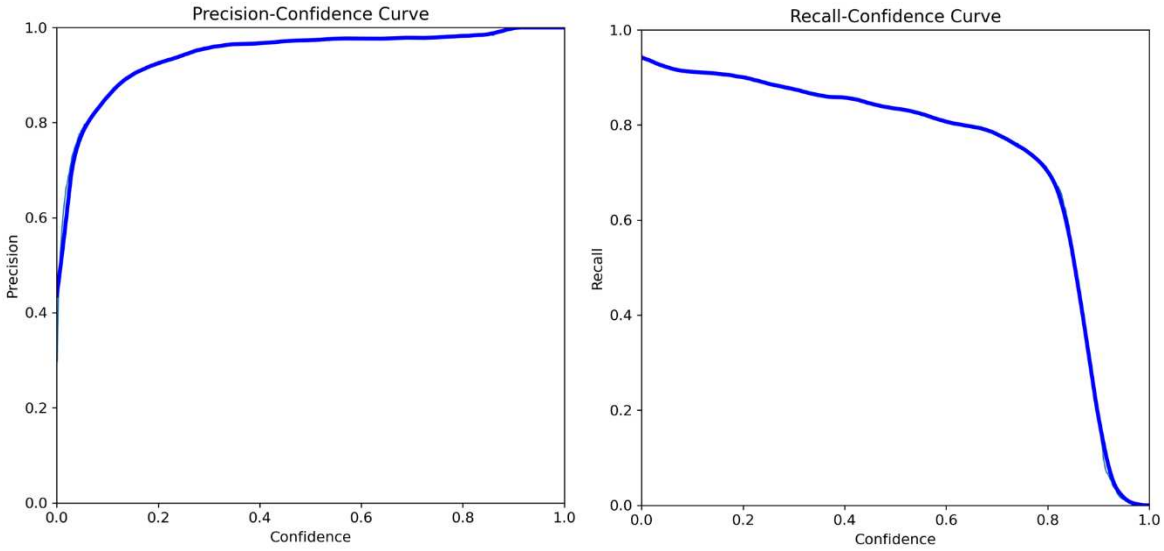
Hodnoty precision a recall se dále využívají pro utváření dalších metrik a grafů, podle kterých se vyhodnocují modely. Často se uvádí precision-recall křivka, která zobrazuje poměr mezi těmito hodnotami, nebo F1 křivka, která kombinuje tyto hodnoty a zobrazuje je v poměru ku hodnotě confidence, která udává, nakolik je model přesvědčen svou predikcí. [40]

Často využívanou metrikou je mAP (Mean Average Precision, střední průměrná přesnost). Tato metrika vypočítává prostory pod precision-recall křivkami různých tříd a zprůměruje je. Existuje více typů mAP metrik. Mezi používané patří mAP50, která počítá přesnost pro jednoduše detekovatelné objekty, nebo mAP50-95, která bere v potaz složitěji detekovatelné objekty. [40]

7.1 Vyhodnocení detekčního modelu

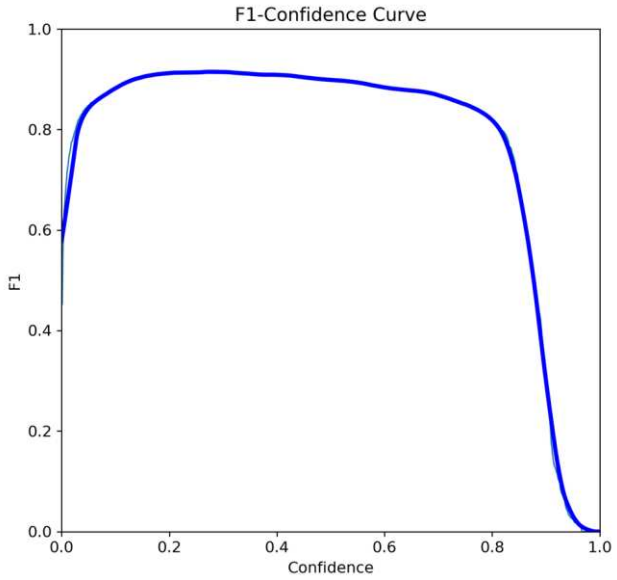
Obrázek 24 obsahuje křivku precision vůči hodnotě confidence a křivku recall vůči confidence. Z precision křivky je zjevné, že model odvádí dobrou práci ve vyhodnocování detekce objektu spadajícího skutečně do detekované třídy, tedy třídy hlava. Od míry confidence s hodnotou více jak 0.2 má model presicion 0.9. Což znamená, že již od malé confidence je velká pravděpodobnost, že je objekt detekován správně.

Z křivky recall lze vyčíst, že ne všechny detekované objekty byly detekovány s vysokou míru confidence.



Obrázek 24. Křivky precision a recall vůči confidence detekčního modelu

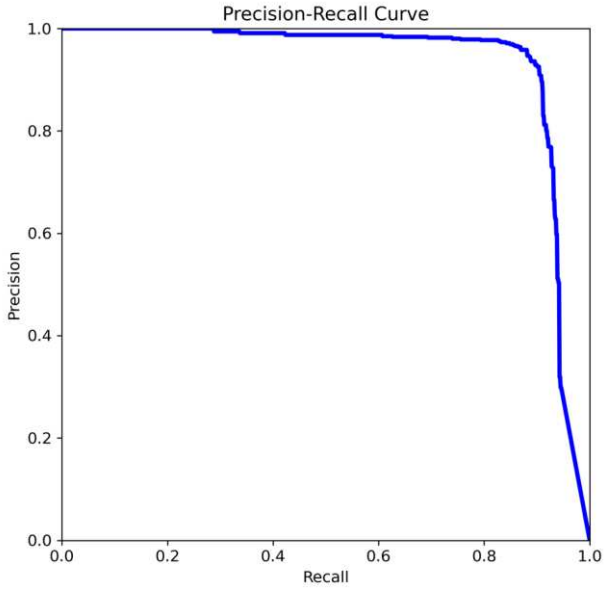
Kombinací těchto dvou křivek vznikne F1 křivka (Obrázek 25). Tato křivka ukazuje oboje, od které míry confidence je velká pravděpodobnost, že byla hlava správně detekována, po rozpoložení confidence hodnot pro jednotlivé detekce. F1 křivka tohoto detekčního modelu má ploché místo ve vrchní části, což je obvykle známka dobře natrénovaného modelu.



Obrázek 25. F1 křivka modelu pro detekci hlav

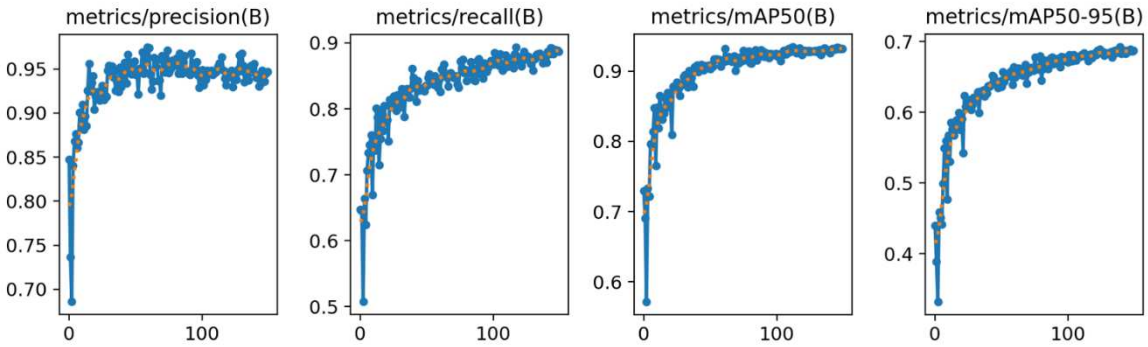
Balanc mezi hodnotami presicion a recall je zobrazen v presicion-recall křivce Obrázek 26. Z této křivky je zjevné, že má precision před recallem přednost, což je žádoucí, jelikož tento

model může být využit pro statistiku výskytu osob v místnosti, kde je lepší na chvíli nějakou osobu nezachycovat, než konstantě chybně vyhodnocovat nehybný objekt jako člověka.



Obrázek 26. Precision-recall křivka modelu pro detekci hlav

Obrázek 27 ukazuje, jak se měnily metriky v závislosti na čísle epochy v průběhu trénování modelu. Model dosáhl vysokých precision hodnot relativně rychle, ale vysokého recallu až v průběhu trénování. Metrika mAP50 dosahuje lepších hodnot než mAP50-95, což je očekávatelné, jelikož mAP50-95 se zaměřuje na složitější detekce. Celkově model dosahuje dobrých výsledků, nicméně je nutné zmínit že detekce jedné třídy je relativně jednoduchý úkol.



Obrázek 27. Metriky v závislosti na epochách u detekčního modelu

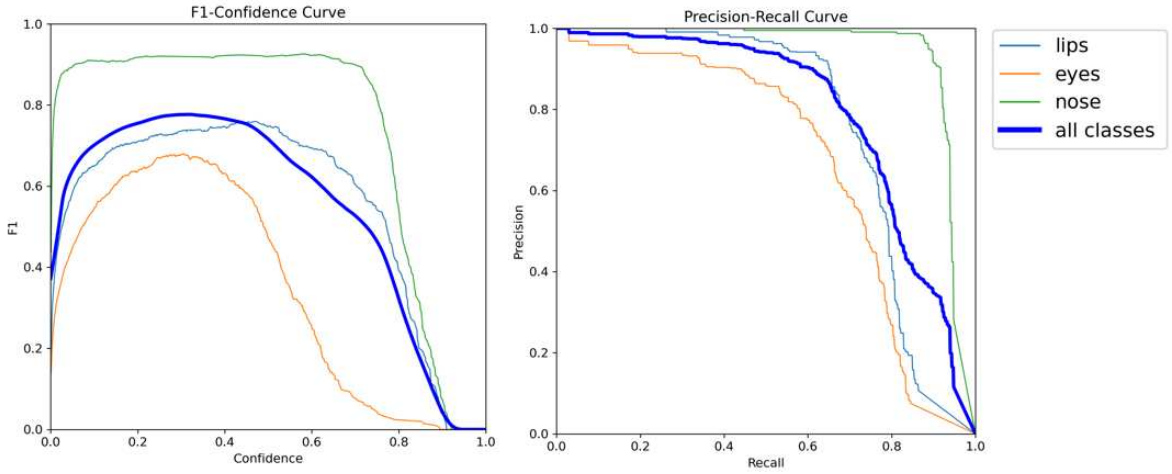
7.2 Vyhodnocení segmentačního modelu

Vyhodnocení segmentačního modelu je komplexnější, jelikož nebere v potaz pouze ohraničující rámečky okolo objektů, ale taky segmentační masky. Zároveň detekční model vyhodnocuje vůči pozadí pouze hlavu, což je pouze jedna třída. Segmentační model vyhodnocuje 3 odlišné třídy a to oči, nos a ústa.

7.2.1 Vyhodnocení pro ohraničující rámečky

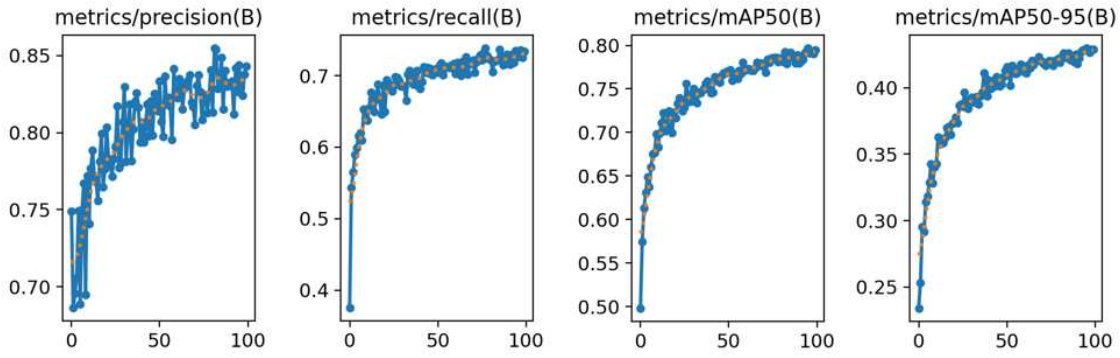
Při pohledu na F1 křivku na Obrázku 28 je zjevné, že model nevyhodnocuje všechny třídy se stejnou úspěšností. Jedinou třídou, u které se křivka blíží ideálnímu průběhu, je nos. U vyhodnocování ostatních tříd model zdaleka neodvádí tak dobrou práci. Nejhorší výsledky jsou pak při vyhodnocování očí. F1 křivka naznačuje, že i v případě, kde model oči detekuje, přiřazuje těmto detekcím nízké hodnoty confidence.

Precision-recall křivka na Obrázku 28 ukazuje, že u všech tříd má precission přednost před recallem, což je žádané, nicméně třídy eyes a lips neodpovídají ideálnímu průběhu.



Obrázek 28. F1 a PR křivka pro ohraničující rámečky u segmentace

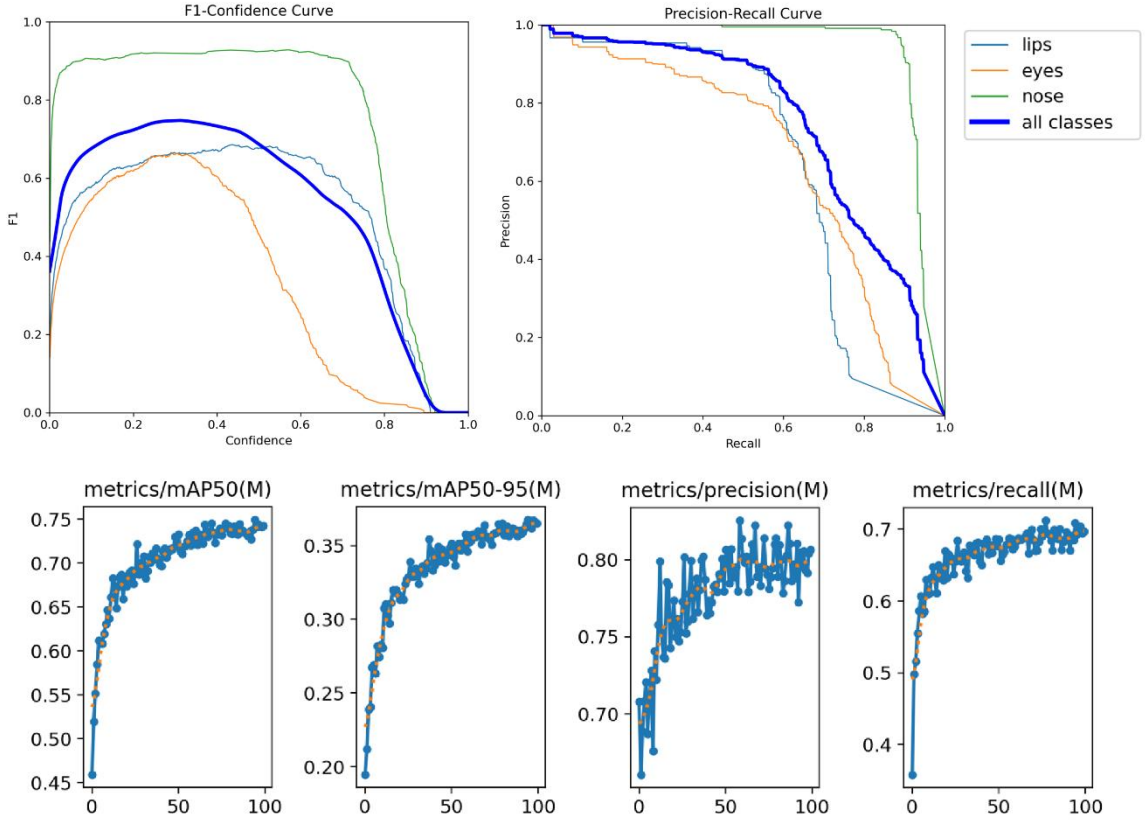
Metriky v závislosti na jednotlivých epochách jsou zobrazeny v obrázku 29. Zde je vidět, že míra precision je v průběhu trénování velmi nestabilní. Rozdíl mezi metrikami mAP50 a mAP50-95 je poměrně velký. Důvodem toho je pravděpodobně fakt, že do metriky mAP50-95 spadají objekty, které se složitěji vyhodnocují, a to jsou v tomto případě z velké části oči, které má model problém vyhodnotit.



Obrázek 29. Metriky v závislosti na epochách pro ohraničující rámečky u segmentace

7.2.2 Vyhodnocení pro segmentačních masky

Metriky vyhodnocující segmentační masky se nacházejí v obrázku 30. Charakteristika těchto metrik se příliš neliší od těch, které vyhodnocují ohraničující rámečky. Jediný výrazný rozdíl je, že u všech metrik v závislosti na epochách dosahuje model slabších výsledků, jelikož vyhodnocování segmentačních masek je komplikovanějším úkolem.



Obrázek 30. Metriky vyhodnocování segmentačních masek

8 KONVERTOVÁNÍ MODELŮ

V základu jsou YOLO modely ukládané s koncovkou .pt. Znamená to tedy, že jsou to PyTorch modely. Ne ve všech případech využití výsledných modelů této práce však lze modely využít v tomto formátu. Například bylo by potřeba použít modely v zařízení, které využívá TPU, musel by se model konvertovat například do formátu tensorflow-lite.

Obvyklým postupem při přenosu modelů do jiných formátů je konvertovat model do formátu ONNX (Open Neural Network Exchange). ONNX je standardizovaný formát modelu, který slouží jako mezikrok pro další převádění do formátů, které jsou specializované pro konkrétní použití, nebo optimalizované pro konkrétní typ zařízení. Existuje-li případ užití, ve kterém by se model konvertoval do jiného formátu, očekává se, že bude konvertován do formátu ONNX.

8.1 Testování konvertovaných modelů

Při konvertování modelu do jiných formátů je potřeba vyhodnotit, zdali se přenosnost modelu výrazně nezhoršila, a dále si udělat přehled rychlosti v konkrétních formátech.

Pro testování formátů byl použit benchmark nástroj od Ultralytics, který zároveň konvertuje YOLOv8 model do dalších formátů. Ne u všech formátů proběhlo konvertování úspěšně, což je problém vývojového prostředí, nicméně důležitou věcí bylo, aby bylo možné provést testování pro formát ONNX, do kterého se modely konvertovaly úspěšně.

8.1.1 Testování detekčního modelu v různých formátech

Nástroj benchmark automaticky testuje různé formáty na validačních datech. Vzhledem k tomu, že pro detekční model byla data rozdělena i na testovací sadu, bylo žádané, aby testování proběhlo na této sadě. Testování na datech, která nebyla jakkoliv vystavena modelu při trénování, má lepší výpovědní hodnotu o tom, jak je model úspěšný při reálném využití. Z tohoto důvodu bylo zaměněno umístění validačních dat za cestu k těm testovacím ve složce data.yaml, která slouží pro konfiguraci benchmark nástroje.

8.1.1.1 Testování detekce na grafické kartě

Tabulka 3 obsahuje výsledky testování na grafické kartě NVIDIA GeForce GTX 1050 Ti, 4096MiB. Formáty ncnn a ONNX mají výrazně nižší rychlost, což může znamenat, že test proběhl na procesoru místo grafické karty. Samostatné testování ukázalo podobné výsledky.

Tabulka 2: Výsledky testování detekce na grafické kartě

Formát	Velikost (MB)	mAP50-95(B)	Rychlost (ms/im)
PyTorch	6.0	0.7496	12.76
TorchScript	11.9	0.7474	12.69
ONNX	11.7	0.7474	77.20
ncnn	11.6	0.7474	92.54

8.1.1.2 Testování detekce na procesoru

Vzhledem k tomu, že pravděpodobně ne u všech formátů proběhlo správné testování na grafické kartě, bylo na místě provést ještě trénování na procesoru z důvodu porovnání formátů z hlediska rychlosti. Při testování detekčního modelu na procesoru se automaticky vyhodnotil ještě formát OpenVINO. Testování bylo provedeno na procesoru AMD Ryzen 5 1600X Six-Cor. Výsledky testování jsou v tabulce 4.

Tabulka 3: Výsledky testování detekce na procesoru

Formát	Velikost (MB)	mAP50-95(B)	Rychlost (ms/im)
PyTorch	6.0	0.7496	135.01
TorchScript	11.9	0.7474	118.50
ONNX	11.7	0.7474	76.66
OpenVINO	11.8	0.7474	56.67
ncnn	11.6	0.7474	88.48

8.1.2 Testování segmentačního modelu v různých formátech

Stejný postup byl proveden i pro testování různých formátů segmentačního modelu. Pro tento model však nebyl testován na separátním testovacím datasetu, ale na validačním datasetu, z důvodu úspory času. Toto však nemusí být nutně špatně, jelikož nástroj benchmark automaticky počítá s tím, že bude testovat na validačních datech.

8.1.2.1 Testováni segmentace na grafické kartě

V Tabulce 5 jsou zaznamenány výsledky testování pro segmentaci. Vyhodnocení v metrice mAP50-95 je zde pro segmentační masky.

Tabulka 4. výsledky testování segmentace na grafické kartě

Formát	Velikost (MB)	mAP50-95(M)	Rychlost (ms/im)
PyTorch	6.5	0.3636	13.91
TorchScript	12.9	0.3577	11.33
ONNX	12.6	0.3577	108.65

8.1.2.2 Testováni segmentace na procesoru

Stejně jako u detekčního modelu bylo provedeno i testování na procesoru. Výsledky jsou v tabulce 6.

Tabulka 5. Výsledky testování segmentace na procesoru

Formát	Velikost (MB)	mAP50-95(M)	Rychlost (ms/im)
PyTorch	6.5	0.3636	169.15
TorchScript	12.9	0.3577	151.89
ONNX	12.6	0.3577	110.40
OpenVINO	12.8	0.3577	78.93

8.2 Výsledky testování

Důležité informace jsou, jak si vedl formát ONNX v porovnání s formátem PyTorch. Rozdíly v metrikách těchto formátů jsou v tabulce 7. Rozdíly vyhodnoceny metrikou mAP50-95 nejsou zásadní ani v případě segmentace, ani v případě detekce, což se dá považovat za úspěšné převedení.

Tabulka 6. Porovnání výsledků formátů [45]

Formát	segmentace mAP50-95(M)	detekce mAP50-95(B)
PyTorch	0.3636	0.7496
ONNX	0.3577	0.7474

Rychlost u ONNX modelu je mírně nejasná, jelikož je těžké posoudit, zdali funguje správně na grafické kartě, nicméně bude-li se využívat formát ONNX, bude to za účelem dalšího konvertování do specializovaných formátů, které budou mít odlišné rychlosti. Dále je v kontextu rychlosti třeba mít na paměti, že konvertováním YOLOv8 do ONNX se odstranila poslední vrstva, která zastávala postprocessing. Při využívání konvertovaného modelu by bylo nutné tuto vrstvu nahradit a tento proces může mít za následek nižší rychlost.

9 IMPLEMENTACE

Cílem implementace je kombinovat predikce obou modelů pro dosažení jak detekce, tak segmentace ideálně v reálném čase a dostatečně přesně, aby bylo možné další využití.

9.1 Snímání obrazu

Pro vytvoření programu, který tyto modely implementuje, je potřeba nejprve zaznamenat obraz, který bude sloužit jako vstupní data pro modely. Pro nahrávání videa je využita kamera – 8MP Sony IMX179, která má automatické zaostřování a je k počítači připojena přes USB. Obraz je snímán v rozlišení 640x480. S touto kamerou pracuje knihovna OpenCV, která čte výstup z kamery jako jednotlivé snímky.

9.2 Implementace detekce

Na jednotlivých snímcích je dále prováděna detekce za použití YOLOv8 modelu pro detekci hlav osob, který je jedním z výsledků této práce. Detekce je prováděna s nastavením device=0, což znamená, že je prováděna na grafické kartě, v tomto případě na kartě NVIDIA GeForce GTX 1050 Ti, 4096MiB. Dalším parametrem pro vytvoření predikce je conf=0.4. Tento parametr vyfiltruje všechny detekce, které nemají vyšší míru confidence než 0.4. Ostatní parametry jsou ponechány na výchozích hodnotách, které jsou popsány v YOLOv8 dokumentaci [44]. Veškerý postprocessing, jako například non-max-suppression, se provádí automaticky. Po převedení výstupu do formy, která lze přečíst procesorem, jsou z výsledku extrahovány ohraničující rámečky, které jsou ve formě pole obsahující souřadnice středu rámečku a rozměry rámečku.

9.3 Implementace segmentace

U segmentace nemá smysl dávat jako vstup celý snímek z kamery. Segmentace se provádí pouze tam, kde byla dříve detekována hlava. Pro vytvoření vstupních dat pro segmentační model jsou využity dříve získané ohraničující rámečky z detekčního modelu. Pro každou detekci je vytvořen výřez z původního snímku kamery, který je na souřadnicích jednotlivých detekcí a má odpovídající rozměry. Tyto výřezy jsou dále použity jako vstup pro segmentační model. Predikce segmentačního modelu má stejné nastavení jako u detekčního modelu, až na parametr conf, který je snížen na hodnotu 0.25. Důvodem je dříve vyhodnocená charakteristika segmentačního modelu, která značí, že model nepřiřazuje segmentovým oblastím velkou míru confidence.

9.4 Zobrazení predikcí

Pro vykreslení detekce a segmentace se používá funkce plot, kterou lze aplikovat na výstup z YOLOv8 modelu. Tato funkce převede výsledné rámečky a segmentační masky na formát, který lze přepsat do obrázku v podobě numpy pole. Pro zobrazení jsou vypnuty možnosti vykreslovat názvy tříd a u segmentace je vypnuto zobrazování ohraničujících rámečků pro čistější zobrazení.

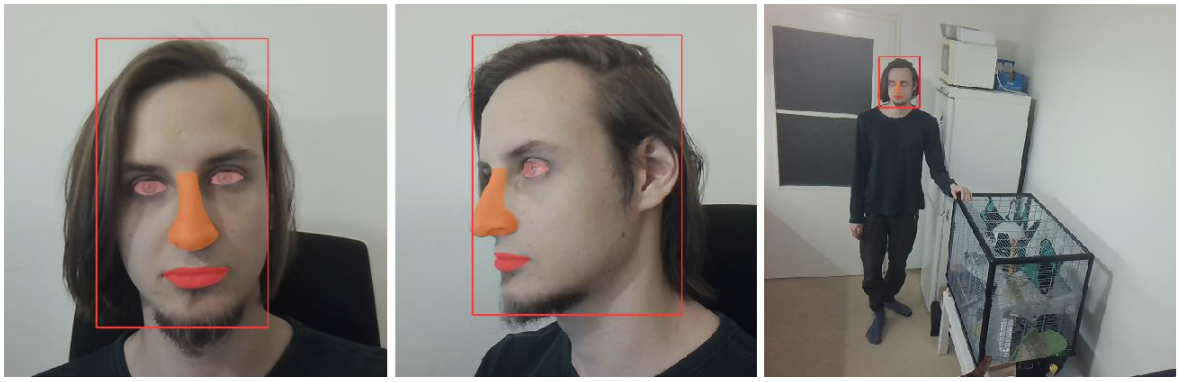
9.5 Zhodnocení použití

Hlavní obavou při vytváření aplikace bylo, že segmentace bude fungovat pouze na velmi krátkou vzdálenost. Model je v tomto ohledu lepší, než se očekávalo, čemuž mohla pomoct augmentace. Pracuje-li segmentační model na velké vzdálenosti nepřesně, většinou za to může špatné zaostření u kamery. Kamera má zároveň malé rozlišení, které segmentaci značně ovlivňuje.

Detekční model v této implementaci funguje bez problémů a správně vyhodnocuje hlavy ze všech stran, i ty, které jsou částečně zakryté.

Segmentace má uspokojivé výsledky, co se týče vyhodnocování obličeje, který je z profilu, nebo jinak natočen. Díky tomu lze model použít pro vyhodnocení pózy hlavy, což je jeden z případů užití.

Obrázek 31 ukazuje výsledná zobrazení pro různé případy.



Obrázek 31. Výsledky implementace modelů

Nevýhodou této implementace je, že s vyšším počtem osob na snímku klesá rychlost díky většímu počtu vyhodnocování segmentace, což může mít za následek, že ne příliš výkonné systémy nebudou pracovat v reálném čase.

ZÁVĚR

Cílem této práce bylo vytvořit modely pro systém, který vytváří přehled o osobách nacházejících se poblíž reklamního panelu ve veřejných prostorech. Konkrétněji to byl model pro detekci osob a model pro segmentaci lidského obličeje. Nejprve byly v teoretické části práce probrány principy umělé inteligence. Dále byly zanalyzovány metody v oblasti počítačového vidění za použití konvolučních neuronových sítí a konkrétněji taky metody segmentace v oblasti lidského obličeje. V praktické části práce je pospána samotná tvorba modelů, pro ktero bylo použito YOLOv8 nano, které bylo vybráno kvůli své rychlosti s ohledem na předchozí analýzu. U segmentačního modelu bylo zvoleno Facial Landmark Extraction kvůli své multifunkčnosti. Nejprve byly vybrány datasety, a dále na nich byly modely natrénovány. Praktická část popisuje jak výsledky v průběhu trénování, tak finální vyhodnocení podle metrik. U detekčního modelu bylo dosaženo uspokojivých výsledků a vlastností. Model, který segmentuje části obličeje, vykazoval při trénování výrazné známky overfittingu a řešení tohoto problému bylo časově náročné, jelikož bylo trénování prováděno na ne příliš výkonném počítači. Výsledná verze tohoto modelu si počíná hůře při vyhodnocování některých tříd. Modely byly dále převedeny do formátu ONNX a byly v tomto formátu otestovány. Výsledky tohoto testování ukazují, že byly převedeny úspěšně. Poslední částí práce je příklad implementace těchto modelů, který slouží pro zkontrolování funkčnosti s reálnou kamerou.

Výsledkem této práce jsou modely ve formátu ONNX, kde model na detekci hlav dosahuje přesnosti 0.7474 mAP50-95(B) a model pro segmentaci obličeje dosahuje přesnosti 0.3577 mAP50-95(M).

Do budoucna může být dalším směrem vývoje vytvoření modelů, které vyhodnocují konkrétní informace, jako je póza hlavy, nálada či věk nebo pohlaví. Dále potom optimalizace jak implementace, tak jednotlivých modelů a konvertování a kvantování modelů do formátu, který může běžet na TPU zařízeních.

SEZNAM POUŽITÉ LITERATURY

[1] ROSEBROCK, Adrian. Deep learning for computer vision with Python: Starter bundle. PyImageSearch, 2017. ISBN 9781986538138.

[2] REBALA, Gopinath; RAVI, Ajay a CHURIWALA, Sanjay. An Introduction to Machine Learning. Online. Springer Cham, 2019. ISBN 978-3-030-15729-6. Dostupné z: https://doi.org/https://doi.org/10.1007/978-3-030-15729-6.

[3] SARKER, Iqbal H. Machine Learning: Algorithms, Real-World Applications and Research Directions: Algorithms, Real-World Applications and Research Directions. SN Computer Science. 2021, roč. 2, č. 3, s. 160. ISSN 2661-8907. Dostupné z: https://doi.org/10.1007/s42979-021-00592-x.

[4] JIANG, Xiaoyue; HADID, Abdenour; PANG, Yanwei; GRANGER, Eric a FENG, Xiaoyi (ed.). Deep learning in object detection and recognition. Singapur: Springer Singapore, [2019]. ISBN 978-981-10-5151-7.

[5] GOODFELLOW, Ian, Yoshua BENGIO a Aaron COURVILLE. Deep learning: ImageNet bundle. Adaptive computation and machine learning. Cambridge, Massachussetts ; London: The MIT Press, 2016. ISBN 9780262035613.

[6] ABIODUN, Oludare Isaac; JANTAN, Aman; OMOLARA, Abiodun Esther; DADA, Kemi Victoria; MOHAMED, Nachaat Abdelatif et al. State-of-the-art in artificial neural network applications: A survey: A survey. Heliyon. 2018, roč. 4, č. 11, s. 1-41. ISSN 2405-8440. Dostupné z: https://doi.org/10.1016/j.heliyon.2018.e00938.

[7] SCHMIDHUBER, Jürgen. Deep learning in neural networks: An overview: An overview. Neural Networks. 2015, roč. 61, s. 85-117. ISSN 0893-6080. Dostupné z: https://doi.org/https://doi.org/10.1016/j.neunet.2014.09.003.

[8] AKGÜN, Ergün a DEMIR, Metin. Modeling Course Achievements of Elementary Education Teacher Candidates with Artificial Neural Networks. Online. International Journal of Assessment Tools in Education. 2018, roč. 5, č. 3, s. 491-509. ISSN 2148-7456. Dostupné z: https://doi.org/10.21449/ijate.444073. [cit. 2024-05-07].

[9] ROSENBLATT, Frank. The perceptron: a probabilistic model for information storage and organization in the brain: a probabilistic model for information storage and organization in the brain. Psychological review. 1958, roč. 65, č. 6, s. 386-408. Dostupné také z: https://api.semanticscholar.org/CorpusID:12781225.

[10] CHOI, Rene Y; COYNER, Aaron S; KALPATHY-CRAMER, Jayashree; CHIANG, Michael F a CAMPBELL, J Peter. Introduction to machine learning, neural networks, and deep learning. Transl. Vis. Sci. Technol. 2020, roč. 9, č. 2, s. 14. ISSN 2164-2591.

[11] SHARMA, Sagar; SHARMA, Simone a ATHAIYA, Anidhya. Activation functions in neural networks. Towards Data Sci. 2017, roč. 6, č. 12, s. 310-316.

[12] KARLIK, Bekir a OLGAC, A Vehbi. Performance analysis of various activation functions in generalized MLP architectures of neural networks. International Journal of Artificial Intelligence and Expert Systems. 2011, roč. 1, č. 4, s. 111-122.

[13] WANG, Qi; MA, Yue; ZHAO, Kun a TIAN, Yingjie. A Comprehensive Survey of Loss Functions in Machine Learning. Annals of Data Science. 2022, roč. 9, č. 2, s. 187-212. ISSN 2198-5812. Dostupné z: https://doi.org/10.1007/s40745-020-00253-5.

[14] NIE, Feiping; HU, Zhanxuan a LI, Xuelong. An investigation for loss functions widely used in machine learning. Commun. Inf. Syst. 2018, roč. 18, č. 1, s. 37-52. ISSN 1526-7555. Dostupné z: https://doi.org/10.4310/cis.2018.v18.n1.a2.

[15] SARANGAM, Ajay. Classification vs Regression: An Easy Guide in 6 Points. Online. In: U-next. 2021. Dostupné z: https://u-next.com/blogs/artificial-intelligence/classification-vs-regression/. [cit. 2024-05-07].

[16] SHANKAR297. Understanding Loss Function in Deep Learning. Online. In: Analytics Vidhya. 2024. Dostupné z: https://www.analyticsvidhya.com/blog/2022/06/understanding-loss-function-in-deep-learning/. [cit. 2024-05-07].

[17] GAD, Ahmed. A Comprehensive Guide to the Backpropagation Algorithm in Neural Networks. Online. In: Neptune.ai. 2023. Dostupné z: https://neptune.ai/blog/backpropagation-algorithm-in-neural-networks-guide. [cit. 2024-05-07].

[18] VERMA, Ajay. Navigating the Terrain: Convex vs. Non-Convex Functions in Optimization. Online. In: Medium. 2023. Dostupné zhttps://ai.plainenglish.io/navigating-the-terrain-convex-vs-non-convex-functions-in-optimization-86812e9a1989#. [cit. 2024-05-07].

[19] ROSEBROCK, Adrian. Deep learning for computer vision with Python: Practitioner Bundle. PyImageSearch, 2017. ISBN 978-1-986723817.

[20] ALZUBAIDI, Laith; ZHANG, Jinglan; HUMAIDI, Amjad J.; AL-DUJAILI, Ayad; DUAN, Ye et al. Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. Journal of Big Data. 2021, roč. 8, č. 1, s. 2-57. ISSN 2196-1115. Dostupné z: https://doi.org/10.1186/s40537-021-00444-8.

[21] Image Segmentation: The Basics and 5 Key Techniques. Online. Datagen. Dostupné z: https://datagen.tech/guides/image-annotation/image-segmentation/#. [cit. 2024-05-08].

[22] GUO, Yanming; LIU, Yu; GEORGIOU, Theodoros a LEW, Michael S. A review of semantic segmentation using deep neural networks. International Journal of Multimedia Information Retrieval. 2018, roč. 7, č. 2, s. 87-93. ISSN 2192-662X. Dostupné z: https://doi.org/10.1007/s13735-017-0141-z.

[23] What is image segmentation? Online. IBM. Dostupné z: https://www.ibm.com/topics/image-segmentation. [cit. 2024-05-07].

[24] WANG, Zhuoyue. SEG-YOLO: Real-time instance segmentation using YOLOv3 and fully convolutional network. Master. Stockholm: School of Electrical Engineering and Computer Science, 2019.

[25] SHELHAMER, Evan; LONG, Jonathan a DARRELL, Trevor. Fully convolutional networks for semantic segmentation. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2015, s. 3431-3440. Dostupné také z: https://api.semanticscholar.org/CorpusID:1629541.

[26] WANG, Zhengyang; ZOU, Na; SHEN, Dinggang a JI, Shuiwang. Non-local U-Net for Biomedical Image Segmentation. ArXiv: Computer Vision and Pattern Recognition. 2020, roč. 34, č. 4, s. 6315--6322. Dostupné také z: https://api.semanticscholar.org/CorpusID:263793182.

[27] ZHAO, Zhong-Qiu; ZHENG, Peng; XU, Shou-Tao a WU, Xindong. Object Detection With Deep Learning: A Review. IEEE Transactions on Neural Networks and Learning Systems. 2018, roč. 30, s. 3212-3232. Dostupné také z: https://api.semanticscholar.org/CorpusID:49862415.

[28] DENG, Jun; XUAN, Xiaojing; WANG, Weifeng; LI, Zhao; YAO, Hanwen et al. A review of research on object detection based on deep learning. Journal of Physics: Conference Series. 2020, roč. 1684, č. 1. ISSN 1742-6596. Dostupné z: https://doi.org/10.1088/1742-6596/1684/1/012028.

[29] DALAL, Navneet a TRIGGS, Bill. Histograms of oriented gradients for human detection. 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05). 2005, roč. 1, s. 886-893. ISSN 1063-6919. Dostupné z: https://doi.org/10.1109/CVPR.2005.177.

[30] CORTES, Corinna a VAPNIK, Vladimir Naumovich. Support-Vector Networks. Machine Learning. 1995, roč. 20, s. 273-297. Dostupné také z: https://api.semanticscholar.org/CorpusID:52874011.

[31] LOHIA, Aditya; KADAM, Kalyani; JOSHI, Rahul a BONGALE, Arunkumar. Bibliometric Analysis of One-stage and Two-stage Object Detection. Online. Library Philosophy and Practice. 2021/02/13, roč. 4910. Dostupné z: https://digitalcommons.unl.edu/libphilprac/4910. [cit. 2024-05-08].

[32] TSANG, Sik-Ho. Review: SSD — Single Shot Detector (Object Detection). Online. In: Medium. 2018. Dostupné z: https://towardsdatascience.com/review-ssd-single-shot-detector-object-detection-851a94607d11. [cit. 2024-05-07].

[33] TERVEN, Juan; CORDOVA-ESPARZA, Diana. A comprehensive review of YOLO: From YOLOv1 to YOLOv8 and beyond. arXiv preprint arXiv:2304.00501, 2023.

[34] KHAN, Khalil; KHAN, Rehanullah; AHMAD, Kashif; ALI, Farman a KWAK, Kyung. Face Segmentation: A Journey From Classical to Deep Learning Paradigm, Approaches, Trends, and Directions. IEEE Access. 2020/03/24, roč. 4, s. 1-17. Dostupné z: https://doi.org/10.1109/ACCESS.2020.2982970.

[35] KUMAR, Ashu; KAUR, Amandeep a KUMAR, Munish. Face detection techniques: a review. Artif. Intell. Rev. 2019, roč. 52, č. 2, s. 927-948. ISSN 0269-2821. Dostupné z: https://doi.org/10.1007/s10462-018-9650-2.

[36] KORTLI, Yassin; JRIDI, Maher; AL FALOU, Ayman a ATRI, Mohamed. Face Recognition Systems: A Survey. Online. Sensors. 2020, roč. 20, č. 2. ISSN 1424-8220. Dostupné z: https://doi.org/10.3390/s20020342. [cit. 2024-05-07].

[37] ADJABI, Insaf; OUAHABI, Abdeldjalil; BENZAOUI, Amir a TALEB-AHMED, Abdelmalik. Past, Present, and Future of Face Recognition: A Review. Online. Electronics. 2020, roč. 9, č. 8. ISSN 2079-9292. Dostupné z: https://doi.org/10.3390/electronics9081188. [cit. 2024-05-07].

[38] ZENG, Dan; VELDHUIS, Raymond a SPREEUWERS, Luuk. A survey of face recognition techniques under occlusion. IET biometrics. 2020, roč. 10, č. 6, s. 581-606. Dostupné také z: http://arxiv.org/abs/2006.11366.

[39] ZHANG, Teng; DENG, Lirui; ZHANG, Liang a DANG, Xianglei. Deep Learning in Face Synthesis: A Survey on Deepfakes: A Survey on Deepfakes. In: 2020 IEEE 3rd International Conference on Computer and Communication Engineering Technology (CCET). Beijing, China: IEEE, 2020, s. 67-70. ISBN 978-1-7281-8811-9. Dostupné z: https://doi.org/10.1109/CCET50901.2020.9213159.

[40] YOLO Performance Metrics. Online. Ultralytics. 2024. Dostupné z: https://docs.ultralytics.com/guides/yolo-performance-metrics/. [cit. 2024-05-07].

[41] REVINA, I.Michael a EMMANUEL, W.R. Sam. A Survey on Human Face Expression Recognition Techniques. Journal of King Saud University - Computer and Information Sciences. 2021, roč. 33, č. 6, s. 619-628. ISSN 1319-1578. Dostupné z: https://doi.org/https://doi.org/10.1016/j.jksuci.2018.09.002.

[42] ZHANG, Teng; DENG, Lirui; ZHANG, Liang a DANG, Xianglei. Deep Learning in Face Synthesis: A Survey on Deepfakes: A Survey on Deepfakes. In: 2020 IEEE 3rd International Conference on Computer and Communication Engineering Technology (CCET). Beijing, China: IEEE, 2020, s. 67-70. ISBN 978-1-7281-8811-9. Dostupné z: https://doi.org/10.1109/CCET50901.2020.9213159.

[43] NIRKIN, Yuval; MASI, acopo; TUAN, Anh Tran; HASSNER, Tal a MEDIONI, Gerard. On Face Segmentation, Face Swapping, and Face Perception. In: 2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018). Xi'an, China: IEEE, 2018, s. 98-105. ISBN 978-1-5386-2335-0. Dostupné z: https://doi.org/10.1109/FG.2018.00024.

[44] ULTRALYTICS. Https://docs.ultralytics.com/. Online. 2024. Dostupné z: https://docs.ultralytics.com/. [cit. 2024-05-07].

[45] Github - ultralytics. Online. Github. 2024. Dostupné z: https://github.com/ultralytics/ultralytics. [cit. 2024-05-07].

[46] PennFudanPed. Online. Kaggle. 2019. Dostupné z: https://www.kaggle.com/datasets/jiweiliu/pennfudanped. [cit. 2024-05-08].

[47] Head Detection (CCTV) Computer Vision Project. Online. Roboflow. 2023. Dostupné z: https://universe.roboflow.com/trisha-then/head-detection-cctv. [cit. 2024-05-08].

[48] KARTHIKA, N J a SARAVANAN, Chandran. Pedestrian Detection Data set. Online. Kaggle. 2020. Dostupné z: https://www.kaggle.com/datasets/karthika95/pedestrian-detection. [cit. 2024-05-08].

[49] Face/Head Segmentation Dataset Community Edition. Online. MUT1NY'S DEEP LEARNING STORE. Dostupné z: https://store.mut1ny.com/product/face-head-segmentation-dataset-community-edition?v=928568b84963. [cit. 2024-05-08].

[50] ZHAO, Xu, et al. Fast Segment Anything. arXiv preprint arXiv:2306.12156, 2023.

[51] ROSEBROCK, Adrian. Deep learning for computer vision with Python: ImageNet bundle. PyImageSearch, 2017. ISBN 978-1-986723848.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

ADAM	Adaptive Moment Estimation
AI	Artificial Intelligence
ANN	Artificial Neural Network
CCTV	Closed-Circuit Television
CPU	Central Processing Unit
FLOPS	Floating point Operations Per Second
HOG	Histogram of Oriented Gradients
mAP	Mean Average Precision
ONNX	Open Neural Network Exchange
RCNN	Region-based Convolutional Neural Network
RELU	Rectified Linear Unit
SPP	Spatial Pyramid Pooling
SPM	Spatial Pyramid Matching
SSD	Single Shot Detector
SVM	Support Vector Machine
VGG	Visual Geometry Group
TPU	Tensor Processing Unit
YOLO	You Only Look Once

SEZNAM OBRÁZKŮ

Seznam tabulek

SEZNAM PŘÍLOH

PŘÍLOHA P I: CD NOSIČ

CD obsahuje tyto soubory

bp/

prilohy/

detection_models/

- detect.pt
- detect.onnx

prilohy/segmentation_models/

- segment.pt
- segment.onnx

implementation.py