VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

APLIKACE PRO AUTOMATICKÉ VYHODNOCENÍ VĚROHODNOSTI GENEROVANÉHO SNÍMKU OBLIČEJE

APPLICATION FOR AUTOMATIC EVALUATION OF THE FIDELITY OF THE GENERATED FACIAL IMAGE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **JIŘÍ ŠOTOLA**
AUTHOR

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

BRNO 2024

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání bakalářské práce

162135

Ústav:	Ústav inteligentních systémů (UITS)
Student:	**Šotola Jiří**
Program:	Informační technologie
Název:	**Aplikace pro automatické vyhodnocení věrohodnosti generovaného snímku obličeje**
Kategorie:	Zpracování obrazu
Akademický rok:	2023/24

Zadání:

1. Seznamte se s problematikou generování syntetických snímků obličeje a zjistěte, jaké jsou dostupné generované datasety se snímky obličeje.
2. Sumarizujte informace o alespoň 3 algoritmech pro rozpoznávání osob podle obličeje.
3. Navrhněte řešení, které bude určovat věrohodnost generovaných snímků obličeje.
4. Navržené řešení implementujte v programovacím jazyce Python.
5. Aplikaci otestujte a vyhodnoťte, čím je věrohodnost generovaných snímků obličeje nejvíce ovlivňována.

Literatura:

- CHOI, Yunjey, et al. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018. p. 8789-8797.
- DENG, Jiankang, et al. Arcface: Additive angular margin loss for deep face recognition. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019. p. 4690-4699.

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

Vedoucí práce:	**Goldmann Tomáš, Ing., Ph.D.**
Vedoucí ústavu:	Hanáček Petr, doc. Dr. Ing.
Datum zadání:	1.11.2023
Termín pro odevzdání:	31.7.2024
Datum schválení:	10.7.2024

Abstrakt

Bakalářská práce se zaměřuje na návrh a implementaci aplikace pro ověření věrohodnosti synteticky generovaných snímků. Z důvodu rozlehlosti tématu se práce zaměřuje na ověření podobnosti obličejových znaků originálního snímku a snímku z něj vygenerovaného. Pro aplikaci je vyvinut model založený na siamských neuronových sítích, které používají ztrátovou funkci contrastive loss. Model byl trénován a testován na datové sadě LFW, kde dosáhl přesnosti až 91 %. Pro testování generovaných snímků je použit model StarGAN, který generoval snímky obličeje se změnou barvy vlasů, pohlaví a stáří. Výsledné testování na generovaných snímcích ukázalo, že model StarGAN vytváří obličeje, které se s originálem shodují z 87,53 %.

Abstract

The bachelor's thesis focuses on the design and implementation of an application for verifying the authenticity of synthetically generated images. Due to the vastness of the topic, the work concentrates on verifying the similarity of facial features between the original image and the generated image. For the application, a model based on Siamese neural networks, which uses the contrastive loss function, was developed. This model was trained and tested on the LFW dataset, where it achieved an accuracy of up to 91 %. For testing the generated images, the StarGAN model was used, which generated facial images with changes in hair color, gender and age. The resulting tests on the generated images showed that the StarGAN model creates faces that match the original ones by 87.53 %.

Klíčová slova

neuronové sítě, neuron, perceptron, siamská síť, rozpoznávání obličeje, GANs, StarGAN, RelGAN, ArcFace, MagFace

Keywords

neural networks, neuron, perceptron, siamese network, facial recognition, GANs, StarGAN, RelGAN, ArcFace, MagFace

Citace

ŠOTOLA, Jiří. *Aplikace pro automatické vyhodnocení věrohodnosti generovaného snímku obličeje*. Brno, 2024. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Tomáš Goldmann, Ph.D.

Aplikace pro automatické vyhodnocení věrohodnosti generovaného snímku obličeje

Prohlášení

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

.....
Jiří Šotola
31. července 2024

Poděkování

Touto cestou bych chtěl poděkovat svému vedoucímu doktorovi Tomáši Goldmannovi za poskytnutí cenných rad a za pomoc při psaní mé bakalářské práce.

Obsah

Seznam obrázků

Kapitola 1

Úvod

V poslední době došlo k výraznému rozvoji počítačových systémů, které se dělí do mnoha odvětví. Jedno z těchto odvětví se zaměřuje na tvorbu snímků obličeje, například tvorbou úplně nových snímků nebo tvorbou upravených snímků. Obecně se jedná o velmi různorodý a komplikovaný úkol, který by měl splňovat potřebná kritéria pro dosažení kvalitního výstupu. Kritéria jsou rozčleněna na jednoduché potíže, které jsou z lidské perspektivy snadno viditelné. Například obličej má své charakteristické rysy. Pro počítač však představuje tento úkol komplexní problém, který nelze jednoduše vyřešit. Z tohoto důvodu jsou navrhovány nové systémy pro řešení výše uvedených úskalí.

Těchto systémů existuje nezměrné množství a každý má stanovené cíle. Například tvorba náhodného obličeje, který se může použít při prezentaci nového vzhledu sociálních sítí. Užitečnost je především v tom, protože obličej nemůže být spojen s reálnou osobou. Fiktivní obličeje mohou působit více objektivněji a není nutné získávat žádné dokumenty o souhlasu s použitím osobních údajů. Dalším příkladem je úprava obličeje hledané osoby pro její snadnější identifikaci. Pro tento úkol jsou vytvořeny kolekce snímků s různými styly vlasů nebo výrazy, které mohou pomoci odhalit rozdíly ve vzhledu. Vygenerované snímky mohou být užitečné nejen pro policii, ale také pro různé modely, které se na nich mohou učit.

V těchto případech je nezbytné ověřovat podobnost mezi originálním a vygenerovaným snímkem obličeje. Je důležité, aby obličej zůstal nepoškozený pro správnou identifikaci. Bohužel některé modely tento aspekt zanedbávají, protože se nezaměřují na kontrolu obličejů. Jejich největší kontrola spočívá v použití dotazníků pro skupinu lidí, kteří hodnotí snímek jako celek, a nikoli jeho přesnou shodu s originálem.

Cílem práce je vyvinout aplikaci v programovacím jazyce *Python* pro vyhodnocení věrohodnosti syntetických snímků generovaných z reálných snímků. Aplikace je schopna ověřit podobnost identit na snímcích a poskytuje jednoduchý výstup ve formě pravdivostní hodnoty. Další její funkcí je skupinové zpracování, při kterém se využívá struktur jmen a adresářů k analýze širokého množství snímků, přičemž výsledky jsou zobrazeny na výstupu. Výstup může být jednoduchý výpis do terminálu nebo jako extrakce do souboru.

Pro naplnění výše uvedených cílů je nutné mít informace a znalosti, které začínají generováním syntetických snímků, viz kapitola 2. Pro podrobnější pochopení a realizaci funkcí modelů jsou zde popsány generativní soupeřící sítě, které jsou konstruovány z neuronových sítích. V kapitole 3 je uvedeno rozpoznávání osob podle obličeje společně s důležitými datovými sadami. Následuje kapitola Návrh a implementace 4, kde je uveden popis vyvinutého modelu a jeho zpracování do konzolové aplikace spolu s přípravou datových sad. Výsledný program je testován v kapitole Testování a experimenty 5, kde je model porovnáván s jiným modelem na reálných snímcích a následně je testován na generovaných snímcích.

Kapitola 2

Generování syntetických snímků obličeje a neuronové sítě

Generování syntetických snímků obličeje je proces, při kterém se pomocí strojového zpracování vytvářejí nové obličejové snímky nebo modifikují již existující. Tento proces využívá strojové učení k získání znalostí o obličejových znacích z reálného světa, což umožňuje dosažení větší věrohodnosti a realističnosti výsledných snímků [31].

Způsobů generování syntetických obličejů je ohromné množství například generování náhodných obličejů, generování snímků se změněným výrazem či vytvoření 3D obrazu ze 2D snímku obličeje s následným otočením a vytvořením nového snímku [10, 12, 36].

V této kapitole je uveden základní popis neuronových sítí viz. sekce 2.1, které reprezentují technologii strojového učení. Dále v sekci 2.2 je možné nalézt vysvětlení jednoho podtypu neuronových sítí tj. generativní soupeřící sítě. Jedná se o generátory, které modifikují obličeje s nejvyšší věrohodností a přesností. Základem pro generativní soupeřící sítě jsou dvě neuronové sítě, které fungují na principu hry s nulovým součtem. Popsaný způsob pomáhá ke vzniku kvalitnějších snímků [10]. V závěru kapitoly jsou uvedeny algoritmy pro generování syntetických snímků obličeje viz. sekce 2.3.

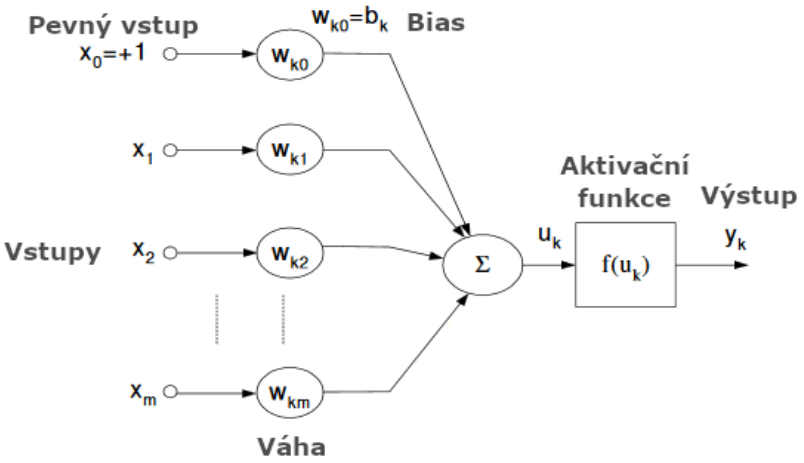
2.1 Neuronové sítě

Neuronové sítě jsou založené na sofistikovaném fungování lidského mozku, kde stovky miliard neutronů jsou paralelně spojeny mezi sebou za účelem zpracování určité informace. Tato funkcionalita se vědcům podařila do určité míry napodobit a přenést do počítačového světa. Neuronové sítě nám pomáhají s řešením složitějších problémů jako například u překladu jazyků nebo rozeznávání různých vzorů ve snímku, Nicméně, stále se nedaří úspěšně napodobit lidské emoce a vědomí [37].

Tato podkapitola se zabývá neuronovými sítěmi ze zdrojů [37, 26]. V sekci je uveden krátký popis neuronových sítí (jsou zde vysvětleny pojmy perceptron, aktivační funkce a princip trénování neuronové cítě). Dále je představena konvoluční neuronová síť, která je speciálním typem neuronové sítě. Na závěr je popsána siamská neuronová síť, která se specializuje na porovnávání výstupů ze stejných neutronových sítí.

Perceptron

Perceptron je základní jednotkou v neuronových sítích, která je definovaná jako matematický model biologického neuronu. Umělý model perceptronu se skládá z množiny vstupů, vah, *bias*, aktivační funkce a právě jednoho výstupu, který je vyobrazen na obrázku 2.1.



Obrázek 2.1: Model perceptronu [16].

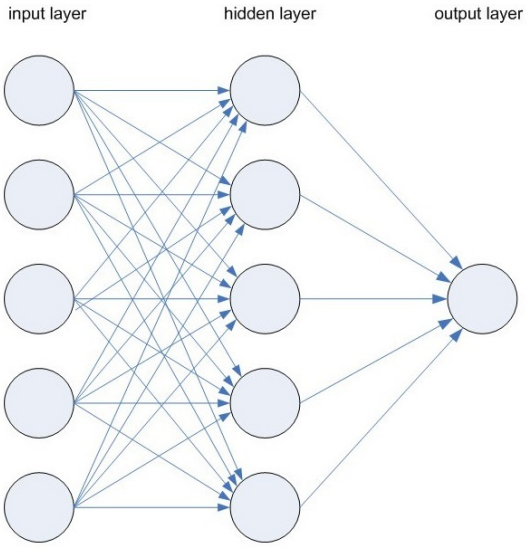
Každý vstup se skládá z příchozí hodnoty. Tato hodnota je po vstupu do perceptronu vynásobena jeho přiřazenou váhou. Dále se v perceptronu všechny vynásobené hodnoty sečtou a přičte se hodnota zvaná *bias*. V dalším kroku se vloží sumarizovaná hodnota do aktivační funkce a výsledek se předá na výstup perceptronu. Celkový mechanismus je poté možné popsat vzorcem 2.1.

$$y_k = f(b_k + \sum_{i=1}^{n} x_i w_{ki}) \tag{2.1}$$

Tento vzorec znázorňuje x_n jako vstupní hodnoty, w_{ki} jako váhy jednotlivých vstupů a y_k jako výstupní hodnotu. b_k představuje *bias* a k označení perceptronu. n reprezentuje počet vstupů a f je aktivační funkce, jejíž podrobnější popis je uveden níže [37, 16].

Struktura

V neuronových sítích se perceptrony skládají do určitých vrstev k řešení složitějších problémů. Každá vrstva se skládá z množiny perceptronů, které jsou pomocí vazeb napojeny na ostatní perceptrony předchozí vrstvy a podle typu sítě i mezi sebou. Příklad je vyobrazen na obrázku 2.2 s třívrstvou neuronovou sítí [19]. Vrstvy poté dělíme na vstupní vrstvu, skryté vrstvy a výstupní vrstvu. Vstupní vrstva slouží k přijetí dat ze vstupu a distribuci do skryté vrstvy. Skryté vrstvy mají za úkol zpracovat data. Dosahují toho pomocí úpravy *bias* a vah během fáze učení, aby byla zajištěna správná funkce sítě. Skrytých vrstev může být více než jedna, aby se zlepšila kvalita dat před jejich předáním do výstupní vrstvy [37]. Výstupní vrstva pak zobrazuje výsledky pomocí hodnot perceptronů. Z výsledků mohou vzniknou kategorie, predikce, pravděpodobnosti a i snímky.



Obrázek 2.2: Třívrstvá neuronová síť [19].

Aktivační funkce

Aktivační funkce je část perceptronu, která určuje jeho výstup. Má za úkol transformovat hodnoty podle předem vybrané matematické funkce. Z důvodu velkého množství různých matematických funkcí se do aktivační funkce vybírají funkce podle požadavků, které má perceptron splňovat a které budou mít pozitivní vliv na architekturu sítě [30].

Aktivačních funkcí je nezměrné množství. Jedna z nejjednodušších matematických funkcí je binární skoková funkce. Binární skoková funkce, transformuje výstup na hodnotu 0 do určité vstupní hodnoty a nad tuto hodnotu ji začne transformovat na hodnotu 1. Další jednoduchá matematická funkce je lineární funkce. Lineární funkce hodnoty nezpracovává, ale okamžitě je odesílá na výstup. Mezi nejznámější a nejzákladnější aktivizační funkce patří *sigmoid* [9], *ReLu* [6], *tanh* [30] a *softmax* [8]. K těmto funkcím existují ještě jiné varianty jako *leaky ReLu* [30], které kompenzují nevýhody původních funkcí nebo rozšiřují jejich funkčnost. V případě *leaky ReLu* jde o opravení transformace u záporných čísel z původní funkce *ReLu*, která se v některých případech mohla transformovat všechny hodnoty pouze do 0.

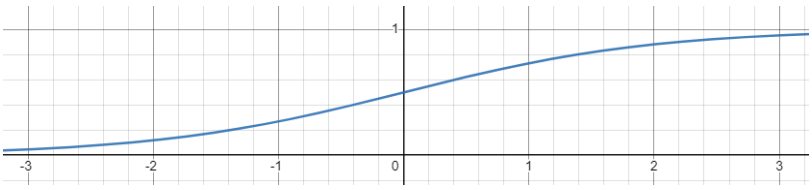
Sigmoid

Sigmoid je typ aktivační funkce, která mapuje všechny vstupní hodnoty v podobě reálných čísel do intervalu od 0 do 1. Funkce je hojně využívána ve statistických modelech z důvodu jejího mapování do intervalu $(0, 1)$. Funkce je daná předpisem 2.2.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.2}$$

Podle obrázku 2.3 mapování hodnot lze rozdělit do tří částí. První část, která zahrnuje vstupní hodnoty menší než -3, lze ohodnotit hodnotou 0, protože funkce nabývá hodnot extrémně blízkých k 0. V intervalu od -3 do 3 se jedná téměř o lineárně rostoucí křivku. Poslední část, která zahrnuje hodnoty větší než 3, lze ohodnotit jako 1, ze stejného důvodu jako první interval. Z hlediska informatiky není tato funkce moc efektivní, protože

funkce je náročná na výpočet. Současně je nevhodná pro některé hodnoty, protože je velmi citlivá pří přechodu hodnoty 0,5, které odděluje záporné a kladné vstupní hodnoty. Pozitivně vnímanou vlastností funkce je mapování odlehlých hodnot. Jedná se o hodnoty, které jsou extrémně vysoké nebo extrémně nízké. Funkce je namapuje je na hodnoty blízké 1 či 0. Pro tyto vlastnosti funkce se používají primárně u výstupní vrstvy [9].

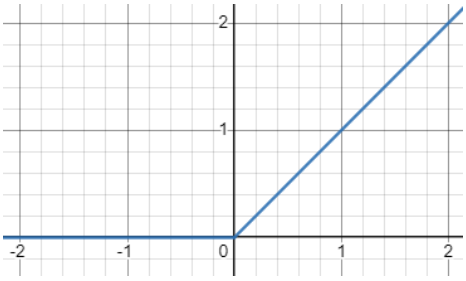


Obrázek 2.3: Aktivační funkce *sigmoid*.

ReLU

Uvedená funkce je jedna z nejpoužívanějších z důvodu její jednoduchosti a snadné předvidatelnosti. Její průběh je znázorněn v grafu 2.4, kde je možné vidět, že definiční obor hodnot (vstupní hodnoty) jsou všechna reálná čísla, ale obor hodnot (výstupní hodnoty) jsou pouze v rozsahu reálných nezáporných čísel. Funkce zpracovaná všechny záporná čísla do 0, čímž mnohem snadněji předpovídá výstup. Nevýhodou této funkce je možnost zablokování perceptronu. Problém vznikne, pokud se při učení váhy nastaví tak, že jejich součet s jakýmkoliv vstupem je záporný nebo nulový. Následný výstup $ReLU$ je pak vždy 0. Tímto způsobem nelze ovlivňovat celkové hodnocení a ani změna vah není možná, protože při učení změna vah se spočítá pomocí vlivu na hodnocení a ten je vždy nulový (popis nastavování vah je uveden níže v sekci Trénování neuronové sítě). Další negativní vlastností je mapování kladných odlehlých čísel na rozdíl od *sigmoid*. Hlavní výhodou této aktivační funkce je její rychlost, náročnost na výpočet a poměrně rychlé učení. Primární využití zmíněné funkce je ve skrytých vrstvách. Funkce je dána tímto předpisem 2.3.

$$f(x) = \begin{cases} x & \text{pro } x \geq 0 \\ 0 & \text{jinak} \end{cases} \tag{2.3}$$



Obrázek 2.4: Aktivační funkce $ReLU$.

Softmax

Softmax je speciální aktivační funkce používaná u klasifikačních neuronových sítí v poslední plně propojené vrstvě. Tato funkce převádí výstupy perceptronů na relativní pravděpodobnosti, přičemž součet těchto pravděpodobností v každém perceptronu dané vrstvy je roven

jedné. Konkrétně zjistí hodnoty vstupující do aktivační funkce a výsledek je vypočítán jako hodnota vstupující do aktivační funkce děleno součtem hodnot vstupujících do aktivační funkce ve všech perceptronech [8].

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{2.4}$$

Je reprezentovaná funkcí u rovnice 2.4, kde K je počet perceptronů ve vrstvě a z_i je označení perceptronu.

Trénování neuronové sítě

Neuronové sítě jsou tvořeny velkým množstvím perceptronů, které jsou složené z trénovatelných vah a *bias*. Pro zvýšení kvality a přesnosti je nutné nastavit tyto váhy a *bias*. Jeden ze způsobů je nastavit je ručně. Tento způsob je velmi neefektivní a prakticky nemožný pro lidi, protože komplexnost některých řešení je neskutečně velká. Druhý způsob je učením, kde se používá nastavování vah a *bias* pomocí datových sad, které slouží jako příklady s výsledky. Celkově existují tři typy učení [37]:

1. učení s učitelem,
2. učení bez učitele,
3. posilované učení.

Učení s učitelem (*supervised learning*) funguje na principu známého vstupu a výstupu, kde je předem připravená datová sada s informacemi o vlastnostech vzorků. Neuronová síť se pak učí tak, že vezme vstupní vzorek, zpracuje ho, výsledek porovná s předpokládaným výsledkem a na závěr upraví váhy neuronové sítě. Toto je v základu *back-propagation*, která jsou popsané níže. Pro správnou funkcionalitu je potřeba ztrátová funkce, které kvantifikuje, jak dobře model funguje a vede proces optimalizace k dosažení co nejlepších výsledků.

Učení bez učitele (*unsupervised learning*) je založeno na principu, že nejsou poskytnuty žádné vlastnosti navíc jako s učitelem. Jsou poskytnuty parametry například: kolik má mít shluků, do kterých třídit vstupy. Hlavním principem je vložení velkého množství dat, které se neuronová síť snaží roztřídit do shluků. Tento princip je velmi praktický, ale je složitější na tvorbu, protože potřebuje víc dat na učení.

Poslední typem je posilované učení (*reinforcement learning*). Posilované učení získává nějaké o správnosti modelu, ale na rozdíl od učení s učitelem jsou získané jinak. Odlišuje se tím, že neuronové sítě ohodnocují své akce na základě penalizací nebo odměn získaných od vnějších ohodnocený koncových stavů. Cílem je získat, co největší odměny skrze proces pokus a omyl.

Back-propagation

Tento algoritmus se skládá ze tří fází. První fáze je *Forward Pass*, kde neuronové sítě počítají výsledek pomocí vah a *bias*. Výsledek je poté číselný vektor, který je porovnán ve ztrátové funkci (podrobnější popis níže) za účelem vypočítání chyby mezi predikovaným a požadovaným výstupem. V druhé fázi zvané *Backward Pass* se propaguje vypočtená chybovost do *gradientu ztráty* v každém perceptronu směrem od výstupu ke vstupu. V poslední fázi *Weight Update* se nastaví váhy a *bias* pomocí *gradientu ztráty*. Tento typ nastavování vah

a *bias* v *back-propagation* se jmenuje gradientní sestup (*gradient descent*). Proces se opakuje, dokud není hodnota chyby dostatečně nízká [37].

Ztrátová funkce

Ztrátová funkce neboli *loss function* je funkce, která vypočítá odchylku výsledku neuronových sítí a požadovaného výsledku pro každý výstup neuronové sítě. Ztrátová funkce slouží pro hodnocení kvality neuronových sítí, výpočet chybovosti a zpětného trénování.

Cílem trénování je minimalizovat hodnotu ztrátové funkce. Minimalizace ztrátové hodnoty funkce se obvykle dělá pomocí optimalizačních algoritmů například pomocí metody gradientní sestup, kde se upravují váhy a *bias* tak, aby se snížila ztrátová funkce [37].

Jedním z příkladů ztrátové funkce je *Binary Cross-Entropy Loss* [41]. Tato funkce předpovídá pravděpodobnost příslušnosti do jedné ze dvou skupin na základě vloženého vzorku. K tomuto účelu je v poslední vrstvě aktivační funkce *Sigmoid* [8], která pomocí ztrátové funkce určuje pravděpodobnost, do které skupiny patří. Je popsána rovnicí 2.5, kde p je pravděpodobnost, že budoucí vstup bude patřit do jedné skupiny a y je označení pro vstup.

$$L = -(y \log(p) + (1 - y) \log(1 - p)) \tag{2.5}$$

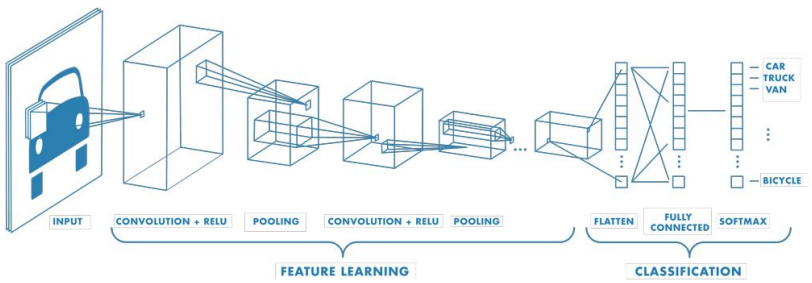
Konvoluční neuronové sítě

Konvoluční neuronové sítě jsou jedním z typů neuronových sítí. Tyto sítě se specializuji na zpracovaní obrazu pomocí konvoluce, ale jinak se chovají jako základní neuronové sítě [7]. Sítě jsou složené z:

- vstupní vrstvy,
- alespoň z jedné konvoluční vrstvy,
- libovolného množství *pooling* vrstev,
- libovolného množství plně propojených vrstev,
- výstupní vrstvy.

Standardně jsou tvořeny ve skupinách, kde se konvoluční a *pooling* vrstvy střídají za sebou. Za nimi následují plně propojené vrstvy. Tuto strukturu je možné vidět na obrázku 2.5 [29].

Popis jednotlivých vrstev je uveden níže včetně modelu konvoluční sítě *ResNet-50*.

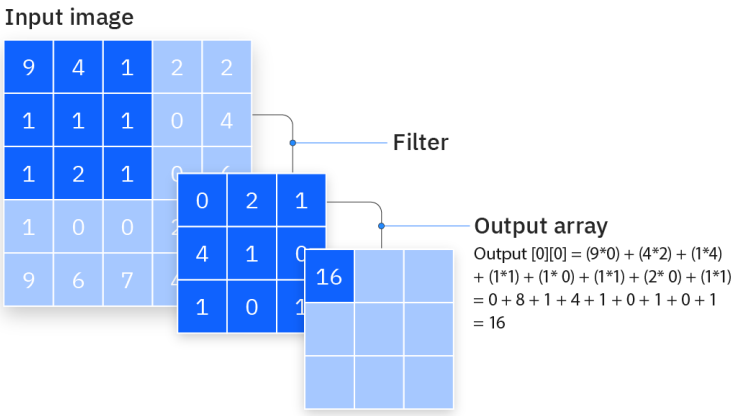


Obrázek 2.5: Struktura konvoluční neuronové sítě [23].

Konvoluční vrstva

Konvoluční vrstva pracuje na principu matematické konvoluce, kde pomocí filtrů s trénovatelnými koeficienty dokáže zvýraznit různé vzory (hrany, čáry, atd.) [29].

Vstupem této vrstvy jsou vstupní data a filtry ve 2D maticích a výstup je 2D vrstva zvaná příznaková mapa (*feature map*). Z příznakové mapy je možné pomocí intenzity odstínu zjistit výsledek konvoluce jako na příkladovém obrázku 2.6.



Obrázek 2.6: Příklad konvoluce obrázku s filtrem [20].

Konvoluce je znázorněna výrazem 2.6.

$$G[i,j] = X * H = \sum_{k} \sum_{l} X[k,l] \cdot H[i-k, j-l] \tag{2.6}$$

kde X je vstupní pole 2D hodnot, H je vstupní pole filtru, G je výstupní pole filtru, i a j jsou pozice právě vypočítávaného bodu.

Pooling vrstva

Pooling vrstva [29] se nachází za vrstvou konvoluční. Slouží k snížení prostorové dimenze vstupu tím, že určité skupiny hodnot agregují do jedné hodnoty. Způsob agregace je možný ovlivnit povolovacím polem. Vstupem *pooling* vrstvy je matice zvané příznaková mapa z konvoluční vrsty a výstup je zmenšená příznaková mapa podle snížení dimenze.

Tato vrstva má dva typy: *max pooling* a *average pooling*. První typ *max pooling* vrací maximální hodnotu z agregované oblasti a druhý typ *average pooling* vrací průměr hodnot z předané oblasti, kde oblast je pole vstupních hodnot [28].

Flatten vrstva

Flatten vrstva [28] slouží ke změně topologie výstupu předchozí vrstvy do jednorozměrného složeného pole, které pak může být přiváděno do následujících plně propojených vrstev. Obvykle se objevuje po konvolučních a *pooling* vrstvách. Pokud má například vstup do vrstvy rozměry (výška, šířka, kanály), *flatten* vrstva jej přetvoří na oblast o velikosti danou šířkou, výškou a potom kanálem [28].

Batch Normalization vrstva

Jednou z důležitých vrstev konvoluční neuronové sítě je *Batch Normalization* [28], která se specializuje na normalizaci hodnot a stabilizaci modelu. Výpočet normalizace je uskutečněn skrze skupinky vzorků (*batch*), kde je spočítán průměr μ a směrodatná odchylka σ. Tyto hodnoty se přivedou do vzorce pro normalizaci $x_n = \frac{x_i - \mu}{\sigma}$. Normalizované hodnoty se poté ještě vynásobí, aby se měřítko škálovalo a na závěr se přičte koeficient, který souží k posunutí osy. Koeficienty pro násobení a sčítání se pak nastavují během trénování.

Nevýhodou *Batch Normalization* vrstvy je, že se nechová stejně při učení a poté při testování, protože používá jiný výpočet. Tento výpočet je ovlivněn průměrováním hodnot. Při učení se používá průměr ze dávky vzorků, ze kterých je učena. Při testování se používá pohyblivý průměr ze vstupních hodnot a z naučených při učení. Tento princip umožňuje ustálit hodnoty na výstupu, pokud přijde vysoký nebo nízký extrém z předchozí vrstvy [28].

Plně propojená vrstva

Tato vrstva se dává na konec konvolučních neuronových sítí. Standardní počet vrstev je dvě a více. Slouží jako základní neuronová vrstva na učení ze vstupu, kde v každé vrstvě jsou všechny výstupy z předchozí vrstvy spojeny do každého perceptronu ve stávající vrstvě. Tyto perceptrony poté nastavují váhy a *bias* podle učení.

Dropout vrstva

Dropout vrstva [28] používá regularizační techniku, která zahrnuje náhodné ignorování nebo vypouštění některých výstupů vrstvy během tréninku, aby se zabránilo *overfitting*. K *overfitting* dochází, když se model nemůže už více zobecnit a místo toho se příliš přizpůsobí trénovací datové sadě. Tréning pak vypadá tak, že vrstva náhodně deaktivuje vybraný podíl perceptronů (a jejich spojení) z předchozí vrstvy. Tím je v podstatě dočasně odstraní ze sítě. Deaktivované perceptrony jsou vybrány náhodně pro každou iteraci tréninku [28].

Deaktivací vybraných perceptronů dochází ke snížení výstupu vrstvy, aby došlo k zachování výstupu, je nutné výstupy zbývajících aktivních neuronů zvětšit o faktor rovný pravděpodobnost udržení aktivního perceptronu. To znamená, pokud se deaktivuje 50 % perceptronů, zbývající aktivní perceptrony se vynásobí hodnotou 2. Nevýhoda této vrstvy je, že přidává čas potřebný pro trénování [28].

ResNet-50

ResNet-50 [17] je typ konvoluční neuronové sítě, která byla navržena pro zlepšení výkonu v úlohách počítačového vidění, jako je klasifikace obrázků, detekce objektů a segmentace. Klíčovým prvkem *ResNet-50* je koncept residualních bloků (*residual block*), který pomáhá při trénování velmi hlubokých sítí.

Reziduální bloky jsou základní stavební jednotky architektury *ResNet*. Každý reziduální blok obsahuje jednu nebo více konvolučních vrstev, ale místo přímého výstupu z těchto vrstev se k výstupu přidá reziduální spojení. Toto spojení přenáší originální vstup z bloků přímo na jejich výstup, čímž umožňuje, aby se gradienty během trénování efektivněji propagovaly zpět. Tento přístup zmírňuje problém vymizení gradientu, kde by gradienty při zpětném šíření mohly být velmi malé a tím bránily učení v hlubokých vrstvách.

ResNet-50 je architektura hluboké neuronové sítě, která se skládá z 50 vrstev. Architektura začíná konvoluční vrstvou, která používá 7×7 konvoluční filtr s 64 filtry. Tato vrstva

je následována *max pooling* (3×3), což redukuje prostorové rozměry obrázku. Další částí ResNet-50 jsou residualní bloky, které jsou organizovány do čtyř skupin. Každá skupina obsahuje residualní bloky s konvolučními vrstvami o rozměrech 1×1, 3×3 a 1×1. První skupina obsahuje tři residualní bloky. Druhá skupina zahrnuje čtyři bloky. Třetí skupina má šest bloků. Poslední skupina obsahuje tři bloky .

Celkově je *ResNet-50* považován za velmi úspěšný model díky své schopnosti efektivně trénovat hluboké sítě a poskytovat vysokou úroveň přesnosti v různých úlohách [17].

Siamská neuronová síť

Siamská neuronová síť je další typ neuronové sítě, která se specializuje na porovnávání výstupů ze stejných neutronových sítí [3].

Obecně se skládá ze dvou a nebo více stejných podsítí, které produkují výstupní vektory (*feature vector*). Tyto vektory jsou následně předány do vrstvy, která spočítá jejich vzdálenosti. Tato vzdálenost umožňuje zjistit odlišnosti. Tento způsob je možné použít u dvou podsítí, ale je nutné prvně analyzovat data na zjištění intervalu vzdáleností a mezního bodu na jejich rozlišení. U třech podsítí je příklad v porovnávaní chtěného snímku s jedním odlišným a jedním podobným. Pokud vstup má menší vzdálenost s podobným snímkem, tak je vstup podobný, pokud to je naopak, tak je odlišný.

Použité podsítě u všech těchto typů musí splňovat určité podmínky. Hlavní z nich je, že musí mít stejnou architekturu a sdílet konfiguraci *bias* a vah tak, aby pro stejné snímky vytvořily vždy stejné vektory nehledě na podsíti. Jedním z příkladů jsou sítě pro verifikaci podpisů či podobnosti obličejů.

Používané ztrátové funkce v siamských sítích jsou primárně *Triplet loss* a *Contrastive loss* (jejich popis je uveden níže). Avšak používají se i jiné ztrátové funkce, ale často nejsou tak přesné [3].

Triplet loss

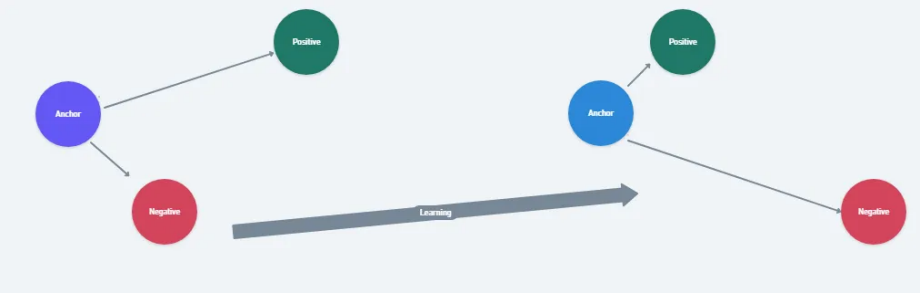
Triplet loss [3] funguje na principu porovnání tří vstupů, kde jeden je zakotvený jako výchozí bod, druhý je pozitivní (je podobný) a třetí je negativní (je odlišný). Cílem modelu je zmenšit vzdálenosti od pozitivního vstupu a od negativního vstupu se co nevíce oddálit. Uvedený postup je znázorněn na obrázku 2.7. Toto fungování pomáhá snížit vzdálenosti mezi stejnými třídami vstupů a oddálit rozdílné třídy vstupů. Nicméně tímto postupem se výpočet pro funkci stane náročnější.

$$L\left(A, P, N\right) = \max\left(\| \mathrm{f}\left(A\right) - \mathrm{f}\left(P\right) \|_2 - \| \mathrm{f}\left(A\right) - \mathrm{f}\left(N\right) \|_2 + \alpha, 0\right) \tag{2.7}$$

Ztrátová funkce je definovaná výrazem 2.7, kde $f(x)$ je funkce, která zpracovává vstupy a vytváří z nich pole vektorů, α je nastavitelný *bias*, přičemž poslední trojice A, P a N jsou vstupy znázorňující výchozí bod, pozitivní vstup a negativní vstup [3].

Contrastive loss

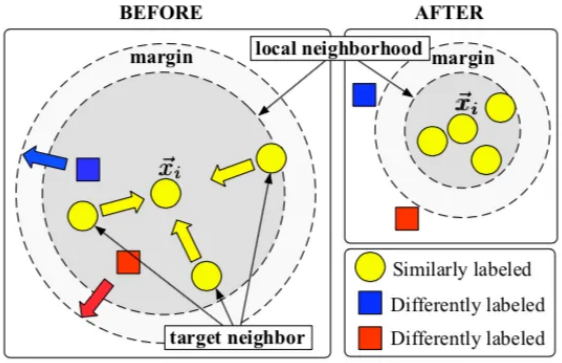
Contrastive loss [3] je ztrátová funkce, která porovnává pouze dva vstupy. První je zakotvený a má funkcionalitu výchozího bodu. Druhý vstup je variabilní bod na porovnání. Cílem této funkce je zmenšit vzdálenost s podobnými snímky a zvětšit ji u odlišných. Tento úkol je znázorněn na obrázku 2.8. *Contrastive loss* je poté zapsaná výrazem 2.8, kde y_{pred} je



Obrázek 2.7: Obrázek ztrátové funkce *Triplet loss* [3].

vypočítaný výsledek z modelu, y_{true} je pravdivý výsledek a m velikost rozmezí mezi nimi [3].

$$L = y_{true} \cdot y_{pred}^2 + (1 - y_{true}) \cdot (max(m - y_{pred}, 0))^2 \tag{2.8}$$



Obrázek 2.8: Obrázek ztrátové funkce *Contrastive loss* [38].

2.2 Generativní soupeřící sítě

Generativní soupeřící sítě anglicky *Generative Adversarial Networks* (GANs) jsou model neuronových sítí, které fungují na speciálním typu učení pro zvýšení jejich kvality. Jak již jméno napovídá, tak je model složen ze sítí, které soupeří mezi sebou s cílem být lepší než druhá s tím, že jedna získá na začátku vzor a jiná se ho snaží napodobit. Tyto sítě se postupně začnou vylepšovat mezi sebou než se dostanou k limitům jejich architektury. Generativní sítě se obecně skládají ze dvou hlavních složek – generátor a diskriminátor.

Tato sekce je ze zdrojů [21] a [5].

Generátor

Generátor je část modelu GANs, která se stará o generování falešných snímků za účelem se zlepšit a získat kvalitnější výstup. Toho docílí při projití dostatečného množství pokusů s ohodnocením ověřovacího prvku – diskriminátor.

Prvním výstupem je vždy náhodný šum, který se pomocí vah upravuje tak, že zašle snímek soupeřící síti neboli diskriminátoru a ten jí vrátí číselnou hodnotu neboli skóre,

jak moc špatně dopadl jeho výstup. Generátor poté vezme odeslaný snímek, vypočítá hodnotu, o jakou se váhy musí posunout pro každý výstup a propaguje ho zpět do sítě ať je lepší nebo horší než minulý snímek. V tom hraje roli diskriminátor, protože čím lepší bude výsledek, tím vetší bude posunutí vah a naopak čím horší bude výsledek, tím menší bude poté posunutí vah. V rámci trénovacího procesu je cílem zlepšovat neuronové sítě, nicméně při některých vzorcích může docházet ke zhoršování výkonosti.

Uvedená síť má vnitřní proměnnou, která se nastavuje náhodně při každém spuštění a slouží pro získání variability výstupu a možného zlepšení. Každá iterace tento vstup jemně omezí cestou na výstup pomocí vah. Obecně funguje na principu pokusu a omylu.

Generátor nepracuje vždy pouze s náhodným šumem, ale na vstupu může mít i snímek. Z tohoto důvodu se do generátorů přidávají i jiné sítě, které se starají o analýzu a o úpravu snímků. Jedním z příkladů je, že první síť zpracovává a analyzuje snímek, a následně hledá objekty, které má změnit. Druhá síť vygenerované objekty nahradí, nebo jinak změní a zakomponuje je do snímku.

Tento typ generátorů nemusí mít jen jeden vstup se snímkem, ale muže mít i vstup s vektory, ve kterých je uloženo, do jakého typu se má snímek změnit. Popsaný princip umožní zpracovat více typů změn do jednoho modelu.

Generátory požívají technologie neuronových sítí, kde jednou z používaných ztrátových funkcí je *Binary Cross-Entropy loss*. V případě analyzační sítě jsou použity konvoluční neutronové sítě, které byly prezentovány výše.

Diskriminátor

Diskriminátor je část modelu GANs, která se stará o kontrolu celého modelu. Diskriminátorů je široké množství a v jednom modelu GANs se může jich použít více pro dosažení lepších výsledků.

Diskriminátor musí umět kontrolovat výstup, zda splňuje dané požadavky. Z toho důvodu při první inicializaci je diskriminátor naučen, jak požadované snímky vypadají, a kterým má dávat lepší hodnocení.

V případě generování snímků, generátor vždy v počátku vygeneruje náhodný šum, který přijde k diskriminátoru. Diskriminátor náhodný šum musí ohodnotit podle snímků, ze kterých se učil. Skóre je často od nuly do jedné. Daným hodnocením říká, že se s tímto snímkem generátor zmýlil s určitou pravděpodobností. Může docházet k vyhodnocování celku, nebo jednotlivých stavů výstupu. U snímku to může být například pixel. Je-li generátor už na vysoké úrovni, tak se diskriminátor začne mýlit. V toto případě diskriminátor ví, že vše, co přijde od generátoru, je vygenerované, a tím ho bere jako falešný snímek, který ohodnotil špatně. Tím to může zjistit svoji chybu a změnit své hodnocení. Také se tímto může učit i ze vstupních snímků u generátorů, které je požadují jako vstup. Tyto snímky se poté dají použit pro zlepšeni určitých typů diskriminátorů.

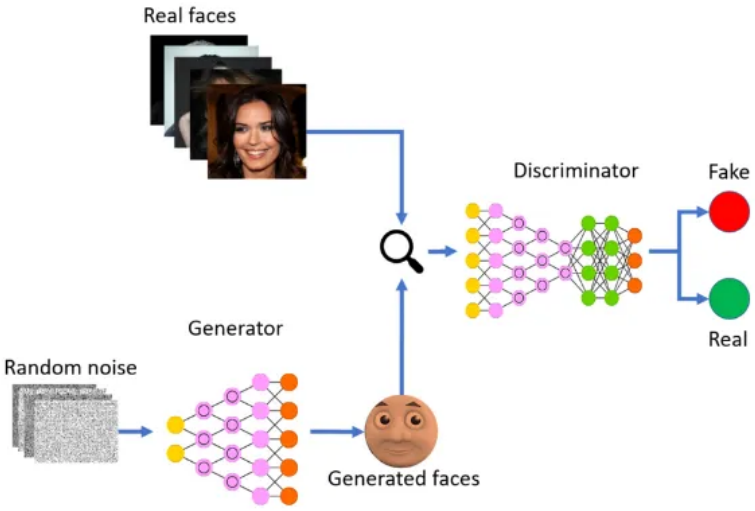
Jednu z kontrol, které by měl diskriminátor provádět, je kontrola realističnosti snímků nebo podoby k originálu. Pokud generátor nemá žádný vstup, tak diskriminátor musí umět prakticky jen dva typy kontrol. To jsou, zda snímek splňuje typ, který má mít na výstupu, a zda je realistický, protože pokud generátor vytváří ze začátku z větší části snímek náhodně, tak je velká možnost, že přijde s něčím novým. Pokud se poukáže na realitu, tak se musí počítat například s tím, že některé plochy jsou pouze značené jedním typem barev jako bělmo očí. Jednoduše řečeno je nutné diskriminátor naučit základní poznatky, co musí umět vyhodnotit.

U generátorů se vstupy je dobré kontrolovat podobnost výstupu se vstupem. Je důležité zachovat základní strukturu, která se nemá měnit, aby model zůstal podobný svému zdroji.

Pro diskriminátory se požívají často technologie konvolučních neuronových sítí, kde jednou z používaných ztrátových funkcí je *Binary Cross-Entropy loss*.

Struktura a učení

Generátor a diskriminátor jsou spojeny za sebou, jak je znázorněno v obrázku 2.9. Dohromady pracují tak, že generátor vygeneruje snímek. Následně vygenerovaný snímek zašle do diskriminátoru, který ho zpracuje a vyhodnotí shodnost vygenerovaného snímku s předlohou. Nesplňuje-li požadavky, tak se aktualizuje generátor pro zajištění lepšího generování. Pokud splní požadavky, tak je aktualizován diskriminátor, protože ho ohodnotil špatně. Tímto způsobeném se poté síť sama učí. Výše uvedený princip používá dva typy učení. U generátoru je použito posilované učení a u diskriminátoru je použité učení s učitelem. Oba typy učení byly popsány výše v sekci 2.1.



Obrázek 2.9: Model generativní soupeřící sítě [1].

2.3 Algoritmy pro generování syntetických snímků obličeje

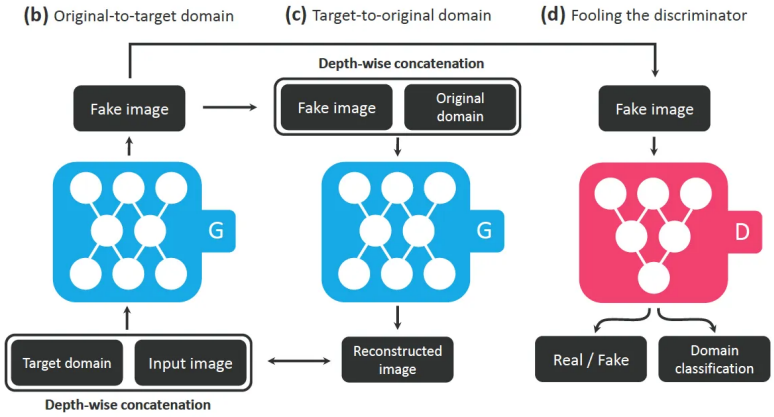
Modelů na generování syntetických snímků obličeje je už vytvořeno značné množství a každý byl vytvořen se specifickým cílem. V této sekci jsou představeny modely *StarGAN*, *RelGAN*, atd.

Další důležitou informací je využití pojmů doména a atributy ve specifickém významu. Atribut snímku představuje jeho vlastnost. Například u snímků s obličejem může být atributem barva vlasů. Hodnota atributu pak označuje konkrétní charakteristiku této vlastnosti. Například hodnoty atributu barva vlasů mohou být černé vlasy, hnědé vlasy a podobně. Doména je soubor snímků, které mají stejné hodnoty atributů.

StarGAN

StarGAN [12] je model pro generování syntetických snímků založený na generativní soupeřící síti. Generativní soupeřící síť obsahuje jeden generátor a jeden diskriminátor v podobě dvou konvolučních neuronových sítí. Jeho výsadou je transformace vstupních snímků na výstupní v cílové doméně. Typ cílové domény není daný pouze jednou doménou od jeho trénování, ale může změnit více domén najednou. Další z pozitivních vlastností je, že se dokáže učit z jedné datové sady se snímky o různých atributech místo použití více modelů, které by uměly změnu na zpět.

Funkci modelu *StarGAN* je možná shrnout to tří procesů. První je generování snímku se zadanou cílovou doménou. Druhý proces je porovnávání v diskriminátoru, který zjišťuje, zda je snímek reálný a v jaké doméně se nachází. Třetí proces je zpětné generování, kde vygenerovaný snímek je generován do originální domény pro zjištění zkreslenosti od originálního snímku. Celý tento proces je znázorněn na obrázku 2.10.



Obrázek 2.10: Model *StarGAN* [12].

Tento model se učí reálnosti vstupního snímku, správnému převedení do cílové domény a jeho zpětnou generací na penalizaci odlišností od originálu. Tyto úkoly jsou zpracovávány ztrátovou funkcí, která je zde rozdělena do několika jednodušších funkcí.

Adversarial Loss je funkce, která slouží pro rozeznaní reálnosti snímku. Je znázorněna rovnicí 2.9, kde generátor $G(x, c)$ generuje snímek x do cílové domény c a $D_{src}(x)$ je rozdělení pravděpodobnosti přes zdroje v diskriminátoru. Generátor se snaží hodnotu L_{adv} zmenšit a diskriminátor ji zvětšit.

$$L_{adv} = logD_{src}(x) + log(1 - D_{src}(G(x, c))) \tag{2.9}$$

Domain Classification Loss je funkce, která slouží pro optimalizaci tvorby snímku do různých domén. Je složena ze dvou funkcí 2.10 a 2.11, kde 2.10 je pro diskriminátor a 2.11 je pro generátor.

$$L^r_{cls} = -logD_{cls}(c'|x) \tag{2.10}$$

$$L^f_{cls} = -logD_{cls}(c|G(x, c)) \tag{2.11}$$

$D_{cls}(c'|x)$ představuje rozdělení pravděpodobnosti přes domény, kde c' je originální doména. Diskriminátor pomocí něj optimalizuje rozeznání cílové domény. Generátor se jí snaží minimalizovat pro lepší optimalizaci výsledků cílové domény.

Reconstruction Loss je funkce, která funguje za účelem dosažení věrnosti svého původního snímku. Užívá tuto rovnici a je aplikována v generátoru.

$$L_{rec} = ||x - G(G(x, c), c')||_1 \tag{2.12}$$

Full Objective je konečný výraz je možný reprezentovat těmito dvěma rovnicemi 2.13, 2.14.

$$L_D = -L_{adv} + \lambda_{cls} L^r_{cls} \tag{2.13}$$

$$L_G = L_{adv} + \lambda_{cls} L^f_{cls} + \lambda_{rec} L_{rec} \tag{2.14}$$

λ_{cls} a λ_{rec} jsou hyper-parametry, které řídí důležitost jednotlivcích ztrátových funkcí a rekonstrukce ve srovnání se soupeřící ztrátou.

Z důvodu nejednoznačnosti cílové domény z učení na vícero datových sadách byl vytvořen vektor masky na zaznačení klasifikace domény a na vytřídění nespecifikovaných atributů. Tato klasifikace je znázorněna rovnicí 2.15, kde C_n jsou jednotlivé klasifikace hodnot atributů datových sad m.

$$c = [c_1, ..., c_n, m] \tag{2.15}$$

Na porovnání byly použity modely *DIAT*, *CycleGAN* a *IcGAN*. Na těchto sítích byly použité datové sady *CelebA* [22] s *RaFD*, kde 90 % snímků bylo použito na trénování a zbylých 10 % bylo použito na experimenty pro porovnání věrohodnosti výsledků. Ve výsledném porovnání StarGAN měl mnohem lepší výsledky než ostatní sítě. Bohužel se zobrazili některé nedokonati v transformaci brýlí a šperků. Dále při změně barvy vlasů se změnil trochu i výraz [12].

RelGAN

Model pro generování syntetických snímků *RelGAN* je založen na generativních soupeřících síti. Model obsahuje jeden generátor a tři diskriminátory. Tato síť je tvořena čtyřmi konvolučními neuronovými sítěmi s tím, že diskriminátory sdílí šest konvolučních vrstev. Vstup je tvořen snímkem s vektorem, který určuje, do jaké domény se má dostat relativně k vstupnímu snímku. Na výstupu je snímek v cílové doméně, která také podporuje více změn najednou. Tento model si velmi cení vektoru, kde se snímek mění podle hodnoty atributu. Je-li hodnota nulová, tak se nic nemění. Je-li hodnota 1 nebo -1, tak je snímek změněn pouze o tyto hodnoty. Respektive model se snaží tento snímek upravit jen pro tento atribut a současně neměnit žádný jiný atribut. Model ho však mění relativně (například přidá více úsměvu, ale nemusí se potom úplně smát). Tato sekce je ze zdroje [39].

Funkcionalita je rozdělena do čtyř částí. První je generování snímku včetně změn, které byli nastaveny ve vstupním vektoru. Druhá část je tvořena prvním diskriminátorem D_{Real},

který se snaží kontrolovat reálnost snímku. Třetí část je druhý diskriminátor D_{Match}, který porovnává generovaný snímek se snímkem z cílové domény podle originálního snímku a relativního vektoru. Poslední část je tvořena třetím diskriminátorem D_{Interp}. Třetí diskriminátor se snaží zjistit míru interpolace generovaného snímku s originálním, kde zkoumá, zda mají oba snímky stejnou strukturu.

Tento model byl porovnáván se *StarGAN* a *AttGAN*. Z výsledku plyne, že vznikl výraznější pokrok u propagace změny atributu na obličeji bez vedlejších změn a zachování obličejových rysů z originálního snímku. Model byl rovněž testován prostřednictvím dotazování lidí. Cílem testu je vybrat nejlepší vygenerovaný snímek z předešlých modelů a *RelGAN* při stejné změně atributu. Z výsledků dotazování lidí vyplynulo, že model *RelGAN* je v této problematice (změně výhradně jednoho atributu) nejlepší.

Adam²Net

V modelu *Adam²Net* je zvolen jiný přístup pro generování snímků. Model generuje snímky jen s pomocí hlubokých konvolučních neuronových sítí s grafem adaptivní inference (*Adaptive Inference Graph*). Tato síť funguje na bázi spínání určitých neuronových vrstev podle vstupu. Cílová doména zde povoluje více změn najednou. Tento model je cílený na generování kvalitních snímků mezi více doménami za limitovaných výpočetních zdrojů. Tato sekce je ze zdroje [25].

Funkcionalita *Adam²Net* je tvořena enkodérem obsahu, enkodérem stylu a zpětným dekodérem. Enkodéry slouží k extrakci obsahu a stylu snímku. Extrakce obsahu a stylu prochází sekvencí bloků. Tyto bloky za běhu pomocí adaptivního grafu modifikují obsah tak, aby obsahoval požadovaný výstup v dané doméně. Po projití bloků se obsah dekóduje na výstupní snímek. Tento způsob umožňuje použít lineární množství bloků dle generačních domén místo kvadratického množství generátorů.

Tento model byl testován na omezené velikosti domény na datové sadě *CelebA* [22], kde si vedl poměrně dobře. Bohužel jeho kvalita výstupu nebyla tak dobrá jako u modelu *StarGAN*, který používal výkonnější technologii.

Kapitola 3

Rozpoznání osob podle obličeje a datové sady

Rozpoznávání obličeje na snímcích představuje problém, který se skládá z identifikace a ověření osob podle charakteristických rysů tváře. Tento problém je snadno řešitelný pro lidi. Bohužel až do nedávna to byl náročný problém pro počítačové vidění, protože můžou nastat nečekané vlivy. Některé tyto vlivy jsou zranění obličeje nebo působením externích vlivů (sluneční svit atd.).

K řešení počítačového problému rozpoznávání obličeje pomáhají metody hloubkového učení, které jsou schopny využít rozsáhlé datové sady s pestrým množstvím tváří pro možný vývoj a trénování modelů. Tento přístup umožňuje moderním modelům fungovat na stejné úrovni jako lidské rozpoznávání obličeje. V průběhu času už dokázali nové modely překonat lidské rozpoznávání obličeje.

Rozpoznávání osob se obecně opírá o vědu nazývanou Biometrie[34]. Biometrie je vědecká disciplína, která se zabývá měřením a analýzou biologických dat. V kontextu identifikace a bezpečnosti se Biometrie využívá k rozpoznávání jednotlivců na základě jedinečných fyzických nebo behaviorálních charakteristik, jako jsou otisky prstů, rysy obličeje, sítnice oka, hlas nebo podpis [34].

V této kapitole je popsán problém identifikace a ověřování obličejů na fotografiích v sekci 3.1. Dále jsou uvedeny základy části procesů na rozpoznávání obličeje ze snímku. Detekce postavy, zarovnání a extrakce rysů je prezentována v sekci 3.2. Dále je čtenář seznámen s procesem na rozpoznání obličejů včetně vybraných algoritmů v sekci 3.3. Na závěr této kapitoly jsou uvedeny důležité datové sady se snímky obličejů a jejich vlastnostmi. Datové sady je možné nalézt v sekci 3.4.

Tato kapitola je tvořena větší částí ze zdroje [4].

3.1 Identifikace a ověřování obličejů na fotografii

Nad problémem identifikace a ověřování obličeje se začalo uvažovat od 50. a 60. let 20. století [33]. Výzkum automatického rozpoznávání obličeje se zahájil v 70. letech. Tento výzkum vyústil ve studie věnující se rozpoznávání obličejů. První modely zaměřené na rozpoznávání obličejů byly na začátku 90. let [33]. V posledních dvou desetiletí se posunul jejich vývoj z důvodu pokročilejších technologií a lepšímu technickému vybavení.

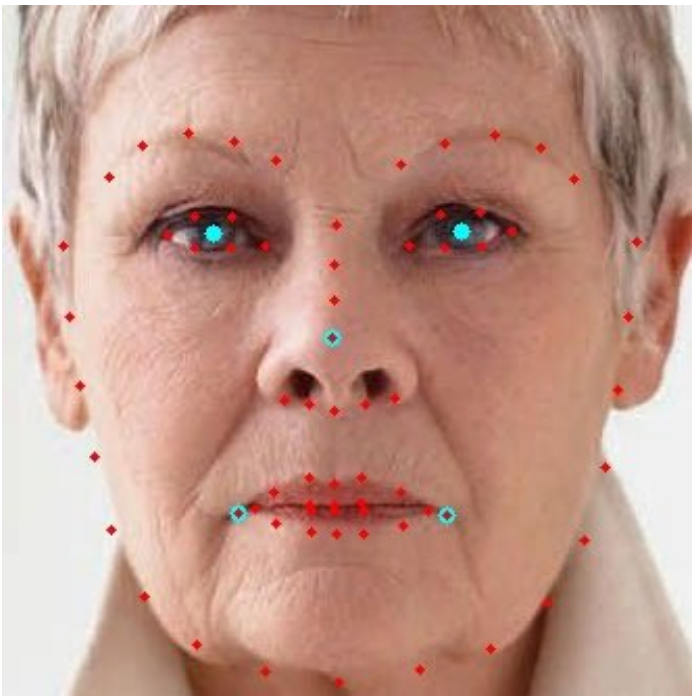
Často je potřeba rozeznat osoby z důvodu přiřazení obličeje k již zmíněné identifikaci nebo ověření, anebo i z důvodu autentizace. Tato sekce se věnuje 3 kategorií: identifikace, ověřování a autentizace.

První kategorie je identifikace. Tato část se stará o nalezení obličejů na snímku. Má široké využití například v sociálních sítích, kde se používá u snímků se skupinami lidí. U snímků se skupinami lidí je možné detekovat obličeje a detekované obličeje je možné popsat. Tím dochází k informovanosti osob, že jsou na daném snímku nebo je možné zjistit identifikaci neznámé osoby na snímku.

Druhá kategorie je ověření, které se stará o porovnávání obličejů. Používá se na snímky s už nalezenými obličeji. Například u automatického potvrzení osoby podle obličeje s fotkou na občanském průkazu. K potvrzení osoby se také přidává autentizace, která slouží k zpracování zdrojům s omezeným přístupem. Autentizace se zaměřuje na porovnávaní snímku s databází obličejů a hledání jednoho podobného.

Výše uvedené kategorie se shluknou do jednoho problému a ten se obecně nazývá automatické rozpoznávání tváře. Problematika identifikace o ověřování obličeje se věnuje statickým fotografiím, videům a i přímým přenosům.

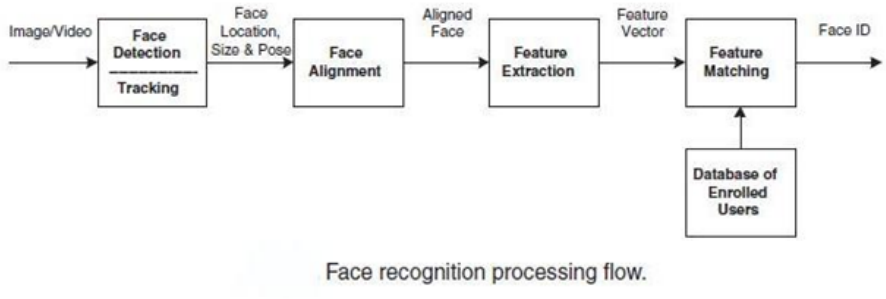
Základním principem, na kterém je vystavěno rozpoznávání tváří z Biomerie je měření vzdáleností od různých obličejových rysů (*facial features*) jako jsou nos nebo ústa. Konkrétní body, mezi kterými je měřena vzdálenost, se nazývají *landmarks* nebo *nodal points*. Mezi příklady *landmarks* se řadí koutky očí anebo okraje líček [2]. Více těchto bodů je na obrázku 3.1. Vzdálenosti mezi *landmarks* se poté skládají do jednoho souboru vektorů (*feature vector*), který je unikátní pro každý obličej.



Obrázek 3.1: *Nodal points* obličeje [2].

Rozpoznávání obličeje je proces, který je rozdělen do čtyři kroků: detekce obličeje, zarovnání obličeje, extrakce rysů a rozpoznání obličeje. Je reprezentován na obrázku 3.2.

1. **Detekce obličeje** – Lokalizování obličejů na snímku a jejich extrahování.



Obrázek 3.2: Proces rozpoznávání obličeje [4].

2. **Zarovnání obličeje** – Zarovná snímek tak, aby byl konzistentní s databází anebo jiným snímkem.

3. **Extrakce rysů** – Extrahuje ze snímku znaky obličeje, které jsou užitečné pro rozpoznání.

4. **Rozpoznávání obličejů** – Provádí porovnání obličeje s biometrickými rysy jiných obličejů z databáze.

Tento proces muže být implementován moduly pro každý krok. V určitých případech se mohou některé kroky zkombinovat do jednoho například zarovnání obličeje se spojí s detekcí obličeje.

V dalších sekcích je popsána detekce obličeje a rozpoznávání obličeje včetně jejich algoritmů. Zarovnání obličeje pracuje na jednoduchém oříznutí nebo transformaci snímku. V modelech je zarovnání obličeje obsažené v detekci společně s částí extrakce. Zbylá extrakce rysů je vysvětlena v sekci rozpoznávání obličeje.

3.2 Detekce obličeje a jejich algoritmy

Detekce obličeje je netriviálním krokem v rozpoznávání obličeje. V tomto kroku nachází rozpoznávaní objektů, u kterého je potřeba nejen najít umístění každého obličeje, ale i jeho rozsah z oblasti (*Region of Interest*). Největší ztížení je v tom, že lidská tvář se chová jako dynamický objekt, který má vysokou míru variability, což činí detekci obličeje v počítačovém vidění o to obtížnější.

Detekce musí být robustní obzvlášť na možné komplikace, které se můžou stát při pořizování snímků vnějšími ději. Nemělo by nastat, že osoba trochu pootočí hlavou, má vousy anebo snímek má vyšší jas, a obličej není rozeznán, protože pak není připuštěn ani k samotnému rozeznávání obličeje a výsledek skončí neúspěchem.

Algoritmy zabývající se detekcí obličeje

Pro detekci obličeje na snímku se používají dva typy přístupů. První přístup využívá ručně vyrobené filtry, které lokalizují tváře na snímku na základě hlubokých znalostí domény. Tento princip být velmi rychlý a účinný, když se filtry shodují. Bohužel může drasticky selhat při nevytvoření všech variant. Druhý přístup je pomocí strojového učení. Model se učí, jak automaticky lokalizovat a extrahovat obličeje na snímku. Tento princip je založen

na neuronových sítích. Konkrétní případy algoritmů pro detekci obličeje ze snímku jsou *Multi-task Cascaded Convolutional Networks* a *RetinaFace*.

Multi-task Cascaded Convolutional Networks

Pro detekci obličejů existuje model *Multi-task Cascaded Convolutional Networks*, který se skládá ze tří kaskádovitě spojených konvolučních neuronových sítí. Tento model zvládne nejen detekci obličejů, ale dokáže najít umístění obličeje a také obličejové rysy. Informace o modelu jsou citovány ze zdroje [42].

Model *Multi-task Cascaded Convolutional Networks* vylepšil od svých předchůdců výkonost a přesnost, protože rozdělil detekci do tří částí. Tyto části jsou tvořené konvolučními neuronovými sítěmi. První část má za účel najít možné kandidáty na obličej v celém snímku, a proto je použitá konvoluční síť mělká a rychlá. Tato sít používá jako ztrátovou funkci *Cross-entropy loss*. Druhá část je více komplexní. Z vybraných kandidátů najde obličeje, které projdou hodnocením silnější konvoluční sítě. Vybrané obličeje jsou umístěné do rámečku. Poslední část už jen detekuje obličejové znaky a vylepšuje výsledek vycentrováním. Použitá konvoluční síť v třetí části je už větší a pomalejší. Poslední dvě konvoluční sítě používají ztrátovou funkci *Euclidean loss* [42], která je v rovnici 3.1, kde $\hat{y}_i$ je regresní cíl získaný ze sítě a y_i je pevně daný koordinát s obsahem buď vlastností rámečku obličeje nebo charakteristickými znaky obličeje.

$$L_i = ||\hat{y}_i - y_i||_2^2 \tag{3.1}$$

Tento model byl porovnáván se sedmi dalšími modely *RCPR*, *TSPM*, *Luxand face SDK*, *ESR*, *CDM*, *SDM* a *TCDCN* na datové sadě *WIDER FACE easy* [22], kde dopadl nejlépe s 81 % přesností.

RetinaFace

RetinaFace je pokročilejší model, který je vnitřně složen ze tří paralelních procesů. Tyto procesy na sebe dohlížejí a společně se ovlivňují pomocí *Feature Pyramid Network*, která efektivně detekuje objekty na různých úrovních měřítka. První proces má na starost najít místo s obličejem a vypočítat jeho skóre společně s nalezením jeho oblasti. Další proces má na starost rozeznat obličejové znaky. Poslední proces se věnuje porovnávání nalezeného obličeje s vytvořeným 3D obličejem, který je mapovaný na 2D snímek. Základní struktura je tvořena z hlubokých konvolučních neuronových sítí, kde se vyskytovaly i reziduální bloky (*residual blocks*) [14].

Tento model používá speciální víceúlohovou ztrátovou funkci, která je znárodněna výrazem 3.2.

$$L = L_{cls}(p_i, p_i^*) + \lambda_1 p_i^* L_{box}(t_i, t_i^*) + \lambda_2 p_i^* L_{pts}(l_i, l_i^*) + \lambda_3 p_i^* L_{pixel} \tag{3.2}$$

Výpočet je rozdělen do čtyř celků. První se stará o detekci obličeje. Je dán výrazem $L_{cls}(p_i, p_i^*)$, kde p_i je pravděpodobnost, že v místě i je obličej, a p_i^* je hodnota, která představuje nulu nebo jedničku podle toho, zda to je, nebo není obličej. L_{cls} poté představuje ztrátovou funkci *softmax loss for binary classes*.

Druhá část je zaměřena na predikci vlastností oblasti, kde t_i jsou předpovídané koordináty odsazení a proměnné x a y jsou šířka a výška. t_i^* jsou poté vlastní pevné koordináty. Jako ztrátová funkce L_{box} je použita *smooth-*L_1 [15].

Třetí část $L_{pts}(l_i, l_i^*)$ slouží pro rozeznávání obličejových rysů v pěti bodech, kde l_i jsou předpovídané body a l_i^* jsou skutečné body. Ztrátová funkce v tomto případě je podobná jako u predikce rámečků.

Balanční proměnné jsou nastaveny $\lambda_1 - \lambda_3$ na hodnoty 0,25, 0,1 a 0,01, což znamená zvýšení významu prvních celků kvůli přesnosti a důležitosti. Je to z důvodu možného nalézaní nějakého obličejového rysu ve vyšších částech a nenalezení v nižších částech. Například vyšší část najde tvary podobné nosu, ústům a očím, zjištěné znaky nemusíte dat dohromady celý obličej, čímž se poté zabývá nižší část.

Poslední je L_{pixel}. se jménem *Dense Regression Loss* pro porovnání vybraného obličeje s obličejem, který byl vytvořen ve 3D a zachycen na 2D plátno s podobnými podmínkami jako reálný porovnávaný obličej. Tento proces je popsán rovnicí 3.3

$$L_{pixel} = \frac{1}{W \times H} \sum_{i}^{W} \sum_{j}^{H} ||\mathcal{R}(\mathcal{D}_{P_{ST}}, P_{cam}, P_{ill})_{i,j} - I_{i,j}^*||_1 \tag{3.3}$$

Tato rovnice obsahuje W a H, které reprezentují šířku a výšku k příslušnému místu obličeje $I_{i,j}^*$. Písmeno $\mathcal{R}$ neboli *render* je vyobrazení 3D obrázku na 2D plochu. Dále $\mathcal{D}_{P_{ST}}$ nastavuje předvybraný 3D pohled na před-připravený obličej, P_{cam} jsou parametry pro pozici kamery a P_{ill} přestavuje parametry pro osvícení snímku.

Model *RetinaFace* byl trénován na upravených snímcích z *CelebA* [22] s *WIDER FACE* [40] a byl testován na mnoha datových sadách, kde jedna z nich byla *WIDER FACE easy* [22], kde dopadl s 96 % přesností. Pokud se zapojil do celku pro rozpoznávání obličeje, tak s modelem *ArcFace* dosáhli lepšího výsledku než s *Multi-task Cascaded Convolutional Networks* průměrně o 0,5 % s datovými sadami *LFW*, *CFP-FP* a *AgeDB-30*. Jak je vidět, tak *RetinaFace* zlepšila detekci i u snímků s horší kvalitou.

3.3 Rozpoznávání obličeje a jejich algoritmy

Existuje několik variant, jak rozpoznat obličej ze snímku. Vhodná varianta závisí na řešení konkrétního problému. Obecně je možné rozpoznávání obličeje ze snímku popsat jako řízenou úlohu prediktivního modelování trénováním na vzorcích se vstupy a výstupy. Vstupy jsou vždy snímky, které obsahují jeden obličej. Výstup záleží na úloze, ale obecně to je označení nějaké třídy (číslo v kategorické třídě, atd.) anebo pravděpodobnosti či redigované číslo. V dnešní době to je výstup z neuronové sítě v podobě vektoru s názvem *feature vector*.

Úlohy, které by měly modely zvládat, jsou nalezení shodného obličeje, nalezení podobných obličejů a schopnost generování obličeje ze svých znalostí uchovávaných v databázi zpracovaných obličejů [4], které byli převeditelné například na *feature vector*.

Rozeznávání by mělo fungovat ve dvou režimech. První je režim ověření, kde je vstupní snímek porovnáván s jiným. Ověření má zajistit, že se obě identity na snímcích shodují. Druhý je identifikace, kde ke vstupnímu snímku je hledána podoba v databázi obličejů.

Algoritmy zabývající se rozpoznáním obličeje

Modelů pro rozeznání obličeje je pestré množství a vznikali již od 90. let, kde byly sestrojeny jedeny z prvních modelů na základě holistického přístupu [4]. Poté se kolem roku 2000 začal vyvíjet přístup k učení místních funkcí (metoda mělkého učení), které zlepšil přesnost

z přibližně 60 % na hodnoty okolo 90 %. Později kolem let 2014 a 2015 se vyvinuly systémy hlubokého učení pro rozpoznávání obličejů, které posunuly přesnost datových sad v průběhu let na 99 % [4].

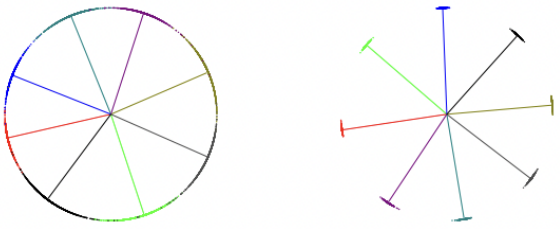
Nyní se používají systémy hlubokého učení, které nejčastěji pracují na neuronových sítích. Systémy hlubokého učení jsou představeny na malém vzorku modelů: *ArcFace*, *MagFace* a *QMagFace*.

ArcFace

ArcFace je model na rozeznání obličejů ze snímků vytvořený na hlubokých konvolučních neurálních sítích. Je založen na myšlence úhlového okraje (*angular margin*), který snadněji rozděluje třídy obličejových rysů. Další výrazný pokrok je v trénování, kde se docílilo použití mechanismu, který snadněji zpracovává snímky s horší kvalitou nebo s nechtěným šumem, a tím docílil ještě větší robustnosti modelu. *ArcFace* se nejen používá k identifikaci či ověření, ale má funkčnost generování obličejů. Generování snímků je založeno na aproximaci cílové identity pomocí gradientu sítě a *batch normalization* vrstev, které pomáhají s uloženými hodnotami pro normalizaci.

První myšlenka úhlového okraje, která zde byla propagována, byla vnesena do ztrátové funkce se jménem *Additive Angular Margin loss*. Cílem této funkce je rozřadit obličejové rysy do různých tříd a potom tyto třídy posunovat v úhlovém prostoru dál od sebe. Pro snadnější porozumění je možné si přestavit, že se jednotlivé vektory, které hodnotí stejnou třídu, mapují na povrchu koule kolem určitého úhlu. Pro každou třídu je jeden úhel a toto vektorové ohodnocení má určitý rozptyl, takže se můžou jednotlivé třídy okrajově překrývat. Do této doby byly vytvořeny modely na ohraničení prostoru, ale *ArcFace* přišel se způsobem, jak soustředit vektorové ohodnocení co nejblíže k úhlu třídy od rozhodovací hranice v lineární složitosti vůči jiné třídě. Výsledkem jsou třídy s přesnými hranicemi a prostory mezi nimi, kde se nalézají zřídka vzorky. Tento vzor je vyobrazen na obrázku 3.3 a mechanismus je poté převeden do matematické rovnice 3.4, kde parametr s, který slouží pro škálovaní, dále θ_j popisuje úhel mezi třídou j a m je velikost mezery mezi třídami.

$$L_i = -\log\left(\frac{e^{s\cos(\theta_{y_i}+m)}}{e^{s\cos(\theta_{y_i}+m)} + \sum_{j=1,j\neq y_i}^{N} e^{s\cos(\theta_j)}}\right) \tag{3.4}$$



(a) Norm-Softmax (b) ArcFace

Obrázek 3.3: Vyobrazení, jak *ArcFace* posouvá vektory prvků různých tříd dál od rozhodovací hranice [13].

Další funkcionalitu, která je zavedená v *ArcFace*, je technika učení pod-center, které usnadňují učení a třídění snímků od velkého šumu z reálného světa. Obecně se chová *ArcFace*

jako *Triplet loss*, která se snaží přiblížit k pozitivním vzorkům a oddálit od negativních, ale u *ArcFace* se snaží přiblížit k jednou z pod-centrů, které reprezentují jednotlivé třídy vzorků se společnými znaky. U negativních je to na stejném principu, ale vzdálenost neporovnávají pouze s pod-centrem reprezentující jednu třídu, ale se všemi ostatními třídami. Tímto se přidá vliv ostatních tříd na výpočet vzdálenosti a třídy se stanou nezávislé. Vzdálenost počítá pomocí Euklidovy norma (*L2 norma*)

Další výhodou je, že pod-centra se řadí do dvou skupin: dominantní a nedominantní. Při trénování se model učí na různých vzorcích, které zapadnou k některým pod-centrům příslušící třídě. Zde se začnou projevovat typy pod-center. Pokud vzorek obsahuje šum, tak se přiřadí k nedominantnímu pod-centru pomocí vzdálenosti od nejbližšího centra, a poté má menší vliv u hodnocení. Pokud je vzorek kvalitní (neobsahuje šum atd.), tak je přiřazen do dominantního pod-centra, který má největší vliv na hodnocení. Dominantní pod-centrum je jen jedno v pod-třídě, kterou v tom případě reprezentuje. Shromažďuje naprostou většinu čistých snímků, kterých je větší množství, a tudíž mají potom i větší. Toto rozdělení poté pomáhá i se zátěží, kde všechny vzorky nejdou přes jednu třídu, ale přes pod-centra, kde vzorky nejvíce přecházejí přes dominantní pod-centra, a tak zátěž není centralizovaná.

Tento model byl vyzkoušen na několika různých datových sadách jako *LFW* [18] nebo *CFP-FP* [32], kde prošel s nejlepším hodnocením 99,5 %. Tímto lze celkově shrnout, že *ArcFace* skvěle používá svoje vyhodnocení s vylepšenou odolností vůči variacím. Na datových sadách bylo použito velké množství snímků, a tím lze říct, že dobře funguje v úlohách identifikace obličeje ve velkém měřítku, kde počet identit není pevně omezený. Zdroj této sekce je [13].

MagFace

MagFace je jeden z rozšiřujících modelů *ArcFace*, který se zaměřuje na vnesení kvality vzorku do učení modelu. Hlavním cílem je učení na principu třech stejných snímku s rozdílnou kvalito. Tím se model více naučí z kvalitního než z toho méně kvalitního. Respektive se zhoršující kvalitou se bude zvětšovat vzdálenost od středu třídního rozdělení. Druhým cílem je minimalizovat cenu změny na již předchozím modelu pro zajištění větší přesnosti.

Řešení nabízí tento model v podobě zvýšení nebo snížení délky počátečního vektoru reprezentujícího vlastnost okraje. Tato změna se poté projeví do ztrátové funkce *MagFace* 3.5, kde originál rovnice *ArcFace* je na 3.4.

$$L_i = -\log \frac{e^{s\cos(\theta_{y_i}+m(a_i))}}{e^{s\cos(\theta_{y_i}+m(a_i))} + \sum_{j=1,j\neq y_i} e^{s\cos(\theta_j)}} + \lambda_g g(a_i) \tag{3.5}$$

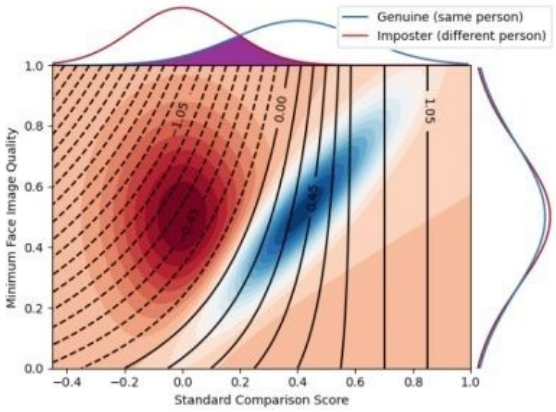
Rovnice popisuje parametr s, který slouží pro škálovaní, dále θ_j popisuje úhel mezi třídou j a f_i (vlastnost snímku). $m(a_i)$ je aditivní úhlový okraj kalkulovaný na kvantitě snímku a $g(a_i)$ je regulatizér, který oceňuje kvalitnější snímky. λ_g je hyper-parametr (balanční proměnná) užívaní pro balancování klasifikační a regularizační ztráty.

Toto vylepšení se promítlo do experimentů s datovými sadami LFW a CFP-FP, kde *MagFace* získal ohodnocení 99,83 % a 98,46 % průměrně lepší skóre jak *ArcFace* o 0,04 %. U těžších datových sad jako *IJB-B* a *IJB-C* měl okolo 90 % a 95 %. Toto hodnocení s jednou výjimkou bylo lepší průměrně kolem 1 %. Z těchto výsledků je možné vyvodit, že přidání kvality má pozitivní dopad na celý model. Vytvořeno ze zdroje [24].

QMagFace

QMagFace pokračuje v rozvoji *MagFace*, kde přidává speciální funkci *Quality-aware comparison function*. Tato funkce slouží pro zvýšení přesnosti rozhodování v případech, kdy se třídy se vzorky obličejů v hodnotícím prostoru hodně překrývají, ale obsahují stejné obličeje. Model je poté používán pro obtížné situace jako porovnávání různě starých lidí, otočených nebo snímků s různou kvalitou.

Metodu *Quality-Aware Comparison Scoring* je možné popsat jako upravený způsob hodnocení rozdílů mezi dvěma obličeji. Tento způsob je vyobrazen na obrázku 3.4, kde na osách je znázorněna dřívější způsob vyhodnocení (při spojení by vzniky svislé a vodorovné přímky) a čárami v obrázku je znázorněna nová metoda s pomocí *Quality-Aware Comparison Scoring*. Různé třídy jsou reprezentována červenou a modrou barvou. Na obrázku je možné vidět překrývající se třídy včetně efektu metody, která snižuje překrývání tříd pomocí čar.



Obrázek 3.4: Vizualizace *Quality-Aware Comparison Scoring* [35].

Výše uvedená funkcionalita je dosažena funkcí 3.6, kde $s = cos(e_1, e_2)$ je skóre vyjádřené podobností dvou šablon, které jsou vytvořené *MagFace* ztrátovou funkcí se dvěma snímky a $q1, q2$ jsou znázornění kvalit těchto snímků. α a β jsou proměnné pro trénování.

$$s'(s, q_1, q_2) = min\{0, \beta * s - \alpha\} * minq_1, q_2 + s \tag{3.6}$$

S tímto modelem bylo experimentováno na datových sadách AgeDB, CFP-FP a dalších, kde se umístil první s 98,50 % a 98,74 % o 0,2 % před ostatními modely *MagFace*, *ArcFace*, atd. Z těchto výsledků je patrné vylepšení kvality hodnocení modelu, ale naskytli se i limitující faktory. Tato metoda není moc účinná u nízko-kvalitních snímků a při práci s videm je potřeba agregace pro získání snímků s větší kvalitou. Tato sekce je ze zdroje [35].

3.4 Datové sady

V této části jsou představeny některé datové sady, které jsou použitelné pro generování snímků na rozpoznávání. Současně jsou využitelné pro vyhodnocování identit ze snímků. Mají dobré využití pro hodnocení modelů, protože mají dobře popsané vlastnosti snímků

například atributy nebo identity na snímcích. Tyto datové sady se jmenují *CelebA* a *LFW - People*.

CelebA

CelebA je zkratka pro datovou sadu se jménem *Celeb Faces Attributes Dataset* [22]. Tato datová sada obsahuje přes 200 tisíc obličejových snímků od 10 177 lidí. Každý tento snímek je obsahuje až 40 atributů (barva vlasů, pohlaví, výrazy obličeje například jako smích a další). Uvedená datová sada se může využít pro trénování nebo verifikaci správně zařazených atributů.

LFW - People

LFW - People je datová sada pro rozpoznávání obličejů a celým názvem je *The Labeled Faces in the Wild face recognition dataset* [18]. Datová sada obsahuje přes 13 000 snímků s popiskami jmen lidí, kteří na nich byli zachyceni při veřejných akcích. Tyto snímky jsou seřazeny do složek, kde 1680 osob obsahuje dva a více snímků. Převážná část těchto lidí jsou celebrity, se kterými se nacházeli snímky na různých internetových webech.

Kapitola 4

Návrh a implementace

Tato práce se zaměřuje na vyhodnocení podobnosti generovaných snímků. Práce se konkrétněji zaměřuje na tvorbu modelu, porovnání podobnosti obličeje originálního snímku a vygenerovaného snímku, který byl vytvořen z originálního snímku úpravou jeho atributu. Cílem je vytvořit robustní program, který na výstupu předá číslo informující o podobnosti obličeje.

Pro výše uvedený úkon je vytvořena siamská neuronová síť, která porovnává identity dvou vstupních snímků a na výstup podává číslo od nuly do jedné.

Obsah kapitoly je rozřazen do tří částí. První část 4.1 obsahuje stávající řešení s návrhem řešení a výsledný popis, jak by aplikace měla pracovat. Další část 4.2 je tvořena přípravou dat potřebných pro učení modelu, ověření funkčnosti modelu a následným testováním úspěšnosti modelu. Poslední část 4.3 se skládá ze samotné implementace v jazyce *Python*. Je zde uveden popis sítě včetně vytvořené architektury sítě. Dále se v sekci 4.3 nacházejí použité prvky a předpokládaný výstup. Na závěr je v sekci přehled kódu a trénování modelu.

4.1 Stávající řešení a návrh

Tato sekce se specializuje na návrh modelu a funkčnost aplikace. Jsou zde stručně shrnuty stávající řešení a jak jsem již zmínil, tak je popsán i základ modelu společně s jeho důvodem a jeho cílem. Následuje návrh funkčnosti aplikace. Také je popsána práce konzolové aplikace včetně argumentů, které lze použít.

Stávající řešení

Rozvoj generativních modelů se v posledních letech hodně zlepšil. Bohužel kvalita nástrojů na hodnocení těchto modelů je stále na nízké úrovni, protože se jedná o složitý proces s mnoha problémy, které jsou špatně technologicky řešitelné.

Jeden z problémů u generování snímků je například realističnost. U tohoto problému se řeší, zda snímek správně popisuje reálný svět. Modely je možné naučit standardní strukturu obličeje na porovnání, ale porovnávaný snímek musí být také standardní. Chyby nastávají například při obarvení pokožky nebo při úrazu, který zasáhne obličej. V případě úrazu model může mít problém, protože se jedná o reálný obličej, ale nevypadá jako standardní obličej, a tak nemusí dojít k nerozeznání rysů na obličeji.

Další problém, který nastává při generování obličeje, je u modelů, které různorodě upravují snímky obličejů. Problém nastává v samotné podobnosti originálního a generovaného

snímku, kde vygenerovaný snímek by měl mít stejné rozmístění obličejových rysů jako originální. Bohužel z důvodu generování vybraných částí se změnou může dojít k posunu nebo překrytí jiných objektů. Tím se poté mohou posunout klíčové body pro porovnávání, nebo se dokonce nemusí nalézt obličejové rysy, kvůli podobnému problému jako u realističnosti. Pro zlepšení snímků některé generativní modely přidávají části, které pomáhají k jejich správnému přenesení jako u *RelGAN* 2.3, kde je diskriminátor s interpolací nebo i *StarGAN* 2.3 se zpětným převedením do původní domény, které přímo přenesení nepomáhají, ale pomáhají jim jako celku. Těchto zlepšení je i v ostatních modelech obrovské množství a fungují na různých principech, a proto je nutné vytvořit jednotné hodnocení.

Jeden z typů vyhodnocení je v podobě dotazníku na lidi nebo pro společnost, kde se dostávají otázky na generované snímky nebo se vybírají snímky, které splňují nějaké kritérium nebo jsou v něčem nejlepší. Příkladem je výběr ze tří snímků, kde dva jsou generované a jeden reálný. Účel je vybrat reálný a při každém omylu se přičítá skóre vybranému modelu. Tento typ testování je poměrně pomalý, ale samotná kvalita ověření na velkém množství lidí je poměrně dobrá. S tímto typem byl například testován *RelGAN* 2.3.

Další možností je používání modelů na řešení různých problémů. Každý model se použije ke kontrole specifického problému a umožní zjistit komplexní výsledky. Výhodou tohoto způsobuje rychlý přístup k výsledkům i celkový přehled, jak model dopadl. Nevýhoda je, že výsledky budou mít určitou míru chybovosti, což je složité na výrobu. Další nevýhodou je, že každé odvětví, do kterého spadá jeden problém, obsahuje minimální množství modelů, takže je ještě složitější snížit chybovost použití různých specializovaných modelů. Toto řešení je velmi náročné, a proto se většinou testuje jen pár modelů, které jsou velmi přesné.

Do těchto odvětví spadá vyhodnocení věrohodnosti originálního a generovaného snímku, kde nejbližší model je *ArcFace* 3.3, který porovnává dva reálné snímky obličeje a vypočítá vzdálenost mezi nimi, která vyjadřuje, zda se jedná o stejnou osobu. Tento model má vysokou přesnost přesahující 99 %, ale má pár nevýhod. Jedna z nevýhod je, že model není zaměřený na generované snímky, které mohou mít pochybnou realističnost, a proto může v některých případech selhat, protože nemůže extrahovat rysy ze snímku, které mohou být špatně. Další zhoršení je z důvodu jeho zpracování. Tento model je poměrně komplikovaný a uzpůsobený na rozeznávání reálných obličejů, a proto je jeho modifikovatelnost pro dosažení zpracování generovaných snímků velmi složitá. Poslední jeho pozitivní i negativní vlastností je jeho výstup. Tento model má vzdálenosti na výstupu. Pokud jsou vzdálenosti velmi podobné, tak výsledek je od 0 do hodnotu x, pokud jsou snímky odlišné, tak výsledek je od hodnoty x do nekonečna. Nepříjemné je to, že hodnotu x je třeba najít ze vzorků. Není možné, aby byla například 0,5, kde 1 představuje rozdílné obličeje a 0 jsou stejné obličeje.

Z výše uvedených důvodů navrhuji vytvořit nový model, který bude založen na neuronových sítí. Nově navržený model se bude snažit výše popsané nedokonalosti vyřešit a vytvořit model zaměřující se přímo na porovnávaní obličejů se sníženou závislostí na realističnost.

Siamská neuronová síť

Jeden z typů neuronových sítí je siamská síť. Důvod, proč jsem vybral siamskou síť, je v jeho porovnání vstupů. Ostatní typy jako klasifikační neuronové sítě potřebují mít pevný počet tříd na to, aby je mohly správně rozdělit do třídy na rozdíl od siamských neuronových sítí, kde se používá výpočet vzdáleností mezi vektory, které reprezentují data ze vstupu. Tímto výpočtem jsem schopen analyzovat různé modely skrze velké množství různých snímků. Další výhodou je její robustnost. Tato síť je výpočetně složitější na trénování, které probíhá pouze jednou. Poté poskytuje rozpoznávání snímků o malé kvantitě s velkou přesností [3].

Dále je možné zvolit jako podsítě konvoluční sítě, které se podle naučených dat můžou lehce měnit. Z důvodu možnosti zvolení konvoluční podsítě je částečně možné i ovlivňovat realističnost vstupních vzorků dle faktoru, zda jsou trénované na reálných nebo generovaných snímcích. Poslední výhodou je z důvodu použití dvou stejných podsítí. Kvůli tomu je možné uložit výstupní vektor místo celého snímku pro rozpoznáváni. Z uvedených důvodů jsem vybral i ztrátovou funkci *Contrastive loss*. Tato funkce používá pouze dva snímky, a proto konečná architektura je snadnější na výpočet a jednodušší na ovládání.

Funkcionalita programu

Plánovaná funkcionalita výsledného programu je následující. Vytvořený program se jmenuje `Gen_Verifier`. Program je aplikace do příkazového řádku, který má čtyři argumenty, kde jeden je možné rozdělit do tří možností. Jeho zápis vypadá takto:

```
Gen_Verifier file_path_A file_path_B [Options] [file_path_result]

Optinons:
    [-v|--version]
    [-h|--help]
    [-dr]
```

První i druhý argument jsou pro zadání cesty ke snímkům nebo složky se snímky k porovnávání. Třetí argument je určen pro zadání výsledku s verzí, nebo nápovědou. Poslední argument je pro možné uložení výsledku do souboru.

Aplikace má tři módy. Mód je zvolen podle zadání dvou argumentů `file_path_A` a `file_path_B`. První mód je porovnávání snímku se snímkem. Aplikace je porovná a vypíše výsledek. Druhý mód je porovnávání snímku se složkou. V tomto případě program prolistuje složku a porovná každý snímek se snímkem v argumentu. Z těchto dat udělá průměr a vypíše ho na výstup. U posledního módu je porovnávána složka se složkou. V tomto případě program vezme za sebou abecedně jdoucí snímky a porovná je mezi sebou, výsledek zprůměruje a vypíše.

Argument `-v` anebo `--version` spustí vypsání verze do příkazového řádku. Pokud jsou přítomny i jiné argumenty, tak jsou ignorovány a tento se zpracuje s předností.

Argumenty `-h` anebo `--help` dělají podobnou věc jako předchozí argument. Na výstup vypisují nápovědu a v prioritách je až po argumentu s verzí.

Pátý argument `-dr` modifikuje podrobnost výpisu. Pokud není obsažený, tak bude vypisovat strohé informace, jak porovnávání dopadlo. Porovnání snímku se snímkem vypíše jednu hodnotu. V ostatních modelech, tedy v módu dva a tři, vypíše průměr. Pokud je obsažený, tak u módů dva a tři vypíše podrobnější výsledky. Jsou vypsány i jednotlivé hodnoty porovnávaných dvojic snímků společně s jejich jmény.

Poslední argument je `file_path_result`. Tento argument slouží ke zjištění, kam se má výstup vypsat. Pokud není obsažen v argumentech, tak se všechny informace zapisují na standardní výstup. Pokud je zapsaný v argumentech, tak se výsledek zapíše do souboru nakonec a oddělí se novým řádkem. Soubor musí mít koncovku `.txt`. Pokud soubor neexistuje, tak ho vytvoří. Pokud cesta ukazuje na složku, tak se program ukončí se speciálním návratovým kódem a vypíše hlášku na standardní chybový výstup.

4.2 Příprava dat

Příprava dat je nedílnou součástí tvorby modelu. Pro potřeby trénovaní, porovnávání a testování je nutné mít přichystané datové sady odpovídající požadavkům modelu. Konkrétně pro navrhovaný model je potřeba datová sada zajišťující jeho funkcionalitu. Dále je potřeba zajistit datovou sadu s generovanými snímky a jejich originály. Pro tento účel je nutné získat model, který generuje syntetické snímky obličeje, a datovou sadu, na kterých se model bude učit. Tato datová sada musí být dostatečně velká, aby se z ní model dokázal dobře naučit a aby zbyly snímky na další zpracování při porovnávání. Dohromady je nutné vytvořit dvě datové sady, kde jedna musí být částečně vytvořena přes generativní model *StarGAN* 2.3. Tímto modelem se vytvoří generované syntetické snímky.

První datová sada je cílená na právě vyvíjený model. Datová sada bude využita pro trénování a verifikaci modelu na reálných datech. K tomuto je potřeba mít snímky se záběrem na obličej. U každého snímku je nutné znát jeho identitu. Uvedené parametry splňuje datová sada *LFW - People*, která je popsána v sekci 3.4 s příklady na obrázcích 4.1. Než se tato datová sada může použít na trénování modelu, tak se musí vytřídit snímky osobností, které mají jenom jeden snímek, aby datová sada byla vyvážená vůči dvojicím stejných a rozdílných obličejů. To znamená, že se při trénování z datové sady vytvoří vzorek ze dvojice snímků, které obsahují dva stejné obličeje nebo dva rozdílné. Současně dvojice snímků se budou z datové sady vybírat náhodně. Tím může docházet k situaci, že se rozdílné obličeje budou párovat se snímky jednotlivě na rozdíl od stejných, a tím si zvětšit množství vzorků, které se mohou vybrat. Toto pravděpodobně může způsobit nerovnosti v trénování a zhoršenou kvalitu modelu. Obecné trénování bude probíhat tak, aby dvojice dvou stejných obličejů a dvou rozdílných obličejů byly ve stejném zastoupení v jedné skupině (*batch*). Tuto datovou sadu je dál potřeba rozdělit na dvě části. První část, 10 % snímků, bude použita na testování. Druhá část (90 % snímků) je použito na trénování.

Obrázek 4.1: Příklady z datové sady obrázků *LFW - People* [18].

Druhá datová sada snímků je potřebná pro generování syntetických snímků. Od této datové sady je potřeba, aby byla dostatečně velká a obsahovala atributy snímků, protože se s ní bude učit generující model *StarGAN* ze zdroje [11] a druhá část bude použita jeho výsledné testování. Pro tento účel je použita sada *CelebA*, která je popsána v sekci 3.4. Tato sada je rozdělena na tři části. První dvě třetiny jsou použité pro trénování generátoru *StarGAN* a poslední třetina je pro porovnání a testování do právě vyvíjeného modelu. Cílem této datové sady je vytvořit novou datovou sadu složenou ze dvojice originálního a vygenerovaného snímku, kde vygenerovaný snímek je vytvořen pomocí modelu *StarGAN* změnou jednoho atributu. Výsledkem je několik datových sad, kde je změněn právě jeden atribut, například pohlaví nebo barvy vlasů. Příklady originálních a generovaných snímků jsou na obrázku 4.2 a 4.3, kde první má změnu pohlaví, druhý obrázek má změnu barvy vlasů a třetí změnu věku.

Obrázek 4.2: Příklad snímků z datové sady *CelebA*.

Obrázek 4.3: Příklad generovaných syntetických snímků.

4.3 Implementace aplikace

Tato sekce se zaměřuje na popis navrhovaného modelu, který je zpracován v jazyce *Python*. Jazyk *Python* obsahuje i potřebné knihovny pro správnou funkci. Následuje podrobný popis modelu s jeho funkcionalitou rozdělenou do vrstev siamské neuronové sítě. Na závěr jsou podrobně popsány jednotlivé třídy a metody s ukázkou kódu pro jejich volání. Dále je prezentováno trénování modelu.

Python

Python je vysokoúrovňový skriptovací programovací jazyk, který je dynamicky typovaný a obsahuje *Garbage collection* [27]. Tento jazyk podporuje více programovacích paradigmat (objektově orientovaného a funkcionálního programování). Jelikož je tento programovací jazyk skriptovací, tak je i interpretovatelný a to mu pomáhá mít velký rozsah mezi velké

množství platforem, ale za cenu snížení jeho výkonu. Některé funkce se však výrazně zrychlily pomocí optimalizací, které ale někdy přidají čas strávený při instalacích. Dynamické typování přináší určitou volnost při implementaci, ale je třeba s ním zacházet obezřetně. Statická analýza nemusí odhalit tolik chyb nebo chyby se zjistí až po spuštění. Dále je nutné se chovat obezřetně, protože při častých změnách typu nemusí programátor vědět jaký je vždy typ parametrů nebo návratových hodnot při nedokonalé dokumentaci, i když v posledních verzích přidali možnost slabého typování u argumentů, návratových hodnot a proměnných. Tyto vlastnosti však pomáhají k rychlému a snadnému vývoji obzvláště na prototypech nebo malých projektech. Z výše uvedených důvodů se stal programovací jazyk *Python* jedním z nejoblíbenějších programovacích jazyků.

Jak bylo uvedeno výše, pro tvorbu modelu byl vybrán programovací jazyk *Python* z níže uvedených důvodů. *Python* je jedním z nejrozšířenějších jazyků pro práci s neuronovými sítěmi, ale hned za ním je R a Java. Je to z důvodu, že *Python* nabízí vyšší úroveň abstrakce a je snadnější a pohodlnější na používání. Obsahuje knihovny jako *TensorFlow* a *Keras*, které neskutečně usnadní tvorbu a trénování modelů. Na druhé straně silně typované jazyky, jako je C++, se kvůli své komplexnosti používají méně často, ale nacházejí své uplatnění v kritických oblastech, jako jsou jádra umělé inteligence a další specializované části.

Potřebné knihovny

Jak bylo zmíněno v předchozí sekci, tak *Python* má širokou škálu knihoven pro práci nejen s neuronovými sítěmi, ale i pro práci s velký množstvím dat. Tyto knihovny mají výhodu nejen v tom, že už jsou implementované, ale i toto, že jsou již odladěny a optimalizovány. Při správném použití tyto výhody výrazně přispívají k rychlosti a stabilitě výsledného programu.

V této aplikaci jsou použity knihovny specializované primárně na neuronové sítě, strojové učení a testování. Dále obsahují knihovny na zpracování obrázků, manipulaci s velkými datovými poli a grafů. Níže jsou tyto knihovny podrobně představeny.

První knihovnou, kterou je třeba zmínit, je `NumPy`[1]. Tato knihovna přidává možnost pracovat s vícerozměrnými poli, vektory a i maticemi. Dále nabízí širokou škálu matematických funkcí. Je vynikající pro práci se statistikou a správou větších dat. V programu je využito z této knihovny například vícerozměrné pole `ndarray`, které je použité k načítání vzorků.

Další knihovnou je `OpenCV` [2], která nabízí ohromné množství funkcí pro zpracování snímků a videí. Například v knihovně jsou funkce na zvětšení a zmenšení snímků, vkládání obrazců do snímku nebo vložení okrajů. Knihovna v aplikaci je použita k načtení snímků, které jsou následně použity jako vstupy pro model.

`Pandas`[3] je knihovna určená pro manipulaci a analýzu dat. Nabízí především datové struktury a operace pro práci s číselnými tabulkami a časovými řadami. V případě tohoto projektu byla za pomocí této knihovny tvořena většina grafů.

Knihovna `FaceAIKit`[4] se specializuje na detekci a rozpoznávání obličejů. Používá model *Retina* pro detekci obličejů a modely *ArcFace* a *MagFace* pro jejich rozpoznávání. Tato knihovna byla vyvinuta vedoucím práce doktorem Tomášem Goldmannem.

[1] https://numpy.org/
[2] https://opencv.org/
[3] https://pandas.pydata.org/
[4] https://pypi.org/project/face-ai-kit/

`Scikit-learn`[5] je knihovna pro strojové učení. Nabízí široké spektrum algoritmů pro klasifikaci, regresi a shlukování, včetně nástrojů pro dělení datových sad na trénovací a testovací.

`Keras`[6] je knihovna, která poskytuje rozhraní *Pythonu* pro tvorbu neuronových sítí. Funguje jako rozhraní pro knihovnu `TensorFlow`, která se specializuje na hluboké učení a detailní implementaci neuronových sítí.

Knihovna `TensorFlow`[7] je určena pro strojové učení a umělou inteligenci. Je široce používána pro trénink a implementaci hlubokých neuronových sítí a dalších složitějších modelů.

Kromě těchto knihoven jsou v projektu použity také další knihovny, jako například *Miscellaneous operating system interfaces* (`os`) a *System-specific parameters and functions* (`sys`). Tyto knihovny jsou součástí standardní knihovny *Pythonu* a z důvodu jejich široké funkčnosti nejsou zde podrobně popsány, ale jsou zmíněny pro úplnost.

Struktura modelu

Na níže uvedených řádcích je prezentována struktura modelu, na kterých je vystavěná celá aplikace. Tato sekce se detailně zaměřuje na popis siamské neuronové sítě a jejich vrstev, které jsou popsány, co dělají a z jakého důvodu jsou obsaženy. Na závěr sekce je zmíněn i výstup, jak by se aplikace měla chovat.

Základní struktura modelu je rozdělena do dvou hlavních částí. První část se zaměřuje na podsítě, především na extrakci vektorů z obrázků. Druhá část se věnuje výpočtu odlišnosti mezi těmito vektory. Celková struktura je znázorněna na obrázku 4.4, kde jsou jednotlivé vrstvy barevně odlišeny. Šipky s popisky na obrázku znázorňují velikost vstupů a výstupů jednotlivých vrstev.

Podsíť na extrakci vektorů je realizována ve formě konvoluční neuronové sítě. Tato konvoluční síť zpracovává vstupní snímky s rozlišením 100×100 pixelů a 3 barevnými kanály.

Před tím, než snímek vstoupí do konvoluční sítě, je nutné provést přípravu dat. Příprava zahrnuje následující kroky:

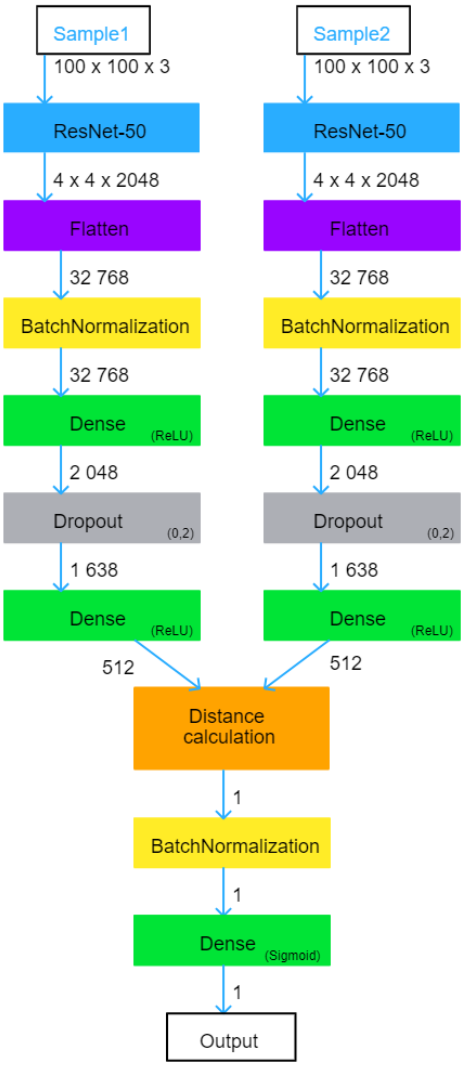
1. **Převedení obrazu z RGB do BGR:** Snímky jsou původně ve formátu RGB (červený, zelený, modrý). Při přípravě se obrazy převádějí do formátu BGR (modrý, zelený, červený). Tento krok je důležitý pro zajištění kompatibility s modelem a standardizaci dat.

2. **Normalizace barevných hodnot:** Po převodu do formátu BGR se hodnoty barevných kanálů upravují tak, aby měly průměrnou hodnotu nula. Toho se dosahuje odečtením průměrných hodnot barevných kanálů z každého pixelu. Tento krok pomáhá stabilizovat trénování sítě tím, že odstraní případné posuny v intenzitě barev, což usnadňuje rozpoznávání vzorců během průchodu konvolučními vrstvami.

Tyto přípravné kroky jsou zásadní pro zajištění kvalitních a konzistentních vstupních dat, které snižují šum a zlepšují výkon konvoluční sítě při extrakci relevantních rysů ze snímků.

Při vstupu do modelu je první vrstva založena na architektuře *ResNet-50*. *ResNet-50* se používá k zpracování snímku pomocí série 50 konvolučních vrstev. Model je blíže popsán

[5] https://scikit-learn.org/stable/
[6] https://keras.io/
[7] https://www.tensorflow.org/



Obrázek 4.4: Obrázek modelu.

v sekci 2.1. Výstupem z konvolučních vrstev modelu *ResNet-50* je multidimenzionální pole o velikosti $4 \times 4 \times 2048$. Výstupní pole obsahuje bohaté informace o snímku, které dále zpracovávají následující vrstvy modelu. Tyto rysy jsou klíčové pro další fáze analýzy, kde se porovnávají a hodnotí podobnost mezi snímky.

Další vrstva je *Flatten*, která všechny zpracované snímky z konvolučních vrstev ve formě trojrozměrného pole převede na jednorozměrné pole o délce, která je násobkem tří rozměrů předchozího pole, v tomto případě 32 768. Tímto způsobem je umožněno zpracování dalšími vrstvami.

Hodnoty přivedené z předchozí vrstvy jsou již seřazené v jednom poli, ale stále nejsou nijak upravené od konvoluce a mohou zahrnovat celé spektrum reálných čísel včetně extrémních hodnot, které jsou složité na zpracování plně propojenými vrstvami. K vyřešení popsaného problému je potřeba vrstva *BatchNormalization*, která normalizuje hodnoty a transformuje hodnoty blízké k nule. Na výstupu z této vrstvy poté nejsou extrémní hodnoty. Výstup této vrstvy je poté vhodný pro zpracování plně propojenými vrstvami.

Následující vrstvy jsou dvě plně propojené vrstvy a jedna vrstva *dropout*. Plně propojené vrstvy mají aktivační funkci ReLU. Tyto vrstvy se starají o výpočet vektorů pomocí učení z trénovací sady nastavováním vah a *bias*, kde v první vrstvě je 2048 neuronů a ve druhé je 512 neuronů. Vrstva *dropout* slouží ke snížení efektu *overfitting* tím, že vypustí některé perceptrony z hodnocení v průběhu učení. Vyřadí 10 % neuronů.

Ve druhé části se počítá odlišnost pomocí vzdálenosti. Hlavní vrstva této části je určena pro výpočet Euklidovy vzdálenosti mezi vektory z podsítí. Vzdálenost je počítána výrazem 4.1, kde x a y jsou pole vektorů podsítí, mezi kterými se vypočítává vzdálenost rozhodující o podobnosti.

$$d = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{4.1}$$

Další vrstvou je *BatchNormalization*, která se stará o transformaci vzdáleností, aby se u podobných snímků přibližovaly k nule a u odlišných snímků k jedničce. Dále pomáhá nastavitelným posunutím a zvětšením číselného spektra pro další vrstvu.

Pro snadnější nalezení mezní hodnoty, která rozhoduje jako dělič mezi stejnými a rozdílnými obličeji, je přidána ještě jedna vrstva. Vrstva je tvořena jedním perceptronem s aktivační funkcí *Sigmoid*, která hledá rozhodující *bias* a odděluje hodnoty více od sebe. Tato vrstva také nastavuje výstup v intervalu od nuly do jedné. Interval znázorňuje pravděpodobnost výsledku.

Předpokládaný výstup by měl být po těchto operacích okolo nuly, pokud se snímky podobají. Pokud jsou snímky rozdílné, výstup se pohybuje okolo jedné. Bohužel i přes všechny tyto operace je možné, že se některé vzorky budou pohybovat okolo poloviny vzdálenosti mezi nulou a jedničkou a budou obtížně oddělitelné z důvodu variability vstupních snímků. Tímto je možné zjistit, zda jsou odlišné nebo ne, a na výstupu programu je zvýraznit. Ve výsledku výstupu modelu je pravdivostní jednotka, která bude ovlivněna vstupními snímky.

Implementace programu

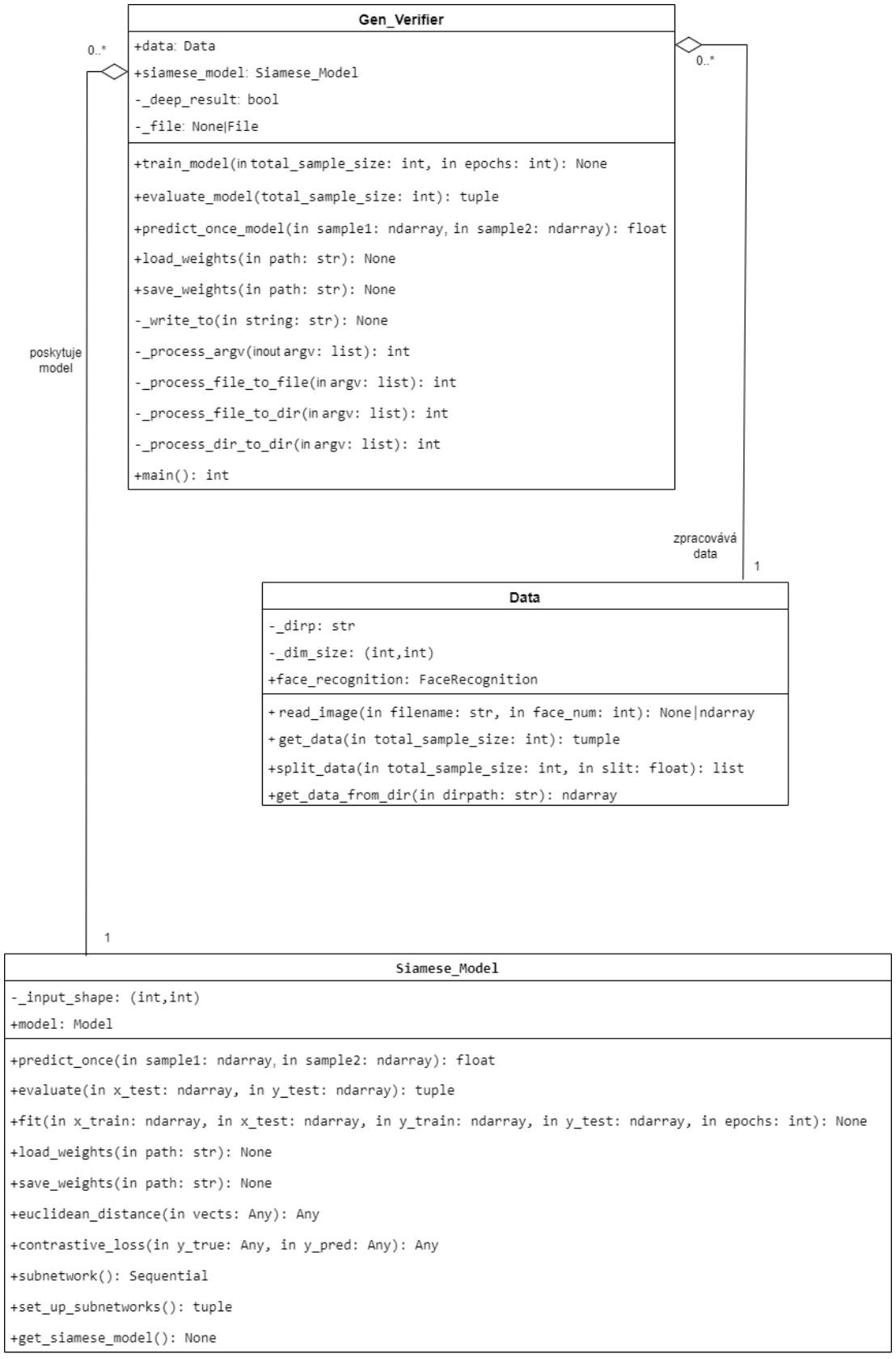
Tato sekce slouží k představení struktury a rozdělení funkcionality programu. V sekci jsou především představeny třídy a metody. Současně je níže popsán datový tok a možné problémy. Pod každou metodou je také uveden kód, který ukazuje její deklaraci spolu s argumenty.

Program je rozdělen do tří tříd: `Gen_Verifier`, `Data` a `Siamese_Model`, přičemž každá z těchto tříd má svoji úlohu. Tyto třídy mají svoji hierarchii, která umožňuje abstrahovat různé funkce v mnohem jednodušší formě, než jsou v původní funkci. Hlavní třídou v tomto programu je `Gen_Verifier` a zbylé dvě slouží k podpůrným funkcím, kde první abstrahuje model a druhá přístup k datové sadě a zpracování vstupních snímků. Tyto třídy jsou poté zobrazeny na UML diagramu, který je na obrázku 4.5.

Třída Data

Třída `Data` se zaměřuje na zpracování snímků a načítání vzorků z datové sady. Objekt této třídy obsahuje tři atributy a pět metod.

První atribut je `face_recognition`. Tato proměnná obsahuje model pro detekci snímků z knihovny `FaceAIKit`, která je použita dále v metodách. Je uložena v atributu, protože je



Obrázek 4.5: UML diagram tříd.

často používána a její inicializace trvá okolo 20 sekund na kancelářském stolním počítači, který pracuje na operačním systému *Windows* v prostředí *Jupyter Notebook* a má procesor Intel i5 8. generace s integrovanou grafikou. Tento počítač sice není moc výkonný, ale může reprezentovat nižší hranici počítačů, na kterém program může být používán. Jestliže by byla inicializována tato třída při každém spuštění metody na detekci obličeje u 1 000 snímků, trvalo by to bez optimalizací okolo 6 hodin. To je pouze započítán čas inicializace, nikoli vykonání kódu na detekci, nalezení a vystřižení obličeje ze snímku. Proto je tento atribut inicializován v konstruktoru a je nastaven na výstup z modelu *ArcFace*, protože budou porovnávány snímky stejné kvality, tudíž není nutné ho učit.

Druhý atribut je `_dim_size`. Do tohoto atributu jsou uloženy konečné rozměry snímku obličeje. Je inicializován v konstruktoru na výchozí hodnotu `(100,100)` s datovým typem `tuple`. Hodnotu je možné případně změnit.

Poslední atribut je alokován na uložení cesty k datové sadě. Atribut je používán v metodách na získání vzorků a je pojmenován `_dirp`. Jeho výchozí hodnota je nastavena na řetězec `'img/LFW - People/lfw_funneled'`.

První metoda třídy je konstruktor, který nastavuje tři předchozí atributy. Jeho dva argumenty jsou `dim_size` a `dirp`. Argument `dim_size` slouží k nastavení druhého atributu, který obsahuje rozměry snímku obličeje. Argument `dirp` nastaví cestu do složky, kde je uložena datová sada *LFW* s výchozí hodnotou `'img/LFW - People/lfw_funneled'`. Metoda je deklarována následujícím kódem:

```
__init__(self,dirp = 'img/LFW - People/lfw_funneled')
```

Další metodou je `read_image`. Metoda slouží k načtení snímku s jeho formátováním. Obsahuje argumenty `filename` a `face_num`, kde `filename` je cesta ke snímku. V argumentu `face_num` je výběr správného obličeje ze snímku, pokud jich je na snímku více. Cílem metody je zpracování snímku z uvedené cesty a příprava snímku pro potřeby modelu. Pro tuto funkci se snímek přečte z cesty a provede detekci obličeje. Pokud snímek obsahuje více než jeden obličej, tak si vybere požadovaný obličej. Následuje kontrola existence obličeje na snímku a jeho kvality. Pokud snímek nesplní vytyčené vlastnosti, tak se metoda vrátí s hodnotou `None`. Dále následuje transformace na požadovanou velikost, předzpracování snímku pro potřeby modelu a nakonec se v návratové hodnotě vrátí snímek. Pokud metoda selže z důvodu špatného vstupu, tak se opět vrátí s `None`. Tato metoda je deklarována kódem:

```
read_image(self,filename,face_num = 0)
```

Následuje metoda pro získání vzorků z datové sady. Tato metoda se jmenuje `get_data` a obsahuje argument `total_sample_size`, který určuje množství vzorků. Jejím úkolem je vytvořit skupinu vzorků z datové sady *LFW*. V datové sadě *LFW* se vzorky tvoří jako náhodně vybrané dvojice o velikosti specifikované argumentem `total_sample_size`, přičemž vzorky se stejným obličejem a vzorky s různými obličeji jsou zastoupeny rovnoměrně. Vzorky jsou seřazeny do dvou skupin za sebou a nejsou zamíchané. Tyto vzorky se ukládají do několika dimenzionálních polí a jsou poté vráceny jako návratová hodnota společně s polem o výsledcích jejich porovnávání v typu `tuple`, kde první člen je pole vzorků ve tvaru `[číslo vzorku, pořadí snímku ve vzorku (maximálně 2), výška, šířka, kanály]`. Výchozí nastavení výšky, šířky a kanálů je 100, 100 a 3. Pokud se nepodaří získat takovou sadu vzorků, vrátí se `None`. Hlavička metody je definována tímto kódem:

```
get_data(self, total_sample_size)
```

`Split_data` je metoda, která vytvoří náhodně zamíchanou vzorkovou sadu o velikosti `total_sample_size`. Metoda rozdělí vzorkovou sadu v poměru `split` pro testování. Základní nastavení je na hodnotě 0,2, což znamená, že 20 % vzorků bude použito pro testování. Navrací čtyři hodnoty: `x_train`, `x_test`, `y_train` a `y_test`, kde `x_train` s `x_test` obsahují vzorky pro model a `y_train` s `y_test` obsahují výsledky, které mají být trénovány nebo dosaženy. Deklarace této funkce je dána tímto kódem:

```
split_data(self, total_sample_size, slit =.2)
```

Poslední metoda je `get_data_from_dir`, která načte všechny snímky ve složce `dirpath` a následně je vrátí jako návratovou hodnotu. Předpis hlavičky metody je:

```
get_data_from_dir(self, dirpath)
```

Třída Siamese model

Druhá třída je `Siamese_Model`, která obsluhuje a abstrahuje model siamské sítě. Tento model slouží k tvorbě a nahrávání vah modelu. Dále slouží k trénování, vyhodnocování a předpovídání hodnot. Objekt této třídy má dva atributy a osm metod.

První z atributů této třídy je `_input_shape`. Do atributu `_input_shape` jsou uloženy vlastnosti vstupu do modelu a jeho rozměry. Výchozí nastavení je na `(100,100)`.

Druhý atribut je `model` datového typu třídy `Model`, který je základní jednotkou celé třídy. Třída `Model` je z knihovny `Keras`, která je založena na neuronových sítích. Z toho důvodu je třída `Siamese_Model` pouze zapouzdřením nad třídou `Model`.

První metodou je opět konstruktor, který nastavuje `_input_shape` přes argument se stejným jménem. Atribut `_input_shape` může být nastaven na výchozí hodnotu. Dále se zde vytvoří a zkompiluje model siamské sítě, který je uložený v atributu `model`. Pokud je atribut `input_shape` nastaven na výchozí hodnotu, je zde možnost nastavení argumentu `load` a třída automaticky načte váhy a *bias* do modelu. Pokud se však rozměry vstupu v argumentu změní, tuto možnost už nelze využít. Tato metoda je deklarována tímto kódem:

```
__init__(self, input_shape = (100,100),load = True)
```

Druhá metoda je `predict_once`, která ze dvou snímků zjistí číslo, které předpoví, zda jsou obličeje na nich podobné nebo ne. Snímky se dosazují do argumentů `sample1` a `sample2`. Tato funkce vrací číslo od nuly do jedné, kde jednička znamená stejnou osobu a nula představuje rozdílnou osobu. Hlavička metody má předpis:

```
predict_once(self,sample1, sample2)
```

Metoda `evaluate` slouží k vyhodnocení přesnosti sítě a vypočítání ztrátové funkce. Jejími argumenty jsou `x_test` a `y_test`, kde `x_test` je pětirozměrné pole se vzorky a `y_test` je jednorozměrné pole s výsledky se stejným indexováním. Návratová hodnota se skládá ze dvou číselných hodnot. První je hodnota ztrátové funkce a druhá je hodnota přesnosti. Je nutné podotknout, že výsledky z modelu jsou inverzní. Podrobnější vysvětlení je v následující sekci Učení modelu 4.3. Deklarace této funkce je dána tímto kódem:

```
evaluate(self,x_test,y_test)
```

Metoda `fit` je určena k trénování modelu na základě vzorků `x_train`, `x_test`, `y_train` a `y_test`. Vzorky `x_train` a jejich výsledky `y_train` se používají k trénování modelu, zatímco `x_test` a jejich výsledky `y_test` slouží k ověření přesnosti modelu. Poslední argument `epochs` určuje, kolikrát se má učit na skupině vzorků. Předpis hlavičky metody je:

```
fit(self,x_train, x_test, y_train, y_test, epochs = 10)
```

Metody `load_weights` a `save_weights` slouží k práci s ukládáním a nahráváním vah modelu na disk, kde atribut `path` určuje cestu k souboru.

```
load_weights(self,path)
save_weights(self,path)
```

Metoda `euclidean_distance` je statická funkce pro výpočet eukleidovské vzdálenosti mezi dvěma vektory. Argument `vects` je pole dvou vektorů, mezi kterými bude eukleidovská vzdálenost počítána. Výraz pro výpočet eukleidovské vzdálenosti je možné najít v sekci Struktura modelu 4.3 a je popsán rovnicí 4.1.

```
euclidean_distance(vects)
```

Metoda `contrastive_loss` je statická funkce, která slouží k výpočtu ztrátové funkce *contrastive loss*.

```
contrastive_loss(y_true, y_pred)
```

Poslední tři metody `subnetwork`, `set_up_subnetworks` a `get_siamese_model` se soustředí kolem definování siamské neuronové sítě, která je popsána v sekci Struktura modelu 4.3.

Metoda `subnetwork` slouží k vytvoření konvoluční podsítě pro zpracování vstupních snímků. Výstup této metody je instance třídy `Model`. Je deklarována tímto kódem:

```
subnetwork(self)
```

Z konvolučních podsítí, které jsou udělané v metodě `subnetwork`, se vytvoří dvě instance, které se spojí v metodě `set_up_subnetworks`. Metoda `set_up_subnetworks` je používána na vytvoření dvou vstupů do siamské sítě a zajištění jejich správné funkce pomocí sdílení vah. Její hlavička vypadá následovně:

```
set_up_subnetworks(self)
```

Metoda `get_siamese_model` nakonec vytvoří instanci třídy `Model`, která reprezentuje siamskou neuronovou síť. Je vytvořena z metody `set_up_subnetworks`, která vytvoří vstupy. Dále jsou přidány poslední vrstvy pro výpočet euklidovské vzdálenosti a optimalizaci výstupu. Celý tento proces je završen sestavením siamské sítě se ztrátovou funkcí *contrastive loss* a optimalizátorem *RMSprop*. Je deklarována kódem:

```
get_siamese_model(self)
```

Třída Gen Verifier

`Gen Verifier` je hlavní třída programu, která se stará o propojení tříd `Siamese_Model` a `Data` pomocí výměny proměnných. Další funkcí třídy je ovládání celého programu a zpracování vstupu v podobě argumentů programu. Dále má třída na starost vypočítat výsledek a vypsat ho na stanovené místo. Tato třída obsahuje čtyři atributy a třináct metod.

První dva atributy jsou `model` a `siamese_model`. Oba atributy jsou instance stejnojmenných tříd, které vykonávají funkce popsané v předešlých podsekcích 4.3 a 4.3. Oba atributy jsou vytvořeny v konstruktoru pro snazší ovladatelnost.

Druhé dva atributy jsou `_deep_result` a `_file`. Tyto atributy jsou nastavovány dle argumentů programu. Proměnná `_deep_result` je pravdivostním typem a nese v sobě informaci o podrobnosti výstupu, který je popsán v podsekci Funkcionalita 4.1. Atribut `_file`

je určen k zaznamenání výstupu programu. Pokud je nastavena na `None`, výsledky se vypisují na standardní výstup. Pokud neobsahuje hodnotu `None`, tak výstup programu je zapsán do souboru, který je pod tímto atributem uložen.

Konstruktor v této třídě má jeden argument, který nastavuje adresu složky pro datovou sadu od třídy `Data`. Konstruktor nastavuje atributy objektu na výchozí hodnoty. Atributy `data` a `siamese_model` jsou nastaveny jako objekty stejnojmenných tříd a atributy `_deep_result` a `_file` jsou nastaveny na `False` a `None`.

Metoda `train_model` se stará o učení modelu, přípravu dat a výpis výsledku. Má dva argumenty: `total_sample_size` a `epochs`. Argument `total_sample_size` nastavuje počet vzorků, na kterých se má model učit. Argument `epochs` nastavuje počet kol, během kterých se má model učit. Výchozí nastavení je 10 kol. Tato metoda je deklarována kódem:

```
train_model(self,total_sample_size, epochs = 10)
```

Metoda `evaluate_model` poskytuje možnost zjistit hodnotu ztrátové funkce a přesnost modelu, kde přesnost modelu je inverzní. Dále argumentem `total_sample_size` poskytuje možnost kontrolovat množství vzorků, na kterých je model testován. Výchozí nastavení je 1000 vzorků. Hlavička metody má tvar:

```
evaluate_model(self,total_sample_size = 1000)
```

Čtvrtá metoda je `predict_once_model`, která ze dvou snímků získá číslo, které předpoví, zda si jsou osoby na nich podobné, či nikoliv. Snímky se vkládají do argumentů `sample1` a `sample2`. Metoda vrací číslo od nuly do jedné, kde jednička znamená stejnou osobu a nula představuje rozdílnou osobu. Metoda zapouzdřuje funkcionalitu metody `predict_once` ze třídy `Data`. Hlavička metody je:

```
predict_once_model(self,sample1, sample2)
```

Metody `load_weights` a `save_weights` zapouzdřují metody ze třídy `Siamese_Model` pro snadný a přehledný přístup. Jsou deklarované kódem:

```
load_weights(self,path)
save_weights(self,path)
```

Metoda se jménem `_write_to` slouží pro výpis výsledků programu podle nastaveného umístění v atributu `_file`. Pokud je `None`, výstup se napíše na standardní výstup. Hlavička metody má tvar:

```
_write_to(self,string)
```

Metoda `_process_argv` zpracovává argumenty programu a nastavuje cílové umístění výstupu společně s informací o podrobnosti výstupu. Argumenty jsou považovány za validní, pokud se vrátí návratová hodnota nula, jinak nastala nějaká chyba při zpracování. Definice hlavičky metody je:

```
_process_argv(self,argv)
```

Metody `_process_file_to_file`, `_process_file_to_dir` a i `_process_dir_to_dir` slouží ke zpracování snímků na základě vstupních argumentů programu a výstup je poté vypsán podle módu, ve kterém jsou zaslané snímky. Tyto módy jsou popsány v podsekci Funkčnost 4.1. Obecně zpracovávají a vypisují výsledky podle předzpracovaných dat. Hlavičky metod jsou:

```
process_file_to_file(self,argv)
process_file_to_dir(self,argv)
process_dir_to_dir(self,argv)
```

Poslední metoda je `main`. Tato metoda slouží ke spuštění modelu pro rozpoznávání osob z příkazového řádku a tudíž i k práci s celým programem. Do metody je zařazena většina metod, které byly použity a výše popsány. Nejsou do ní zařazeny metody pro učení a validaci, protože to není hlavní účel této metody a programu.

Tok programu

Aplikace je konstruována ke spuštění jedním příkazem s nutností inicializace třídy. V tomto příkazu se vykonává vyhodnocení podobnosti, zpracování argumentů a výpis výsledků. Pro názornost je kód na spuštění uveden níže.

```
from Classes.Gen_Verifier import Gen_Verifier

verifier = Gen_Verifier()
verifier.main()
```

Nejdříve probíhá inicializace, která je popsána v předešlé podsekci. Atributy v třídě `Gen Verifier` jsou nastaveny v konstruktoru. Následně proběhne metoda `main`, která předá kontrolu nad programem metodě `process_argv`. Metoda `process_argv` zpracuje argumenty a poté předá kontrolu opět metodě `main`. Metoda `main` podle typu prvních dvou argumentů spustí jeden ze tří módů s příslušnou metodou `process_file_to_file`, `process_file_to_dir`, nebo `process_dir_to_dir`, které mají vestavěný rozeznávací model a výpis hlášek. Poté se vše ukončí s návratovým kódem 0 při úspěchu a jiným číslem při neúspěchu.

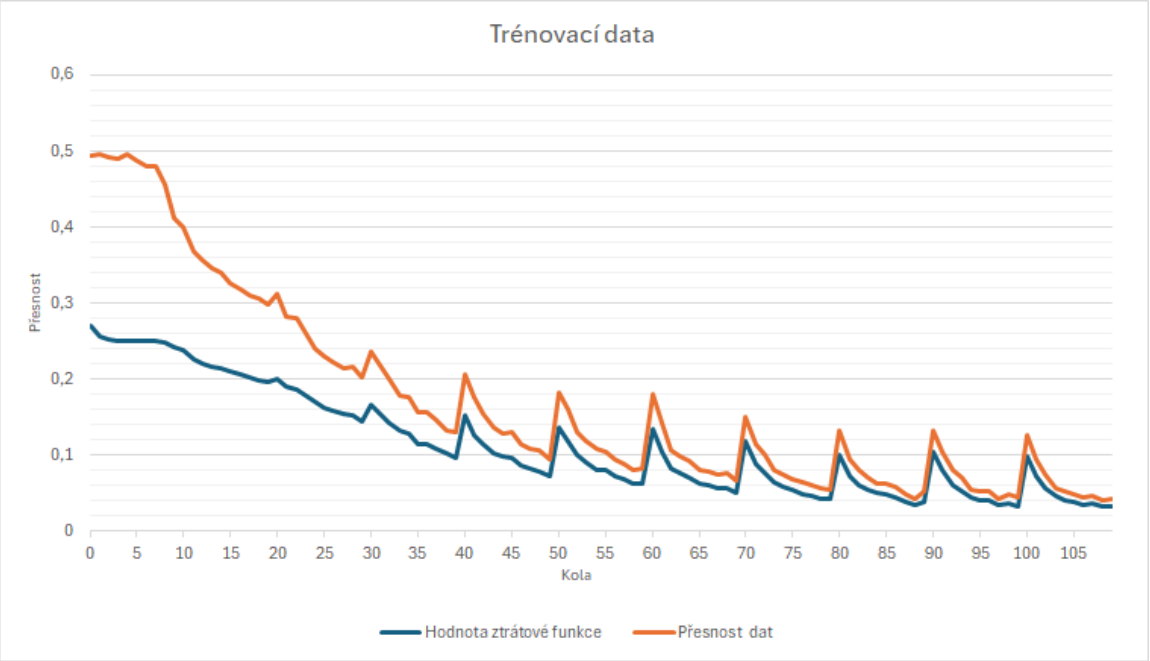
Trénování modelu

Neuronové sítě jsou velmi pokročilým nástrojem pro analýzu složitých dat, jako jsou snímky. Tyto sítě se nastavují pomocí *bias* a vah, které jsou systematicky tvořeny učením. Těchto vah a *bias* je i v nejjednodušších neuronových sítích několik stovek, což způsobuje, že jedinou možností zlepšení přesnosti modelu je učení. Základem učení je trénovat model do největší přesnosti. Nicméně model nesmí být trénován moc ani málo. Je-li model natrénovaný málo, tak má nižší přesnost, než by mohl mít celkově. Na druhé straně, pokud je přetrénován, model začne ztrácet svoji obecnost a přesnost klesá také. Tyto jevy se nazývají *underfitting* – nedostatečné trénování, a *overfitting* – přetrénování modelu.

Model popsaný v této práci byl učen na sekvencích 8 000 vzorků, které jsou rozděleny na deset kol. Počet sekvencí, které jsou na tento model použity, je jedenáct. Na konečné váhy je použito jen deset sekvencí.

Na obrázcích 4.6 a 4.7 jsou znázorněny grafy, které reprezentují učení modelu s vyznačenou hodnotou ztrátové funkce a přesnosti. Na první pohled je vidět, že hodnota ztrátové funkce modelu klesá, ale zároveň klesá i hodnota přesnosti. Toto je způsobeno kombinací tohoto modelu a ztrátové funkce, která se snaží co nejvíce stlačit výsledky k nule, čímž se vytvoří invertovaný model. Jelikož jsou vstupem modelu dva snímky a výstup je od nuly do jedné, tak při náhodném výběru dostaneme 50 % šanci, že se trefíme, jak je možné vidět na počátku těchto grafů. Tím pádem, pokud začne přesnost klesat nebo i stoupat, model se stává vždy přesnější, akorát jeho hodnoty v klesajícím případě jsou inverzní. Model by se stal nefunkčním pouze, pokud by přesnost stále oscilovala okolo 50 % a neklesala by ani nestoupala.

Dále je vidět na trénovacích datech z grafu 4.6, jak klesají hodnoty ztrátové funkce i přesnosti, ale v grafu jsou viditelné výstupky. Tyto výstupky jsou způsobeny kombinací



Obrázek 4.6: Graf vývoje trénovacích dat.

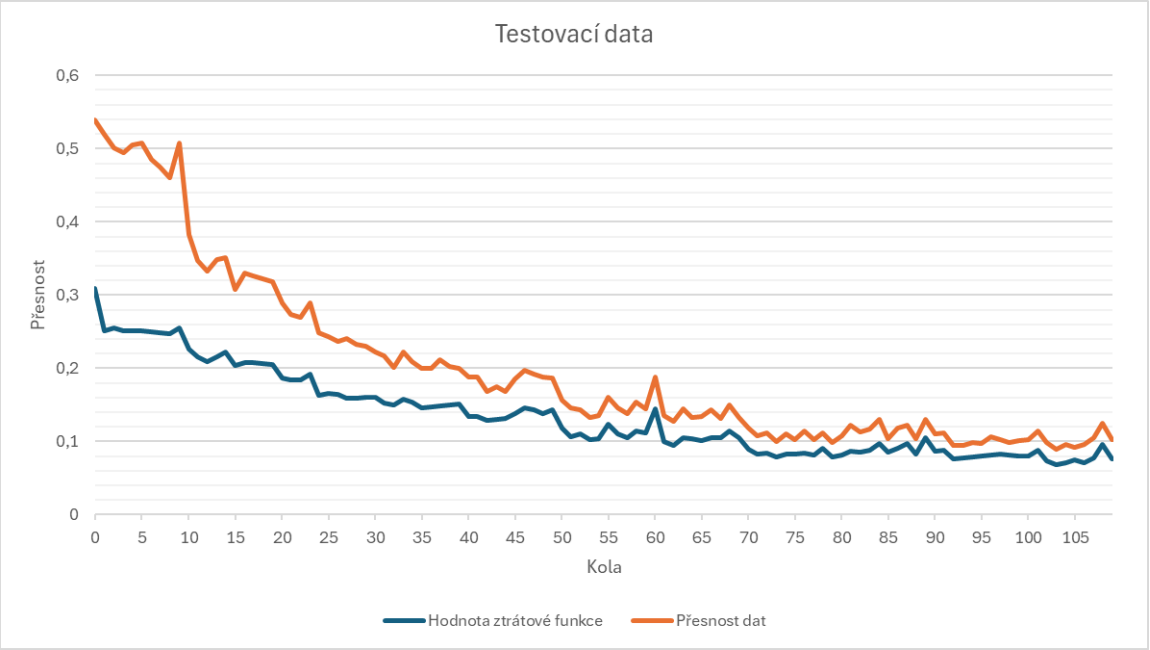
způsobu trénování a struktury modelu. Konkrétněji to při učení způsobují tři faktory – nastavení parametrů kol (*epochs*), velikost trénovací skupiny (*batch_size*) a primárně vrstva *BatchNormalization*. Tato vrstva, jak bylo zmíněno dříve, slouží k normalizaci dat. K normalizaci používá průměr z trénovací skupiny. Problém nastane, když se zmenší velikost trénovací skupiny. V tom případě, vždy když se na začátku trénování (spouštění metody `fit`) nahrají data potřebná pro normalizaci, hodnota ztrátové funkce se zvětší. Je to způsobeno nedostatkem dat a směrodatná odchylka s průměrem se nevypočítají správně. Poté se s dalším kolem data zlepší díky příchodu nových dat, která se zprůměrují a hodnota ztrátové funkce se zase zmenší. Toto se opakuje s každým spuštěním trénovaní. Tento problém neovlivnil příliš tento model, protože počet kol byl 10 a ty dokázaly vykompenzovat tuto ztrátu i s malým zlepšením. Důvod, proč to nebylo vidět na obrázku 4.7 s testovacími daty, je ten, že tato vrstva má jiný výpočet při testování, společně s potřebnými daty.

Ustalování je lépe vidět na datech z grafu 4.7, kde je přesnost zobrazena na testovacích datech. Při každé sekvenci model odložil 20 % vzorků stranou a provádí na nich testování hodnoty ztrátové funkce a přesnosti modelu. Tento přístup umožňuje zjistit, zda v modelu nenastává *overfitting*, neboli zaměřování modelu na jeden typ vzorků, nikoliv na celek. Naštěstí k tomuto chování v tomto obecném modelu zatím nedošlo. Je patrné, že tento trend se ještě neobjevil, protože model je trénován sekvenčně a současně se zatím nedostal přes kritickou hodnotu.

Model byl hodnocen na základě datové sady *LFW*, kde získal přesnost až 90,7 %. Pro toto hodnocení se použil model trénovaný během 100 *epoch*.

Chybové hlášky

Při zadání nesprávného vstupu do programu: program vrátí kódy chyby a na standardní chybový výstup vypíše chybovou hlášku popisující problém, který nastal.



Obrázek 4.7: Graf vývoje testovacích dat.

Návratové kódy

0: Program byl úspěšně ukončen.

1: Chyba v počtu argumentů. Počet argumentů se liší od očekávaného.

2: Nesprávné použití argumentu `-dr`. Tento argument byl použit více než jednou.

3: Chybný formát argumentu pro výpis do souboru. Soubor nekončí příponou *.txt*, nebo soubor nelze otevřít.

4: Chybný formát argumentu `file_path_A`. Soubor nelze správně otevřít nebo formátovat.

5: Chybný formát argumentu `file_path_B`. Soubor nelze správně otevřít nebo formátovat.

4.4 Instalace a spuštění aplikace

Program byl navržen pro použití z konzole. Pro dosažení nejlepších výsledků je však nezbytné trénovat jej na specificky modifikovaných datových sadách. Proto jsem se rozhodl jej šířit jako knihovnu, která obsahuje model s modifikovatelnými váhami. Tato knihovna může být také použita jako samostatná aplikace po správné instalaci.

Implementovaný program lze používat dvěma způsoby. První a hlavní způsob, pro který byl vyvinut, je použití z příkazové řádky jako konzolová aplikace. Druhý způsob použití v kódu jako knihovna, kde lze model přeučit a nastavit nové váhy.

Pro použití je nejprve nutné program nainstalovat. Jako první se instaluje prostředí programovacího jazyka. Konkrétně je nutné mít nainstalovaný *Python* verze 3.11.5. Dále

je třeba získat potřebné knihovny, které jsou uvedeny v souboru `requirements.txt`. Tyto knihovny lze snadno nainstalovat pomocí následujícího příkazu v příkazové řádce:

```
python -m pip install -r requirements.txt
```

Při instalaci knihoven je důležité dbát na výstupy, protože některé knihovny mohou vypisovat další požadavky. Například může být potřeba přidat některé cesty složek do systémové proměnné `PATH`.

Po instalaci lze program použít jako konzolovou aplikaci. Příkaz pro zobrazení nápovědy z kořenové složky je uveden níže:

```
python -m Gen_Verifier --help
```

Druhým způsobem lze program integrovat jako knihovnu do jiného projektu a používat model přímo pro výpočty. Stačí složku `Gen_Verifier` zkopírovat společně s váhami z kořenové složky a vložit je do jiného projektu. Tento způsob také umožňuje použití různých metod, například pro opětovné učení, při kterém se může použít méně realistická datová sada. Toto bude mít za následek, že model bude více benevolentní. Dále bude možnost modifikovat model, a tím více usnadnit zlepšení jeho kvality. Příklady jak použít knihovnu jsou v části Implementace programu 4.3.

Seznam souborů je ve struktuře adresáře, kde první složka s názvem `Gen_Verifier` slouží k uložení tříd a struktur. Druhá složka s názvem `saves` obsahuje váhy pro model. Další obsažený soubor je `README.md`, který obsahuje příručku k použití programu. Poslední soubor je `requirements.txt`, který obsahuje knihovny nutné k instalaci před použitím.

Kapitola 5

Testování a experimenty

Testování je nejdůležitější část vývoje softwaru i hardwaru, neboť nám umožňuje porovnávání předpokládaného výstupu s výstupem z určitých celků programu za daných podmínek. Z tohoto je poté možné zjistit, zda model funguje správně, nebo zda někde nejsou chyby. Obecně testování ověřuje schopnosti programu, které by měl mít. Oproti tomu experimenty hledají nové řešení a prozkoumávají nedostatky v testování.

V této kapitole je model testován za účelem ověření jeho správné funkčnosti vůči jeho předpokládanému výstupu. U reálných snímků je provedena analýza hodnot na výstupu a následné porovnání s již zavedeným řešením. Nakonec testování reálných snímků je zhodnocení přesnosti a kvality. Toto vše je tématem první sekce 5.1. Následující sekce 5.2 se zabývá prozkoumáním funkčnosti na vygenerovaných snímcích. Poslední sekce 5.3 se zaměřuje na ladění, kterým tento model prošel, aby získal požadované výsledky.

5.1 Testovaní a porovnávání modelu na reálných snímcích

Pro zjištění kvality je důležité porovnat stávající model s ostatními modely, neboť je složité zjistit, jak si vede model mezi ostatními, když není testován na stejných vzorcích. Proto je model nejprve testován na reálných snímcích, protože na nich byl trénován.

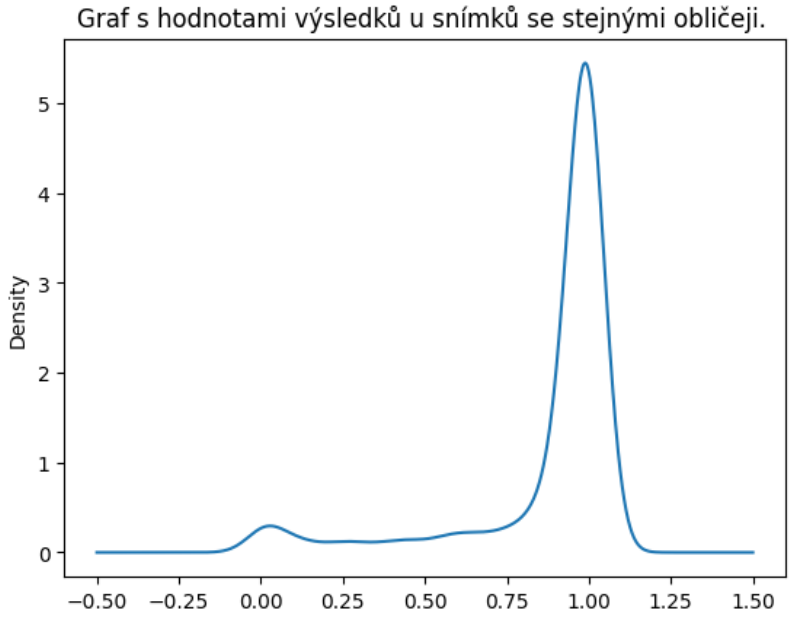
Pro tento úkol používám modul z knihovny `FaceAIKit`, který byl učen se ztrátovou funkcí *ArcFace*. Na vstupních snímcích jsou provedeny stejné úpravy jako na vstupu do stávajícího modelu, a to jsou nalezení obličeje, oříznutí obličeje ze snímku a transformace rozlišení snímku obličeje.

Pro porovnání jsou vytvořeny čtyři grafy, na kterých jsou zobrazeny vlastnosti výstupu. Tyto grafy jsou získány ze dvou sad vzorků, které jsou vytvořeny podle snímků osob se stejnými obličeji a osob s rozdílnými obličeji. Počet vzorků je 2 000 na každou sadu.

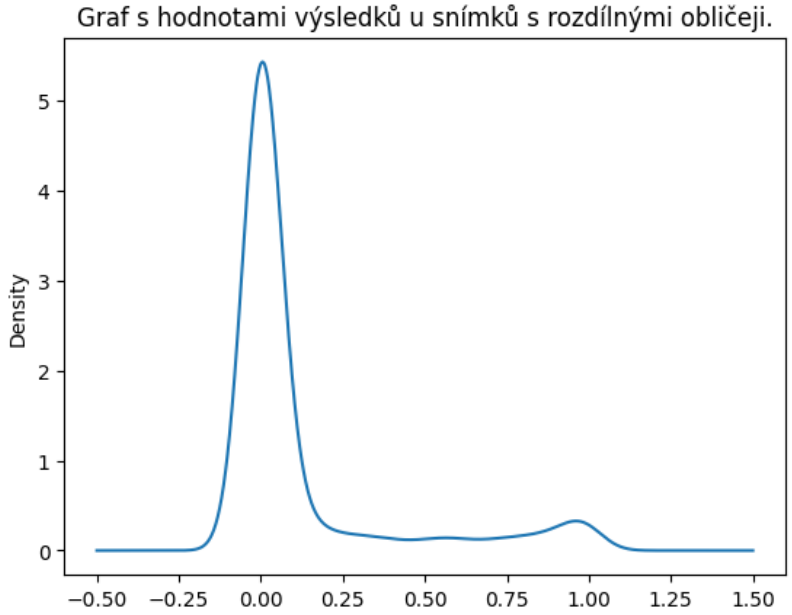
Výsledky modelu Gen Verifier

Výsledky modelu `Gen_Verifier` jsou zpracovány v grafech typu *Kernel density estimate* (*KDE*) se jmény Graf s hodnotami výsledků u snímků se stejnými obličeji a Graf s hodnotami výsledků u snímků s rozdílnými obličeji. Tyto grafy jsou na obrázcích 5.1 a 5.2, kde je vidět, jak se chová výstup, který se má pohybovat kolem 0 nebo 1.

Z těchto grafů je vidět, že model je invertovaný, jak je zmíněno v sekci Implementace 4.3, tudíž má hodnotu jedna na výstupu u snímků se stejnými obličeji a hodnotu nula u snímků s rozdílnými obličeji.



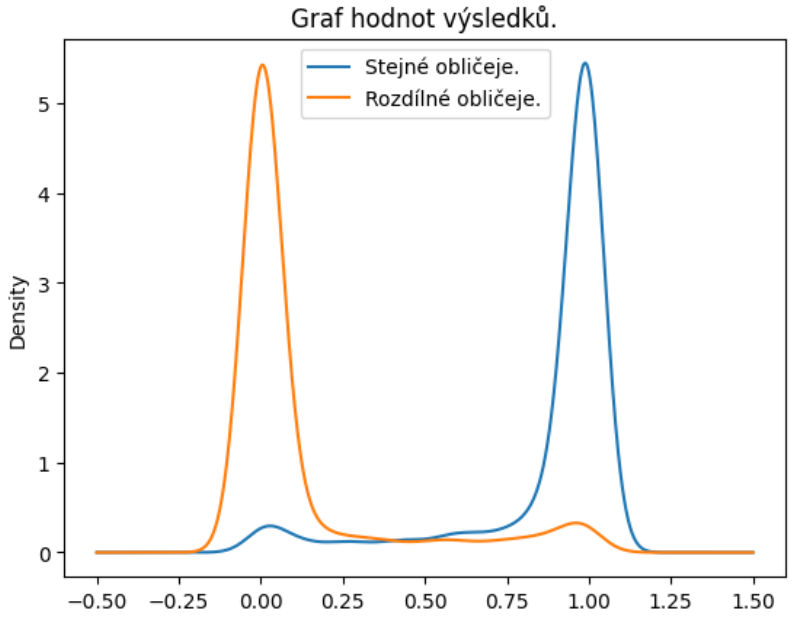
Obrázek 5.1: Graf s hodnotami výsledků u snímků se stejnými obličeji modelu Gen Verifier.



Obrázek 5.2: Graf s hodnotami výsledků u snímků s rozdílnými obličeji modelu Gen Verifier.

Další chtěnou vlastností je, že má přesně oddělené hodnoty mezi typy dvojic snímků ve vzorcích až na chybové vzorky, které se přesně promítly na opačnou část spektra.

Hodnoty se primárně pohybují okolo nuly a jedné. Pod nulu a nad jednu se nic nevyskytuje, ale malé množství dat se pohybuje mezi nulou a jednou, konkrétněji mezi hodnotami 0,2 až 0,8. Tudíž většina hodnot se soustředí do dvou konkrétních bodů.



Obrázek 5.3: Graf hodnot výsledků modelu Gen Verifier.

Pro nalezení rozhodující hodnoty je použit graf, který sloučil dva předešlé a průsečíkem hodnot v intervalu od nuly do jedné se nachází mezní hodnota. Tento graf je zobrazen na obrázku 5.3 a zobrazuje pásmo podobných hodnot výsledků kolem hodnoty 0,41.

Výsledky z modelu FaceAIKit

Výsledky těchto dat jsou v grafu Graf hodnot výsledků modelu *FaceAIKit* na obrázku 5.4, který obsahuje obě křivky, protože už není nutné zobrazovat průběh obou grafů.

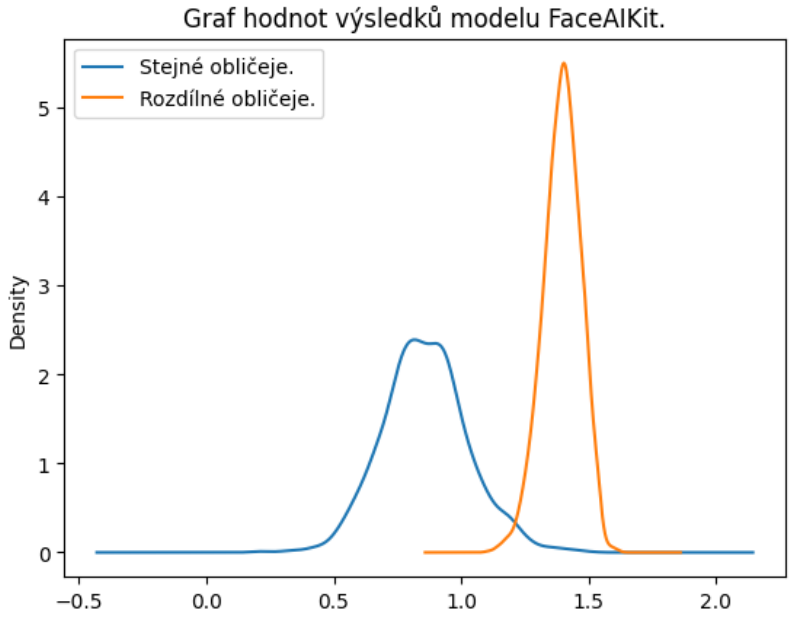
Z tohoto obrázku je vidět, že se hodnoty u obou typů pohybují na kladné ose grafu, kde se pohybují přibližně okolo stejných hodnot. Rozdílné obličeje se pohybují okolo hodnoty 1,4 a mají malou směrodatnou odchylku, na rozdíl od stejných obličejů, které se rozdělily do tří skupin. První dvě jsou blízko u sebe a mají hodnoty 0,8 a 0,9. Jejich směrodatná odchylka se také zvětšila. Poslední skupinkou, kde se obličeje rozmístily, vrcholí u hodnoty 1,1, která obsahuje jen málo obličejů. Tyto hodnoty mohou být způsobeny nedokonalostí datové sady nebo nízkou chybovostí modelu.

Jelikož data nejsou nijak omezeny, taj jsou roztroušena pouze do dvou Gaussových křivek, které se protínají v jednom bodě mezi vrcholy. Průsečík je v bodě s hodnotou 1,14, kterou lze používat jako mezní bod.

Celkové výsledky

Pro celkové hodnocení úspěšnosti je model rozdělen na dvě skupiny vypočítané pomocí mezního bodu, který byl vyobrazen v předchozích podsekcích u každého modelu.

Výsledek je možné spočítat z předchozích hodnot, kde se použijí hodnoty ze vzorků se stejnými obličeji a spočítá se počet vzorků, které spadají pod hranici. To samé se udělá u vzorků s rozdílnými obličeji, ale spočítají se všechny vzorky, které spadají nad hranici. Proces je opakován pro druhý model.



Obrázek 5.4: Graf hodnot výsledků modelu FaceAIKit.

Výsledky těchto modelů jsou následující. Přesnost modelu `Gen_Verifier` je 90,7 % a přesnost modelu `FaceAIKit` je 97,13 %. Z toho lze usoudit, že výsledky tohoto modulu nejsou lepší než z knihovny `FaceAIKit`, ale jsou stále uspokojivé. Naproti tomu model, který byl vytvořen v této práci, docílil většího prostoru mezi vzorky se stejnými obličeji a rozdílnými obličeji. Dále jeho hodnoty spadaly do intervalu od nuly do jedné, kde se nejvíce vzorků pohybovalo v okolí nuly a jedné, což způsobilo, že model není tolik závislý na hledání hranice a může tam dosadit i polovinu intervalu bez větších ztrát přesnosti.

Podrobné porovnání

U podrobnějšího výpisu model `Gen_Verifier` dospěl celkově k 9,3 % chybovosti, kde úspěšnost u vzorků se stejnými obličeji je 91,5 % a u vzorků s rozdílnými obličeji je 89,9 % . Dále jsou vzorky rozděleny na správně a falešně ohodnocené. Nově vzniklé skupiny se nazývají správně pozitivní, falešně pozitivní, falešně negativní a správně negativní, kde pozitivní vzorky reprezentují stejné snímky a negativní vzorky reprezentují rozdílné snímky. Ze skupin z předchozího testování je dále vybráno 100 vzorků pro každou skupinu a následně jsou vyhodnocovány, jak moc a v čem jsou stejné nebo odlišné, aby se zajistily podrobnější informace o modelu.

První skupina vzorků se stejnými obličeji je rozdělená na správně pozitivní a falešně negativní vzorky. U těchto vzorků je testovaná rozdílnost mezi stejnými obličeji. Konkrétněji je zjišťováno porovnávání obličejové mimiky (M), pohled na obličej (P), výskyt brýlí, cenností a různých pokrývek hlavy (C). Dále bylo zkoumáno zakrytí části obličeje (Z), zhoršená kvalita (K), rozdílný věk obličeje (S), nebo přítomnost vlasů nebo vousů v obličeji (O). Poslední je změna výskytu vousů nebo vlasů mezi snímky (V) a změna barevnosti obličeje kvůli barevnému filtru nebo vnějším podmínkám, jako je blesk (B). Výsledky těchto testů jsou zobrazeny v tabulce 5.1, kde jednotlivá písmena znázorňují výše popsané vlastnosti.

(%)	Správně pozitivní	Falešně negativní
M	30	56
P	31	38
C	8	25
Z	1	17
K	5	3
S	3	2
O	7	7
V	0	12
B	7	29

Tabulka 5.1: Tabulka výsledků rozdílnosti u stejných obličejů.

Z těchto výsledků je patrné, že model měl občas potíže s hodnocením mimiky obličeje, nošením cenností, barevností, zakrytím části obličeje a se změnou vousů nebo vlasů. Příklady těchto vzorků jsou zobrazeny na obrázcích 5.5 a 5.6.

U negativních vzorů je provedeno stejné ohodnocení, které je uvedeno podle tabulky 5.2. Na základě těchto výsledků lze zjisti, jak skupiny vzorků jsou závislé na vnějších vlivech a jak tyto vlivy ovlivňují model.

(%)	Falešně pozitivní	Správně negativní
M	53	62
P	44	41
C	30	37
Z	15	15
K	8	9
S	14	6
O	10	11
V	23	19
B	42	43

Tabulka 5.2: Tabulka výsledků rozdílnosti u rozdílných obličejů.

Z této tabulky je patrné, že vnější vlivy jsou poměrně stejné a chybovost je primárně způsobena špatným rozeznáváním obličeje. Příklady těchto vzorků jsou na obrázcích 5.7 a 5.8. Zajímavé je porovnání negativních snímků s falešně negativními vzorky. Ukazuje se, že některá procenta jsou velmi podobná. Tudíž například pokud obličeje mají jiný výraz, mohou být přesto přiřazeny do skupiny negativních, i když ve skutečnosti jsou pozitivní. Tento jev se podobně projevuje u pohledu na obličej, cenností a brýlí. Naopak je zajímavé, že ve skupině falešně pozitivních obličejů je více přiřazených obličejů, které mají změnu vousů, než ve skupině správně negativních obličejů. Tento rozdíl však může být způsoben náhodným výběrem, protože rozdíl 4 % není příliš výrazný.

Celkově je vidět z těchto výsledků, že model poměrně dobře pracuje se stejnými snímky, pokud se obličeje výrazně nemění v mimice, pohledu nebo v přítomnosti brýlí a cenností. Oproti tomu rozdílné obličeje mají tyto vlastnosti poměrné vyvážené a model má větší problémy přímo s rozeznáváním obličejů, ale pokud poukázáno na příkladové obrázky, tak je většina velmi podobná, což může být způsobeno stárnutím některých obličejů nebo přidáním nějakých cenností.

Obrázek 5.5: Správně pozitivní vzorky.

Obrázek 5.6: Falešně negativní vzorky.

Obrázek 5.7: Falešně pozitivní vzorky.

Obrázek 5.8: Správně negativní vzorky.

5.2 Testování modelu na generovaných syntetických snímcích

Hlavním cílem tohoto projektu je vytvořit program na vyhodnocení věrohodnosti obličejů u synteticky generovaných snímků, a proto jsou doposud nabyté zkušenosti z předchozích kapitol použity k porovnávání identit na snímcích. Zaměření je na analýze snímků, vytvoření postupu a shrnutí výsledků z provedeného porovnání v modelu *StarGAN*.

Analýza datové sady

Pro vyhodnocení věrohodnosti je použita datová sada, kterou jsem vytvořil podle sekce Příprava dat 3.4 ve druhé části. Tato sada obsahuje dvojici originálního snímku a vygenerovaného po 2 000 vzorcích s rozlišením 128×128. V tomto případě jsou takové datové sady snímků vytvořeny tři a každá obsahuje změnu jiného atributu pro zlepšení vyobrazení pohledu na celý model *StarGAN*.

První datová sada je tvořena změnou atributu barvy vlasů. Každý vstupní snímek je analyzován a je mu změněna barva vlasů na hnědou aniž by se měnil účes. Tato činnost by neměla měnit žádné vlastnosti obličeje, a tím pádem by neměla měnit ani obličej na snímku. Je možné, že se obličej nějak změní kvůli nezachycenému atributu v datové sadě nebo šumu z pozadí.

Druhá datová sada je založena na změně pohlaví. To znamená, že mužskému obličeji jsou přiděleny ženské rysy a ženskému obličeji jsou přiděleny mužské rysy. Tyto akce už zasahují do identifikace a je očekáváno, že se úspěšnost sníží, protože se může poškodit i realističnost obličeje.

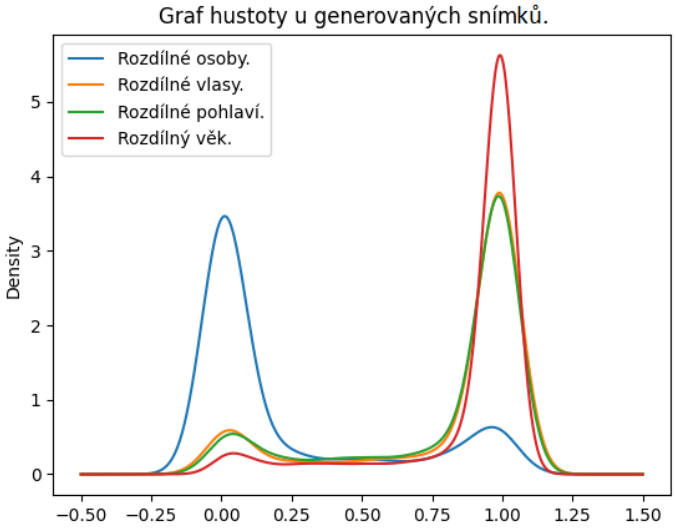
Poslední datová sada je vytvořena simulací stárnutí obličeje. Její výsledek je, že obličej zestárne o několik desetiletí, ale není zaručena podobnost. U této datové sady se očekává ještě nižší úspěšnost, protože většina snímků nebyla vytvořena velmi kvalitně a nemusí přesně odpovídat reálnému stavu.

Vyhodnocení věrohodnosti vzorků

Vyhodnocení věrohodnosti datových sad z generátorů syntetických snímků probíhá ve dvou fázích. První fáze zahrnuje zjištění samotné podobnosti na datech získaných prostřednictvím modelu při porovnávání snímků, které jsou vloženy do grafu pro podrobnější analýzu.

Ve druhé fázi se spočítá výsledek v procentech na základě prahu uvedeného v minulé sekci Porovnání modelů 5.1. Proces probíhá celkem třikrát: poprvé se vyhodnocuje podobnost originálního a vygenerovaného snímku, podruhé se porovnávají vygenerované snímky se stejnou změnou, a naposled se porovnávají vygenerované snímky s rozdílnou změnou od společného originálního snímku.

Originální a vygenerovaný snímkem



Obrázek 5.9: Graf hustoty u generovaných snímků.

Prvním krokem je porovnání originálního a vygenerovaného snímku. Pro tento účel byl vytvořen graf, který je zobrazen na obrázku 5.9, kde je vidět, jak jsou podobné různé datové sady. Tento graf je vytvořen odhadem hustoty pomocí Gaussových jader, takže zobrazuje množství vzorků pod křivkou v jeho obsahu. Jelikož je založený na Gaussových křivkách, tak nejvyšší lokální bod zobrazuje střední hodnotu, která značí průměr hodnot v okolí, a šířka křivky poskytuje přibližný rozptyl. Podle tohoto grafu je vidět také, že hodnoty jsou pod nulou a nad jedničkou, ale to je jen iluze odhadování Gaussových křivek, protože křivka je osově souměrná přes osu y, kde se promítá rozptyl do obou směrů, a tudíž i pod nulu a nad jedničku.

První vyznačená křivka slouží jako reference a zobrazuje vzorky s různými osobami na originálních snímcích.

Druhá křivka vyznačuje podobnost vlasů, kde je možné si všimnout, že mají vysokou podobnost a málo výsledků na opačném konci spektra. To znamená, že generátory se držely původního snímku a zaměnily pouze vlasy.

Třetí křivka představuje pohlaví, kde je vidět, že má vysoké hodnoty. Také ukazuje, že generátory neměnily rozměry a znaky obličejů, ale měnily pouze jejich vzhled. Podle mého názoru to není tak převratné, protože každé pohlaví má určité podobnosti v obličeji. Pokud se znaky více podobají originálu než skupině, do které mají patřit, výsledky mohou vypadat podivně. Toto je pouze připomínka, že toto vyhodnocení nemůže mít pevně dané hranice, protože není možné je přesně určit.

Poslední křivka se týká věku, který vykazuje velmi vysokou shodu v porovnání s jinými výsledky. To může být způsobeno tím, že starší obličeje, na kterých model trénoval, měly

Obrázek 5.10: Originální a vygenerované snímky ohodnocené jako shodné.

Obrázek 5.11: Originální a vygenerované snímky ohodnocené jako rozdílné.

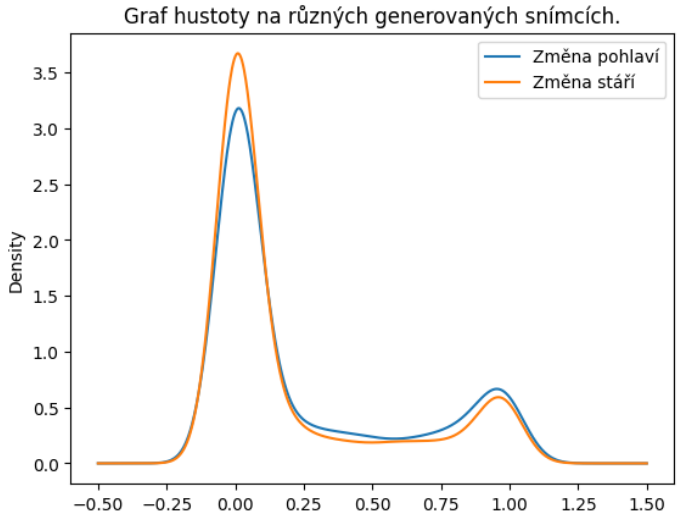
odhadem maximálně 60 let, a do této doby se obličej tolik nezmění. Přibude více vrásek a tuková vrstva pod kůží se může zmenšit, což nezmění výrazně obličejové rysy, ale může to ztížit rozpoznání. Na druhou stranu, primární datová sada pro učení neobsahuje tolik vzorků s různým věkem, což může do určité míry ovlivnit výsledky.

Po získání výsledků vyhodnocení bylo zjištěno, že u 23 vzorků nebylo možné detekovat obličej na snímku, a tyto vzorky byly automaticky vyřazeny z datové sady. U zbylých vzorků dopadlo vyhodnocení tak, že model `Gen_Verifier` přiřadil vzorkům s rozdílnými vlasy přesnost 86,83 %, vzorkům s rozdílným pohlavím přesnost 84,17 % a vzorkům s rozdílným věkem 91,6 %. Průměrné hodnocení modelu *StarGAN* je 87,53 %. Přehled vzorků ohodnocených jako stejné a různé je na obrázcích 5.10 a 5.11

Z toho lze usoudit, že model *StarGAN* má poměrně vysokou schopnost zachovat podobnost obličejových rysů. Pokud vezmeme v úvahu, že model `Gen_Verifier` má přenosnost 91 %, reálná hodnota může být až o 9 % vyšší, což by celkově průměrnou hodnotu zvýšilo na přibližně 96 %. To by znamenalo, že model může jistě měnit obličejové rysy u 4 % obličejů, a u 22 % vzorků je možnost, že se snímek nepodobá originálu. Tento výsledek zahrnuje také chyby z předchozího testování, kde 9 % může být falešně pozitivních nebo negativních.

Dva vygenerované snímky se stejnou změnou

Druhý test je založen na porovnávání dvou vygenerovaných snímků, které mají stejnou změnu, ale zachycují různé obličeje. Tento test se zaměřuje na porovnání podobnosti změny provedené generujícím modelem a jeho schopnosti obecně upravovat obličej, aniž by všechny obličeje byly podobné jednomu nebo skupině podobných obličejů, čímž by se ztrácely charakteristiky původní předlohy. Pokud je podobnost minimální, vygenerované obličeje se nepodobají žádnému skutečnému obličeji. Pokud je vysoká, existuje vyšší pravděpodobnost, že se podobají nějakému skutečnému obličeji. Toto testování se zaměřuje na změny ve stáří a pohlaví, které jsou znázorněny na grafu 5.12.



Obrázek 5.12: Graf hustoty na různých generovaných snímcích.

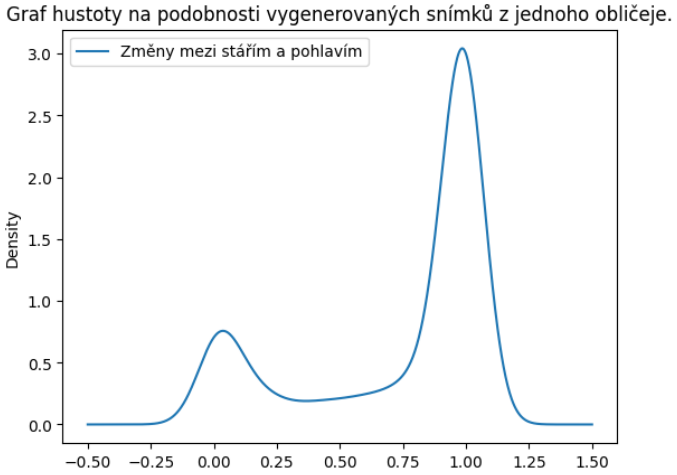
Obrázek 5.13: Vygenerované snímky s rozdílnými obličeji ohodnocené jako shodné.

Na tomto grafu je vidět, že hodnoty se primárně pohybují kolem nuly, což znamená, že snímky jsou poměrně rozdílné, i když se objevují i hodnoty blízko jedničky. Při přepočítání vyšlo, že shoda mezi snímky je 19,6 % u změněného pohlaví a 15,2 % u změněného stáří. Vzorky a jejich příklady jsou znázorněny na obrázcích 5.13 a 5.14. Z toho lze usoudit, že vygenerované snímky se přímo nepodobají žádnému jinému obličeji a že původní předloha má mnohem větší váhu.

Obrázek 5.14: Vygenerované snímky s rozdílnými obličeji ohodnocené jako rozdílné.

Dva vygenerované snímky s rozdílnou změnou

Poslední test je na podobném principu jako první, ale porovnávají se zde dva vygenerované snímky, které byly vytvořeny z jednoho obličeje, ale s různými změnami. Cílem tohoto testu je analyzovat, jak výrazně se mohou lišit vygenerované snímky, i když pocházejí z téže předlohy, a zjistit, zda se zvýrazní odlišnosti. Tento test probíhá na změnách v stáří a pohlaví.



Obrázek 5.15: Graf hustoty na podobnosti vygenerovaných snímků z jednoho obličeje.

Test je zobrazen na grafu 5.15, kde je vidět, že podobnost mezi snímky je většinou stále vysoká, ale dochází k vyšší rozdílnosti než v předchozích testech. Při převedení na procenta jsou snímky ohodnoceny na 81,8 %. Tento výsledek není příliš odlišný od prvního testu generovaných snímků, což naznačuje, že model se výrazně nezhoršil, ale podle vzorků na obrázku 5.16 a 5.17 se také nezlepšil. To může být způsobeno tím, že velké množství snímků s pochybnou realističností bylo zařazeno spíše do kategorie stejných snímků, což mohlo statistiku zkreslit.

Celkové výsledky generovaných snímků

Z těchto výsledků lze odvodit dvě sady vlastností. První se týkají modelu *Gen_Verifier* a druhé modelu *StarGAN*. Pokud začneme od prvního, model *Gen_Verifier* je značně ovlivněn reálností snímků, protože byl trénován na reálných snímcích, a tak se jeho rozpoznávání

Obrázek 5.16: Vygenerované snímky se stejnými originálními obličeji ohodnocené jako shodné.

Obrázek 5.17: Vygenerované snímky se stejnými originálními obličeji ohodnocené jako rozdílné.

zhoršuje při práci s nereálnými snímky. Tuto vlastnost lze částečně vyřešit přidáním kvalitních generovaných snímků do trénovací sady, ale v případě stávajícího modelu se primárně tyto snímky přidělí jako rozdílné. To je patrné z prvního testu na generovaných snímcích, kde se obličej příliš nezměnil při změně vlasů a pohlaví. U změny stáří se však vyskytla anomálie ze dvou důvodů. První se týká modelu *Gen_Verifier* a jeho nedostatečné trénovací sady *LFW*. Tato datová sada obsahuje různé obličeje, ale jen malé množství osob s rozdílným věkem, a tudíž není dobře připravena na porovnávání takových obličejů. Druhý důvod je spojen s modelem *StarGAN*, který byl trénován na sadě *CelebA*. Tato datová sada obsahuje velké množství obličejů, ale většina z nich je maximálně do 60 let, což znamená, že model se nenaučil generovat opravdu starší obličeje, a omezenost sady *LFW* se může projevit přeměnou na nižší věk. Druhé a třetí testování proběhlo podle očekávání, ale z pohledu na model *Gen_Verifier* se nepřinesly nové poznatky.

Z těchto poznatků lze vyvodit, že pro zlepšení modelu v určité míře je možné použít více modifikovaných sad pro trénování, čímž se model přesněji naučí, co hledat. Pokud je však potřeba zvýšit jeho celkovou přesnost, je nutná modifikace samotného modelu.

Základem těchto testů bylo ohodnocení modelu *StarGAN*, který se choval poměrně dobře až na nějaké chyby. Z prvního testování bylo možné zjistit, že poměrně zachovává původní rysy obličeje, ale některé jeho přeměny nebyly dokonalé jako ty, které jsou zobrazené v příkladech vzorků. Obecně jeho hodnocení bylo přes 84 %, což není úplně nejhorší, ale se započítanou chybovostí to je chvalitebné. Jediný problém byl u stáří, kde nastaly neočekávané hodnoty, které jsou popsané v předchozím odstavci. Dále byly testy zaměřující se na podobnost dvou vygenerovaných snímků, které byli vytvořeny stejnou změnu vlastností. U této změny vyšlo velice nízké číslo, čímž je možné říct, že model nepřeměňuje snímky k nějaké naučené předloze. Jako poslední bylo testování dvou vygenerovaných snímků ze stejného originálu. Tento test je na podobné bázi jako první test s generovanými snímky, ale tento je na podobnost výsledků pro další možnost zpracování například pro jiný model. Tento test vyšel poměrně dobře, ale nikoli tak dobře jako první test. Toto vše je možné shrnout, že *StarGAN* je poměrně dobrý model v zachování obličejů, ale má své nedostatky.

Na základě těchto testů bylo možné zhodnotit výkon modelu *StarGAN*, který se obecně choval poměrně dobře, i když vykazoval určité chyby. Z prvního testování vyplynulo, že model zachovává původní rysy obličeje, avšak některé jeho transformace nebyly dokonalé, což je patrné z příkladů vzorků. Celkové hodnocení modelu dosahovalo přes 84 %, což není špatné, ale se započítáním chybovosti jde o výborný výsledek. Jediný větší problém nastal u změny věku, kde se objevily neočekávané hodnoty, jak bylo popsáno v předchozím odstavci.

Další testy se zaměřily na podobnost dvou vygenerovaných snímků, které byly vytvořeny se stejnou změnou atributů. U této změny vyšlo velmi nízké číslo, což naznačuje, že model

nepřeměňuje snímky k nějaké naučené předloze, ale že upravuje originální snímky. Poslední test byl zaměřen na porovnání dvou vygenerovaných snímků ze stejného originálu. Tento test je podobný prvnímu, ale hodnotí podobnost výsledků pro další možnosti zpracování, například pro jiný model. Tento test dopadl poměrně dobře, ale ne tak dobře jako první test.

U testů byly také zahrnuty příklady porovnávaných snímků. U některých z nich bylo vidět, že model *StarGAN* má problémy s atributy nebo prostředím. U jednoho vzorku při změně barvy vlasů se například změnila barva kšiltovky nebo se změnilo pozadí na stejnou barvu, jakou měla původní barva vlasů.

Celkově lze shrnout, že *StarGAN* je poměrně dobrý model pro zachování obličejových rysů, avšak má své nedostatky.

5.3 Ladění modelu a experimenty

Než byl vytvořen model `Gen_Verifier`, bylo provedeno velké množství prototypů, které v kontrolovaných podmínkách fungovaly poměrně dobře. Nicméně, když byly tyto prototypy testovány na snímcích z reálného světa, jejich přesnost se výrazně zhoršila. Toto vedlo k provedení úprav a experimentů, které byly integrovány do finálního modelu prezentovaného v práci.

První model

První model, který byl sestrojen pro rozpoznávání, byla siamská neuronová síť, která byla použita na černobílé datové sadě obsahující 400 snímků. Tyto snímky byly vytvořeny od čtyřiceti různých osob, přičemž každá osoba měla deset snímků. Struktura snímků byla velice jednoduchá. Obličej byl vždy uprostřed snímku, tělo nebylo moc vidět a pozadí bylo bílé. Další výhodou bylo, že obličeje nebyly nijak otočené a všechny osoby byly jednoho typu. Tím je myšleno, že šlo o dospělé muže bez vousů, krátce ostříhané, bez klobouků nebo brýlí, a všichni měli stejnou barvu pleti. Jako ulehčení měly všechny snímky stejnou velikost. Tyto ideální snímky byly použity pro natrénování a analýzu.

Model vypadal velice jednoduše. Předzpracování bylo založeno na principu normalizace, kdy se snímky pouze vydělily číslem 255.

Jako podsíť model používal konvoluční neuronovou síť, která obsahovala sekvence 3×3 konvolučních vrstev s aktivační funkcí *ReLU*. Následovala 2×2 *max pooling* vrstva a poté *dropout* vrstva s parametrem 0,1. Tato sekvence se opakovala třikrát. Poté přišlo vyhlazení snímků do jedné řady pomocí vrstvy *flatten*. Poslední tři vrstvy byly plně propojené, kde první z nich obsahovala 128 neuronů, následovala *dropout* vrstva s parametrem 0,1 a poslední byla opět plně propojená s 50 neurony. Obě plně propojené vrstvy měly aktivační funkci *ReLU*.

Tyto sítě se spojily a směřovaly do vrstvy *lambda* s výpočtem Euklidovské vzdálenosti. Jako ztrátovou funkci jsem použil *Contrastive loss* s optimalizátorem *RMSprop*.

Tento model dosahoval 94,% úspěšnosti, ale jeho výstup byl stále poměrně rozptýlený. Proto byla do modelu přidána ještě jedna plně propojená vrstva s jedním neuronem a aktivační funkcí *Sigmoid*. Poté se přesnost zvýšila na přibližně 97 %, ale model měl jednu nepříjemnou vlastnost jako jeho předchůdce, a to, že se po učení stal invertovaný.

Model na barevné obrázky

Původní návrh modelu pro barevné obrázky zahrnoval nalezení obličejů na snímcích, jejich převod do černobílé podoby, transformaci na potřebnou velikost a následné vložení do stávajícího modelu. Tento experiment však skončil neúspěchem, protože model dosáhl maximálně 62 % úspěšnosti na omezené datové sadě *LFW* obsahující 400 snímků.

V následujících modelech byl zvýšen počet konvolučních vrstev s různými typy a rozšířeno množství perceptronů v plně propojených vrstvách. Dále bylo zvětšeno rozlišení přijímaných snímků na 50×50. Tyto kroky vedly ke zvýšení účinnosti modelu na hodnoty okolo 75 %, ale s každou přidanou vrstvou se míra učení zpomalovala a stávala se méně účinnou.

Proto byly do modelu přidány reziduální bloky s konvolucí místo tradičních konvolučních vrstev. Reziduální bloky byly implementovány v rámci modelu *ResNet50*, přičemž byly odstraněny všechny plně propojené vrstvy a většina nastavení byla přepsána. Dalším vylepšením byla příprava dat, která zahrnovala normalizaci a změnu kanálů. Tímto krokem se podařilo zvýšit počet konvolučních vrstev a zlepšit kvalitu modelu, přičemž přesnost dosáhla hodnoty kolem 86 %.

Jelikož z reziduálních bloků při vycházeli občas vysoká čísla, tak byli přidané normalizace po reziduálních blocích a po výpočtu Euklidovské vzdálenosti a společně s navýšením perceptronů u plně propojených vrstev.

Jelikož z reziduálních bloků občas vycházela vysoká čísla, byly přidány normalizační vrstvy za reziduálními bloky a po výpočtu Euklidovské vzdálenosti. Současně bylo zvýšeno množství perceptronů na 1024 a na 128 v plně propojených vrstvách. Tyto úpravy vedly ke zlepšení modelu, který dosáhl přesnosti 89 %.

Po dalších experimentech s počtem perceptronů v plně propojených vrstvách se počty zvýšily až na 2048 a 512. Přesnost modelu se začala ustalovat kolem 91 % a přidání dalšího počtu perceptronů již nemělo výrazný vliv na zlepšení výsledků.

Kapitola 6

Závěr

Úkolem této práce byla tvorba aplikace pro vyhodnocení věrohodnosti synteticky generovaných snímků obličeje. Toto vyhodnocení se nezaměřuje na realističnost nebo vzhled snímků. Ale její cíl je porovnávání obličejů. Obličeje jsou extrahovány z originálních a vygenerovaných snímků a následně porovnávány na základě podobnosti obličejových rysů.

Pro uskutečnění aplikace jsou v práci představeny témata, která se zabývají touto problematikou. První téma, které je představeno, je generování syntetických snímků obličeje. Zde jsou popsány neuronové sítě, základní bloky pro generování snímků a generativní soupeřící sítě, které se starají o samotné generování syntetických snímků.

Druhé téma se zabývá rozpoznáním osob podle obličeje. Zde je uvedeno, jak tento proces funguje a jaké jsou vytvořené modely pro jeho řešení, které jsou později používány pro referenci kvality na reálných snímcích. K tomuto tématu jsou také připojeny dvě datové sady, které jsou použity při učení modelů a následném generování snímků.

Po znalostech získaných z předchozích témat je navrhnut model *Gen_Verifier*, který je jádrem aplikace pro porovnávaní. Model je tvořen siamskou neuronovou sítí, která je poté implementována a trénována s použitím několika tříd pro snadnější ovládání a správnou funkčnost aplikace. Model *Gen_Verifier* je navržen tak, aby vyhodnocoval podobnost mezi obličeji v rámci ohraničeného intervalu, což zajišťuje snadnější interpretaci výsledků.

Testování proběhlo na reálných a generovaných datových sadách. Výsledky po trénování modelu na reálných snímcích a následném testování ukázaly, že model dosahuje 91 % přesnosti na datové sadě *LFW*, kde se většina výstupních hodnot se pohybuje blízko 0 a 1, což usnadňuje rozhodování mezi rozdílnými a stejnými obličeji. U testování na generovaných snímcích z datové sady *CelebA* model dosáhl průměrné přesnosti 87,53 %. To naznačuje, že přibližně 4 % vygenerovaných snímků od modelu *StarGAN* se neshodují s originálním snímkem, přičemž je zohledněna chybovost z prvního testování.

Z toho lze usoudit dva hlavní závěry. Prvním je hodnocení modelu *Gen_Verifier*, který měl chybovost 9 %, kde nejvíce měl problémy s rozpoznáváním osob s různým věkem. Tato chybovost není příliš vysoká, což naznačuje, že model je relativně spolehlivý pro vyhodnocování věrohodnosti. Nicméně, pro budoucí použití bude potřeba zvýšit přesnost, což lze dosáhnout buď zahrnutím více generovaných snímků do trénovací sady pro zlepšení rozpoznávaní méně reálných snímků, nebo úpravou struktury modelu.

Druhý závěr se týká testovaného generátoru *StarGAN*, u kterého bylo zjištěno, že přibližně 5 % vygenerovaných obličejů se neshoduje s originálem a některé snímky nebylo možné ani porovnat. Model je však zaměřen pouze na generování co nejpodobnějších snímků, a tím jsou jeho výsledky velmi dobré. Pokud by však obsahoval specializovanou jednotku pro ověřování věrohodnosti, aby jeho přesnost mohla být ještě lepší.

Literatura

[1] Algoscale. *How To Use GAN To Generate Images* [online]. 2022 [cit. 2023-12-12]. Dostupné z: https://algoscale.com/blog/how-to-use-gan-to-generate-images/.

[2] Amato, G., Falchi, F., Gennaro, C. a Vairo, C. *A Comparison of Face Verification with Facial Landmarks and Deep Features*. Duben 2018.

[3] Benhur, S. *A friendly introduction to Siamese Networks* [online]. Towards Data Science, 2020 [cit. 2024-03-20]. Dostupné z: https://towardsdatascience.com/a-friendly-introduction-to-siamese-networks-85ab17522942.

[4] Brownlee, J. *A Gentle Introduction to Deep Learning for Face Recognition* [online]. Deep Learning for Computer Vision, 2019 [cit. 2023-12-27]. Dostupné z: https://machinelearningmastery.com/introduction-to-deep-learning-for-face-recognition/.

[5] Brownlee, J. *A Gentle Introduction to Generative Adversarial Networks (GANs)* [online]. 2019 [cit. 2023-12-30]. Dostupné z: https://machinelearningmastery.com/what-are-generative-adversarial-networks-gans/.

[6] Brownlee, J. *A Gentle Introduction to the Rectified Linear Unit (ReLU)* [online]. 2020 [cit. 2024-03-30]. Dostupné z: https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/.

[7] Brownlee, J. *How Do Convolutional Layers Work in Deep Learning Neural Networks?* [online]. Deep Learning for Computer Vision, 2020 [cit. 2023-12-27]. Dostupné z: https://machinelearningmastery.com/convolutional-layers-for-deep-learning-neural-networks/.

[8] Brownlee, J. *Softmax Activation Function with Python* [online]. 2020 [cit. 2024-03-30]. Dostupné z: https://machinelearningmastery.com/softmax-activation-function-with-python/.

[9] Brownlee, J. *A Gentle Introduction To Sigmoid function* [online]. 2021 [cit. 2024-03-30]. Dostupné z: https://machinelearningmastery.com/a-gentle-introduction-to-sigmoid-function/.

[10] Chauhan, N. S. *Generate Realistic Human Face using GAN* [online]. KDnuggests, 10. března 2020 [cit. 2023-12-12]. Dostupné z: https://www.kdnuggets.com/2020/03/generate-realistic-human-face-using-gan.html?__cf_chl_tk=DkjPXVyYj1604EMWgzLxoLGq7ND_Hw1VGJXtNj.KVDU-1703156934-0-gaNycGzND9A.

[11] Choi, Y. *StarGAN - Official PyTorch Implementation*. 2020. Dostupné z: https://github.com/yunjey/stargan.

[12] Choi, Y., Choi, M., Kim, M., Ha, J.-W., Kim, S. et al. *StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation*. 2018.

[13] Deng, J., Guo, J., Yang, J., Xue, N., Kotsia, I. et al. ArcFace: Additive Angular Margin Loss for Deep Face Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 1. vyd. Institute of Electrical and Electronics Engineers (IEEE). říjen 2022, sv. 44, č. 10, s. 5962–5979. DOI: 10.1109/tpami.2021.3087709. ISSN 1939-3539. Dostupné z: http://dx.doi.org/10.1109/TPAMI.2021.3087709.

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

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

[16] Guiné, R. *The Use of Artificial Neural Networks (ANN) in Food Process Engineering*. Březen 2019. DOI: 10.18178/ijfe.5.1.15-21.

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

[18] Huang, G. B., Ramesh, M., Berg, T. a Learned Miller, E. *Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments*. 07-49. University of Massachusetts, Amherst, October 2007.

[19] IBM. *The Neural Networks Model* [online]. IBM Corporation, 2021 [cit. 2023-12-12]. Dostupné z: https://www.ibm.com/docs/en/spss-modeler/18.0.0?topic=networks-neural-model.

[20] IBM. *Convolutional neural networks* [online]. IBM Corporation, 2022 [cit. 2023-12-12]. Dostupné z: https://www.ibm.com/topics/convolutional-neural-networks.

[21] Keen, M. *What are GANs (Generative Adversarial Networks)?* [online]. IBM Technologyt, 2021. Dostupné z: http://neuralnetworksanddeeplearning.com/.

[22] Liu, Z., Luo, P., Wang, X. a Tang, X. *Deep Learning Face Attributes in the Wild*. December 2015.

[23] *How CNNs Work* [online]. Math works, 2021 [cit. 2023-12-12]. Dostupné z: https://www.mathworks.com/discovery/convolutional-neural-network-matlab.html.

[24] Meng, Q., Zhao, S., Huang, Z. a Zhou, F. *MagFace: A Universal Representation for Face Recognition and Quality Assessment*. 2021.

[25] Nguyen, T.-P., Lathuilière, S. a Ricci, E. *Multi-Domain Image-to-Image Translation with Adaptive Inference Graph*. 2021.

[26] Nielsen, M. A. *Neural Networks and Deep Learning* [online]. Determination Press, 2018. Dostupné z: http://neuralnetworksanddeeplearning.com/.

[27] Pythonu, V. *What is Python? Executive Summary* [online]. 2020 [cit. 2024-04-15]. Dostupné z: https://www.python.org/doc/essays/blurb/.

[28] Rosebrock, A. *Convolutional Neural Networks (CNNs) and Layer Types* [online]. PyTmageSearch, květen 2021 [cit. 2024-05-17]. Dostupné z: https://pyimagesearch.com/2021/05/14/convolutional-neural-networks-cnns-and-layer-types/.

[29] Saha, S. *A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way* [online]. Towards Data Science, 2018 [cit. 2024-04-15]. Dostupné z: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

[30] Sharma, S. *Activation Functions in Neural Networks* [online]. Towards Data Science, 2017 [cit. 2023-12-15]. Dostupné z: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6.

[31] Shinde, S., Pradhan, T., Ghorpade, A. a Tale, M. *Face Generation from Textual Features using Conditionally Trained Inputs to Generative Adversarial Networks*. 2023.

[32] Taherkhani, F., Talreja, V., Dawson, J., Valenti, M. C. a Nasrabadi, N. M. *Profile to Frontal Face Recognition in the Wild Using Coupled Conditional GAN*. 2021.

[33] Taskiran, M., Kahraman, N. a Erdem, C. Face recognition: Past, present and future (a review). *Digital Signal Processing*. 1. vyd. Červenec 2020, sv. 106, č. 1, s. 102809. DOI: 10.1016/j.dsp.2020.102809.

[34] Taskiran, M., Kahraman, N. a Erdem, C. E. Face recognition: Past, present and future (a review). *Digital Signal Processing*. 1. vyd. 2020, sv. 106, č. 1, s. 102809. DOI: https://doi.org/10.1016/j.dsp.2020.102809. ISSN 1051-2004. Dostupné z: https://www.sciencedirect.com/science/article/pii/S1051200420301548.

[35] Terhörst, P., Ihlefeld, M., Huber, M., Damer, N., Kirchbuchner, F. et al. *QMagFace: Simple and Accurate Quality-Aware Face Recognition*. 2022.

[36] Tran, L., Yin, X. a Liu, X. Representation Learning by Rotating Your Faces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 1. vyd. Institute of Electrical and Electronics Engineers (IEEE). prosinec 2019, sv. 41, č. 12, s. 3007–3021. ISSN 1939-3539.

[37] Wang, S.-C. *Artificial Neural Network*. 1. vyd. Boston, MA: Springer US, 2003. 81–100 s. ISBN 978-1-4613-5046-0, 978-1-4615-0377-4.

[38] Wang, Z. *Contrasting contrastive loss functions* [online]. Towards Data Science, 2020 [cit. 2024-03-20]. Dostupné z: https://towardsdatascience.com/contrasting-contrastive-loss-functions-3c13ca5f055e.

[39] Wu, P.-W., Lin, Y.-J., Chang, C.-H., Chang, E. Y. a Liao, S.-W. *RelGAN: Multi-Domain Image-to-Image Translation via Relative Attributes*. 2019.

[40] Yang, S., Luo, P., Loy, C. C. a Tang, X. *WIDER FACE: A Face Detection Benchmark*. 2015.

[41] Yathish, V. *Loss Functions and Their Use In Neural Networks* [online]. Towards Data Science, 2020 [cit. 2023-12-15]. Dostupné z: https://towardsdatascience.com/loss-functions-and-their-use-in-neural-networks-a470e703f1e9.

[42] Zhang, K., Zhang, Z., Li, Z. a Qiao, Y. Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks. *IEEE Signal Processing Letters*. 1. vyd. Institute of Electrical and Electronics Engineers (IEEE). říjen 2016, sv. 23, č. 10, s. 1499–1503. DOI: 10.1109/lsp.2016.2603342. ISSN 1558-2361. Dostupné z: http://dx.doi.org/10.1109/LSP.2016.2603342.