VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

ROZPOZNÁNÍ OSOB S ČÁSTEČNĚ ZAHALENOU TVÁŘÍ
MASKED FACE RECOGNITION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR
Jan Kašpar

VEDOUCÍ PRÁCE
SUPERVISOR
Ing. Jiří Přinosil, Ph.D.

BRNO 2021

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Informační bezpečnost**

Ústav telekomunikací

Student: Jan Kašpar ***ID:*** 211796
Ročník: 3 ***Akademický rok:*** 2020/21

NÁZEV TÉMATU:

Rozpoznání osob s částečně zahalenou tváří

POKYNY PRO VYPRACOVÁNÍ:

V rámci práce se seznamte s principem identifikace osob v obraze na základě rozpoznání tváří. Na základě získaných znalostí vyberte vhodný stávající algoritmus, který upravte pro možnost rozpoznávání i částečně zahalených tváří (např. rouška či šátek). Ověřte realizovaný algoritmus na vhodné databázi obrazových nahrávek z reálného prostředí a stanovte podmínky použití.

DOPORUČENÁ LITERATURA:

[1] DENG, Jiankang, et al. Arcface: Additive angular margin loss for deep face recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2019. p. 4690-4699.

[2] WU, Cho Ying; DING, Jian Jiun. Occluded face recognition using low-rank regression with generalized gradient direction. Pattern Recognition, 2018, 80: 256-268.

Termín zadání: 1.2.2021 ***Termín odevzdání:*** 31.5.2021

Vedoucí práce: Ing. Jiří Přinosil, Ph.D.

doc. Ing. Jan Hajný, Ph.D.
předseda rady studijního programu

Abstrakt

Tato bakalářská práce se zaměřuje na rozpoznání osoby se zahalenou tváří. V rámci práce jsou popsány čtyři algoritmy InsightFace, FaceNet, OpenFace a FaceRecognition. Z těchto algoritmů je vybrán nejvhodnější pro rozpoznávání osob se zahalenou tváří. Jsou vytvořeny dvě databáze osob a natrénovány dva modely pro rozpoznávání osob, přičemž jeden je pro rozpoznávání osob se zahalenou tváří a druhý pro nezahalené osoby. Pro tyto modely jsou vytvořeny ROC křivky k zjištění přesnosti modelů. Vytvořený program slouží k rozpoznání osoby nebo jejího zařazení do databáze vytvořené pomocí SQLite.

Klíčová slova

Detekce tváře, rozpoznání tváří, Python, SQLite, ROC křivky

Abstract

This bachelor thesis focuses on the recognition of people with covered faces. The work describes four algorithms InsightFace, FaceNet, OpenFace, and face recognition. From these algorithms, the most suitable for recognizing people with a covered face is selected. Two databases of persons are created, and two models for the recognition of persons are trained; one is for the recognition of persons with a covered face and the other for non-covered persons. ROC curves are created for these models to determine the accuracy of the models. The created program is used to recognize people or to add them to the database created by SQLite.

Keywords

Face detection, face recognition, Python, SQLite, ROC curves

KAŠPAR, Jan. *Rozpoznání osob s částečně zahalenou tváří.* Brno, 2021. Dostupné také z: https://www.vutbr.cz/studenti/zav-prace/detail/133503. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce Jiří Přinosil.

Prohlášení autora o původnosti díla

Jméno a příjmení studenta:	Jan Kašpar
VUT ID studenta:	211796
Typ práce:	Bakalářská práce
Akademický rok:	2020/21
Téma závěrečné práce:	Rozpoznání osob s částečně zahalenou tváří

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

.

V Brně dne: …… ----------

podpis autora

Poděkování

Děkuji vedoucímu bakalářské práce Ing. Jiřímu Novotnému, Ph.D. za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé bakalářské práce.

V Brně dne: .....

---

podpis autor

Obsah

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

Úvod

V této době je rozvoj technologií na velmi vysoké úrovni. Tyto technologie se dají použít k mnoha dobrým, ale i špatným účelům, proto je důležité chránit bezpečí státu a jeho občanů. Pro zajištění bezpečnosti je možné použít technologie určené k identifikaci osob.

K identifikaci osob se používá rozpoznávání osob zachycených na snímku. K tomu se nejčastěji využívá umělá inteligence, jinak řečeno algoritmy strojového učení. Počítač je možné naučit, aby určil, zda se na snímku nachází obličej a poté ho i rozpoznat.

Proti identifikaci je možné se bránit, např. použitím šátku, roušky nebo jiného prostředku, kterým dojde k zakrytí tváře. Tím dojde ke zhoršení nebo znemožnění identifikace osoby. Další faktory, které ztěžují identifikaci, jsou: kvalita snímku a úhel natočení obličeje. Dnešní počítače si s těmito problémy identifikace neumí úplně poradit. Rozpoznávání obličejů může být použito všude, kde je potřeba identifikovat osobu na základě jejího obličeje, například pro autentizaci pro přístup na zabezpečená místa.

V této době se lze setkat s identifikací na základě obličeje pro odemknutí počítače nebo pro odemknutí mobilního telefonu. Nejvíce používané je však na letištích, bankách nebo vlakových nádražích pro zjištění a následnou eliminaci hrozby. Jednoduše řečeno, lze ho použít tam, kde se nachází ve stejnou dobu velký počet lidí a je nutné dbát na ochranu obyvatelstva nebo jiných cenných aktiv jako tajné informace apod.

Jako příklad lze uvést letiště, zde může identifikace osoby odhalit potenciální hrozbu, tím pomáhá pracovníkům ostrahy rychleji kontrolovat a odbavovat cestující. Kdo už alespoň jednou letěl letadlem, ví, že se na letištích nachází spousta kamer, které mohou na zachycených snímcích detekovat obličej a následně provést porovnání, zda se daná osoba nenachází na seznamu hledaných nebo potenciálně nebezpečných, kteří by mohli ohrozit bezpečnost. Nebo tuto identifikaci lze provést při osobní kontrole, kdy osoba předloží svůj průkaz totožnosti a systém po načtení fotografie z průkazu porovná s databází.

Jako další příklad lze uvést hokejová nebo fotbalová utkání, kde příliš vášniví fanoušci mohou způsobovat velký rozruch a tím vyvolat nepříjemnou potyčku mezi fanoušky soupeřova týmu, nebo přerušit zápas například hozením dýmovnice na hrací plochu. Tyto události jsou pro pořadatele zápasu nepříjemné a většinou končí peněžitým trestem. Pro předcházení těchto událostí je možné využít systém pro identifikaci obličeje. Jelikož jsou nejčastějšími výtržníky na těchto událostech právě skalní fanoušci týmu, tak je velmi pravděpodobné, že budou také vlastníky permanentky, což jim umožňuje volně chodit na zápasy. V případě nějaké nepříjemné situace se využijí záznamy z kamer na stadionu, aby bylo zjištěno, kdo situaci vyvolal a mohl být potrestán. Pomocí snímků z kamer dojde k rozpoznání obličeje, a porovnání s fotkami fanoušků, kteří vlastní permanentku. Po tomto zjištění může být pachatelovi zakázán vstup na stadion na několik zápasů, nebo i doživotně podle závažnosti situace, kterou vyvolal. Tento systém funguje

například v Americe, kde se takto omezení fanoušci musí v době, kdy jejich tým hraje, hlásit na policejní stanici.

Jako další příklad lze uvést firmy, které mají stovky nebo tisíce zaměstnanců pohybujících se po objektu. V případě takového množství lidí je takřka nemožné pro ostrahu objektu si pamatovat všechny zaměstnance a hrozí, že se do oblasti, kam by neměl mít nikdo nepověřený přístup, dostane cizí osoba. Proti vzniku takové situace lze využít biometrické zabezpečovací systémy, jako autentizace pomocí otisku prstu, oční duhovky, nebo krevního řečiště dlaně, všechny tyto typy autentizace mají přísná pravidla a jdou těžko obejít, jejich nevýhodou je však jejich cena a čas, který je potřeba pro autentizaci osoby a její zdržení nutností provést autentizace. Pokud se ale po objektu pohybuje velké množství osob a není potřeba mít vysokou úroveň zabezpečení pomocí biometrického systému, lze využít rozpoznávání osob pomocí snímků obličeje. Po vstupu osoby do zabezpečené oblasti bude pomocí kamer pořízen snímek obličeje, který se porovná s fotografiemi zaměstnanců pracujících ve firmě a pokud nebude nalezena shoda, bude ostraha objektu upozorněna, že se v oblasti nachází osoba, co zde nemá co dělat.

1. TEORETICKÁ ČÁST

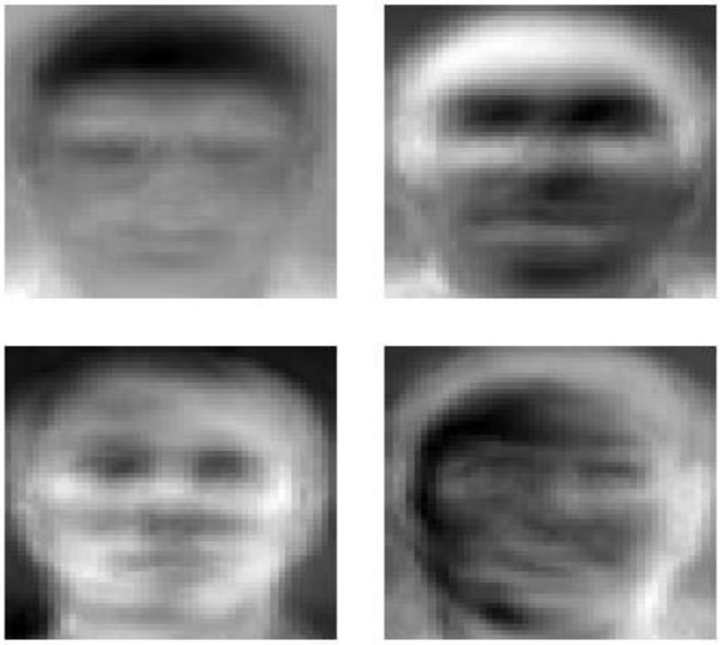
V této části je popsán historický vývoj algoritmů pro rozpoznávání obličeje, problematika související s rozpoznáváním obličejů, a popis několika dnes existujících algoritmů pro rozpoznávání obličeje.

1.1 Historický vývoj algoritmů pro rozpoznávání obličeje

První krok pro vývoj algoritmů pro rozpoznávání obličeje se uskutečnil v 60 letech minulého století, kdy byl vytvořen projekt „člověk-stroj“. Nevýhoda tohoto projektu byla, že specifické rysy obličeje musel určit člověk, až poté mohla být fotografie vložena do počítače a ten provedl výpočet vzdáleností pro každou fotografii a podle rozdílu vzdáleností rysů určit, zda se jedná o shodu.

V roce 1970 Takeo Kanade demonstroval systém shody tváře, který lokalizoval rysy obličeje bez zásahu člověka a vypočítal vzdálenost mezi rysy. Tento systém nemohl ale vždy určit rysy spolehlivě, ale díky tomu vzrostl zájem o toto téma.

V roce 1987 Sirovich a Kirby vyvinuli PCA (Principal Compotent Analysis). Nejdříve se vypočte průměrný obraz tváře, který je poté odečten od všech ostatních tváří, z těchto tváří se poté určí kovariační matice, nad kterou se provede převod na vlastní čísla a vektory, které reprezentují odlišnosti dané tváře od tváře průměrné. Celkový obraz tváře je kombinace světlých a tmavých oblastí [24].



Obr. 1.1 Tváře převedené metodou PCA [24]

V roce 1993 došlo k prvnímu plošnému využívání systému pro rozpoznávání obličeje v Západní Virginii a Novém Mexiku pro odhalování lidí, kteří měli více řidičských průkazů. Z důvodu, že ve Spojených státech byly běžně přijímány řidičské průkazy s fotografií jako doklad totožnosti.

Do roku 1997 metoda detekce obličeje od společnosti Malsburg překonala většinu ostatních systému pro detekci obličeje. Bochumův algoritmus byl komerčně prodáván na trhu do oblastí letišť a dalších rušných lokalit. Tento systém byl dostatečný k identifikaci člověka i z méně kvalitních fotografií, dokázal obejít překážky jako brýle, vousy a kníry. V roce 2001 došlo k možnosti detekce tváře v reálném čase na videozáznamu.

V současnosti jsou systémy pro rozpoznávání obličejů velice rozšířené a používané v bankách, luxusních hotelech a obchodech, policejních stanicích, celních úřadech, mezinárodních letištích, vladních zařízeních, vlakových nadražích, stadionech, a v mnoha dalších s cílem zvýšit bezpečnost a efektivitu možného vyšetřování.

Největší státy, které využívají systém pro rozpoznávání obličejů, jsou mimo ostatních: Velká Británie, Spojené státy, Čína, Latinská Amerika a Nizozemí.

V únoru roku 2020 po propuknutí koronaviru v Číně došlo k optimalizaci toho systému, pro detekci pomocí tělesné teploty a tím idenfikování lidí, kteří jsou nemocní, protože bylo nutné zlepšit přesnost systému pro identifikaci lidí, kteří mají zakryté dychací cesty.

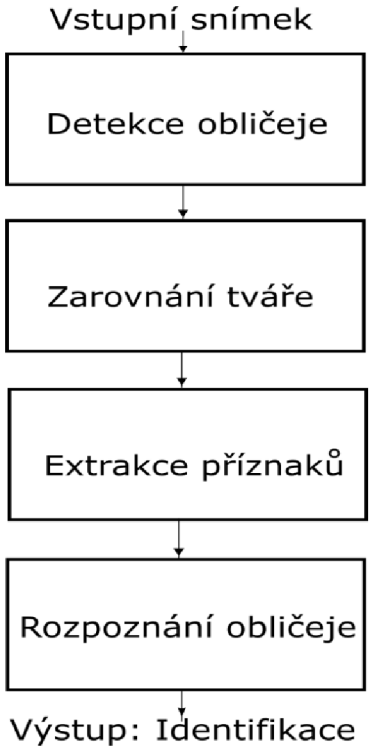
S používáním detekce obličejů, ale přichází i špatná stránka jako porušení ochrany osobních údajů. Aktivisté v oblasti ochrany soukromí jsou znepokojeni využíváním bezpečnostních technologií jako rozpoznávání tváře, neboť může sloužit nejen k identifikaci jednotlivce, ale také odhalení osobních údajů o jednotlivci, (např. s kým se stýká, příspěvky a profily na sociálních sítí, chování na internetu a cestovní vzorce).
Z toho důvodu několik států ve Spojených státech zakázalo používání technologie pro rozpoznávání tváře.

K zamezení rozpoznání tváře v roce 2013 japonští vědci vytvořili brýle “clony soukromí”, které vyzařují infračervené světlo a obličej je pod ním nerozpoznatelný pro systém rozpoznávání tváře [11].

1.2 Problematika související s rozpoznáváním obličejů

Pro lidi je vcelku jednoduché si zapamatovat, jak někdo vypadá a poté ho identifikovat. Aniž si to uvědomujeme, podvědomě si zapamatováváme různé rysy jako tvar hlavy, barvu očí, vzdálenost očí, nos, ústa atd. Podobně pracují i algoritmy pro rozpoznávání obličejů, ty se zaměřují na specifické rysy jako nos, ústa, čelist, oči, obočí. Nejčastěji se však používají oči, nos a ústa. Při rozpoznávání osob pomocí snímku obličeje je každé osobě přiřazen specifický identifikátor.

Proces pro rozpoznávání obličeje se dělí na 4 kroky:
Detekce obličeje – Nalezení tváře na obrázku a její významné rysy- nos, ústa a oči
Zarovnání tváře – Upravení tváře, tak aby odpovídala formátu databáze - natočení
Extrakce rysů – V závislosti na použitém algoritmu se vytvoří daný počet vektorů, které lze použít pro rozpoznávání.
Rozpoznání obličeje – Porovnávání s tváří nebo více známými tvářemi v databázi.



Obr. 1.2 Obecný systém rozpoznávání obličeje

1.3 Algoritmy pro rozpoznání obličeje

Na internetu lze nalézt mnoho volně šiřitelných algoritmů pro rozpoznávání obličeje (FaceNET, InsightFace, OpenFace, FaceRecognition), některé umožňují uživateli natrénovat vlastní model pro rozpoznávání obličeje, a to buď pomocí poskytnuté databáze, nebo databáze, kterou poskytne sám uživatel a převede ji do potřebného tvaru, který je nutný pro trénování, a jiné lze pouze využít bez možnosti natrénování vlastního modelu pro rozpoznávání obličeje.

1.3.1 FaceNET

Tento algoritmus byl vytvořen v roce 2015 trojicí Florian Schroff, Dmitry Kalenichenko a James Philbin ze společnosti Google.

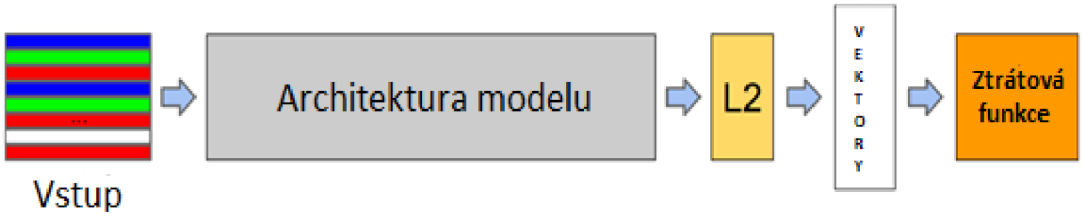
Tento algoritmus pracuje s konvulenční neuronovou sítí, která se přímo učí mapování z tváře na fotografii do kompaktního euklidovského prostoru, kde vzdálenosti přímo odpovídají míře podobnosti tváře. Jakmile je tento prostor vytvořen, tak detekce obličeje je jednoduše provedena pomocí standartních technik s pomocí takzvaných embeddings, jedná se o vektor funkcí. Tento systém využívá hlubokou neuronovou síť natrénovanou přímo k optimalizaci vektorů, tato síť pro každý obličej na fotografii vytvoří 128dimenzionální vektor. Pro tváře LFW (označené tváře ve volné přírodě), dosahuje systém přesnosti 99,63 %, na databázi od YouTube dosáhl přesnosti 95,15 % na síti, která byla trénovaná pomocí CPU s hodnotou učení 0,05 po dobu 1000 až 2000 hodin, kde po 500 hodinách tréninku došlo k výraznému zlepšení přesnosti sítě.

Síť byla natrénována na snímcích o velikosti 220 × 220 pixelů, ale je možné do natrénovaného modelu vložit i snímky o velikosti 120 × 120 pixelů nebo i 80 × 80 pixelů. Dle autorů by trénování sítě na snímcích s nižním rozlišení mohlo zlepšit výkon i pro snímky o ještě menší velikosti.

Triplet Loss - je funkce kde se vyberou tři obrázky anchor (referenční), Positive (obrázek, na kterém je stejná osoba jako na referenčím) a Negative (obrázek, na kterém je jiná osoba než na referenčním).

$$Ztrátová\ Funkce = \sum_{i=1}^{N}[||f_i^a - f_i^p||_2^2 - ||f_i^a - f_i^n||_2^2 + \alpha], \quad (1.1)$$

kde f_i^a je referenční obrázek, f_i^p obrázek, na kterém je stejná osoba jako na referenčním a f_i^n obrázek, na kterém je jiná osoba než na referenčním kde α je hodnota mezi pozitivními a negativními obrázky. Tato funkce je použita ke zvýšení rychlosti učení modelu.
Z dokumentu FaceNET, který je veřejně dostupný, vychází OpenFace a další různé implementace [2][14].



Obr. 1.3 Architektura modelu sítě [2]

1.3.2 InsightFace

Algoritmus ArcFace byl vytvořen v roce 2018 čtveřicí Jiankang Deng, Jia Guo, Niannan Xue a Stefanos Zafeiriou. První dva výše zmínění později vytvořili systém pod názvem insightFace. Jedná se o volně dostupný systém pro analýzu tváře, který je převážně založen na MXNetu. Síť pracuje s programovacím jazykem Python od verze 3.0 a výše.

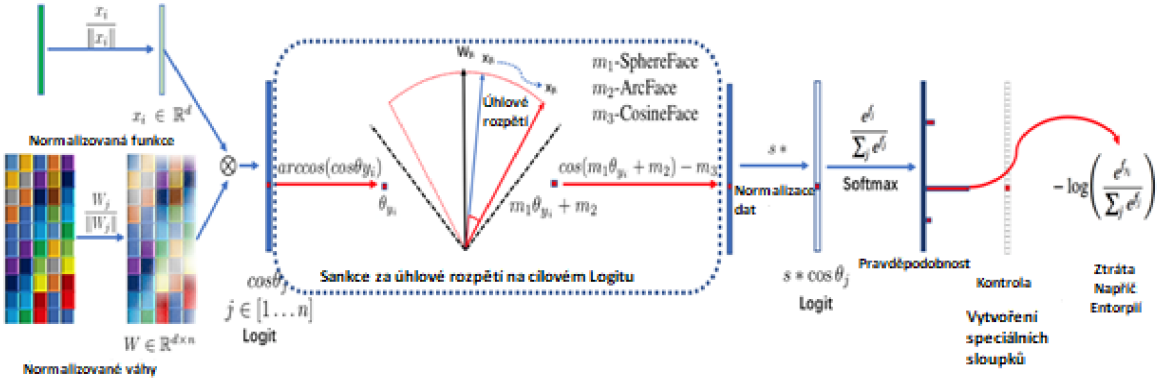
Na stránkách InsightFacu, jsou poskytnuta data ke trénování, nastavení sítě a ztrátové vzory pro rozpoznávání obličeje. Trénovací database je vytvořena ze třech dalších databází, MS1M, VGG2 a CASIA – Webface, které jsou předpřipraveny v binárním MXNet formátu. Mezi páteřní sítě patří ResNet, MobilefaceNet, InceptionResNet_v2 a DenseNet. Mezi ztrátové vzory patří Softmax, SphereFace, CosineFace, ArcFace, Sub-Center ArcFace a Triplet (Euclidian/Angular).

Použitím ArcFacu lze dosáhnout na snímcích LFW úspěšnosti 99,83% a na Megafacu 98%. Tento model pomáhá lidem, kteří se zajímají o problematiku rozpoznávání obličejů, vytvořit algoritmus pro rozpoznávání obličejů rychle díky snadnému používání modelu ve dvou krocích.

1) Stažení předem připravené databáze od tvůrců
2) Spuštění trénování

Síť je natrénována na předem zpracovaných snímcích s obličeji a převedených na velikost 112 × 112 pixelů a zarovnána pomocí 5 obličejových bodů. Pro zabudování sítě jsou použity konvulenční neuronové architektury ResNet50 a ResNet100. Výstupem sítě je 512 dimenzionální vektor, který reprezentuje rysy tváře.

ArcFace může být použit na milióny identit zároveň, ale to by přinášelo problémy jako velkou spotřebu paměti grafické karty a obrovské výpočetní zatížení. Z toho důvodu je v implementaci ArcFacu také možno využít paralelní zpracování k snížení těchto problémů. Tvůrci uvádí, že při paralelním zpracování na 8 grafických kartách s pamětí 11GB, využitím jejich implementace ResNet 50, batch size 8 × 64, feature dimension 512 a float point 32 je rychlost trénování 800 snímků za vteřinu, při milionu identit [1][3].



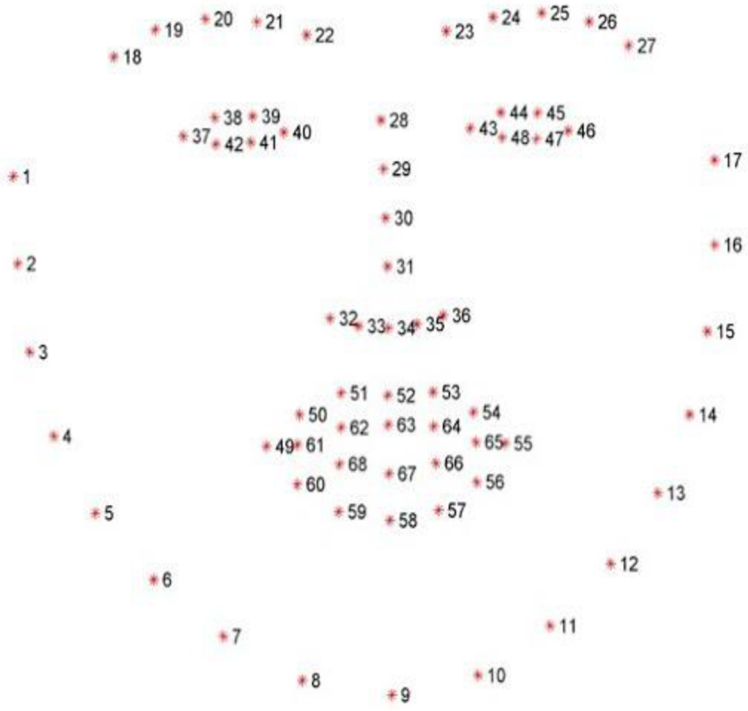
Obr. 1.4 Architektura modelu sítě [1]

1.3.3 OpenFace

OpenFace byl vytvořen trojcí Tadas Baltrušaitis, Peter Robinson a Louis-Philippe Morency. Jedná se o volně dostupný systém, určený pro počítačové vidění a počítačové učení pro ty, kteří mají zájem o vývoj interaktivních aplikací založených na analýze chování obličeje.

OpenFace je první nástroj s otevřeným kódem schopný detekovat orientační body obličeje, odhadnout pozici hlavy, odhadnout výraz obličeje a směr očí. Tento systém je schopný pracovat v reálném čase a provádět úkony pomocí webkamery bez nutnosti dalšího specifického hardwaru. Umožňuje snadnou integraci s jinými aplikacemi a zařízeními prostřednictvím odlehčeného systému zasílání zpráv.

K nalezení obličeje a jeho významných bodů je využita knihovna Dlib. Systém využívá pro detekci rysů obličeje 68 bodů (založeno na knihovně Dlib). Model byl natrénován na databázích LFW, FaceScrub a CASIA – WebFace. Pro optimální výsledky je dle autorů zapotřebí snímek o velikosti 100 × 100 pixelů. Model taktéž dovoluje na snímku, kde se nachází více osob, detekovat všechny obličeje a sledovat obličeje na videu. Na databázi LFW je dle autorů úspěšnost 92,92%. Implementace OpenFace je založena na FaceNetu. A je možné natrénovat vlastní neuronovou síť na vlastní databází snímků [10].



Obr. 1.5 Zobrazení 68 bodů na obličeji [13]

1.3.4 FaceRecognition

FaceRecognition byl vytvořen v roce 2017 Adamem Geitgeyem, z důvodu nepřehlednosti ostatních systémů pro rozpoznávání tváří, které podle Adama jsou seskupením několika programovacích jazyků a potřebují stáhnout dodatečné soubory. Implementace vychází z OpenFace a FaceNet, princip fungování FaceRecognition je stejný jako u OpenFace, ale s tím rozdílem, že neumožňuje natrénovat vlastní neuronovou síť. K nalezení obličejů se stejně jako u OpenFace používá knihovna Dlib, která obsahuje od roku 2017 díky Davisu Kingovi, vlastní model pro rozpoznávání obličeje. Knihovna Dlib má úspěšnost 99,38 % na databázi LFW.

FaceRecognition umožňuje na snímku zobrazit zvýrazněné rysy obličeje, a díky tomu upravit specifickou oblast obličeje, např. zvýraznění rtů přidáním červené barvy. Tato knihovna umožňuje společně s dalšími knihovnami z Pythonu provádět detekci nebo i identifikaci tváře v reálném čase. FaceRecognition také umožňuje nastavit senzitivitu pro vyhodnocování tváří, v případě, že se porovnávaný obličej shoduje s více lidmi z databáze, je to pravděpodobně proto, že jsou si tito lidé velmi podobní, nastavením tolerance dojde k užšímu výběru a výsledky budou přesnější. Detekce obličeje může být také urychlena použitím více jader procesoru, pokud je procesor obsahuje [5].

2. ZPROVOZNĚNÍ PROSTŘEDÍ

V této části je uvedeno, jak zprovoznit algoritmus InsightFace, jednoduchý popis použitých algoritmů, návod, jak převést model z formátu MXNet do formátu ONNX a výsledky modelu bez roušek.

2.1 Výběr algoritmu

Pro vytvoření modelu, který by dokázal rozpoznat obličej s částečně zahalenou tváří, byl vybrán algoritmus InsightFace z důvodu možnosti vytvoření vlastního modelu pro rozpoznávání tváří na předem poskytnuté nebo na vlastní databázi. InsightFace má úspěšnost 99,83 %, což je největší úspěšnost ze všech zmíněných algoritmů v této práci a také využívá 512 embeddings, což při identifikaci pro osobu, co má zahalenou tvář, může být velice nápomocné, neboť bude k dispozici více nezakrytých bodů, se kterými bude možné jednoduše porovnávat.

V první části bude InsightFace použit pro vytvoření modelu, který dokáže rozpoznávat tváře z databáze, kterou poskytli tvůrci algoritmu. Takto natrénovaný model bude použit pro vytvoření programu k identifikaci lidí.

V druhé části bude vytvořena databáze lidí s rouškami, a tato databáze bude použita pro vytvoření modelu pomocí InsightFace, pro rozpoznávání lidí s částečně zahalenou tváří.

2.2 Použité Frameworky

CUDA – je zkratka pro Compute Unified Device Architecture, jedná se o hardwerovou a softwarovou architekrutu, která umožňuje na GPU – grafický procesor, spouštět programy napsané v jazycích C, C++, Fortran, Python a MATLAB, nebo programy postavené na technologiích OpenCL, DirectCompute, a jiných. Pro použití této architektury je nutné vlastnit grafickou kartu společnosti NVIDIA, která tuto architekturu vyvinula. S CUDOu, mohou vývojáři dramaticky urychlit výpočetní aplikace použitím výkonu GPU. V tomto případě šlo o verzi CUDY 10.1 pro grafickou kartu 2047MB NVIDIA GeForce GTX 1050 (HP) [8].

CuDNN - knihovna pro GPU vytvořená společností NVIDIA, která poskytuje základy pro hluboké neuronové sítě jako vysoce vyladěné implementace pro standartní rutiny jako je dopředná a zpětná konvoluce, sdružování, normalizace a aktivační vrstvy, které jsou později použity platformou pro strojové učení zvanou TensorFlow [9].

TensorFlow – otevřená knihovna pro strojové učení, která je zaměřena na trénink a práci s hlubokými neuronovými sítěmi. Byl vyvinut společností Google pro vnitřní použití, ale v roce 2015 byl představen veřejnosti. Tensorflow může být použit k trénování a

provozování hluboké neuronové sítě pro klasifikaci ručně psaných číslic, rozpoznávání obrázků, vkládání slov, rekurentní neuronové sítě, zpracování přizozeného jazyka a pro simulaci založenou na parciálních diferenciálních rovnicích [12].

Anaconda – veřejně dostupný program z distribuce Python. Tato distribuce je vhodná pro operační systémy Windows, Linux a macOS. Anaconda byla vytvořena společností Anaconda Inc, kterou založil Peter Wang společně s Travis Oliphient v roce 2012. Anaconda umožňuje vytvořit virtuální prostředí, kde je možné určit s jakou verzí programovacího jazyka chceme pracovat, a instalovat balíčky starších nebo novějších verzí programu, bez jejich změny v normálním pracovním prostředí. Tím je možné vytvořit programovací prostředí pro různé učely jako, vývoj webových aplikací nebo pro jiné odvětví, kde je nutné využít jiné verze používaných knihoven [22].

Pycharm – jedná se o integrované vyvojové prostředí používané v počítačovém programování, specializující se na programovací jazyk Python. Byl vytvořen českou společností JetBrains. Lze ho použít s kooperaci s Anacondou a lze ho použít na operačních systémech Windows, Linux a macOS. Jedná se o volnou verzi od roku 2013 [15].

OpenCV – je otevřená multiplatformní knihovna pro manipulaci s obrazem a počítačové vidění v reálném čase. Knihovnu lze využít v prostředí programovacích jazyků C, C++, Python a Octave [7].

Numpy – základní knihovna pro vědce a analytiky, kteří pracují s Pythonem. Numpy definuje typ pro n-rozměrné homogenní pole a API pro práci s takovým polem. Základní vlasností polí Numpy je, že aritmetické operace s čísly fungují po prvcích [6].

Dlib – knihovna z jazyka C++ obsahující algoritmy pro strojové učení a nástroje pro vytváření komplexního softwaru pro řešení reálných problému. Dlib je volně dostupná knihovna s otevřeným zdrojovým kódem pro každou aplikaci, kterou lze využít v jazyce Python [16].

SQLite – knihovna v jazyce C, která dovoluje vytvářet nenáročné databáze bez nutnosti spolupráce serveru a dovoluje k databázi přistupovat pomocí nestandartních příkazů v jazyce SQL. Je součástí programu PyCharm [28].

Cython – je nadmnožinou jazyka Python, ale Cython navíc podporuje volitelné statické psaní pro nativní volání funkcí C, práci s třidami C++ a reklaraci rychlých typů C na proměnné a atributy třídy [30].

2.3 Zprovoznění Frameworku

Pro práci s algoritmy pro rozpoznání obličeje je nejprve nutné zvolit správné pracovní prostředí. V tomto případě byl zvolen operační system LINUX, konkrétně distribuce Ubuntu verze 18.04. Poté bylo potřeba naklonovat repozítář z githubu, a upgradovat ovladače grafické karty, s podporou CUDA, kvůli rychlosti zpracování.

Po instalaci ovladačů pro grafickou kartu bylo nutné stáhnout CuDNN pro odpovídající verzi CUDY. Poté je potřeba upravit soubor bashrc přidáním těchto příkazů:

```
export
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/cuda/extras/CUPTI/lib64:/u
sr/local/cuda/lib64

export CUDA_HOME=/usr/local/cuda
```

Dále je nutné nainstalovat MXNet pro danou verzi CUDY s podporou GPU pro rychlejší výpočty. V mém případě chyběla knihovna libcublas.so.10, proto ji bylo nutné dodatečně stáhnout.A v neposlední řadě bylo nutné nainstalovat programy, se kterými byla dále prováděná práce. Anaconda a PyCharm.

Po dokončení těchto nezbytných kroků a instalaci chybějích knihoven pro InsightFace bylo přikročeno k dalšímu kroku a to vytvoření vlastního modelu pro rozpoznávání obličejů.

Nejprve byly staženy databáze LFW, CFP_FP a Agedb_30, v binárním tvaru o celkové velikosti 14GigaByte, se kterou InsightFace pracuje, a je poskytnuta od autorů projektu. Tyto databáze byly poté vloženy do složky /insightface/datasets/. Toto je místo kde program hledá data k trénování, a proto je nutné je vložit právě sem.
Poté je pomocí příkazu

```
CUDA_VISIBLE_DEVICES='0,1,2,3' python -u train.py --network r100 --loss arcface
--dataset emore
```

spuštěno trénování.
Celý postup i s příkazy, které nejsou zveřejněny jsou dostupné na [27].
LFW – databáze tohoto typu obsahují velké množství fotografií lidí zachycených v různých situacích s různými výrazy, pozadím, věkem, barvou pleti, vlasů, kvality atd. Poskytnutá databáze obsahuje 5749 identit, 13223 fotografií a 7000 párů. 1680 osob v databázi má 2 nebo více fotografií, a zbylí po jedné fotografii. Každá osoba v databázi má unikátní označení, v případě, že se jedná o shodná jména jsou od sebe rozlišeny[20][26].

Obr. 2.1 Příklad databáze celebrit v různých situacích a výrazem

CFP_FP – tyto databáze obsahují fotografie celebrit z předního pohledu a z profilu. Poskytnutá databáze obsahuje 500 identit, 7000 obrázků a 7000 párů. Pro vytvoření této databáze byla vždy vyhledána osoba z profilu a předního pohledu. Pro každou osobu bylo vybráno 10 fotografií z předního pohledu a 4 z profilu. Tato databáze je vyvážená, protože obsahuje podobné množství mužů i žen a podobné množství fotografií s ohledem na rasu. [19].

Obr. 2.2 Příklad databáze celebrit z předního pohledu a profilu

AgeDB – tato databáze obsahuje fotografie lidí zachycených v různých situacích, pozicích a věku. Poskytnutá databáze obsahuje 570 identit, 12 240 obrázků a 6000 párů průměrný počet fotografií pro každou osobu je 29, přičemž nejmenší počet fotografií pro osobu je 1 a největší 101, průměrný věk je 50,3 let. Tato databáze může být použita pro naučení modelu pro odhadování věku nebo pro naučení modelu pro odhadnutí podoby se zvyšujícím se věkem [18].

Obr. 2.3 Příklad databáze s věkem

Na mém počítači nebylo možné spustit trénování z důvodu malé paměti na grafické kartě, a tak bylo trénování sítě spuštěno na výpočetním serveru po dobu jednoho týdne, a natrénováná síť dosáhla těchto vlastností:

```
(testovací sada z databází LFW, CFP_FP a AgeDB_30):
testing verification..
[lfw][1614000]XNorm: 22.025256
[lfw][1614000]Accuracy-Flip: 0.96733+-0.00821
testing verification..
(14000, 512)
infer time 108.98324300000009
[cfp_fp][1614000]XNorm: 19.023561
[cfp_fp][1614000]Accuracy-Flip: 0.72800+-0.01385
testing verification..
(12000, 512)
infer time 93.42299000000006
[agedb_30][1614000]XNorm: 20.849768
[agedb_30][1614000]Accuracy-Flip: 0.84133+-0.01903
[1614000]Accuracy-Highest: 0.84817
```

Obr. 2.4 Výsledky modelu

Jak lze vidět na výše uvedených číslech, natrénovaný model nedosahuje takových výsledků jakých dosáhli autoři z důvodu nutnosti ukončit trénování k uvolnění práce na výpočtovém serveru, ale pro účely testování tyto hodnoty stačí, neboť se jedná o model pro rozpoznávání obličejů bez roušky a cílem je pouze ověřit jeho funkčnost.

2.4 Ověření funkčnosti modelu

Po vytvoření modelu pro rozpoznávání obličejů, byl tento model ve formátu MXNet, protože knihovna OpenCV neumí s tímto formátem pracovat, bylo nutné model převést do formátu ONNX, se kterým už pracovat lze.

K převedení do MXNet formátu je využit program PyCharm kde pomocí funkce onnx.export_model(), dojde po poskytnutí potřebných parametrů k převedení do ONNX formátu. Potřebnými parametry jsou soubor .json, soubor .param, vstupní tvar a místo kam se má výsledný formát onnx uložit. Použití funkce:

```
onnx_mxnet.export_model('/home/kaspis/ArcFace/model-symbol.json',
'/home/kaspis/ArcFace/model-0001.params', (1, 3, 112, 112), np.float32,
'/home/kaspis/converted-model.onnx)
```

Poté dojde k využití modelu, kde v rámci procesu jeho trénování, dojde k nalezení optimálních příznaků k rozlišení jednotlivých obličejů.

Po načtení fotografie do programu se pro danou fotografii, na které je obličej, vypočítají tyto příznaky. Takže daný obraz tváře je reprezentován daným vektorem příznaků, v tomto případě vektorem s 512 hodnotami. Tyto hodnoty jsou uloženy do databáze pro pozdější využití. Tato databáze obsahovala 1000 osob kde pro každou osobu byly její 2 fotografie převedeny na vektory a uloženy do textového souboru. 1000 osob bylo zvoleno z důvodu velké škály různorodých obličejů a také z důvodu časové náročnosti. Při použití 1000 osob, kde každý má 2 fotografie se jedná o 2000 vektorů, které se mezi sebou budou porovnávat. Což se celkově rovná 1999000 porovnání, která vždycky proběhnou pro jednu hodnotu euklidovské vzdálenosti. Takto vzniklo 1000 textových souborů, pomocí kterých proběhla N:N identifikace, tedy každý vektor se porovnal s každým. Nejdřívě jsou načteny vektory jedné osoby a porovnány mezi sebou, pokud jsou si podobné tak se jedná o shodu tedy TP – Pravdivě pozitivní, pokud nejsou tak se jedná o FN – Falešně negativní. Poté jsou postupně načítany vektory dalších osob a porovnávány s vektory první osoby, pokud zde nastane shoda tak se jedná o FP – Falešně pozitivní, a pokud shoda nenastane jedná se o TN – Pravdivě negativní. Jakmile jsou vektory první osoby porovnány se všemi ostatními, tak se načtou vektoru druhé osoby a celý proces se opakuje dokud nebou porovnány všechny vektory, přičemž vektory, které již byly porovnány se všemi ostatními se dále neporovnávají. Je nutné si uvědomit, že hodnoty TP, FN, FP a TN se budou měnit v závislosti na velikosti euklidovské vzdálenosti.

Euklidovská vzdálenost – V matematice je euklidovská vzdálenost, vzdálenost mezi dvěma body v euklidovském prostoru číslo, což je délka úsečky mezi dvěma body. Lze

jej vypočítat z kartézských souřadnic bodů pomocí Pythagorovy věty a příležitostně se nazývá Pythagorova vzdálenost [17].

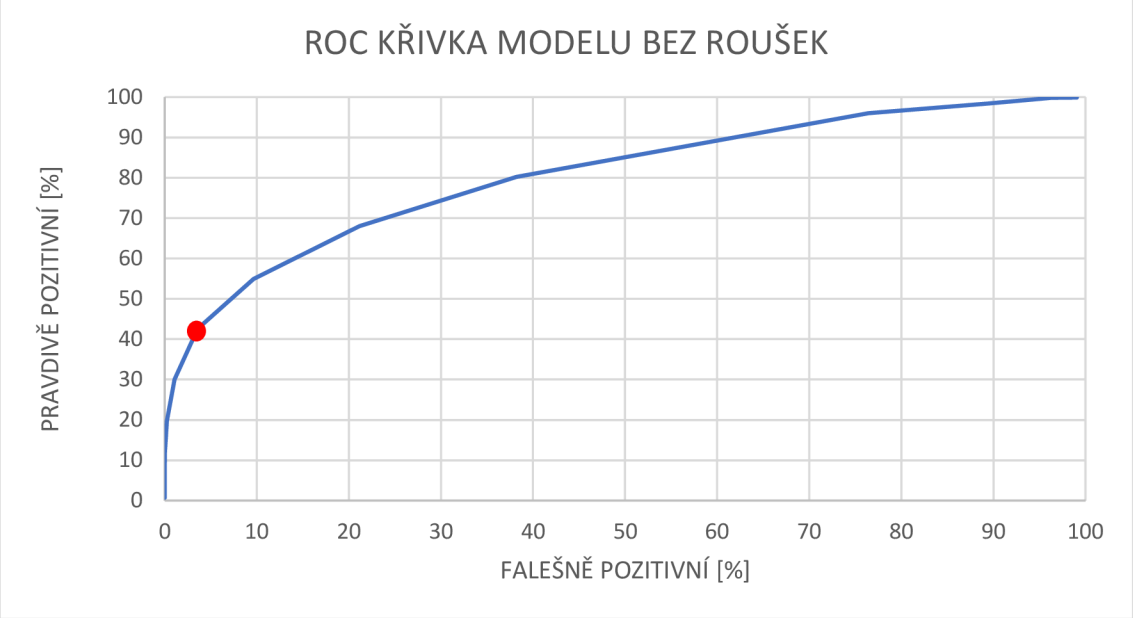
Vzorec pro zjištění euklidovské vzdálenosti pro n dimenzionální rozměr.

$$\mathrm{d}(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 \ldots + (p_n - q_n)^2}, \quad (2.\ 1)$$

kde p je souřadnice na ose X a q souřadnice na ose Y.

Tabulka 1 Výsledky N:N identifikace modelu bez roušky

Vzdálenost	TP [%]	FN [%]	FP [%]	TN [%]
0,55	0,70	99,30	0	100
0,6	1,00	99,00	0	100
0,65	1,90	98,10	0	100
0,7	3,60	96,40	0	100
0,75	7,00	93,00	0,01	99,99
0,8	12,10	87,90	0,04	99,96
0,85	19,70	80,30	0,25	99,75
0,9	30,00	70,00	1,06	98,94
0,95	42,60	57,40	3,58	96,42
1	54,90	45,10	9,67	90,33
1,05	68,00	32,00	21,14	78,86
1,1	80,20	19,80	38,22	61,78
1,15	88,50	11,50	58,19	41,81
1,2	96,00	4,00	76,42	23,58
1,25	98,40	1,60	89,32	10,68
1,3	99,80	0,20	96,26	3,74
1,35	99,90	0,10	99,06	0,94



Obr. 2.5 ROC Křivka modelu bez roušek

Tento graf ukazuje, jak se projeví zvýšení hodnoty euklidovské vzdálenosti na TP (pravdivě pozitivních) a FP (falešně pozitivních). V bodě [0,0] je hodnota euklidovské vzdálenosti nastavena na hodnotu 0,55 a postupně se zvyšuje o 0,05 na hodnotu 0,95 na grafu označenou červeným bodem, při této vzdálenosti dochází ke zlomu a velkému navyšování FP. Dalším zvyšováním hodnot začne docházet k nárůstu TP a FP. Při hodnotě 1,35 je dosaženo maximálního počtu TP v tomto případě dojde ke shodě se všemi fotografiemi v databázi pro danou celebritu. Dalším zvyšováním dojde pouze ke zvýšení počtu FP, protože TP již dosáhlo maxima.

3. VYTVOŘENÍ MODELU

3.1 Vytvoření databáze lidí s rouškami

Jelikož téma této semestrální práce je identifikace osob s částečně zahalenou tváří, je nutné mít natrénovaný model, který dokáže na fotografii, kde je částečně zakrytá tvář, identifikovat obličej a převést nalezené rysy do 512 vektorů.

Pro úspěšné natrénování modelu, který by dokázal rozpoznat částečně zahalené tváře, je důležité mít databázi, která bude obsahovat snímky částečně zahalených obličejů. Protože databáze, která by obsahovala velké množství identit a fotek lidí, kteří mají částečně zahalený obličej, nebyla volně dostupná, došlo k vytvoření vlastní databáze, kde byly roušky na obličeje přidány pomocí programu. Roušky, které byly použity pro vytvoření databáze, jsou k dispozici na přiloženém CD.

Příklad celebrity před a po přidaní roušky pomocí programu.

Obr. 3.1 Před rouškou

Obr. 3.2 Po přidání roušky

Program pro přidávání roušek není dokonalý a v některých případech dojde ke špatnému přidání roušky.

Obr. 3.4 Špatně přidaná rouška

Obr. 3.3 Špatně přidaná rouška

Program pro přidávání roušek funguje následovně a je vysvětlen pomocí Obr. 1.5 pro nejlepší možný případ.

Po načtení obrázku je detekována tvář a její specifické body (1-67). Pokud jsou nalezeny body pro čelist (1–17) a nosní přepážku (28-31), program pokračuje, pokud nejsou nalezeny, program se ukončí. Poté jsou podle počtu nalezených přiznaků pro nosní přepážku a čelist určeny nejlepší orientační body pro přidání roušky. Nosní přepážka (29), čelist (9), levý bod čelisti (2) a pravý bod čelisti (16). Dále je vypočítána velikost roušky podle bodu nosu (29) a čelisti (9). Následně je rouška rozdělena na pravou a levou část a pro obě části je upravena jejich velikost, pro levou část podle vzdáleností bodů (29), (9) a (2), pro pravou část podle vzdáleností bodů (29), (9) a (16). Každý bod na obrázku lze vyjádřit pomocí souřadnic x a y, tedy všechny využívané body lze použít, neboť je známo, kde se nachází a jsou uloženy ve formátu [x, y], například [120, 150] Následující rovnice popisuje výpočet vzdálenosti, která je poté využita k změnění velikosti roušky.

$$Vzdálenost = \frac{|(a[y]-b[y])*c[x]+(b[x]-a[x])*c[y]+(a[x]+b[y])*b[y]+(b[y]-a[y])*b[x]|}{\sqrt{(a[y]-b[y])*(a[y]-b[y])+(b[x]-a[x])*(b[x]-a[x]}}, \qquad (3.\ 1)$$

Kde a je spodní bod brady (9) b bod nosu (29) a c může být buď bod (16) nebo (2) v závislosti, zda je počítána pravá nebo levá strana tváře, x představuje x-ovou souřadnici daného bodu a y y-lonovou souřadnici daného bodu. Takto vypočítaná vzdálenost je poté vynásobena 1,2 a část roušky zvětšena o danou velikost. Takto upravené části jsou spojeny dohromady a je vypočítán úhel natočení podle bodů (29) a (9). Poté dojde k výpočtu souřadnic pro umístění roušky na tvář.

$$X = (\lfloor b[x] + a[x]) : 2 \rfloor + (\lfloor W : 2 \rfloor - WL) * \cos\left(U * \frac{\pi}{180}\right) - (\lfloor WRM : 2 \rfloor), \qquad (3.\ 2)$$

$$\mathrm{Y} = (\lfloor b[y] + a[y]) : 2 \rfloor + (\lfloor W : 2 \rfloor - WL) * \sin\left(U * \frac{\pi}{180}\right) - (\lfloor HRM : 2 \rfloor), \qquad (3.\ 3)$$

Kde b je bod nosu (29) a bod brady (9) x představuje x-ovou souřadnici daného bodu a y y-lonovou souřadnici daného bodu, W je šířka masky, WL je šířka levé části masky, U je vypočítaný úhel, WRM je šířka natočené masky, HRM je výška natočené masky.

Výsledná databáze je založena na volně dostupné databázi celebrit CASIA – WebFace, která obsahuje více než 450 000 fotografií a 10 500 různých identit. Pro přidávání roušek na fotografii byl využit volně dostupný program face-mask verze 0.3.0[23], který po zadání

```
python face-mask.py /cesta/k/obrázku/kam/chceme/přidat/roušku
```

dokáže na obličej přidat roušku. Tento program byl upraven pro přidávání roušek na fotografie v databázi.

3.2 Převedení dat do formátu pro trénování

Takto vytvořené fotografie je poté nutné převést do formátu pro trénování, protože InsightFace neumí pracovat s daty jako takovými, ale pouze s daty v MXNet binárním formátu. Avšak je důležité, aby data byla uložena ve správném formátu, tak aby se poskytnuté programy nemusely upravovat, jedná se především o názvy složek a souborů ve složkách. Ty musí být ve formátu LFW, tedy název složky je ve tvaru Jméno_Přijmení a názvy souborů ve složkách Jméno_Přijmení_000x.jpg, kde x reprezentuje číslo snímku, a s každým dalším souborem ve složce se zvětšuje o 1. Příklad, jak by měl vypadat správný zápis: Bruce_Lee/Bruce_Lee_0001.jpg

Bohužel databáze nebyla v tomto formátu, a tak byl vytvořen program, který ji do tohoto formátu převedl. K trénování není nutné, aby bylo známo jméno osoby, a tak byly místo jména a přijmení použita čísla, tak aby to odpovídalo LFW formátu tedy: 1_1/1_1_0001.jpg a pro každou další osobu se čísla o 1 zvětsily tedy 2_2/2_2_0001.jpg.

Pro převedení do formátu MXNet bylo postupováno podle návodu [25] od bodu 6.
1) Zmenšení dat na velikost 112 × 112 pixelů a zarovnání tváře.

Obr. 3.6 Před zarovnáním a zmenšením

Obr. 3.5 Po zmenšení a zarovnání

Pro převedení na tuto velikost a tvar je použit modul align_dataset_mtcnn.py() z projektu facenet [14], který je volán s parametry: vstupní složka (zdrojová složka obrázků), výstupní složka (složka, kam budou umístěny upravené obrázky) a cílová velikost. Příklad použití modulu pomocí terminálu :

```
python align_dataset_mtcnn.py /home/kaspis/ToAlign /home/kaspis/Aligned --image_size 112
```

2) Rozdělení dat do složek Train, Test a Valid, k tomu slouží programy dostupné [25] train_test() a train_valid() .

3) Vytvoření list.lst, kde dojde k vytvoření seznamu v abecedním pořadí, které je potřebné pro správné trénování, v tomto kroku je použita složka Train. Pro vytvoření list.lst je použit modul insightface_pairs_gen_v1.py a jeho funkce write_item_label_abc(), do které jsou předány parametry: vstupní složka (zdrojová složka obrázků), list file (cesta k souboru list.lst) a označení koncovky souboru (.png, .jpg, .jpeg) Příklad použití modulu pomocí terminálu:

```
python insightface_pairs_gen_v1.py --dataset-dir /home/kaspis/Train --list-file /home/kaspis/list.lst --img-ext .jpg
```

Příklad, jak by mohl vypadat obsah souboru list.lst:

```
1                    /cesta/k/obrázku1                    0
1                    /cesta/k/obrázku2                    0
.....
1                    /cesta/k/obrázku3                    1
1                    /cesta/k/obrázku4                    1
.....
1                    /cesta/k/obrázku5                    10000
1                    /cesta/k/obrázku6                    10000
```

4) Vytvoření souboru property, tento soubor obsahuje informace o počtu identit a velikosti dat, například pro 10 000 identit a velikost obrázků 112 × 112 pixelů bude obsah 10000,112,112

5) Vytvoření souborů train.rec a train.idx, pro jejich vytvoření je použit soubor property a složka Train. Tím dojde k vytvoření seznamu, kde je uvedena cesta k obrázku a ID osoby, kde ID je identické pro obrázky stejné osoby. Pro vytvoření těchto souborů je použit modul face2rec2.py s parametrem list.lst dostupný [3] ve složce /recognition/tools. Příklad použití modulu pomocí terminálu:

```
python face2rec2.py /home/kaspis/list.lst
```

6) Pro vytvoření souboru pairs.txt je využita složka Valid. Zde dojde k vytvoření shodných párů, tedy obrázků, které jsou od jedné osoby, a neshodných párů, tedy obrázků, které jsou od 2 různých osob. Pro vytvoření soubor pairs.txt je využit modul gen_pairs_lfw.py a jeho funkce generate(), do které jsou předány parametry: vstupní složka (zdrojová složka obrázků) a text file (cesta k souboru pairs.txt) Příklad použití modulu pomocí terminálu:

```
python gen_pairs_lfw.py --data-dir /home/kaspis/Valided --txt-file /home/kaspis/Valided/pairs.txt
```

7) Pro vytvoření souboru lfw.bin je použita složka Valid a soubor pairs.txt. Ze souboru pairs.txt dojde k přečtení všech párů a jejich uložení podle toho, zda se jedná o shodné páry. Pro vytvoření souboru lfw.bin je použit modul dataset2bin.py a modul lfw.py, na který se dataset2bin.py odkazuje. Modul je volán s parametry: vstupní složka (zdrojová složka obrázků), velikost obrázků a cesta, kam se má lfw.bin uložit. Je nutné, aby soubor pairs.txt byl ve stejné složce jako vstupní složka, protože program bude předpokládat, že tam ji nalezne. Zde je nutné dávat pozor, aby soubor lfw.bin nebyl moc velký a pokud by byl, trénování by nezačalo kvůli chybě grafické karty a nedostatku její paměti. Pro trénování na výpočtovém serveru, kde má grafická karta velikost 11GB, by soubor lfw.bin měl mít kolem 120MB – 170MB. Příklad použití modulu pomocí terminálu:

```
python dataset2bin.py --data-dir /home/kaspis/Valided --image-size 112,112 --output /home/kaspis/lfw.bin
```

8) Soubory property, list.lst, train.rec a train.idx jsou uloženy do složky faces_emore a ta umístěna do /insightface/datasets/
9) Pokud vše proběhlo v pořádku je možné spustit trénování.
10) U všech příkazů pro používání daných modulů je nutné se nacházet ve složce, kde se nacházejí dané moduly nebo modul volat celou cestou. Pokud bude modul uložen na místě /home/kaspis/123/aa/bb/modul.py a v terminálu se budeme nacházet na /home/kaspis, bude příkaz k použití modulu pomocí terminálu:

```
python /home/kaspis/123/aa/bb/modul.py
```

3.3 Spuštění trénování na výpočetním serveru

Pro spuštění trénování modelu pro rozpoznávání osob se zakrytým obličejem byl použit výpočtový server.
Hardwarová konfigurace výpočtového serveru:
Pevné disky – Samsung SSD 960 EVO 250 GB, ST2000DM006-2DM164 2TB
Grafické karty – NVIDIA GeForce RTX 2080 8 GB, NVIDIA GeForce RTX 2080 Ti 11 GB
Procesor – 16x AMD Ryzen 7 1800X 8 jader
RAM – 32 GB

Jako první byla zjištěna verze CUDy na serveru

```
nvidia-smi
```

Poté byl nainstalován MXNet s podporou pro grafickou kartu a odpovídající verzí CUDy 10.1

```
pip install mxnet-cu101
```

A naklonován repositář InsightFace

```
git clone –recursive https://gitbug.com/deepinsight/insightface.git
```

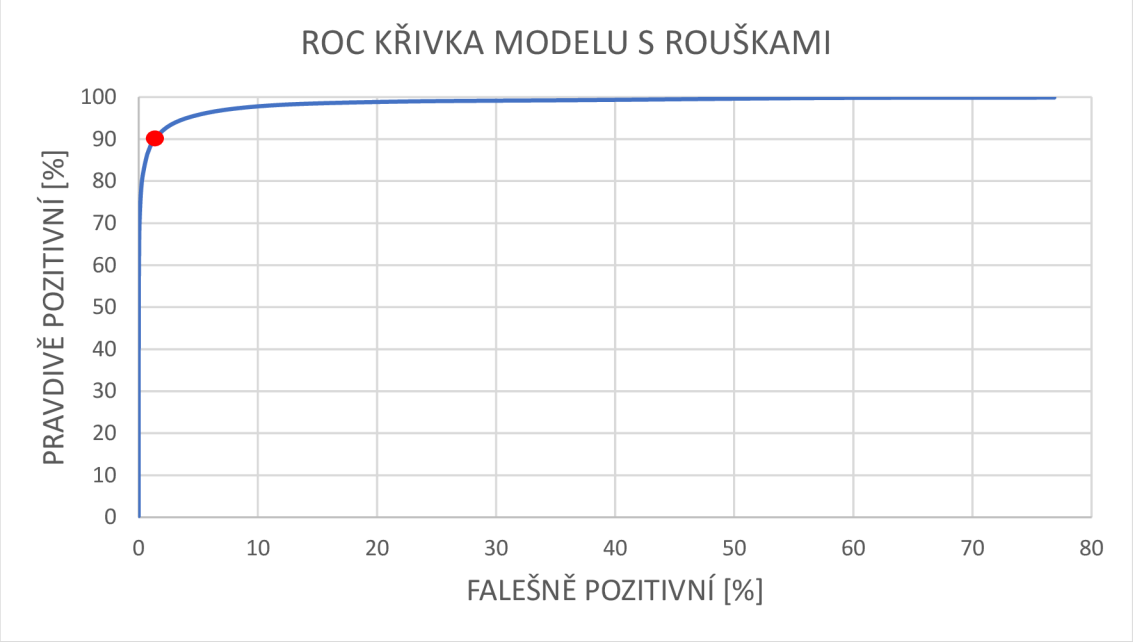
Poté byly do složky insightface/recognition/datasets umístěny soubory property, lfw.bin, train.idx a train.rec, které byly vytvořeny v minulém kroku. Následně došlo ke zkopírování složky sample_config.py a jejímu přejmenování na config.py. V této složce byla poté změněna batch_size z původních 128 na 16, protože tvůrci používají pro trénování 4 grafické karty s pamětí 18 GB. Nakonec byl v příkazovém řádku spuštěn příkaz

```
set CUDA_VISIBLE_DEVICES=1 & python -u train.py --network r100 --loss arcface --dataset emore
```

čímž se spustilo trénování. Je dobré trénování sledovat, a v případě, že dochází k rychlému snižování loss function a přesnost je stále 0,00, tak zastavit trénování a změnit learning_rate na 0,001 z původních 0,1. Trénování dosáhlo přesnosti 93,321%.

3.4 Ověření přesnosti modelu pro zakryté tváře

Natrénovaný model byl převeden do formátu ONNX stejně jako v 2.4 a bylo náhodně vybráno 1000 osob, pro každou osobu 2 fotografie. Těmto osobám byly přidány roušky a proběhlo N:N testování ke zjištění přesnosti modelu.



Obr. 3.7 ROC Křivka modelu s rouškami

V bodě [0,0] je hodnota euklidovské vzdálenosti nastavena na hodnotu 0,5 a postupně se zvyšuje o 0,05 při hodnotě 1,1, na grafu označenou červeným bodem, dochází k zlomu a velkému navyšování FP. Dalším zvyšováním hodnot začne docházet k malému nárůstu TP a velkému nárůstu FP. Při hodnotě 1,4 je dosaženo maximálního počtu TP, v tomto případě dojde ke shodě se všemi fotografiemi v databázi pro danou osobu. Dalším zvyšováním dojde pouze ke zvýšení počtu FP, protože TP již dosáhlo maxima.

Při porovnání s grafem pro model bez roušek lze vidět, že graf modelu s rouškami dosahuje mnohem lepší přesnosti v poměru pravdivě pozitivní / falešně pozitivní a lze u něho zvolit větší euklidovskou vzdálenost a tím správně rozeznat snímky k sobě patřící beze strachu, že nastanou falešně pozitivní shody.

Tato skutečnost je pravděpodobně způsobena maximální natrénovanou přesností obou modelů, kdy model s rouškami dosáhnul 93 % a model bez roušek celkové přesnosti 84 %. Dalším důvodem by mohlo být, že při testování modelu s rouškami jsou použity tváře pouze z předního pohledu, a i na takové tváře je model natrénován. Ale při testování modelu bez roušek jsou použity i tváře z profilu, což mohlo ovlivnit výslednou přesnost grafu.

Tabulka 2 Výsledky N:N identifikace modelu s rouškou

Vzdálenost	TP [%]	FN [%]	FP [%]	TN [%]
0,5	0,30	99,70	0	100
0,55	1,00	99,00	0	100
0,6	2,00	98,00	0	100
0,65	4,60	95,40	0	100
0,7	8,00	92,00	0	100
0,75	14,20	85,80	0	100
0,8	23,20	76,80	0	100
0,85	33,20	66,80	0	100
0,9	46,10	53,90	0	100
0,95	58,70	41,30	0,02	99,98
1	71,70	28,30	0,09	99,91
1,05	82,10	17,90	0,38	99,62
1,1	90,00	10,00	1,34	98,66
1,15	95,00	5,00	4,01	95,99
1,2	97,80	2,20	9,97	90,03
1,25	98,90	1,10	20,98	79,02
1,3	99,30	0,70	37,69	62,31
1,35	99,80	0,20	57,88	42,12
1,4	99,90	0,10	76,87	23,13

Hodnoty v tabulce popisují, jaká nastala procentuální shoda u pravdivě pozitivních, falešně negativních, falešně pozitivních a pravdivě negativních v závislosti na velikosti euklidovské vzdálenosti. Přičemž nejlepší možný výsledek u pravdivě pozitivních a pravdivě negativních by se měl pro nejlepší možný případ blížit 100 % a u falešně negativních a falešně pozitivních by se naopak měl blížit 0 %.

4. PROGRAM PRO IDENTIFIKACI

V této části bakalářské práce je popsán program pro identifikaci osob, detektor RetinaFaceAntiCovid pro detekci zahaleného obličeje a návod, jak ho zprovoznit ve formátu MXNet, je vysvětleno, z jakého důvodu nejsou použity textové soubory, ale databáze vytvořené pomocí SQLite, uveden blokový diagram rozšířeného programu pro identifikaci a výpis programu po provedení identifikace.

4.1 Program pro identifikaci osob se zakrytým obličejem a odkrytým obličejem

Stejně jako při ověřování přesnosti modelu pro zakryté tváře byla vytvořena databáze obsahující 1000 osob, kde každá osoba má 2 fotografie a druhá databáze, kde osoby mají nezakrytý obličej. Vektory osob byly uloženy do textových souborů, jejichž názvy odpovídají osobě, jejíž vektory jsou zde uloženy. Po zadání cesty do programu k příslušné fotografii osoby dojde k vytvoření vektorů z fotografie a tyto vektory jsou poté porovnány se všemi vektory z příslušné databáze. Výsledkem je výpis všech osob, se kterými si při dané euklidovské vzdálenosti je fotografie podobná. Čas pro zjištění, s kým si je daná osoba podobná, trval při 1000 osobách a 2 fotografiích pro osobu asi 14,5 vteřiny. Tyto databáze v podobě textových souborů nejsou moc praktické a jejich výsledná velikost byla větší, než mají samotné fotografie okolo 22 MB, a tak byly vytvořeny databáze pomocí SQLite v jazyce python, které databáze v textových souborech nahradily.

V první části byly stejně jako u textových souborů vytvořeny databáze, které obsahovaly 1000 osob, pro každou 2 fotografie, s tím rozdílem, že místo do textových souborů se data ukládají do SQLite databáze. Každý záznam obsahuje unikátní ID, které se s každým přidaným záznamem o 1 zvětší, jméno, aby bylo možné osobu identifikovat, a vektory, které jsou ukládány ve formě BLOB. Po zadání cesty k fotografii se prohledá celá databáze a na konci je vypsáno, se kterými osobami si je zadaná fotografie nejvíce podobná. Toto prohledávání zabere asi 6,5 vteřiny, což je více než dvojnásobná rychlost oproti použití textových souborů a velikost těchto databází se pohybuje okolo 2,3 MB, takže se jedná o velké zlepšení oproti textovým souborům.

Problémem jak u databází pomocí SQLite tak u databází pomocí textových souborů je nutnost vždy zadat jako vstupní obrázek, který má velikost 112 × 112 pixelů, protože na tuto velikost je model natrénován. Pokud je jiného rozměru, tak program zamrzne. Pro vyřešení tohoto problému byl použit model RetinaFaceAntiCovid [29] od tvůrců InsightFace.

4.2 RetinaFaceAntiCovid detektor

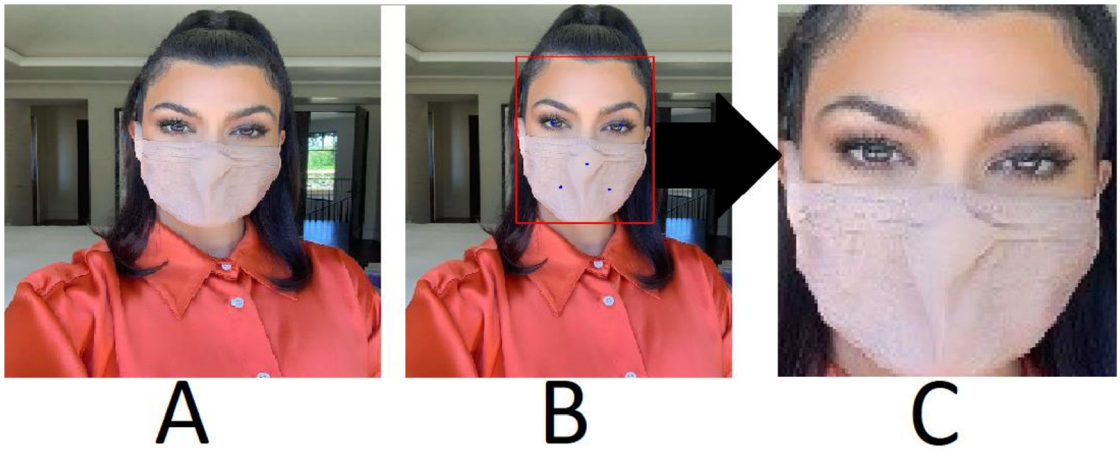
Tento detektor dokáže na obrázku najít tvář, i když je zakrytá, a označit důležité části, toho je využito pro vytvoření snímku obličeje o velikosti 112 × 112 pixelů.

V první řadě byly staženy soubory params a json dostupné [29], které jsou ve formátu MXNet. Poté byl učiněn pokus převést tyto soubory do formátu ONNX jako v bodě 2.4 Bohužel se nejedná o stejný formát jako mají modely pro trénování, a tak se toto převedení nepovedlo.

Z tohoto důvodu je tento detektor používán ve formátu MXNet. Potřebné soubory jsou součástí InsightFace, které již bylo staženo v předchozích krocích. Nejprve byla ze složky RetinaFace zkopírována složka cython, která obsahuje potřebné soubory a hlavně setup.py. Tato složka byla poté umístěna do složky rcnn ve složce RetinaFaceAntiCov. Poté pomocí terminálu byl spuštěn příkaz

```
python setup.py build_ext –inplace
```

ve složce cython. Tím došlo k vytvoření potřebných knihoven v jazyce C a souborů pro fungování tohoto detektoru. Detektor byl poté otestován, zda pracuje správně pomocí vložení fotografie.



Obr. 4.1 Upravený model pro detekci zahalené tváře

Na snímku A je fotka, která vstupuje do programu, na snímku B je detektorem označena tvář čtvercem a modrými body významné body tváře – oči, špičku nosu a koutky úst. Na snímku C je výsledek po úpravě snímku A, který vychází z označené oblasti na snímku B. Po zprovoznění a ověření funkčnosti tohoto detektoru byl tento program upraven tak, aby vytvořil snímek obličeje o velikosti 112 × 112 pixelů, aby mohl program správně fungovat. Takto upravený detektor pro detekci tváře byl zapracován do programu pro identifikaci osob.

4.3 Rozšíření programu pro identifikaci osob se zakrytou a odkrytou tváří

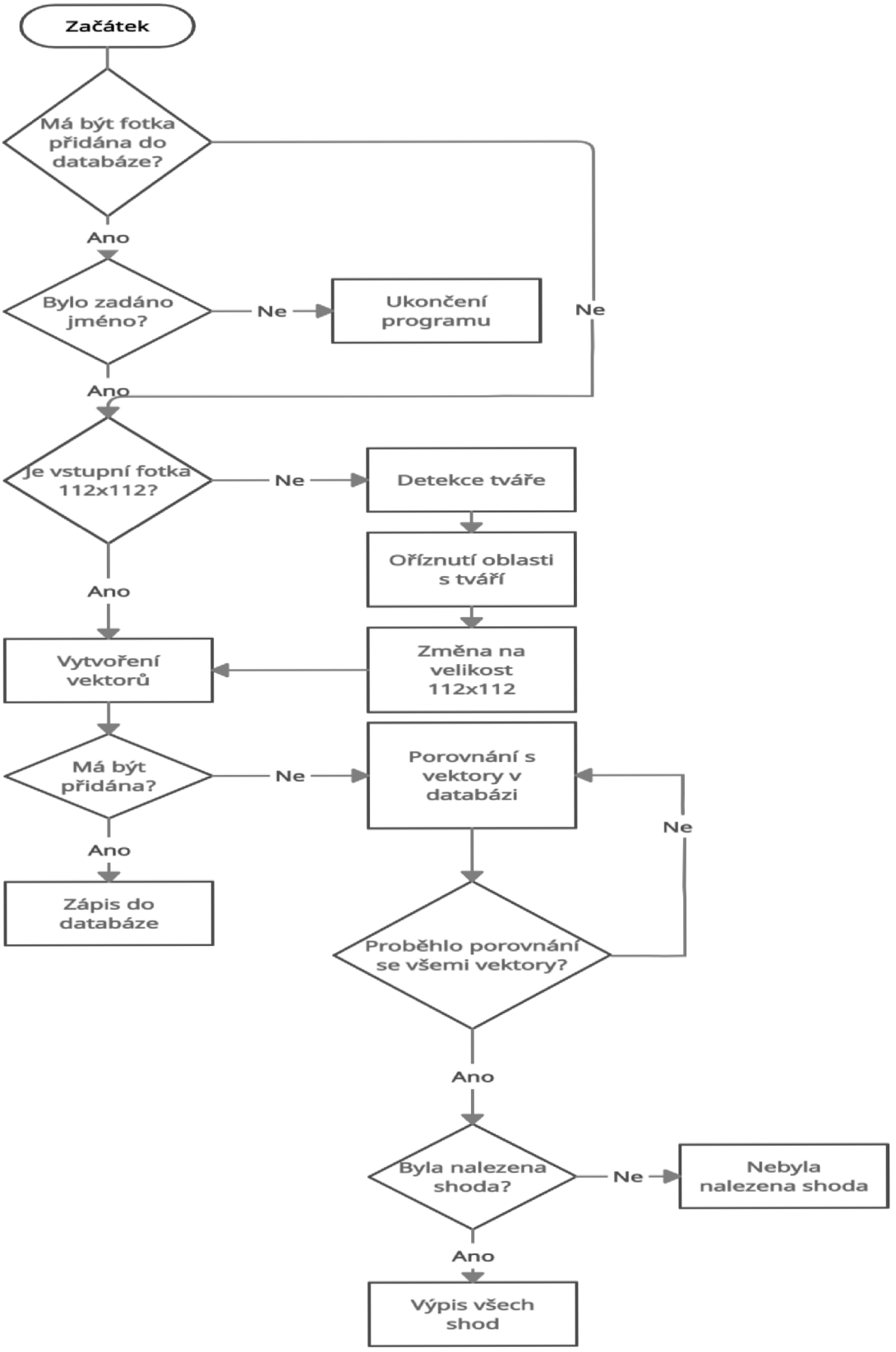
Stejně jako je to i u předchozích kroků i zde je použito 1000 osob a pro každou osobu 2 její fotografie. Tento program je již vylepšen a na vstup může být použita jakákoliv fotografie, z níž bude extrahován obličej a změněn na velikost 112 × 112 pixelů. Oproti ostatním verzím je zde přidán další parametr v databázi, celkově jsou zde tedy 4 parametry a to ID, jméno, vektor ve formě BLOB a fotografie ve formě BLOB. V parametru fotografie je uložena fotografie o velikost 112 × 112 pixelů, která odpovídá fotografii, jež byla použita při vstupu, proto je možné na konci zobrazit fotografii s nejlepší shodou, která nastala.

Aby bylo možné používat program pro identifikaci osob i mimo prostředí PyCharm, byl vytvořen modul, který lze spustit v terminálu. Program funguje ve 2 módech. První mód je porovnávání vektorů, tak aby došlo k identifikaci osoby, a druhý je možnost přidat osobu do databáze. Základem je cesta k obrázku, který chceme identifikovat. V programu je tento parametr označen –photo_path. Uživateli tedy stačí zadat

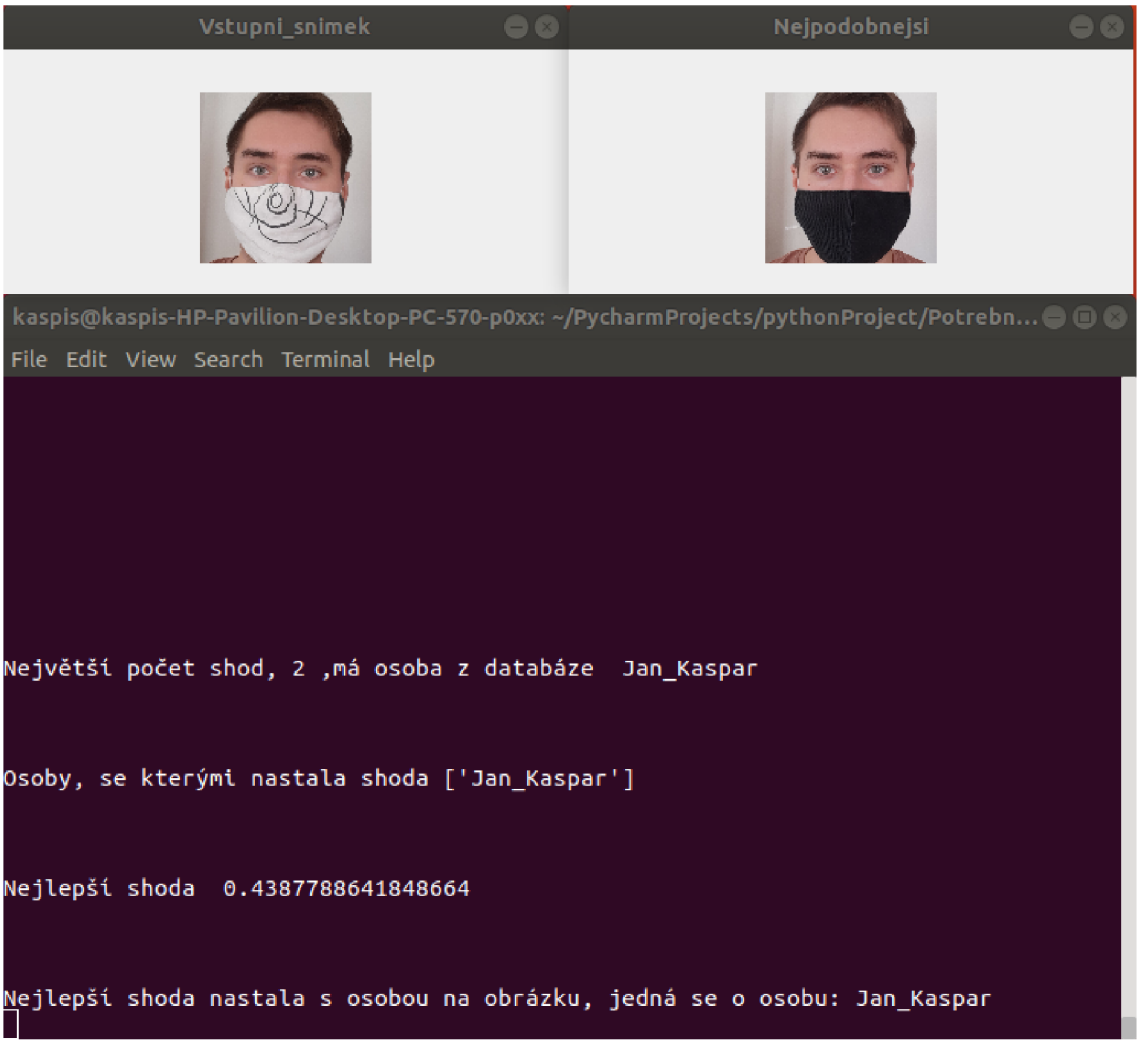
```
python ProgramProIdentifikaci.py –photo_path fotka.jpg
```

a bude provedena identifikace. Dalšími nepovinnými parametry jsou –modelType pro zvolení modelu pro extrakci vektorů. Bez uvedení tohoto parametru bude vždy použit model pro roušky, v případě použití parametru –modelType 2 bude použit model pro identifikaci bez roušek. Podle tohoto parametru bude také rozhodnuto, zda se použije databáze lidí s rouškou či bez. Dalším nepovinným parametrem je –distance, v základním nastavení je hodnota nastavena na 1. Uživatel si však může zvolit libovolnou vzdálenost, čím blíže 0, tím méně fotografií bude mít shodu, protože bude zpřísněna podmínka pro identifikaci, a čím více bude vzdálenost zvětšována nad 1, tím více bude přibývat falešně pozitivních shod. Dalším specifickým parametrem je –add, v základním stavu je 0. V případě, že chce uživatel přidat fotku do databáze, použije tento parametr –add 1, ale dále také musí zadat parametr –name, pod tímto parametrem bude osoba uložena v databázi. Zároveň si také může zvolit –modelType podle toho, jestli se jedná o osobu s rouškou nebo bez, aby byl vektor uložen do správné databáze.

Výstupem programu je výpis, s kolika vektory dané osoby si byla zadaná fotografie podobná, všechna jména, se kterými byla nalezena alespoň jedna shoda, nejmenší hodnotu euklidovské vzdálenosti, při které došlo ke shodě, zobrazení fotografie z databáze, se kterou nastala nejpřesnější shoda a zobrazení fotografie, která byla použita jako vstup pro porovnání, zda jsou si vstupní fotografie a výsledná opravdu podobné.



Obr. 4.2 Blokový diagram programu



Obr. 4.3 Výpis programu po provedení identifikace

ZÁVĚR

Cílem této bakalářské práce bylo seznámení se s principem identifikace osob v obraze na základě rozpoznávání tváří, vybrání vhodného algoritmu pro rozpoznávání tváří s částečně zahaleným obličejem, ověření funkčnosti algoritmu na vhodných databázích z reálného prostředí, vytvoření databáze osob s částečně zahalenou tváří a upravení tohoto algoritmu pro rozpoznávání osob s částečně zahalenou tváří.

První kapitola bakalářské práce se zabývá teoretickou částí, ve které jsou popsány základní principy a algoritmy pro rozpoznávání tváře.

Zbylé kapitoly jsou věnovány praktické části – Zprovoznění prostředí, Vytvoření modelu a Programu pro identifikaci.

Nejdříve byl natrénován model pro rozpoznání osob na databázi osob poskytnuté od tvůrců InsightFace, který dosáhl celkové přesnosti 84,48%. Tato přesnost je menší než bylo zamýšleno a tento fakt je promítnut na ROC křivku tohoto modelu, na které je zřejmá nízká úspěšnost při N:N identifikaci.

Poté byla vytvořena databáze osob se zakrytým obličejem pomocí programu, toto přidání funguje velmi dobře na tváře z předního pohledu, ale s postupným natočením tváře se přesnost přidání zhoršuje. Takto vytvořená databáze osob byla převedena do formátu MXNet nutného pro trénování a trénování bylo spuštěno. Tato část byla velmi obtížná a zabrala velmi mnoho času z důvodu špatného pochopení návodu, ze kterého bylo čerpáno. Výsledkem tohoto trénování je model pro rozpoznání osob se zahaleným obličejem, tento model dosáhl přesnosti 93,32%, což předčilo mé očekávání.

Jako poslední byl vytvořen program pro identifikaci osob s detektorem RetinaFaceAntiCovid pro možnost používání snímků nejen o velikosti 112 × 112 pixelů. Tento program byl vyzkoušen na snímcích, kde byly použity reálné snímky o velikosti 2472 × 3296 pixelů. Roušky, které byly použity pro tuto identifikaci, nebyly přidány pomocí programu, a i když snímky obsahují roušky, které nebyly použity pro natrénování modelu, tak došlo k bezproblémové identifikaci.

Zadání bylo, dle mého názoru, splněno. Jako možné vylepšení se nabízí úprava programu pro přidání roušek, tak aby nedocházelo ke špatnému přidání roušky.

REFERENCE

[1] DENG, Jiankang, et al. Arcface: Additive angular margin loss for deep face recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2019. p. 4690-4699.

[2] F. Schroff, D. Kalenichenko and J. Philbin, "FaceNet: A unified embedding for face recognition and clustering," 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, 2015, pp. 815-823, doi: 10.1109/CVPR.2015.7298682.

[3] GUO, Jia a Jiankang DENG. InsightFace: 2D and 3D Face Analysis Project [online]. [cit. 2020-12-02]. Dostupné z: https://github.com/deepinsight/insightface

[4] SCHROFF, Florian; KALENICHENKO, Dmitry; PHILBIN, James. Facenet: A unified embedding for face recognition and clustering. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015. p. 815-823.

[5] GEITGEY Adam. Face Recognition [online]. 2020 [cit. 2020-12-03]. Dostupné z: https://facerecognition.readthedocs.io/

[6] NumPy [online]. [cit. 2020-12-03]. Dostupné z: https://naucse.python.cz/lessons/intro/numpy/

[7] OpenCV [online]. [cit. 2020-12-03]. Dostupné z: https://opencv.org/

[8] CORPORATION, NVIDIA. CUDA [online]. [cit. 2020-12-03]. Dostupné z: https://developer.nvidia.com/cuda-zone

[9] CORPORATION, NVIDIA. NVIDIA cuDNN [online]. [cit. 2020-12-03]. Dostupné z: https://developer.nvidia.com/cudnn

[10] Baltrusaitis, Tadas & Robinson, Peter & Morency, Louis-Philippe. (2016). OpenFace: An open source facial behavior analysis toolkit. 1-10. 10.1109/WACV.2016.7477553.

[11] Facial recognition system [online]. [cit. 2020-12-08]. Dostupné z: https://en.wikipedia.org/wiki/Facial_recognition_system

[12] TensorFlow [online]. [cit. 2020-12-08]. Dostupné z: https://www.tensorflow.org/

[13] How to detect facial landmarks using DLIB and OpenCV [online]. [cit. 2020-12-08]. Dostupné z: http://datahacker.rs/009-how-to-detect-facial-landmarks-using-dlib-and-opencv/

[14] SANDBERG, David. Face Recognition using Tensorflow [online]. [cit. 2020-12-08]. Dostupné z: https://github.com/davidsandberg/facenet

[15] PyCharm [online]. [cit. 2020-12-08]. Dostupné z: https://www.jetbrains.com/pycharm/

[16] Dlib C++ Library [online]. [cit. 2020-12-08]. Dostupné z: http://dlib.net/

[17] B.V., ELSEVIER. Euclidean Distance [online]. [cit. 2020-12-08]. Dostupné z: https://www.sciencedirect.com/topics/mathematics/euclidean-distance

[18] Moschoglou, Stylianos et al. “AgeDB: The First Manually Collected, In-the-Wild Age Database.” 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW) (2017): 1997-2005.

[19] S. Sengupta, J.C. Cheng, C.D. Castillo, V.M. Patel, R. Chellappa, D.W. Jacobs, Frontal to Profile Face Verification in the Wild, IEEE Conference on Applications of Computer Vision, 2016.
[20] Face verification using large feature sets and one shot similarity – Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/Examples-of-some-images-from-the-LFW-dataset-with-variations-in-Top-rowleft-partial_fig1_261431622 [cit. 8 DEC, 2020]
[21] Wenyi Zhao, Rama Chellappa, P Jonathon Phillips, and Azriel Rosenfeld. Face recognition: A literature survey. ACM computing surveys (CSUR), 35(4):399–458, 2003.
[22] Anaconda [online]. [cit. 2020-12-08]. Dostupné z: https://www.anaconda.com/
[23] Face-mask 0.3.0 [online]. [cit. 2020-12-08]. Dostupné z: https://pypi.org/project/face-mask/
[24] Principal Component Analysis [online]. [cit. 2020-12-08]. Dostupné z: https://www.doc.ic.ac.uk/~hh4017/PCA
[25] TALGIN, Talgin. Preparing_data [online]. [cit. 2020-12-09]. Dostupné z: https://github.com/Talgin/preparing_data
[26] Huang, Gary & Mattar, Marwan & Berg, Tamara & Learned-Miller, Eric. (2008). Labeled Faces in the Wild: A Database forStudying Face Recognition in Unconstrained Environments. Tech. rep..
[27] AGARWAL, Rahul. A definitive guide for Setting up a Deep Learning Workstation with Ubuntu 18.04 [online]. [cit. 2020-12-09]. Dostupné z: https://towardsdatascience.com/a-definitive-guide-for-setting-up-a-deep-learning-workstation-with-ubuntu-18-04-5459d70e19c3
[28] Sqlite3: DB-API 2.0 interface for SQLite databases [online]. [cit. 2021-4-26]. Dostupné z: https://docs.python.org/3/library/sqlite3.html
[29] RetinaFace Anti Cov Face Detector [online]. 2020 [cit. 2021-4-26]. Dostupné z: https://github.com/deepinsight/insightface/tree/master/detection/RetinaFaceAntiCov
[30] Cython C-Extensions for Python [online]. [cit. 2021-4-27]. Dostupné z: https://cython.org/
[31] Update2 models: arcface_retinaface_mxnet2onnx [online]. [cit. 2021-5-13]. Dostupné z: https://github.com/zheshipinyinMc/arcface_retinaface_mxnet2onnx
[32] Lindovský, Michal Rozpoznání osob s pomocí snímků obličeje. Brno, 2019 Dostupné z: https://www.vutbr.cz/studenti/zav-prace/detail/115508 Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce Martin Rajnoha.

SEZNAM SYMBOLŮ A ZKRATEK

AgeDB In the Wild Age Database – věková databáze osob ve volné přírodě

API Application Programming Interface – rozhraní pro programování aplikací

BLOB Binary Large OBject – datový typ blíže nespecifikovaných dat v databázi

CFP_FP Celebrities in Frontal-Profile in the Wild – celebrity z předního pohledu a boku ve volné přírodě

CPU Central Procesor Unit – centrální procesorová jednotka

CUDA Compute Unified Device Architecture – jednotná výpočetní architektura

FP False Positive – falešně pozitivní

FN False Negative – falešně negativní

GB Gigabyte

GPU Graphics Processing Unit – grafický procesor

LFW Labeled Faces in the Wild – označené tváře ve volné přírodě

MB Megabyte

ONNX Open Neural Network Exchange – volně otevřený systém umělé inteligence

PCA Principal Compotent Analysis – analýza hlavních komponent

ROC Receiver Operating Characteristic – operační charakteristika přijímače

SQL Structured Query Language – standardizovaný strukturovaný dotazovací Jazyk

TN True Negative – pravdivě negativní

TP True Positive – pravdivě pozitivní

SEZNAM PŘÍLOH

Příloha A - Obsah přiloženého CD

Z důvodu velkého množství dat je obsah přiloženého CD možné stáhnout na odkaze:

https://www.dropbox.com/s/fu9mbr3mrlnentw/Potrebne_Programy.zip?dl=0

Stažený soubor obsahuje oba modely pro rozpoznávání tváří, model pro detekce obličeje s rouškou, obě vytvořené databáze osob, roušky, které byly použity pro natrénování modelu, programy pro identifikaci, přidání roušky a vytvoření databáze, soubory potřebné pro funkčnost modelu detekce obličeje s rouškou, návod popisující, jak ovládat přiložené programy, složku s 10 testovacími fotografiemi a soubor requirements.txt, který obsahuje verze používaných frameworků. Příkazem

```
pip install requirements.txt
```

dojde k nainstalování těchto verzí.

Obsah přiloženého CD

- Zdrojové kódy
  - ProgramProIdentifikaci.py
  - ProgramProPridaniRousek.py
  - ProgramProVytvoreniDatabaze.py
  - Facechange.py
  - Retinaface_cov.py
- Modely pro rozpoznávání osob
  - bez_rousek.onnx
  - hodne_rousek.onnx
- Model pro detekci obličeje s rouškou
- Použité roušky
  - Rouška-1.png
  - Rouska-2.png
  - .
  - .
  - Rouska-10.png
- Soubory nutné pro funkčnost modelu detekce obličeje s rouškou
  - Obsah složky rcnn
- Databáze osob
  - Databaze_bezrousek.db
  - Databaze_hodnerousek.db
- Testovací fotografie
- Soubor requirements.txt
- Návod.pdf