VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

ROZPOZNÁVÁNÍ TVÁŘÍ

FACE RECOGNITION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. TOMÁŠ BENDA**
AUTHOR

VEDOUCÍ PRÁCE — **Doc. RNDr. PAVEL SMRŽ, Ph.D.**
SUPERVISOR

BRNO 2017

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2016/2017

Zadání diplomové práce

Řešitel: **Benda Tomáš, Bc.**
Obor: Informační systémy
Téma: **Rozpoznávání tváří**
Face Recognition
Kategorie: Umělá inteligence

Pokyny:

1. Seznamte se se způsoby vyhledávání tváří a s dostupnými implementacemi, např. http://docs.opencv.org/trunk/modules/contrib/doc/facerec/
2. Shromážděte datovou sadu pro průběžné testování systému.
3. Na základě získaných poznatků navrhněte a implementujte systém, který dokáže identifikovat účastníky různých akcí, např. vědeckých konferencí.
4. Vyhodnoťte výsledky systému na reprezentativním vzorku dat.
5. Vytvořte stručný plakát prezentující práci, její cíle a výsledky.

Literatura:

- dle doporučení vedoucího

Při obhajobě semestrální části projektu je požadováno:

- funkční prototyp řešení

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci dřívějších projektů (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Smrž Pavel, doc. RNDr., Ph.D.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 24. května 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 66 Brno, Božetěchova 2

doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Abstrakt

Tato práce se zabývá rozpoznáváním lidí ve videozáznamu, respektive z kamery. K rozeznávání obličejů byla použita konvoluční neuronová síť, díky které získáme několika dimensionální vektor rysů, pomocí kterého je možno zjistit totožnost osoby. Na systém jsou kladeny nároky, aby byl schopen pracovat v reálném čase a mohl být použit například pro rozpoznávání osob na různých konferencích, či jako součást bezpečnostního systému. Samotný systém je napsán v jazyce Python. Součástí této práce byla vytvořena datová sada ve formě videí s osobami.

Abstract

This thesis deals with human recognition on a videorecording. Convolution neural network was used for face recognition, from which we will get multidimensional vector, which will allow to determine person's identity. There are demands imposed on the system, for it to be able to work in real time and could be used for example for person recognition at various conferences, or as a part of security system. Whole system is written in Python language. Part of this thesis is dataset in form of videorecords with persons.

Klíčová slova

convolutional neural network, face recognition, face detection, classification

Keywords

convolutional neural network, face recognition, face detection, classification

Citace

BENDA, Tomáš. *Rozpoznávání tváří.* Brno, 2017. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Doc. RNDr. Pavel Smrž, Ph.D.

Rozpoznávání tváří

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením doc. Smrže. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Tomáš Benda
24. května 2017

Poděkování

Na tomto místě bych rád poděkoval vedoucímu diplomové práce doc. RNDr. Pavlu Smržovi, Ph.D, za cenné připomínky a odborné rady, kterými přispěl k vypracování této práce. Dále bych chtěl poděkovat panu Ing. Michalu Hradišovi, Ph.D za jeho cenné informace a rady ohledně konvolučních neuronových sítí.

Obsah

Kapitola 1

Úvod

Jedinečnou a hlavně na první pohled viditelnou částí každého člověka je právě jeho tvář. Stále častěji se setkáváme s implementacemi, které dovolují rozpoznávat lidi mezi sebou a to právě na základě jejich obličeje. Díky rychlému vývoji zejména hardwaru je možné tyto systémy implementovat i na nespecializovaných zařízeních, jako je například notebook spolu s webovou kamerou.

Nejrozšířenější oblast použití těchto systému je za použití bezpečnostních kamer, kde se často dožaduje požadavku na rozpoznávání v reálném čase. Úloha rozpoznání obličeje je stále otevřenou oblastní, jelikož se ještě nikomu nepodařilo vyvinou systém, který má úspěšnost v rozpoznávání 100%, ačkoliv existují systémy(implementace konvoluční neuronové sítě FaceNet od společnosti Google dosahuje úspěšnosti přes 99%), které se této hranice dotýkají. Běžný člověk je schopen rozeznat osobu s velkou přesností a to hlavně díky tomu, že nerozeznává osobu jen podle obličeje, ale všímá si více rysů, které člověka popisují jako celek.

Ve své práci využívám k rozeznávání obličejů konvoluční neuronové sítě, kde jedna je použita na rozpoznávání obličejů, které jsou přímo natočeny do kamery a druhá síť je optimalizovaná pro rozeznávání obličejů, jež jsou natočeny z profilu. Aby bylo možné obličej zpracovat a rozeznat na základě jeho rysů, je nutné jej ve videozáznamu nejprve nalézt. K nalezení a vyříznutí obličeje potřebujeme několik kroků: nalezení pohybujících se předmětu, nalezení obličeje v místech pohybujících se předmětů a nakonec samotné rozpoznání obličeje pomocí konvoluční neuronové sítě.

V první části práce, která bude ryze teoretická, se budu zabývat algoritmy používaných při sekreci pohybujícího se popředí od statického pozadí videa, algoritmy zabývajícími se hledáním obličejů a nakonec samotným rozpoznáváním obličejů k čemuž jsou použité konvoluční neuronové sítě.

Druhá část práce se zabývá implementací algoritmů a postupů, které jsou rozebrány v teoretické části práce.

Poslední část práce budou tvořit výsledky vytvořeného systému a to ve formě porovnání úspěšnosti v rozeznávání obličejů.

Kapitola 2

Vyhledávání obličejů ve videu

Abychom mohli spolehlivě nalézt a validovat obličeje na přiloženém video-záznamu, respektive datovém toku z kamery, musíme zvláště u video-souborů s vysokým rozlišením(například HD a vyšším) nějakým způsobem zmenšit hledanou plochu a to zejména kvůli požadavku na systém pracující v reálném čase. Jedna z možností je jednoduše obrázek zmenšit a tím docílit mnohem rychlejší odezvy systému. Tuto metodu však nelze vždy použít, zejména v případech kdy máme plochu záznamu sice velkou, ale obličeje nacházející se na této videosekvenci zabírají pouze malou část a to v řádech několika desítek pixelů. Pravděpodobně bychom redukcí celé plochy ve většině případů docílili toho, že oblast kde se obličej nachází bude tak malá, že žádný algoritmus tento obličej nebude schopen rozeznat.

Vzhledem k výše uvedeným faktům je vhodnější prohledávanou plochu upravit pomocí separace pohybujícího se popředí(kde se dá předpokládat, že na pohybující se ploše budou z velké části lidé) od statického, nepohybujícího se pozadí.

2.1 Separace popředí

Pro extrakci pohybujícího se popředí je nejprve nutné vymodelovat statické pozadí ze kterého se vytvoří maska, která v pozdější fázi bude sloužit jako rozdíl ke každému dalšímu obrázku, který bude ve videosekvenci následovat. Podle masky poté vyseparujeme pohybující se předměty, respektive osoby na videu a to tak, že každý následující snímek se porovnává s vytvořenou maskou a výsledný rozdíl je poté brán jako pohybující se objekt. Je vhodné masku po určitém čase aktualizovat a to z důvodů malých, či větších změn napříč časem, kde se ku příkladu může na video objevit nějaký předmět(například vozík), který bude na místě setrvávat delší dobu.

2.1.1 Směs Gaussovských modelů

Jedná se o rekurzivní modelovací techniku pro modelování prostředí. Princip spočívá v popsání každého pixelu k gaussových rozložení(k je malé číslo od 3 do 5). Pravděpodobnost, že určitý pixel má hodnotu X_N v čase N může být vyjádřena jako:

$$p(X_N) = \sum_{j=1}^{K} W_j \eta(X_N; \theta_j) \tag{2.1}$$

Kde W_k je váha k-tého gaussiánu, $\eta(X; \theta_k)$ je normální distribuce k-té složky reprezentované jako:

$$\eta(X;\theta_k) = \eta(X;\mu_k,\sum_k) = \frac{1}{(2\pi)^{\frac{D}{2}}|\sum_k|^{\frac{1}{2}}} e^{-\frac{1}{2}(x-\mu_k)^T(\sum_k)^{-1}(x-\mu_k)} \tag{2.2}$$

Kde μ_k je průměr a $\sum_k = \sigma_k^2 I$ je kovariance k-té složky. k distribuce jsou seřazeny v závislosti na fitness hodnotě $w_k/\sigma/k$ a prvních B distribucí je použito jako model pozadí, kde B je vyjádřeno jako:

$$B = arg_b min(\sum_{j=1}^{b} w_j > T) \tag{2.3}$$

Práh T je minimální zlomek modelu pozadí. První gaussián, který se shoduje s testovací hodnotou bude upraven podle těchto rovnic:

$$\hat{w}_k^{N+1} = (1-\alpha)\hat{w}_k^N + \alpha\hat{p}(\Omega_k|X_{N+1}) \tag{2.4}$$

$$\hat{\mu}_k^{N+1} = (1-\alpha)\hat{\mu}_k^N + \rho X_{N+1} \tag{2.5}$$

$$\hat{\sum}_k^{N+1} = (1-\alpha)\hat{\sum}_k^N + \rho(X_{N+1}-\hat{\mu}_k^{N+1})(X_{N+1}-\hat{\mu}_k^{N+1})^T \tag{2.6}$$

$$\rho = \alpha\eta(X_{N+1};\hat{\mu}_k^N,\hat{\sum}_k^N) \tag{2.7}$$

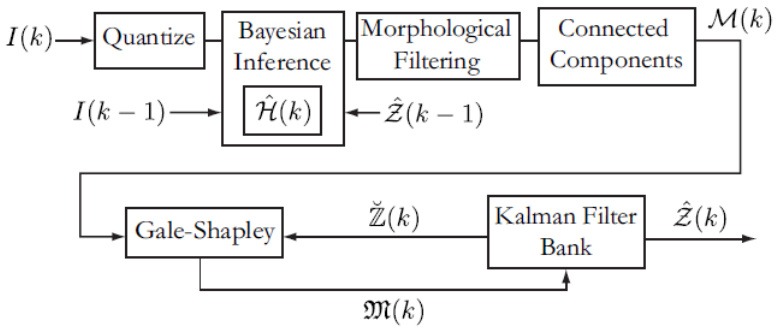
$$\hat{p}(\omega_k|X_{N+1}) = \{ \begin{matrix} 1 & ;if\ \omega_k\ je\ prvn\ shodn\ gaussin \\ 0 & ;jinak \end{matrix} \} \tag{2.8}$$

Kde ω_k je k-tý gaussián, $1/\alpha$ definuje časovou konstantu, která znamená změnu. Pokud žádná z K rozdělení neshoduje s hodnotou pixelu, poté je vytvořen nový gaussián, jelikož je počet gayssiánů K, tak se po vytvoření nového musí jeden odstranit (ten s nejnižší vahou). Rozdílné gaussiány reprezentují rozdílné barvy. Váhový parametr daného rozložení reprezentuje čas, po který tyto barvy setrvávají na scéně. Do pozadí jsou poté zařazeny ty gaussiány, které mají dostatečně nízký rozptyl a zároveň jejich hodnoty pixelů pod jeho křivkou se v obraze objevují s velkou četností. Hodnoty nad křivkou poté můžou být považovány za popředí[14][20].

2.1.2 Geometric multigrid

Algoritmus kombinuje statický odhad modelu pozadí, Bayesovkskou segmentaci a přibližné řešení problému sledování více cílů pomocí banky Kalmanových filtrů a Gale-Shapley přizpůsobení. Činnost algoritmu znázorňuje obrázek 2.1

Vstupní obrázek je kvantizovaný v barevném prostoru a je porovnán vůči statistickému modelu pozadí ke generování předpovídaného dalšího obrázku. Tento obrázek je vyfiltrován pomocí morfologických operací a následně je segmentován do množiny ohraničujících obdélníků pomocí algoritmu spojující komponenty. banka kalmanova filtru udržuje množinu trackovaných pohybujících se předmětů a predikuje jejich následující pozice v dalším snímku. Gale-Shapleyův algoritmus páruje obdélníky ze současného a predikovaného obrázku, tyto páry jsou poté pužity pro aktualizaci banky kalmanova filtru. Výsledkem je kolekce pixelů identifikovaných jako popředí[11].



Obrázek 2.1: Blokový diagram činnosti algoritmu GMG. Převzato z [11].

2.2 Hledání obličejů

Detekce obličeje je v rámci počítačového vidění za poslední léta na vzestupu. Jako největší problém při detekci obličeje se jeví jeho rozmanitost, co se týče tvarů, barvy kůže, ale největším faktorem při nesprávné detekci obličeje je jeho částečné zakrytí a to v podobě vousů, různých roušek zakrývajících ústa, či jako z nejrozšířenějších se jeví brýle, které dokážou úspěšně zakrýt celé oči a oblast kolem nich. Algoritmy pro detekci obličejů se dají rozdělit do čtyř hlavních skupin:

- metody zakládající se na bázi znalostí
- metody zakládající se na rysech
- metody zakládající se na šabloně
- metody zakládající se na statistice

Metody založené na bázi znalostí vyhodnotí oblast, jako oblast ve které se nachází obličej právě tehdy, pokud splní určitá kritéria. Tyto kritéria jsou vytvořena z báze znalostí o lidském obličeji(v zásadě se jedná o rysy lidského obličeje, jako je pár očí, ústa a nos).

Metody zakládající se na příznacích, mohou detekovat obličej ne jen podle existujících obličejových příznacích, ale také podle jejich geometricky vzájemné pozici, které jsou neměnné, tedy zůstávají stejné z jakéhokoliv úhlu pohledu na daný obličej. Jedná se o opak k metodám využívající báze znalostí. Běžně se používají hranové detektory k extrakci obličejových rysů, ze kterých se poté vytvoří statistický model popisující jejich vztahy mezi sebou.

Šablonou založené metody mohou být rozděleny do dvou podkategorií:

- předurčené šablony
- deformovatelné šablony

Pomocí předurčené šablony se počítá přiřazená hodnota prohledávané oblasti. Pokud je tato hodnota v souladu, je oblast označena jako obličej. Druhá metoda nejprve vytvoří parametr šablony, poté podle detekční oblasti upravuje parametry, dokud nekonverguje, za cílem k dosažení lokace pozice obličeje.

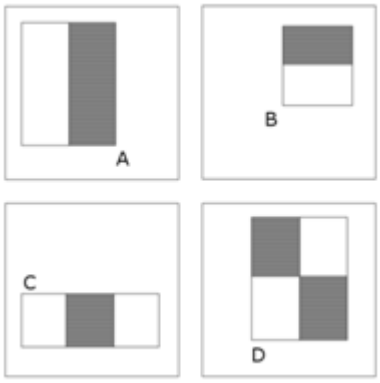
Posledními metodami zvanými jako statisticky založenými, jsou metody, které berou oblast ve které se nachází obličej jako třídní model, který používá mnoho obličeje nacházejících se a nenacházejících se trénovacích vzorků, zkonstruovaný klasifikátor k rozlišení

všech možných oblastí v obrazové ploše, který vyhodnotí, zda se tam obličej nachází, či nikoliv[24].

2.2.1 Violův-Jonesův algoritmus

Jedná se o velice rychlý detektor, který je velice vhodný k využití v aplikacích pracujících v reálném čase. Tento detektor je založen na strojovém učení. Základním principem je kaskádová funkce trénovaná z mnoha pozitivních a negativních obrázků, což je poté využito k detekování objektů v jiných obrázcích. Vzhledem k rychlostí této metody a jejímu implementování v knihovně OpenCV. Výhodou tohoto algoritmu může být i to, že je možné nastavit několik parametrů a to zejména hloubku rozkladové pyramidy a krok po kterém se posouvá okno. Tyto parametry mají velice významný vliv na výpočetní náročnost a přesnost detekovaných bodů. Pokud je prohledávaná plocha o značné velikosti, je vhodnější parametry nastavit tak, že se nejprve naleznou kandidáti na hledané objekty a poté se použije další iterace s parametry nastavenými tak, aby detekce byla přesnější (zároveň ale pomalejší) a nalezené oblasti se prohledávají znovu.

Detektor pracuje tak, že klasifikuje obrázek podle hodnot jednoduchých haarových příznaků.[22]



Obrázek 2.2: Haarovy příznaky. Převzato z [2].

Na obrázku 2.2 jsou zobrazeny jednoduché haarovy příznaky. Sumy intenzity pixelů které leží v bílých obdélnících jsou odečteny ze sumy intenzity pixelů v šedých obdélnících. Dvou-obdélníkové rysy jsou zobrazeny pod (A) a (B). (C) zobrazuje troj-obdélníkový rys a (D) znázorňuje čtyř-obdélníkový rysy.

Algoritmus pro detekci využívá tři druhy rysů. Hodnota dvou (A) respektive (B) rysů je rozdíl mezi sumou intenzity pixelů v dvou-obdélníkových regionech. Regiony mají stejnou velikost a tvar a jsou horizontálně, nebo vertikálně orientovány. Rysy (C) počítají sumu ve dvou venkovních obdéldnících odečtených od sumy v prostředním obdélníku. Nakonec rysy (D) počítají rozdíl mezi diagonálními páry obdélníků. Základní rozlišení detektoru je 24x24 pixelů. Jelikož je většina rysů irelevantních(z počátku jich je obrovské množství), je tento počet zredukován pomocí algoritmu *AdaBoost*, který vybere jen ideální rysy.

Pro rychlejší výpočet haarových rysů se využívá převedení obrazu na integrální obraz. Integrální obraz uchovává na souřadnicích x a y sumu intenzity pixelů nad a nalevo od x, y, tedy podle vztahu:

$$ii(x, y) = \sum_{x1<x, y1<y} i(x1, y1) \tag{2.9}$$

kde ii(x,y) je souřadnice v integrálním obraze a i(x1,y1) je souřadnice ve zdrojovém obraze.

Díky převodu do integrálního obrazu lze jednoduše spočítat intenzitu určitého obdélníku jen pomocí čtyř rohových bodů, namísto počítáním všech pixelů obsažených v uvedeném obdélníku. Pro výpočet rozdílu dvou obdélníků potřebujeme tedy pouze osm bodů, tedy pro každý další obdélník potřebujeme jeho čtyři rohové body.

Pro rychlé zpracovávání výstupního obrazu se používá kaskáda klasifikátorů. Z klasifikátorů se vytvoří degenerovaný rozhodovací strom, který se nazývá kaskáda. Pozitivní výsledek prvního klasifikátoru spustí ohodnocení následujícího klasifikátoru. Negativní výsledek klasifikátoru vede o odmítnutí celého podokna. Klasifikátory v kaskádě jsou sestaveny pomocí *AdaBoost* a následně se upraví prahy kvůli minimalizaci negativních výsledků[22].

2.2.2 HOG detektor

Histogram of Oriented Gradients je metoda zakládájící se na zobrazování objektu pomocí jednoduchých příznaků. Základní ideou je, že vzhled a tvar lokálního objektu může být dobře popsán pomocí lokální intenzity gradientu, nebo směru pomocí hrany. Nejprve je nutné spočítat horizontální a vertikální gradienty, pomocí vhodného kernelu (například [-1, 0, 1]). Směr gradientu v bodě x a y můžeme spočítat podle následujícího vzorce:

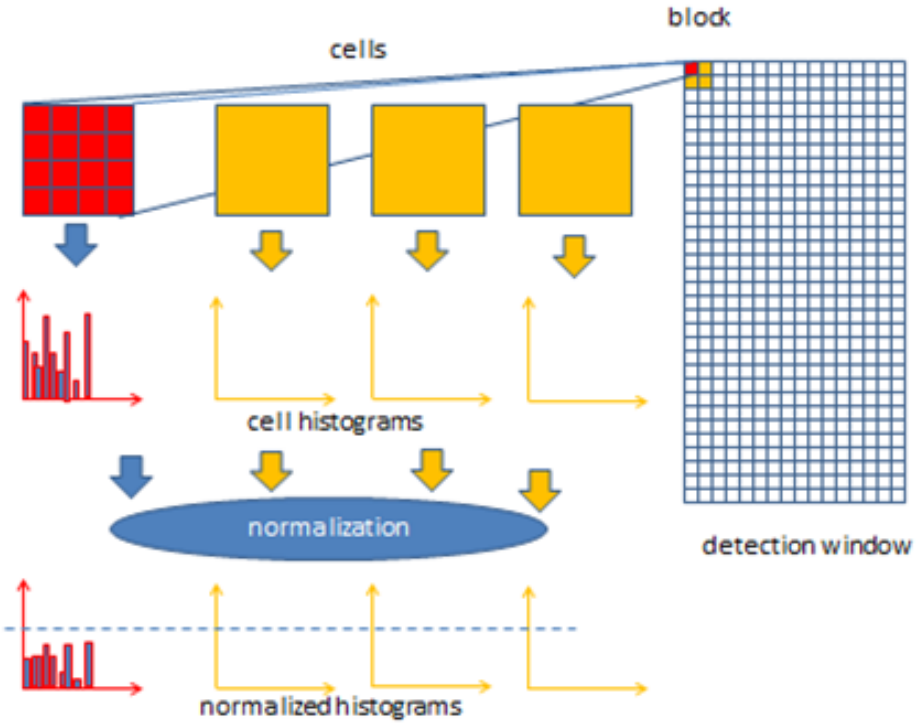
$$tan^{-}1\frac{gy}{gx} = 0 \tag{2.10}$$

[8]

Výpočet gradientu spočívá v nalezení změn intenzity pixelů v okolí daného pixelu. Gradient se počítá pro každou barevnou složku zvlášť, tedy u černobílých obrázků se získá pouze jedna hodnota u barevných obrázků (například RGB) se získají hodnoty 3. Mějme obraz popsán funkcí $f(x, y)$, kde x a y jsou souřadnice v obraze. Poté můžeme gradient ∇f vyjádřit jako:

$$\nabla f = \frac{gx}{gy} = \frac{\frac{\partial f}{\partial x}}{\frac{\partial f}{\partial y}} \tag{2.11}$$

Vy výpočtu gradientu v každém pixelu následuje rozdělení obrázku do malých prostorových buněk, jak lze vidět na obrázku2.3.

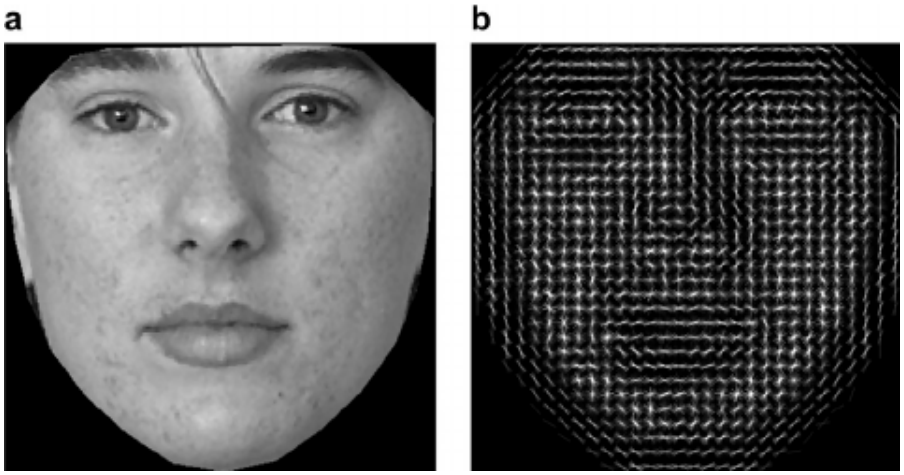


Obrázek 2.3: Ilustrace výpočtu gradientu[1].

Pro každou buňku je následně akumulován lokální jednorozměrný histogram gradientu směru, nebo hranové orientace skrze pixely obsažené v buňce. pro lepší neměnnost z hlediska osvětlení a stínů je vhodné použít kontrastní normalizaci. K tomu se používá akumulace lokální energie histogramu do větších bloků a poté se výsledek použije k normalizace všech buněk v bloku. Tyto bloky se nazývají *Histogram of Oriented Gradient* (HOG) popisovače. Doporučené parametry pro HOG detektor jsou: velikost detekčního okna 64x128 pixelů, velikost bloku 2x2 buňky a velikost buňky 8x8 pixelů. Změna těchto hodnot může vést ke zhoršení přesnosti, respektive snížení výkonnosti[8][16].

Výsledný HOG je poté znázorněn na obrázku 2.4.

[1]Převzato z: https://software.intel.com/en-us/node/529070



Obrázek 2.4: Ilustrace výsledného HOG[2].

Obrázek 2.5: Řetěz extrakce rysů a detektoru objektů[3].

Pokrytí detekovaného okna pomocí mřížky HOG popisovačů a za použití kombinace vektoru rysů v konvenční SVM založeném klasifikátoru nám dává schéma uvedeném na obrázku 2.5 [8].

2.2.3 K-means

Jedná se o metodu shlukové analýzy která si klade za cíl rozdělit v nejčastěji euklidovském prostoru N objektů do K shluků, kde každé pozorování náleží právě shluku s nejbližším průměrem a očekává se, že počet shluků je předem známý. Každý shluk je popsán centroidem. Objekty se poté zařazují do toho shluku, k jehož centroidu je objekt nejblíže. Problém je výpočetní náročnost, nicméně existují efektivní heuristické algoritmy, které jsou běžně činné a které rychle konvergují k lokálnímu optimu.

Vstupem algoritmu je tedy množina dat $x_1, ..., x_i$ a číslo k jež udává počet tříd $u_1, ..., u_k$. Nejprve se centroid každé třídy inicializuje náhodně, popřípadě pomocí nějakého heuristického algoritmu viz například [17]. Poté se opakují následující dva kroky:

1. Všechna data $x_1, ..., x_i$ se přiřadí do tříd pomocí minimální euklidovské vzdálenosti vůči centroidu z určité třídy.

2. Přepočtení hodnot centroidů všech tříd a to podle vztahu : $u_j = \frac{1}{l_j} \sum_{i=1}^{L} x_i$, kde L je počet objektů, respektive vektorů v příslušné třídě.

Postup se opakuje tak dlouho, dokud bude nastávat změna v podobě náležitosti objektů k určitým třídám[11].

[2]Převzato z: https://www.researchgate.net/figure/284243500_fig5_Fig-5-Histogram-of-Oriented-Gradients-HOG-features-in-4-4-cells

[3]Převzato z: http://soc.fudan.edu.cn/vip/projects/gradproj/wiki/Histograms_of_Oriented_Gradients_for_Human_De

2.3 Sledování objektů

Po nalezení objektu je vhodné tento objekt sledovat, abychom mohli říci, na jakých pozících se nacházel objekt A ve snímcích jdoucích od snímku, na kterém byl tento objekt nalezen. Pokud budeme chtít objekt nějakým způsobem validovat na každém snímku, je vhodné abychom věděli, že se jedná o objekt A a nikoliv o žádný jiný. Poté můžeme s validacemi dále pracovat například tak, že vytvoříme množinu dvojic, do které budeme postupně ukládat výsledek validace objektu pro každý snímek. Spolu s počtem snímků na kterých byl objekt validován stejně. Tímto způsobem můžeme přesněji říci o jaký objekt se jedná a na jakých snímcích spolu s pozicí byl tento objekt rozpoznán, respektive na kterých pozicích se objekt v historii nacházel. Pokud by to bylo potřeba, je pomocí sledování objektů možné vytvořit statistiku, kde pomocí průchodů můžeme zjistit jakým směrem se sledované objekty nejčastěji vydávají.

2.3.1 Sledování na základě korelačních filtrů

Tento algoritmus je implementován v knihovně *dlib* a zakládá si na tom, že je oproti ostatním algoritmum, které se používají pro sledování objektu, velice rychlý a to s velmi přesnými výsledky. Základní funkcí tohoto algoritmu je naučení diskriminačních korelačních filtrů založeného na reprezentaci stupnic pyramidy. Algoritmus sledování učí diskriminační korelační filtr použitý pro lokalizaci a sledovaného cíle v novém snímku.

Metoda používá snímky cílového objektu v odstínech šedi, jako posloupnost $f1, ..., ft$ trénovacích snímků, které jsou přiřazeny k žádaným výstupům korelačního filtru $g1, ..., gt$. Optimální korelační filtr h_t v čase t je získán minimalizací sumy čtverečních chyb:

$$\varepsilon = \sum_{j=1}^{t} \|h_t \star f_j - g_j\|^2 = \frac{1}{MN} \sum_{j=1}^{t} \|\overline{H_t} F_j - G_j\|^2 \tag{2.12}$$

funkce f_j, g_j a h_t jsou velikosti MxN. Symbol $\star$ znamená kruhovou korelaci. Velká písmena značí diskrétní Furierovu transformaci korespondujících funkcí(tedy H_t, F_j,G_j). Symbol $\overline{H_t}$ reprezentuje komplexně sdružené číslo a $\overline{H_t} F_j$ značí součin v bodech. Po úpravě rovnice 2.12 výše získáme:

$$H_t = \frac{\sum_{j=1}^{t} \overline{G_j} F_j}{\sum_{j=1}^{t} \overline{F_j} F_j} \tag{2.13}$$

Žádaný výstup korelace g_j je zkonstruován jako Gaussova funkce s vrcholem ve středu obrázku f_j. V praxi čitatel A_t a jmenovatel B_t v H_t z rovnice 2.13 jsou aktualizovány separátně s příchodem nového f_t. Po předložení obrázku velikosti M x N v novém snímku jsou korelace ohodnoceny jako $y = \Psi^{-}1\overline{H_t}Z$, kde $\Psi^{-}1$ značí inverzní operátor diskrétní Furierovy transformace. Nová lokace sledovaného objektu je poté odhadnuta podle skóre y. Trénovací a detekující kroky jsou jsou vykonány efektivně pomocí rychlé Furierovy transformace.

Diskriminační korelační filtry pro multidimenzionální příznaky jsou rozšířením principu popsaného výše a to pro použití v aplikacích, kde je třeba sledovat objekty byly diskriminační korelační filtry rozšířeny o multidimenzionální příznaky. Úprava spočívá v konkatenaci HOG příznaků s příznaky intenzity obrázku. Uvažujme d-dimenzionální příznakovou mapu reprezentující obrázek. Nechť je f obdélníkový výřez cíle extrahovaného v příznakové mapě. Pro každou dimenzi $l\varepsilon 1, ..., d$ je označeno jako f^l. Cílem je získání filtru h, skládajícího se

z h^l pro každou dimenzi, toho je dosaženo pro jeden případ pomocí:

$$\varepsilon = \| \sum_{l=1}^{d} h^l \star f^l - g \|^2 + \lambda \sum_{l=1}^{d} \|h^l\|^2 \tag{2.14}$$

Kde g je požadovaný korelační výstup asociovaný s trénovacím vzorkem f. Parametr $\lambda \geq 0$ reguluje dopad regulačního termu. Regulační parametr zmírňuje dopad nulových komponent ve spektru f, které by vedly k dělení nulou. Optimální filtr může být získán minimalizováním skrze všechny trénovací snímky, což vyžaduje řešení dxd lineárního sytému rovnic na pixel. K získání robustní aproximace aktualizujeme čitatel A_t^l a jmenovatel B_t korelačního filtru H_t^l jako:

$$A_t^l = (t-\eta)A_(t-1)^l + \eta \overline{G_t} F_t^l B_t = (t-\eta)B_(t-1) + \eta \sum_{k=1}^{d} \overline{F_t^k} F_t^k \tag{2.15}$$

Kde η je učící koeficient. Korelační skóre obdélníkové oblasti z příznakové mapy je spočteno pomocí 2.16. Nový cílový stav je poté získán pomocí maximalizace skóre y.

$$y = \Psi^(-1)\{\frac{\sum_{l=1}^{d}}{B+\lambda}\} \tag{2.16}$$

[9]

2.4 Rozpoznávání obličeje

Existuje mnoho přístupů jak na videozáznamech nalézt a rozeznat obličeje. V posledních letech se však stále více vyjímá rozpoznávání zejména obličejů pomocí konvolučních neuronových sítí a jejich kombinacemi s dalšími přístupy zpravidla kvůli nutnosti rozeznání problematických částí. Těmito částmi zpravidla rozumíme části takové, kde jsou obličeje zakryty, či jsou špatně rozeznatelné díky vlivům osvětlení a podobně. V této části bude popsána problematika rozpoznávání obličeje a to za použití konvoluční neuronové sítě, spolu se základy neuronových sítí a samotného vzniku neuronu.

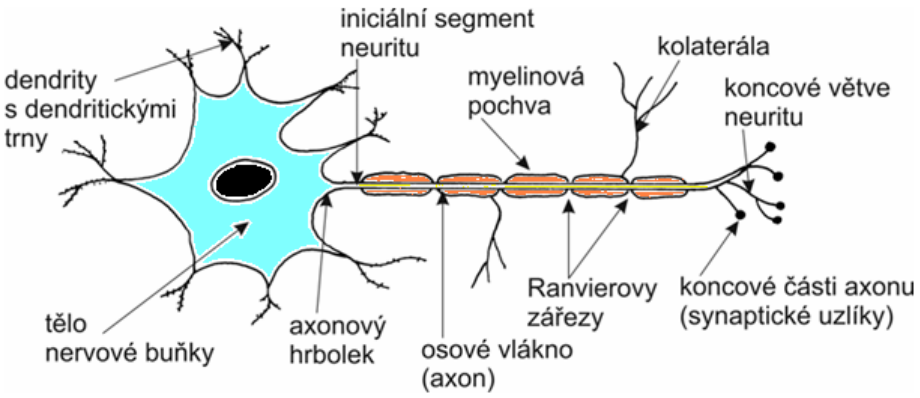
2.4.1 Neuronové sítě

V této kapitole popíší základy fungování neuronových sítí, včetně popisu umělého neuronu, jenž se inspiruje funkcí živého neuronu. Nástavbou dopředných neuronových sítí jsou právě konvoluční neuronové sítě.

Perceptron

Jelikož je umělý neuron inspirován neuronem biologickým, uvedu zde nejprve přibližný popis funkce biologického neuronu.

Neuron je základní stavební a funkční jednotkou nervové soustavy a je obsažen v mozcích všech živočichů a to ve značném počtu. Neuron zajišťuje vedení vzruchů. Na povrchu neuronu je vzrušivá membrána, prostřednictvím které jsou elektrické signály v synapsích (spojení dvou neuronů nebo smyslové buňky a neuronu) předávány dál[23].

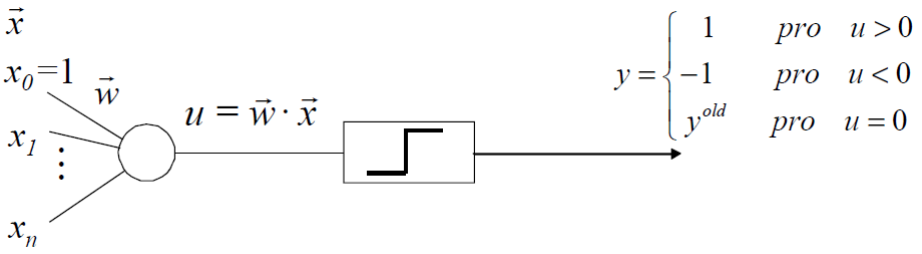


Obrázek 2.6: Biologický neuron[4].

Složení neuronu podle obrázku 2.6 je následující:

- Tělo neuronu
  - Výběžky
    - Dendrity - velký počet, bývají kratší, bohatě se větví, vedou vzruchy do těla neuronu.
    - Neurit (axon) - jeden dlouhý výběžek, vede vzruch z těla neuronu.
  - Ranvierovy zářezy – umožňují skoky akčních potenciálů a jejich vedení.
  - Vnější pochva Schwannova a vnitřní pochva myelinová – izolace neuritu.

Umělým modelem je poté perceptron, který je zobecněním McCullochova a Pittsova modelu umělého neuronu.



Obrázek 2.7: Model perceptronu. Převzato z[10]

Jako vstup perceptronu slouží vektor x, neboli perceptron má n vstupů, kde n je velikost vektoru x. Hodnoty vstupů x mohou být binární(0 nebo 1), ale také bipolární(hodnoty -1, 0 a 1). Vektor w je takzvaný vektor vah. Koeficient u je koeficientem učení a výstup každého perceptronu je značený jako y a jeho hodnota závisí na výsledku aktivační funkce patrné z obrázku 2.7. Váhové hodnoty jsou poté adaptovány podle adaptačního pravidla perceptronu tak, aby diference mezi skutečným a požadovaným výstupem byla co nejmenší.[23][10]

[4]Převzato z: https://eluc.kr-olomoucky.cz/verejne/lekce/89

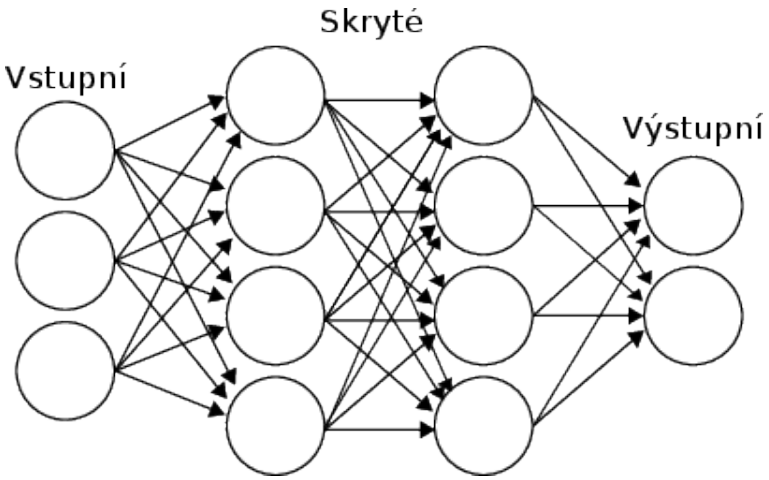
$$\vec{w}(0) = \textit{libovolné}$$
$$\vec{w}(k) = \vec{w}(k-1) + 2\mu\,\vec{z}(k) \qquad \textit{pro} \quad (\vec{w}(k-1)\cdot\vec{z}(k)) \le 0$$
$$\vec{w}(k) = \vec{w}(k-1) \qquad \textit{jinak}$$

Obrázek 2.8: učení perceptronu. Převzato z [10].

Na obrázku 2.8 je znázorněn základní vzorec pro úpravu vah, respektive základní vzorec učení. Základní činností perceptronu je tedy klasifikace nadroviny do dvou nadrovin. Díky tomuto principu jsou jednovrstvé sítě na bázi perceptronů schopné řešit úlohy, kde jsou lineárně separovatelné vstupy. Pro příklad problém XOR není možné řešit pomocí perceptronu. Jako řešení pro lineárně neseparovatelné vstupy jsou vícevrstvé sítě.[23]

Backpropagation

Algoritmus zpětného šíření chyby (backpropagation), je algoritmus, jež je používán ve valné většině všech aplikací neuronových sítí. Je tedy nejrozšířenějším adaptačním algoritmem vícevrstvých neuronových sítí viz obrázek 2.9.



Obrázek 2.9: Příklad vícevrstvé dopředné sítě.

Jak je z obrázku 2.9 patrné, první vrstva této sítě nemá jako vstup neuron, ale surová data, která beze změny předává dál. Za vstupní vrstvou následuje n skrytých vrstev a poslední vrstva je vrstvou výstupní a reprezentuje výstup celé neuronové sítě.

Algoritmus jako takový můžeme rozdělit do tří základních etap. První etapou je dopředné šíření, následující etapou je zpětné šíření chyby a nakonec probíhá aktualizace váhových hodnot.

Prvním krokem algoritmu při fázi učení je takzvané dopředné šíření. Během tohoto procesu obdrží každý neuron vstupní vrstvy signál x_i a zprostředkovává jej neuronům další skryté vrstvy. Neurony skryté vrstvy poté vypočítají svojí aktivační funkci a delegují svůj výstup další vrstvě. Tímto způsobem se hodnoty dostanou až do výstupní vrstvy. Stejný postup se odehrává i u bilogického systému, kde jako vstup může být obraz zachycený pomocí tyčinek, respektive čípků v oku.

Síť je nejprve inicializována s náhodnými váhami a to z intervalu $< -0.5, 0.5 >$. Při zpětném šíření je snaha o přiblížení aktuálního výstupního vektoru k vektoru přiloženému jakožto požadovaný výstup na příslušná data, respektive vektor z trénovací množiny. Hledá se tedy globální minimum chybové funkce. Chyba sítě E(w) je vzhledem k tréninkové množině definována jako parciální součet chyb sítě vzhledem ke všem vzorkům v trénovací množině a závisí na konfiguraci sítě w:

$$E(w) = \sum_{l=1}^{q} E_l(w) \tag{2.17}$$

Cílem je tedy minimalizace ve váhovém prostoru sítě. jedná se však o netriviální úkol. Jako řešení se u jednoduchých modelů používá varianta gradientní metody, která vyžaduje diferencovatelnost chybové funkce. Hlavním problémem gradientní metody je, že pokud již nalezne lokální minimum, pak toto minimum nemusí být globální.[23]

Učení neuronových sítí

Při učení probíhá úprava hodnot parametrů, které se nacházejí v jednotlivých vrstvách neuronové sítě. Pro tuto činnost se používá algoritmus, jež má za následek snižování hodnot gradientu, tento algoritmus se nazývá *gradient descent*. Tento algoritmus je založen na minimalizaci chybové funkce, která je dána na základě vstupu do sítě a odpovídajícího výstupu ze sítě. Mezi nejpoužívanější chybové funkce patří funkce střední kvadratické odchylky a křížová entropie. Střední kvadratická odchylka je použita při výpočtu lineární regrese a je definována jako:

$$E_n = \frac{1}{2} \sum_{n=1}^{N} (y(x_n, w) - t_n)^2 \tag{2.18}$$

Kde E je chybová funkce, x značí vektor vstupu neuronové sítě a t_n je očekávaný výstup ze sítě a $y(x_n, w)$ je výstup neuronové sítě, který odpovídá vstupu v závislosti na aktuálních hodnotách vah w.

Křížová entropie je využita v případech, kdy je neuronová síť trénovaná ke klasifikaci výstupních dat k jednomu výstupu sítě. Tato funkce je tedy definována podle následujícího vztahu:

$$E_n = -\sum_{n=1}^{N} \sum_{k=1}^{K} t_{kn} lm(y(y_n, w)) \tag{2.19}$$

Snižování hodnoty gradientu je poté iterační algoritmus, který je založen na postupném snižování gradient spolu s chybovou funkcí. Gradient chybové funkce vzhledem k aktuálním váhám sítě představuje poté směr, ve kterém chyba nejvíce roste a tento princip lze popsat pomocí vztahu:

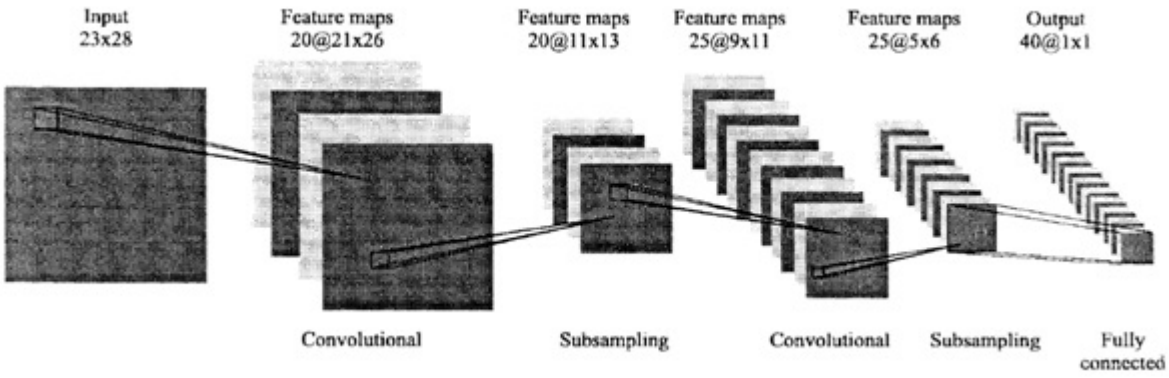
$$w^{\tau+1} = w^{\tau} - \eta \bigtriangledown E(w^{\tau}) \tag{2.20}$$

Kde τ znamená hodnotu iterace a η značí krok učení (velikost po které se mění váhy)[6].

2.4.2 Konvoluční neuronové sítě

Konvoluční neuronové sítě jsou speciálním typem dopředných neuronových sítí. Využívají několik speciálních vrstev neuronů a struktura jejich propojení je pevně daná. Při roz-

poznávání obrazu klasickými neuronovými sítěmi nastává problém v okamžiku, kdy jsou například na dvou obrázcích dva totožné objekty a pouze jsou vůči sobě osově posunuté (například horizontální, respektive vertikální osa). Klasická neuronová síť by tento problém vyhodnotila jako objekty s velkým rozdílem(záleží na velikosti posuvu). Konvoluční sítě tento problém řeší tak, že obraz postupně rozdělují na menší části a sdílejí váhy[7][13][4].



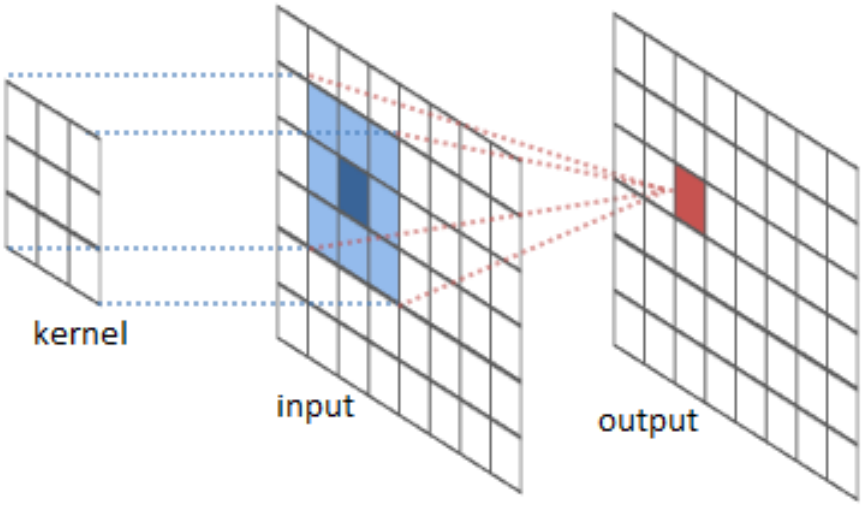
Obrázek 2.10: Architektura konvoluční neuronové sítě[15].

Na obrázku je znázorněna architekture velice jednoduché konvoluční neuronové sítě. V praxi a zejména pro rozpoznávání obličejů, bývají tyto sítě mnohem složitější, avšak s tím, že si zachovávají relativně vysokou rychlost z hlediska odezvy na vstup. Největší problém u konvolučních neuronových sítí bývá jejich náročné učení.

Jelikož jsou konvoluční neuronové sítě speciálním případem dopředných neuronových sítí, tak učení konvolučních neuronových sítí probíhá na stejném principu, jako učení hlubokých dopředných sítí. Konvoluční neuronová sít využívá některé speciální vrstvy a to konvoluční vrstvu a vrstvu podvzorkování[15].

Konvolucni vrstva

Základem je rozdělení vstupního obrázku na mnoho vzájemně se překrývajících se čtverců o fixní velikosti. Jak lze vidět na obrázku 2.11, kernel značí konvoluční jádro, které na vstupním obrázku představuje jakési posuvné okénko, které se posouvá po jednom pixelu. Postupnou operací konvoluce s konvolučním jádrem, respektive filtrem a místem na vstupním obrázku získáváme příznakovou mapu.



Obrázek 2.11: Konvoluční vrstva[5].

Díky různě natrénovaným filtrům můžeme například detekovat hrany, rohy a podobně, tedy každý typ filtru generuje jinou příznakovou mapu.

Každý filtr v konvoluční vrstvě generuje jednu příznakovou mapu, kde její velikost je dána velikosstí kroku, který udává o kolik pixelů se plovoucí okno posune a velikostí nulového paddingu, kdy je vstupní obrázek rozšířen o nulový rám[4].

Vrstva podvzorkování

Úkolem této vrstvy je zmenšit vstupní obrázek a to na základě velikosti okna spolu s funkcí max/mean transformuje plochu pixelů obsaženou v okně na jeden pixel a to na základě typu funkce. Pokud se jedná o typ funkce max, z okna se vybere pixel obsahující největší hodnotu. V opačném případě se použije funkce mean, která vybere průměrnou hodnotu pixelů[13][4].

Rectified Linear Unit

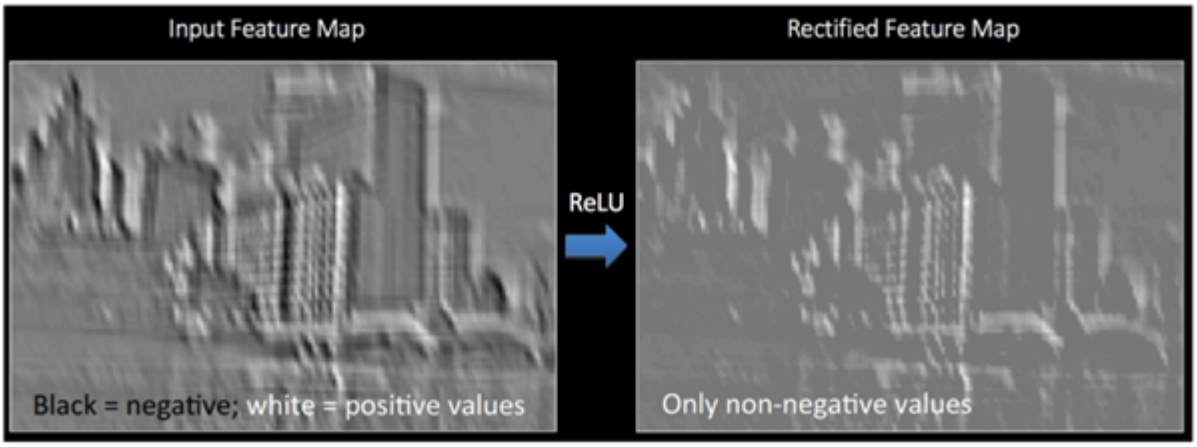
Zkráceně ReLU. Jedná se o techniku používanou v konvolučních neuronových sítích, která usnadňuje učení sítí. Odstraňuje zdroje nelinearity a to tak, že nahrazuje všechny negativní hodnoty pixelů hodnotou nula a nezáporné hodnoty kopíruje beze změny na výstup. Je tedy definována jako:

$$Vstup = Max(0, Vstup) \tag{2.21}$$

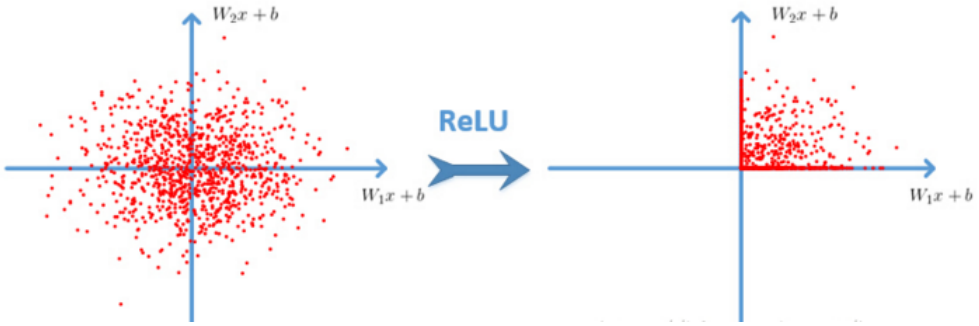
Princip ReLU ilustruje obrázek2.12 [4].

Obrázek poté znázorňuje ReLU použitou v dvojrozměrném prostoru.

[5]Převzato z: http://xrds.acm.org/blog/2016/06/convolutional-neural-networks-cnns-illustrated-explanation/



Obrázek 2.12: Ilustrace ReLU. Obrázek převzat z [4].



Obrázek 2.13: Rozložení ReLU[6].

2.4.3 Klasifikace

Klasifikace, respektive třídění, je zobrazování určitých objektů do předem definovaných tříd. Algoritmy klasifikace mohou být jednoduché (pro vektory například na bázi euklidovké vzdálenosti), ale i složité (využívající strojového učení). Níže uvedené algoritmy jsou vhodné pro klasifikaci vektorů.

Klasifikátor SVM

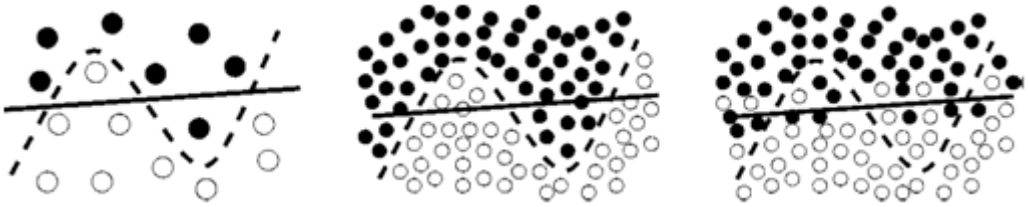
Klasifikátor Support Vector Machine je jeden ze zástupců klasifikátoru využívajících při své činnosti strojového učení. V nejjedonušším případě, kdy rozdělujeme datovou množinu do dvou tříd je úkolem SVM rozdělit data pomocí nadroviny tak, aby rozdělení bylo rovnoměrné a aby v okolí dělící nadroviny byla co největší oblast bez výskytu dat. Místo použití nelineární křivky, aby bylo možné data rozdělit, namapují se tato data do vyšších dimenzí a ty se poté stanou lineárně oddělitelnými.

Pro klasifikační problém SVM používá jedno z několika druhů jader, čímž mohou být například radial basis function, popřípadě sigmoidální funkce. Každý typ jádra má určité parametry, které je třeba upravit na základě trénovacích dat. Úprava parametrů je možná

[6]Převzato z: http://prog3.com/sbdm/blog/cyh_24/article/details/50593400

na základě vytvoření několika modelů, kde každý model je nakonfigurován pomocí jiných parametrů jádra a poté se vybere ten model, jenž vykazuje nejlepší vlastnosti.

Modifikace klasifikátoru pro klasifikaci vícedimenzionálních dat probíhá tak, že je pro každou třídu použit oddělený klasifikátor, který bude klasifikovat, zda vstupní vzorek do dané třídy patří, či nikoliv.



Obrázek 2.14: Ilustrace dilematu příliš přesného rozdělení (přeučení). Převzato z [16].

Na obrázku 2.14 lze vidět typický problém při rozdělování dat to dvou tříd, kde je použita lineární křivka a křivka vyššího řádu. U vzorku vidíme, že data lépe rozděluje křivka vyššího řádu, nicméně se jedná o malou sadu dat a pro větší přesnost je potřeba mít sadu větší. Vzorek umístěn uprostřed poukazuje na vhodnější rozdělení dat pomocí křivky vyššího řádu, tedy hypotéza podle prvního vzorku s menším počtem dat byla pravdivá. Poslední vzorek však poukazuje na to, že se může nastat situace, kde nově příchozí data se nacházejí na takových pozicích, kdy je vhodnější použít lineární křivku pro rozdělení dat.

Výhodou tohoto algoritmu je, že obsahuje regularizační parametr, který, pokud je správně zvolen, zamezuje přeučení[16].

Klasifikátor k - nejbližších sousedů

K-Nearest Neighbourg, neboli K-nejbližších sousedů je klasifikátor využívající vzdálenosti (například euklidovské, manhatanské a podobně) a spadá mezi neparametrické klasifikátory.Algoritmus pracuje s příznakovou množinou o N přiznakových vektoru ve vícerozměrném prostoru. Na základě klasifikace jsou prvky řazeny do jedné z tříd. Následuje hledání k - nejbližších sousedů pro neznámý prvek a to z hlediska vzdálenosti definované metrikou z přiložené trénovací množiny. Pro 1 NN predikuje stejnou třídu jako má nejbližší instance v trénovací množině. V případě k NN algoritmus hledá k nejbližších trénovacích bodů, kde třída je poté vybrána podle četnosti bodů z určité třídy. Postup klasifikace je tedy definován jako:

$$\{(x_i, w_i)\}_{i=1,..,K} \tag{2.22}$$

Kde xi je vzorek ke kterému je přiřazena třída wi a K je velikost trénovací množiny.

Neznámý prvek, respektive vektor je poté přiřazen k té třídě, která se vyskytuje nejčastěji u nalezených k-sousedů.

Výhodou algoritmu je, že nepoužívá žádné explicitní trénovaní, či natrénovaný model. Nevýhody jsou paměťová náročnost, velká chybovost pokud je nějaká třída majoritní z hlediska počtu trénovacích bodů a algoritmus je též velice citlivý na šum[1][12].

klasifikátor minimální vzdálenosti

Nearest centroid, neboli klasifikátor minimální vzdálenosti. Jedná se o jednoduchý a rychlý algoritmus, kde základní myšlenkou tohoto algoritmu je fakt, že algoritmus předpokládá

korespondenčnost cílové třídy s individuálním shlukem a používá průměrnou hodnotu se shluku ke zjištění třídy nového vzorku. Vzor pro třídu Cj je definován jako:

$$\mu = \frac{1}{|C_j|} \sum_{x_j \in C_j} x_i \tag{2.23}$$

Kde xi je trénovací vzorek označen jako třída Cj. Během klasifikace je označení třídy neznámého vzorku zjištěno jako[3]:

$$C(x) = \underset{C_j}{argmind}(\mu_{Cj}, x) \tag{2.24}$$

2.4.4 Hodnocení kvality klasifikace

Jedny z nejpoužívanějších metrik použitých zejména v biometrických systémech, jsou metriky, které se zakládají na ohodnocování pomocí dosaženého skóre. Níže uvedené metody pracují na principu srovnávání sejmutých vlastností (například vícedimenzionální vektor), které jsou porovnávány s uloženými vlastnostmi.

FRR (False rejection rate) taktéž označován jako chyba prvního stupně. Jedná se o chybu, kdy se 2 vzorky mají klasifikovat jako stejné, případně patřící do stejné skupiny, ale systém je vyhodnotí jako odlišné, respektive patřící do jiné třídy. Jelikož je výstupem neuronových sítí 128dimenzionální vektor, kde je vzdálenost dvou vektorů definována jako euklidovská s tím, že minimální vzdálenost je 0 a maximální vzdálenost je 4. Tato vzdálenost je poté brána jako práh, podle kterého se usuzuje podobnost vektorů. Vztah pro výpočet hodnoty FRR je poté:

$$FRR = \frac{N_{FR}}{N_{EIA}} * 100[\%] \tag{2.25}$$

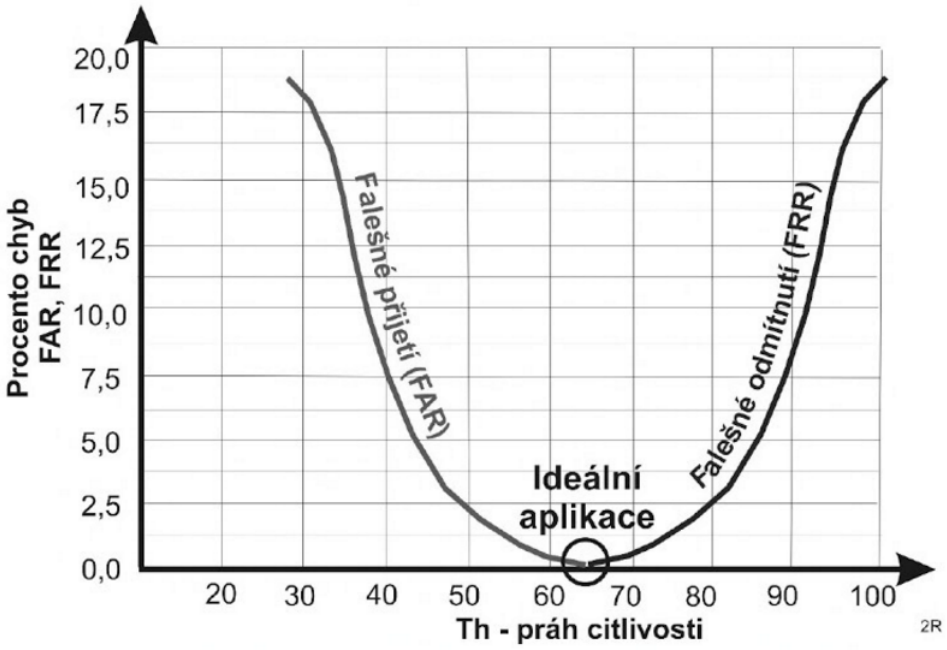
Kde N_{FR} značí počet chybných odmítnutí a N_{EIA} znamená počet všech pokusů o porovnání vektorů.

FAR (False Acceptance Rate) respektive chyba druhého stupně. Jedná se o chybu, kdy se mají dva vzorky, respektive vektory klasifikovat jako odlišné, ale klasifikují se jako totožné, popřípadě patřící do stejné třídy. Výpočet tohoto koeficientu poté popisuje následující vztah:

$$FAR = \frac{N_{FA}}{N_{EIA}} * 100[\%] \tag{2.26}$$

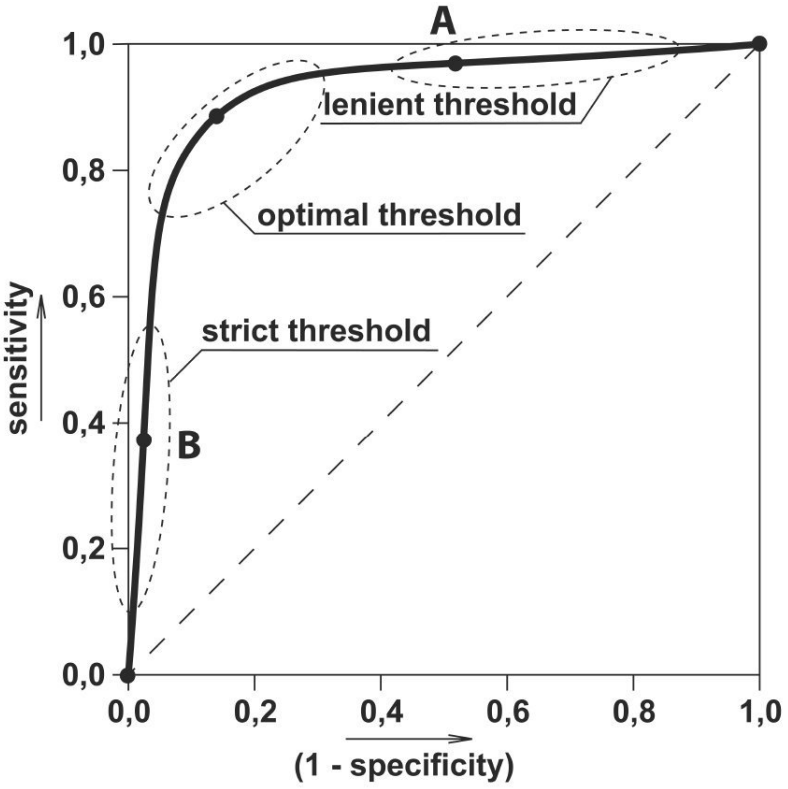
kde N_{FA} znamená počet chybných přijetí, respektive klasifikování.

Výše uvedené výpočty chybovosti (FAR a FRR) vycházejí z předpokladu, že počty chyb a pokusů jsou vyrovnané. Ideální zařízení, vztah, respektive klasifikace nevykazuje žádnou chybovost, tedy všechny vzorky jsou klasifikovány správně tj. neexistují ani neoprávněně odmítnuté, ani neoprávněně akceptované vzorky. Tento stav je v bodě, kdy $FAR = FRR = 0$, což znázorňuje obrázek



Obrázek 2.15: Zobrazení ideálního stavu křivky FRR/FAR. Převzato z [18].

ROC (Receiver Operating Characteristics) Na základě této křivky, která určuje vzájemný vztah pravděpodobností FRR a FAR, můžeme objektivněji posuzovat kvalitu klasifikace. Bod, v němž se křivky protínají se nazývá Equal Error Rate (ERR). Ilustraci ROC křivky znázorňuje obrázek 2.16.[18]



Obrázek 2.16: Ilustrace ROC křivky[7].

2.4.5 Architektury CNN

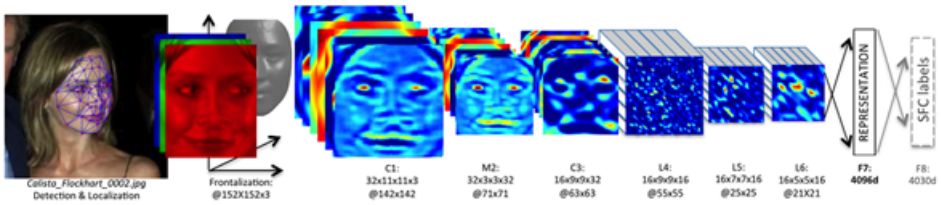
Mezi nejznámější architektury konvolučních neuronových sítí patří LeNet, která je jednou z prvních sítí a byla použita pro rozpoznávání vzorů v obraze. Moderní konvoluční neuronové sítě dosahující vysoké přesnosti z hlediska rozpoznávání obličejů, zpravidla však mají velice komplikovanou architekturu a tyto sítě jsou trénovány na obrovských často neveřejných datových sadách.

DeepFace

Konvoluční neuronová síť představena společností Facebook v roce 2014. Síť byla trénovaná na více než čtyřech milionů obrázků obličejů, které představovali více než čtyři tisíce identit a může se pyšnit úspěšností 97.35% na datové sadě LFW.

Síť zahrnuje více než 120 milionů parametrů, kde 95% z nich pochází z lokálně a plně propojených vrstev. Síť je trénovaná pro vstup 3 dimenzionálního zarovnaného obrázku obličeje o rozměrech 152x152 pixelů. Výše uvedená přesnost rozpoznávání osob vychází z faktu, že celá síť tvoří několik kombinací sítí uvedených na obrázku 8 s tím, že každá síť byla trénovaná na náhodně zvolených obrázcích obličejů[].

[7]Obrázek převzat z: http://www.prolekare.cz/casopis-lekaru-ceskych-clanek/roc-analyza-a-vyuziti-analyzy-nakladu-a-prinosu-k-urceni-optimalniho-deliciho-bodu-5403



Obrázek 2.17: Architektura DeepFace[21].

FaceNet

Jedná se o konvoluční neuronovou síť trénovanou na privátní datové sadě obličejů společnosti Google. Tato síť byla představena v roce 2015 a může se chlubit úspěšností rozpoznání 99,63% na datové sadě LFW a úspěšností 95,12% na datové sadě YouTube Faces DB. Síť je na bázi známe sítě GoogLeNet s tím, že je upravena právě pro použití při rozpoznávání obličejů.

type	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj (p)	params	FLOPS
conv1 (7×7×3, 2)	112×112×64	1							9K	119M
max pool + norm	56×56×64	0						m 3×3, 2		
inception (2)	56×56×192	2		64	192				115K	360M
norm + max pool	28×28×192	0						m 3×3, 2		
inception (3a)	28×28×256	2	64	96	128	16	32	m, 32p	164K	128M
inception (3b)	28×28×320	2	64	96	128	32	64	L_2, 64p	228K	179M
inception (3c)	14×14×640	2	0	128	256,2	32	64,2	m 3×3,2	398K	108M
inception (4a)	14×14×640	2	256	96	192	32	64	L_2, 128p	545K	107M
inception (4b)	14×14×640	2	224	112	224	32	64	L_2, 128p	595K	117M
inception (4c)	14×14×640	2	192	128	256	32	64	L_2, 128p	654K	128M
inception (4d)	14×14×640	2	160	144	288	32	64	L_2, 128p	722K	142M
inception (4e)	7×7×1024	2	0	160	256,2	64	128,2	m 3×3,2	717K	56M
inception (5a)	7×7×1024	2	384	192	384	48	128	L_2, 128p	1.6M	78M
inception (5b)	7×7×1024	2	384	192	384	48	128	m, 128p	1.6M	78M
avg pool	1×1×1024	0								
fully conn	1×1×128	1							131K	0.1M
L2 normalization	1×1×128	0								
total									7.5M	1.6B

Obrázek 2.18: Tabulka architektury konvoluční sítě[19].

Síť na obrázku 2.18 popisuje vnitřní architekturu sítě. Konvoluční síť je trénována pomocí sestupného stochastického gradientu se standardním učením zpětného šířeníbackpropagation a byla trénována na více než milionu obrázků. Síť je stavěná pro vstup o velikosti 224x224 pixelů a generuje 128dimenzionální výstupní vektor [19].

OpenFace

OpenFace je veřejně dostupnou sítí implementovanou ve frameworku Torch. Tato síť je použita v této práci pro rozpoznávání obličejů. Síť je trénována na více než 500 tisících obrázků obličejů, které zahrnují největší dostupné datové sady a to CASIA-WebFace a FaceScrub a LFW. Síť se chlubí úspěšností téměř 97.3% na datové sadě LFW[5].

type	output size	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj
conv1 ($7 \times 7 \times 3, 2$)	$48 \times 48 \times 64$						
max pool + norm	$24 \times 24 \times 64$						m $3 \times 3, 2$
inception (2)	$24 \times 24 \times 192$		64	192			
norm + max pool	$12 \times 12 \times 192$						m $3 \times 3, 2$
inception (3a)	$12 \times 12 \times 256$	64	96	128	16	32	m, 32p
inception (3b)	$12 \times 12 \times 320$	64	96	128	32	64	ℓ_2, 64p
inception (3c)	$6 \times 6 \times 640$		128	256,2	32	64,2	m $3 \times 3, 2$
inception (4a)	$6 \times 6 \times 640$	256	96	192	32	64	ℓ_2, 128p
inception (4e)	$3 \times 3 \times 1024$		160	256,2	64	128,2	m $3 \times 3, 2$
inception (5a)	$3 \times 3 \times 736$	256	96	384			ℓ_2, 96p
inception (5b)	$3 \times 3 \times 736$	256	96	384			m, 96p
avg pool	736						
linear	128						
ℓ_2 normalization	128						

Obrázek 2.19: Architektura OpenFace ve verzi nn4.small2[5].

Jak lze z tabulky obrázku 2.19 vypozorovat, konvoluční neuronová síť OpenFace je inspirována sítěmi od společnosti Google a Facebook. Výstupem sítě je 128 dimenzionální vektor rysů, který je dále klasifikován.

Kapitola 3

Návrh a implementace

Tato kapitola popisuje praktickou část práce. Vzhledem k existenci různých algoritmů, které v různých prostředích probíhají různě rychle s odlišnou efektivitou a výsledkem, tak během celé implementace probíhaly paralelně experimenty, kde se porovnávaly výsledky a rychlost implementace za účelem vybrat ty algortimy, které tyto dva parametry splňují nejlepší mírou.

Velkou míru na úspěšnost systému mají kamery z hlediska jejich umístění vzhledem k procházejícím lidem, kde ideální pozice kamery je taková, kde je obličej viděn celý a natočen směrem ke kameře. Vliv osvětlení je dalším nezanedbatelným parametrem, kde příliš málo světla s kombinací s objektivem kamery, který má nízkou světelnost a dalšími parametry má za následek drastického zhoršení kvality záznamu, který má za následek horší výsledky z hlediska detekce a validace obličejů nacházejících se na samotném záznamu.

3.1 Systémové požadavky

Systém pro rozpoznávání osob je implementován v jazyce *Python 2.7* a byl testován nad systémem Ubuntu 14.04. Použita je knihovna *OpenCv*[1]ve verzi 3.2.0, ale je možné použít i verzi 3.1.0, dále knihovna *dlib*[2]]ve verzi 18.16 a knihovna *openface*[3]ve verzi 0.2.0. Práci s neuronovou sítí zprostředkovává framework *torch*[4]. Framework *torch* je uzpůsoben pro kooperaci s grafickou kartou od společnosti NVIDIA, která mnohonásobně urychluje práci s neuronovou sítí.

3.2 Struktura systému

Systém je navrhován tak, aby byl schopen rozpoznávat osoby a to pokud možno v reálném čase. Za tímto účelem je celý systém částečně paralelizován a umožňuje použití grafické karty od společnosti NVIDIA vlastnící CUDA jádra. Grafická karta značně urychluje odezvu z konvoluční neuronové sítě. Jelikož se jevilo rozpoznávání osob pomocí jen jedné neuronové sítě jako nedostačující a to zejména v situacích, kdy byla osoba natočena ke kameře tak, že byl její obličej natočen z profilu. Tento problém byl částečně vyřešen pomocí druhé sítě trénované právě na obličejích natočených z profilu. Přidnání funkcionality

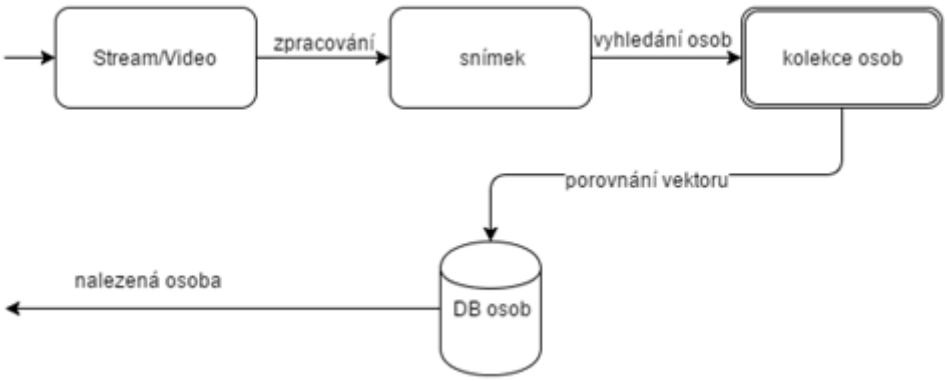
[1]Knihovna OpenCv je dostupná z: http://docs.opencv.org/3.2.0/

[2]Knihovna dlib je dostupná z: https://github.com/davisking/dlib/releases/download/v18.16/dlib-18.16.tar.bz2

[3]Knihovna openface je dostupná z: https://cmusatyalab.github.io/openface/setup/

[4]Framework torch je dostupný z: http://torch.ch/

pro rozpoznávání obličejů z profilu však přineslo značné zpomalení celého systému. Systém umožnuje běh i s prázdnou databází osob, kde si sám vytvoří ke každé nově nalezené osobě vektor rysů, který si poté uloží do databáze. Vzhledem k předchozímu faktu byl použit klasifikátor lokálního centroidu, který díky své jednoduchosti a nepotřebě trénování umožňuje učení systému za ostrého běhu. Jako výstup systému slouží vizualizace nalezených obličejů přímo na videozáznamu spolu s log. souborem, do kterého jsou po nalezení osoby zapsány s příslušným jménem a časem sestrvávání na videozáznamu. Systém rovněž podporuje jako vstup i přímý stream z kamery.



Obrázek 3.1: Schéma systému vyhledávání osob.

Jak je patrné na velmi zjednodušeném obrázku reprezentující činnosti systému 3.1, systém nejprve předzpracovává jednotlivé snímky jdoucí z kamery, respektive videozáznamu. jelikož systém separuje pohyblivé popředí od statického pozadí, je vhodné, aby kamera byla pevně uchycena a nedocházelo k pohybu, či větším otřesům, které by značně omezili výkonnost celého systému z hlediska počtu snímku za vteřinu. Použité algoritmy jsou implementovány ve zmíněných knihovnách a jejich výběr byl posuzován na základě požadavku pro práci v reálném čase. Po separaci pohybujícího se popředí, které může být rozděleno i na více částí, následuje samotné vyhledávání obličejů. Pokud je na některé části z pohybujícího se popředí nalezen obličej, je tento obličej sledován a každým dalším nalezením obličeje na pozici právě sledovaného obličeje jsou pomocí neuronových sítích separovány vektory rysů, které jsou následně zpracovávány klasifikátorem. Po ztrátě sledované osoby (osoba opustila videozáznam) je tento výskyt zapsán do logu.

3.2.1 Zpracování snímku

První fází celého rozpoznávání obličejů je specifická úprava každého snímku ze záznamu, respektive streamu z videokamery. Jelikož je detekce obličejů časově náročná operace vzhledem k velkému rozlišení snímků, je vhodné udělat výřez plochy, na které se obličeje mohou nacházet (touto plochou je zpravidla pohybující se popředí videa).

Existuje mnoho metod jak vytvořit masku pro pohybující se část videa, podle které se poté označí místa, která budou později dále zpracovávána jako výřezy s potencionálním obličejem. Důraz na algoritmus provádějící tuto operaci byl na rychlost a přesnost, aby nedocházelo k příliš velkému počtu velice malých ploch, které by v extrémním případě mohli způsobit i to, že by se plocha tímto způsobem značně zredukovala (malé segmenty by se vyfiltrovali místo toho, aby se spojili do celistvé plochy) a mohlo by se tak ohrozit následné vyhledávání obličejů.

Po četných experimentech byl poté vybrán algoritmus využívající metody *k - nejbližších sousedů*, tato metoda není z metod které jsou v knihovně OpenCv implementovány tou nejrychlejší, nicméně se ukázala jako vhodná vzhledem k faktu, že nalezenou plochu nesegmentuje na malé části tolik, jako ostatní algoritmy a díky tomu je mnohem snazší výsledné plochy obalit do obdélníkových entit, ze kterých se následně udělají výřezy na kterých se předpokládá, že se mohou nacházet obličeje. Časová náročnost i vzhledem k vyšší časové náročnosti oproti ostatním metodám je vůči časové náročnosti vyhledávání obličejů téměř zanedbatelná. Nižší časová náročnost je také způsobena tím, že je snímek nejdříve zmenšen a převeden z barevné složky do odstínů šedi, na kterém je teprve proveden algoritmus a poté jsou výsledné obdélníkové plochy opět přepočteny na původní velikost snímku.

Implementovaný algoritmus umožňuje měnit masku popředí na základě historie snímku a to podle zadávaného parametru. Tato možnost se hodí zejména, pokud na videu dojde k velké změně, která bude zabírat velkou plochu videa a zároveň se na záznamu bude nacházet delší dobu. Příkladem mohou být například otevřené dveře, či odložený předmět větších rozměrů (velikost lidské hlavy).

3.2.2 Detekce obličejů

Nejpoužívanější přístupy k detekci obličejů jsou použitím Viola-Jonsonova algoritmu (využívající haarovy příznaky), který je implementován v knihovně *OpenCV*, nebo použití algoritmu využívající HOG detektoru, jehož implementace je dostupné v knihovně *dlib*.

Výhody Viola-Jonsonova algoritmu je zejména jeho rychlost vyhledávání potencionálních míst, kde se mohou obličeje nacházet a to při vhodně zvolených parametrech. V práci je tento algoritmus využit pro jakousi predikci oblastí potencionálních obličejů a při vhodně zvolených parametrech vykazuje rychlou detekcí, oproti algoritmu implementovaného v knihovně dlib. Jeho největší slabinou jsou však falešné detekce, kdy při výše uvedených parametrech pro tento algoritmus jsou okamžiky, kdy je nalezeno více míst neobsahujících žádný obličej, než míst obličej opravdu obsahující. Rovněž je jeho přesnost z hlediska vycentrování ohraničujícího čtverce místy nepřesná, proto tuto oblast zvětšuji na dvojnásobek úhlopříčky nalezeného čtverce s tím, že jejich střed je totožný.

Vyhledávání obličejů natočených na záznamu z profilu je poněkud složitější a to zejména z toho důvodu, že není k dispozici přesný HOG detektor z knihovny *dlib* ale je použit pouze Viola-Jonsonův algoritmus s tím, že hledání probíhá ve třech fázích. Dostupné detektory jsou schopné vyhledávat pouze obličeje natočené doleva, je tedy nutné následně celou plochu určenou k prohledávání zrcadlově otočit podle osy y a provést vyhledávání znovu. Všechny tyto oblasti, kde by se mohly obličeje nacházet, jsou rovněž zvětšeny na dvojnásobek úhlopříčky nalezeného čtverce. Poslední fází je znovu prohledávání takto zvětšených oblastí rovněž stejným algoritmem, tentokrát jsou však parametry uzpůsobeny tak, aby vyhledávání probíhalo s mnohem větší přesností. Navzdory upraveným parametrům je stále toho vyhledávání nepřesné a může tak vyvolávat falešné místa s obličeji.

Algoritmus implementovaný v knihovně dlib využívající HOG detektor je využit pro přesnou detekci obličejů natočených čelně na kameru. Algoritmus ve srovnání s Viola-Jonsonovým algoritmem je pomalejší, nicméně dosahuje podobné rychlosti, pokud jsou pro algoritmus Viola-Jonsonův zvoleny parametry pro přesné hledání, ale se zachovanou přesností, kde oponující algoritmus stále hlásí případy falešné detekce.

V praxi se ukázalo, že tento způsobem vyhledávání je výpočetně velice náročný a znesnadňuje tak požadavek na práci systému v reálném čase.

3.2.3 Zarovnání a rozpoznávání obličeje

Před samotným zpracováním obrázku neuronovou sítí je vhodné kvůli větší přesnosti tento obrázek nějakým způsobem zarovnat a to ideálně z hlediska trojrozměrného prostoru. Kvůli výpočetní náročnosti a nutnosti záznamů každého obličeje z několika úhlů bylo zvoleno zarovnávání ve dvojrozměrném prostoru a to pomocí klíčových bodů popisujících každý obličej. Knihovna *dlib* obsahuje několik předtrénovaných konvolučních neuronových sítí, které jsou schopny na každém obličeji nalézt 68 respektive 196 klíčových bodů.

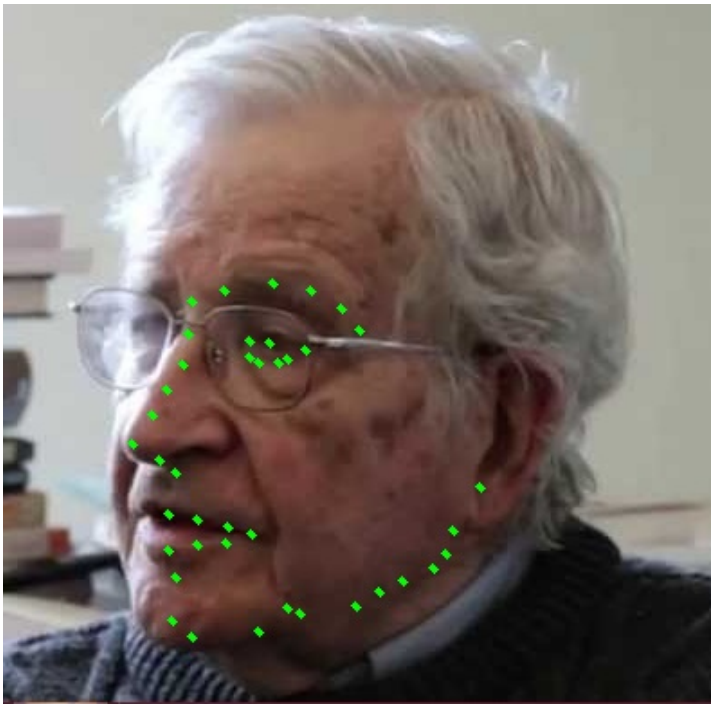
Jelikož je v této práci použita konvoluční neuronová síť pro rozpoznávání obličejů z profilu, bylo tedy nutné předzpracovat snímky obličejů i pro tuto síť. K tomuto účelu zprostředkovává knihovna *dlib* prostředí pro natrénování vlastní sítě. Snímky je ale nejprve nutné těmito body označit ručně a uložit do souboru pomocí něhož se poté detektor učí. Trénování probíha na základě parametrů, které jsou vysvětleny v [9].

na obrázku 3.2 je uvedena ukázka nalezení klíčových bodů, kde na obrázku A je nalezeno 68 bodů pomocí sítě hledající body na obličejích ze předu a na obrázku B je znázorněno 40 klíčových bodů, pomocí sítě hledající klíčové body na obličejích z profilu.

Obrázek 3.2: Klíčové body na obličeji z profilu.

Při kombinaci hledání obličeje může nastat situace taková, že oba detektory (obličej ze předu, obličej z profilu) vyhodnotí obličej jako vhodného kandidáta pro zpracování příslušnou sítí. Abychom zamezili zbytečnému průchodu oběma sítím pro tentýž obličej, je vhodné vybrat pouze obličej z jednoho detektoru. Nejvhodněji se pro tento účel postup, kdy se naleznou na obličeji klíčové body pomocí a poté se vyhodnotí pozice špičky nosu vzhledem k pozicím vnějších okrajů očí, jak ilustruje obrázek 3.4.

Na obrázku jsou označeny body A, B a C. Abychom tyto body získali, je nutné nejprve nalézt 68 klíčových bodů, poté spočítat rovnici přímky mezi vnějším okrajem levého a pravého oka. Poté se spočte rovnice přímky, která je kolmá na předešlou přímku a která



Obrázek 3.3: Klíčové body na obličeji z profilu.

prochází bodem, kde se nachází špička nosu. Mezi těmito přímkami se vytvoří průnik, který reprezentuje bod B. Poté se porovná pozice bodu B s pozicemi bodů A a C. Pokud bod B je velice blízko, či přímo zasahuje za vnější okraje očí, poté je tento obličej vyhodnocen jako vhodný pro průchod neuronovou sítí trénovanou na obličejích z profilu. Pokud se špička nosu nachází mezi okraji očí, je tento obličej vyhodnocen jako vhodný pro průchod neuronovou sítí trénovanou na obličejích ze předu.

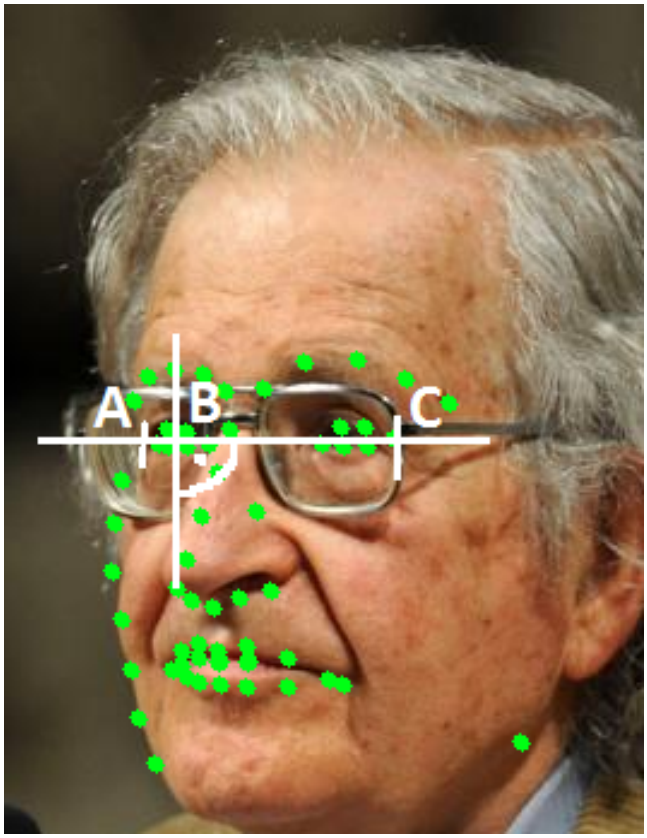
3.2.4 Sledování

Při vyhledávání obličejů na snímcích nastávají situace, kdy na N snímcích nejsou nalezeny pozice vyhodnoceny jako obličeje, načež se na těchto pozicích obličeje opravdu nacházejí. Pokud nastane situace, že je obličej nalezen a poté nějakou dobu žádný z detektorů tento obličej znovu nenalezne, mohl by být tento obličej chybně vyhodnocen z hlediska nedostatků snímků, které by se přiřadili právě téže osobě.

Z výše uvedených důvodů bylo implementováno sledování osob podle algoritmu 2.3.1, který je součástí knihovny *dlib*. Jedná se o relativně robustní způsob sledování objektů, kde se sledovací algoritmus učí za běhu sledování objektů.

Pokud se nalezne pozice obličeje pomocí výše uvedených detektorů, vytvoří se nad tímto místem nový objekt, zajišťující sledování dané osoby. Pokud je nalezena pozice obličeje na pozici právě sledovaného obličeje, je poté tento obličej zpracován a výsledný vektor rysů z příslušné sítě je uložen do jednoho z polí, které si sledovací objekt uchovává (pole pro obličeje z profilu a pole pro obličeje ze předu.)

Pokud jsou při sledování vyhodnoceny špatné sledovací parametry (velká chyba) a je na této pozici nalezen obličej, tak se aktualizuje pozice podle nalezeného obličeje. Pokud se již nějakou dobu nad sledovanou osobou nenalezly žádné obličeje, tak sledování probíhá dál do doby, dokud jsou vykazovány dobré parametry sledování.



Obrázek 3.4: Ilustrace bodů na obličeji pro rozhodnutí mezi detektory.

Pokud se sledovaný objekt ztratí, což může nastat z důvodu odchodu osoby ze sledované oblasti, nedokonalostí sledování (čímž je způsobena ztráta objektu) a nebo skončí videozáznam, poté se zpracují všechny uložené vektory rysů. Zpracování probíhá tak, že se všechny vektory reprezentující rysy obličeje z profilu zpracují separátně od vektorů pocházejících z obličeje ze předu. Nejprve se příslušné vektory zpracují pomocí shlukové analýzy *DBSCAN* [12], čímž se vytvoří shluky a vybere se pouze jeden shluk, který obsahuje nejvíce vektorů. Shluky se vytváří pomocí euklidovské vzdálenosti mezi vektory. Tento postup má rovněž za následek vyfiltrování vzdálených vektorů. Nad vítězným shlukem je nakonec vypočítán centroid a to pomocí aritmetického průměru.

Poslední fází je porovnání získaných vektorů spolu s databází. Databáze pro ukládání osob spolu s příslušnými vektory je implementována pomocí *MongoDB* [13]. jedná se o *NoSQL* dokumentově orientovanou databází, kde jednotlivé dokumenty jsou zde uloženy jako klíč a hodnota, kde hodnota mohou být dále libovolně zanořené dokumenty. Záznamy se ukládají ve formátu *JSON*. Do databáze se ukládá vždy jméno příslušné osoby a k ní vektor rysů spolu s identifikátorem, zda se jedná o vektor extrahovaný pomocí profilové neuronové sítě, či neuronové sítě zpracovávající obličeje ze předu.

Oba vektory od každé osoby se porovnají se všemi vektory, které jsou uloženy v databázi. Poté se vyberou ty vektory, jejichž euklidovská vzdálenost od současných vektorů je nejnižší. Jelikož systém umožňuje automatické ukládání neznámých osob do databáze, jsou tyto

[12]DBSCAN analýza: https://www.aaai.org/Papers/KDD/1996/KDD96-037.pdf
[13]MongoDb: https://docs.mongodb.com/

vzdálenosti porovnány s mezní definovanou vzdáleností, která určuje, zda se jedná o osobu neznámou, nebo je vektor dostatečně blízko vektoru příslušné osoby v databázi. Vzhledem k odlišnosti obou neuronových sítí je pro každou síť nastaven jiný mezní parametr. Pokud je jeden z vektorů dostatečně blízko od vektoru osoby z databáze a druhý vektor nikoliv, je poté vektor, který nesplňuje tuto vzdálenost uložen do databáze pod příslušnou osobou nalezenou pomocí prvního vektoru. Pokud oba vektory odpovídají různým osobám, je pro identifikaci osoby použit ten vektor, který vykazuje menší vzdálenost.

Díky výše uvedeným postupům je možné systém spustit i nad videozáznamem obsahující osoby i bez příslušné databáze. Systém začne sám ukládat osoby, kde ke každé osobě bude přiřazeno automaticky vygenerované jméno.

3.3 Neuronové sítě

Jelikož se na záznamech z kamer pohybuje mnoho osob a to různými směry s různým natočením obličeje vůči ke kameře, bylo rozhodnuto použít více konvolučních sítí pro rozpoznávání obličeje a to pro rozpoznávání obličeje směrem ze předu, což je běžné pro úlohy zabývajícími se rozpoznáváním obličejů, ale byla také natrénovaná konvoluční neuronová síť pro rozpoznávání obličejů z profilu.

Konvoluční neuronové sítě pro rozpoznávání obličejů ze předu i z profilu mají totožnou topologii převzatou z OpenFace popsané výše. První síť, tedy síť pro rozpoznávání obličejů ze předu je k dispozici předtrénovaná v rámci knihovny OpenFace a byla trénovaná na více než 500 tisících snímcích s tím, že byla trénována na grafické kartě NVIDIA Tesla K40 a celková doba trénování trvala okolo 12hodin. Trénovací množina byla vytvořena z barevných snímků obličejů, které byli následně zarovnány a zmenšeny, respektive zvětšeny na velikost 96 x 96 pixelů. Trénování probíhalo pod frameworkem Torch. Výstup ze sítě je reprezentován 128dimenzionálním vektorem.

3.3.1 Profilová síť

Jelikož je síť používaná pro rozpoznávání obličejů ze předu natrénovaná na obličejích vykazující tuto vlastnost, má velice špatné výsledky co se týče rozpoznávání obličejů z profilu.

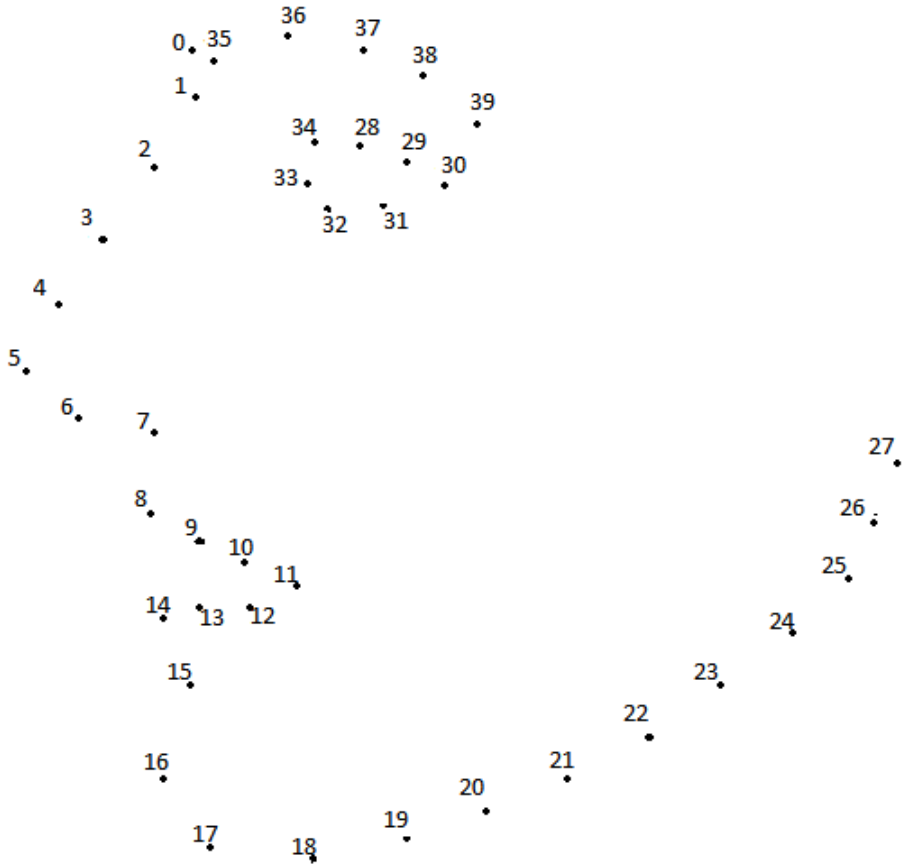
Předzpracování obličeje

Pro účel zarovnání obličeje byla použita konvoluční neuronová síť, která hledá klíčové body. Knihovna *dlib* zprostředkovává prostředí pro trénování tohoto typu sítě. Byla vytvořena sada 50 snímků, kde snímky tvořili množinu z hlediska identity. Rozložení klíčových bodů lze vyčíst z obrázku 3.5 níže.

Klíčové body byli vybírány tak, aby popisovali obličej na místech, které bývají nejméně zahalené a jsou lehce rozeznatelné. na každém obličeji z trénovací množiny bylo nejprve nutné všechny klíčové body ručně zaznamenat a uložit do textového souboru. Následné trénování bylo provedeno podle dokumentace[14].

Po nalezení obličeje a zjištění pozice všech klíčových bodů nacházejících se na specifickém obličeji, je nutné tento obličej nějakým způsobem zarovnat. Zarovnání je prováděno pomocí Procrustrově analýze, kde její princip je znázorněn na obrázku[15]:

[14]Převzato z: http://www.csc.kth.se/ vahidk/papers/KazemiCVPR14.pdf
[15]Převzato z: http://www.en.wikipedia.org/wiki/Procrustes_analysis

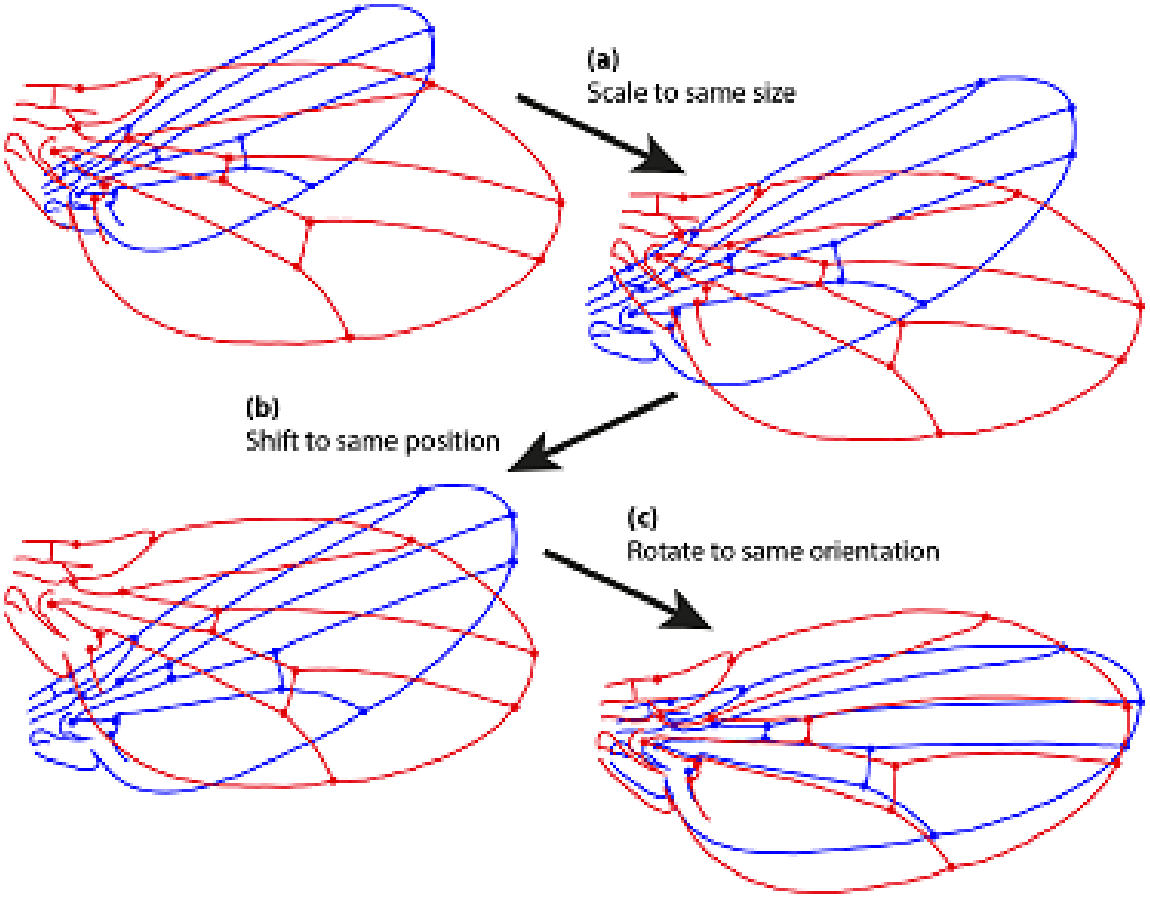


Obrázek 3.5: Klíčové body na obličeji z profilu.

Pro transformaci byl vyextrahována matice obsahující průměrnou pozici všech 40 klíčových bodů nacházejících se na obličejích z profilu. Tato matice je uložena v poli jako konstanta programu a pozice klíčových bodů byli přepočítány tak, aby odpovídali výslednému zarovnanému obličeji v rozměru 96 x 96 pixelů. Pro provedení následné transformace zarovnávaného obrázku na uloženou matici (operace jako zvětšení, zmenšení a natočení) hledáme tedy T, s a R minimální jako:

$$\sum_{i=1}^{40} \|sRp_i^T + T - q_i^T\|^2 \tag{3.1}$$

Kde R je ortogonální matice o velikosti 2 sloupců a řádků. Malé s značí skalár a T je dvou dimenzionální vektor. Řádky matic jsou poté reprezentovány pomocí p_i a q_i.



Obrázek 3.6: Zarovnání pomocí Procrustově analýze.

Trénování profilové sítě

Po úspěšném provedení sledu operací popsaných výše, získáme zarovnaný obličej z hlediska dvou-dimenzionální roviny, který je zároveň zmenšen, respektive zvětšen na rozměry 96x96 pixelů.

Jelikož je rozpoznávání obličejů z profilu oproti klasickému rozpoznávání obličejů ze předu velice málo rozšířené, tak nejsou k dispozici tak rozsáhlé datové sady. Datová sada byla vytvořena z datové sady FEI Face Database dostupné na[5]. Několik obličejů je i z datové sady LFW. Z obou databází byli vyseparovány pouze obličeje z profilu a to nehledě na směr otočení.

Pro potřeby zobrazení ROC křivky byla použita metoda x-křížové validace. Databáze byla tedy rozdělena do složek, kde ke každé složce náleží právě jedna identita. Celkový počet složek, respektive identit je 244 s tím, že minimální počet obličejů u každé identity je 1. Tato sada byla rozdělena na trénovací a validační a to tak, že se vyjmulo 20 identit, kde celkový počet obrázku na tento počet je 88. Tato množina se rozdělila na podmnožiny a to na 2 z důvodu malé sady (běžné rozdělení je na 10 složek). V každé podmnožině se tedy nacházelo 44 obrázků.

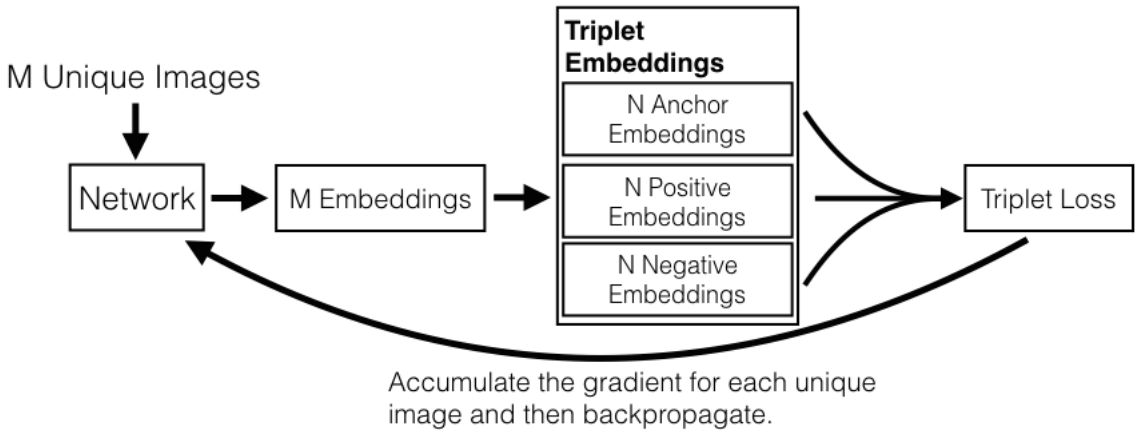
Snímky nejprve byli předzpracovány, kde prvním krokem bylo nalezení obličeje na každém snímku s tím, že detektor využívající Violův-Jonsonův algoritmus je trénován na obli-

[5] http://fei.edu.br/ cet/facedatabase.html

čejích natočených na levou stranu, bylo tedy nutné vyhledat obličeje na každém snímku a poté snímek zrcadlové překlopil a vyhledat obličeje znovu. Tímto postupem se docílilo toho, že byla nalezena většina obličejů skrze všechny snímky. Po identifikaci každého obličeje bylo nutné udělat jeho výřez, ten poté zarovnat a nakonec celý takto vytvořený snímek zvětšit, respektive zmenšit na velikost 96x96 pixelů, aby snímky mohli být předloženy jako vstup pro neuronovou síť.

Vzhledem k faktu, že je problém u trénování neuronových sítí nastavit parametry tak, aby síť nebyla přetrénována, ale zároveň aby byla schopná validovat testovací data s co nejmenší chybou.

Pomocí frameworku troch poté trénování probíhalo tak, že se nejprve nahrál model sítě do paměti a poté se náhodně inicializovali počáteční váhy. Síť poté počítá 128-dimenzionální vektor rysů nad 128-dimenzionální hypersférou a je optimalizována pomocí *triplet loss*, kde triplet je definován jako trojice a to: vektor získaný ze vstupního obrázku, jiný vektor téže osoby a vektor jiné osoby. Vektor ze vstupního obrázku by se poté měl co nejvíce přibližovat vektoru z jiného obrázku téže osoby a zároveň by měl být co nejdál od vektoru jiné osoby, kde vzdálenost je definována jako euklidovská. Kritická část v optimalizování *triplet loss* funkce je ve výběru vzorů, respektive obrázku pro každý triplet, tento výběr je zvolen tak, že se vždy náhodně vyberou dva obrázky každé osoby pro reprezentaci vektoru osoby a jiného vektoru stejné osoby. Tento princip je znázorněn na obrázku 3.7.

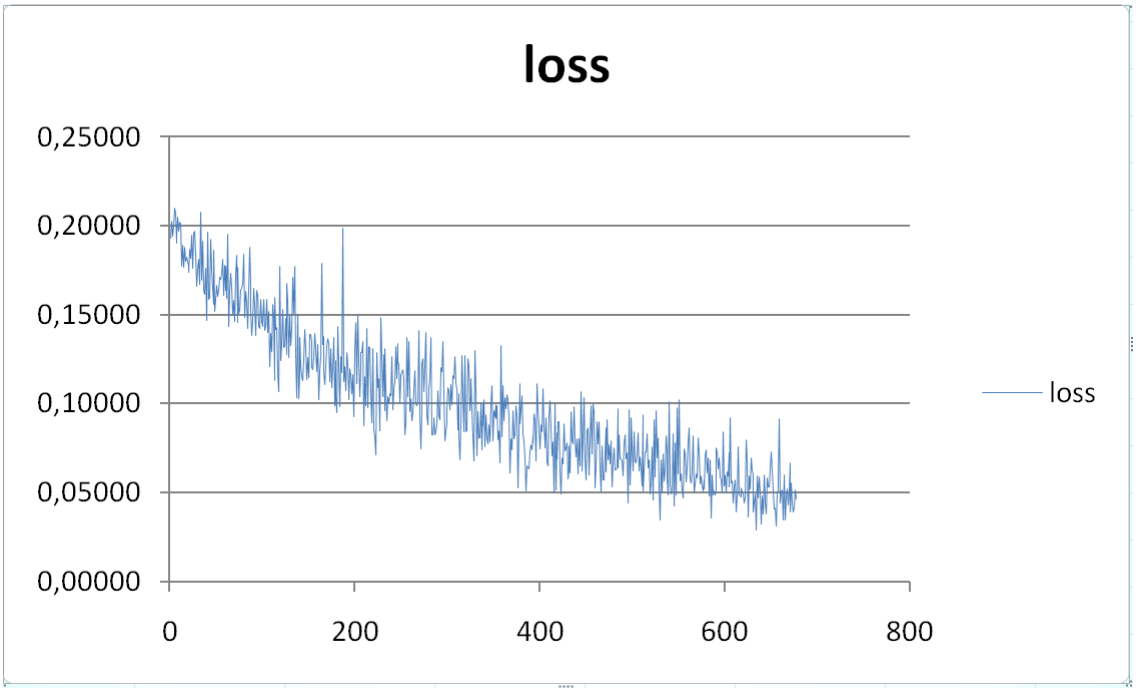


Obrázek 3.7: Ilustrace učení pomocí triplet loss.

Nejlepších výsledků dosahovala síť která byla trénovaná po dobu 400 epoch, kde epocha je definována jako přiložení všech trénovacích snímku na vstup trénované neuronové sítě. Při každém průchodu bylo poté najednou zpracováváno 4 snímku z 11 identit (omezení paměti grafické karty), po každém tomto průchodu se parametry sítě aktualizovali.

Na obrázku 3.8 je vidět průběžné snižování *loss*, kde se sice hodnotu i po 400 epochách stále snižovala, čímž by se dalo říci, že by síť měla mít lepší parametry, jevila se však jako přetrénovaná i na úkor trénování pomocí *triplet loss* funkce, která by této situace měla zabránit. Tento fakt byl nejspíše způsobem tím, že trénovací sada obsahuje z většiny snímky, které jsou si velice podobné z toho hlediska, že se osoby nacházejících se na těchto snímcích tváří neutrálně, obličej nemají ničím zakrytý a i úhlů natočení směrem ke kameře je jen několik. Byla tedy vytvořena mini sada, kde se zachycovali právě nestandardní výjevy z hlediska mimických projevů a různých natočení směrem ke kameře, kde bylo několik sítí

mezi sebou ručně porovnáno a jako nejlepší z tohoto testu výšla právě síť natrénovaná po 400 epochách.



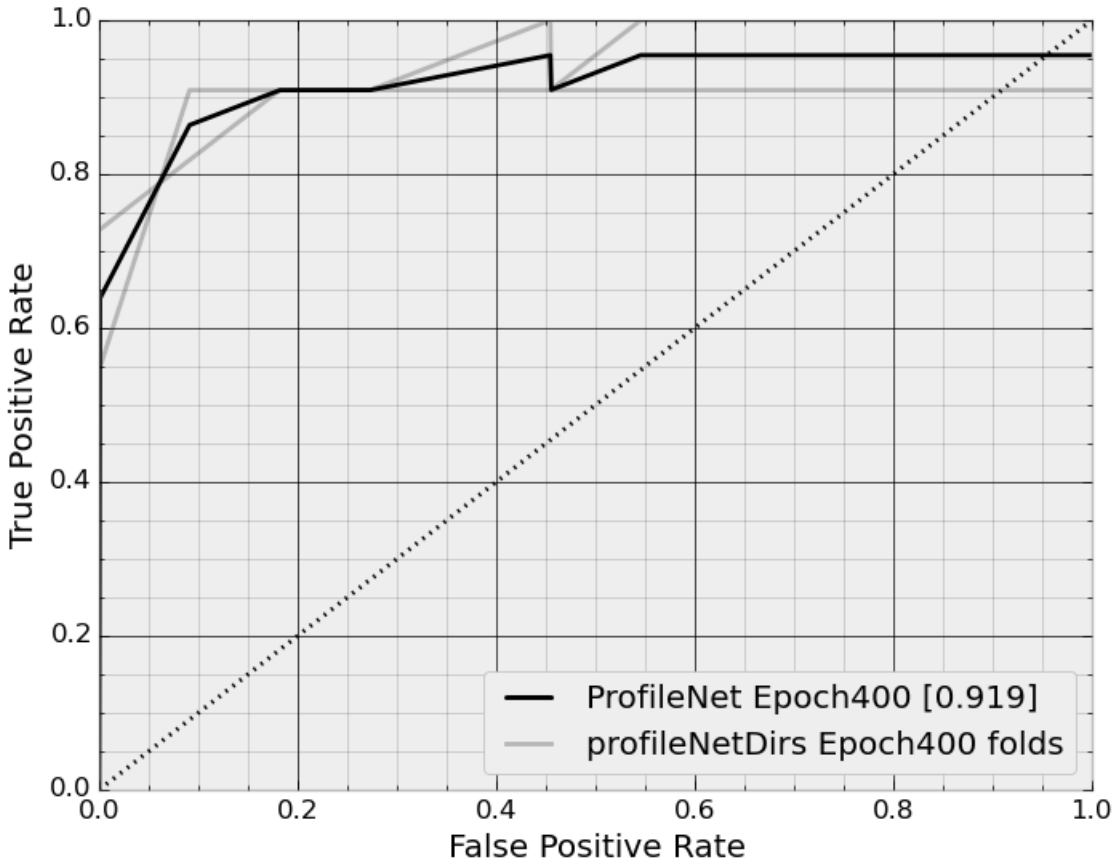
Obrázek 3.8: Zobrazení postupného se snižování loss funkce napříč epochami

Obrázek 3.9 poté zobrazuje výslednou ROC křivku vytvořenou nad daty vyčleněny z trénovací množiny právě pro tento účel. Jak lze y tvary křivky vyčíst, nejeví se příliš plynule, což je dáno tím, že testovací sada obsahovala pouze 22 dvojic obrázků stejné identity a 22 dvojic obrázku rozdílné identity. Výsledná přesnost nad touto minimalistickou testovácí sadou poté byla 84 procent, což se nejeví jako nejhorší výsledek, ale vzhledem k velikosti testovací a trénovací sady a z přihlédnutím k faktu, že variací obličejových póz není mnoho, tak může být tento výsledek zavádějící.

3.3.2 validace obličejů

Pro validaci vektorů jsou nejčastěji používány různé variace SVM algoritmu (variace jádra), používá se také algoritmus k-means a v neposlední řadě je využíváno algoritmu hledání lokálních centroidů, který je s uvedených algoritmů nejjednodušší.

Pokud chceme obličeje klasifikovat na základě jeho vektoru extrahovaného pomocí konvoluční neuronové sítě, musíme nejprve vytvořit databázi osob. Tato databáze je v práci implementována pomocí *NoSql* velice rozšířené databáze *MongoDB*, která je implementována pro jazyk Python s názvem *PyMongo*. Data v databázi jsou reprezentována jako dokumentově orientované ve formátu JSON. Hlavní výhodou této databáze je její jednoduchost z hlediska implementace a její vysoká rychlost. Data v rámci práce jsou v databázi uloženy jako dvojice jméno požadované osoby a její vektor rysů. Vektor pro klasifikace osob je pro databázi vytvořen tak, že je zaznamenáno několik snímků každé osoby, ze kterých se vytvoří shluk pomocí nástroje *DBSCAN*. Nástroj má dva parametry a to ϵ, který představuje rádius pro hledání sousedních bodů a parametr udávající minimální počet vektorů ve shluku (tedy minimální počet dostatečně blízkých vektorů, nad kterými je počítán cent-



Obrázek 3.9: ROC křivka

roid). Díky těmto operacím jsou vyfiltrovány vzdálené vektory. Z výsledného shluku je poté vytvořen průměr, který představuje výsledný vektor pro uložení do databáze.

Zjištění identity osoby poté probíhá tak, že každý vektor rysů obličeje nacházejícího se na záznamu je porovnán s vektory v databázi a je vybrán ten, který má nejmenší euklidovskou vzdálenost s klasifikovaným vektorem. Maximální hodnotu vzdálenosti představuje hodnota 4 a minimální, tedy identický vektor, představuje hodnota 0. V průběhu záznamu každé osoby se její vektor rysů mění a pro klasifikaci je poté vybírán vektor s nejmenší euklidovskou vzdálenosti od vektoru v databázi.

Kapitola 4

Výsledky a testování

Testovány byli jednotlivé neuronové sítě a to na statických obrázcích z velice rozšířené datové sady LFW[15] a poté byl systém jako celek testován na databázi z ústavu ÚPGM a na YouTube faces database [16], které obsahují záznamy známých osobností ze serveru youtube.com.

4.1 Datové sady

Zkušební datovou sadu tvoří upravené videozáznamy z chodby ústavu ÚPGM. Záznamy jsou z roku 2016 a jsou rozděleny do souborů, kde jejich celková délka tvoří 67 hodin videozáznamu. V rámci této práce byla vytvořena pouze malá datová sada, k ní však byla přidána výše zmíněná YouTube faces database.

Databáze z chodbu ústavu ÚPGM je vytvořena z X hodin záznamů, na kterých se nachází XX průchodů pocházejících od 56ti osob. A záznamy jsou upraveny na krátké videosekvence, kde byli vyseparovány oblasti videozáznamu, na které se nenacházel žádný obličej. Databáze je taktéž rozčleněna do N složen, kde N je číslo reprezentující počet osob. Každá osoba je uložena ve složce spolu s výřezem obrázku celé osoby a s přiloženým testovacím záznamem, kde pokud se nacházela nějaká další osoba, tak byla zakryta pomocí rozmazání.

Databáze vytvořena ze záznamů na serveru youtube.com je rovněž rozdělena do složek, kde každá složka reprezentuje jedno jméno. V každé složce je poté několik videozáznamů téže osoby zachycených pod různým úhlem a z jiné doby. Celkový počet osob je poté 47 a celkový počet videozáznamů je 1910.

Poslední databáze je složena ze statických snímků tvořící obličeje z profilu. Tato databáze byla vytvořena pomocí kombinace databáze LFW, kde byli vyseparovány pouze obličeje z profilu a kombinace databáze FEI, kde byli použity rovněž pouze obličeje natočené z profilu. Databáze je rozčleněna na databázi, kde jsou nezarovnané snímky obličejů a na databázi, která je dále rozdělena na testovací a trénovací sadu. Trénovací sada spolu s testovací sadou jsou navzájem disjunktní a obsahují zarovnané snímky obličeji natočené na levou stranu a snímky jsou také zmenšené na velikost 96x96 pixelů.

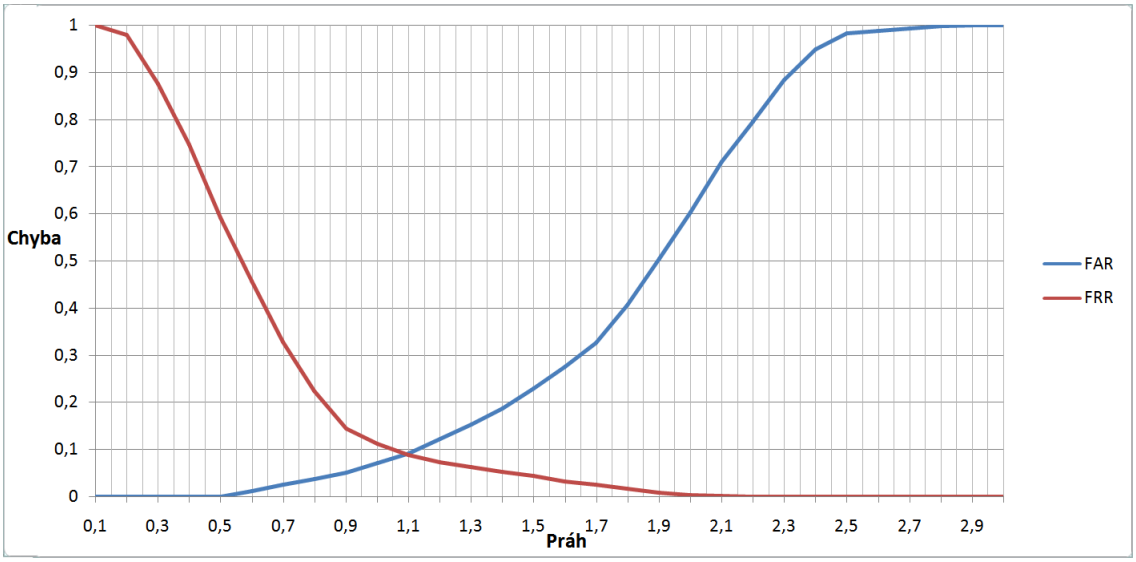
[15] LFW: http://vis-www.cs.umass.edu/lfw/

[16] Youtube faces database: https://www.cs.tau.ac.il/ wolf/ytfaces/

4.2 Verifikace

Jelikož je výstupem obou sítí 128dimenzionální vektor rysů, je vhodné provést verifikaci například pomocí křivek FRR/FAR.

Nejprve byl tento experiment proveden na neuronové síti, která je součástí knihovny *openface*, tedy jedná se o síť rozpoznávacích obličeje ze předu. Tento experiment byl proveden na statických snímcích databáze LFW a to tak, že se každý obrázek nejprve zarovnal a výřez obličeje transformoval na velikost 96x96 pixelů pro vstup do neuronové sítě.



Obrázek 4.1: FRR/FAR sítě pro obličeje natočené ke kameře přímo.

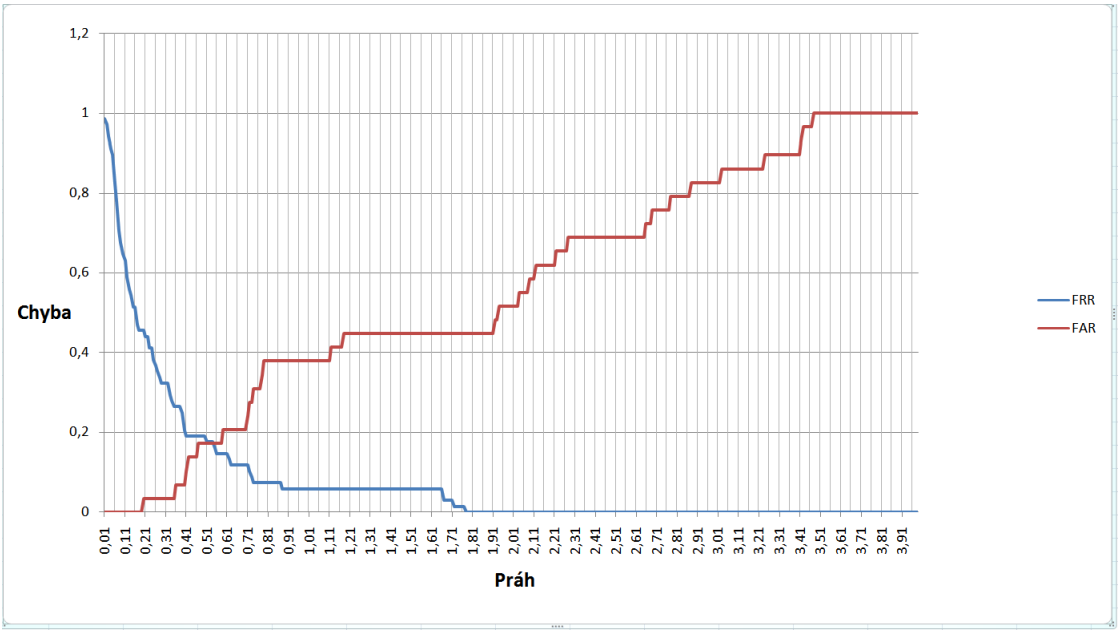
Jak lze z grafu vyčíst, tak při nastavení prahu na hodnotu 1,089 je úspěšnost sítě 92%. Tedy z celkového počtu bylo 8% vzorků chybně klasifikováno jako shoda a 8% vzorků bylo chybně klasifikováno jako rozdílné vzorky. Testování probíhalo na statických snímcích pocházejících z datové sady LFW.

Stejný experiment byl proveden i na neuronové síti, která byla na trénovaná na malém datasetu obsahující obličeje natočené z profilu. Jelikož je problém s množstvím snímků, experiment byl proveden pouze na 88 disjunktních obrázků obličeje, což jsou obrázky z testovací sady použité k průběžnému testování sítě při učení a k zobrazení ROC křivky. Křivky FRR a FAR zobrazuje obrázek 4.2.

Graf v porovnání s předešlým grafem 4.1 nevykazuje tam pěkné křivky, což je způsobeno velice malým počtem snímků. Jako práh se dá považovat hodnota 0,53 při které síť vykazuje přesnost 83%, tato přesnost však není směrodatná vzhledem k malému počtu testovacích vzorků.

4.3 Experimenty na videozáznamech

Hlavní nevýhodou videozáznamu oproti statickým snímkům obličeje je především v kvalitě jednotlivých obrázků, kde jsou obličeje ve videozáznamu často rozmazané a v nízkém rozlišení. Rozpoznávání identit z videozáznamu však přináší i výhody a to zejména fakt, že je každá osoba ve videozáznamu na více snímcích a pomocí sledování jsme pak schopni tyto snímky přiřadit jednotlivým osobám.



Obrázek 4.2: FRR/FAR profilové sítě

Testy probíhaly na datové sadě získané z ústavu ÚPGM a na databázi YouTube faces database. Test probíhal nejprve na prvním datasetu a to tak, že se nejprve prováděl test na jednotlivých snímcích, kde se nachází pouze jedna osoba, díky čemuž si systém vygeneroval databázi s příslušnými automaticky vygenerovanými jmény pro každou nalezenou osobu. Z celkového počtu 56ti osob 2 osoby nebyli nalezeny pomocí detektoru hledající obličeje ze předu a to z toho důvodu, že po celou dobu záznamu byli tyto osoby natočeny vzhledem ke kameře z profilu. Po zaznamenání všech osob následoval test na celém videozáznamu, kde byli označeny pozice osob, které poté byli porovnávány s log souborem vygenerovaným systémem. V testu se příliš neprojevila neuronová síť trénovaná na obličejích z profilu a to z toho důvodu, že detektor hledající tento typ obličejů reaguje u na pouze částečně natočené obličeje. Tento problém je řešen pomocí hledání pozice nosu vůči očím viz. 3.2.3. Nastávaly však situace, kdy při tomto částečném natočení nebyl obličej rozpoznán detektorem na obličeje ze předu a díky tomu se obličej, respektive jeho vektor přiřadil do pole s obličeji natočenými z profilu. Pokud bylo takto špatně přiřazených obličejů více, než obličejů opravdu z profilu, tak pomocí hledání lokálního centroidu, kdy se vybral největší shluk docházelo ke spočtení vektoru nad špatným shlukem, což vytvořilo velkou chybu při rozpoznávání téže osoby, kde byl její obličej vystaven před kamerou pouze z profilu.

Výsledkem z toho experimentu je tedy celková úspěšnost 79% s tím, že 5 osob nebylo detekováno a 12 osob bylo chybně klasifikováno.

Experiment probíhal podobně i na databázi YouTube faces database. Databáze je v podobě složek se jménem osoby, kde v každé složce je několik videí k dané osobě. Jelikož jsou videa osob různorodá a zachycují osoby v různých pózách a časových oblastech, byla naměřena úspěšnost pouze 68%.

Kapitola 5

Závěr

V rámci této práce byla nastudovaná problematika detekce osob, včetně metod k úpravě videozáznamu z hlediska extrakce pohybujícího se popředí vůči statickému pozadí videozáznamu. Dále byla nastudovaná problematika rozpoznávání osob podle obličeje a to za použití konvolučních neuronových sítí. V práci byly použity obě tyto sítě. První síť byla natrénovaná na obličejích natočených směrem ke kameře. Úspěšnost této sítě na datové sadě LFW je 92%. Druhá síť byla trénovaná na obličejích z profilu. Úspěšnost této sítě dosahovala hodnot 83% s tím, že test byl proveden pouze na malé datové sadě a obsahoval obličeje natočené z profilu vůči kameře.

Systém byl implementován v jazyce Python verze 2.7 a aplikace je vícevláknová. Databáze uzpůsobené pro ukládání vektorů reprezentující osoby je implementována pomocí NoSQL systému MongoDB. Vzhledem k možnosti ukládání osob za běhu aplikace, byl využit klasifikátor počítající *centroidy* nad shluky vektorů. Ty byly vytvořeny pomocí shlukové analýzy DBSCAN. Tato analýza při vhodně zvolených parametrech je schopna odfiltrovat vzdálené vektory a zvýšit tak přesnost klasifikátoru. Identifikace osoby z videozáznamu probíhá tak, že se ukládají všechny vektory získané z dané osoby při pobytu na videozáznamu. Nad těmito vektory je spočten centroid a poté pomocí euklidovské vzdálenosti je vybrána příslušná osoba z databáze.

Testy probíhaly na videozáznamech ze dvou databází, kde se celkem nacházelo 103 osob. Výsledná přesnost dosahovala 74% napříč oběma databázím. Díky použitím dvou konvolučních neuronových sítí spolu s příslušnými detektory je systém schopen zpracovat video s rozlišením 1920x1080 pixelů v počtu průměrně 6ti snímků za vteřinu. Pokud budeme provádět detekci ob snímek, dosáhneme rychlosti až 12ti snímků za vteřinu. Největší problém nastával při nalezení obličeje pomocí detektoru hledající obličeje natočené z profilu. Nenalezením téhož místa pomocí detektoru hledajícího obličeje natočené ze předu, kdy obličej byl pouze částečně natočen z profilu. Díky tomuto jevu byl obličej zpracován neuronovou sítí určenou pro obličeje z profilu, čímž nastávala velká chybovost.

Rozpoznávání obličejů je stále otevřeným problémem, jelikož neexistuje systém, který by byl schopen osoby z videozáznamu rozpoznávat s přesností okolo 100%. Pro vyšší úspěšnost by bylo tedy vhodné kombinovat více různých sítí, které by rozpoznávály například obličej natočen ke kameře, obličej natočen z profilu a síť která by byla schopna rozpoznávat celou postavu. Pro další rozvoj systému by tedy bylo vhodné vylepšit detektory hledající obličeje a přidat uvedené sítě, které by byli trénovány nad velkou datovou sadou.

Literatura

[1] K Nearest Neighbors algorithm. ročník 2008: str. 33.
URL http://www.lkozma.net/knn2.pdf

[2] Viola–Jones object detection framework. 2001-.
URL https:
//en.wikipedia.org/wiki/Viola%E2%80%93Jones_object_detection_framework

[3] Feature selection and nearest centroid classification for protein mass spectrometry. In *US National Library of Medicine*, University of Alberta: BioMed Central, 2005, str. 1.

[4] Convolutional Neural Networks (CNNs): An Illustrated Explanation. 2016.
URL http://xrds.acm.org/blog/2016/06/convolutional-neural-networks-cnns-illustrated-explanation/

[5] Amos, B.; Ludwiczuk, B.; Satyanarayanan, M.: OpenFace: A general-purpose face recognition library with mobile applications. Technická zpráva, CMU-CS-16-118, CMU School of Computer Science, 2016.

[6] Bishop, C. M.: *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006, ISBN 0387310738.

[7] Chintala, S.: An Intuitive Explanation of Convolutional Neural Networks.
URL https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

[8] Dalal, N.; Triggs, B.: Histograms of Oriented Gradients for Human Detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, ročník 1, 2005, ISSN 10636919, s. 886–893 vol. 1, doi:10.1109/CVPR.2005.177.

[9] Danelljan, M.; Häger, G.; Shahbaz Khan, F.; aj.: Accurate Scale Estimation for Robust Visual Tracking. In *Proceedings of the British Machine Vision Conference*, BMVA Press, 2014, doi:http://dx.doi.org/10.5244/C.28.65.

[10] Doc. Ing. František Vítězslav Zbořil, C.: Biologický a umělý neuron,umělé neuronové sítě. Perceptron a Adaline. ročník 2016.

[11] Godbehere, A. B.; Matsukawa, A.; Goldberg, K.: In *2012 American Control Conference (ACC)*, 2012, ISSN 07431619, s. 4305–4312, doi:10.1109/ACC.2012.6315174.

[12] Houdek, M.; Svoboda, T.; Procházka, T.: Klasifikace podle nejbližších sousedů. In *Klasifikace podle nejbližších sousedů*, 2011, str. 12.

[13] Ing. Michal Hradiš, P.: Deep CNN for Computer Vision.

[14] Kaewtrakulpong, P.; Bowden, R.: An Improved Adaptive Background Mixture Model for Realtime Tracking with Shadow Detection. 2001: s. 2–5, doi:10.1007/978-1-4615-0913-4_11.
URL https://www.researchgate.net/publication/2557021_An_Improved_Adaptive_Background_Mixture_Model_for_Realtime_Tracking_with_Sh

[15] Lawrence, S.; Giles, C. L.; Tsoi, A. C.; aj.: Face recognition: A convolutional neural-network approach. *IEEE Transactions on Neural Networks*, ročník 8, č. 1, 1997: s. 98–113, ISSN 10459227, doi:10.1109/72.554195.

[16] Muller, K. R.; Mika, S.; Ratsch, G.; aj.: *IEEE Transactions on Neural Networks*, ročník 12, č. 2, 2001: s. 181–201, ISSN 10459227, doi:10.1109/72.914517.

[17] Ortega, J. P.; Pires, C. E. S.; Marinho, L. B.; aj.: Early Classification: A New Heuristic to Improve the Classification Step of K-Means. In *XXVII Simpósio Brasileiro de Banco de Dados - Short Papers, São Paulo, São Paulo, Brasil, October 15-18, 2012.*, 2012, s. 185–192.
URL http://www.lbd.dcc.ufmg.br/colecoes/sbbd/2012/0024.pdf

[18] Rak, R.; Matyáš, V.; Říha, Z.: *Biometrie a identita člověka*. Praha: Grada, první vydání, 2008, ISBN 978-80-247-2365-5.

[19] Schroff, F.; Kalenichenko, D.; Philbin, J.: Facenet: A unified embedding for face recognition and clustering. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, ISSN 10636919, s. 815–823, doi:10.1109/CVPR.2015.7298682.

[20] Stauffer, C.; Grimson, W. E. L.: Adaptive Background Mixture Models for Real-Time Tracking. In *CVPR*, IEEE Computer Society, 1999, ISBN 0-7695-0149-4, s. 2246–2252.
URL http://dblp.uni-trier.de/db/conf/cvpr/cvpr1999.html#StaufferG99

[21] Taigman, Y.; Yang, M.; Ranzato, M.; aj.: DeepFace: Closing the Gap to Human-Level Performance in Face Verification. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[22] Viola, P.; Jones, M.: In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, ročník 1, 2001, ISSN 10636919, s. I–511–I–518 vol.1, doi:10.1109/CVPR.2001.990517.

[23] Volná, E.: *Neuronové sítě a genetické algoritmy*. Ostrava: Učební texty Ostravské univerzity, první vydání, 1998, ISBN 8070427620.

[24] Zhu, X.; Ren, D.; Jing, Z.; aj.: In *Proceedings of 2012 2nd International Conference on Computer Science and Network Technology*, 2012, s. 1528–1533, doi:10.1109/ICCSNT.2012.6526210.

Příloha A

Obsah CD

- /profile_train/ - data pro trénování neuronové sítě
- /src/ - zdrojové kódy aplikace
- /examples/ - vyexportované video-soubory reprezentující funkčnost systému
- /poster/ - plakát
- /data/ - testovací databáze
- /text/ - textová zpráva se zdrojovými soubory

Příloha B

Plakát