VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

LOKALIZACE OBLIČEJE POMOCÍ NEURONOVÉ SÍTĚ

NEURAL NETWORK BASED FACE LOCALIZATION

DIPLOMOVÁ PRÁCE
DIPLOMA THESIS

AUTOR PRÁCE Bc. PAVEL HENDRYCH
AUTHOR

VEDOUCÍ PRÁCE Ing. MIROSLAV ŠVUB
SUPERVISOR

BRNO 2008

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2007/2008

Zadání diplomové práce

Řešitel: **Hendrych Pavel, Bc.**
Obor: Inteligentní systémy
Téma: **Lokalizace obličeje pomocí neuronové sítě**
Kategorie: Počítačová grafika

Pokyny:

1. Prostudujte základy zpracování obrazu. Zaměřte se zejména na problematiku detekce lidského obličeje v obraze.
2. Prostudujte dostupné materiály na téma neuronové sítě a jejich využití ve zpracování obrazu.
3. Vyberte vhodnou metodu a navrhněte postup při lokalizaci obličeje pomocí neuronové sítě.
4. Experimentujte s vaší implementací a případně navrhněte vlastní modifikace metod.
5. Porovnejte dosažené výsledky a diskutujte možnosti budoucího vývoje.
6. Vytvořte stručný plakát prezentující vaši diplomovou práci, její cíle a výsledky.

Literatura:

- Dle pokynů vedoucího.

Při obhajobě semestrální části diplomového projektu je požadováno:

- Bez požadavků.

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 ročníkového a semestrálního projektu (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, DVR-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Švub Miroslav, Ing.**, UPGM FIT VUT
Datum zadání: 5. února 2007
Datum odevzdání: 19. května 2008

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

L.S.

doc. Dr. Ing. Pavel Zemčík
vedoucí ústavu

Abstrakt

Tato práce se teoreticky i prakticky zabývá současnými přístupy k detekci a lokalizaci obličeje v obraze. Popisuje možné přístupy k řešení tohoto problému, detailněji se zabývá lokalizací pomocí neuronových sítí a operacemi před vlastní detekcí a následné správné reprezentaci výsledků. Obsahuje implementaci několika přístupů k lokalizaci obličeje pomocí neuronových sítí, především pak přístupu založeného na vlastních tvářích. Součástí je i jednoduchá implementace klasifikátoru založeného na vzdálenosti tváří zrekonstruovaných pomocí množiny vlastních tváří. Detailně je popsána implementace tohoto systému, dosažené výsledky a závislost výkonnosti systému na jeho parametrech.

Klíčová slova

Počítačové vidění, lokalizace obličeje, neuronová síť, PCA, obličejová databáze, vlastní tváře, klasifikace na základě vzdálenosti, FANN

Abstract

This thesis issues with possible methods for face detection and localization according to the state of the art. It describes various approaches and it is aimed at localization by neural networks and at necessary operations that have to be done before localization and after that for correct results representation. This project contains implementation of few approaches to neural netwok based face localization with emphasis on eigenfaces based face localization as well as implementation of simple classifier using distance of reconstructed face to the original one. Detailed description of implemented system, achieved results and dependecy of system performance on it's inner settings is also provided.

Keywords

Machine Vision, face localization, neural network, PCA, principal component analysis, face database, eigenface, distance based classification, FANN

Bibliografická citace:

Pavel Hendrych, Lokalizace obličeje v obraze pomocí neuronových sítí, diplomová práce, Brno, FIT VUT v Brně, 2008.

Prohlašuji, že jsem tento diplomový projekt vypracoval samostatně pod vedením Ing. Miroslava Švuba. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Brně dne 19. května 2008 Bc. Pavel Hendrych

Na tomto místě bych rád poděkoval vedoucímu projektu Ing. Švubovi za cenné připomínky a odborné vedení. Poděkování patří též Ing. Grézlovi za rady a pomoc při experimentech s neuronovými sítěmi.

Uznání a dík bych rád projevil i autorům softwarových knihoven použitých v rámci této práce, konkrétně knihoven GSL a FANN.

Obsah

1 Úvod 6
1.1 Neuronové sítě 7
2 Vybrané metody zpracování obrazu pro potřeby detekce obličeje 10
2.2 Současné přístupy k lokalizaci obličeje 17
3 Návrh systému pro lokalizaci obličeje 23
3.1 Motivace 23
3.2 Předzpracování vstupních dat 23
3.3 Návrh klasifikátoru 24
3.4 Zpracování výsledků 25
4 Implementace systému pro lokalizaci obličeje 27
4.1 Základní informace 27
4.2 Lokalizace obličeje pomocí neuronových sítí 30
4.3 Trénovaní neuronové sítě 30
5 Vlastní tváře 42
5.2 Lokalizace na základě vzdálenosti 42
5.3 Klasifikace pomocí neuronových sítí s použitím vlastních tváří 47
5.4 Lokalizace pomocí rekonstruovaných tváří 52
6 Eliminace falešných detekcí 57
7 Závěr 58
8 Použitá literatura a odkazy 59
9 Přílohy 60

1 Úvod

Původně byly počítače vytvořeny jako stroje pro automatizované zpracování dat. Během své existence prošly velmi rychlým vývojem a již po mnoho let nejsou jen užitečnými kancelářskými pomocníky, ale přidáváním nových, důmyslných funkcí se z nich vyvinuly velmi vážní konkurenti lidské pracovní síly. V některých oblastech dokonce člověka porážejí. Není to způsobeno pouze nárůstem výkonu, ale také důmyslnějším procesem zpracování dat, od jejich pořízení do vykonání příslušné akce. Zdrojem dat již není jen klávesnice, data mohou být pořízena z nejrůznějších senzorů, jako jsou například pohybová čidla, sonary atd.
V posledních několika letech zaznamenaly značný vývoj kamery. Již v 70. letech 20. století, kdy byly počítače schopné zpracovávat 2D obrazy, se začal vyvíjet nový obor – Počítačové vidění (angl. Computer Vision), avšak dnes se v souvislosti s průmyslovými roboty vybavenými různými senzory používá pojem Machine Vision. Computer vision se zaměřuje spíše na rekonstrukci 3D informace z kamer (avšak tyto obory se značně překrývají). Je tedy přirozená snaha využít možnosti počítačových systémů o tyto optické senzory. Naneštěstí jsou takto získaná data dvourozměrná a velmi často zatížená šumem. To klade velký důraz na následné zpracování získaného obrazu, kde už si s klasickými přístupy často nevystačíme.

Tato práce se zabývá jednou z oblastí zpracování obrazu, konkrétně lokalizací obličeje ve snímku. První teoretická část rozebírá základní pojmy z oblasti umělé inteligence a počítačového vidění a vytváří podklad pro části navazující. Shrnuje současný stav metod, které je možno pro lokalizaci použít. Navazuje návrh klasifikátoru pro detekci a lokalizaci obličeje pomocí neuronových sítí. V druhé polovině práce je pak popsán implementovaný systém, který je poměrně komplexní a umožňuje srovnání výsledků několika různých přístupů. Rozebrány jsou jeho základní parametry a dosažené výsledky. Přiloženo je též větší množství grafů popisující výstup a výkonnost systému a jejich rozbor.

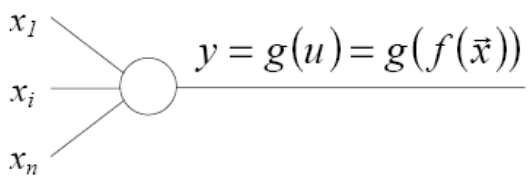
Práce je ukončena sumarizací poznatků a nástinem možného rozšíření implementovaného systému.

1.1 Neuronové sítě

Umělé neuronové sítě patří do oblasti umělé inteligence zvané soft-computing. Ten, na rozdíl od konvenčních algoritmů, toleruje data, která mohou obsahovat nepřesnosti, částečnou pravdivost či aproximace, za účelem dosažení řešitelnosti daných problémů. Neuronové sítě jsou jedním z výpočetních modelů soft-computingu, stejně jako fuzzy logika, evoluční algoritmy a teorie chaosu. Model může být implementován jak softwarově, tak i hardwarově. Výhodou těchto přístupů je, že při vhodné aplikaci na některé problémy poskytnou řešení rychleji, byť někdy jen částečné, a to právě v případech, kdy klasické algoritmy selhávají zcela. Důvodem selhání konvenčních algoritmů je většinou jejich přílišná složitost a z toho plynoucí nedostatek paměti nebo času.

1.1.1 Umělý neuron

Inspirací pro vznik umělých neuronových sítí byly znalosti z biologie. Matematicky byl poprvé umělý neuron popsán v roce 1943, ovšem princip činnosti byl znám již z konce 19. století. Činnost umělého neuronu je podobná jako v případě biologického, pouze je abstrahováno od pro funkci umělého neuronu nepodstatných vlastností, např. synapse je modelována číslem představujícím váhu, nikoliv množstvím proteinů, iontů a nervových přenašečů, jako je tomu u biologického neuronu. Nejjednodušším modelem neuronu je perceptron, poprvé představeným Frankem Rosenblattem v roce 1958. Perceptron má následující strukturu:



Obrázek 1: Schéma perceptronu

Vstupy neuronu jsou označeny x_i, váha vstupu x_i je w_i. Funkce f(**x**) se nazývá bázová funkce. Lineární bázová funkce je definována rovnicí 1, radiální bázová funkce má tvar dle rovnice 2. Samozřejmě lze použít i jiné typy bázových funkcí, ale s lineární a radiální se můžeme setkat nejčastěji.

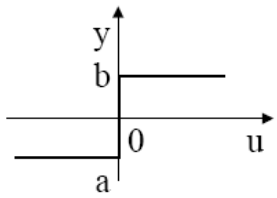
$$u = \sum_{i=i}^{N} w_i x_i \tag{1}$$

$$u = \left\| \vec{x} - \vec{w} \right\| = \sqrt{\sum_{i=i}^{N} (x_i - w_i)^2} \qquad (2)$$

Výstup neuronu je definován aktivační funkcí g(u) = g(f(**x**)). Obrázek 2 znázorňuje některé z používaných aktivačních funkcí. Aktivační funkce může být bipolární, v tom případě nabývá hodnot {–1,1}, nebo binární, jejímž výstupem jsou {0,1}, a v případě spojitých funkcí jsou pak výstupem hodnoty z intervalu <-1,1>, respektive <0,1>.

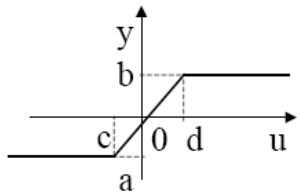
Skoková aktivační funkce

$$y^{new} = \begin{cases} a & pro\ u < \Theta \\ b & pro\ u > \Theta \\ y^{old} & pro\ u = \Theta \end{cases}$$



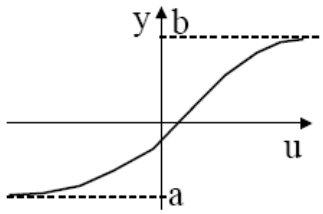
Po částech lineární aktivační funkce

$$y^{new} = \begin{cases} a & pro\ u < c \\ b & pro\ u > d \\ a + \dfrac{(b-a)(u-c)}{d-c} & pro\ c \le u \le d \end{cases}$$



Sigmoidální funkce

$$y = a + \frac{b-a}{1+e^{-\lambda u + c}}$$



Obrázek 2: Definice a průběhy často používaných aktivačních funkcí

1.1.2 Učení perceptronu

Základní pravidlo učení perceptronu má tvar dle rovnice 3. Výstup neuronu je označen y, d je požadovaná hodnota výstupu pro vektor **x**. Koeficient učení p má nejčastěji hodnotu 0,5, nebo může být adaptivně snižován během cyklu učení. Proměnná k značí k-tý vstupní vektor.

$$\vec{w}(1) = random$$
$$\vec{w}(k+1) = \vec{w}(k) + p(d(k) - y(k))\vec{x}(k) \tag{3}$$
$$pro\ k = 1,2..N, \quad (\vec{x}(k), d(k)) \in T$$

Perceptron dokáže klasifikovat vstupní obrazy do dvou tříd. Tyto třídy musejí být lineárně oddělitelné, tj. v prostoru obrazů musí existovat nadrovina, která obrazy dokáže rozdělit. Nadrovina je útvar v n-rozměrném prostoru o dimenzi n-1. Dá se matematicky dokázat, že buď tato nadrovina neexistuje, nebo existuje nekonečně mnoho dělících nadrovin. V případě radiálních bázových funkcí je pak dělícím útvarem hypersféra. Při vhodně zvoleném koeficientu učení p vstupní váhy neuronu konvergují k jednomu z řešení.

1.1.3 Zapojování neuronů do sítí

Neuron ve smyslu samostatné výpočetní jednotky nepřináší žádná praktická zlepšení výpočtu. Toho však dosahujeme spojováním většího počtu neuronů do struktur zvaných neuronové sítě. Síť dovoluje modelovat mnohem složitější separační oblasti. Dělícím útvarem není nadrovina (hypersféra), ale několik nadrovin (hypersfér). Každá síť se skládá z několika vrstev. Jde o vrstvu vstupní (ovšem i v odborné literatuře neexistuje shoda, zda se má díky své jednoduchosti také nazývat vrstvou), několik vrstev skrytých, které nejsou viditelné uživatelem, a vrstva výstupní, která slouží k prezentaci výsledků. Neuronové sítě lze klasifikovat podle několika různých hledisek, např. podle stupně propojení, směru propojení mezi jednotlivými vrstvami apod. Pro každý typ sítě lze najít specifickou oblast použití a hlavně odpovídající algoritmus učení. Typickým příkladem učícího se algoritmu je zpětné šíření chyby (angl. Back Propatation, BP), navržený pro stejně pojmenovanou dopřednou plně propojenou neuronovou síť. Jde o učení s učitelem, které je odvozeno z myšlenky minimalizace energetické funkce definované rovnicí 4, podoba vzorce pro učení má pak tvar dle rovnice 5.

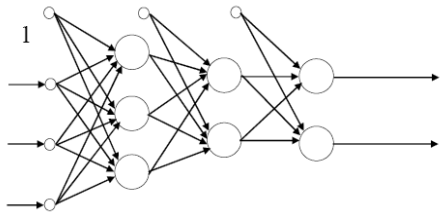
$$E = \frac{1}{2}\sum_{i=1}^{N}(y_i - d_i) \tag{4}$$

$$\Delta^l w_i = \mu^l \ \delta_j^l x_i \tag{5}$$

kde w_i je váha i-tého neuronu, μ je koeficient učení a ${}^l\delta_j = -\frac{\partial E}{\partial\ {}^l u_j}$

Možnost nasazení neuronových sítí je poměrně široká v oblastech spadajících do soft-computingu. Lze je použít jako asociační paměti, v různých oblastech rozpoznávání obrazů, či

například shlukování. Možnosti nasazení neuronových sítí v jedné z oblastí rozpoznávání – detekci lidského obličeje ve fotografiích, jsou popsány v následující kapitole. Neuronové sítě jsou použity jako klasifikátory, které klasifikují vstupní data do tříd obličej/neobličej. Neuronové sítě jsou však pouze jednou z možností, jak vytvořit klasifikátor, nikoli však jedinou. Alternativou k neuronovým sítím je například Support Vector Machines, které jsou krátce zmíněny dále.



Obrázek 3: Dopředná neuronová síť se dvěmi skrytými vrstvami

2 Vybrané metody zpracování obrazu pro potřeby detekce obličeje

Jak jsem již uvedl v úvodu této práce, pojem „počítačové vidění" je velmi široký. V této práci se budu zabývat počítačovým viděním z pohledu zpracování obrazu jako dvourozměrného signálu a následným procesem rozpoznávání.

Operace s obrazy vyžadují specifické modely zpracování. Nelze použít postupy známé z klasického zpracování informací, kde data jsou získána jako jednoznačné hodnoty ze senzorů nebo vstupem z klávesnice od uživatele. Před vlastním zpracováním je nutno provést některé kroky pro transformaci vstupního obrazu, jako je například filtrace. V případě rozpoznávání nebo jakéhokoli zpracování obrazu obličeje je velmi důležitou částí sytému detekce místa v obraze, kde se obličej nachází. Detekce je velmi náročný úkol, protože vstupní obrazy vykazují značnou variabilitu např. ve výrazu obličeje, velikosti, orientaci apod. Detekce pracuje s libovolným obrazem. Cílem detekce je určit, zda se v obraze nacházejí nebo nenacházejí nějaké obličeje. Cílem lokalizace je pak výsledky detektoru správně prezentovat a určit pozici obličeje (obličejů). V některých případech může být žádaná též orientace obličeje.

Variabilita vstupních dat způsobuje jisté problémy, které lze rozdělit do několika kategorií. Vlastní rozdělení je převzato z [3]. Následující aspekty mají značný vliv na výkonnost detektoru a při návrhu detektoru je potřeba na ně brát ohled.

- Pozice vzhledem ke kameře: Teoreticky může být obličej natočen ve všech třech osách relativně k pozici kamery. Hlava na snímku může být předkloněna nebo zakloněna, ukloněna na obě strany nebo otočena podle středy obličeje. To může způsobovat absenci některých obličejových rysů na snímku např. očí, což může činit některé metody detekce nepoužitelnými.
- Variabilita obličejových rysů: Může se jednat například o presenci či absenci brýlí nebo vousů. V případě vousů, kníru a vlasů je potřeba vzít v úvahu častou změnu tvaru, délky a v případě vlasů též barvu.
- Výraz obličeje: Změna nálady může vyvolat změnu tvaru některých obličejových částí, především úst a oblasti kolem očí.
- Částečné zakrytí: Některé části obličeje mohou být zakryty jinými objekty, v případě skupinové fotografie též jinými obličeji.
- Podmínky při snímání obrazu: Obraz obličeje ve snímku je ovlivněn různými světelnými podmínkami (intenzita, směr světla atd.) nebo nastavením objektivu kamery (ohnisková vzdálenost, polarizační filtr atd.)

Výčet faktorů, které mohou daný obraz ovlivnit, je poměrně široký a je potřeba před návrhem systému vyšetřit, které z nich mohou nastat, a zvolit vhodnou metodu pro detekci. Používané metody detekce jsou popsány dále. Přesto je vhodné před vlastní detekcí provést předzpracování obrazu, kterým zajistíme maximální uniformitu zpracovávaného obrazu. Do předzpracování lze zahrnout následující metody:

- Ekvalizace histogramu: Ekvalizací myslíme úpravu kontrastu ve snímku nebo jeho části tak, aby nebyla změněna globální intenzita celého snímku.
- Normalizace: Hodnoty barev jsou převedeny do intervalu <0,1>, případně <-1,1>. Několik normalizačních metod je představeno dále.
- Změna měřítka: Snímek je zmenšen či zmenšován tak, aby jeho velikost přesně vyhovovala požadavkům použité klasifikační metody.
- Filtrace obrazu: K rozpoznávání je použita pouze část frekvenčního spektra snímku.

Další fází zpracování obrazu je vlastní detekce obličeje. Souhrn metod použitelných pro detekci je popsán dále. Výsledek těchto metod však nemusí vždy odpovídat požadavkům, proto je často nutné výsledky detekce vhodně prezentovat. Možností, jak správně prezentovat dosažené výsledky, může být například eliminace duplikátních nálezů.

2.1.1 Ekvalizace histogramu

Jedná se o úpravu kontrastu barev ve vstupním snímku. Používá se především pro obrazy reprezentované stupni šedi, ale je možno tuto techniku použít i na barevné obrazy. Pomocí rozložení barev v histogramu dokáže lépe distribuovat frekventované hodnoty intenzit, čímž lze dosáhnou zvýšení kontrastu v málo kontrastních oblastech, aniž by došlo ke změně globálního kontrastu vstupního snímku. Je tedy velmi vhodné ji použít v případech, kdy barva pozadí i barva objektu v popředí je v obou případech velmi světlá, nebo velmi tmavá. Dokáže tedy zlepšit kvalitu přeexponovaných nebo podexponovaných snímků v důsledku změny světelných podmínek nebo nesprávného nastavení kamery. Pro snímek ve stupních šedi je pravděpodobnost výskytu bodu s hodnou šedi definována jako

$$p(x_i) = \frac{n_i}{N}\ ,\quad i=0\ ..\ L-1 \tag{6}$$

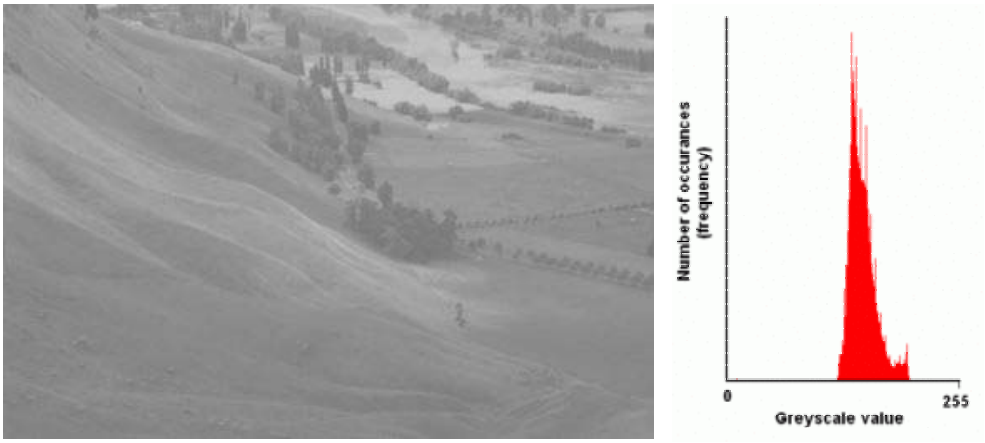
kde L je počet stupňů šedi v obraze a n je celkový počet bodů obrazu. Kumulativní pravděpodobnostní funkce je pak definována jako

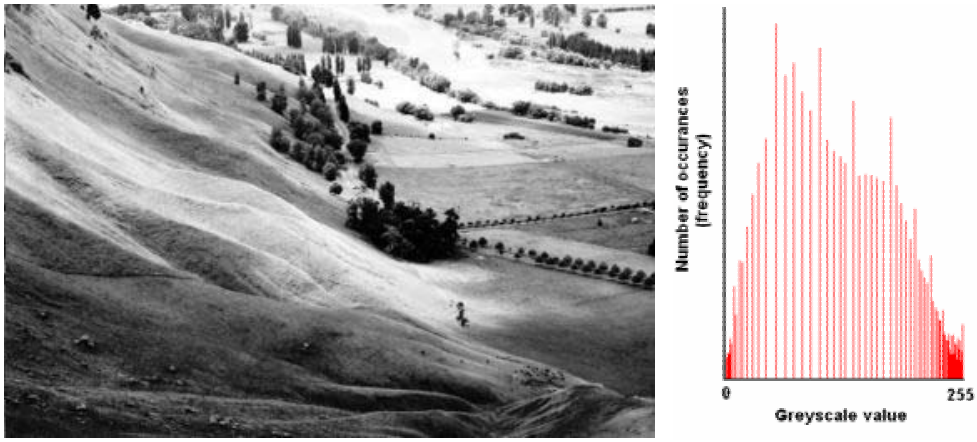
$$c(i) = \sum_{j=0}^{i} p(x_j) \tag{7}$$

kde **c** je normalizovaný histogram obrazu. Ekvalizace histogramu je pak definována následujícím způsobem

$$y_i' = y_i(\max - \min) + \min \tag{8}$$

V případě potřeby ekvalizovat barevné snímky, zle ekvalizaci provést zvlášť pro všechny barevné složky.





Obrázek 4: Ukázka ekvalizovaného a nekvalizovaného snímku, včetně histogramu. Převzato z http://en.wikipedia.org/wiki/Histogram_equalization

2.1.2 Výběr reprezentace dat

Pro správnou funkci systému je potřeba zajistit, aby vstupní data byla vhodně reprezentována. Touto reprezentací máme na mysli výběr vhodného datového typu a případně omezení rozsahu tohoto datového typu vstupního obrazu. Nejprve je nutno omezit počet barev, kterými je obraz vzorkován. Pro většinu aplikací je obecným doporučením používat obraz ve stupních šedi. Barevný obraz často nepřináší zásadní přínos informací potřebných ke správné klasifikaci, ba právě naopak, celý proces významně komplikuje. Klasifikátor pak musí zpracovávat zbytečné informace a časová složitost vlastní klasifikace by velmi rychle rostla. Samozřejmě by rostla též prostorová složitost, i když zde není nárůst tak strmý. Vzhledem k výše uvedeným skutečnostem použijeme tedy obrazy ve stupních šedi.

I zde však zůstává problém nalezení správného rozsahu. Lidské oko je schopno rozeznat přibližně 64 odstínů šedi, šedotónové obrazy jsou většinou kódovány na 256 stupních šedi. Pro další zpracování je nutno zvolit tedy univerzální rozsah. Neuronové sítě a ostatně i jiné klasifikátory pracují většinou nativně v rozsahu <-1,1>, případně <0,1>. Vhodný rozsah vzhledem k použitému klasifikátoru bude upřesněn v implementační části a bude vybrán jako nativní pro daný klasifikátor či po provedení testů jako výhodnější z výše uvedených. Dále budeme uvažovat rozsah <0,1>, převod na druhý uvedený je možno provést jednoduchými úpravami.

Změnu rozsahu je možno provést dělením největším číslem, které je možno na zdrojovém datovém typu zobrazit. Další z možností je normalizace vektoru. Ta se provede podle rovnice 9. Výhodou tohoto postupu je změna rozsahu, ale i další důležitá vlastnost – velikost vektoru vstupního obrazu je 1. Tento postup budeme dále používat jako výchozí.

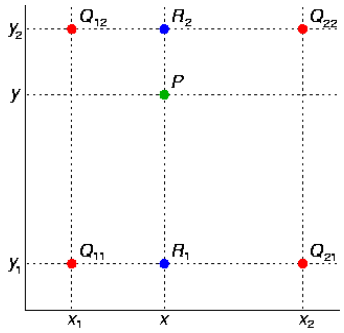
$$a' = \frac{a}{\|a\|} \tag{9}$$

kde a je vstupní vektor a a' vektor normalizovaný.

2.1.3 Změna měřítka

Vzhledem k tomu, že většina metod je navržena pro detekci obličeje určitých rozměrů, je nutno postupně zmenšovat vstupní snímek. Obličej je pak hledán na snímku v několika měřítkách, aniž by různá absolutní velikost obličeje vyžadovala různá nastavení systému. Tak je to prezentováno například v [2].

Naneštěstí postupné vynechávání řádků či sloupců obrazu vede ke značné degradaci kvality obrazu. Proto je nutno použít jiné techniky, které ve výsledném zmenšeném obraze interpolují barvy původních, vynechaných bodů. Barvy bodů, které mají v obraze zůstat, se změní na základě váženého průměru z okolních vynechaných bodů. Váhou může být vzdálenost středu bodů od přesně vypočítaného místa, kde se má vzorkovat. Barvu bodu P pokud známe barvu bodů Q_{ij} vypočítáme např. dle postupu uvedeného na v rovnici 10. Nejjednodušší interpolací je bilineární, složitější pak bikvadratická a bikubická.



$$f(R1) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{1,1}) + \frac{x - x_1}{x_2 - x_1} f(Q_{2,1}) \quad kde\ R = (x, y_1)$$

$$f(R2) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{1,2}) + \frac{x - x_1}{x_2 - x_1} f(Q_{2,2}) \quad kde\ R = (x, y_2) \qquad (10)$$

$$f(P) = \frac{y_2 - y}{y_2 - y_1} f(R1) + \frac{y - y_1}{y_2 - y_1} f(R2)$$

Obrázek 5: Výpočet hodnoty pomocí bilineární interpolace, převzato z http://en.wikipedia.org/wiki/Bilinear_interpolation

2.1.4 Filtrace obrazu

Ne vždy je vhodné zpracovávat obraz ve formátu, v jakém byl pořízen z kamery, protože zpravidla obsahuje velké množství informací, které jsou pro daný účel nepodstatné a mohou stěžovat výběr informací podstatných. Proto se často obraz před vlastním zpracováním často filtruje. Výsledný obraz je konvolucí vstupního obrazu a vlastního filtru. Každý bod výsledného obrazu je vypočítán jako součet součinů bodu obrazu se středem filtru a jeho sousedů s příslušnými sousedy v matici filtru. Právě kvůli zajištění existence středu filtru má jeho matice liché rozměry, nejčastěji 3x3 nebo 5x5 bodů.

Filtrace je vhodná například pro zaostření obrazu či detekci hran v oblasti prostoru barev zejména v případech, kdy je velikost filtrační matice malá. V případech, kdy je

zapotřebí použít filtr větších rozměrů, operace konvoluce značně zvyšuje výpočetní náročnost, proto se často používá filtrace ve frekvenční oblasti. Často používanou operací je Fourierova transformace, pro zpracování obrazů pak zejména rychlá Fourierova tranformace (FFT), která vypočítá frekvenční spektrum. Ve frekvenčním spektru je pak možno aplikovat požadované filtry, například horní propust (HPF, High Pass Filter) pro detekci hran nebo dolní propust (LPF, Low Pass Filter) pro efekt rozmazání obrazu. Po rekonstrukci obrazu pomocí inverzní Fourierovy transformace pak dostaneme žádaný filtrovaný obraz.

Často používaným filtrem ve frekvenční oblasti v oblasti počítačového vidění je Gaborův filtr. Používá se zejména k segmentaci obrazu. Jedná se o lineární tzv. pass band filtr, který zesiluje frekvence v daném frekvenčním zespoda i shora ohraničeném pásmu definovaný jako:

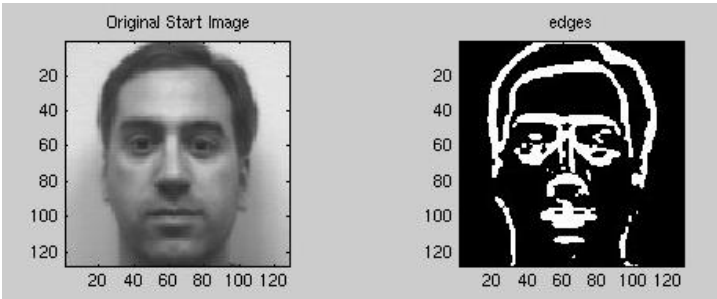
$$g(x,y:\lambda,\Theta\psi,\sigma,\gamma)=\exp\left(\frac{-x'^2+\gamma^2y'^2}{2\sigma^2}\right)\cos\left(2\pi\frac{x'}{\lambda}+\psi\right)$$

kde (11)

$$x'=x*\cos\Theta+y\sin\Theta$$

$$y'=-x*\sin\Theta+y\cos\Theta$$

kde λ reprezentuje vlnovou délku kosinového faktoru, Φ orientaci normály na paralelní pruhy Gaborovy funkce ve stupních, ψ fázové posunutí ve stupních, γ prostorový koeficient. Aplikací Gaborových funkcí získáme Gaborovy vlnky, které jsou zpravidla navrženy a předpočítány pro různé úhly otočení a šířku filtru.



Obrázek 6: Ukázka detekce hran z původního snímku, převzato z http://www.owlnet.rice.edu/~elec539/Projects97/morphjrks/moredge.html

2.1.5 Analýza hlavních komponent

Analýza hlavních komponent (angl. Principal Component Analysis – PCA) je často používanou metodou z oblasti statistiky pro redukci dimenze vstupních dat. Přesněji řečeno, jedná se o lineární transformaci, která transformuje data z jednoho souřadného systému do jiného tak, že v druhé dimenzi leží na první souřadnici dimenze s největší variancí hodnot

v prostoru prvním, na druhé souřadnici dimenze s druhou největší variancí v prostoru prvním atd. V praxi se tak PCA použije k výběru podmnožiny složek vstupních dat tak, aby ztráta informace byla co nejmenší.

Cílem výpočtu PCA je získání vlastních vektorů a jim příslušných vlastních čísel ze vstupní matice. Vlastní čísla a vlastní vektory získáme posloupností následujících operací.

$$\overline{X} = \frac{1}{N}\sum_{i=1}^{N} X_i \tag{12}$$

$$\sigma = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(X_i - \overline{X})^2} \tag{13}$$

Vstupní data jsou uložena v matici **X**, která má následující strukturu:

$$X = \begin{pmatrix} obrázek\ 1 \\ obrázek\ 2 \\ \vdots \\ obrázek\ N \end{pmatrix} \tag{14}$$

Rovnice 12 ukazuje způsob výpočtu střední hodnoty (průměrné tváře), rovnice 13 pak směrodatné odchylky. Dále předpokládáme, že data jsou normalizována.

$$C^{m\times n} = (c_{i,j}, c_{i,j} = Cov(Dim_i, Dim_j)$$

kde

$$Cov(X,Y) = \frac{\sum_{i=1}^{N}(X_i - \overline{X})(Y_i - \overline{Y})}{N-1} \tag{15}$$

Rovnice 15 ukazuje výpočet kovarianční matice **C**. Další operací je výpočet vlastních čísel a vlastních vektorů matice X. Pro vlastní vektory platí rovnice 16, kde **A** je operátor řádu n a **x** je vlastní vektor a λ je jemu příslušné vlastní číslo. V našem případě je operátorem kovarianční matice **C**.

$$A\vec{x} = \lambda\vec{x} \tag{16}$$

Pro nalezení vlastních čísel kovarianční matice **A** převedeme matici na soustavu rovnic dle 17.

$$\begin{vmatrix} a_{11}-\lambda & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22}-\lambda & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1n} & a_{2n} & \cdots & a_{nn}-\lambda \end{vmatrix} = 0 \tag{17}$$

Po vyřešení této soustavy rovnic tedy známe vlastní vektory a vlastní čísla matice **C** a můžeme říci, že hlavní komponentou je vektor s největším vlastním číslem. Vlastní vektory

kovarianční matice složené z obrazů obličejů se nazývají vlastní tváře (angl. Eigenfaces). Seřadíme-li vlastní vektory podle velikosti jejich vlastních čísel, dostaneme složky seřazené podle rozptylu. Pro redukci počtu dimenzí pak stačí vzít tolik složek ze začátku seznamu vlastních tváří (sestupně seřazeného dle příslušných vlastních čísel), kolik uznáme za vhodné. Tyto vybrané složky pak budou tvořit bázi pro další výpočty a tyto dále použité vlastní vektory budeme dále nazývat eigenfaces.

2.2 Současné přístupy k lokalizaci obličeje

2.2.1 Základní požadavky na systém

Lokalizací obličeje rozumíme určení pozice a otočení všech obličejů nacházejících se na vstupní fotografii, nebo žádnou odezvu v případě, že se na vstupním snímku žádný obličej nevyskytuje. Ideální systém je nezávislý na invarianci ve vstupních datech, a to ve všech výše jmenovaných bodech. Detekce je většinou pouze první částí systému pro rozpoznávání, je proto nutno návrh obou částí systému synchronizovat. Například velmi precizní systém pro rozpoznávání nepotřebuje přesný detektor, ale poradí si i s větším množství neobličejových dat chybně detekovaných jako obličejové (angl. false positives). K ohodnocení výkonnosti dané metody budeme používat náročnost na výpočetní prostředky. Úspěšnost je možno hodnotit pomocí míry chybně klasifikovaných obličejů jako neobličejová data (angl. false negatives) a chybně klasifikovaných neobličejových data jako obličeje (false positives). Pro detekci se v současné době používají různé modifikace metod, které lze shrnout do čtyř základních přístupů, které jsou popsány na několika následujících stranách. Bylo představeno přibližně 150 různých přístupů k detekci, které často využívají znalosti a postupy z více než jedné z níže popsaných kategorií. Všechny se primárně zabývají detekcí obličejů ze statických snímků (tedy nikoliv z videa), předpokládají snímky ve škále šedých odstínů a předpokládají též, že obličeje jsou natočeny přímo na kameru a normála roviny obličeje je opačná oproti normále snímku (angl. frontal upright face detection). U některých metod je diskutováno možné rozšíření i na detekci otočených obličejů. Následujících několik stran poskytuje základní přehled metod, přičemž jsem čerpal převážně z [3]. Je nutno podotknout, že hranice mezi nimi nejsou pevné a některé se vzájemně překrývají.

- Metody založené na znalostech (Knowledge-based methods): Metoda předpokládá zakódování lidských znalostí o prvcích, ze kterých se obličej typicky skládá, a popisu jejich vzájemných relací.

- Příznakové invariantní metody (Feature invariant approaches): Metoda se soustředí na nalezení strukturálních příznaků. Tyto příznaky se vyskytují v obličeji nezávisle na jeho otočení či světelných podmínkách.
- Šablonové porovnávání (Template-matching methods): Metoda předpokládá existenci šablon, které popisují obličej jako celek nebo jeho jednotlivé části. Detekce se pak provádí výpočtem korelace mezi vstupním obrazem a šablonami. Tuto metodu lze použít i následně ve vlastním rozpoznávání.
- Metody založené na vzhledu (Appearance-based methods): Metoda je velmi podobná jako předchozí, avšak modely (šablony) nejsou předem dodány. Znalosti o modelech jsou získány (naučeny) z tréninkové množiny vstupních obrazů, u nichž je dodán žádaný výsledek klasifikace, tedy příslušnost k jedné z cílových tříd. Jedná se o učení s učitelem (angl. Supervised Learning) Úspěšnost metody je pak silně ovlivněna složením testovací množiny.

K jednotlivým metodám bylo vypracováno množství studií, jejichž stručný přehled lze nalézt v [3].

2.2.2 Metody založené na znalostech

Jak již název metody naznačuje, metoda předpokládá množství apriori znalostí o lidském obličeji. Nepracuje se šablonou, ale se sadou pravidel, které obličej obecně popisují. Vstupní snímek se pak prohledává pomocí okna dané velikosti a v každém okně se hledají oblasti, které vyhovují zadaným pravidlům. Pravidla mohou mít tvar například:

- střed obličeje má podobné hodnoty intenzity
- obličej se skládá z dvou očí, které jsou symetrické, nosu a úst, které se nacházejí v prostoru pod očima, může být též definována průměrná relativní vzdálenost mezi těmito obličejovými prvky [3].

Je poměrně jednoduché zavést takováto pravidla, ovšem vzhledem k vysoké variabilitě obličejů není již tak jednoduché takto zavedená pravidla správně aplikovat. Nejprve je tedy nutno zvolit vhodný kompromis mezi výkonností systému a precisností pravidel. Obecně platí, že precisně zavedená pravidla poskytují nižší míru neobličejových částí detekovaných jako obličeje, ale některé obličeje nemusí systém vůbec detekovat.

Zvýšení výkonnosti metod založených na pravidlech přináší vícestupňová aplikace pravidel. Jednou možností je podvzorkovávání a aplikací různých pravidel na různém stupni podvzorkování. Tento hierarchický model přináší kromě snížení počtu chyb též snížení nároků na výkon počítače. Jinou alternativou je zjištění přítomnosti obličejových rysů například

pomocí statistických metod. K tomu může sloužit histogram rozložení intenzity v právě zpracovávaném okně a použití pravidel pouze k validaci, zda takto rozložená intenzita je náhodná a odpovídá pozadí nebo zda signalizuje přítomnost obličeje. Pravidla určená pro validaci používají již velmi specifické obličejové prvky, jako je například poloha obočí. Metoda selhává při příliš komplexním pozadí nebo u fotografií vyššího počtu osob. Nevýhodou je též nemožnost rozšíření na detekci pootočených obličejů.

2.2.3 Příznakové invariantní metody

Na rozdíl od metod založených na znalostech, metody založené na příznacích, se nesnaží hledat pomocí předem známých informací strukturální příznaky, které potvrdí existenci obličeje. Naopak se snaží jednoduchými cestami nalézt množinu příznaků a vyšetřit, zdali se může jednat o obličej. Nalezení příznaků tedy není řešením, ale výchozím bodem. Vychází z metod umělé inteligence pro příznakové rozpoznávání obrazů a předpokladu, že pokud člověk dokáže různé objekty v různých pozicích jednoznačně identifikovat, musí existovat nějaká invariantní charakteristika těchto objektů. Nalezení těchto příznaků jako například obočí není složitou záležitostí a lze ji provést například detekcí hran pomocí filtrace v barevném (intenzitním) nebo frekvenčním spektru snímku.

Opět bylo představeno několik konkrétních metod pracujících na základě příznaků. Všechny lze rozdělit do dvou fází. První fází je nalezení příznaků, které mohou být částí obličeje. Jednou z metod je filtrování pomocí pásmových filtrů. Pásmové filtry odstraňují frekvence vyšší i nižší než je dané pásmo, pro které je filtr navržen.

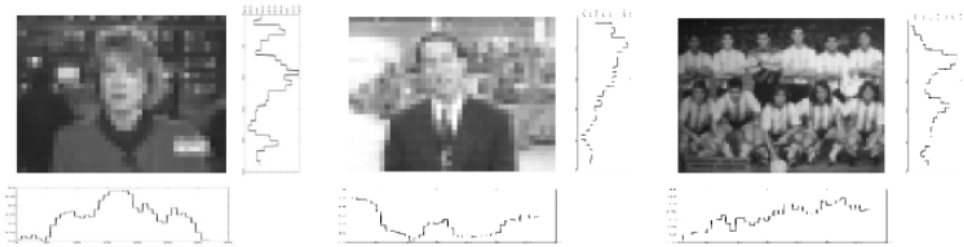
Další z možností je vytváření tzv. mapy hran. Systém nejprve nalezne hrany a pak postupně odebírá ty, které nemohou být součástí obličeje na základě seskupování. Pravidla pro jednotlivé skupiny jsou předem známa.

Vzhledem k vysoké pravděpodobnosti výskytu šumu ve vstupních datech jako jsou například nepříznivé světelné podmínky, je nutno před vlastní aplikací metod tyto nepříznivé jevy potlačit, například pomocí ekvalizace histogramu, který je definován výše. Obecně nejsou tyto metody příliš často nasazovány, a i když je úspěšnost těchto metod poměrně vysoká pro nějakou konkrétní databázi obličejů, ukazuje se, že lze nalézt a upravit jiné metody, které pro daný účel vykazují úspěšnost vyšší.

Velmi zajímavá je též možnost vyjít ze specifické barvy lidského obličeje a tuto informaci použít při detekci. Metoda předpokládá snímky získané při kontrolovaných podmínkách, které nemají vliv na případné barevné zkreslení snímku. Používá se specifické schéma prohledávání snímku oknem a po aplikaci filtrů hledání oblastí obsahujících barvu obličeje a vlasů. Pro rozhodnutí, zda se v oblasti nachází obličej, lze použít neuronovou síť

nebo shlukovací algoritmus, který žádané shluky vytvoří z trénovací množiny, u níž je známa správná klasifikace. Velkou výhodou těchto metod je skutečnost, že dokáží lokalizovat i obličeje otočené a dokáží si do jisté míry poradit i s tvářemi obsahujícími například brýle či vousy. Nevýhodou však je již zmíněná citlivost na světelné podmínky při snímání obrazu a různá barva vlasů, která též může způsobovat komplikace.

Další rozšíření nabízí přístup kombinující hledání několika různých rysů v obraze. Jako prvotní nástroj pro výběr kandidátních oblasti lze použít např. metody využívající barvy společně s metodami pracujícími s jednoduchými příznaky. Pro vyloučení či potvrzení kandidátů lze pak použít metody složitější, například hledání detailnějších obličejových prvků.



Obrázek 7: Detekce obličeje pomocí barvy, převzato z [3]

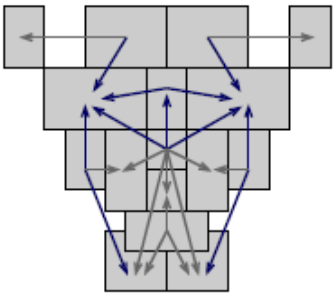
2.2.4 Metody založené na šablonovém porovnávání

Metody jsou založené na jednoduchém principu, že obličej se dá obecně popsat pomocí jednoduchých geometrických útvarů. Vlastní šablona je pak takto popsaná sada obličejových prvků včetně jejich rozmístění. Šablona je většinou vytvořena člověkem na základě znalostí o obličeji. Vlastní detekce pak probíhá výpočtem korelace mezi vstupním snímkem (případně právě zpracovávaným oknem) a šablonou. Pokud je korelace dostatečně velká, je vstupní obraz klasifikován jako obličej.

Jako i ostatní metody však tento základní přístup není dostatečně přesný pro praktické použití. Vzhledem k vysoké variabilitě obličejů je často nutno použít ne jednu globální šablonu, ale hierarchii šablon. Na nejvyšší úrovni jsou ze vstupního snímku vybíráni kandidáti splňující minimální požadavky na šabloně nejvyšší úrovně. Postupně jsou aplikovány další zpřesňující šablony jak pro celý obličej, tak pro jeho jednotlivé oblasti.

Problémem však zůstává, jak jednoznačně namapovat šablonu na vstupní obraz. Jedním z řešení je detekce hran ve snímku a následný pokus vyhledat v seznamu hran takové, kterým je možno opsat ovál představující obličej. Další možností je například pro nalezení kandidátní oblasti použití nějaké jednoduché příznakové metody, hledající pro obličej specifické rozdíly kontrastu ve vstupním snímku.

Dalším problém, který stěžuje možnost nasazení metod založených na šablonách, je skutečnost, že tyto metody se jen velmi špatně vypořádávají s transformovanými obrazy. Různé velikosti zkoumaných obličejů, stejně jako jejich různé otočení, znemožňuje nasazení šablonových metod v systémech, kde nelze zaručit nízkou variabilitu vstupních dat. Přesto existují i metody, které dokáží pracovat i s takto částečně transformovanými obrazy. Jednou z nich může být například šablona popsaná parametrickými funkcemi. Až po nalezení prvních příznaků a na základě znalostí relativních vzdáleností mezi jednotlivými obličejovými rysy se vytvoří šablona s příslušnou velikostí, případně i natočením. U systémů s nízkou variabilitou vstupních dat je velkým plusem metod založených na šablonovém porovnání jejich vysoká úspěšnost.



Obrázek 8: Ukázka šablony pro oblasti obličeje včetně jejich vzájemných vztahů [2]

2.2.5 Metody založené na vzhledu

Na rozdíl od metod založených na šablonách, metody založené na vzhledu nepočítají s existencí uživatelem zadaných pravidel, ale „šablony" jsou získány učením z trénovací množiny. Právě toto spolehnutí se na výběr znaků typických pro obličej učícím algoritmem za podpory statistických metod umožňuje metodám založeným na vzhledu dosáhnout poměrně vysoké efektivnosti.

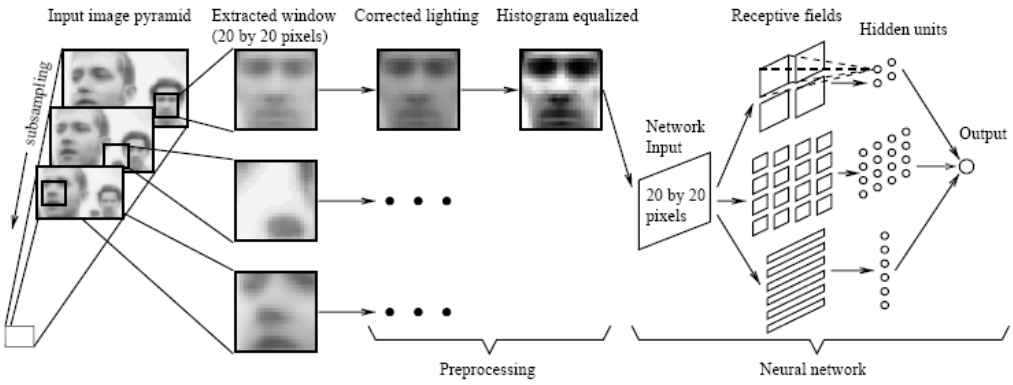
Tyto metody pracují se vstupními daty jako s náhodnou proměnnou. Vstupem klasifikátoru je pak celý obraz (okno) nebo část jeho prostoru. Může pak být klasifikován například pomocí Bayesovského klasifikátoru. Jiným přístupem je hledání nějaké funkce, která dokáže oddělit třídu obličejů od neobličejových dat. Taková funkce může mít tvar například rozdělující nadroviny a implementována může být neuronovou sítí, pomocí Support Vector Machines, ale existují i jiné způsoby implementace. Vzhledem k tomu, že dimenze vstupních dat je poměrně velká, jsou často před vlastní klasifikací aplikovány na tato data metody, které jejich dimenzi snižují. Mezi takovéto metody patří typicky výše zmíněná PCA. Právě PCA lze použít k výpočtu tzv. Eigenfaces, což je podmnožina vlastních vektorů

kovarianční matice. Klasifikace se pak provádí nad váhami vstupního obrazu k vektoru vlastních tváří.

Shlukování je jednou z dalších použitelných metod. Obecně nemusí docházet k redukci obrazového prostoru, pokud jsou obrazy dostatečně malé (často používanou velikostí jsou výřezy snímku oknem o velikosti kolem 20x20 bodů). V čase trénování se rozdělí trénovací množina do několika shluků pro obličeje a pro pozadí. Klasifikace probíhá na základě porovnání vzdáleností od jednotlivých shluků, případně na základě složitějších metrik. Samozřejmě i zde je možné použít přístup založený na vlastních tvářích a prostor tak zredukovat.

Další častou architekturou pro výstavbu klasifikátoru jsou neuronové sítě. Vstupem neuronových sítí mohou být výřezy snímků, váhy k vektoru vlastních tváří, obrazy zrekonstruované pomocí vlastních tváří či obrazy různě filtrovaných snímků. Neuronové sítě pro klasifikaci se skládají obvykle z několika skrytých vrstev. Mezi nejvýznamnější aplikace neuronových sítí patří bezesporu systém představený autory Rowley, Baluja, Kanade v [2]. Pro klasifikaci je použita několikavrstvá neuronová síť se specifickým zapojením jednotlivých vrstev. Následně byla síť rozšířena o detekci otočení obličeje, která tak umožnila detekovat i otočené obličeje.

Výhodou neuronové sítě je poměrně vysoká schopnost korektně reagovat na vysokou variabilitu vstupních dat, nevýhodou může být nutnost manuálně nastavovat její parametry. Ty totiž velmi významně ovlivňují celkovou detekční schopnost.



Obrázek 9: Systém pro detekci obličeje představený v [2]

3 Návrh systému pro lokalizaci obličeje

3.1 Motivace

V minulé kapitole jsem představil základní kategorie metod, které mohou být použity k detekci. Celkový výčet metod je poměrně široký a vzhledem k množství kombinací nejsou zdaleka popsány všechny metody. Dále jsem již zmínil důležitost lokalizace obličeje pro další fáze zpracování především pak rozpoznávání. Téma této práce vzniklo na základě obou těchto skutečností a úkolem je navrhnout a implementovat systém, který nevychází z některého jiného úspěšného systému jako například [2], ale umožní otestovat a zdokumentovat některé z méně známých postupů. Tato kapitola popisuje návrh systému krok po kroku, včetně zdůvodnění, proč jsem jednotlivé komponenty do návrhu zařadil.

3.2 Předzpracování vstupních dat

Systém je zaměřen na lokalizaci obličeje na komlexních snímcích o větších rozměrech, je tedy nutno vstupní snímek rozdělit na množství menších o velikosti 19x19 obrazových bodů (tyto části budou dále nazývána okna). Velikost není volena náhodně, jedná se o rozměry snímků z použité trénovací databáze obličejů. Použití právě této databáze umožní srovnat výsledky dosažené tímto systémem se systémy jinými, jejichž výsledky jsou velmi často prezentovány na testovacích množinách této databáze. Po klasifikaci všech oken dojde vždy ke zmenšení snímku, přičemž výchozí koeficient zmenšení je odhadnut na 1,2 a celý proces bude takto pokračovat, dokud bude velikost snímku větší nebo rovna velikosti okna. Při vytváření nového zmenšeného snímku budou barvy jeho bodů interpolovány z původních pomocí bilineární interpolace.

V další fázi bude ze vstupních oken vypočítán histogram a okna budou ekvalizována. Ekvalizace oken namísto ekvalizace celého původního snímku by měla přinést zlepšení kontrastu pro jednotlivá okna. Pro potřeby neuronové sítě budou barvy bodů převedeny do normalizované formy.

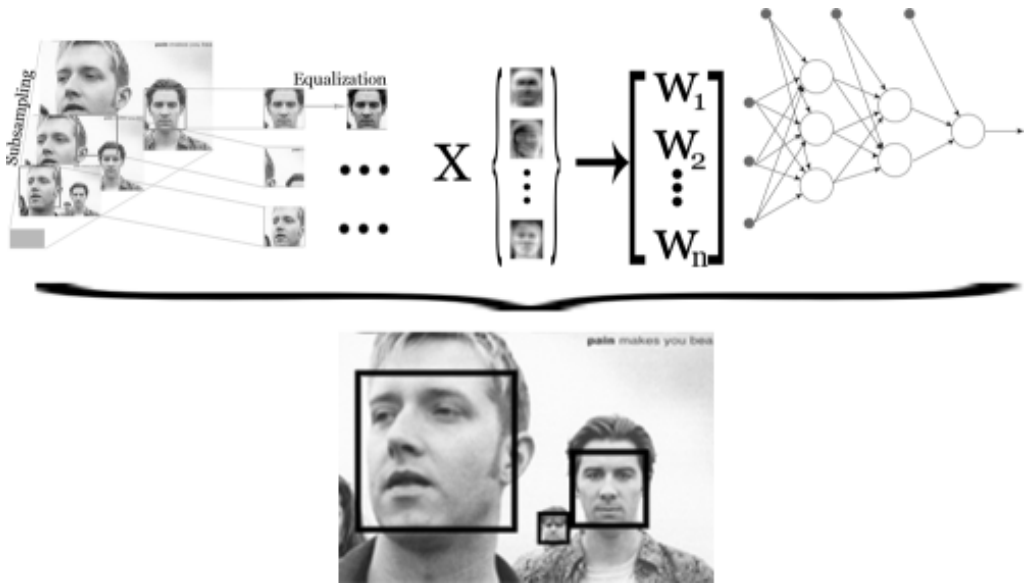
Za účelem snížení dimenze obrazu a následné snížení výpočetní náročnosti neuronové sítě jsem se rozhodl použít přístupu založeném na vlastních tvářích. Výběr právě této metody s sebou přináší dvě výhody. První z nich je možné snížení dimenze vstupních dat. Neuronová síť tak může mít menší počet vstupů, což může značně urychlit klasifikační fázi. Výsledkem by tedy mělo být nejen zvýšení rychlosti, ale i zjištění závislosti úspěšnosti klasifikace na počtu hlavních komponent, které jsou pro výpočet použity.

Onou další výhodou je pak skutečnost, že vstupem neuronové sítě nejsou data v surové podobě, ale vektor vah k jednotlivým vlastním tvářím. Cílem je dosažení nezávislosti na vstupních datech, neboť klasifikace probíhá v prostoru vlastních tváří. Problematika výpočtu vektoru vah k matici vlastních tváří je záležitostí spíše implementační, proto není popsána zde v části návrhu systému. Čtenář s ní bude seznámen v druhé polovině této práce, kde je tento postup zdokumentován. Toto řešení snad nabídne čtenáři lepší pochopení souvislostí v kontextu s aktuální fází klasifikace a s návaznostmi na provedené testy.

3.3 Návrh klasifikátoru

3.3.1 Neuronová síť

Jako klasifikátor jsem vybral neuronovou síť. Neuronové sítě mají obecně velmi slušnou výkonnost a pro účely detekce jsou vhodné. Z množství v oblasti softcomputingu používaných sítí jsem zvolil síť Back Propagation, jejímž algoritmem učení je zpětné šíření chyby. Jedná se o plně propojenou dopřednou síť. Plně propojená architektura je vhodná pro tyto účely, nejsou tedy potřeba žádné apriori manuální optimalizace propojení. Samozřejmě budou otestovány i jiné architektury neuronových sítí na základě schopností použité knihovny pro neuronové sítě. Není důvod síť implementovat v rámci tohoto projektu, cílem je najít vhodné parametry, na něž se budu plně koncentrovat. K implementaci bude pravděpodobně použita některá dostupná knihovna pro neuronové sítě s vhodnou licencí. Tato volba je zdůvodněna v následující kapitole na základě konzultace s vedoucím diplomové práce.



Obrázek 10: Blokové schéma navrženého systému

3.3.2 Trénovací množina

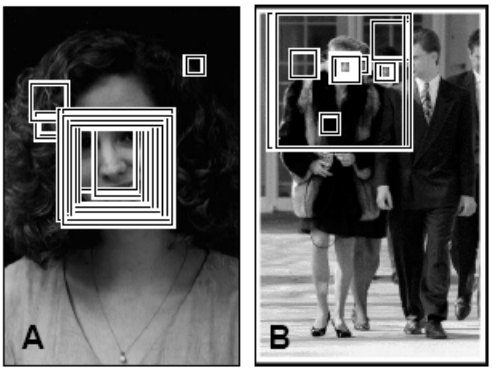
Existuje větší množství dat přístupných pro trénování. Po dohodě s vedoucím jsme zvolili databázi CBCL dostupnou na webových stránkách MIT [5]. Databáze obsahuje 2.429 obličejů a 4.548 neobličejových dat pro trénování a 472 obličejů a 23.573 neobličejových dat pro testování systému. Jsou ve formátu PGM (Portable Grey Map), což je jednoduchý formát pro uložení obrázků ve stupních šedi. Velikost obrázků pro trénování je zmíněných 19x19 bodů. Není tedy nutná žádná manuální příprava snímků.

Obrázek 11: Ukázka obličejů z databáze CBCL (obrázky zvětšeny)

3.4 Zpracování výsledků

3.4.1 Eliminace duplikátů

Obrázek 12 ukazuje situaci, kdy použitá metoda lokalizuje jeden obličej na několika místech ve snímku. Nejedná se zpravidla o chybu metody, která takto nízko nastaveným prahem pro rozhodnutí, zda se ve zkoumané oblasti nachází obličej, zvyšuje počet správně nalezených obličejů. Naopak, ve skutečnosti lze na snímku s právě jedním obličejem nalézt větší množství výřezů tohoto snímku, na kterých je obličej stále dobře rozeznatelný. Dále pak navržená metoda pracuje s fixní velikostí zpracovávané oblasti obrazu, tzv. okna. Pro detekci obličejů i na snímcích, kde se mohou nacházet různé velikosti obličejů, je tedy nutné zpracovávat snímek několika průchody. Sloučení nálezů pak je provedeno na základě prahu, pokud se plocha nálezů překrývá o více než je stanovený práh, přičemž jako výsledek může být průnik nebo sjednocení oblastí nálezů. Pokud se oblasti překrývají méně nebo se jedná o nálezy na v obraze s velmi rozdílným měřítkem, jsou nálezy prezentovány jako dva různé detekované obličeje.



Obrázek 12: Několikanásobná detekce, převzato z [3]

3.4.2 Očekávaný výstup systému

Očekávaným výstupem neuronové sítě je rozhodnutí, zda se v daném okně nachází obličej či nikoli. Výstupem celého systému je však zanesení pozice nalezeného obličeje ve vstupním snímku. Z nálezu bude vytvořen seznam a atributy nálezů budou pozice a aktuální zmenšení vstupního snímku. Protože koeficient zmenšení je poměrně malý a okno bude posouváno po jednom bodě, očekávám množství detekcí, které se budou překrývat. Bude tedy použit postup popsaný výše ke sloučení takovýchto nálezů. Ke zhodnocení systému pak bude použita úspěšnost jak při slučování vícenásobných nálezů pomocí sjednocení tak při slučování pomocí průniku.

4 Implementace systému pro lokalizaci obličeje

4.1 Základní informace

Popisovaný systém je implementován v jazyce C++. Ke správné funkčnosti systému je třeba mít k dispozici další knihovny jako GNU Scientific Library a FANN – Fast Artifical Neural Networks Library. Pro vizualizaci výstupů systému je použit program GnuPlot, který je nutno k tvorbě grafů mít nainstalován v systému. Základnímu popisu výše zmíněných komponent jako i návod k instalaci a popisu licenčního ujednání k jejich použití bude věnována část textu dále. Jako implementační prostředí bylo zvoleno po dohodě s vedoucím práce Microsoft Visual Studio 2005, které bylo v době psaní této práce k dispozici na počítačích Centra výpočetní techniky FIT. Z použití tohoto studia plyne i systémová platforma, na které byl systém odladěn, tedy Microsoft Windows XP. Visual Studio 2005 je plně profesionální nástroj a jeho volba se ukázala jako vhodná pro projekt tohoto typu, protože poskytuje značnou přehlednost kódu i ve větších projektech.

Při psaní kódu jsem se snažil zachovávat konvence firmy Sun, jejichž definice je k nalezení na stránkách uvedených v [12]. Ta je sice určena především pro jazyk Java, její velká podmnožina je však velmi dobře použitelná pro všechny moderní objektově orientované jazyky.

4.1.1 Struktura projektu

Celý projekt je možno rozdělit do 4 základních částí. Každá z nich bude postupně rozebrána. V první fázi jsem se pokusil vytvořit systém, který bude klasifikovat testovací data čistě pomocí neuronové sítě, bez speciální přípravy dat. Až po této fázi zavedeme pojem vlastních tváří. Druhá část je tak věnována jejich výpočtu a způsobu použití. Protože tato poměrně složitá transformace dat není určena pouze pro neuronové sítě, třetí část je věnována detekci právě a jen pomocí jednoduchého klasifikátoru využívajícího vlastních tváří, bez přítomnosti neuronové sítě. Ve čtvrté části je pak popsána klasifikace obrazů pomocí neuronových sítí s použitím množiny vlastních tváří.

4.1.2 Fast Artifical Neural Networks Library

Jedná se o Open Source knihovnu (dále bude uváděna pod zkratkou FANN) neuronových sítí, kterou jsem po dohodě s vedoucím vybral k implementaci. Knihovna je napsána v jazyce C, avšak k dispozici jsou verze určené pro použití v mnoha programovacích

jazycích. K dispozici je samozřejmě i verze pro C++, kterou jsem použil i já. FANN je distribuován pod licencí GNU General Public Licence, kterou je možno nalézt na webových stránkách uvedených v [13]. Tato licence je velmi vhodná pro použití v práci akademického charakteru. V době psaní tohoto projektu byla k dispozici verze knihovny 2.1.0beta.

Knihovna neobsahuje žádné platformově specifické závislosti, její překlad je tedy velmi jednoduchý, a celá knihovna je tak dodána ve zdrojovém tvaru jako součást diplomové práce. Výsledkem překladu je statická knihovna, která je automaticky připojena linkovacím programem. Datový typ, se kterým pak knihovna pracuje, závisí na hlavičkovém souboru, který je vložen do zdrojového kódu. FANN umožňuje pracovat s jednoduchou (float), dvojitou (double) přesností desetinných čísel nebo s pevnou řádovou čárkou. Použití pevné řádové čárky je velmi specifické a nebylo použito. Provedl jsem řadu experimentů s přesností double, síť však vykazovala velkou nestabilitu a v některých případech dokonce absolutní neschopnost se učit. Mé poznatky o přítomnosti závažných chyb v knihovně potvrzují některé příspěvky v diskuzním fóru tohoto projektu a v poštovních příspěvcích (mailing-lists). Byl jsem tedy nucen použít jednoduchou přesnost, která tyto nestability nevykazovala. Nastavení jednoduché přesnosti se děje jednoduše vložením hlavičkových souborů fann.h nebo floatfann.h (na rozdíl od doublefann.h pro dvojitou přesnost).

4.1.3 Použití tříd pro obsluhu neuronové sítě

Ve všech částech systému je použito jednoduché schéma pro obsluhu neuronové sítě. To je složeno ze tří tříd. Tyto třídy mají ve všech částech systému podobný název i funkci. První z nich je třída zajišťující trénovaní sítě, druhá pak získání výstupu natrénované sítě. Nad nimi stoji komponenta určená k testování dvou výše uvedených a celého systému zároveň. Zde je možno určovat parametry trénovaní i testů a též specifikovat informace, které budou sloužit jako výstup testovacího procesu.

Všechny třídy mají podobnou, velmi jednoduchou strukturu. Inicializace se provede voláním metody init(), jejímž parametrem je většinou jméno skriptu obsahujícího informace, které se během běhu nemění, většinou vstupně-výstupního charakteru, jako je například umístění trénovací množiny. Destrukce se pak provádí voláním destroy(), která uvolní všechny alokované zdroje a uzavře otevřené soubory. Důvodem pro zavedení těchto metod je praktická nemožnost signalizace chyb návratovou hodnotou nebo lépe vyhozením výjimky z konstruktorů/destruktorů. Systém reaguje na většinu chyb výjimkou, je tedy nutno potenciálně nebezpečné operace důsledně ošetřit. Všechny třídy jsou použitelné pouze mezi voláním metod init() a destroy().

Třídy určené k trénování a testování mají dále metody pro nastavení svých parametrů, jako je např. použitá velikost trénovací množiny. Změnu těchto parametrů lze provést až po zavolání init(). Změna parametrů je velmi užitečná při provádění cyklických testů odezvy systému pro různé parametry. Vzhledem k většímu množství trénovacích a testovacích postupů pak příslušné třídy obsahují metody pro různé způsoby trénování a testování.

4.1.4 Trénovací model

Knihovna FANN nabízí trénovací rozhraní k neuronové síti podporující několik způsobů trénovaní. Prvním z nich, a to tím nejjednodušším, je provedení jedné trénovací epochy na jednom vstupním vzorku dat. Dalším způsobem je poskytnutí celé trénovací množiny a řídit celý trénovací proces definicí chybové funkce a požadované maximální chyby, která je vyčíslena automaticky po každé trénovací epoše. Trénování je pak automaticky zastaveno při dosažení této maximální chyby. Další, nejvíce sofistikovaný model, který FANN nabízí, je kaskáové přidávání neuronů do skrytých vrstev. Na začátku je vytvořena neuronová síť bez skrytých vrstev. Neurony jsou pak do skrytých vrstev přidávány na základě užitečnosti takto přidaných neuronů, která je vyčíslována po každé trénovací epoše. Algoritmus, který je v tomto případě použit, je Cascade2.

Velmi důležitými parametry určujícími výkonnost neuronové sítě je počet skrytých neuronů, volba algoritmu pro učení neuronové sítě a volba aktivačních funkcí skryté a výstupní vrstvy. Tyto parametry jsou součástí testů a jsou popsány dále pro každý testovaný případ.

V dalším textu se poměrně často setkáme s případem, kdy množina velikost trénovacích tváří byla různá od množiny trénovacích netváří. Tato skutečnost je dána strukturou databáze CBCL, neboť ta nabízí větší množství netváří jak v trénovací tak v testovací podmnožině. Bylo by tedy škoda získaná data nevyužít, a proto jsem se snažil trénovat neuronové sítě s dvojnásobným a čtyřnásobným počtem neobličejů. Samozřejmě většina testů byla provedena i se stejnou velikostí trénovacích množin, ale obecně nevykazovala vyšší výkonnost. Pokud by tomu tak někde skutečně bylo, bude to v následujícím textu dostatečně zvýrazněno. Další nepříjemnou vlastností použité obličejové databáze je, že obsahuje vždy několik snímků pořízené od jedné osoby, které navíc spolu sousedí pří řazení dle jména souboru. Bylo tedy bezpodmínečně nutné tuto silnou korelaci po sobě jdoucích trénovacích vzorků narušit. Vzhledem k tomu, že vstupem většiny trénovacích metod je i soubor obsahující seznam souborů s trénovacími vzorky, je vhodné tuto úpravu provést právě zde. Součásti implementace je i jednoduchý projekt pojmenovaný Permutator,

který má na vstupu právě soubor se seznamem souborů se vzorky a pomocí standardní C++ knihovny provede jejich permutaci, kterou uloží do souboru, jehož jméno je specifikováno v argumentu programu, ve stejném formátu jako je soubor vstupní. Takto permutovaný soubor je pak dále použit pro trénování. V případě testovacích dat tato závislost mezi po sobě následujícími vzorky není tak kritická, ale pro jistotu jsou používány též permutované posloupnosti testovacích dat.

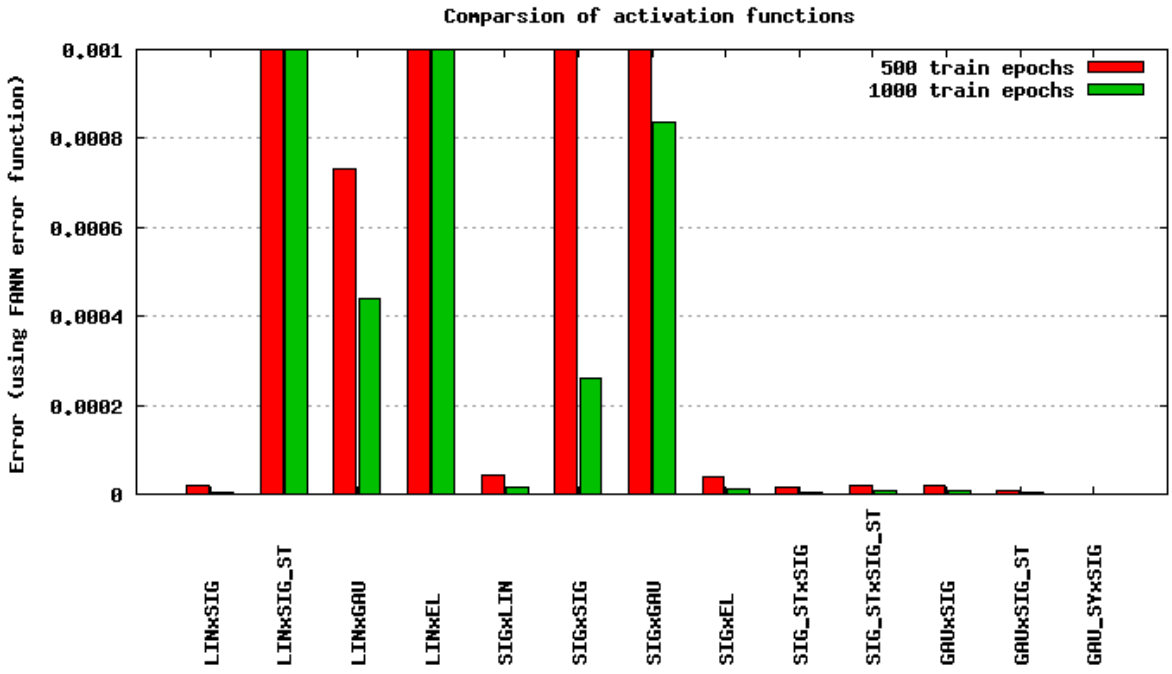
4.2 Lokalizace obličeje pomocí neuronových sítí

V této úvodní části práce jsem se pokusil sestavit a natrénovat jednoduchou neuronovou síť, která má sloužit jako klasifikátor pro surová data, tedy bez jakýchkoli úprav a předzpracování. Ačkoli jsem se snažil soustředit na klasifikaci pomocí vlastních tváří, tento dnes již klasický přístup nemohu vynechat. Zároveň bude tento model sloužit jako referenční model pro následující bod této práce, aniž bych musel srovnávat činnost a výsledky s pracemi někoho jiného. Toto srovnání je samozřejmě možné, ale je silně závislé na stupni rozpracování, na použitých technologiích a hlavně na účelu produktů třetích stran. Tato volba se zdá rozumná, není třeba tedy výsledky třetích stran nijak ověřovat či dodatečně testovat za shodných podmínek.

4.3 Trénovaní neuronové sítě

4.3.1 Výběr vhodných aktivačních funkcí

Správné nastavení všech základních parametrů neuronové sítě není jednoduchá záležitost a nedají se předem odhadnout. Záleží na přesnosti výpočtu uvnitř použité knihovny pro neuronovou síť, jakož i na povaze testovaných dat. Knihovna FANN nabízí větší množství aktivačních funkcí (použitá verze jich měla k dispozici 11) a ke správnému odhadu je potřeba udělat jejich kartézký součin pro aktivační funkce neuronů ve skryté vrstvě a stejně tak pro neurony ve vrstvě výstupní. V ideálním případě je tak nutno prozkoumat všech 121 kombinací, což je ovšem velmi zdlouhavé. Proto jsem pro tuto část použil jednoduchou utilitu FANTool, která je k dispozici ke stažení na webu knihovny FANN [8], a distribuována je pod stejnou GNU GPL licencí. V době psaní této práce byla k dispozici verze 0.7. Tato utilita je navržena právě pro testy podobného typu. Výsledky testu aktivačních funkcí jsou na obrázku 13. Tabulka 1 obsahuje legendu ke zkratkám použitým v grafu. Zkratky jsou ve tvaru $<f_1>x<f_2>$, přičemž f_1 znamená aktivační funkce neuronů ve skryté vrstvě a f_2 aktivační funkce neuronů ve vrstvě výstupní.



Obrázek 13: Srovnání úspěšnosti klasifikace při použití různých kombinací aktivačních funkcí.

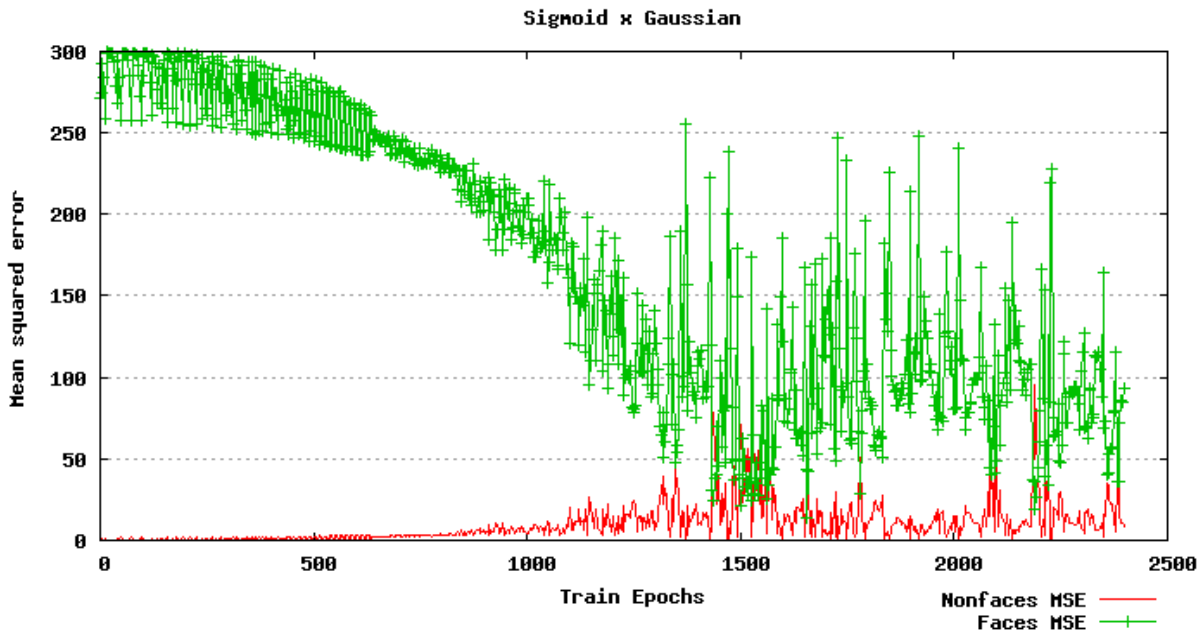
Zkratka	Celý název v angličtině	Název v češtině
LIN	Linear	Lineární funkce
SIG	Sigmoid	Sigmoidální funkce
SIG_ST	Sigmoid Stepwise	Sigmoidální kroková funkce
GAU	Gaussian	Gaussova funkce
EL	Elliot	Elliotova funkce
GAU_SY	Gaussian Symmetric	Gaussova symetrická funkce

Tabulka 1: Význam zkratek aktivačních funkcí použitých v obrázku $.

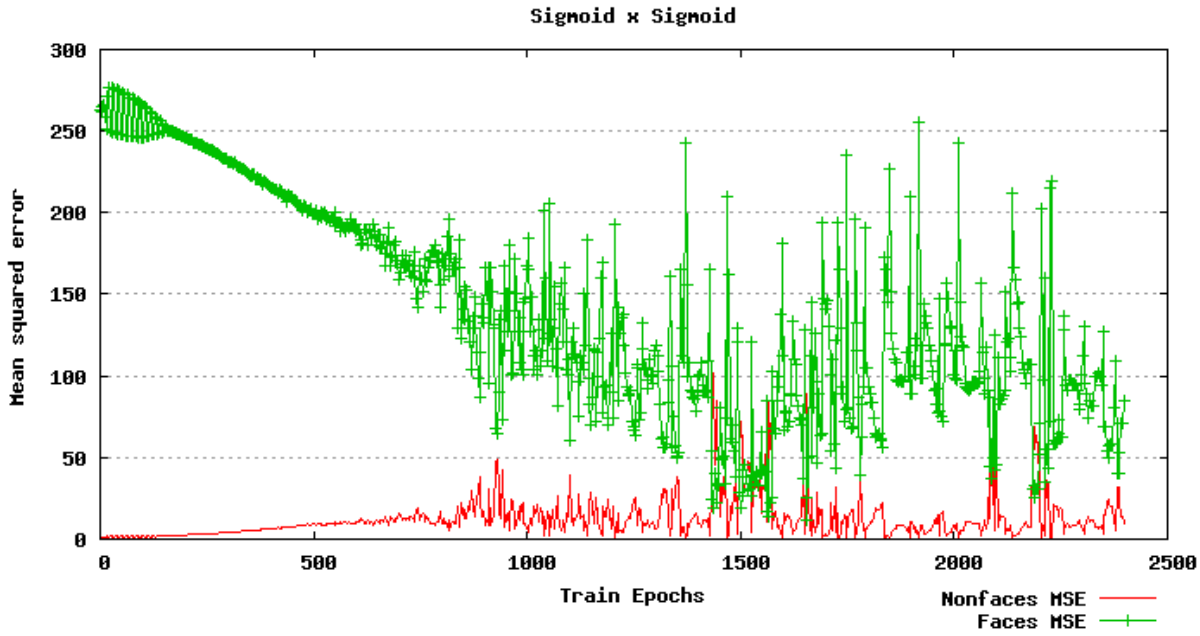
Pro trénování sítě byl použit algoritmus RPROP (rozbor a popis trénovacích algoritmů je o několik odstavců dále), o kterém se dá obecně říci, že vykazuje dobré výsledky, což platilo i v našem případě, jak uvidíme dále. Pro účely vizualizace jsem vybral výslednou chybu po pěti stech, resp. tisíci epochách. Ačkoli síť není plně natrénována (k tomu by bylo třeba vzorků mnohem více), právě pro možnosti vizualizace se tato čísla ukázala jako nejvhodnější. Při více epochách je chyba významně zkreslena a je velmi malá. Zajímavé je též zlepšení klasifikace u některých aktivačních funkcí po 1000 vzorcích a právě tento rozdíl dost možná významnější než absolutní hodnota této chyby.

Výstup systému FANNTool neumožňuje přímou konverzi do formátu použitelném v prostředku GNUPlot, konverzi proto bylo nutno provést manuálně. Právě z tohoto důvodu, jakož i prostorové náročnosti zobrazení všech kombinací, jsem vybral do grafu pouze funkce, které byly zajímavé, obecně často používané nebo vykazovaly výborné výsledky. Při použití některých kombinací (např. lineární aktivační funkce ve skryté i výstupní vrstvě) se

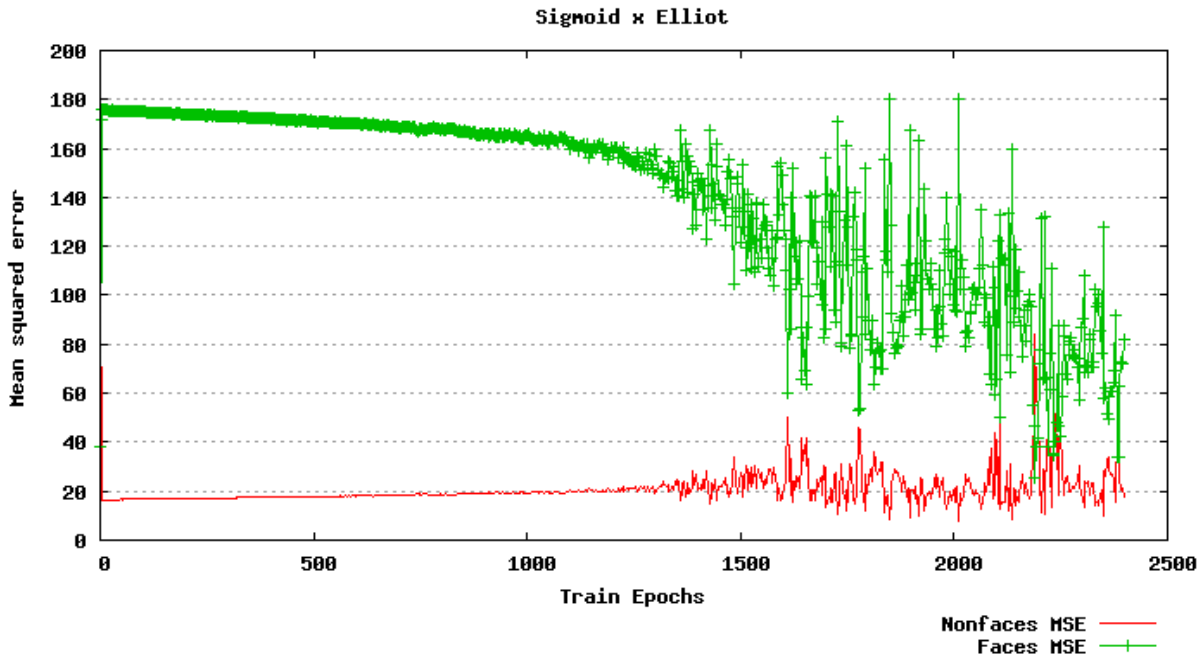
nepodařilo neuronovou síť natrénovat vůbec. Lze vidět, že poměrně velké množství funkcí vykazuje výbornou úspěšnost klasifikace. Bohužel FANNTool používá k vyčíslení úspěšnosti ne zcela optimální metodu, kdy na vyčíslení chyby v daném okamžiku trénovaní jsou použita trénovací data. Jedná se právě o data, na kterých byla síť natrénována, a dá se tedy předpokládat vysoká úspěšnost při klasifikaci takovýchto dat. Bylo by tedy chybou takovéto výsledky přijmou jako finální a je nutno ověřit úspěšnost daného nastavení ještě jednou na testovací množině obličejové databáze (obrázky 14, 15, 16, 17). Tyto testy ukázaly, že nejlepší výsledky pro účel této práce vykazuje kombinace sigmoidální aktivační funkce ve skryté vrstvě a sigmoidální nebo gaussova funkce ve výstupní vrstvě. Ukázky některých dalších kombinací jsou uvedeny v příloze 1. Použití krokových aktivačních funkcí celý výpočet značně urychlí, jejich výsledky však jsou ve většině případů horší než v případě daleko jemnější diskrétní aproximace jejich spojitých ekvivalentů.



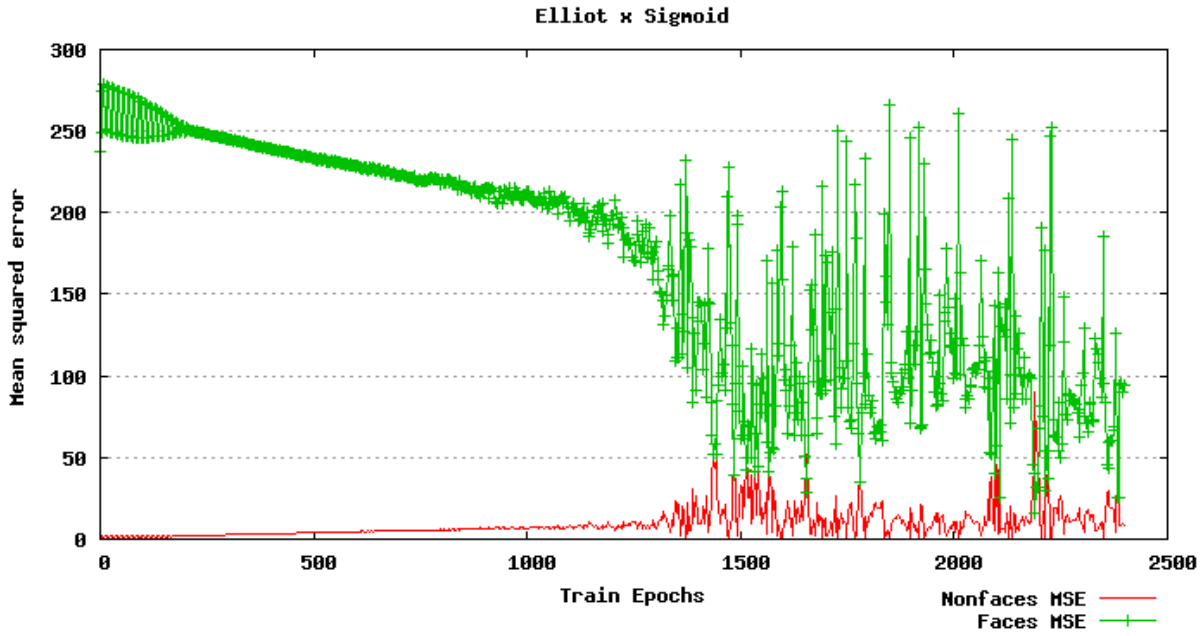
Obrázek 14: Chyba na testovací množině pro sigmoidální aktivační funkcí ve skryté vrstvě a gausovu aktivační funkci ve výstupní vrstvě.



Obrázek 15: Chyba na testovací množině pro sigmoidální aktivační funkcí ve skryté i výstupní vrstvě.



Obrázek 16: Chyba na testovací množině pro sigmoidální aktivační funkci ve skryté vrstvě a elliotovu aktivační funkcí ve výstupní vrstvě.



Obrázek 17: Chyba na testovací množině pro elliotvu aktivační funkci ve skryté vrstvě a sigmoidální aktivační funkci ve výstupní vrstvě.

Výše uvedené grafy ukazují závislost chyby klasifikace testovací množiny na velikosti trénovací množiny zvlášť pro množinu faces a nonfaces. Funkce použitá pro vyčíslení chyby byla $MSE(T)$ - čtverec střední chyby (mean-squared error, MSE), a je definována dle rovnice 19.

$$MSE(\Theta) = (\overline{\Theta} - \Theta)^2 \tag{18}$$

$$MSE(T) = \sum_{i=1}^{N} (\overline{\Theta_i} - \Theta_i)^2 \quad ,(\forall \Theta \in T) \tag{19}$$

Symbol Θ značí třídu, do které byl vstupní obraz klasifikován, a $\overline{\Theta}$ pak jeho žádanou klasifikaci. V grafech je použita chyba akumulovaná pro celou testovací množinu T o velikosti *N*, kde pro *N* byla použita velikost 300.

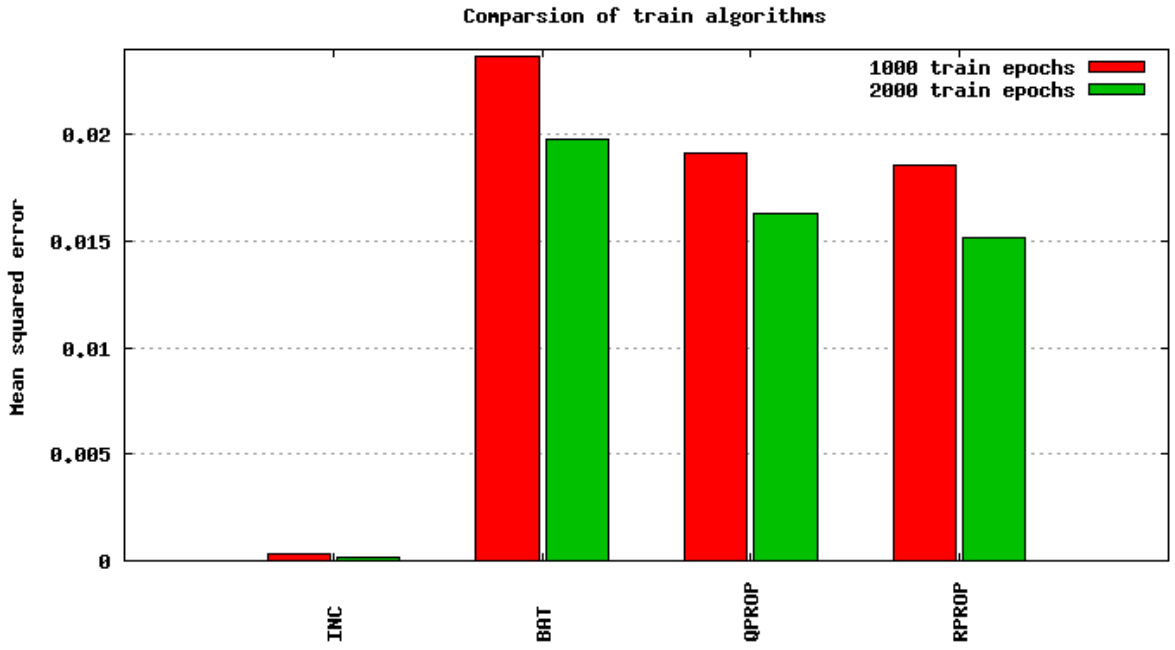
Důležitým bodem je správná interpretace těchto grafů k pochopení výsledků testů. Lze vidět, že chyba pro množinu nonfaces je poměrně malá a značně nezávislá na velikosti trénovací množiny. Lze říci, že klasifikace obrazů neobsahující obličej je celkem přesná. Vysoký rozptyl však lze vidět u MSE pro obličeje. V první třetině většiny grafů je chyba poměrně vysoká, neuronová síť se obličeje snaží klasifikovat jako nonfaces. Klesající charakter této funkce znamená, že s rostoucí velikostí trénovací množiny se chyba klasifikace zmenšuje. Dále lze vidět kmitání této funkce. Toto je způsobeno variabilitou dat v trénovací a testovací množině CBCL databáze. Při použití jiné podmnožiny dat na trénink i testy vypadají

výsledky velmi podobně, kmitání je stále přítomno. Způsobeno může být vysokou odlišností právě trénovaných vzorků a testovací množiny. Této vlastnosti se obecně dá zbavit jen velmi těžko, hlavním hlediskem pro posouzení úspěšnosti aktivačních funkcí bude jejich klesající charakter a její derivace (rychlost změny).

Neuronové sítě použité k získání výše uvedených výsledků byly trénovány standardním způsobem, tzn. na trénovací podmnožině databáze CBCL a testovány na její testovací podmnožině. Pro učení byl použit algoritmus RPROP. Síť měla jednu skrytou vrstvu se 60 neurony. Velikost trénovací množiny nonfaces byla dvakrát větší než velikost množiny faces.

4.3.2 Srovnání trénovacích algoritmů

Stejným způsobem a opět pomocí utility FANNTool jsem provedl srovnání dostupných trénovacích algoritmů neuronové sítě. Výsledky tohoto srovnání jsou na obrázku 18. Pro srovnání trénovacích algoritmů byla použita sigmoidální aktivační funkce neuronů ve skryté vrstvě a sigmoidální aktivační funkce ve výstupní vrstvě. Velikost skryté vrstvy byla 60 neuronů.



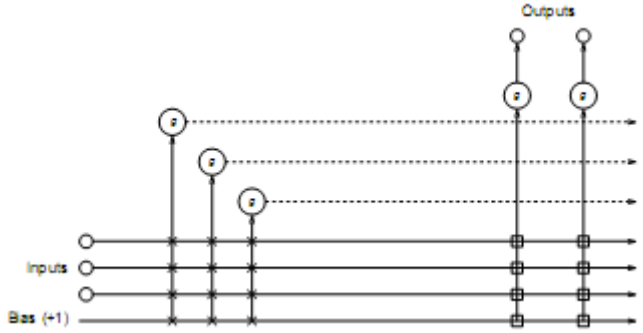
Obrázek 18: Srovnání učících algoritmů.

Zkratka	Celý název v angličtině
INC	Incremental. Standard Backpropagation.
BAT	Batch. Standard Backpropagation.
RPROP	Resilient Backpropagation.
QPROP	Quick Propagation.

Tabulka 2: Význam zkratek označení algoritmů použitých v obrázku $.

Inkrementální algorimus (Incremental - INC) používá standardní algoritmus zpětného šíření chyby, avšak oproti jiným vyčísluje chybu a přehodnocuje váhy po každém trénovacím vzorku dat. Při vysoké různorodosti dat tedy dochází k přehodnocení velmi často, což může činit značné problémy. Dávkový algoritmus zpětného šíření chyby pracuje podobně, ale k přehodnocení vah dochází po vyčerpání všech vzorků celé trénovací množiny, kdy je použita akumulovaná chyba počítaná pro celou validační množinu. Algoritmus RPROP (Resilient Back Propagation) používá adaptivní nastavení některých vnitřních parametrů v závislosti na průběhu testu, jinak se jedná pouze o vylepšenou verzi dávkového algoritmu. Jedná se o jeden z nejrychlejších algoritmů vůbec, jeho nejvýznamnější charakteristikou je, že k přehodnocení vah nepoužívá velikost chyby, pouze její znaménko. Algoritmus rychlého zpětného šíření chyby (Quick Back Propagation - QPROP) je upravená verze dávkového algoritmu, pracuje s větším množstvím vnitřních parametrů, jejich správné nastavení je však velmi složitým problémem. Komplexnost použitých dat je dosti velká a jejich analýza poměrně složitá, a právě i z tohoto důvodu je algoritmus RPROP výhodný, neboť nevyžaduje důkladnou znalost uživatele v oblasti neuronových sítí.

Učení pomocí zpětného šíření chyby je běžnou a ověřenou praktikou v oblasti neuronových sítí, FANN však nabízí ještě jeden zajímavý přístup k trénování, konkrétně kaskádové přidávání neuronů do skrytých vrstev. Jedná se o implementaci algoritmu Cascade2, jejíž základní princip je znázorněn na obrázku 19. Je náhodně vybráno několik neuronů, které jsou jednosměrně spojeny s některými ze skrytých vrstev. Trénování sítě pak spočívá v trénování takto přidaných neuronů (přepočítání vah), zatímco váhy všech ostatních neuronů jsou v této fázi konstantní (přesněji řečeno, váhy neuronů v síti jsou zmrazeny po dobu trénovaní kandidátních neuronů). Poté je z nově přidaných vybrán neuron, jehož přidáním se chyba klasifikace nejvíce sníží, a ten je pevně zařazen do příslušné vrstvy, zatímco ostatní nově trénované neurony jsou zapomenuty. Myšlenkou tohoto přístupu je zlepšení úspěšnosti řešení lokálních podproblémů za účelem zlepšení řešení celého globálního problému.



Obrázek 19: Kaskádové přidávání neuronů do skrytých vrstev.

Hlavní výhodou, kterou kaskádové učení nabízí, je snadnost použití. Uživatel není nucen předem odhadovat parametry sítě, všechny parametry jsou nastaveny automaticky během učení na jejich nejvýhodnější hodnotu. Tento způsob učení je však výhodný pouze pro některé druhy problémů. Právě možnost vytvářet sítě s velkým počtem skrytých vrstev a z toho plynoucí možnost zpracování i vysoce nelineárních dat s sebou přináší riziko přílišné specializace na trénovací data a následně nemožné schopnosti generalizovat testovací data – říkáme, že síť je přeučená (angl. Overfitting).

Při testování kaskádového učení jsem musel přímo čelit výše zmíněnému problému. Jak ukazuje tabulka 3, chyba na datech, která jsou použita ke trénovaní neuronové sítě je velmi nízká a má klesající charakter a při porovnání s testy publikovanými výše se zdá být naprosto ideální. Avšak na datech testovacích, která jsou použita pouze pro testy, je chyba pro množinu faces velmi vysoká, a nemá snižující se tendenci. Neuronová síť není schopna generalizovat a použití kaskádového učení pro tento problém přímo vylučuje. Lze též vyčíst, že síť se snaží příliš velké procento obličejů klasifikovat jako nonfaces, což je pro účel této práce kritické. Raději bychom přijali řešení nabízející vyšší míru špatně klasifikovaných neobličejových dat (false positives), s nutností dalšího post-processingu vedoucího k částečné eliminaci duplikátů. Vzhledem ke značně rozdílným rozsahům hodnot pro jednotlivé sloupce dokonce nebylo možno smysluplně tyto výsledky reprezentovat grafem. FANN nenabízí možnost kaskádové učení přímo řídit, pouze definovat podmínky pro jeho ukončení. Bylo tedy nutno provést několik trénovacích cyklů, popořadě s jedním až několika přidanými neurony, zastavit učení po každém přidaném neuronu a provést test chyby na trénovací a testovací podmnožině CBCL, což vysvětluje nezarovnaná čísla ve sloupci pro počet trénovacích epoch. Ačkoliv to není z níže uvedené tabulky zcela zřejmé, každý přidaný neuron chybu na trénovacích datech skutečně zmenšil. Výkyvy jsou způsobeny nutností pro každý řádek provést autonomní test, přičemž váhy jsou na začátky nastaveny vždy na jinou náhodnou hodnotu. Počet trénovacích epoch znamená, že pro daný počet neuronů byl trénink zastaven, protože nedocházelo ke zmenšení chyby a bylo nutno přidat další neuron.

Počet neuronů ve skrytých vrstvách	Počet trénovacích epoch	MSE – Trénovací množina, nonfaces	MSE – Trénovací množina, faces	MSE – Testovací množina, nonfaces	MSE – Testovací množina, faces
1	358	14,88	18,39	8,11	195,38
2	612	8,97	10,53	3,23	262,26
3	794	7,47	9,68	2,14	232,74
4	1109	8,19	6,81	1,82	213,08
5	1340	5,64	3,97	1,24	235,45
6	1716	6,97	4,53	1,54	248,76

Tabulka 3: Srovnání hodnot MSE pro dvě testovací množiny s použitím kaskádového učícího algoritmu.

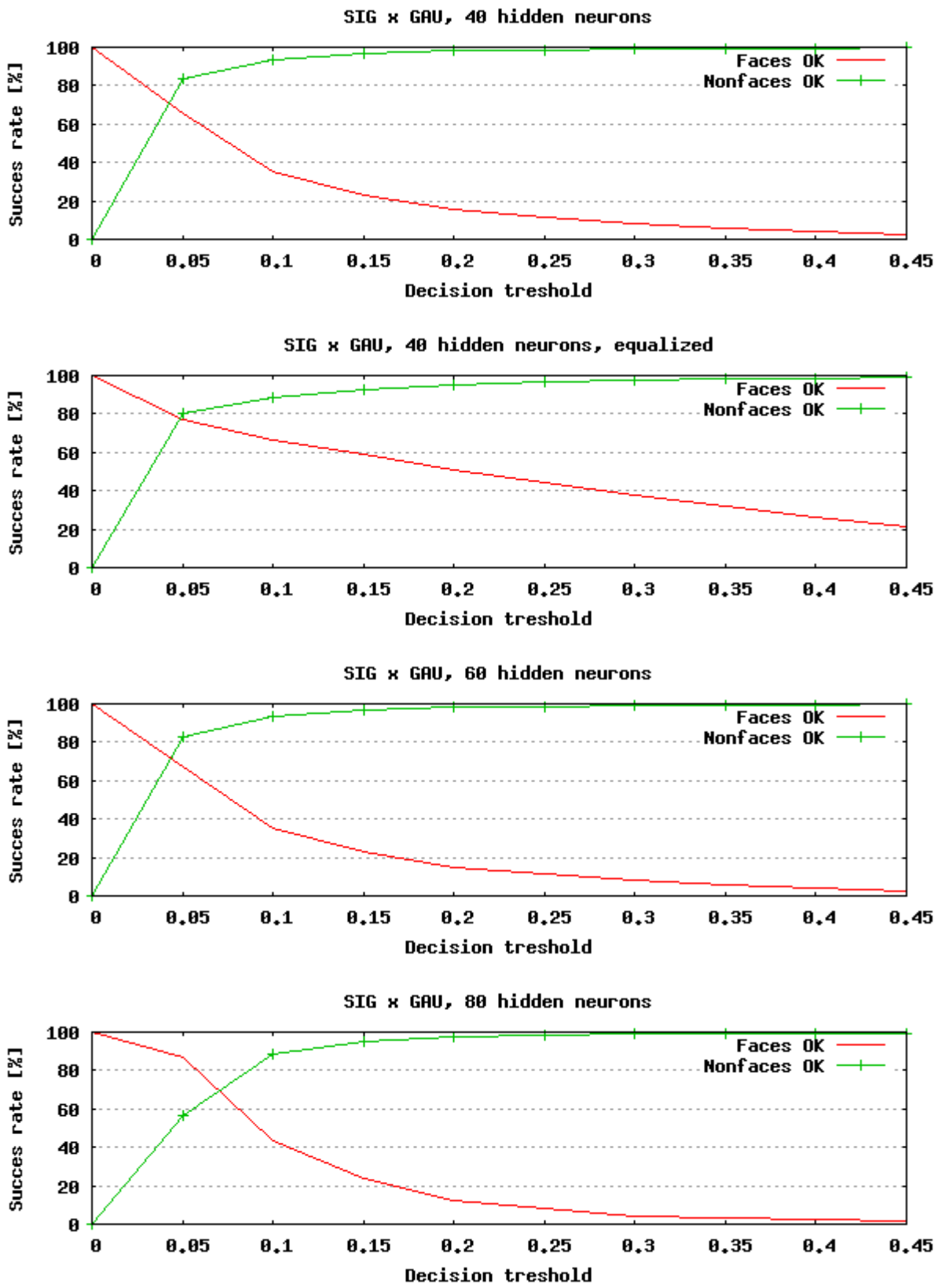
Při popisu výše uvedených algoritmů jsem čerpal převážně z dokumentace ke knihovně FANN [8] a tuto mírně upravil. Detailní informace o Cascade2 algoritmu a podobných přístupech lze nalézt v [6]. Další možné, více pokročilé techniky použití neuronových sítí budou diskutovány na závěr této práce.

Implementace trénovacího modelu je zapouzdřena ve třídě PureNNTrainer. Ta obsahuje kromě metod init() a destroy(), metod pro změnu parametrů trénovaní (metody s prefixem set…), čtyři důležité metody. První z nich je run(std::string &), která zajistí trénink sítě tak, aby nebyl překročen počet trénovacích dat a zároveň aby bylo dosaženo požadované maximální chyby. Trénovaní končí porušením jedné z výše uvedených podmínek. Další metodou je run(std::string &, const int), která zajistí, že přesně daný počet trénovacích epoch je vyčerpán (kromě případu, kdy chyba během trénování klesne na nulu – zde je z podstaty zajištěno, že se nejedná o lokální extrém). Třetí metodou je trainSpecial(std::string &, std::string &). Jedná se o speciální způsob trénování, kdy po několika málo epochách, přesněji řečeno po jednom vzorků obličejových data a jednom až čtyřech vzorcích neobličejových dat (záleží na nastavených velikostech trénovacích množin), je spuštěna zpětná validace (Cross Validation) pro celou testovací množinou a vypočítána chyba (MSE). Můžeme tak přesně sledovat vývoj chyby po libovolném množství trénovacích dat a lépe tak odhadnou čas, kdy je neuronová síť ideálně natrénována pro dané nastavení. Visualizaci výstupů této metody lze vidět na obrázcích 15 az 17. Poslední metodou je cascadeInternal(void), která používá pro trénování algoritmus Cascade2. Tato metoda je neveřejná a přístupná pouze ze třídy PureNNTester, protože vyžaduje mírně odlišný přístup k nastavení parametrů. Výstupem všech zmíněných metod je kromě informací o průběhu testování též konfigurační soubor pro FANN s natrénovanou sítí. Pro přesné umístění a pojmenování toho souboru je nutno nahlédnout do programové dokumentace přiložené k diplomovému projektu ve formátu html.

Proces trénování je řízen ze třídy PureNNTester. Manuální změnu parametrů lze provést právě zde a právě zde se očekává největší aktivita následníků této práce v oblasti experimentů s nastavením a s změnou průběhu a struktury testů a trénovaní. Třída obsahuje větší počet metod pro různé průběhy testů, pro jejich přesný obsah je nutno nahlédnout do dokumentace, případě do implementace těchto metod. Pro testování natrénované sítě slouží třída NNExecutor, která zajišťuje různé módy exekuce testovacího vzorku dat, případně celé testovací množiny pro neuronovou síť, jejíž konfigurační soubor nebo reference na objekt neuronové sítě je předána do metod této třídy. Opět se jedná pouze o zapouzdření volání knihovny FANN a pro přesný popis činnosti jednotlivých metod je potřeba nahlédnout do programové dokumentace.

4.3.3 Pokročilé testy

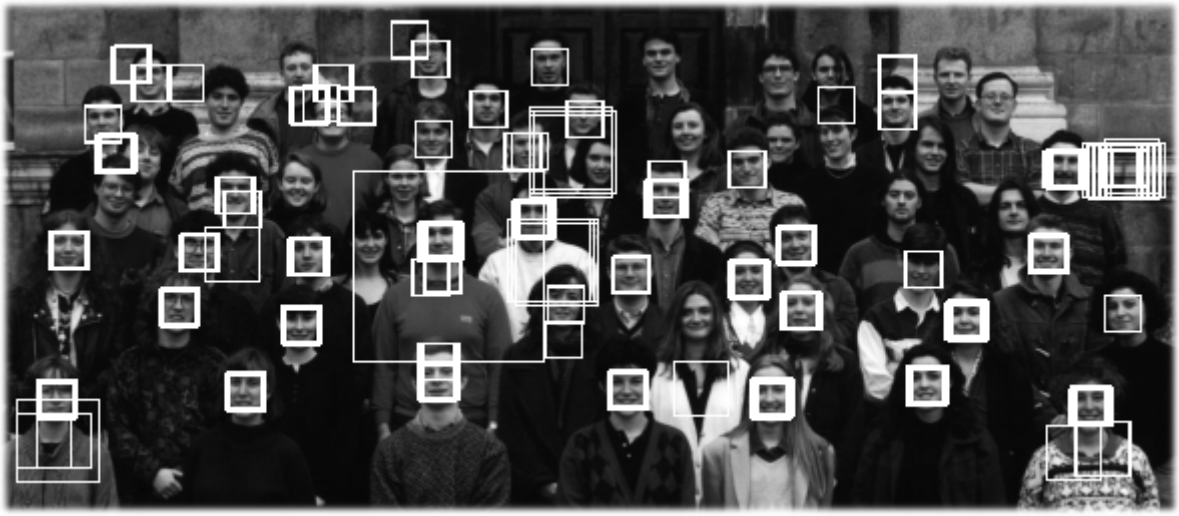
V předchozím textu jsem popisoval provedené testy pro jednotlivé parametry sítě a nyní jsme ve fázi, kdy máme dostatek znalostí o chování neuronové sítě a pokusíme se provést rozsáhlejší testy. Nejprve jsem se pokusil odhadnou ideální množství neuronů ve skryté vrstvě. Následující obrázky ukazují značnou vyrovnanost při použití 40, 60 a 80 neuronů. Pro testy jsem použil 470 tváří testovací množiny CBCL (bohužel k dispozici je jich pouze 472) a 10.000 neobličejových dat. Pro trénovaní bylo použito 2.200 obličejů a 4.400 neobličejů. Proběhly i testy s 1.100 obličeji a 4.400 neobličeji, ale síť nedokázala správně reagovat na testovací obličeje, poměr 1:2 se ukázal jako jednoznačně výhodnější. V této fázi jsem měl opět problémy s knihovnou FANN a učícími dávkovými algoritmy, které nejsou schopny takto velké množství trénovacích dat zpracovat. Zde se projevil určitý paradox, totiž že síť ke správnému natrénovaní potřebuje velké množství dat, ale pro tak různorodou množinu, jakou CBCL jistě je, algoritmy jako RPROP zcela selhávají. Bylo tedy nutno převzít kontrolu nad trénovacím procesem, trénink provádět po jednom vzorku a vždy přepočítat váhy. Tento postup jsem zpočátku zavrhl, neboť první myšlenky na něj evokují malou rychlost a nízkou odolnost proti lokálním extrémům. Na velkých testovacích množinách se však ukázalo, že opak je pravdou. Tento postup je několikrát rychlejší i přesnější, proto bude dále použit. Byla použita sigmoidální aktivační fukce ve skryté vrstvě a gausova funkce ve výstupní. První a druhý graf znázorňují stejnou síť, ovšem v druhém případě jsou použita ekvalizovaná data. Lze vidět nárůst správně detekovaných obličejů v řádu desítek procent, avšak úbytek správně detekovaných neobličejů. Ekvalizace se zdá býti výhodná, avšak pro reálné testovací snímky (např. skupinové fotografie) nemusí přinášet nutně zisk, jak uvidíme dále.



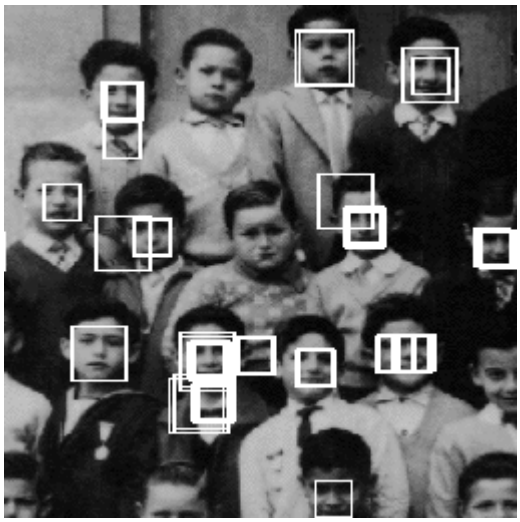
Obrázek 20: Porovnání úspěšnosti klasifikace při různém počtu skrytých neuronů. Výstup neuronové sítě je v intervalu <0,1>.

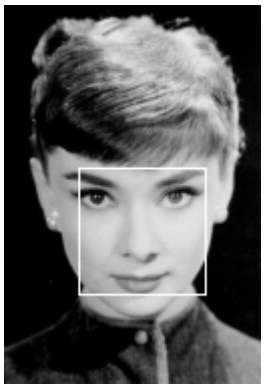
Na závěr této části jsem provedl několik testů na skupinových fotografiích. Lze vidět poměrně vysoká úspěšnost, daná často individuálním nastavením prahu. To ovšem není

chybou, pouze se jedná o přizpůsobení různému charakteru vstupních dat. Nutnost použít poměrné nízkou hodnotu prahu je z velké části dána průběhem gausovy funkce ve výstupní vrstvě. Povšimnout si též lze, že velká část nedetekovaných obličejů je pootočena kolem osy *z*, na což není tento systém stavěn. Práh je nutno volit velmi opatrně, neboť například první obrázek obsahuje přibližně 230.000 klasifikovaných oken a i 1% špatně klasifikovaných obličejů (false positives) způsobí zakreslení více než dvou tisíc čtverců, což činí snímek absolutně nečitelným. Pro následující obrázky nebylo možné použít ekvalizaci kvůli její schopnosti zvýraznit vzory a následné velké množství falešných detekcí.



Obrázek 21: Lokalizace obrázku testset/big/class3.pgm pomocí 40 skrytých neuronů, práh nastaven na 0,4.





Obrázek 22: Vlevo lokalizace obrázku testset/big/nensb.pgm, 40 skrytých neuronů, práh 0,45, celkem 97.000 klasifikovaných oken. Vpravo testset/big/audrey3.pgm, 40 skrytých neuronů, práh 0,3, 30.000 klasifikovaných oken.

5 Vlastní tváře

5.1.1 Výpočet vlastních tváří

Pro výpočet jsem zvolil knihovnu GNU Scientific Library (dále jen GSL). Tato knihovna je volně dostupná na stánkách [11] a je stejně jako FANN distribuována pod licencí GPL, kterou lze v nejnovější verzi nalézt na [13]. GSL je velmi široce využívána, dokonce i v profesionálních nástrojích pracujících s vědeckými výpočty. Proto i já jsem se rozhodl využít jejích vlastností. Knihovna není nativně určena pro systémy na platformě Microsoft Windows, která byla zvolena pro tuto práci, a překlad knihovny je tedy poněkud komplikovanější. Rozhodl jsem se tedy přeloženou knihovnu přiložit jako část diplomové práce, aby čtenáři, kteří se rozhodnou se zdrojovými kódy manipulovat, měli ulehčenou pozici. K projektu je nutno přilinkovat soubory gsl.lib a cblas.lib, které používají standardní multivláknovou dynamickou knihovnu (multithreaded DLL), překlad zbytku projektu je tedy nutno této skutečnosti přizpůsobit, aby nedocházelo ke konfliktům. Knihovna nebyla nijak pozměněna, její použití a následná distribuce probíhá v souladu s licenčním ujednáním. Čtenář samozřejmě může překlad vyřešit ve vlastní režii pomocí nástrojů jako je CygWin apod.

Výpočet vlastních tváří probíhá ve třídě PCAPerformer. Ta má pouze jednu veřejnou metodu run(std::string &), jejímž parametrem je skript obsahující cesty k adresářům a souborům, které jsou při výpočtu použity. Celý výpočet je tak zapouzdřen a neklade na uživatele nároky na znalost matematického pozadí této operace.

Prvním stupněm výpočtu vlastních tváří je výpočet kovarianční matice, jak je definována výše. Tato matice je pak vstupem do GSL, další výpočty probíhají již v její režii. Knihovna GSL používá pro výpočet vlastních vektorů několik algoritmů, já jsem použil výpočet pomocí bidiagonalizace vstupní matice a následné QR redukci. Použití této rychlejší metody jsem si mohl dovolit na základě znalosti, že kovarianční matice je čtvercová, symetrická a obsahuje pouze reálná čísla. Tímto odpadla nutnost použít obecnější, avšak pomalejší a robustnější výpočetní postup. Vypočítané vlastní vektory jsou seřazeny dle odpovídajících vlastních čísel sestupně a uloženy do souborů specifikovaných uživatelem ve vstupním skriptu. Zároveň s nimi je uložen vektor vypočítaný jako průměr vstupních vektorů, dále bude označován jako průměrná tvář. Průměrná tvář je potřeba při dalších výpočtech.

5.2 Lokalizace na základě vzdálenosti

Vlastní tváře nejsou předurčeny pro použití pouze s neuronovými sítěmi, vlastní klasifikátor může mít téměř libovolnou formu. Já jsem se pokusil o experiment

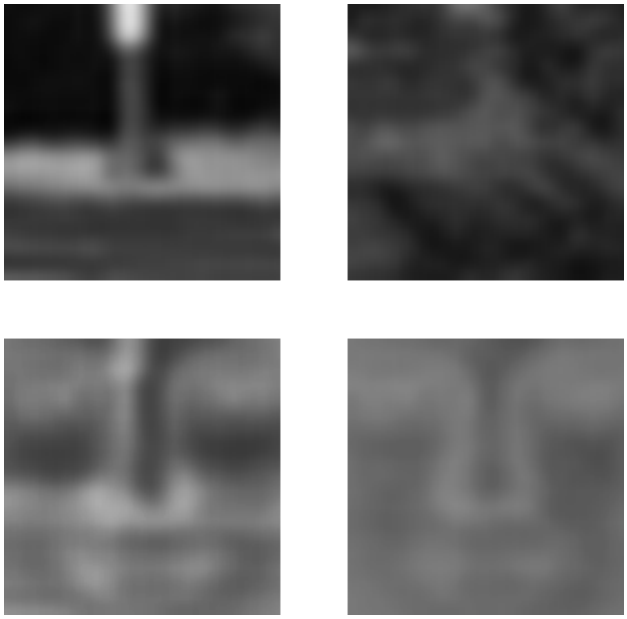
s klasifikátorem nejjednodušším, který je schopen rozdělit data do dvou množin na základě vzdálenosti – provádějící tzv. dichotomii. Klasifikátor pracuje velmi jednoduchým způsobem. Před vlastní klasifikací předpokládá, že na vstupu je obraz tváře, a každá tvář může být reprezentována jako lineární kombinace vlastních tváří. Na základě tohoto předpokladu vypočítá váhy tohoto vstupního obrazu k vektoru vlastních tváří dle rovnice 21. Vstupní tvář je pak možno zrekonstruovat pomocí znalosti vlastních tváří a vah vstupního obrazu k nim dle rovnice 20. Posledním krokem je pak zjištění vzdálenosti tváře vstupní od tváře zrekonstruované. Pokud je tato vzdálenost menší než nastavený práh, je obraz klasifikován jako tvář, v opačném případě je označen jako obraz neobsahující obličej. Tyto pojmy budeme v následujícím textu používat velmi často a vzhledem k neexistenci výstižných ekvivalentů v českém jazyce budeme používat zažité anglické pojmy face a nonface.

$$\Phi - \Psi = \sum_{i=1}^{k} w_i u_i \tag{20}$$

$$w_i = u_i^t \Phi \tag{21}$$

kde Ψ je průměrná tvář, Φ vstupní obraz, u_i je i-tá vlastní tvář z množiny eigenfaces.

Obrázek 23: Ukázka dvou tváří (nahoře) a jejich obrazu po rekonstrukci pomocí množiny eigenfaces.



Obrázek 24: Ukázka dvou obrazů neobsahujících tvář (nahoře) a jejich obrazu po rekonstrukci pomocí množiny eigenfaces.

Na obrázku 23 a 24 je příklad obrazu z množiny faces, resp. nonfaces. V prvním případě lze vidět celkem věrnou rekonstrukci vstupního obrazu, i pouhým okem lze snadno rozeznat tvář v obraze. V případě druhém lze též vidět náznak tváře v rekonstruovaném obraze, podobnost tváři je však již mnohem menší. Zrekonstruované tváře vykazují značnou monotónost barev, převažuje šedá s hodnotou blízko středu spektra šedi a velmi malým rozptylem. To je způsobeno normalizací obrazu, interní reprezentace obrazu jsou poměrně malá čísla. Visualizace takovýchto obrazů není jednoduchou záležitostí, musel jsem přistupovat k obrazům velmi individuálně. Export interní reprezentace obrazového vektoru do souboru obrazového formátu pgm je ve třídě Pgm_IO, konkrétně metoda saveNormalizedDouble(). Klasifikace pomocí vzdálenosti pracuje s desetinnými čísly s dvojitou přesností na rozdíl od klasifikace neuronovými sítěmi. Pro visualizaci tváře reprezentované čísly s jednoduchou přesností je potřeba nejprve provést konverzi.

Logika pro klasifikace na základě vzdálenosti je implementována ve třídách DistanceClassifier a DistanceTester. Zatímco první výše zmíněná třída slouží k zapouzdření matematických výpočtů, druhá zmíněná poskytuje logiku pro testování a poskytuje rozhraní k různým druhům testů. Protože tato implementace je skutečně jen experimentální, jako metrika pro výpočet vzdálenosti byla použitá Euklidovská vzdálenost definovaná dle rovnice 25. Jisté zlepšení úspěšnosti klasifikace by mohlo přinést použití jiné metriky, například Mahalanobisovy vzdálenosti, která je definována dle rovnice (22). Její výhodou je, že zachází s variacemi kolem všech použitých os jako se stejně významnými [7].

$$\left\|\Omega - \Omega^k\right\| = \sum_{i=1}^{k} \frac{1}{\lambda_i}\left(w_i - w_i^k\right)^2 \tag{22}$$

Výpočet vzdálenosti pak probíhá následujícím způsobem pro vstupní obraz Γ:

$$\Phi = \Gamma - \Psi \tag{23}$$

$$\Phi^r = \sum_{i=1}^{k} w_i u_i \tag{24}$$

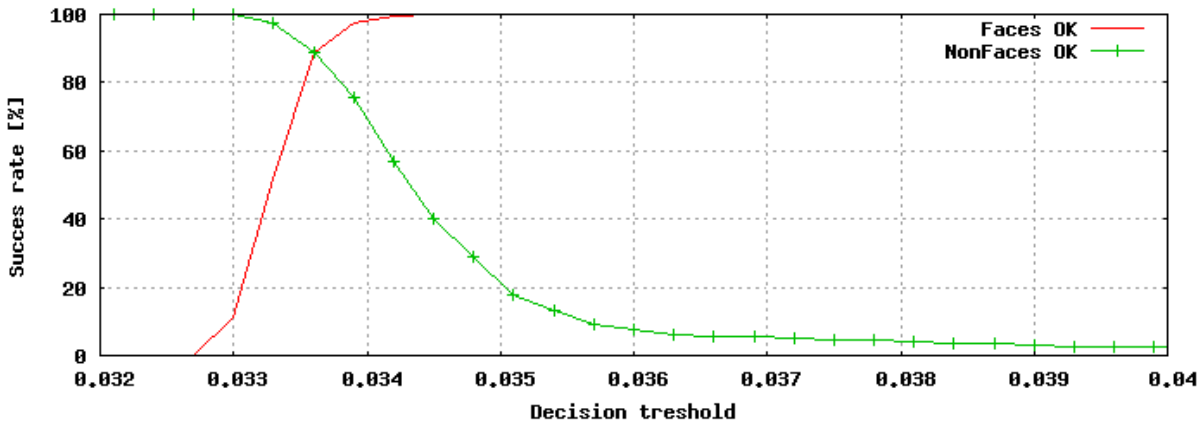
$$e_d = \left\|\Phi - \Phi^r\right\| \tag{25}$$

$$\text{pokud } e_d < T_d \text{ pak } \Gamma \in faces \tag{26}$$

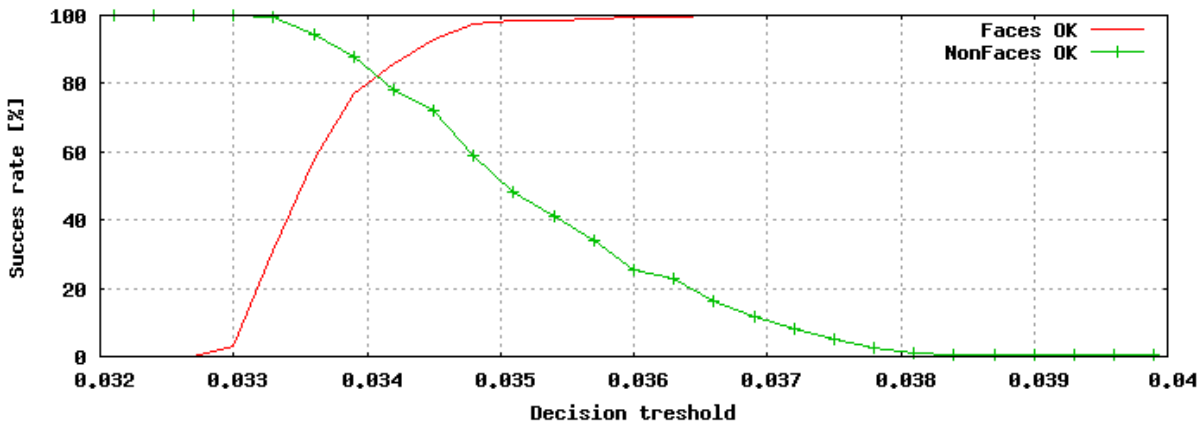
písmenem Ψ je označena průměrná tvář, písmenem Φ^r tvář zrekonstruovaná, e_d je vzdálenost tváře vstupní od zrekonstruované a T_d je práh pro klasifikaci.

5.2.1 Testovací model

Klasifikace založená na vzdálenosti nemá žádnou fázi trénovaní, avšak potřeba odladit vnitřní parametry systému je zde též přítomna. Prvním testem, který bylo nutné provést, byl odhad rozložení vzdálenosti množin faces a nonfaces od svých ekvivalentů zrekonstruovaných pomocí množiny vlastních tváří. Na základě tohoto rozložení je pak možno určit práh pro rozhodnutí, do které množiny bude vstupní obraz klasifikován. Grafické znázornění závislosti úspěšnosti klasifikace na hodnotě prahu pro dichotomii je vyobrazeno na obrázcích 25 a 26.



Obrázek 25: Závislost množství správně klasifikovaných vzorů pro část testovací množiny obličejové databáze CBCL.



Obrázek 26: Závislost množství správně klasifikovaných vzorů pro část trénovací množiny obličejové databáze CBCL.

Z obrázků 25 a 26 lze vyčíst, že průměrná vzdálenost vstupu od jeho zrekonstruovaného obrazu je relativně malá. To je ovšem způsobeno normalizací vstupního vektoru. Zde je nutno si uvědomit, že střední hodnota barvy (stupně šedi) při načítání je 127, vektor o délce 361 tak má průměrnou velikost 2413 a střední hodnota obsažená v normalizovaném vektoru je pak přibližně 0,0526. Z této skutečnosti plyne, že i vzdálenosti budou malá čísla, což ovšem není problém dokonce i pro použití pro čísla s jednoduchou přesností, pokud si představíme způsob interní reprezentace těchto čísel. Čísla s jednoduchou přesností jsou použita pouze pro reprezentaci vstupních obrazů, veškeré výpočty jsou prováděny s dvojitou přesností. Problémem ovšem může být stabilita systému, který používá takto malé diference pro klasifikaci. Proto jsem provedl další test na stabilitu s použitím jiné testovací množiny dat, konkrétně podmnožiny databáze CBCL, která je určena primárně pro trénování neuronové sítě (obrázek 26). Zde žádný trénink proveden není, proto můžeme trénovací množinu použít na testování. Obě množiny by měly být navzájem disjunktní, není však téměř možnost vzhledem velikosti a povaze testovacích dat tuto skutečnost ověřit. Zároveň bylo nutno zajistit, aby žádná z množin neobsahovala data, ze kterých byla vypočítána množina eigenfaces. Tato skutečnost by měla za následek poměrně přesnou rekonstrukci testovacích obrazů a tím i značné zkreslení výstupů systému. Z obrázků 25 a 26 lze velmi podobný průběh chyby, což by mělo mít za následek snadnější nalezení prahu pro rozhodnutí. Zároveň je nutno povšimnout si velmi strmého průběhu křivek. Vzhledem k tomu, že se jednalo stále o jednu konkrétní obličejovou databázi, lze předpokládat, že použité obrazy nezávisle na jejich příslušnosti do testovací či trénovací množiny jsou si velmi podobné. Je tedy nutno ověřit výsledky klasifikace na datech zcela rozdílných, nejlépe na skupinových fotografiích.

Při pokusu odladit systém na skupinových fotografiích se mi však nepodařilo nalézt vhodný práh pro rozhodnutí tak, aby bylo možno systém vůbec považovat za použitelný.

Důvod, proč systém vykazuje dobré výsledky na obličejových databázích, avšak není schopný správně klasifikovat komplexnější vzorky, je nutno hledat v samotném matematickém modelu. Obrazy rekonstruované pomocí vlastních tváří jsou poměrně nezávislé na intenzitě vstupního snímku. Intenzita snímků v trénovacích i testovacích množinách je naprosto nesrovnatelná s intenzitou oken v komplexních snímcích, přesněji řečeno její variabilita. Při použití výpočtu dle rovnice 25 lze však již při pouhém pohledu odhadnout zcela rozdílné výsledky pro hodně tmavé a hodně světlé obrazy. Výsledkem testů bylo příliš velké množství falešných detekcí (false positives). Například při testech na relativně malém snímku, u kterého se klasifikuje 100.000 oken, i jedno procento falešných detekcí způsobí absolutní nečitelnost obrazu, a přes zanesené detekované obličej není snímek čitelný. Tento způsob klasifikace je tedy zcela nepoužitelný. Spokojíme se s faktem, že tento přístup je krátce zdokumentován, pro zlepšení jeho výsledků je nutno nalézt algoritmus invariantní vůči intenzitě vstupního snímku. Zde si již nevystačíme s úpravou vstupních dat jako je ekvalizace (která skutečně zlepšení nepřinesla), je nutno použit mnohem robustnější klasifikátor schopný variabilitu vstupních obrazů potlačit, či vůbec se surovými vstupními obrazy v klasifikační fázi nepracovat. Takovýmto klasifikátorem může být právě neuronová síť.

5.3 Klasifikace pomocí neuronových sítí s použitím vlastních tváří

V předchozí kapitole jsme si představili velmi jednoduchý systém klasifikující na základě vzdálenosti. V této kapitole se budeme zabývat pokročilejšími technikami klasifikace pomocí neuronových sítí s použitím množiny vlastních tváří. Pro tuto fázi jsem navrhl dva možné přístupy k využití vlastních tváří, které zde budou popsány. Pro jednoduchost se pokusím dodržet podobnou strukturu jako je použita v předchozích kapitolách, aby bylo možno srovnávat výsledky v jednotlivých fázích.

5.3.1 Lokalizace pomocí vah k vlastním tvářím

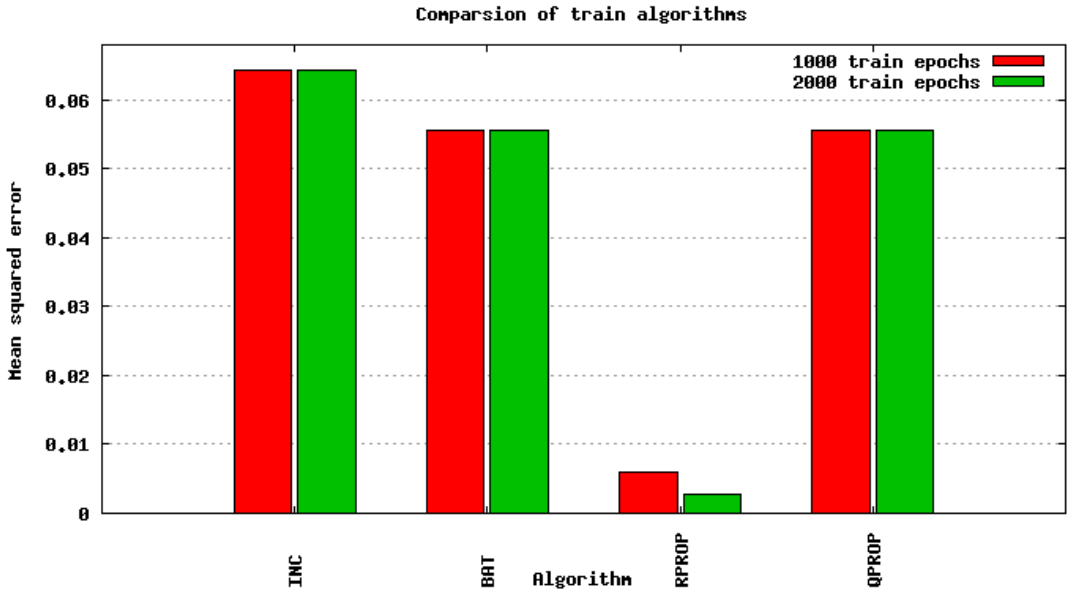
Tento způsob klasifikace spočívá v možnosti výpočtu vah k vektoru vlastních tváří jak již byl nastíněn v rovnici 21. Vlastní tváře jsou obecně množinou, ale pro účel této práce je budeme reprezentovat vektorem, seřazeným dle vlastního čísla odpovídajícího příslušné vlastní tváři. Pomocí tohoto vektoru vah je možné vstupní vzorek s velkou mírou přesnosti

zrekonstruovat. Vektor vah použijeme pro vstup neuronové sítě. V teoretické rovině však zachováme matematicky korektní pojem množina vlastních tváří.

5.3.2 Trénovací model

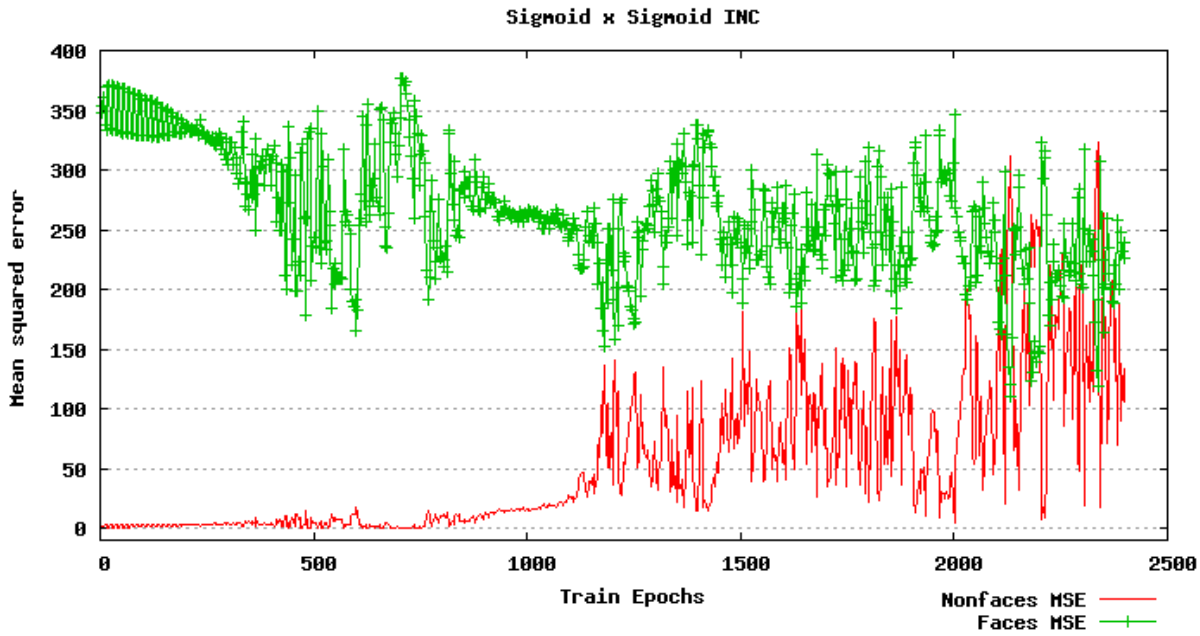
Pro prvotní odhad parametrů sítě jsem opět použil utilitu FANNTool. Vytvořil jsem soubor se 2400 vzorky dat z trénovací podmnožiny databáze CBCL, která byla použita pro zjištění optimálních aktivačních funkcí neuronové sítě. Již zde jsem však narazil na první problém. Některé kombinace aktivačních funkcí naprosto zablokovaly proces učení a chyba tak ani po vyčerpání všech trénovacích vzorků prakticky neklesla. Jiné kombinace způsobily strmý pád aktuální chyby v několika málo stovkách epoch až k hodnotě nula. Získaná data proto není možno nijak rozumně vizualizovat, a tento časově velmi náročný test bohužel nepřináší žádný poznatek, který by bylo možno zhodnotit v prvotní odhad nastavení neuronové sítě.

Při použití FANNToolu pro odhad optimálního učícího algoritmu jsem rovněž nedosáhl očekávaných výsledků. Obrázek 27 ukazuje hodnotu chyby po 1000 a 2000 epoch. Lze vidět, že jediný algoritmus RPROP vykazuje snížení chyby po dvojnásobném počtu epoch, zatímco ostatní algoritmy si nejsou schopny s dodanými daty vůbec poradit. Při srovnání s odhadem optimálního algoritmu pro neuronovou síť bez použití vlastních tváří (obrázek 18) lze též vidět řádově vyšší chybu po vyčerpání stejného množství epoch. Pro tento odhad byly použity obdobné parametry sítě, tzn. 60 neuronů ve skryté vrstvě a sigmoidální funkce ve vrstvě skryté i výstupní.

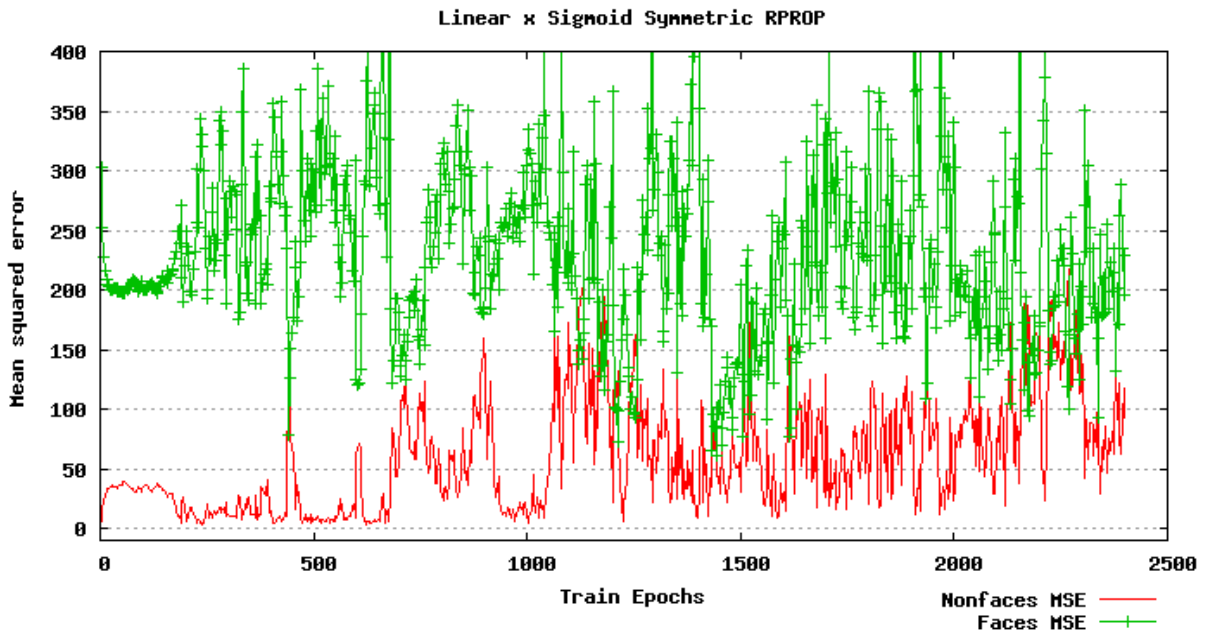


Obrázek 27: Odhad optimálního trénovacího algoritmu pomocí utility FANNTool.

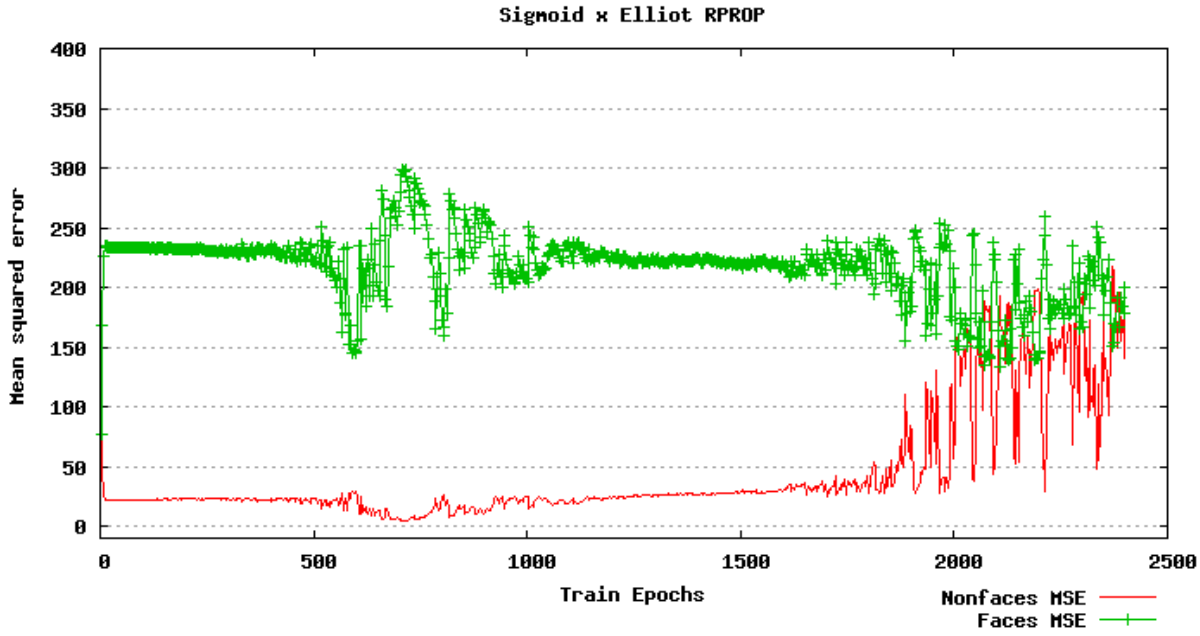
Odhad parametrů pomocí prozkoumání všech možností nastavení úspěšný nebyl, je tedy nutno manuálně tyto parametry nastavit a provést další testy. Pro tuto fázi jsem zvolil stejný postup jako u předchozí neuronové sítě, tedy pomocí akumulované střední chyby v různé fázi učení za pomocí testovací podmnožiny CBCL databáze. Všechny testy probíhaly podobným způsobem. Následují ukázky některých kombinací aktivačních funkcí, které jsem uznal za vhodné umístit do této části práce, některé další lze nalézt v příloze 2. Velikost testovací množiny byla 400 vzorků tváří a stejný počet vzorků neobsahujících obličej.



Obrázek 28: Vývoj chyby při použití sigmoidální funkce ve skryté i výstupní vrstvě a incrementálního učícího algoritmu.

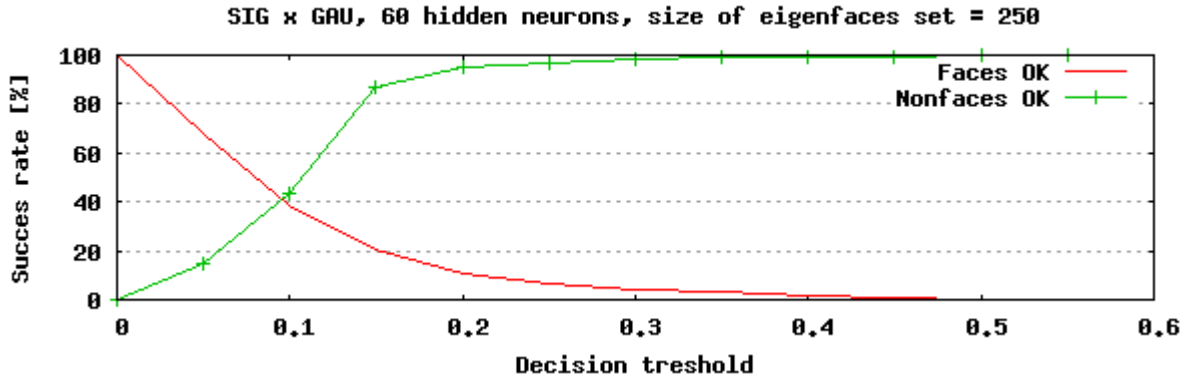


Obrázek 29: Vývoj chyby při použití lineární funkce ve skryté a symetrické sigmoidální funkce ve vrstvě výstupní a učícího algoritmu RPROP.

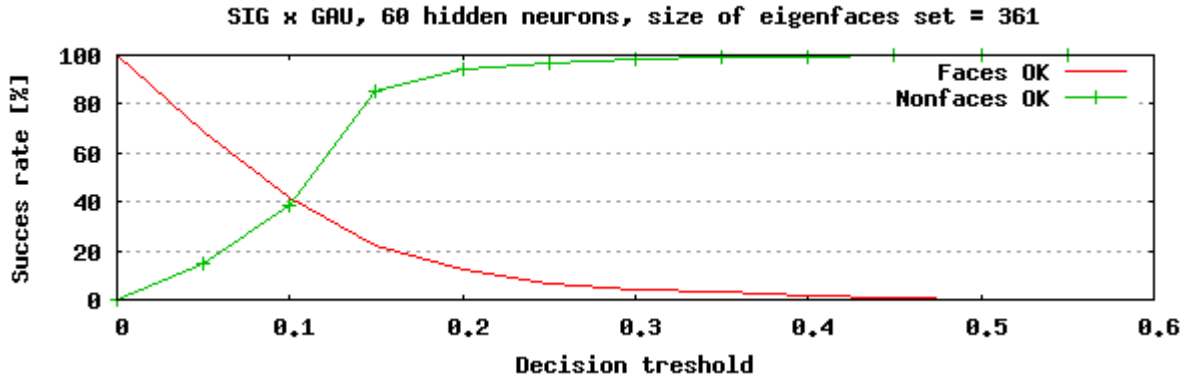


Obrázek 30: Vývoj chyby při použití sigmoidální funkce ve skryté a elliotovy funkce ve výstupní vrstvě a učícího algoritmu RPROP.

Jak lze z uvedených obrázků vyčíst, výsledek trénování je velmi špatný. Chyba ani v jednom případě významně neklesá, ba naopak, jak lze vidět i v dalších grafech uvedených v příloze 2, chyba místy i stoupá. Ve všech případech též velmi silně kmitá. Zajímavé je na začátku trénovaní zakmitání při použití pouze sigmoidálních funkcí, které je velmi podobné průběhu trénování v předchozím testu při použití stejných funkcí. Stejné testy jsem provedl i s jiným počtem neuronů ve skryté vrstvě a menší množinou vlastních tváří, takže vstupní vektory měly jinou velikost. Naneštěstí výsledky byly velmi podobné nebo ještě horší, proto jejich vizualizace nebyla do této práce zařazena. Menší počet použitých vlastních tváří totiž nemá za úkol výsledek klasifikace zpřesnit, ale urychlit.



Obrázek 31: Chyba klasifikace v závislosti na použitém prahu.



Obrázek 32: Chyba klasifikace v závislosti na použitém prahu pro menší množinu vlastních tváří.

Na obrázcích 31 a 32 lze vidět průběh chyby klasifikace v závislosti na prahu pro rozhodnutí, zdali výstup neuronové sítě klasifikovat jako obličej či nikoli. Ačkoli se zdají být oba grafy naprosto shodné, při bližším zkoumání uvidíme mírné odlišnosti. Ideální místo pro nastavení prahu je místo křížení obou křivek. Systém nevykazuje téměř žádné výkyvy pro menší počet použitých vlastních tváří, avšak úspěšnost je velmi nízká. Důvodem, proč chyba pod padesáti procenty je rozdílná velikost testovacích množin (470 obličejových vzorků,

2.000 neobličejových vzorků). Důvodem, proč jsem zde nezařadil srovnání stejně velkých množin, je skutečnost, že toto rozložení je mnohem blíže situaci na skutečném snímku.

Příčinu, proč není možno dosáhnout lepších výsledků při testování, je nutno zřejmě hledat v samotné podstatě vstupních dat do neuronové sítě. Váhy představují pouze multiplikační konstantu pro rekonstrukci vstupního vzorku. Velmi neformálně řečeno, říkají, jak silně ovlivňuje příslušná vlastní tvář rekonstruovaný obraz. Pro dva různé vstupní obličeje tak vzniknou dva různé vektory vah, mezi nimiž nemusí být nutně žádná korelace. Právě tato skutečnost velmi komplikuje použití s neuronovými sítěmi. Principem učících algoritmů je právě automatické nalezení závislostí mezi různými vstupními vzory jednotlivých cílových tříd a odlišností od vzorů ostatních tříd. V případě obličejů nijak neupravených lze i pouhým okem snadno rozeznat základní strukturu obličeje, prostorové i barevné závislosti jednotlivých složek obličeje na ostatních. Toto však již nelze říci o vektoru vah.

Předchozím odstavcem jsem však určitě nechtěl říci, že není možno zkombinovat vlastní tváře a neuronové sítě. Pouze je nutno vzít v úvahu výše zmíněné skutečnosti. Dle mého názoru má tato metoda mnohem větší šanci u značně odlišné obličejové databáze, zejména pak takové, která bude obsahovat vzorky obličejů o výrazně větší velikosti. Pouhým pohledem na vzorky obsažené v CBCL databázi lze vidět značnou variabilitu vzorků a z toho plynoucí značnou variabilitu vektorů vah k vlastním tvářím. Při použití větších snímků by bylo možno použít jen menší počet vlastních tváří pro výpočet vektoru vah, což by mohlo snížit varianci těchto vektorů. Zároveň větší snímky zaručují mnohem větší množinu vzorků pro výpočet vlastních tváří. Analýza hlavních komponent je metodou statistickou a dá se tedy předpokládat, že s větším množstvím vstupních dat bude stoupat i kvalita jejích výstupů.

Implementace tohoto přístupu stejně jako následujícího je obsažena ve třídách NNTrainer, NNExecutor a NNTester. NNTrainer slouží k různému způsobu trénování sítě, hlavní metodou je run(std::string, int), která trénuje síť inkrementálním způsobem pomocí algoritmu Backpropagation a po provedení daného počtu epoch uloží konfiguraci sítě do souboru k dalšímu použití. NNTester obsahuje množství metod pro provedení testů. Poměrně důležitou třídou je EigenfacesHolder, která drží předem spočítané vlastní tváře a umožňuje výpočet vektoru vah nebo rekonstrukci pomocí eigenfaces.

5.4 Lokalizace pomocí rekonstruovaných tváří

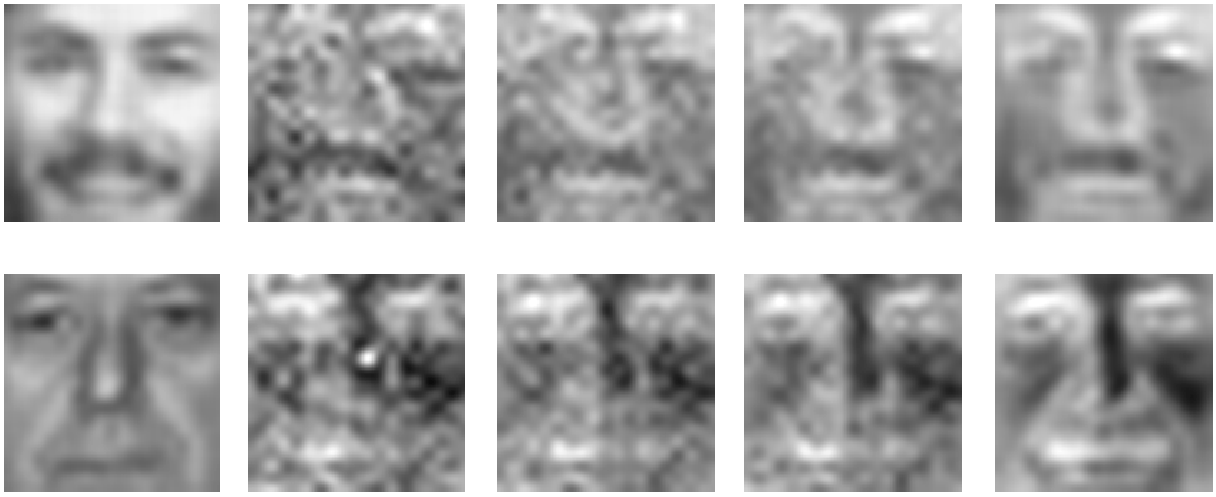
Další možností, jak využít množinu vlastních tváří, je přiložit na vstup neuronové sítě vzorky zrekonstruované pomocí množiny vlastních tváří. Tedy naprosto stejná data, která

jsem používal u klasifikátoru založeném na vzdálenosti. Při pohledu na vzorky takto zrekonstruované (obrázek 33 a 34) od tohoto přístupu očekávám lepší odezvu než v případě vah.

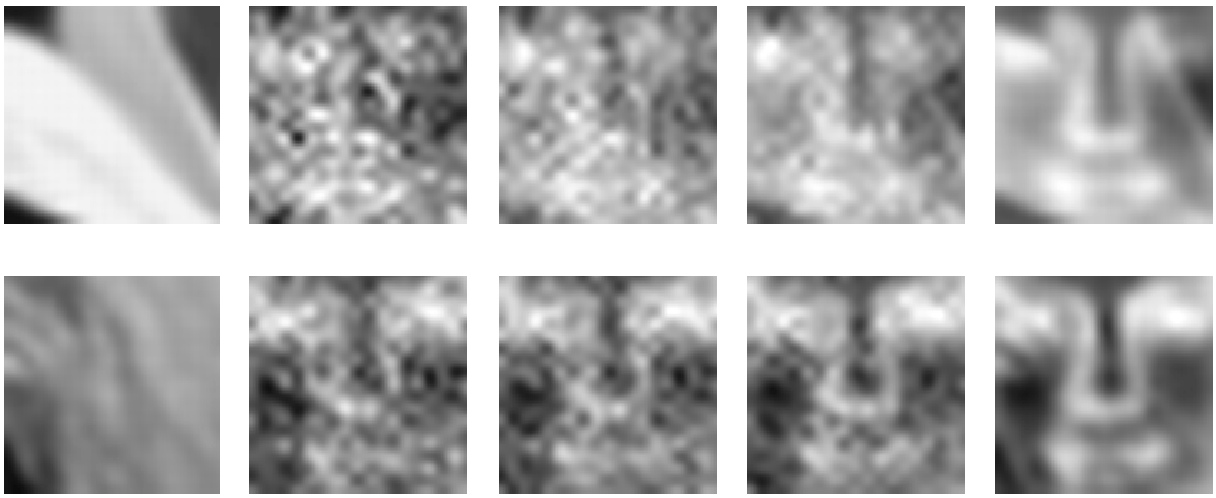
5.4.1 Redukce počtu použitých vlastních tváří

Pro potřeby neuronové sítě můžeme přistoupit k redukci použitých vlastních tváří popsané v teoretické části této práce. Tuto praktiku nebylo možno použít pro klasifikaci na základě vzdálenosti, protože důsledkem její aplikace je zvýšení šumu v obraze, což pro běžné výpočetní metody, jakou je bezesporu výpočet euklidovské vzdálenosti, přináší velmi obtížně řešitelný problém. Hlavním důvodem, proč došlo k rozšíření neuronových sítí jako výpočetního modelu, je právě zvýšená odolnost vůči přítomnosti šumu. Pokusím se zde tedy vyšetřit, zda právě zanesením šumu nedojde ke zlepšení klasifikačních schopností sítě. Modelem pro porovnání snížení počtu vlastních tváří bude výpočet naprosto stejný s použitím maximálního možného počtu vlastních tváří.

Pro lepší představu, jak se liší obraz rekonstruovaný pomocí menší množiny vlastních tváří, bylo nutno provést vizualizaci takto získaných vzorků. Tuto lze vidět na obrázcích 33 a 34 . Dle předpokladů použití nižšího počtu vlastních tváří vnáší do vzorků velké množství šumu. Lze vidět, že vyšší počet vlastních tváří přináší věrnější rekonstrukci obrazu. Naneštěstí si lze povšimnout též, že výsledek rekonstrukce neobličejových dat je velmi podobný rekonstruovaným obličejům. Pro potřeby vizualizace byly obrazy zvětšeny a interpolovány.



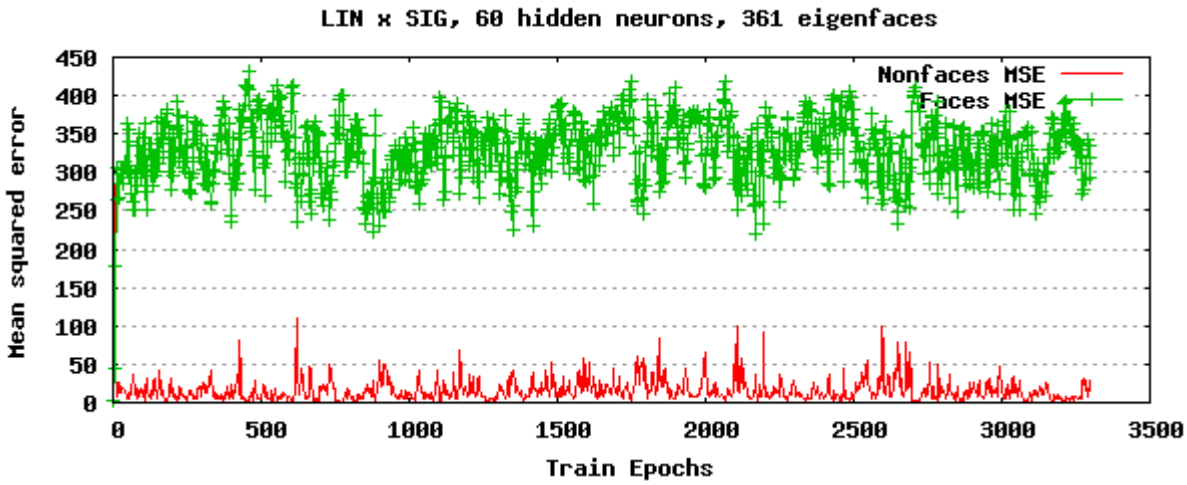
Obrázek 33: Ukázka rekonstrukce pomocí množiny vlastních tváří. Vpravo originální tvář, následují rekonstrukce pomocí 200, 250, 300, 361 vlastní tváře



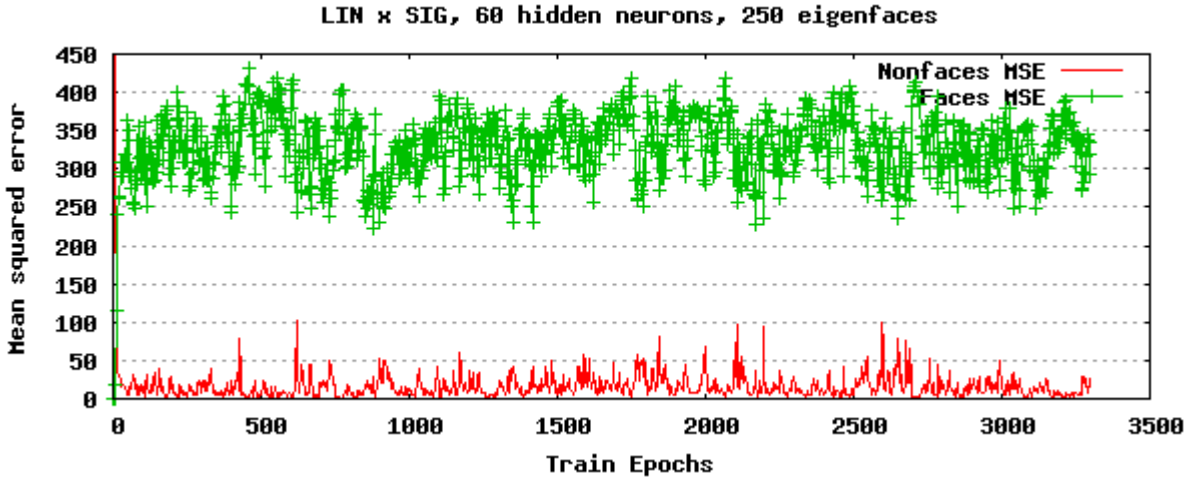
Obrázek 34: Ukázka rekonstrukce neobličejových dat pomocí množiny vlastních tváří. Vpravo originální obraz, následují rekonstrukce pomocí 200, 250, 300, 361 vlastní tváře

5.4.2 Trénovací model

Prvním testem, který jsem provedl, byl již tradičně odhad parametrů pomocí FANNToolu. Tento test však nedopadl vůbec dle očekávání. Neuronová síť nebyla absolutně schopná se učit, chyba dokonce vůbec nekmitala a měla konstantní průběh. Po důkladnější analýze dat, která byla přikládána na vstup sítě, jsem zjistil, že vstupní čísla jsou neobvykle malá. Tento jev je způsoben tím, že čísla, se kterými systém pracuje, jsou již tak malá čísla, a pro rekonstrukci tváře dle rovnice 24 je zapotřebí ještě jednoho násobení. čísla jsem tedy při převodu o několik řádů zvětšil a provedl další test pomocí FANNToolu, který však dopadl velmi podobně. S ne příliš optimálním průběhem testovaní jsem se setkal již dříve, proto jsem provedl testy další, totiž závislost chyby (MSE) na velikosti trénovací množiny.

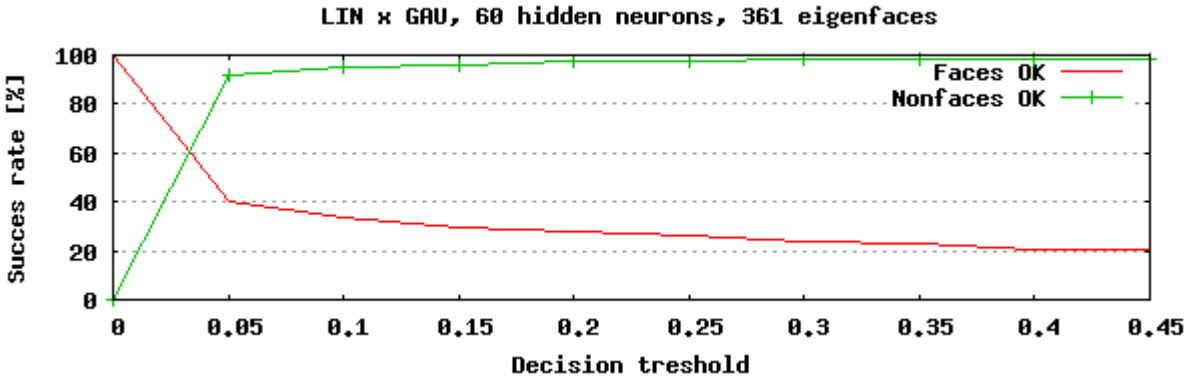


Obrázek 35: Vývoj chyby pro obrazy rekonstruované pomocí 361 vlastní tváře.

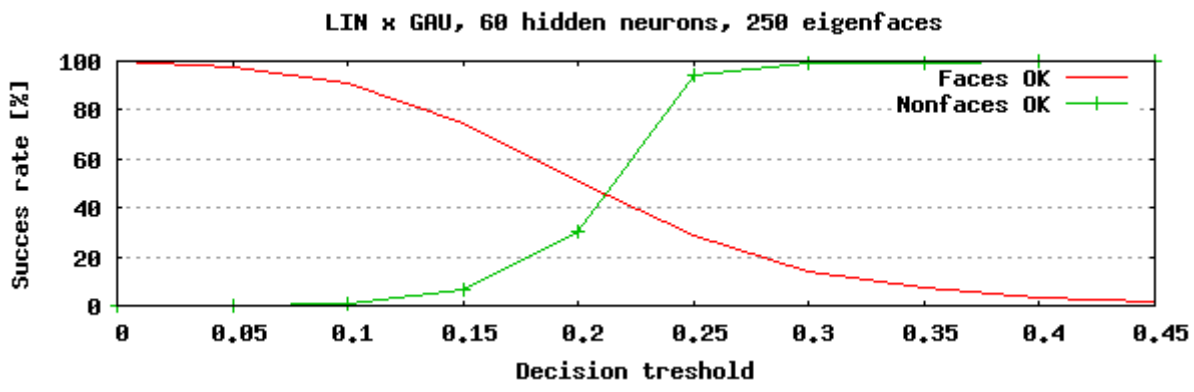


Obrázek 36: Vývoj chyby pro obrazy rekonstruované pomocí 250 vlastních tváří.

Na první pohled se zdají oba testy naprosto stejné. Velmi podobně vypadají i testy jiných aktivačních funkcí, i při jiném počtu skrytých neuronů. Síť nevykazuje schopnost učit se. Důvod tohoto chování lze opět hledat ve vstupních datech. Pohledem na obrázky 33 a 34 zjistíme, že i s použitím různého množství vlastních tváří se v různých fázích rekonstrukce obličeje i neobličejová data velmi podobají, dá se říci, že není možno ze zrekonstruované tváře rozpoznat, zdali zdrojový vzor obsahoval obličej. Přesto jsem provedl ještě několik testů závislosti úspěšnosti na hodnotě prahu pro klasifikaci.



Obrázek 37: Závislost úspěšnosti pro lineární funkci ve skryté a gausovu funkci výstupní vrstvě.



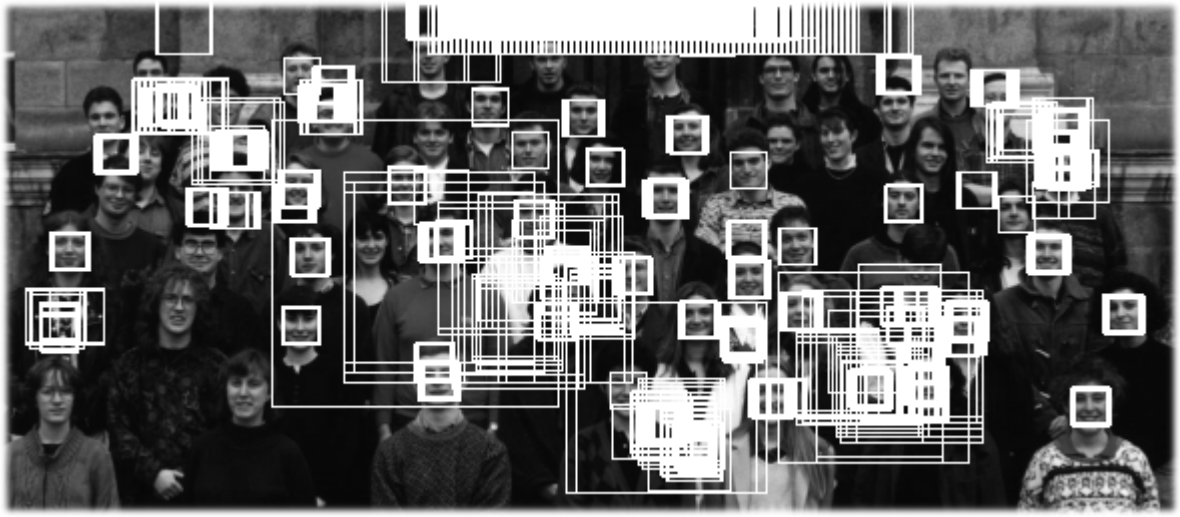
Obrázek 38: Závislost úspěšnosti pro lineární funkci ve skryté a gausovu funkci výstupní vrstvě pro 250 použitých vlastních tváří.

Pro 250 vlastních tváří lze vidět křivku velmi podobnou jako v případě použití pouze vah k vlastním tvářím. Pro 361 vlastních tváří se klasifikace mírně zlepšila, avšak křížení obou křivek na cca. 60 procentech nemůžeme akceptovat. Tato hodnota je sice zkreslena různou velikostí množin (opět 470 obličejových, 2000 neobličejových vzorků), avšak ani tak ji nemůžeme uznat za přijatelnou.

Implementace tohoto modelu je ve třídách NNTrainer, NNTester a NNExecutor. Detailnější popis lze nalézt v předchozí části, tedy klasifikaci pomocí vah. Metody určené pro práci s rekonstruovanými tvářemi mají ve většině případu koncovku reconstructed. Implementace je příliš složitá, aby byla rozebrána podrobněji, daleko podrobnější informace lze nalézt k programové dokumentaci těchto tříd obsažené v elektronické příloze této práce ve formátu HTML.

Provedené testy ukazují, že ani použití vlastních tváří pro rekonstrukci a následnou klasifikaci nepřináší žádané výsledky. Ukázalo se, že nejlepších výsledků lze dosáhnout s použitím obrazů nijak neupravených. Rekonstrukce pomocí eigenfaces lze označit za formu filtrace, která ovšem není v principu navržena pro zvýraznění mimotřídní variability, nýbrž pro účely komprese. Je velmi pravděpodobné, že závěr učiněný pro klasifikaci pomocí vah k eigenfaces, bude platit i pro tento případ. Aby bylo možno užitečnost tohoto přístupu jednoznačně potvrdit či vyvrátit, je nutno provést další testy na zcela jiné obličejové databázi, která bude obsahovat vzorky větších rozměrů.

Závěrem jsem se rozhodl umístit jeden obrázek pro porovnání úspěšnosti na skupinové fotografii s komponentou pracující s neupravenými vstupními daty. Podařilo se nalézt práh pro klasifikaci tak, aby výsledek klasifikace byl srozumitelný lidskému oku. Ačkoli se výsledek klasifikace zdá přijatelný, vzhledem k nestabilitě systému je zde skutečně pouze z ilustračních důvodů.



Obrázek 39: Klasifikace pomocí rekonstruovaných vlastních tváří. Pro klasifikaci byla použita síť s lineární aktivační funkcí ve skryté vrstvě a gausovou funkcí ve vrstvě výstupní. Práh pro klasifikaci byl nastaven na 0,46.

6 Eliminace falešných detekcí

Na testovaných skupinových fotografiích je možno pozorovat větší množství falešných a duplikátních detekcí. V teoretické části byla popsána jedna z myšlenek, tak je možno výsledek klasifikace zpřesnit pomocí dalšího zpracování vzorků klasifikovaných jako obličeje. Tato komponenta nebyla však implementována. Důvodem je velká časová náročnost provedení všech popsaných testů. Při pouhém pohledu na výsledky klasifikace je jasné, že tato komponenta musí být velmi robustní a schopná eliminovat velké procento falešných nálezů. Potom bude možno nastavit práh na mnohem nižší hodnotu, zvýšit pravděpodobnost kladné klasifikace všech obličejů a přítomnost většího množství falešných detekcí nebude tak kritická.

Eliminace falešných detekcí může být tématem samostatné diplomové práce, její implementace by měla být podřízena účelu, pro který je celý systém pro lokalizaci obličeje navržen. Zcela jistě bude navržena zcela odlišně pro videosekvence, kde je rychlost klasifikace více kritická a většina obličejů se tak jak tak přítomna na několika snímcích v několika různých polohách. Eliminace může být provedena pomocí další, jinak natrénované neuronové sítě, nebo pomocí zcela jiného přístupu, například jednoho z kapitoly o současných metodách detekce obličeje.

7 Závěr

V této práci jsem postupně představil některé zajímavé přístupy k lokalizaci obličeje a několik z nich jsem implementoval. Nejlepší výsledky jsem dosáhl pomocí klasifikace neuronovou sítí, kdy na jejím vstupu byla data nijak neupravená. Překvapením pro mne bylo snížení úspěšnosti při použití ekvalizace vstupních vzorků, která ve všech případech významně navýšila procento falešných detekcí. Velkým zklamáním pak byla úspěšnost klasifikace při použití vlastních tváří, a to ve všech popisovaných případech.

Lokalizace a obecně analýza obrazu pomocí neuronových sítí je problematikou velmi širokou a i podle nejnovějších zpráv zdaleka ne uzavřenou. Cílem této práce nebylo nalézt a implementovat systém vyznačující se mimořádnou úspěšností, to vzhledem k rozsahu diplomového projektu a šíři problematiky neuronových sítí není možné, ale doplnit stávající studie o některé další možné přístupy. Klasifikaci s použitím vlastních tváří jsem si vybral již při teoretické analýze v rámci semestrálního projektu, neboť se o ní nastudovaná literatura zmiňuje jen krátce. Zaměřil jsem se tedy na zmapování této problematiky a analýzu problému, kterým je nutno při implementaci čelit, a které nejsou při studiu teorie na první pohled zřejmé. Provedl jsem množství testů, které mají sloužit k základní orientaci v dané problematice, jakož i posloužit jako rámec pro testy více podrobné.

Tato diplomová práce je svým charakterem vhodná pro doplnění stávajících studií a má sloužit i přes ne příliš příznivé výsledky jako základ pro budoucí rozšíření. Jak již bylo zmíněno, je potřeba provést velmi podobné testy na zcela odlišné obličejové databázi. Zároveň je vhodné poohlédnout se po jiné knihovně pro neuronové sítě, aby v ideálním případě nabízela i odlišné trénovací modely. V rámci této diplomové práce bylo napsáno též značné množství zdrojového kódu, který může sloužit jako základ pro budoucí rozšíření. Jednou z dalších slabin neuronových sítí pro podobné problémy je jejich malá rychlost, která snižuje možnost širšího nasazení v reálných aplikacích. Inspirací řešení tohoto problému může být rapidně se zvyšující počet výpočetních jednotek ať již v nejnovějších mikroprocesorech, tak především v grafických kartách, které lze samozřejmě použít k libovolným účelům. Paralelizace jak klasifikace, tak některých učících algoritmů, jako je například kaskádové učení, není složitou záležitostí. Zároveň je nutno navrhnout komponentu pro eliminaci falešných detekcí, aby bylo možno pružněji nastavit rozhodovací práh a výrazně tak zvýšit efektivitu systému.

V krátkosti jsem nastínil některá možná rozšíření, které mohou vést k množství velmi zajímavých navazujících studií. Doufám, že má práce bude jejich oporou a inspirací.

8 Použitá literatura a odkazy

[1] Šnorek, M., Jiřina, M.: Neuronové sítě a neuropočítače, Skriptum ČVUT, vydavatelství ČVUT 1996.

[2] Rowley, H., Baluja, S., Kanade, S. : Neural Network-Based Face Detection, PAMI, 1998.

[3] Yang, M. H., Kriegman D., Ahuja, N.: Detecting Faces in Images – A Survey, IEEE Transaction on Pattern Analysis and Machine Intelligence, leden 2002

[4] Smith, L. I.: A tutorial on Principal Component Analysis, únor 2002, dokument dostupný na http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf

[5] Press, William H, Teukolsky Saul A., Vetterling William T., Flannery Brian P.: Numerical Recipes 3rd Edition: The Art of Scientific Computing, třetí vydání, 2007, dostupná na http://www.nrbook.com/nr3/.

[6] Nissen, Steffen: Large Scale Reinforcement Learning using Q-Sarsa and Cascading Neural Netwoks, říjen 2008.

[7] Případová studie PCA univerzity v Nevadě, Department of Computer Science & Engeneering, University of Nevada, Reno, dostupná na adrese http://www.cse.unr.edu/~bebis/MathMethods/PCA/case_study_pca1.pdf

[8] Knihovna pro neuronové sítě FANN (Fast Artifical Neural Networks Library v. 2.1.0 beta) dostupná ke stažení na http://leenissen.dk/fann/ pod GNU GPL licencí.

[9] Trénovací databáze CBCL (Center for Biological and Computional Learnig), dostupná na adrese http://cbcl.mit.edu/cbcl/software-datasets/FaceData2.html.

[10] Snímky pro testování systémů pro lokalizaci obličeje dostupné na stránkách The Vision and Autonomous System Center na adrese http://vasc.ri.cmu.edu/idb/images/face/frontal_images/images.tar

[11] Knihovna GSL (GNU Scientific Library) pro vědecké výpočty, dostupná pod GPL licencí na adrese http://www.gnu.org/software/gsl/.

[12] Konvence pro psaní zdrojových kódu společnosti Sun Microsystems přístupná na adrese http://java.sun.com/docs/codeconv/.

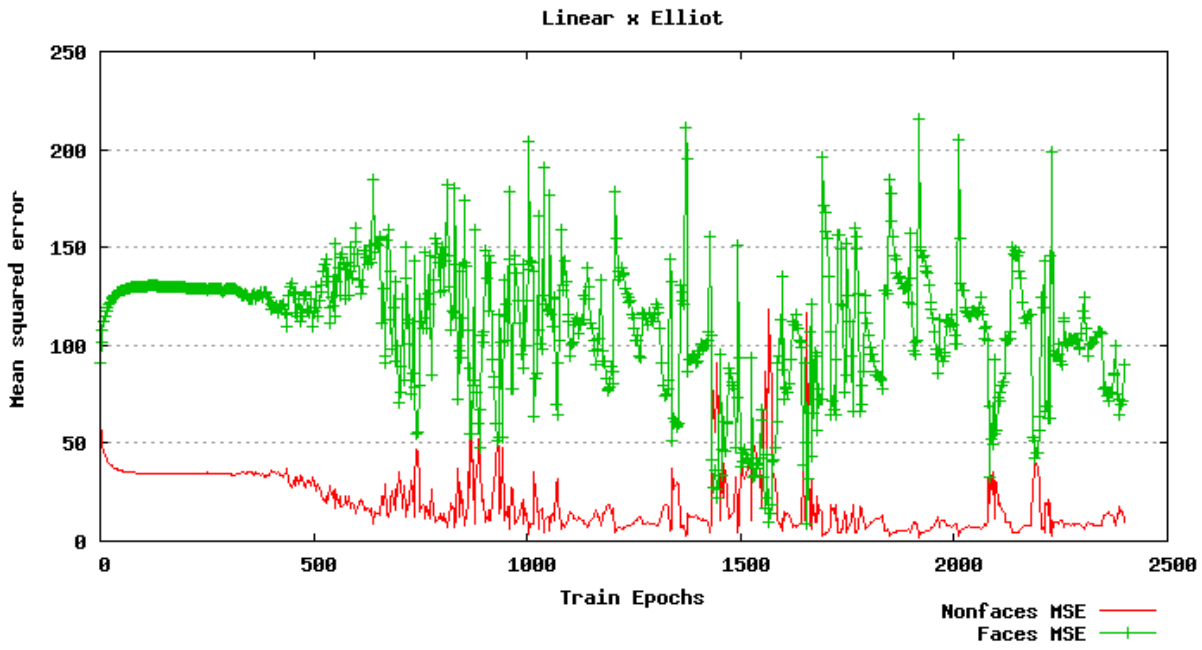
[13] Licence GNU General Public Licence verze 3 dostupná na adrese http://www.gnu.org/copyleft/gpl.html.

9 Přílohy

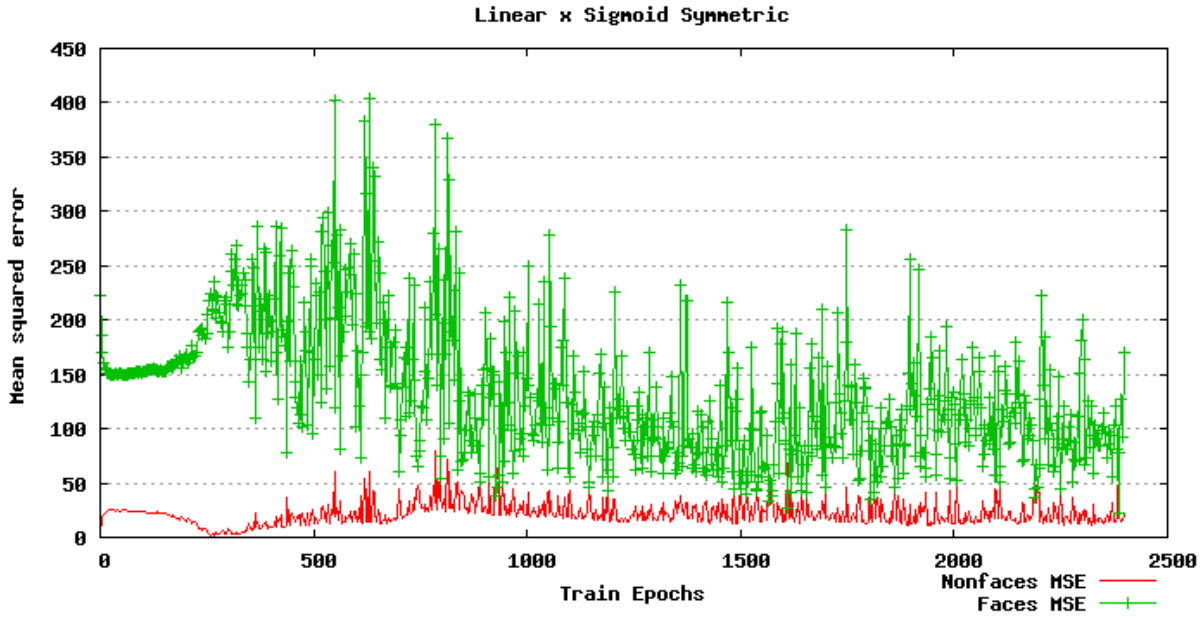
1. Grafy průběhu chyby vybraných kombinací aktivačních funkcí pro data neupravená pomocí vlastních tváří.
2. Grafy průběhu chyby vybraných kombinací aktivačních funkcí, kdy vstupem neuronové sítě byly váhy k vektoru vlastních tváří.
3. Popis instalace a práce s implementovaným systémem.
4. Elektronická příloha na CD. Obsah:
   - Zdrojové kódy systému popisovaného v této práci.
   - Soubory s vypočítanými vlastními tvářemi a jiné datové soubory, které umožňují okamžité použití implementovaného systému bez nutnosti pomocných výpočtů.
   - Skripty použité k inicializaci všech komponent systému.
   - Programová dokumentace zdrojového kódu generovaná programem Doxygen ve formátu HTML.
   - Uživatelská dokumentace – readme.txt s návodem k použití systému a detailnějším popisem elektronické přílohy.
   - Tento dokument ve formátu Microsoft Word a pdf.
   - Licenční ujednání platné pro tuto práci a poskytnuté Fakultou informačních technologií, VUT Brno.
5. Plakát demonstrující vlastnosti a činnost implementovaného systému

Příloha 1

Ukázky průběhu chyby při použití některých dalších aktivačních funkcí. Vstupní data byla čistá obličejová data, učící algoritmus Resilient Backpropagation, 60 neuronů ve skryté vrstvě. Vstupem sítě byla data neupravená pomocí vlastních tváří.



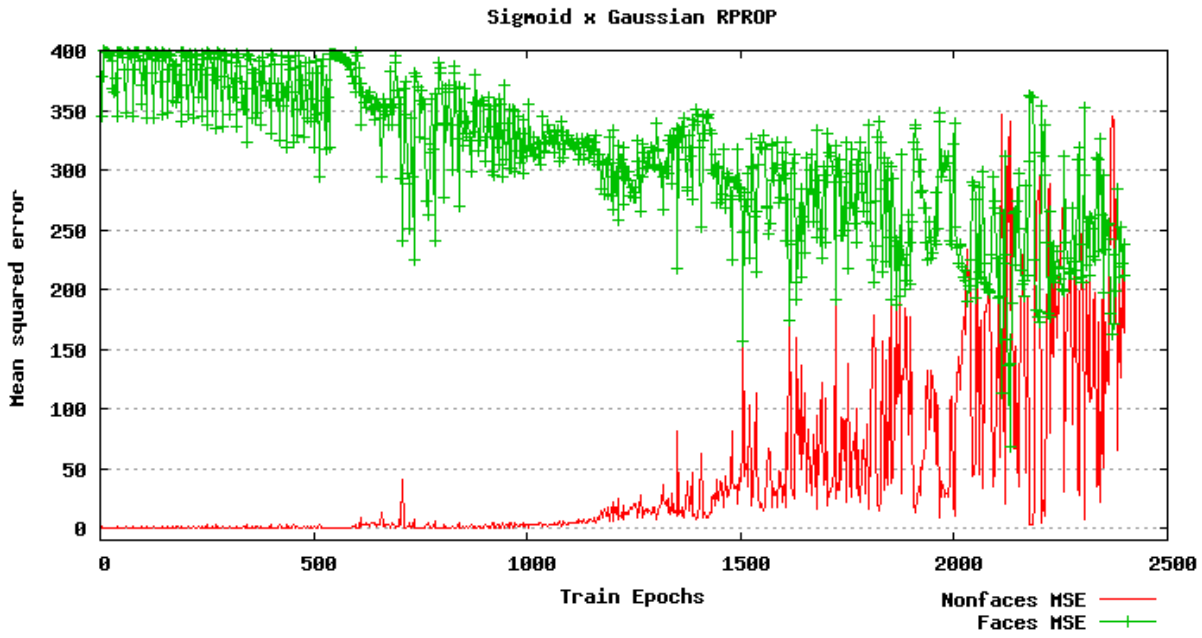
Obrázek 40: Vývoj chyby při použití lineární funkce ve skryté vrstvě a elliotovy funkce ve výstupní vrstvě.



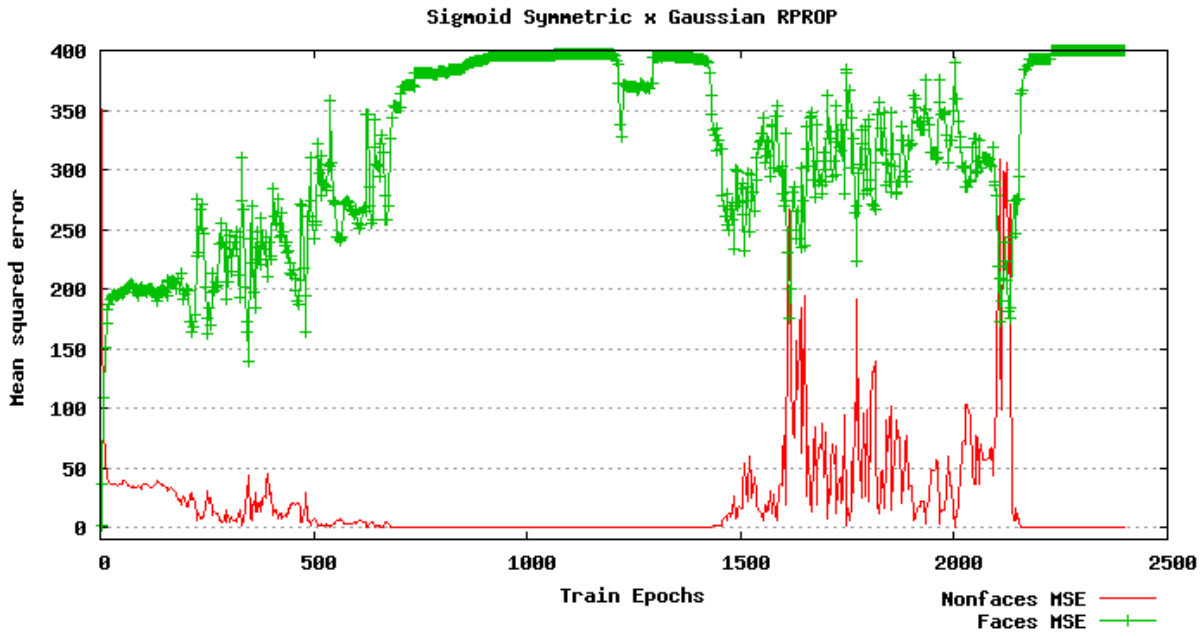
Obrázek 41: Vývoj chyb při použití lineární aktivační funkce ve skryté vrstvě a sigmoidální symetrické funkce ve vrstvě výstupní.

Příloha 2

Ukázky průběhu chyby při použití některých dalších aktivačních funkcí. Vstupními daty byly váhy k množině eigenfaces, učící algoritmus Resilient Backpropagation, 60 neuronů ve skryté vrstvě



Obrázek 42: Průběh chyby pro sigmoidální funkci ve skryté vrstvě a gausovu ve výstupní.



Obrázek 43: Průběh chyby pro sigmoidální symetrickou funkci ve skryté vrstvě a gausovu funkci ve výstupní

Příloha 3

Návod na instalaci a použití systému

1) Popis adresářové struktury přiloženého CD
   - Implementation/Localizator/src – zdrojové kódy
   - Implementation/ Localizator/debug – adresář pro binární soubory, obsahuje datové soubory s předpočítanými vlastními tvářemi a jiné pomocné soubory. Tento adresář je též cílem pro překlad, budou zde binární soubory, jakož i spustitelný soubor dip.exe.
   - Implementation/ Localizator/debug/scripts – skripty pro jednotlivé komponenty systému
   - Implementation/ Localizator/include – přídavné soubory pro překladač, hlavičkové soubory použitých knihoven
   - Implementation/ Localizator/libs – přeložené knihovny FANN a GSL
   - Implementation/ Localizator/debug/dataset – trénovací podmnožina databáze CBCL
   - Implementation/ Localizator/debug/testset – testovací podmnožina databáze CBCL
   - Implementation/ Localizator/ bigImages – velké snímky databáze The Vision and Autonomous System Center
   - Doc – Programová dokumentace ve formátu HTML, výchozí stránkou je index.html.
   - soubor xhendr00.pdf, xhendr00.doc – technická zpráva
   - soubor xhendr00.zip obsahující všechny výše uvedené body. Obličejové databáze obsahují velké množství malých souborů, kopírování po jednom může být velmi pomalé. Doporučuji proto zkopírovat tento archiv na pevný disk a teprve na disku provést extrakci.

2) Překlad systému

Adresář Implementation/ Localizator obsahuje projektové soubory pro Microsoft Visual Studio 2005. Lze je tedy otevřít a překlad provést standardním způsobem. Spustitelným souborem je pak Implementation/ Localizator/debug/dip.exe. Na počítači je nutno mít nainstalováno Platform SDK.

3) Spuštění systému

dip.exe <typ> <zdrojový obraz> <cílový obraz> [<práh pro klasifikaci>]

typ:

-w – pro klasifikaci jsou použity váhy k vlastním tvářím a neuronová síť

-r – pro klasifikaci jsou použity rekonstruované vlastní tváře a neuronová síť

-p – pro klasifikaci jsou použita neupravená data a neuronová síť.

zdrojový obraz:

- snímek ve stupních šedi ve formátu pgm, binární podoba tohoto formátu
- zdrojové snímky jsou k dispozici v Implementation/Localizator/bigImages

cílový obraz:

- jméno výstupního souboru kam je uložen výsledek klasifikace

práh pro klasifikaci:

- tento parametr nejvíce ovlivňuje klasifikaci, proto může být nastaven mimo přepsání příslušných zdrojových kódů. Pro vhodné hodnoty je nutno nahlédnout do technické zprávy pro jednotlivé typy klasifikace.
- parametr není povinný, pokud není specifikován je použit základní. Jeho hodnota je vypsána do konzole.

Pro více pokročilé použití je nutno změnit implementaci systému.