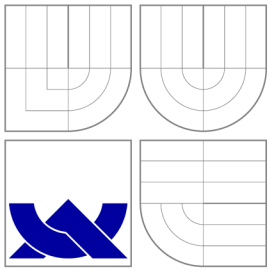
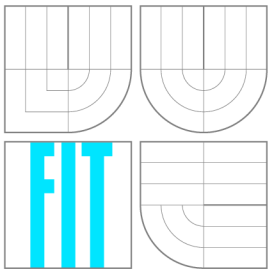


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



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

KONVOLUČNÍ NEURONOVÉ SÍTĚ PRO BEZPEČNOSTNÍ APLIKACE

CONVOLUTIONAL NEURAL NETWORKS FOR SECURITY APPLICATIONS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MARTIN KIŠŠ

VEDOUCÍ PRÁCE
SUPERVISOR

Doc. RNDr. PAVEL SMRŽ, Ph.D.

BRNO 2016

Abstrakt

Tato práce se zabývá návrhem a implementací aplikace pro rozpoznávání osob z bezpečnostní kamery. K rozpoznávání samostatného obličeje jsou zde použity konvoluční neuronové sítě, které vytváří reprezentaci daného obličeje, a algoritmus k-nejbližších sousedů, který slouží ke klasifikaci. K následnému rozpoznávání sekvence obličejů jsou zde implementovány tři algoritmy. Na testovacích datech dosahovala úspěšnost rozpoznávání až 75 %.

Abstract

This thesis deals with design and implementation of application for person recognition in security camera. For single face rocongition are used convolutional neural networks, which creates representation of the face, and k-nearest neighbours algorithm for classification. For recognition of sequence of faces there are three algorithms implemented. On test data success of recognition reached nearly 75 %.

Klíčová slova

Rozpoznávání tváří, bezpečnostní kamery, zpracování obrazu, neuron, konvoluční neuronové sítě, k-nejbližších sousedů.

Keywords

Face recognition, security cameras, image processing, neuron, convolutional neural networks, k-nearest neighbours.

Citace

KIŠŠ, Martin. *Konvoluční neuronové sítě pro bezpečnostní aplikace.* Brno, 2016. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Smrž Pavel.

Konvoluční neuronové sítě pro bezpečnostní aplikace

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Doc. RNDr. Pavla Smrže, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Martin Kišš
17. května 2016

Poděkování

Chtěl bych poděkovat panu doc. RNDr. Pavlu Smržovi, Ph.D. za odborné vedení a poskytnutí prostředků pro tvorbu práce.

Obsah

Kapitola 1

Úvod

Elektronická zařízení postupně pronikají do většiny oblastí lidského života. Ne jinak je tomu taky v případě kamer, které se uplatňují také v oblasti bezpečnosti a zabezpečení. Bezpečnostní kamery zaznamenávají pohyb v rámci interiéru nebo exteriéru daného místa. Aby bylo možné kamery aktivně využívat pro účely zabezpečení určitého místa, je zapotřebí nějaké osoby, popřípadě nějakého nástroje, který bude data z kamery vyhodnocovat.

Cílem této práce je automatizovat proces vyhodnocování dat z kamery, konkrétně rozpoznávání obličejů osob, které před danou kamerou projdou, pomocí navržené a implementované aplikace. Jako zdroj kamerových záznamů byla využita kamera umístěná na jedné z chodeb fakulty. Jako nástroj pro rozpoznávání obličejů osob jsou v této práci použity konvoluční neuronové sítě, jejichž princip bude vysvětlen v kapitole 2. V kapitole 3 bude popsán návrh výsledné aplikace a knihovny, které byly použity v rámci implementace. V této kapitole bude také přiblížena funkcionalita použitých a implementovaných algoritmů. Obsahem kapitoly 4 budou informace tykající se natrénovaných konvolučních neuronových sítí a jejich výsledky na vytvořených testovacích případech. Závěrečná kapitola bude obsahovat celkové shrnutí práce, dosažené výsledky a budou zde navrhnuta možná vylepšení a rozšíření.

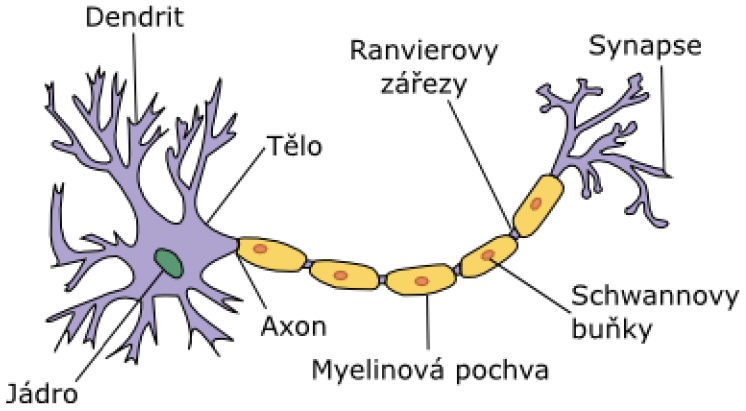
Kapitola 2

Konvoluční neuronové sítě

V této kapitole se nachází informace o konvolučních neuronových sítích. V první části je vysvětleno, jak se umělý neuron inspiruje biologickým neuronem. Dále je zde vysvětlen princip dopředných neuronových sítí a rozšíření těchto sítí o konvoluční a pooling vrstvy. V poslední části je popsán princip učení neuronových sítí.

2.1 Perceptron

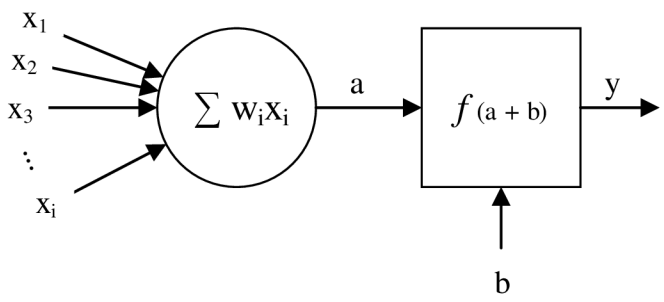
Základními buňkami mozku jsou *neurony*. Jedná se o specializované buňky určené k příjmu signálů, jejich zpracování a následné odpovědi. Neuron se skládá z několika *krátkých výběžků* (*dendritů*), *těla* (*soma*) a jednoho *dlouhého výběžku* (*axonu*). Zatímco dendrity slouží pro příjem signálu z okolí, axon naopak signál vysílá. Mezi jednotlivými neurony se nacházejí *spojení* (*synapse*), které umožňují předávání signálů. Při předání signálu mezi neurony může v synapsi dojít ke zvětšení či zmenšení předávaného signálu. Tělo neuronu obsahuje *jádro* nervové buňky, které určuje, jaká intenzita signálu bude vyslána k ostatním neuronům na základně intenzit signálů přijatých [14].



Obrázek 2.1: Schéma biologického neuronu. Převzato z [14]

Umělý model, který se inspiruje biologickým neuronem, se nazývá *perceptron*. Perceptron, stejně jako neuron, zpracovává vstupní signály (hodnoty získané na vstupu), které zpracuje a zvolí vhodný výstupní signál (hodnota přivedená na výstup). Vektor vstupních hodnot je vynásoben vektorem vah (obdoba synapse) a všechny hodnoty jsou sečteny. K této

hodnotě se také přičítá takzvaná *prahová hodnota* (někdy také *bias*), tedy hodnota, která nezávisí na vstupních hodnotách, ale je trvale uložena v perceptronu. Výsledná hodnota je vstupem takzvané *aktivační funkce*, která určí výstupní hodnotu celého perceptronu [10].



Obrázek 2.2: Schéma perceptronu

Mezi nejběžnější aktivační funkce patří *skoková aktivační funkce* ($y = sgn(x)$), která vrací pouze dvě hodnoty (0 nebo 1, případně -1 nebo 1). Při vstupu menším než určitá mez je na výstup vybrána nižší hodnota, při vstupu větším než mez je vybrána vyšší hodnota. Další aktivační funkcí je *sigmoidní aktivační funkce* [4], která vrací hodnoty z intervalu 0 až 1. Rovnice sigmoidní aktivační funkce je:

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

Jako aktivační funkce se používá také funkce *hyperbolický tangens* [4], jež vrací hodnoty z intervalu -1 až 1. Rovnice hyperbolické tangenty je:

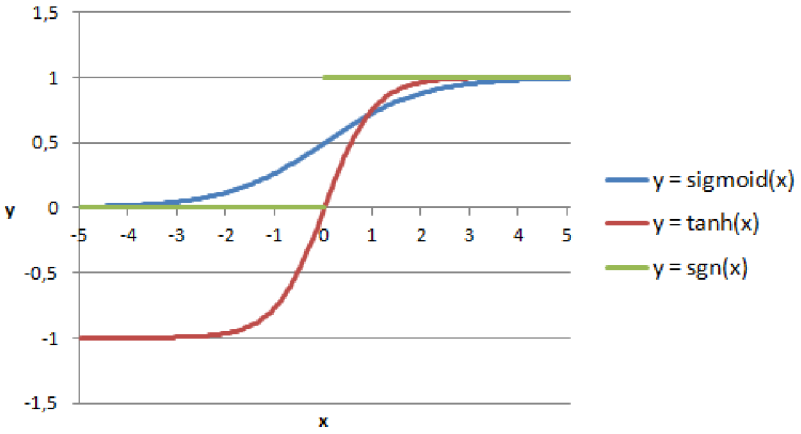
$$tanh\, x = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

Průběhy těchto aktivačních funkcí se nachází na obrázku 2.3.
Umělý neuron je také možné popsat matematicky vztahem:

$$y = f\left(\sum_{i=1}^{n} w_i x_i + b\right)$$

kde:

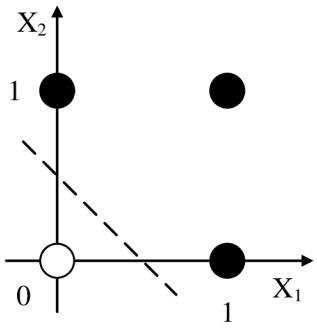
- y je výstupní hodnota neuronu,
- n je velikost vstupního vektoru neuronu,
- x_i je vstupní vektor neuronu,
- w_i je vektor vah neuronu,
- b je prahová hodnota neuronu,
- f je aktivační funkce neuronu.



Obrázek 2.3: Grafy aktivačních funkcí

2.2 Neuronová síť

Jedním neuronem je možné modelovat například binární operace AND nebo OR. Výstupem těchto funkcí jsou totiž *lineárně separovatelné* hodnoty [6]. Tedy hodnoty, které lze klasifikovat pomocí jedné rozhodovací linie (viz obrázek 2.4). Vstupem neuronu jsou dvě hodnoty x_1 a x_2 a výstupem je hodnota 0 nebo 1, podle příslušné funkce.

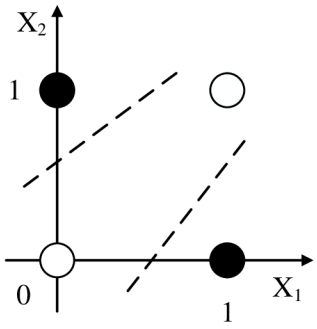


X_1	X_2	Y
0	0	0
0	1	1
1	0	1
1	1	1

Obrázek 2.4: Funkce OR, ukázka lineárně separovatelných dat

Naopak některá data mohou být *lineárně neseparovatelná*. Jedná se například o binární operaci XOR (viz obrázek 2.5) [6]. Aby bylo možné namodelovat tuto operaci, je nutné použít více neuronů. Pokud je použito více *paralelních* neuronů, jedná se takzvaně o *single–layer perceptrons*. Tedy o neurony v jedné *vrstvě*. Použití více neuronů v jedné vrstvě ovšem ke klasifikaci lineárně neseparovatelných dat nestačí. Pokud však je zařazeno více neuronů, či neuronových vrstev, do série tak, že výstup jednoho neuronu (jedné vrstvy) je vstupem dalšího neuronu (další vrstvy), jde o *multi–layer perceptrons*. Vrstvy jsou zde propojeny tak, že výstup každého neuronu aktuální vrstvy je přiveden na vstup každého neuronu následující vrstvy. Jde o typ *plně propojené vrstvy*. S použitím více vrstev je již možné

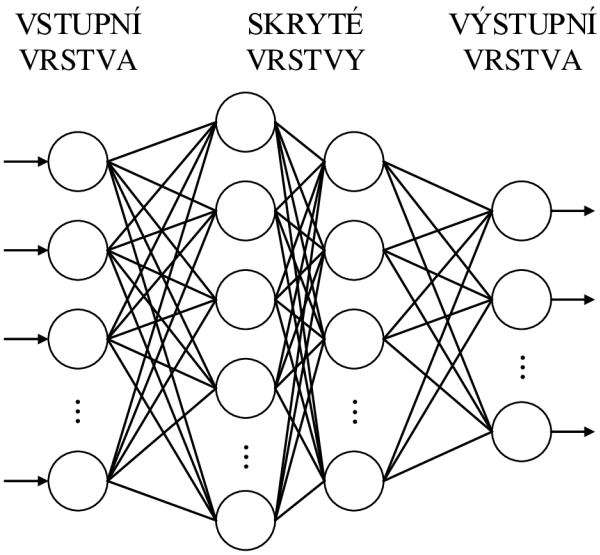
klasifikovat i lineárně neseparovatelné hodnoty.



X_1	X_2	Y
0	0	0
0	1	1
1	0	1
1	1	0

Obrázek 2.5: Funkce XOR, ukázka lineárně neseparovatelných dat

Typická neuronová síť se skládá ze *vstupní vrstvy*, jedné nebo více mezivrstev (skrytých vrstev) a *výstupní vrstvy* (viz obrázek 2.6). Jedná se o takzvané *dopředné* (anglicky *feed–forward*) neuronové sítě. Data, která jsou přivedená na vstup sítě, jdou celou sítí pouze jedním směrem. Druhým typem neuronových sítí jsou *rekurentní* (anglicky *recurrent*) neuronové sítě. V těchto sítí existují smyčky v propojení vrstev [8]. Výhodou rekurentních neuronových sítí je, že mohou obsahovat paměť a výstup tedy nezávisí pouze na vstupních datech, ale také na předchozích datech. Toho se využívá například při zpracovávání nebo generování textu [7].

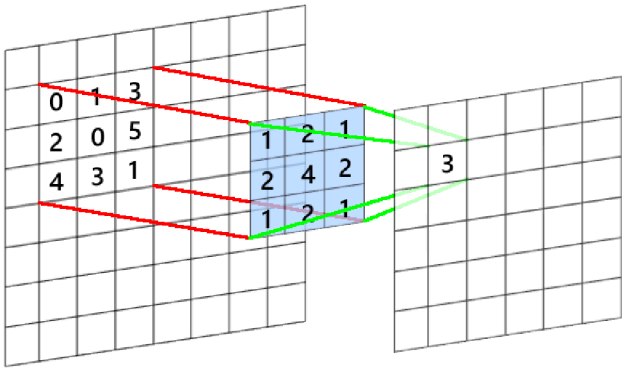


Obrázek 2.6: Schéma dopředné neuronové sítě

2.3 Konvoluční neuronové sítě

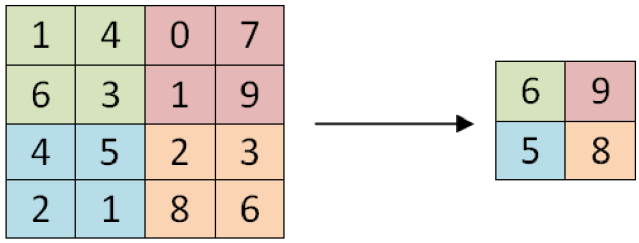
Konvoluční neuronové sítě jsou dopředné neuronové sítě, které využívají *konvoluční vrstvy* a *pooling vrstvy*. Tyto vrstvy se využívají hned po vstupní vrstvě, pro předzpracování vstupu.

Konvoluce je jedna z metod pro výpočet odezvy systému na vstupní signál. Při zpracování obrazu, což je diskrétní 2D signál, se konvoluce využívají pro různé transformace původního obrazu, jako například rozmazání nebo zaostření obrazu, detekce hran, a podobně. Konvoluci obrazu je možné si představit jako posouvání *masky* (*konvolučního jádra*) bod po bodu po obraze, přičemž v každé vzájemné pozici masky a obrazu je vypočtena výsledná hodnota nového obrazu [9]. Výsledná hodnota je součtem původních bodů vynásobených příslušnými koeficienty masky a vydělena velikostí masky (viz obrázek 2.7).



Obrázek 2.7: Schéma konvoluce obrazu

Pooling vrstva (v české literatuře označována také jako vrstva spojení) se využívá pro *podvzorkování* (anglicky *subsampling*) původního vstupu [9]. Tato vrstva funguje podobně jako konvoluční vrstva, kdy přes celý obraz postupuje maska, která z dané matice bodů vybere vhodnou výstupní hodnotu. Rozdíl oproti konvoluční vrstvě však je ten, že maska se posouvá vždy tak, aby nepřekrývala předchozí pozici. Výstupní hodnota se určuje na základě požadované funkce, vybírá se buďto maximum z daných hodnot (*max-pooling*) nebo střední hodnota (*mean-pooling*).



Obrázek 2.8: Schéma max-poolingu

2.4 Klasifikace a regrese

Neuronové sítě obecně mohou řešit mnoho typů problémů. Mezi ty nejběžnější patří *klasifikace* a *regrese*. Při klasifikaci je od neuronové sítě požadováno určení jedné z N klasifikova-

ných *tříd*. V této práci je tento typ sítě použit při klasifikaci obličeje, výstupem sítě jedna z N naučených osob. Při využití neuronové sítě řešící regresi je naopak výstupem obecně N *nezávislých hodnot*. V této práci je regresní typ neuronové sítě použit pro detekci očí v obličeji. Výstupem sítě jsou x–ová a y–ová souřadnice levého a pravého oka.

2.5 Trénování neuronové sítě

Strojové učení (anglicky *machine learning*) je jedna z oblastí umělé inteligence. Výhodou strojového učení je, že není třeba specifikovat konkrétní pravidla, podle kterých se má systém rozhodovat, ale systém je schopen se je sám naučit. Jednou z oblastí strojového učení jsou také neuronové sítě, respektive konvoluční neuronové sítě.

2.5.1 Popis učení

Ještě než se začne síť učit, je nutné nastavit váhy a prahové hodnoty každého neuronu a u konvolučních vrstev je potřeba nastavit všechny konvoluční jádra. Všechny tyto hodnoty se inicializují na *náhodné hodnoty*. Obvykle jsou tyto hodnoty v rozmezí -1 a 1.

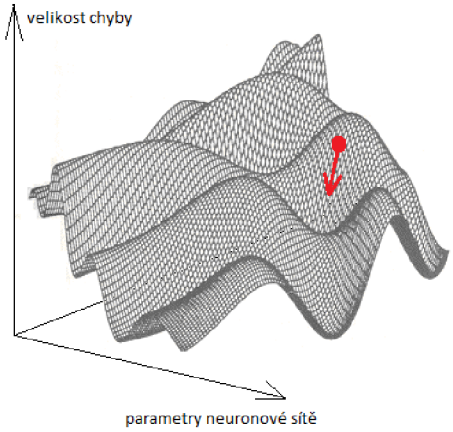
Aby se neuronová síť cokoliv naučila, je třeba mít takzvanou *trénovací sadu*. To je sada *vstupních dat* a *požadovaných výsledků*, která je předkládána neuronové síti, na kterých se provádí učení. Z tohoto pohledu se jedná o *učení s učitelem* (anglicky *supervised learning*).

Další nezbytnou součástí učení, je stanovená funkce, která určí míru naučení. Souhrnně se takovéto funkce nazývají *objektivní funkce* (anglicky *objective function*) [10]. U neuronových sítí se používá například *rozptyl* neboli *střední kvadratická odchylka* (anglicky *Mean Square Deviation* nebo *Mean Square Error*). Na základě vypočtené chyby mezi očekávanými a získanými hodnotami neuronovou sítí se určí, jak je třeba síť upravit tak, aby se výstup co nejvíce blížil požadovaným hodnotám.

Celé trénování se pak odehrává v iteracích, kterým se někdy také říká *epochy*. Během jedné iterace je předložena neuronové síti celá trénovací sada a na základě zjištěných chyb se upraví hodnoty neuronové sítě. Úpravami hodnot neuronové sítě jsou myšleny úpravy hodnot jednotlivých neuronů v síti. Upravují se jednotlivé váhy vstupních hodnot i prahová hodnota neuronu, u konvolučních vrstev se upravují konvoluční jádra.

Jedním z algoritmů, jak určit co nejlepší změny vah a prahových hodnot jednotlivých neuronů dopředné neuronové sítě, je takzvané *zpětné šíření chyby* (anglicky *backpropagation*) [2, 10]. Jedná se o algoritmus, kdy se nejprve zjistí chyba a následně se zpětně vypočítává o jakou hodnotu se mají změnit jednotlivé váhy a prahové hodnoty každého neuronu. Nové hodnoty vah a prahu se vypočítávají například pomocí metody *gradient descent*, kdy se vypočítají derivace pro každou dílčí část neuronu a následně se nastaví hodnoty tak, aby nová chyba byla co nejmenší. Metodu gradient descent si lze graficky představit jako graf (viz obrázek 2.9). V tomto grafu znázorňuje svislá osa velikost chyby a zbylé osy nastavení sítě. Aktuální nastavení sítě je zde znázorněno jako *bod* a předpokládané nové nastavení sítě je ve směru nejrychlejšího klesání (v obrázku 2.9 znázorněno *šipkou*). Cílem je postupovat, dokud není nalezeno *minimum chybové funkce*.

Při učení neuronové sítě je také důležitým parametrem *rychlost učení* (anglicky *learning rate*) [10]. Tato hodnota určuje koeficient, kterým se vynásobí původní vypočtená změna každé hodnoty neuronu. V grafu 2.9 je možné si tento parametr představit jako velikost, o kterou se posune z aktuálního nastavení do nového, v daném směru. Pokud bude hodnota rychlosti učení příliš velká, pak může dojít k příliš velkým změnám v síti a z toho důvodu se síť nebude efektivně učit. Naopak při velmi malé hodnotě bude učení značně pomalé.

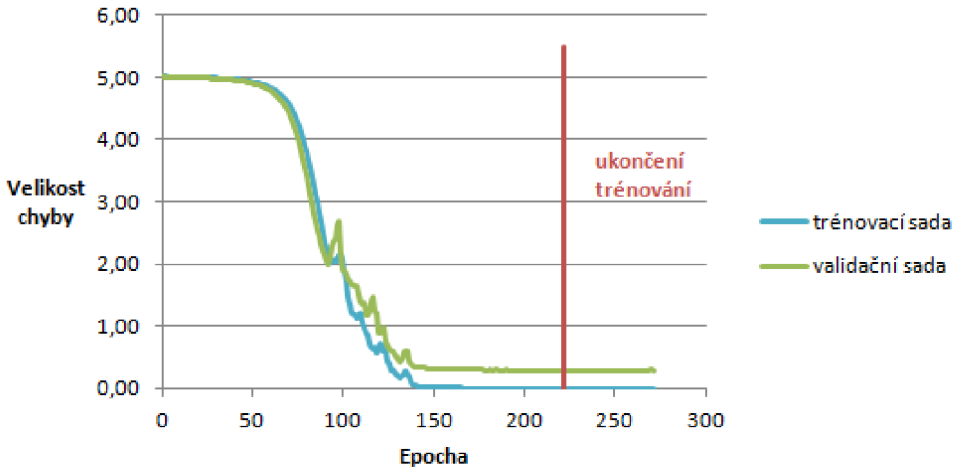


Obrázek 2.9: Ukázka gradient descent. Převzato z [2]

2.5.2 Křížová validace a přetrénování

Při velmi dlouhém učení se síť dostane do stavu takzvaného *přetrénování* (anglicky *overfitting*). To je stav, kdy se síť perfektně naučí testovací sadu, avšak při použití na datech, ze kterých se neučila, ale měla by je být schopna správně určit, selhává. Aby se co nejvíce eliminovalo přetrénování používají se různé techniky. Jednou z nejběžnějších je takzvaná *křížová validace* (anglicky *cross–validation*) [10]. Jde to techniku, kdy je trénovací sada rozdělena na podmnožinu, na které se systém trénuje, a podmnožinu, na které se vyhodnocuje aktuální naučenost. Tato oddělená data se nazývají validační sada.

Během učení je pak důležité správně určit moment, kdy je vhodné učení zastavit. Jednou z možností je po každé iteraci učení zjistit aktuální chybu na validační sadě, a pokud byla lepší než v předchozí iteraci, pak si uložit současné nastavení sítě. Pokud již však síť určitý počet iterací nedosahuje lepších výsledků, pak je lepší učení zastavit a za nejlepší nastavení sítě prohlásit poslední uložené parametry.



Obrázek 2.10: Ukázka průběhu trénování sítě.

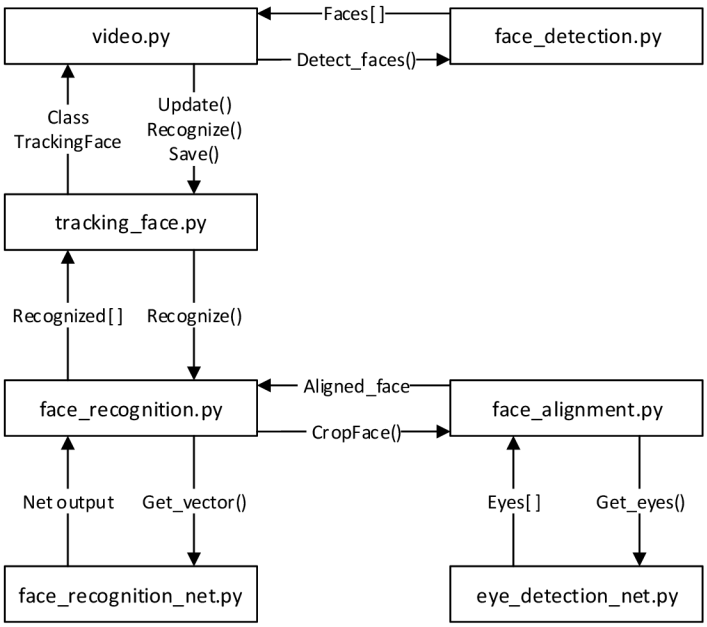
Kapitola 3

Návrh a implementace

V této kapitole se nachází popis výsledné aplikace. První část se zabývá jednotlivými naprogramovanými moduly, v další části je popsáno využití externích knihoven a nakonec jsou popsány vybrané použité algoritmy a techniky.

3.1 Struktura aplikace

Jak již bylo zmíněno, celá aplikace se skládá z několika modulů. Jejich vzájemné propojení je zobrazeno na obrázku 3.1.



Obrázek 3.1: Schéma výsledné aplikace.

3.1.1 Modul tracking_face

Modul *tracking_face.py* definuje třídu *TrackingFace*. Úkolem této třídy je uchovávat informace o aktuálně sledovaném obličeji ve videu. Vedle seznamu, který uchovává jednotlivé detekované obličeje daného člověka během jednoho průchodu před kamerou, obsahuje také takzvaný *Kalmanův filtr*. Tento filtr slouží k predikci pozice daného obličeje na dalším snímku videa. Fungování Kalmanova filtru bude více přiblíženo v části 3.3.1.

Důležitou součástí této třídy je také metoda na rozpoznání obličeje. To se provede zavoláním funkce modulu *face_recognition.py*, které je předána sekvence obličejů k rozpoznání.

3.1.2 Modul video

Hlavním modulem výsledné aplikace je modul *video.py*. Tento modul zpracovává zadaný video soubor, který postupně prochází snímek po snímku a za pomocí třídy TrackingFace zpracovává obličeje osob procházejících před kamerou. Modul si udržuje seznam sledovaných obličejů z předchozích snímků videa. Pokud je však daný sledovaný obličej již delší dobu neaktualizován (obličej nebyl detekován v několika snímcích za sebou), pak dojde k rozpoznání daného obličeje zavoláním příslušné metody třídy TrackingFace. Po dokončení rozpoznávání jsou jednotlivé obličeje uloženy do příslušného adresáře a následně je tato instance třídy TrackingFace odebrána ze seznamu sledovaných obličejů.

K tomu aby bylo možné nějaké sledované obličeje aktualizovat je nejdříve nutné je detekovat. To se děje zavoláním funkce modulu *face_detection.py*, která vrací seznam detekovaných obličejů v daném snímku. Při aktualizaci sledovaných obličejů dochází k párování sledovaných obličejů s nově detekovanými obličeji, a pokud byly všechny stávající obličeje aktualizovány (případně nebyl detekován nový obličej odpovídající stávajícímu obličeji), pak jsou pro zbývající detekované obličeje vytvořeny nové instance třídy TrackingFace, které jsou připojeny k seznamu stávajících sledovaných obličejů.

3.1.3 Modul face_detection

Úkolem modulu *face_detection.py* je volání funkce knihovny *OpenCV*, která vrací seznam detekovaných obličejů v daném obraze. Více bude algoritmus detekce obličejů přiblížen v části 3.3.2. Před samotnou detekcí obličejů je implementována možnost úpravy histogramu snímku, čímž dochází k přesnější detekci obličejů.

3.1.4 Modul face_recognition

V modulu *face_recognition.py* je implementována logika pro rozpoznání dříve detekovaného obličeje. K samotnému rozpoznání sekvence obličejů bylo naprogramováno několik algoritmů, které budou popsány v části 3.3.4. Výstupem funkce, která zajišťuje rozpoznání, je procentuální příslušnost sekvence k dané rozpoznané osobě. Pro rozpoznávání se využívají funkce související s konvolučními neuronovými sítě. Jejich implementace se nachází v modulu *face_recognition_net.py*. Aby bylo rozpoznávání co nejpřesnější, je vhodné detekované obličeje určitým způsobem zarovnat. K zarovnání obličeje dochází voláním funkce modulu *face_alignment.py*.

3.1.5 Modul face_recognition_net

Modul *face_recognition_net.py* obsahuje funkce, které umožňují práci s konvolučními neuronovými sítěmi. Jedná se o funkce pro získání výstupu zadané vrstvy na daný vstup,

vytvoření nové sítě a její natrénování, či uložení a opětovné načtení ze souboru. Dále jsou zde implementovány třídy, které jsou využity při trénování sítě. Jde o třídy, které jsou předány při inicializaci a mají za úkol upravovat parametry učení nebo upravovat samotnou datovou sadu, na které je síť trénována. Tyto třídy budou detailněji popsány v části 3.3.5.

3.1.6 Modul face_alignment

Modul *face_alignment.py* slouží k volání funkce na zarovnání obličeje. To se provádí využitím základních matematických transformací obrazu, jakými jsou *škálování*, *otočení* a *posunutí*. Tento algoritmus byl převzat z [13] a je licencován jako BSD licence. Princip tohoto algoritmu bude nastíněn v části 3.3.3

3.1.7 Modul eye_detection_net

Modul *eye_detection_net.py* obsahuje, podobně jako modul *face_recognition_net.py* (popsán v kapitole 3.1.5), funkce, které slouží k manipulaci s konvoluční neuronovou sítí, jež slouží pro detekci očí v obličeji. Stejně, jako u konvoluční neuronové sítě pro rozpoznávání tváří, je i zde použito několik tříd, které se použijí při inicializaci nové konvoluční neuronové sítě, jež upravují trénování sítě.

3.2 Použité knihovny

3.2.1 OpenCV

OpenCV je knihovna s otevřeným zdrojovým kódem, ve které jsou implementovány optimalizované algoritmy pro *počítačové vidění* (anglicky *computer vision*) a strojové učení. Knihovna je licencována s BSD licencí.

Z této knihovny je využit algoritmus pro detekci obličeje a algoritmy pro načítání, úpravu a ukládání video souborů a obrázků. Dále je v aplikaci použita třída, ve které je naprogramován algoritmus *k-nejbližších sousedů*, jež je využita při rozpoznávání (viz 3.3.4), a také třída s implementací Kalmanova filtru (viz 3.3.1).

3.2.2 Theano

Knihovna *Theano* slouží k definici, optimalizaci a evaluaci matematických výrazů, především pak pro práci s vícerozměrnými poly. Kód je následně možné spouštět také na grafické kartě, čímž dochází ke zvýšení výkonu. Je také vhodná pro zpracování velkého množství dat. Knihovna je licencovaná s BSD licencí.

Tato knihovna není v projektu přímo využitá, avšak další použité knihovny využívají právě knihovnu Theano.

3.2.3 Lasagne

Lasagne je knihovna, která slouží k vytváření a trénování neuronových sítí. Umožňuje vytvářet dopředné konvoluční neuronové sítě i rekurentní neuronové sítě. Tato knihovna je využívá funkcí knihovny Theano, díky které je její kód optimalizován a může být také spouštěn na grafické kartě. Knihovna je licencovaná s MIT licencí.

3.2.4 Nolearn

Knihovna *nolearn* obsahuje třídy, které obalují třídy a funkce knihovny Lasagne, čímž umožňují snadnější práci s neuronovými sítěmi. Tato knihovna je licencovaná s MIT licencí. Ve výsledné aplikaci se pro práci s konvolučními neuronovými sítěmi využívají třídy právě této knihovny.

3.3 Vybrané algoritmy a techniky

3.3.1 Kalmanův filtr

Kalmanův filtr (nazvaný podle Rudolfa E. Kálmána [3] je filtr, který se využívá k odhadu budoucích hodnot na základě hodnot, které byly v minulosti získány. Kalmanův filtr se stal standardní součástí například v navigacích automobilů, letectví [1] apod. V automobilech je kombinován s klasickou GPS navigací, přičemž Kalmanův filtr se používá při situacích, kdy není signál GPS dostatečný (např. v tunelech). V takovýchto situacích dokáže Kalmanův filtr, na základě dříve získaných dat, odhadovat, kde se bude v následujících okamžicích daný automobil nacházet.

Kalmanův filtr pracuje ve dvou fázích. Nejprve odhaduje nové hodnoty na základě předchozích dat (1. fáze) a po následném měření, čímž se zjistí skutečná data, je filtr opraven na správnou hodnotu (2. fáze) [1].

V této práci je Kalmanův filtr uplatněn pro sledování obličeje jedné osoby v rámci jednoho průchodu před kamerou. Při první detekci obličeje je vytvořena nová instance třídy *KalmanFilter* knihovny OpenCV. Při zpracování každého následujícího snímku videa je pro každý sledovaný obličej predikována nová pozice ve snímku a na základě detekovaných obličejů je vybrán jako nový obličej ten, který je nejblíže predikované pozici a zároveň je vzájemná vzdálenost nižší než nastavený práh. Poté co je získána nová pozice obličeje ve snímku, je volána metoda pro korekci Kalmanova filtru. Pokud se stane, že v daném snímku není obličej z nějakého důvodu detekován, a nemůže být tedy volána funkce pro korekci, pak se tento krok přeskakuje a v následujícím snímku je zase volána funkce pro nový odhad, který určí novou následující hodnotu.

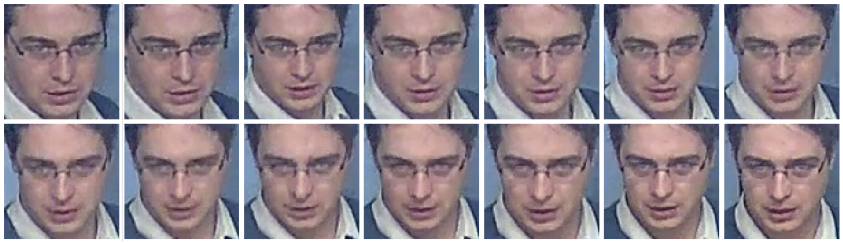
3.3.2 Algoritmus detekce obličeje

Pro detekci obličeje v obraze je využita metoda *detectMultiscale* třídy *CascadeClassifier* knihovny OpenCV. Tato metoda může být použita pro detekci různých objektů v obraze, přičemž detekovaný objekt se určuje zvoleným klasifikátorem specifikovaným v konstruktoru třídy. Metoda prochází obraz posuvným oknem, ve kterém zjišťuje příznaky daného výřezu a určuje, zda se jedná o detekovaný objekt. Příznaky se získávají použitím klasifikátoru, který se skládá z vrstev, přičemž každá vrstva získává z daného výřezu jiné příznaky a pokud některá z vrstev určí, že se o daný objekt nejedná, pak se pokračuje dalším výřezem. V opačném případě, pokud projde výřez přes všechny vrstvy a všechny určí, že se jedná o daný objekt, je výřez prohlášen za místo, kde se daný objekt nachází [5].

Protože v různých obrazech se může detekovaný objekt nacházet různě velký, prochází metoda detectMultiscale obraz několikrát s různě velkým posuvným oknem. Velikost zvětšení se určuje parametrem volání, přičemž se specifikuje, jakým koeficientem se vynásobí současná velikost okna pro získání nové velikosti. Vzhledem k tomu, že je obraz procházen vícekrát, dochází k detekci jednoho objektu v obraze vícekrát. Aby byla detekce přesnější využívá se parametru minimálního počtu sousedů. Tím se určuje, kolik je minimální počet

překrývajících se detekovaných objektů v daném místě, aby bylo možné prohlásit tuto oblast za detekovaný objekt. Je také možné specifikovat parametr minimální velikosti, které má posuvné okno mít při prvním průchodu a obdobně je také možné specifikovat jeho maximální velikost.

V aplikaci je použit klasifikátor obličeje nazvaný *haarcascade_frontalface_alt*, který je standardně dodáván v knihovně OpenCV.



Obrázek 3.2: Ukázka detekované sekvence obličejů.

3.3.3 Zarovnání obličeje

Algoritmus zarovnání obličeje pracuje s pozicemi obou očí, které jsou ve výsledném zarovnaném obrázku vždy na stejném místě. Je tedy třeba předem specifikovat pozici očí v původním obrázku, k čemuž je využita konvoluční neuronová síť. Jak již bylo zmíněno v kapitole 2.4, tato síť má jako výstup čtyři hodnoty, které určují x-ové a y-ové souřadnice levého a pravého oka. Zarovnání obličeje následně funguje tak, že se vypočítají hodnoty všech transformací, které jsou na obrázek aplikovány. Rotace obrázku vychází z pozic očí, ze kterých se vypočte úhel, o který je třeba obrázek otočit. Jako střed otáčení se volí levé oko. Ze vzdálenosti obou očí a velikosti odsazení očí od okraje ve výsledném obrázku, která je určena parametrem volání, je vypočtena míra zvětšení, případně zmenšení. Poté je obrázek otočen a zvětšen nebo zmenšen. Nakonec dojde k oříznutí obrázku a jeho opětovnému zvětšení či zmenšení tak, aby byl obrázek v požadované velikosti.

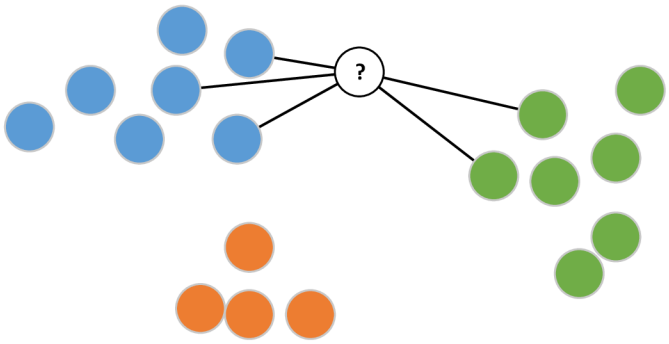
Obrázek 3.3: Ukázka zarovnání obličeje.

3.3.4 Rozpoznání obličeje pomocí algoritmu k-nejbližších sousedů

V moderních přístupech k rozpoznávání tváří se používá následující schéma: nejprve se obličej detekuje, následně se zarovná tak, aby byl co nejvhodnější k rozpoznání, daný zarovnaný obličej je určitým způsobem reprezentován a nakonec je klasifikován [12]. Část detekce je v této práci zastoupena využitím funkcí knihovny OpenCV (algoritmus popsaný v části 3.3.2) a pro zarovnání je využit algoritmus, který byl vysvětlen v kapitole 3.3.3.

K reprezentaci daného obličeje je zde využita konvoluční neuronová síť. Tato síť však neprovádí klasifikaci, ale jejím úkolem je právě reprezentovat daný obličej pomocí vektoru hodnot. Tento vektor hodnot je získán z předposlední vrstvy této sítě.

Pro samotnou klasifikaci je využit algoritmus k-nejbližších sousedů, přičemž v této práci byla využita již naprogramovaná třída *KNearest* z knihovny OpenCV. Aby bylo vůbec možné pomocí tohoto algoritmu cokoliv klasifikovat, je nejdříve nutné specifikovat známá data. Jde o trénování, kdy si třída KNearest zapamatuje dané vstupy, v tomto případě reprezentaci obličeje, a k ní přiřadí danou klasifikovanou třídu (číselná hodnota). Jednotlivé klasifikované třídy reprezentují danou osobu. Následně pak při klasifikaci nového vektoru hodnot, získaného z rozpoznávaného obličeje, je zjištěno, v jaké vzdálenosti je tento nový vektor hodnot od dříve naučených. Výstupem algoritmu pak je množina klasifikovaných tříd několika nejbližších sousedů (vektorů hodnot) a jejich vzdálenost k novému vektoru hodnot. Počet nejbližších sousedů je určen hodnotou, která je jedním z parametrů volání metody pro zjištění nejbližších sousedů.

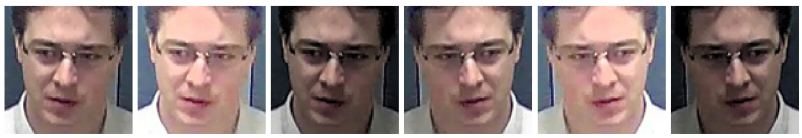


Obrázek 3.4: Ukázka principu algoritmu k-nejbližších sousedů. V tomto případě je hledáno 5 nejbližších sousedů.

Při trénování třídy KNearest se načítají obrázky, které jsou uložené v předem specifikovaném adresáři. Pro každý načtený obrázek, který má být naučen, se vytvoří ještě dalších pět jeho kopií, na nichž se provádí různé úpravy. Úpravy, které jsou použity, jsou horizontální překlopení a úprava jasu. Po úpravách je tedy k dispozici šest obrázků jednoho obličeje, a to:

1. původní obrázek
2. horizontálně překlopený obrázek
3. původní obrázek se zvýšeným jasem
4. horizontálně překlopený obrázek se zvýšeným jasem
5. původní obrázek se sníženým jasem
6. horizontálně překlopený obrázek se sníženým jasem

Důvodem těchto úprav je odstranění chybných klasifikací z důvodu opačného natočení obličeje a rozdílných světelných podmínek.



Obrázek 3.5: Ukázka úprav obličeje.

Výhodou výše zmíněné posloupnosti kroků v rozpoznávání (detekce, zarovnání, reprezentace, klasifikace) je, že konvoluční neuronová síť je natrénovaná na obecné sadě tváří a rozpoznávané osoby jsou specifikovány až při spouštění samotné aplikace. Databáze rozpoznávaných osob se tedy může při každém spuštění lišit, avšak konvoluční neuronová síť zůstává stále stejná.

Protože se v této práci získávají obličeje z videa, neklasifikuje se na základě jednoho obličeje, ale klasifikuje se celá sekvence zachycených obličejů. Pro tyto účely bylo implementováno několik algoritmů.

Algoritmus nejlepší shody

Nejspíše nejjednodušším naprogramovaným algoritmem pro klasifikaci sekvence obličejů v této práci je algoritmus, který mezi naučenými vektory hodnot hledá nejlepší shodu. V rámci aplikace je tento algoritmus označen jako *best_match*. Jeho funkce spočívá v tom, že pro každý obličej z dané sekvence je získán reprezentační vektor hodnot a následně pomocí algoritmu k-nejbližších sousedů hledá jednoho nejbližšího souseda. Mezi všemi obrázky v sekvenci se pak vyhodnocuje, který vektor hodnot byl nejblíže nějakému z naučených vektorů a celá sekvence je pak klasifikována jako třída, která reprezentuje tento nejbližší vektor.

Aby bylo možné určit tímto algoritmem neznámou osobu, je zde zakomponována možnost prahování. Pokud celkově nejbližší soused je ve vzdálenosti větší, než onen práh, pak je celá sekvence klasifikována jako neznámá osoba. Jako výstupní třída je pak použita hodnota -1.

Algoritmus lokálních procent

Dalším algoritmem pro klasifikaci sekvence tváří je algoritmus, který je v rámci aplikace označován jako *local_percentage*. Tento algoritmus vypočítává pro každý obličej ze sekvence procentuální příslušnost k dané třídě a nakonec vypočte průměrnou příslušnost celé sekvence ke všem třídám. Sekvence je pak klasifikována jako třída, která získá nejvíce procent.

Pro získání procentuální příslušnosti obličeje ke klasifikovaným třídám nejprve získají nejbližší sousedé vektoru, jež reprezentuje daný obličej. Poté se pro všechny vzdálenosti od jednotlivých tříd vypočítá převrácená hodnota. Tímto se docílí toho, že bližší vzdálenosti získají vyšší hodnotu, než vzdálenosti, které jsou dál. Procentuální příslušnost se nakonec vypočte jako poměr mezi převrácenou hodnotou vzdálenosti daného nejbližšího souseda a celkovou sumou převrácených hodnot vzdáleností všech sousedů. Pokud je více sousedů se stejnou třídou, pak jsou tato procenta příslušnosti k dané třídě sečtena.

Aby bylo opět možné klasifikovat osobu jako neznámou, je možné nastavit práh. Pokud se některý z nejbližších sousedů nachází ve vzdálenosti větší než je uvedený práh, pak je jeho třída změněna na hodnotu -1, která reprezentuje neznámou osobu.

Algoritmus globálních procent

Poslední algoritmus, který byl v rámci této práce naprogramován pro klasifikaci sekvence tváří, je v aplikaci označen jako *global_percentage*. Algoritmus využívá podobného principu jako algoritmus *local_percentage*, avšak liší se v některých krocích. Pro každý vektor hodnot reprezentující obličej jsou zjištěni nejbližší sousedé a jejich vzájemná vzdálenost. Všechny tyto hodnoty (třídy a jejich vzdálenosti) jsou následně seskupeny do jednoho dvourozměrného pole, nad kterým se provede výpočet převrácené hodnoty vzdálenosti, všechny tyto převrácené hodnoty jsou sečteny a pro každý záznam v onom poli je vypočten podíl dané převrácené hodnoty vzdálenosti a celkové sumy. Toto ohodnocení je nakonec pro stejné třídy sečteno, čímž se získá procentuální příslušnost dané sekvence k dané třídě.

I u tohoto algoritmu se využívá prahování pro určení neznámé osoby. Třídy, jejichž vzdálenost překročí nastavený práh, jsou změněny na hodnotu -1.

Tento algoritmus byl implementován z toho důvodu, aby více upřednostňoval třídy, pro které byla vzdálenost sousedů nižší. U algoritmu *local_percentage* totiž může dojít k tomu, že dva různé výřezy obličeje mohou získat podobná procenta příslušnosti k různým třídám, ale do celkového vyhodnocení již není zahrnut fakt, že pro jeden z obličejů mohli být nejbližší sousedé daleko blíže než u druhého.

3.3.5 Třídy využité při učení konvolučních neuronových sítí

Níže zmíněné třídy byly získány z online návodu, který se zabývá použitím knihovny nolearn [11].

FlipBatchIterator

Třídy *FaceFlipBatchIterator* a *EyeFlipBatchIterator* přispívají k lepšímu naučení sítě tím, že během učení upravují trénovací sadu a případně taky požadované výstupy. Při trénování neuronových sítí je vhodné mít co největší trénovací sadu. Malou velikost sady je však možné nahradit technikami, jako je například převrácení obličejů během trénování. Výsledkem toho pak je, že původní data jsou v průběhu trénování zvětšena až na dvojnásobek (každý obličej je možné mít dvakrát – jednou v původní poloze a jednou v překlopené).

Během trénování se po každé iteraci náhodně vybere několik obličejů a ty se horizontálně překlopí. Pokud se jedná o EyeFlipBatchIterator, je nutné také prohodit levé oko za pravé, respektive přehodit jejich x-ové souřadnice. U FaceFlipBatchIteratoru není třeba měnit výstup, protože klasifikovaná třída je stále stejná.

AdjustVarible

Třídy *FaceAdjustVariable* a *EyeAdjustVariable* slouží při trénování k tomu, aby během učení postupně upravovali zadanou vlastnost neuronové sítě. Jde zejména o úpravu hodnoty rychlosti učení (viz kapitola 2.5.1). Během inicializace konvoluční neuronové sítě se nastaví výchozí hodnota a také hodnota při poslední iteraci (při inicializaci se nastavuje také maximální počet iterací trénování). Po vytvoření nové instance třídy jsou vypočteny hodnoty pro každou iteraci během trénování a po skončení každé iterace se následně upraví rychlost učení pro tu nadcházející. Běžné použití je takové, že na začátku je rychlost učení vyšší a následně se s přibývajícími iteracemi snižuje.

EarlyStopping

Jak bylo zmíněno v kapitole 2.5.2, je vhodné správně odhadnout, kdy je síť již dostatečně natrénovaná a zabránit tak přetrénování. K tomu slouží třída *EarlyStopping*. Jejím úkolem je uchovávat hodnoty jednotlivých neuronů a konvolučních vrstev z poslední iterace, při které došlo ke zlepšení na validační sadě. Pokud již však určitý počet iterací nedošlo ke zlepšení, načte poslední uložené parametry sítě a vyvolá tato třída výjimku *StopIteration*. V tomto okamžiku učení skončí.

Kapitola 4

Výsledky a vyhodnocení

V první části této kapitoly budou poskytnuty informace ohledně natrénovaných konvolučních sítí. V druhé části budou popsány vytvořené testy, na kterých probíhalo vyhodnocování konvolučních neuronových sítí pro rozpoznávání obličejů a implementovaných algoritmů.

4.1 Natrénované konvoluční neuronové sítě

4.1.1 Konvoluční neuronová síť pro detekci očí

Jak již bylo zmíněno v předchozích kapitolách, jedna z konvolučních neuronových sítí v této práci byla natrénována na detekci očí v obličeji. Tato neuronová síť byla trénována na datové sadě *BioID Face Database*. Tato datová sada obsahuje 1521 obrázků obličejů ve stupních šedi 23 různých osob. Dále jsou zde pro každý obličej přidány informace o x-ových a y-ových pozicích levého a pravého oka.

Příprava dat

Nejprve je v každém obrázku detekován obličej, jenž je uložen do seznamu trénovaných obličejů, přičemž hodnoty jednotlivých bodů obrazu jsou převedeny do rozmezí od 0 do 1. Poté jsou z příslušného souboru přečteny souřadnice obou oči, jejichž hodnota se převede z pozice v původním obraze do hodnot v rozmezí od -1 do 1 v obou osách extrahovaného obličeje. Následně jsou tyto hodnoty jako čtveřice přidány do seznamu výstupních hodnot a síť se začne na těchto datech trénovat.

Architektura sítě

Architektura sítě je vidět v tabulce 4.1. Ve sloupci parametry vrstvy je význam hodnot následující:

- Pro vstupní vrstvu udává velikost vstupních dat, tj. rozměry vstupního obrázku v bodech.
- Pro konvoluční vrstvu udává počet filtrů krát velikost jednoho filtru v bodech, tj. 16x 3x3 znamená 16 filtrů o velikosti 3 krát 3 body
- Pro pooling vrstvu udává velikost filtru v bodech
- Pro plně propojenou vrstvu udává počet neuronů ve vrstvě

- Pro výstupní vrstvu udává počet výstupních hodnot

Č.	Název vrstvy	Typ vrstvy	Parametry vrstvy
1	input	vstupní vrstva	64x64
2	conv1	konvoluční vrstva	32x 3x3
3	pool1	max-pooling vrstva	3x3
4	conv2	konvoluční vrstva	64x 2x2
5	pool2	max-pooling vrstva	2x2
6	hidden4	plně propojená vrstva	1500
7	hidden5	plně propojená vrstva	150
8	output	výstupní vrstva	4

Tabulka 4.1: Architektura konvoluční neuronové sítě pro detekci očí.

Trénování sítě

Pro trénování této sítě byla nastavena počáteční hodnota rychlosti učení na hodnotu 0.01, která se zmenšovala až na hodnotu 0.00001 pro poslední iteraci. Trénování bylo nastaveno tak, aby se učení zastavilo po 50 neúspěšných iteracích, přičemž v každé epoše bylo horizontálně převráceno 256 vstupních dat a tomu odpovídajících výstupních hodnot. Po natrénování sítě byla na testovací sadě vypočtena průměrná chyba, tj. průměrná vzdálenost predikované hodnoty a skutečné pozice, 4.561 bodů.

4.1.2 Konvoluční neuronové sítě pro rozpoznávání obličejů

Nejdůležitější součástí aplikace z pohledu rozpoznávání tváří je právě konvoluční neuronová síť. Neuronových sítí, které jsou při rozpoznávání použity, bylo natrénováno hned několik, avšak pro testování byly zvoleny jen dvě. Rozdíl mezi těmito sítěmi tkví v rozdílných velikostech vstupních dat (velikosti obrázku obličeje) a z toho vyplývajících rozdílných architekturách sítí.

Příprava dat

Konvoluční neuronové sítě pro rozpoznávání obličejů byly trénovány na části datové sady *YouTube Faces DB* [15], která obsahuje obličeje lidí z videí serveru YouTube. Z celé datové sady, která čítá 3425 obrázků 1595 různých lidí, bylo náhodně vybráno 150 osob. Pro každou osobu pak bylo náhodně vybráno 15 snímků obličeje, které byly zarovnány. Na této datové sadě následně probíhalo učení, přičemž hodnoty jednotlivých bodů obrazu byly převedeny do rozmezí od 0 do 1.

Síť 1

Tato síť byla trénována s počáteční rychlostí učení o hodnotě 0.001 a po 222 epochách dosahovala na validační sadě úspěšnosti rozpoznávání 94,745 %. Architektura sítě je vidět v tabulce 4.2.

Č.	Název vrstvy	Typ vrstvy	Parametry vrstvy
1	input	vstupní vrstva	48x48
2	conv1	konvoluční vrstva	32x 3x3
3	pool1	max-pooling vrstva	2x2
4	conv2	konvoluční vrstva	64x 2x2
5	pool2	max-pooling vrstva	2x2
6	hidden4	plně propojená vrstva	2500
7	hidden5	plně propojená vrstva	1000
8	hidden6	plně propojená vrstva	400
9	output	výstupní vrstva	150

Tabulka 4.2: Architektura konvoluční neuronové sítě pro rozpoznání obličejů 1.

Síť 2

Tato síť, jež dosáhla na validační sadě po 259 epochách úspěšnosti rozpoznávání 93,555 %, byla inicializována s rychlostí učení o hodnotě 0.001. Architektura sítě je vidět v tabulce 4.3.

Č.	Název vrstvy	Typ vrstvy	Parametry vrstvy
1	input	vstupní vrstva	64x64
2	conv1	konvoluční vrstva	16x 3x3
3	pool1	max-pooling vrstva	2x2
4	conv2	konvoluční vrstva	32x 2x2
5	pool2	max-pooling vrstva	2x2
6	conv2	konvoluční vrstva	64x 2x2
7	pool2	max-pooling vrstva	2x2
8	hidden4	plně propojená vrstva	1200
9	hidden5	plně propojená vrstva	500
10	hidden6	plně propojená vrstva	200
11	output	výstupní vrstva	150

Tabulka 4.3: Architektura konvoluční neuronové sítě pro rozpoznání obličejů 2.

4.2 Výsledky testování

Za účelem testování byly vytvořeny dva testy. Každou testovací sadu tvoří takzvaná databáze a testovací data. Databáze obsahuje seznam osob, které bude aplikace rozpoznávat, přičemž pro natrénování každé osoby je zde použit jeden její průchod před kamerou. Testovací data pak obsahují další průchody osob z databáze i průchody osob, které by aplikace měla vyhodnotit jako neznámé.

Testy byly spouštěny s různým nastavením parametrů, které ovlivňují samotný proces rozpoznávání. Jedná se o použitou neuronovou síť, použitý algoritmus, zjišťovaný počet nejbližších sousedů a hranici prahu.

4.2.1 Vyhodnocení výsledků

Výsledků jednotlivých testování budou reprezentovány *ROC křivkou*, *tabulkou úspěšnosti rozpoznávání* a *tabulkou záměn*.

ROC (*Receiver Operating Characteristic*) křivka je křivka znázorňující vztah mezi *specificitou* a *senzitivitou* daného systému. Tato charakteristika se používá při klasifikaci do dvou tříd. Na horizontální osu se obvykle vynáší *falešná pozitivita* (anglicky *False positive rate*, zkráceně *FP*) a na vertikální osu se vynáší *skutečná pozitivita* (anglicky *True positive rate*, zkráceně *TP*). Obvykle je cílem, aby hodnota skutečné pozitivity byla co nejvyšší (blížila se 100 %) a falešná pozitivita byla naopak co nejnižší (blížila se hodnotě 0 %).

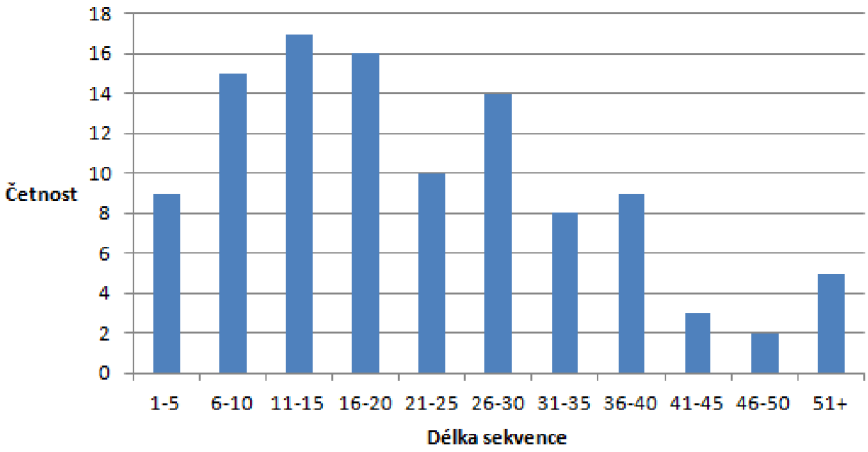
Při vyhodnocování testu ROC křivkou jsou jednotlivé testované sekvence obličejů klasifikovány jako dvě třídy – známá osoba a neznámá osoba. Skutečná pozitivita udává podíl osob, které byly správně určeny jako známé, ku celkovému počtu známých osob. Pokud je tedy tato hodnota 100 %, pak to znamená, že byly všechny známé osoby klasifikovány jako známé. Falešná pozitivita udává podíl osob klasifikovaných jako známá osoba, přičemž se jedná o osobu neznámou, ku celkovému počtu neznámých osob. Pokud je tato hodnota 100 %, pak to znamená, že všechny neznámé osoby byly klasifikovány jako známé.

Tabulka záměn (také *matice záměn*, anglicky *confusion matrix*) je tabulka, která zobrazuje četnost záměn dvou tříd. Sloupce vyjadřují rozpoznanou třídu a řádky reprezentují skutečnou třídu. Hodnota v tabulce následně udává počet záměn daných dvou tříd. Například hodnota tři ve sloupci nazvaném „třída 1“ a na řádku nazvaném „třída 2“ ukazuje, že systém třikrát rozpoznal „třídu 1“, avšak správně měl určit „třídu 2“. Na diagonále se potom nachází počet správně rozpoznaných případů. Vyhodnocení touto tabulkou je použito vždy pro každý použitý algoritmus při nejvyšším nastaveném prahu testu 1.

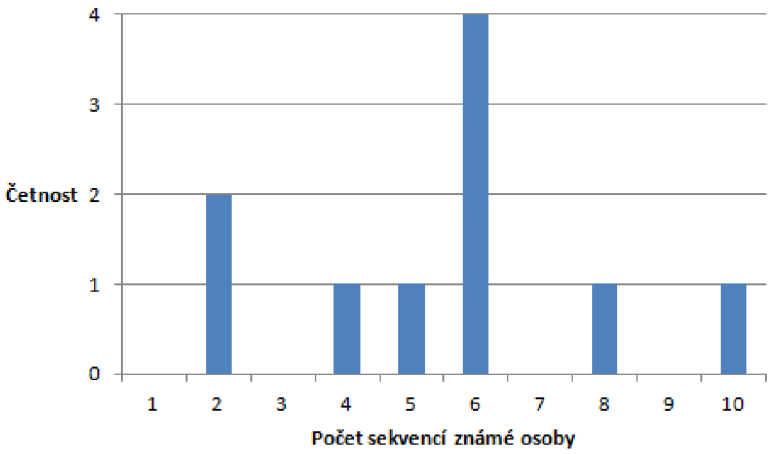
4.2.2 Test 1

V tomto testu je rozpoznáváno 10 osob, přičemž testovací sada obsahuje 109 sekvencí obličejů. Tuto sadu tvoří 55 sekvencí obličejů známých osob a 54 sekvencí obličejů neznámých osob. Na následujících histogramech je znázorněna struktura dat. Histogram 4.1 zobrazuje četnost obrázků obličeje v rámci jedné sekvence, histogram 4.2 zobrazuje počet sekvencí známých osob.

Tento test může simulovat například zabezpečení budovy (určitého místa), kam mají přístup pouze oprávněné osoby s případným zaznamenáním, která osoba se v daném místě nacházela.

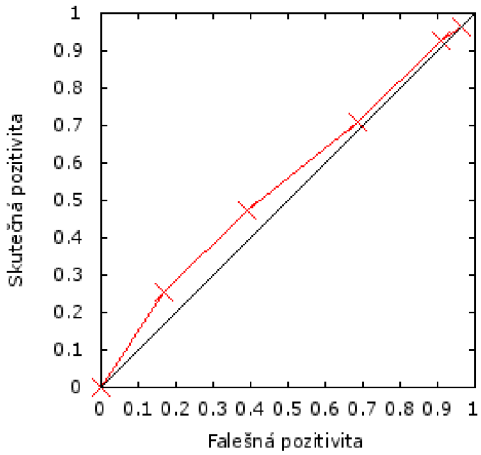


Obrázek 4.1: Histogram délky sekvence.



Obrázek 4.2: Histogram počtu sekvencí známých osob.

Výsledky pro síť 1



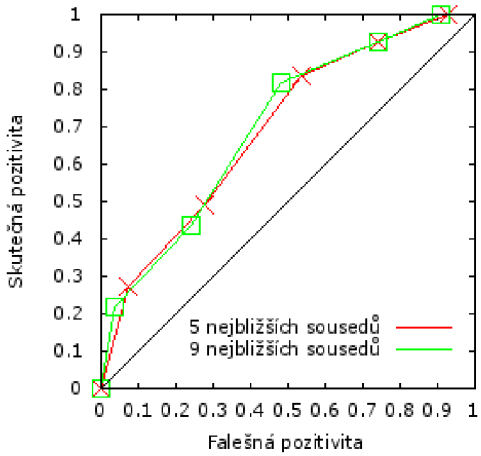
Obrázek 4.3: ROC křivka testu 1 pro algoritmus *nejlepší shody* sítě 1.

Práh	Falešná pozitivita	Skutečná pozitivita	Úspěšnost rozpoznávání
750	16.6667 %	25.4545 %	20.0000 %
1250	38.8889 %	47.2727 %	36.3636 %
2000	68.5185 %	70.9091 %	50.9091 %
3500	90.7407 %	92.7273 %	65.4545 %
5000	96.2963 %	96.3636 %	69.0909 %

Tabulka 4.4: Tabulka výsledků testu 1 pro síť 1 při použití algoritmu *nejlepší shody*.

		Rozpoznaná osoba										
		Kapinus	Bařina	Černocký	Beran	Dytrych	Grézl	Herout	Pešán	Smrž	Hradiš	neznámá osoba
Skutečná osoba	Kapinus	2	0	0	0	0	0	0	0	0	0	0
	Bařina	0	2	0	0	0	1	0	0	1	0	0
	Černocký	0	0	5	0	0	0	0	0	0	0	1
	Beran	0	0	0	4	0	0	1	0	3	0	0
	Dytrych	1	0	0	1	7	0	0	0	1	0	0
	Grézl	0	0	0	0	2	3	0	0	0	0	1
	Herout	0	0	0	0	1	0	4	0	0	0	0
	Pešán	0	0	0	2	0	0	0	4	0	0	0
	Smrž	0	0	0	0	0	0	0	0	1	1	0
	Hradiš	0	0	0	0	0	0	0	0	0	6	0
	neznámá osoba	0	0	0	0	0	0	0	0	0	0	0

Tabulka 4.5: Tabulka záměn testu 1 pro síť 1 při použití algoritmu *nejlepší shody* a prahu 5000.



Obrázek 4.4: ROC křivka testu 1 pro algoritmus *lokálních procent* sítě 1.

Práh	Falešná pozitivita	Skutečná pozitivita	Úspěšnost rozpoznávání
2000	7.4074 %	27.2727 %	21.8182 %
3000	27.7778 %	49.0909 %	36.3636 %
4500	53.7037 %	83.6364 %	60.0000 %
6000	74.0741 %	92.7273 %	67.2727 %
8000	92.5926 %	100.0000 %	72.7273 %

Tabulka 4.6: Tabulka výsledků testu 1 pro síť 1 při použití algoritmu *lokálních procent* s nastavením 5 nejbližších sousedů.

		Rozpoznaná osoba										
		Kapinus	Bařina	Černocký	Beran	Dytrych	Grézl	Herout	Pešán	Smrž	Hradiš	neznámá osoba
Skutečná osoba	Kapinus	2	0	0	0	0	0	0	0	0	0	0
	Bařina	0	1	0	1	0	2	0	0	0	0	0
	Černocký	0	0	6	0	0	0	0	0	0	0	0
	Beran	0	0	0	6	0	0	0	0	2	0	0
	Dytrych	1	0	0	2	7	0	0	0	0	0	0
	Grézl	0	0	0	0	1	5	0	0	0	0	0
	Herout	0	0	0	0	1	0	4	0	0	0	0
	Pešán	0	1	1	0	1	0	0	3	0	0	0
	Smrž	0	0	1	0	0	0	0	0	1	0	0
	Hradiš	1	0	0	0	0	0	0	0	0	5	0
	neznámá osoba	0	0	0	0	0	0	0	0	0	0	0

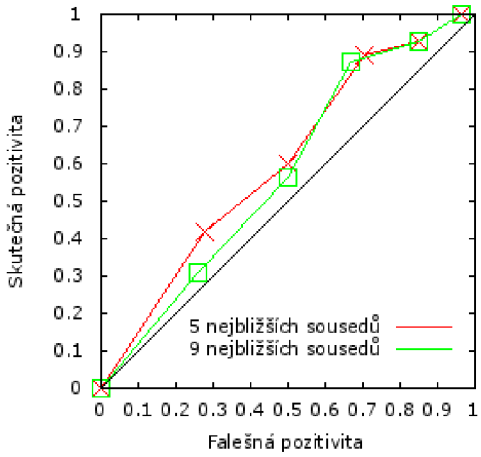
Tabulka 4.7: Tabulka záměn testu 1 pro síť 1 při použití algoritmu *lokálních procent*, 5 nejbližších sousedů a prahu 8000.

Práh	Falešná pozitivita	Skutečná pozitivita	Úspěšnost rozpoznávání
2000	3.7037 %	21.8182 %	16.3636 %
3000	24.0741 %	43.6364 %	34.5455 %
4500	48.1481 %	81.8182 %	58.1818 %
6000	74.0741 %	92.7273 %	69.0909 %
8000	90.7407 %	100.0000 %	74.5455 %

Tabulka 4.8: Tabulka výsledků testu 1 pro síť 1 při použití algoritmu *lokálních procent* s nastavením 9 nejbližších sousedů.

		Rozpoznaná osoba										
		Kapinus	Bařina	Černocký	Beran	Dytrych	Grézl	Herout	Pešán	Smrž	Hradiš	neznámá osoba
Skutečná osoba	Kapinus	2	0	0	0	0	0	0	0	0	0	0
	Bařina	0	1	0	1	0	2	0	0	0	0	0
	Černocký	0	0	6	0	0	0	0	0	0	0	0
	Beran	0	0	0	6	0	0	0	0	2	0	0
	Dytrych	1	0	0	2	7	0	0	0	0	0	0
	Grézl	0	0	0	0	1	5	0	0	0	0	0
	Herout	0	0	0	0	1	0	4	0	0	0	0
	Pešán	0	1	1	0	1	0	0	3	0	0	0
	Smrž	0	0	0	0	0	0	0	0	2	0	0
	Hradiš	1	0	0	0	0	0	0	0	0	5	0
	neznámá osoba	0	0	0	0	0	0	0	0	0	0	0

Tabulka 4.9: Tabulka záměn testu 1 pro síť 1 při použití algoritmu *lokálních procent*, 9 nejbližších sousedů a prahu 8000.



Obrázek 4.5: ROC křivka testu 1 pro algoritmus *globálních procent* sítě 1.

Práh	Falešná pozitivita	Skutečná pozitivita	Úspěšnost rozpoznávání
2000	27.7778 %	41.8182 %	32.7273 %
3000	50.0000 %	60.0000 %	47.2727 %
4500	70.3704 %	89.0909 %	67.2727 %
6000	85.1852 %	92.7273 %	69.0909 %
8000	96.2963 %	100.0000 %	74.5455 %

Tabulka 4.10: Tabulka výsledků testu 1 pro síť 1 při použití algoritmu *globálních procent* s nastavením 5 nejbližších sousedů.

		Rozpoznaná osoba										
		Kapinus	Bařina	Černocký	Beran	Dytrych	Grézl	Herout	Pešán	Smrž	Hradiš	neznámá osoba
Skutečná osoba	Kapinus	2	0	0	0	0	0	0	0	0	0	0
	Bařina	0	1	0	1	0	2	0	0	0	0	0
	Černocký	0	0	6	0	0	0	0	0	0	0	0
	Beran	0	0	0	6	0	0	0	0	2	0	0
	Dytrych	0	0	0	2	7	0	0	0	1	0	0
	Grézl	0	0	0	0	1	5	0	0	0	0	0
	Herout	0	0	0	0	1	0	4	0	0	0	0
	Pešán	0	1	1	0	1	0	0	3	0	0	0
	Smrž	0	0	0	0	0	0	0	0	2	0	0
	Hradiš	1	0	0	0	0	0	0	0	0	5	0
	neznámá osoba	0	0	0	0	0	0	0	0	0	0	0

Tabulka 4.11: Tabulka záměn testu 1 pro síť 1 při použití algoritmu *globálních procent*, 5 nejbližších sousedů a prahu 8000.

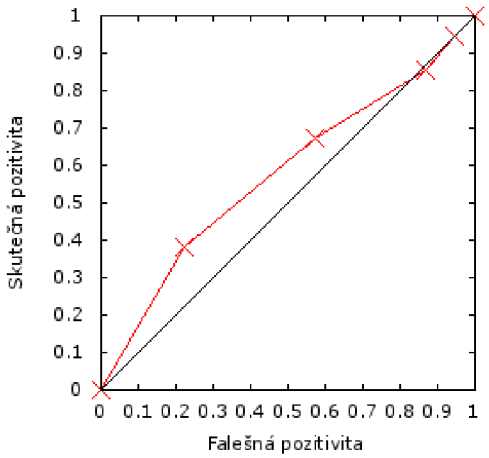
Práh	Falešná pozitivita	Skutečná pozitivita	Úspěšnost rozpoznávání
2000	25.9259 %	30.9091 %	21.8182 %
3000	50.0000 %	56.3636 %	43.6364 %
4500	66.6667 %	87.2727 %	63.6364 %
6000	85.1852 %	92.7273 %	69.0909 %
8000	96.2963 %	100.0000 %	74.5455 %

Tabulka 4.12: Tabulka výsledků testu 1 pro síť 1 při použití algoritmu *globálních procent* s nastavením 9 nejbližších sousedů.

		Rozpoznaná osoba										
		Kapinus	Bařina	Černocký	Beran	Dytrych	Grézl	Herout	Pešán	Smrž	Hradiš	neznámá osoba
Skutečná osoba	Kapinus	2	0	0	0	0	0	0	0	0	0	0
	Bařina	0	1	0	1	0	2	0	0	0	0	0
	Černocký	0	0	6	0	0	0	0	0	0	0	0
	Beran	0	0	0	6	0	0	0	0	2	0	0
	Dytrych	0	0	0	2	7	0	0	0	1	0	0
	Grézl	0	0	0	0	1	5	0	0	0	0	0
	Herout	0	0	0	0	1	0	4	0	0	0	0
	Pešán	0	1	1	0	1	0	0	3	0	0	0
	Smrž	0	0	0	0	0	0	0	0	2	0	0
	Hradiš	1	0	0	0	0	0	0	0	0	5	0
	neznámá osoba	0	0	0	0	0	0	0	0	0	0	0

Tabulka 4.13: Tabulka záměn testu 1 pro síť 1 při použití algoritmu *globálních procent*, 9 nejbližších sousedů a prahu 8000.

Výsledky pro síť 2



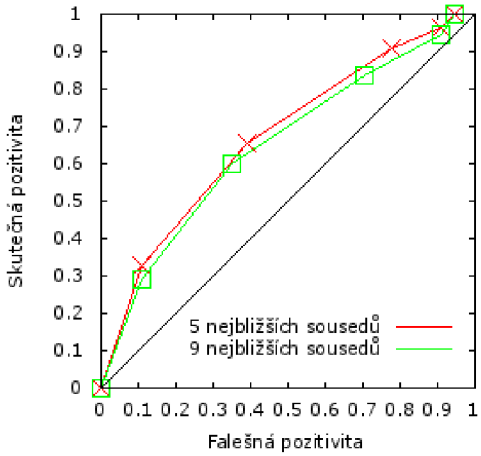
Obrázek 4.6: ROC křivka testu 1 pro algoritmus *nejlepší shody* sítě 2.

Práh	Falešná pozitivita	Skutečná pozitivita	Úspěšnost rozpoznávání
750	22.2222 %	38.1818 %	23.6364 %
1250	57.4074 %	67.2727 %	36.3636 %
2000	87.0370 %	85.4545 %	43.6364 %
3500	94.4444 %	94.5455 %	47.2727 %
5000	100.0000 %	100.0000 %	52.7273 %

Tabulka 4.14: Tabulka výsledků testu 1 pro síť 2 při použití algoritmu *nejlepší shody*.

		Rozpoznaná osoba										
		Kapinus	Bařina	Černocký	Beran	Dytrych	Grézl	Herout	Pešán	Smrž	Hradiš	neznámá osoba
Skutečná osoba	Kapinus	2	0	0	0	0	0	0	0	0	0	0
	Bařina	0	1	0	1	0	2	0	0	0	0	0
	Černocký	0	0	4	2	0	0	0	0	0	0	0
	Beran	1	0	0	3	0	0	1	0	3	0	0
	Dytrych	0	0	0	1	7	0	0	0	2	0	0
	Grézl	0	0	0	0	2	4	0	0	0	0	0
	Herout	0	0	0	1	1	0	1	1	1	0	0
	Pešán	1	0	0	3	0	0	0	2	0	0	0
	Smrž	0	0	0	1	0	0	0	0	1	0	0
	Hradiš	0	0	0	0	1	0	0	0	1	4	0
	neznámá osoba	0	0	0	0	0	0	0	0	0	0	0

Tabulka 4.15: Tabulka záměn testu 1 pro síť 2 při použití algoritmu *nejlepší shody* a prahu 5000.



Obrázek 4.7: ROC křivka testu 1 pro algoritmus *lokálních procent* sítě 2.

Práh	Falešná pozitivita	Skutečná pozitivita	Úspěšnost rozpoznávání
2000	11.1111 %	32.7273 %	21.8182 %
3000	38.8889 %	65.4545 %	34.5455 %
4500	77.7778 %	90.9091 %	52.7273 %
6000	90.7407 %	96.3636 %	54.5455 %
8000	94.4444 %	100.0000 %	58.1818 %

Tabulka 4.16: Tabulka výsledků testu 1 pro síť 2 při použití algoritmu *lokálních procent* s nastavením 5 nejbližších sousedů.

		Rozpoznaná osoba										
		Kapinus	Bařina	Černocký	Beran	Dytrych	Grézl	Herout	Pešán	Smrž	Hradiš	neznámá osoba
Skutečná osoba	Kapinus	2	0	0	0	0	0	0	0	0	0	0
	Bařina	0	1	0	1	0	2	0	0	0	0	0
	Černocký	0	0	4	1	0	0	0	0	0	1	0
	Beran	1	0	0	4	0	0	0	0	3	0	0
	Dytrych	2	0	0	1	7	0	0	0	0	0	0
	Grézl	0	1	0	0	2	3	0	0	0	0	0
	Herout	0	0	0	1	1	0	3	0	0	0	0
	Pešán	0	0	0	2	1	1	0	2	0	0	0
	Smrž	0	0	0	0	0	0	0	0	2	0	0
	Hradiš	0	0	0	0	1	0	0	0	1	4	0
	neznámá osoba	0	0	0	0	0	0	0	0	0	0	0

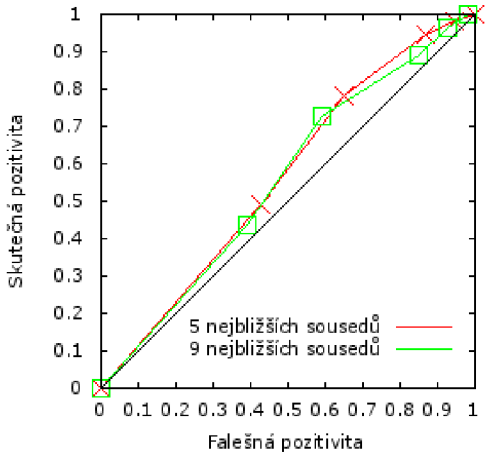
Tabulka 4.17: Tabulka záměn testu 1 pro síť 2 při použití algoritmu *lokálních procent*, 5 nejbližších sousedů a prahu 8000.

Práh	Falešná pozitivita	Skutečná pozitivita	Úspěšnost rozpoznávání
2000	11.1111 %	29.0909 %	20.0000 %
3000	35.1852 %	60.0000 %	32.7273 %
4500	70.3704 %	83.6364 %	45.4545 %
6000	90.7407 %	94.5455 %	54.5455 %
8000	94.4444 %	100.0000 %	58.1818 %

Tabulka 4.18: Tabulka výsledků testu 1 pro síť 2 při použití algoritmu *lokálních procent* s nastavením 9 nejbližších sousedů.

		Rozpoznaná osoba										
		Kapinus	Bařina	Černocký	Beran	Dytrych	Grézl	Herout	Pešán	Smrž	Hradiš	neznámá osoba
Skutečná osoba	Kapinus	2	0	0	0	0	0	0	0	0	0	0
	Bařina	0	1	0	1	0	2	0	0	0	0	0
	Černocký	0	0	4	1	0	0	0	0	0	1	0
	Beran	1	0	0	4	0	0	0	0	3	0	0
	Dytrych	1	0	0	1	7	0	0	0	1	0	0
	Grézl	0	1	0	0	3	2	0	0	0	0	0
	Herout	0	0	0	1	1	0	3	0	0	0	0
	Pešán	0	0	0	1	1	1	0	3	0	0	0
	Smrž	0	0	0	0	0	0	0	0	2	0	0
	Hradiš	0	0	0	0	1	0	0	0	1	4	0
	neznámá osoba	0	0	0	0	0	0	0	0	0	0	0

Tabulka 4.19: Tabulka záměn testu 1 pro síť 2 při použití algoritmu *lokálních procent*, 9 nejbližších sousedů a prahu 8000.



Obrázek 4.8: ROC křivka testu 1 pro algoritmus *globálních procent* sítě 2.

Práh	Falešná pozitivita	Skutečná pozitivita	Úspěšnost rozpoznávání
2000	42.5926 %	49.0909 %	30.9091 %
3000	64.8148 %	78.1818 %	43.6364 %
4500	87.0370 %	94.5455 %	56.3636 %
6000	94.4444 %	98.1818 %	58.1818 %
8000	100.0000 %	100.0000 %	60.0000 %

Tabulka 4.20: Tabulka výsledků testu 1 pro síť 2 při použití algoritmu *globálních procent* s nastavením 5 nejbližších sousedů.

		Rozpoznaná osoba										
		Kapinus	Bařina	Černocký	Beran	Dytrych	Grézl	Herout	Pešán	Smrž	Hradiš	neznámá osoba
Skutečná osoba	Kapinus	2	0	0	0	0	0	0	0	0	0	0
	Bařina	0	1	0	1	0	2	0	0	0	0	0
	Černocký	0	0	4	1	0	0	0	0	0	1	0
	Beran	1	0	0	4	0	0	0	0	3	0	0
	Dytrych	2	0	0	1	7	0	0	0	0	0	0
	Grézl	0	0	0	0	2	4	0	0	0	0	0
	Herout	0	0	0	1	1	0	3	0	0	0	0
	Pešán	1	0	0	2	1	0	0	2	0	0	0
	Smrž	0	0	0	0	0	0	0	0	2	0	0
	Hradiš	0	0	0	0	1	0	0	0	1	4	0
	neznámá osoba	0	0	0	0	0	0	0	0	0	0	0

Tabulka 4.21: Tabulka záměn testu 1 pro síť 2 při použití algoritmu *globálních procent*, 5 nejbližších sousedů a prahu 8000.

Práh	Falešná pozitivita	Skutečná pozitivita	Úspěšnost rozpoznávání
2000	38.8889 %	43.6364 %	25.4545 %
3000	59.2593 %	72.7273 %	41.8182 %
4500	85.1852 %	89.0909 %	50.9091 %
6000	92.5926 %	96.3636 %	58.1818 %
8000	98.1481 %	100.0000 %	60.0000 %

Tabulka 4.22: Tabulka výsledků testu 1 pro síť 2 při použití algoritmu *globálních procent* s nastavením 9 nejbližších sousedů.

		Rozpoznaná osoba										
		Kapinus	Bařina	Černocký	Beran	Dytrych	Grézl	Herout	Pešán	Smrž	Hradiš	neznámá osoba
Skutečná osoba	Kapinus	2	0	0	0	0	0	0	0	0	0	0
	Bařina	0	1	0	1	0	2	0	0	0	0	0
	Černocký	0	0	4	1	0	0	0	0	0	1	0
	Beran	0	0	0	5	0	0	0	0	3	0	0
	Dytrych	1	0	0	1	7	0	0	0	1	0	0
	Grézl	0	0	0	0	3	3	0	0	0	0	0
	Herout	0	0	0	1	1	0	3	0	0	0	0
	Pešán	0	0	0	2	1	1	0	2	0	0	0
	Smrž	0	0	0	0	0	0	0	0	2	0	0
	Hradiš	0	0	0	0	1	0	0	0	1	4	0
	neznámá osoba	0	0	0	0	0	0	0	0	0	0	0

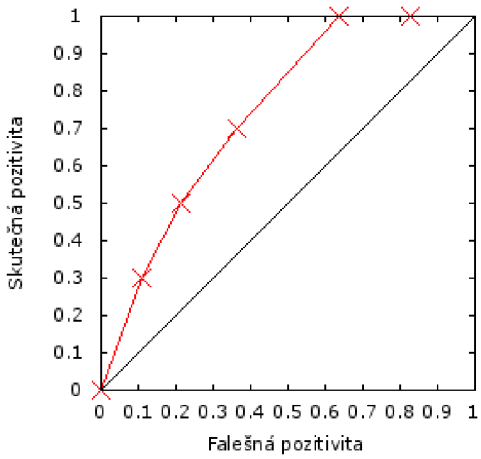
Tabulka 4.23: Tabulka záměn testu 1 pro síť 2 při použití algoritmu *globálních procent*, 9 nejbližších sousedů a prahu 8000.

4.2.3 Test 2

V tomto testu je rozpoznávána jedna osoba mezi množstvím neznámých osob. Testovací sada obsahuje stejnou sadu sekvencí obličejů jako předchozí test. Je zde celkem 109 sekvencí, mezi kterými je deset sekvencí rozpoznávané osoby.

Tento test může napodobovat například rozpoznávání hledané osoby ve veřejném prostoru, který je sledován bezpečnostními kamerami.

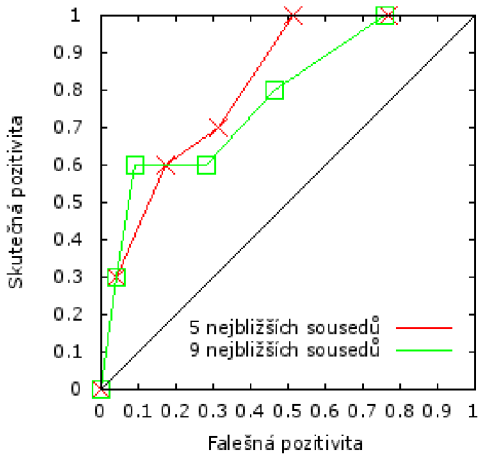
Výsledky pro síť 1



Obrázek 4.9: ROC křivka testu 2 pro algoritmus *nejlepší shody* sítě 1.

Práh	Falešná pozitivita	Skutečná pozitivita	Úspěšnost rozpoznávání
750	11.1111 %	30.0000 %	30.0000 %
1250	21.2121 %	50.0000 %	50.0000 %
2000	36.3636 %	70.0000 %	70.0000 %
3500	63.6364 %	100.0000 %	100.0000 %
5000	82.8283 %	100.0000 %	100.0000 %

Tabulka 4.24: Tabulka výsledků testu 2 pro síť 1 při použití algoritmu *nejlepší shody*.



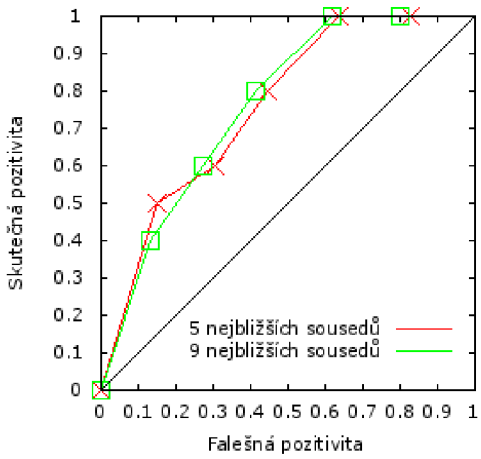
Obrázek 4.10: ROC křivka testu 2 pro algoritmus *lokálních procent* sítě 1.

Práh	Falešná pozitivita	Skutečná pozitivita	Úspěšnost rozpoznávání
2000	4.0404 %	30.0000 %	30.0000 %
3000	17.1717 %	60.0000 %	60.0000 %
4500	31.3131 %	70.0000 %	70.0000 %
6000	51.5152 %	100.0000 %	100.0000 %
8000	76.7677 %	100.0000 %	100.0000 %

Tabulka 4.25: Tabulka výsledků testu 2 pro síť 1 při použití algoritmu *lokálních procent* s nastavením 5 nejbližších sousedů.

Práh	Falešná pozitivita	Skutečná pozitivita	Úspěšnost rozpoznávání
2000	4.0404 %	30.0000 %	30.0000 %
3000	9.0909 %	60.0000 %	60.0000 %
4500	28.2828 %	60.0000 %	60.0000 %
6000	46.4646 %	80.0000 %	80.0000 %
8000	75.7576 %	100.0000 %	100.0000 %

Tabulka 4.26: Tabulka výsledků testu 2 pro síť 1 při použití algoritmu *lokálních procent* s nastavením 9 nejbližších sousedů.



Obrázek 4.11: ROC křivka testu 2 pro algoritmus *globálních procent* sítě 1.

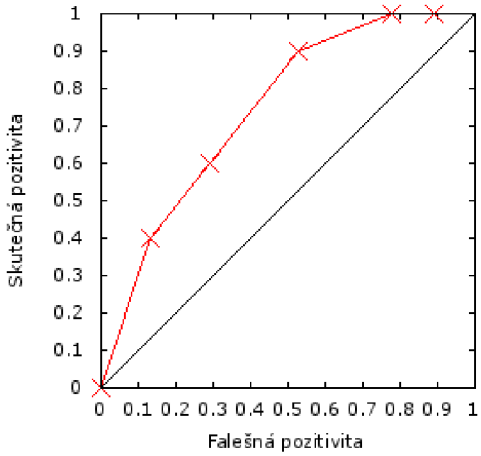
Práh	Falešná pozitivita	Skutečná pozitivita	Úspěšnost rozpoznávání
2000	15.1515 %	50.0000 %	50.0000 %
3000	30.3030 %	60.0000 %	60.0000 %
4500	44.4444 %	80.0000 %	80.0000 %
6000	63.6364 %	100.0000 %	100.0000 %
8000	82.8283 %	100.0000 %	100.0000 %

Tabulka 4.27: Tabulka výsledků testu 2 pro síť 1 při použití algoritmu *globálních procent* s nastavením 5 nejbližších sousedů.

Práh	Falešná pozitivita	Skutečná pozitivita	Úspěšnost rozpoznávání
2000	13.1313 %	40.0000 %	40.0000 %
3000	27.2727 %	60.0000 %	60.0000 %
4500	41.4141 %	80.0000 %	80.0000 %
6000	61.6162 %	100.0000 %	100.0000 %
8000	79.7980 %	100.0000 %	100.0000 %

Tabulka 4.28: Tabulka výsledků testu 2 pro síť 1 při použití algoritmu *globálních procent* s nastavením 9 nejbližších sousedů.

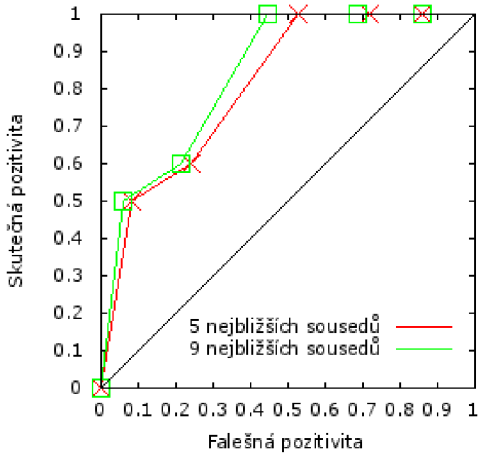
Výsledky pro síť 2



Obrázek 4.12: ROC křivka testu 2 pro algoritmus *nejlepší shody* sítě 2.

Práh	Falešná pozitivita	Skutečná pozitivita	Úspěšnost rozpoznávání
750	13.1313 %	40.0000 %	40.0000 %
1250	29.2929 %	60.0000 %	60.0000 %
2000	52.5253 %	90.0000 %	90.0000 %
3500	77.7778 %	100.0000 %	100.0000 %
5000	88.8889 %	100.0000 %	100.0000 %

Tabulka 4.29: Tabulka výsledků testu 2 pro síť 2 při použití algoritmu *nejlepší shody*.



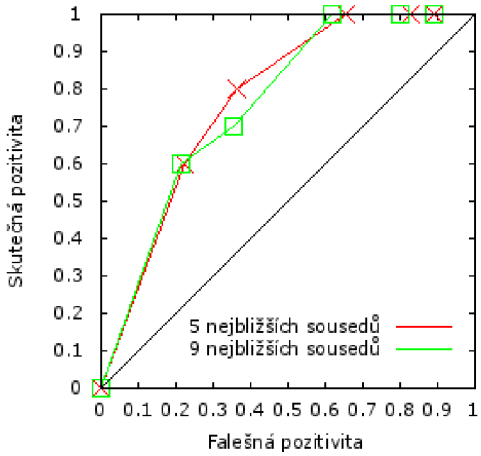
Obrázek 4.13: ROC křivka testu 2 pro algoritmus *lokálních procent* sítě 2.

Práh	Falešná pozitivita	Skutečná pozitivita	Úspěšnost rozpoznávání
2000	8.0808 %	50.0000 %	50.0000 %
3000	24.2424 %	60.0000 %	60.0000 %
4500	52.5253 %	100.0000 %	100.0000 %
6000	71.7172 %	100.0000 %	100.0000 %
8000	85.8586 %	100.0000 %	100.0000 %

Tabulka 4.30: Tabulka výsledků testu 2 pro síť 2 při použití algoritmu *lokálních procent* s nastavením 5 nejbližších sousedů.

Práh	Falešná pozitivita	Skutečná pozitivita	Úspěšnost rozpoznávání
2000	6.0606 %	50.0000 %	50.0000 %
3000	21.2121 %	60.0000 %	60.0000 %
4500	44.4444 %	100.0000 %	100.0000 %
6000	68.6869 %	100.0000 %	100.0000 %
8000	85.8586 %	100.0000 %	100.0000 %

Tabulka 4.31: Tabulka výsledků testu 2 pro síť 2 při použití algoritmu *lokálních procent* s nastavením 9 nejbližších sousedů.



Obrázek 4.14: ROC křivka testu 2 pro algoritmus *globálních procent* sítě 2.

Práh	Falešná pozitivita	Skutečná pozitivita	Úspěšnost rozpoznávání
2000	22.2222 %	60.0000 %	60.0000 %
3000	36.3636 %	80.0000 %	80.0000 %
4500	65.6566 %	100.0000 %	100.0000 %
6000	82.8283 %	100.0000 %	100.0000 %
8000	88.8889 %	100.0000 %	100.0000 %

Tabulka 4.32: Tabulka výsledků testu 2 pro síť 2 při použití algoritmu *globálních procent* s nastavením 5 nejbližších sousedů.

Práh	Falešná pozitivita	Skutečná pozitivita	Úspěšnost rozpoznávání
2000	21.2121 %	60.0000 %	60.0000 %
3000	35.3535 %	70.0000 %	70.0000 %
4500	61.6162 %	100.0000 %	100.0000 %
6000	79.7980 %	100.0000 %	100.0000 %
8000	88.8889 %	100.0000 %	100.0000 %

Tabulka 4.33: Tabulka výsledků testu 2 pro síť 2 při použití algoritmu *globálních procent* s nastavením 9 nejbližších sousedů.

4.3 Zhodnocení výsledků

V prvním testovacím případě, kde bylo rozpoznáváno deset osob, dosahovala síť 1, která má vstupní vrstvu menších rozměrů nežli síť 2, lepších výsledků. U samotného rozpoznávání se úspěšnost této sítě blížila 75 %. Podle výsledků ROC křivek je možné vyčíst, že algoritmus lokálních procent dosahuje v tomto testu lepších výsledků v oblasti klasifikace mezi známou a neznámou osobou, než ostatní dva algoritmy. Počet zjišťovaných sousedů nemá příliš velký vliv na výsledky rozpoznávání. V tabulkách záměn, ve kterých se nachází výsledky rozpoznávání známých osob, je možné vysledovat několik pravidelných záměn. Mezi nejčastější patří záměna p. Bařiny za p. Grézla, či p. Smrže za p. Berana.

V druhém testovacím případě, kde byl rozpoznáván jeden člověk, se výsledky obou sítí příliš neliší. Také v případě použitých algoritmů a počtu zjišťovaných nejbližších sousedů nejsou odlišnosti příliš velké. Pravděpodobně nejlepších nastavení bylo zjištěno při použití sítě 2, algoritmu globálních procent, zjišťování pěti nejbližších sousedů a prahu o hodnotě 3000, kdy úspěšnost detekce dané osoby proběhla s úspěšností 80 % a falešná pozitivita dosahovala hodnoty 36.3636 %.

Kapitola 5

Závěr

Cílem této práce bylo navrhnout a implementovat aplikaci pro rozpoznávání obličejů osob, které projdou před bezpečnostní kamerou. Celá výsledná aplikace se skládá z několika vzájemně propojených modulů, které slouží ke zpracování videa, detekci obličejů, sledování daného obličeje, zarovnání obličeje a následně rozpoznání celé sekvence obličejů. K těmto účelům byly využity konvoluční neuronové sítě a pro klasifikaci byl použit algoritmus k-nejbližších sousedů. Pro rozpoznávání celé detekované sekvence obličejů byly implementovány tři vlastní algoritmy.

Aplikace byla otestována na dvou testovacích případech, ve kterých byly simulovány různé použití bezpečnostních aplikací. V prvním testu byla vyhodnocena úspěšnost rozpoznávání 10 osob, přičemž tato úspěšnost dosahovala 75 %. V druhém testovacím případě byla aplikace otestována pro rozpoznávání jednoho konkrétního člověka v rámci spousty dalších osob.

Z pohledu dalšího vývoje aplikace by mohlo být vhodné upravit algoritmus pro zarovnání obličeje, který je zde řešen poměrně jednoduchou metodou. Pro lepší zarovnání by mohlo být využito například trojrozměrného zarovnání obličeje, které obličej zarovná na základě trojrozměrné reprezentace obličeje. Možným rozšířením by mohlo být také grafické uživatelské rozhraní.

Literatura

[1] ALI, N. H.; HASSAN, G. M.: *Kalman Filter Tracking. International Journal of Computer Applications [online]*, ročník 89, č. 9, 2014, ISSN 0975-8887.

[2] CHALUPNÍK, V.: *Biologické algoritmy (5)* [online]. 2012, [cit. 2016-04-26].
URL http://www.root.cz/clanky/biologicke-algoritmy-5-neuronove-site/

[3] FARAGHER, R.: *Understanding the Basis of the Kalman Filter Via a Simple and Intuitive Derivation* [online]. 2012, [cit. 2016-05-08].
URL https://www.cl.cam.ac.uk/~rmf25/papers/Understanding%20the%20Basis%20of%20the%20Kalman%20Filter.pdf

[4] GALES, M.: *Multi-Layer Perceptrons* [online]. 2015, [cit. 2016-04-27].
URL http://mi.eng.cam.ac.uk/~mjfg/local/4F10/lect6.pdf

[5] GIMENEZ, X.: *Face detection: How to find faces with openCV* [online]. 2010, [cit. 2016-05-08].
URL http://www.xavigimenez.net/blog/2010/02/face-detection-how-to-find-faces-with-opencv/

[6] KAWAGUCHI, K.: *Linear Separability and the XOR Problem* [online]. 2000, [cit. 2016-04-27].
URL http://www.ece.utep.edu/research/webfuzzy/docs/kk-thesis/kk-thesis-html/node19.html

[7] MATERNA, J.: *Středověk umělé inteligence skončil, seznamte se s neuronovými sítěmi, které umí psát básně* [online]. 2015, [cit. 2016-05-08].
URL http://www.mlguru.cz/basnik/

[8] MOLNÁR, K.: *Úvod do problematiky umělých neuronových sítí* [online]. 2000, [cit. 2016-04-27].
URL http://www.elektrorevue.cz/clanky/00013/index.html

[9] NG, A.; NGIAM, J.; FOO, C. Y.; aj.: *Unsupervised Feature Learning and Deep Learning Tutorial* [online]. [cit. 2016-04-27].
URL http://ufldl.stanford.edu/tutorial/

[10] NIELSEN, M. A.: *Neural Networks and Deep Learning* [online]. Determination Press, 2015, [cit. 2016-04-27].
URL http://neuralnetworksanddeeplearning.com/

[11] NOURI, D.: *Using convolutional neural nets to detect facial keypoints tutorial* [online]. 2014, [cit. 2016-05-08].

URL http://danielnouri.org/notes/2014/12/17/
using-convolutional-neural-nets-to-detect-facial-keypoints-tutorial/

[12] TAIGMAN, Y.; YANG, M.; RANZATO, M.; aj.: *DeepFace: Closing the Gap to Human-Level Performance in Face Verification* [online]. 2014, [cit. 2016-05-08].
URL https://research.facebook.com/publications/
deepface-closing-the-gap-to-human-level-performance-in-face-verification/

[13] WAGNER, P.: *Face Recognition with OpenCV* [online]. 2012, [cit. 2016-05-02].
URL http://docs.opencv.org/2.4/modules/contrib/doc/facerec/facerec_
tutorial.html#aligning-face-images

[14] Wikipedia: *Neuron* [online]. 2007, [cit. 2016-04-21].
URL https://cs.wikipedia.org/wiki/Neuron

[15] WOLF, L.; HASSNER, T.; MAOZ, I.: *Face Recognition in Unconstrained Videos with Matched Background Similarity* [online]. 2011, [cit. 2016-05-08].
URL http://www.cs.tau.ac.il/~wolf/ytfaces/WolfHassnerMaoz_CVPR11.pdf

Přílohy

Seznam příloh

Příloha A

Obsah DVD

- Elektronická verze bakalářské práce
- Zdrojové soubory programu
- Neuronové sítě
- Uživatelská příručka

Příloha B

Plakát

Konvoluční neuronové sítě pro bezpečnostní aplikace

CÍL PRÁCE
Cílem této práce bylo vytvořiit aplikaci, která bude schopna rozpoznávat obličeje osob procházejících před bezpečnostní kamerou.

POPIS VÝSLEDNÉ APLIKACE
Výsledkem práce je konzolová aplikace, která dokáže detekovat obličeje lidí procházejících ve video záznamu a tuto osobu sledovat. Následně je každý obličej zarovnán a pomocí jednoho ze tří implementovaných algoritmů rozpoznán.
Pro rozpoznávání obličejů jsou použity natrénované konvolučních neuronové sítě, které vytváří reprezentaci daného obličeje, a algoritmus k-nejbližších sousedů, který vykonává pro samotnou klasifikaci.

VÝSLEDKY
Výsledná aplikace byla testována na testovacích sadách, přičemž úspěšnost rozpoznávání dosahovala téměr 75 %.

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Baklářská práce
Autor: Martin Kišš
Vedoucí: doc. RNDr. Pavel Smrž, Ph.D.
Brno 2016