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

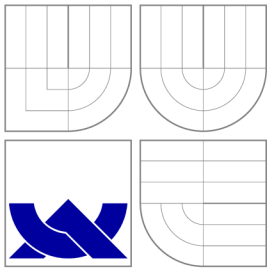
REKURENTNÍ NEURONOVÉ SÍTĚ V POČÍTAČOVÉM VIDĚNÍ

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

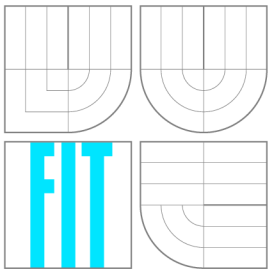
AUTOR PRÁCE
AUTHOR

Bc. JAN KŘEPSKÝ

BRNO 2011



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

REKURENTNÍ NEURONOVÉ SÍTĚ V POČÍTAČOVÉM VIDĚNÍ

RECURRENT NEURAL NETWORKS IN COMPUTER VISION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. JAN KŘEPSKÝ

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. MICHAL ŠPANĚL, Ph.D.

BRNO 2011

Abstrakt

Práce se zabývá využitím rekurentních neuronových sítí v oblasti počítačového vidění. V teoretické části jsou popsány základní poznatky o umělých neuronových sítích se zaměřením na rekurentní architektury. Dále jsou zde prezentovány některé z jejich možných aplikací a nasazení při řešení reálných problémů. Praktická část práce je věnována rozpoznávání obličejů ze sekvence snímků pomocí Elmanovy jednoduché rekurentní sítě. K učení jsou použity algoritmy backpropagation a backpropagation through time.

Abstract

The thesis concentrates on using recurrent neural networks in computer vision. The theoretical part describes the basic knowledge about artificial neural networks with focus on a recurrent architecture. There are presented some of possible applications of the recurrent neural networks which could be used for a solution of real problems. The practical part concentrates on face recognition from an image sequence using the Elman simple recurrent network. For training there are used the backpropagation and backpropagation through time algorithms.

Klíčová slova

neuron, vícevrstvý perceptron, backpropagation, rekurentní neuronové sítě, Elmanova síť, backpropagation through time, rozpoznávání obličejů, eigenfaces, počítačové vidění, umělá inteligence

Keywords

neuron, multilayer perceptron, backpropagation, recurrent neural networks, Elman network, backpropagation through time, face recognition, eigenfaces, computer vision, artificial intelligence

Citace

Jan Křepský: Rekurentní neuronové sítě v počítačovém vidění, diplomová práce, Brno, FIT VUT v Brně, 2011

Rekurentní neuronové sítě v počítačovém vidění

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Michala Španěla, Ph.D. a uvedl jsem všechny literární zdroje, ze kterých jsem čerpal.

.....
Jan Křepský
25. května 2011

Poděkování

Velký dík patří Ing. Michalu Španělovi, Ph.D. za cenné rady a vstřícný přístup při vedení mé diplomové práce. Dále bych na tomto místě chtěl poděkovat Ing. Tomáši Mikolovi a Ing. Karlu Veselému z ÚPGM FIT VUT v Brně za jejich ochotu a pomoc při implementaci některých algoritmů.

Obsah

Seznam použitých zkratek

AAM	Active Appearance Model
ANN	Artificial neural network
ASM	Active Shape Model
BLSTM	Bidirectional long short-term memory
BP	Backpropagation
BPTT	Backpropagation through time
BRNN	Bidirectional recurrent neural network
CCD	Charge–coupled device
CNC	Connectionist temporal classification
CNS	Centrální nervová soustava
CSV	Comma–separated values
EPS	Encapsulated PostScript
HMM	Hidden Markov Model
HNN	Hopfield neural network
ICA	Independent Component Analysis
LGPL	GNU Lesser General Public License
LSTM	Long short–term memory
MLP	Multilayer perceptron
OpenCL	Open Computing Language
OpenCV	Open Source Computer Vision
PCA	Principal Component Analysis
PNG	Portable Network Graphics
RNN	Recurrent neural netwotk

RNNFlib	Recurrent Neural Network Face Recognition Library
SFRM	Single–face reference model
SRN	Simple recurrent network
TFRM	Three–face reference model
XML	Extensible Markup Language

Kapitola 1

Úvod

V dnešní době se stále častěji setkáváme s využitím umělé inteligence v oblasti informačních technologií. Ukázkou jsou expertní a agentní systémy, genetické algoritmy nebo neuronové sítě. Praxe ukazuje, že se tyto metody stávají nenahraditelnou součástí při řešení nejrůznějších úloh, kdy potřebujeme napodobit lidské chování. Umělé neuronové sítě, inspirované svou skutečnou biologickou předlohou, jsou schopny učit se a využívat pak nabytých zkušeností při řešení dalších úkolů. Díky jejich vlastnostem nacházejí uplatnění při rozpoznávání řeči, predikci časových řad, řízení výrobních procesů, ale i v počítačovém vidění. Rekurentní neuronové sítě *(RNN)*, specifické svou cyklickou architekturou, jsou jejich podmnožinou. Existuje celá řada jejich variant, z nichž každá je vhodná pro jiný typ úlohy.

Cílem této práce je prostudovat dostupné materiály na téma aplikace rekurentních sítí v počítačovém vidění. Z těch byla vybrána problematika rozpoznávání obličejů, jíž je věnována praktická část diplomové práce. Funkcemi detekce a identifikace obličejů dnes disponují i levná, všeobecně rozšířená mobilní zařízení a digitální fotoaparáty. Díky tomu se s nimi již každý uživatel informačních technologií může běžně setkat. I přesto je v této oblasti stále spousta nedostatků, na kterých se neustále pracuje. Kromě aplikací určených pro zábavu je správné rozpoznání totiž důležité i v případě biometrických systémů, které mohou zajišťovat naši bezpečnost. Může se jednat např. o přístupové terminály nebo programy pro přihlašování do informačních systémů, kde je jejich vysoká přesnost nezbytná.

Rozpoznávání obličejů provádíme ze sekvence snímků. K extrakci příznaků je použito metody vlastních obličejů – *eigenfaces*. Systém pracuje s Elmanovou rekurentní neuronovou sítí. Pomocí algoritmů backpropagation a backpropagation through time se ji snažíme naučit predikovat následující snímky v sekvenci a této vlastnosti nadále využíváme pro samotnou identifikaci neznámé osoby. Součástí práce je otestování několika různých přístupů ke získání příznakových vektorů, učení a nakonec i rozpoznávání. Navržený systém je také porovnán s běžnou metodou založenou na eigenfaces a hledání nejbližšího souseda.

V druhé kapitole jsou vysvětleny základy teorie umělých neuronových sítí se zaměřením na rekurentní topologie. Je popsán vícevrstvý perceptron, Elmanova a Jordanova síť, long short–term memory a Hopfieldova síť a principy jejich trénování. Třetí kapitola popisuje některá možná využití rekurentních sítí v oblasti počítačového vidění. Jsou zde prezentovány některé ze zajímavých článků věnujících se této problematice. Další, čtvrtá, kapitola se věnuje identifikaci obličejů. Zabývá se popisem technik detekce obličeje v obraze a metodami extrakce příznaků. V páté kapitole představíme navrženou metodu rozpoznávání obličeje ze sekvence snímků pomocí Elmanovy jednoduché sítě. Následuje šestá kapitola, popisující detaily ohledně samotné implementace a použitých knihoven a nástrojů. Předposlední kapitola je věnována experimentům a testování implementovaných algoritmů. Má za úkol seznámit čtenáře s testováním navrženého systému, zhodnotit dosažené výsledky a uvést možnosti budoucího vývoje.

Kapitola 2

Teorie neuronových sítí

V této kapitole se čtenář seznámí se základními pojmy z teorie neuronových sítí. Je zde popsána analogie umělých sítí s těmi biologickými, pojem formální neuron, perceptron a vícevrstvá perceptronová síť, včetně principu jejího učení. Z rekurentních neuronových sítí jsou pak představeny Elmanova, Jordanova a Hopfieldova síť. Část této kapitoly jsem převzal ze své bakalářské práce [14].

2.1 Motivace

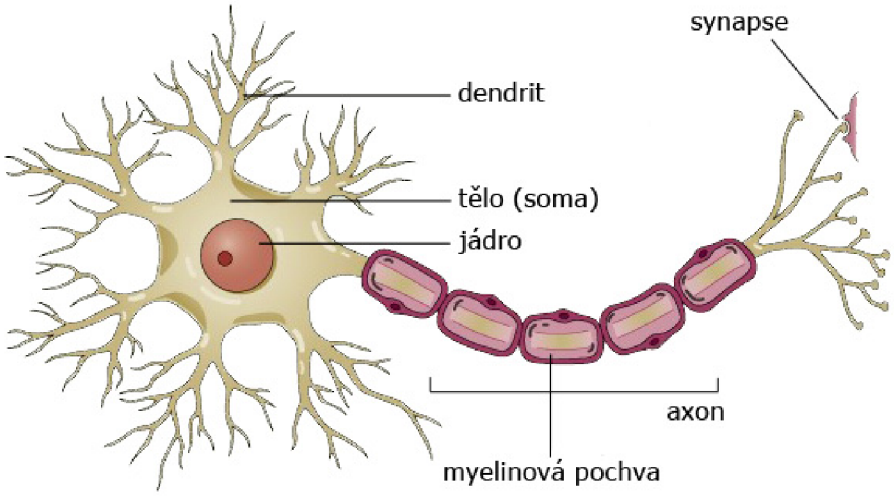
Často se v informačních technologiích setkáváme s problémy jen těžko řešitelnými klasickými výpočetními metodami. Pro rozpoznávání písma, řeči, detekci nebo segmentaci objektů v obraze, predikci, řízení a jiných úloh proto musíme sáhnout po některé z metod umělé inteligence. Jednou z nich jsou i umělé neuronové sítě *(Artificial Neural Networks – ANN)*. Vznik tohoto oboru se datuje do roku 1943, kdy pánové W. McCulloch a W. Pitts přišli s prvním modelem formálního neuronu.

Umělá neuronová síť je matematickým modelem inspirovaným nervovou soustavou člověka. Díky jejím schopnostem učení se a využívání získaných zkušeností k řešení nových úkolů napodobujících lidskou inteligenci se stává stále užívanějším nástrojem v nejrůznějších aplikacích informatiky.

2.2 Biologický základ

Nervová soustava slouží ke komunikaci člověka s okolním světem a k přenosu, zpracování a uchování získaných informací. V lidském těle je několik druhů receptorů (čidel), které přijímají mechanické, termické, světelné a chemické informace z vnějšího i vnitřního prostředí organismu. Při podráždění receptoru vzniká elektrický vzruch, který je veden do různých úrovní CNS a po zpracování je výsledný vzruch veden do výkonných orgánů (efektorů).

Základní jednotkou nervové soustavy je *neuron* (obr. 2.1), jenž se skládá z několika desítek dendritů, kterými je vzruch přiváděn do těla neuronu *(soma)*. Vzruch dále putuje jedinou výstupní drahou – *axonem*. Ten je obalen myelinovou pochvou a může dosahovat délky přes jeden metr. Na svém konci se větví a je zakončen tzv. *synapsemi*, které tvoří spoje mezi jednotlivými neurony. Synapse lze rozdělit na *excitační*, které umožňují šíření vzruchu dál, a *inhibiční*, které jej utlumují. Během lidského života tyto spoje vznikají a přetvářejí se, což je principem učení. Jeden neuron je propojen s cca 10 až 100 000 dalšími neurony a tím je vlastně tvořená celá neuronová síť. Vzhledem k tomu, že hustota neuronů



Obrázek 2.1: Skutečný neuron (zdroj: http://www.utexas.edu/courses/bio365r/Images/neuron.JPG)

v mozku je asi 7–8 $\cdot 10^4 mm^{-3}$, jedná se o velmi složitý systém, který je doposud předmětem zkoumání [22, 26].

2.3 Formální neuron

Pokud chceme matematicky popsat neuronovou síť, musíme nejdříve definovat její základní prvek – formální neuron. Ten do jisté míry napodobuje skutečný neuron a jeho struktura je znázorněna na obrázku 2.2. Červeně jsou znázorněny části odpovídající svému biologickému vzoru. Vstupní vektor $(x_1, \ldots, x_n)$ zde značí dendrity, přičemž každý vstup je ohodnocen *synaptickou váhou* $w_1, \ldots, w_n$ udávající míru, jakou se daný vstup podílí na výstupu neuronu. Tyto váhy modelují propustnost synapsí. *Vnitřní potenciál* ξ je potom spočítán pomocí nějaké agregační funkce, nejčastěji vážené sumy:

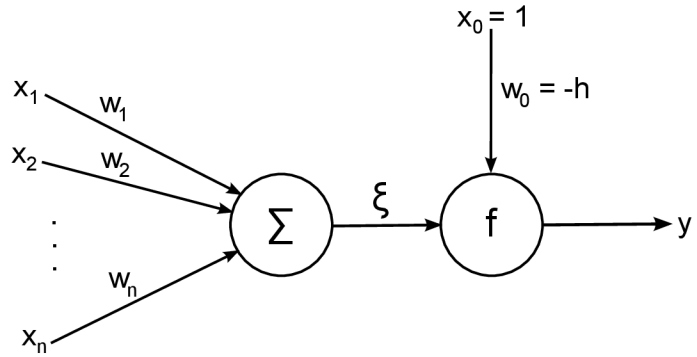
$$\xi = \sum_{i=1}^{n} w_i x_i. \tag{2.1}$$

Výstup neuronu je potom dán jako $y = f(\xi)$, kde f je tzv. *aktivační (přenosová)* funkce. U formálního neuronu se používá *ostrá nelinearita* (obr. 2.3a) daná jako:

$$f(\xi) = \begin{cases} 1 & \text{pro } \xi \geq h \\ 0 & \text{pro } \xi < h \end{cases} \tag{2.2}$$

Tato funkce je inspirována vlastnostmi biologického neuronu. Pokud vnitřní potenciál dosáhne hodnoty prahu, je vzruch předán k dalšímu neuronu. Někdy mezi vstupy počítáme i vstup x_0 ohodnocen záporným prahem *(bias)*. Výstup neuronu je potom popsán následujícím vztahem:

$$f(\xi) = \begin{cases} 1 & \text{pro } \xi \geq 0 \\ 0 & \text{pro } \xi < 0 \end{cases}, \quad \text{kde } \xi = \sum_{i=0}^{n} w_i x_i. \tag{2.3}$$

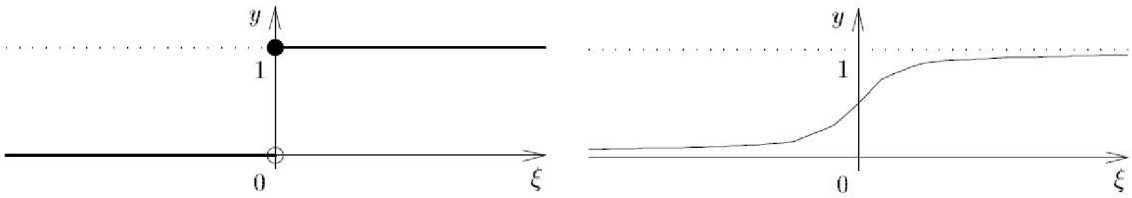


Obrázek 2.2: Formální neuron, $x_1, \dots x_n$ vstupy, $w_1, \dots, w_n$ synaptické váhy, x_0 formální vstup, w_0 bias, h práh, ξ vnitřní potenciál, f aktivační funkce, y výstup

Existují ale i další typy přenosových funkcí. Příkladem je třeba *signum*, nebo *sigmoida* (obr. 2.3b) popsaná rovnicí

$$f(\xi) = \frac{1}{1 + e^{-\lambda\xi}}, \tag{2.4}$$

kde λ je *parametr strmosti* (gain). Používá se i její varianta, tzv. *symetrická sigmoida*, což není nic jiného než hyperbolický tangens λx. Výběr aktivační funkce se provádí na základě typu řešené úlohy, ale i v závislosti na poloze neuronu v síti.



(a) ostrá nelinearita (b) sigmoida

Obrázek 2.3: Příklady aktivačních funkcí. Převzato z [22].

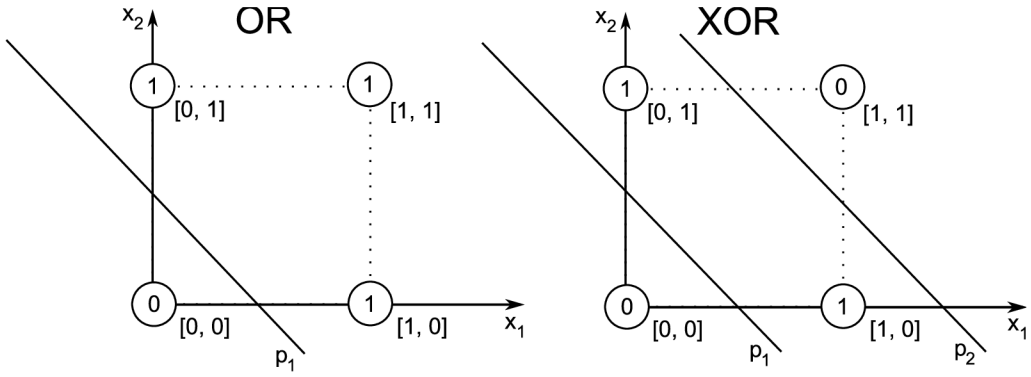
Úkolem neuronu je tedy transformovat vektor vstupních hodnot na jednu výstupní hodnotu na základě určené přenosové funkce. Výstup neuronu se však může dále rozvětvovat a napojovat na vstupy dalších neuronů, čímž je tvořena neuronová síť. Výše popsaný model umělého neuronu s váženým součtem vstupů jakožto agregační funkcí pro výpočet vnitřního potenciálu a ostrou nelinearitou jakožto aktivační funkcí se označuje jako **perceptron** (F. Rosenblatt, 1957) [6, 22].

Funkce perceptronu se dá dobře znázornit graficky. Vstupy neuronu si představme jako souřadnice bodu v n-rozměrném prostoru, kde n udává počet vstupů. Rovnice nadroviny takového prostoru má tvar

$$\sum_{i=1}^{n} w_i x_i + w_0 = 0. \tag{2.5}$$

Pro dva vstupy tedy máme rovnici roviny, pro tři vstupy rovnici prostoru atd. Nadrovina potom dělí tento vstupní prostor na dva podprostory. Pro názornost předpokládejme 2D

prostor, kde je nadrovinou přímka daná obecnou rovnicí $w_1x_1 + w_2x_2 + w_0 = 0$ se směrnicí $-\frac{w_1}{w_2}$ a posunem vůči ose x_2 vyjádřeným jako $-\frac{w_0}{w_2}$. Tato hraniční přímka nám rozdělí vstupní rovinu na dvě poloroviny a klasifikuje tak vstupy do dvou tříd. Takto lze vyřešit například operace AND nebo OR, jelikož jejich body lze rozdělit do dvou disjunktních množin pomocí jedné přímky – jsou *lineárně separabilní*. Naopak operaci XOR takto vyřešit nelze, jelikož k oddělení bodů $[0,0],[1,1]$ a $[1,0],[0,1]$ nám už jedna přímka nestačí (obr. 2.4). Pro tuto úlohu už bychom museli využít vícevrstvou síť. Říkáme, že problém je *lineárně neseparabilní* [22, 16, 26].



Obrázek 2.4: Lineárně separabilní a neseparabilní problémy.

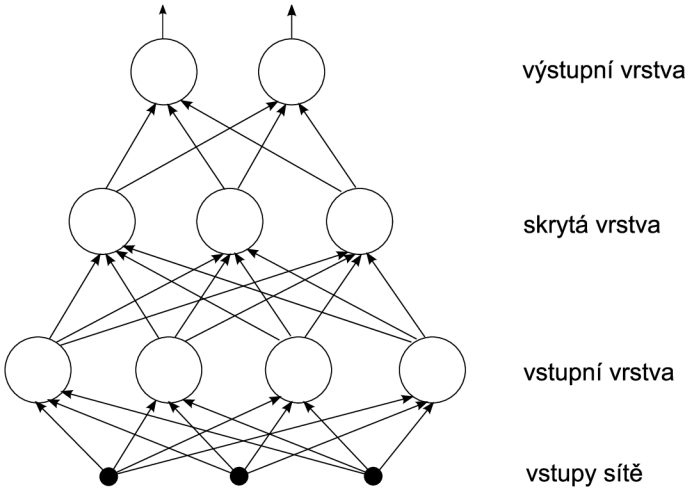
2.4 Vícevrstvá perceptronová síť

Umělá neuronová síť se skládá z několika vzájemně propojených neuronů. Jejich počet a vzájemné propojení určuje *topologii* nebo také *architekturu sítě*. Rozlišujeme dva základní typy: *acyklickou (dopřednou)* a *cyklickou (rekurentní)*. V případě dopředné sítě jsou neurony uspořádány do vrstev, kde výstupy neuronů z jedné vrstvy jsou zároveň vstupy neuronů vrstvy následující. Vrstvy lze rozdělit na vstupní, skryté a výstupní, přičemž signál se síří pouze jedním směrem. U rekurentní sítě jsou některé z neuronů zapojeny do kruhu, šíření signálu tedy může být jak dopředné, tak zpětné.

Vícevrstvá síť perceptronů (vícevrstvý perceptron) je složena z několika vrstev perceptronů, přičemž výstupy jednoho neuronu jsou přivedeny na vstupy všech neuronů následující vrstvy. Stejně tak každý vstup je přiveden na vstupy všech neuronů v první (vstupní) vrstvě. Jedná se tedy o plně propojenou dopřednou síť. Obrázek 2.5 popisuje architekturu třívrstvé perceptronové sítě se třemi vstupy a dvěma výstupy.

Využijme znalostí o formálním neuronu z předchozí podkapitoly. První vrstva nám pomocí nadrovin rozděluje vstupní prostor na n poloprostorů, kde n udává počet neuronů v této vrstvě. Druhá vrstva kombinuje tyto poloprostory. Je tedy možné vytvořit otevřené nebo uzavřené konvexní oblasti. Další vrstva opět kombinuje výstupy předchozí vrstvy, čímž můžeme spojit konvexní útvary do nekonvexních. Takto můžeme pokračovat až k výstupní vrstvě a řešit tak složitější a složitější problémy.

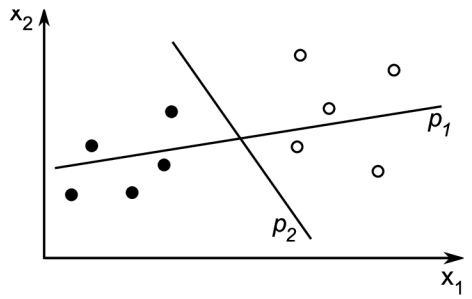
Pro to, aby nám síť fungovala správně, je nutné náležitě nastavit váhy sítě. To se děje během procesu **učení**. Slouží k tomu tzv. *trénovací množina*, což je množina vzorů, kde jsou známy vstupy sítě a k nim požadované výstupy. Když řešíme reálný problém, většinou není možné postihnout všechny možné případy vstupních hodnot. Proto z nich vybíráme pouze



Obrázek 2.5: Příklad architektury třívrstvé perceptronové sítě se třemi vstupy a dvěma výstupy.

reprezentativní vzorek popisující charakteristické rysy vstupních dat a předpokládáme, že ostatní hodnoty budou těmto vybraným hodnotám podobné. Přeneseno do euklidovského prostoru, body představující vstupní data budou blízko sebe (viz plná a prázdná kolečka na obrázku 2.6). Trošku nadneseně si to lze představit tak, že nikdo nemůže mít v mozku uloženy všechny možné podoby písmene „A“. Nicméně, když někde toto písmeno uvidíme napsané, poznáme jej, jelikož bude podobné těm, které již známe. Tedy máme určitou představu o tom, jak písmeno „A“ asi vypadá. V první třídě jsme číst neuměli – hodnoty synaptických vah neuronů v našem mozku starajících se o rozpoznávání písmen byly nastaveny náhodně. Postupně jsme se však učili číst a setkávali se s různými typy písma, čímž se tyto váhy upravovali.

Na začátku jsou tedy váhy sítě zvoleny náhodně a během jejího učení jsou postupně modifikovány, aby dané vzory dokázala síť správně klasifikovat. Opět si to ukažme na jednoduchém problému ve dvourozměrném prostoru znázorněném na obrázku 2.6. Cílem je natrénovat síť tak, aby od sebe dokázala odlišit dvě skupiny vzorů – plná a prázdná kolečka. Váhy w_1 a w_2 jsou na začátku zvoleny náhodně, čili i poloha hraniční přímky p_1 bude náhodná. Jak se během učení tyto váhy mění, mění se s nimi i směrnice a posunutí přímky, dokud neodděluje požadované vzory do dvou disjunktních množin. Její konečná poloha je na obrázku označena jako p_2.



Obrázek 2.6: Ilustrace adaptace vah během učení.

2.4.1 Algoritmus backpropagation

Pro učení vícevrstvé perceptronové sítě se používá **algortimus zpětného šíření chyby** *(backpropagation – BP)*. Tento proces má dvě fáze – *aktivní* (vybavovací) a *adaptivní*. Ve vybavovací fázi se na vstupy sítě přiloží vzor a postupně se počítají výstupy jednotlivých neuronů od vstupní po výstupní vrstvu na základě jejich vnitřního potenciálu a dané aktivační funkce. Ve fázi adaptivní se potom zpětným šířením chyby do předchozích vrstev modifikují váhy mezi neurony. [24]

Informace o tomto algoritmu jsem čerpal převážně z [37], kde je mimo jiné uveden i přehledný pseudokód pro jeho případnou implementaci. Dalším vhodným zdrojem jsou WWW stránky [1]. Zde je naopak k dispozici velmi zdařilá grafická reprezentace celého průběhu aktivní i adaptivní fáze učení pro jeho snadnější pochopení.

Pro další výklad si zavedeme následující značení:

n_l	počet neuronů vrstvy l ($n_0 = n$, $n_L = m$)
$\vec{d}$	m-rozměrný požadovaný výstupní vektor sítě
$\vec{o}$	m-rozměrný skutečný výstupní vektor sítě
${}^l\vec{x}$	$(n_{l-1}+1)$-rozměrný vstupní vektor neuronů vrstvy l
${}^l\vec{y}$	n_l-rozměrný výstupní vektor neuronů vrstvy l
${}^lw_{ij}$	váha synapse i-tého vstupu j-tého neuronu vrstvy l
${}^lw_j$	chyba j-tého neuronu vrstvy l

Úprava hodnot vah sítě probíhá dle vztahu

$$ {}^lw_{ji}(t+1) = {}^lw_{ji}(t) + \Delta^l w_{ji}. \tag{2.6} $$

Změny vah vypočítáme ze vztahu

$$ \Delta^l w_{ji} = -\mu \frac{\partial E}{\partial^l w_{ji}} = -\mu \frac{\partial E}{\partial^l \xi_j} \frac{\partial^l \xi_j}{\partial^l w_{ji}} = -\mu \frac{\partial E}{\partial^l y_j} \frac{\partial^l y_j}{\partial^l \xi_j} \frac{\partial^l \xi_j}{\partial^j w_{ji}} = \mu^l \delta_j \frac{\partial^l \xi_j}{\partial^l w_{ji}} \tag{2.7} $$

kde μ je malá kladná konstanta ($0 \leq \mu \leq 1$) označovaná jako *parametr učení*, anglicky *learning rate* a

$$ {}^l\delta_j = -\frac{\partial E}{\partial^l \xi_j} = -\frac{\partial E}{\partial^l y_j} \frac{\partial^l y_j}{\partial^l \xi_j}. \tag{2.8} $$

V rovnici 2.1 jsme si definovali vzorec pro výpočet vnitřního potenciálu neuronu. Přepišme jej dle naší konvence:

$$ {}^l\xi_j = \sum_{i=0}^{n_{l-1}} w_{ji} x_i. \tag{2.9} $$

Potom můžeme psát

$$ \Delta^l w_{ji} = \mu^l \delta_j {}^l x_i. \tag{2.10} $$

Pro popsanou aktivační funkci - sigmoidu (rovnice 2.4) dále platí

$$ \frac{\partial^l y_i}{\partial^l \xi_j} = \lambda^l y_j (1 - {}^l y_j). \tag{2.11} $$

Chybu sítě pro aktuální trénovací vzorek spočítáme dle rovnice

$$E = \frac{1}{2}\sum_{j=1}^{m}(d_j - o_j)^2 = \frac{1}{2}\sum_{j=1}^{n_L}(d_j - {}^{l}y_j)^2. \tag{2.12}$$

Pro výstupní vrstvu tento vztah stačí zderivovat. Pak dostáváme

$$\frac{\partial E}{\partial {}^{L}y_j} = -(d_j - {}^{L}y_j). \tag{2.13}$$

Dosazením vztahu 2.13 do rovnice 2.8 dostáváme přímo vztah pro chyby neuronů výstupní vrstvy:

$${}^{L}\delta_j = -\frac{\partial E}{\partial {}^{L}y_j}\frac{\partial {}^{L}y_j}{\partial {}^{L}\xi_j} = (d_j - {}^{L}y_j)\frac{\partial {}^{L}y_j}{\partial {}^{L}\xi_j} \tag{2.14}$$

a dále dosazením vztahu 2.11 pak konkrétně pro sigmoidální aktivační funkci:

$${}^{L}\delta_j = (d_j - {}^{L}y_j)\lambda {}^{L}y_j(1 - {}^{L}y_j). \tag{2.15}$$

Můžeme odvodit rovnici pro výpočet chyby všech neuronů v ostatních vrstách:

$${}^{l-1}\delta_j = \sum_{k=1}^{n_l}({}^{l}\delta_k {}^{l}w_{kj})\frac{\partial {}^{l-1}y_j}{\partial {}^{l-1}\xi_j} \tag{2.16}$$

a opět konkrétně pro sigmodu:

$${}^{l-1}\delta_j = \sum_{k=1}^{n_l}({}^{l}\delta_k {}^{l}w_{kj})\lambda {}^{l-1}y_j(1 - {}^{l-1}y_j). \tag{2.17}$$

Nesmíme zapomenout dodat, že učení většinout ukončujeme na základě průměrné chyby sítě pro celou trénovací množinu. Pro její výpočet nejčastěji používáme kvadratický průměr, známý také jako *Root Mean Square*. Pokud E_i označíme jako chybu sítě na jednom trénovacím vzorku (rovnice 2.12) a P jako počet vzorů, potom celkovou chybu sítě E_{rms} spočítáme dle rovnice:

$$E_{rms} = \sqrt{\frac{\sum_{i=1}^{P} E_i^2}{P}}. \tag{2.18}$$

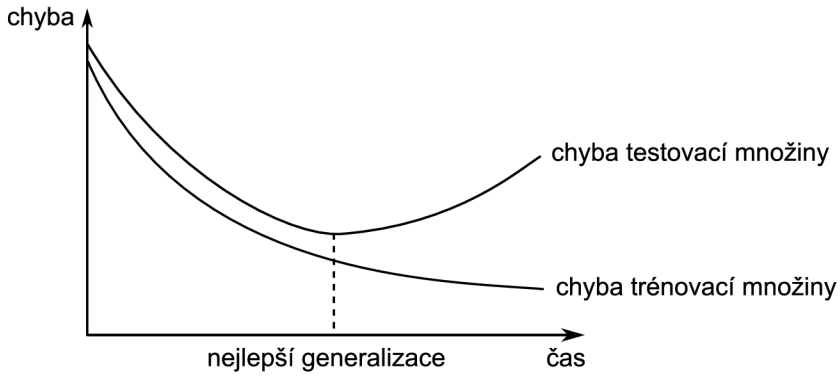
2.4.2 Metody zlepšující chování sítě

V této podkapitole si uvedeme některé metody, které se používají pro zlepšení průběhu učení a chování sítě.

Počet neuronů sítě

Vhodný počet perceptronů ve skrytých vrstvách volíme vždy pro konkrétní úlohu, a to na základě heuristik. Například síť se dvěma skrytými vrstvami by měla v první skryté vrstvě mít o něco více neuronů než ve vrstvě vstupní a ve druhé skryté vrstvě aritmetický průměr mezi počtem neuronů v předchozí a výstupní vrstvě. Pokud bude neuronů málo, síť nedokáže správně rozpoznat závislosti mezi jednotlivými vzory. Naopak pokud bude neuronů hodně, roste doba učení, ale hlavně klesá schopnost tzv. *generalizace*. Síť bude dobře rozpoznávat

všechny vzory z trénovací množiny, ale nedokáže klasifikovat ostatní vzory v trénovacích datech neobsažené. Tento jev nazýváme *přetrénování* (overfitting). Proto parametry sítě samozřejmě upravujeme dle potřeby, hlavně podle výsledků testování.



Obrázek 2.7: Typický průběh chyby během učení

Crossvalidace

Přetrénování lze předejít použitím tzv. *crossvalidační množiny*. Často tedy vzorová vstupní data, která máme k dispozici, rozdělíme do dvou množin. Učení nadále probíhá na trénovacích datech, ale chybu sítě potom počítáme na datech crossvalidačních. Této technice říkáme křížové ověřování správnosti neboli *crossvalidation*. Obrázek 2.7 ukazuje typický průběh chyby při trénování. Nejlepší generalizace tedy síť dosahuje v momentě, kdy je chyba na testovacích datech nejmenší.

Momentum

Síť typu vícevrstvý perceptron využívá během trénování gradientní metodu. Stává se, že síť uvázne v lokálním minimu. Je to způsobeno tím, že nelze pokračovat ve směru minimalizace chybové funkce. Jedním z nejpoužívanějších způsobů, jak tomu zabránit, je použít tzv. *momentum*.

Již známe vztah 2.10 pro výpočet změny váhy mezi jednotlivými neurony. Změna váhy se ale nemusí řídit pouze aktuálním gradientem, ale můžeme sem zahrnout i předchozí změnu váhy - hybnost (momentum). Tu si však musíme během procesu učení někam ukládat. Tedy:

$$\Delta^l w_{ji}(t) = \mu^l \delta_j{}^l x_i + \alpha \Delta^l w_{ji}(t-1) \tag{2.19}$$

Hodnotu koeficientu α volíme z intervalu $\langle 0, 1 \rangle$.[24]

Dynamický parametr učení

Velikost parametru učení μ významně ovlivňuje chování sítě, zejména rychlost konvergence k řešení. Při malé hodnotě μ klesá chyba pomalu. Pokud naopak zvolíme parametr učení příliš velký, je sice učení rychlejší, ale může dojít k divergenci. Proto se doporučuje parametr učení dynamicky měnit během trénování. Jedním ze způsobů je použít následujícího vztahu:

$$\mu = \frac{\mu_0}{1 + \frac{k}{K}}, \tag{2.20}$$

kde obvykle $0.1 \leq \mu_0 \leq 0.9$, k je epocha (krok učení) a K značí větší kladnou konstantu. Dalším ze způsobů je měnit parametr učení dle zjištěné chyby na crossvalidační množině. Např. pokud zjistíme, že tato chyba v aktuálním kroku stoupla oproti předchozímu, vynásobíme parametr učení nějakou zvolenou konstantou v rozsahu $(0, 1)$. [37, 24, 10]

Velikost trénovací množiny

Důležitou roli hraje také velikost trénovací množiny. Zdroj [25] uvádí, že minimální počet vzorů P by měl být

$$P \geq \frac{|W|}{1 - a} \log \frac{n}{1 - a}, \tag{2.21}$$

kde $|W|$ je počet vah v síti, a požadovaná přesnost z intervalu $\langle 0, 1 \rangle$ a n počet neuronů.

2.5 Jordanova a Elmanova síť

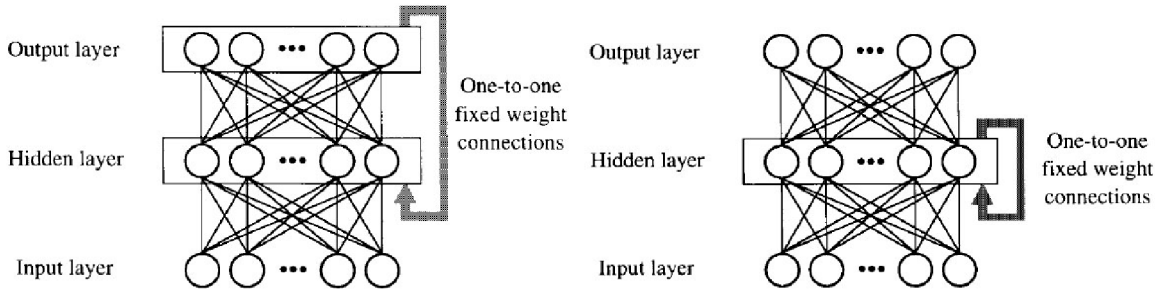
Dopředné sítě nacházejí uplatnění tam, kde výstup sítě závisí pouze na aktuálním vstupu. Existují ale i problémy, kde je ke správnému spočtení výstupu nutné znát i historii, tedy předchozí stavy sítě. To lze vyřešit přidáním zpětných synapsí do architektury sítě tak, jak tomu je např. u Jordanovy a Elmanovy sítě. Obě bývají často souhrně nazývány jednoduché rekurentní sítě *(Simple Recurrent Networks – SRN)*. Jejich autory jsou S. Jordan (1986) a J. Elman (1990). Obě disponují čtyřmi vrstvami – vstupní, skrytou, výstupní a jednou kontextovou *(context layer)*. Neurony kontextové vrstvy bývají někdy označovány jako *state units*, jelikož svým způsobem uchovávají vnitřní stav sítě.

V případě Jordanovy sítě jsou v každém kroku do kontextové vrstvy zkopírovány aktivace neuronů vrstvy výstupní. U Elmanovy sítě se kopíruje výstup vrstvy skryté – toto je zásadní rozdíl mezi těmito druhy SRN. Váhy mezi výstupní a skrytou, resp. skrytou a kontextovou vrstvou, jsou pevné a mají hodnotu 1. Kontextová vrstva obsahuje právě tolik neuronů, kolik neuronů má vrstva, jejíž výstup je kopírován. Každý z nich je potom propojen se všemi neurony skryté vrstvy. Díky tomu může stav neuronů skryté vrstvy v čase t ovlivnit jejich stav v čase $t + 1$. Ilustrace topologie sítí je na obrázku 2.8. [17, 20]

Shrnutí fungování Elmanovy sítě:

1. aktivace (vnitřní stavy) neuronů kontextové vrstvy nastav na 0, $t = 1$
2. z přiloženého vstupního vektoru a obsahu kontextové vrstvy vypočti stavy neuronů skryté vrstvy
3. vypočti výstup sítě
4. zkopíruj obsah skryté vrstvy do kontextové
5. $t \leftarrow t + 1$, jdi na bod 2

K učení jednoduchých rekurentních sítí lze použít algoritmus backpropagation, popsaný v kapitole 2.4.1. Pokud si ale budeme uchovávat stav skryté vrstvy pro více předchozích časových kroků, můžeme použít algoritmus zpětného šíření chyby v čase *(backpropagation through time)*.



Obrázek 2.8: Jordanova (vlevo) a Elmanova (vpravo) síť. Převzato z [19].

2.5.1 Algoritmus backpropagation through time

Algoritmus zpětného šíření chyby v čase *(backpropagation through time – BPTT)* je jednoduchým rozšířením algoritmu backpropagation (kap. 2.4.1) pro rekurentní neuronové sítě [2, 15]. Výpočet chyb při zpětném šíření je ovlivněn i chybami z předchozích časových kroků. Jeho základním principem je rozvinutí (angl. *unfolding*) rekurentní sítě na vícevrstvou dopřednou síť. Zde již můžeme s jistými úpravami použít klasického BP. Při tomto rozvinutí každá z vrstev nově vzniklé dopředné sítě reprezentuje jednu rekurenci v čase. Celkový počet vrstev udává počet časových kroků do minulosti (označme τ). Efekt rozvinutí je znázorněn na obrázku 2.9.

Z rovnice 2.16 dostáváme vztah pro výpočet chyby v předchozích vrstvách (čase):

$$ {}^{l}\delta_j(t-1) = \sum_{k=1}^{n_l} ({}^{l}\delta_k(t)\, {}^{l}w_{kj}) \frac{\partial^{l} y_j(t-1)}{\partial^{l} \xi_j(t-1)} \tag{2.22} $$

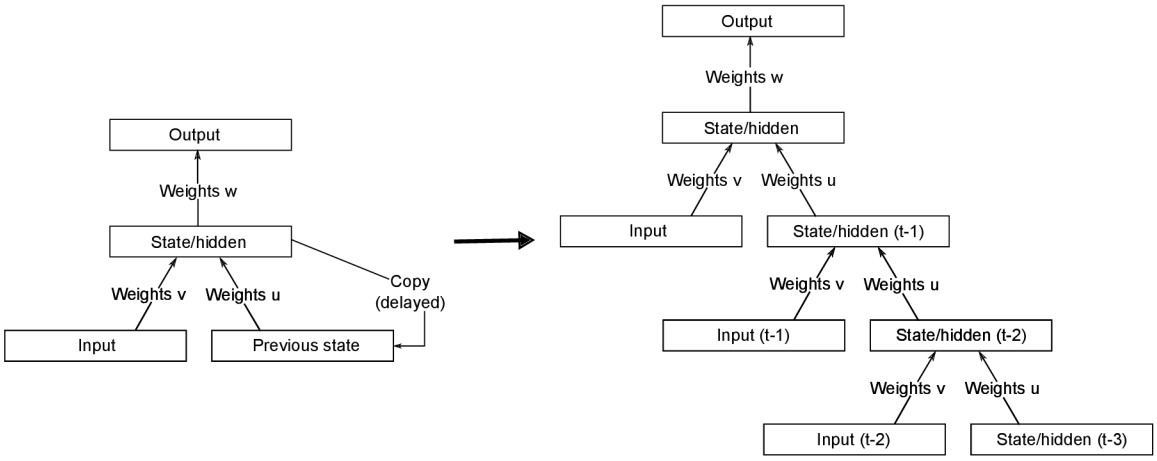
Je důležité si uvědomit vztahy mezi jednotlivými váhami u takto rozvinuté sítě. U vah w mezi skrytou a výstupní vrstvou není žádná rekurence a tak je upravujeme stejně jako u BP. Tedy dle vztahu 2.10:

$$ \Delta^{L} w_{ji} = \mu^{L} \delta_j(t)\, {}^{L}x_i(t) \tag{2.23} $$

Váhy mezi vstupní a skrytou vrstvou (v) a váhy mezi skrytými vrstvami (u) jsou potom společné pro všechny vrstvy. Každá z nich nám dává příspěvek do celkové změny váhy. Adaptaci vah u a v provádíme až naráz, po spočtení těchto celkových příspěvků ze všech předchozích časových kroků:

$$ \Delta^{l} v_{ji} = \mu \sum_{t=1}^{\tau} {}^{l}\delta_j(t) x_i(t), \quad \Delta^{l} u_{ji} = \mu \sum_{t=1}^{\tau} {}^{l}\delta_j(t) y_i(t-1) \tag{2.24} $$

Z uvedeného je zřejmé, že algoritmus BPTT bude mít daleko větší nároky na paměť. Čím větší parametr τ zvolíme, tím více paměti budeme při učení potřebovat. Standardní RNN, využívající k trénování gradientní metody (jako například Elmanova síť), nejsou schopny využívat informací o kontextu po delší dobu. Čím více do minulosti totiž jdeme, tím je propagovaná chyba menší a menší, až vymizí úplně. Tak ztrácíme informaci o kontextu. Tento problém je v literatuře [2, 9] označován jako *vanishing gradient effect*. Efekt je ilustrován na obrázku 2.11a. Právě z tohoto důvodu v praxi parametr τ často omezujeme na nějakou rozumnou hodnotu.



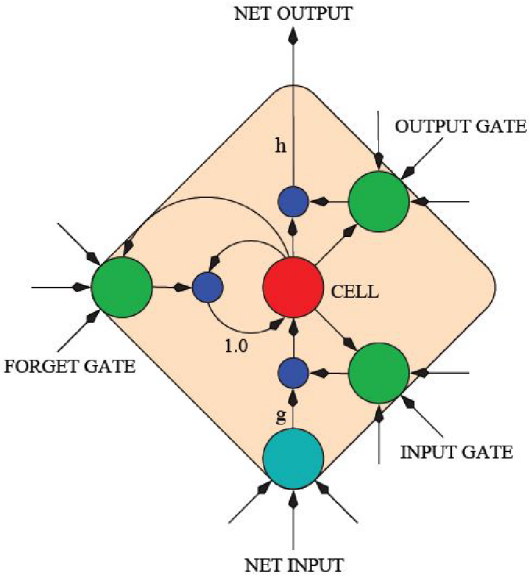
Obrázek 2.9: Efekt rozvinutí Elmanovy rekurentní sítě na dopřednou síť ($\tau = 3$). Převzato z [2].

Jednou z velkých výhod trénování pomocí BPTT je ta, že můžeme síť učit po sekvencích. Pokud nám při trénování záleží na pořadí vzorů, např. při predikci, můžeme postupně předkládat jednotlivé vzory ve zvoleném pořadí. Po vyčerpání všech vzorů aktuální sekvence vymažeme obsahy kontextových vrstev a pokračujeme další sekvencí. Tak nedojde k ovlivnění adaptace vah daty z předchozích sekvencí.

2.6 Long short–term memory

V předchozím textu jsme zmínili tzv. vanishing gradient effect. Jedním z typů rekurentních neuronových sítí, navržených ve snaze tomuto problému zabránit, je právě *long short–term memory (LSTM)* [9]. Poprvé byl tento druh sítě publikován v roce 1997. Skrytá vsrtva LSTM se skládá z paměťových bloků. Každý takový blok obsahuje několik rekurentně propojených buněk sloužících k uchování kontextu mezi jednotlivými kroky sítě. Dále jsou zde tři druhy bran – *input gate*, *forget gate* a *output gate*. Ty mají za úkol řídit to, kdy je buňkám umožněno ukládat nové informace, popř. kdy uložené informace dávat na svůj výstup. Díky nim je možné uchovávat si informace o kontextu po delší dobu. Např. dokud je vstupní brána zavřená (tzn. její aktivace je blízko 0), aktivace buňky není ovlivněna novými vstupy sítě a zůstává nezměněna. Stějně tak aktivace buňky je dostupná pouze v okamžiku, kdy je otevřená výstupní brána. Rekurentní spoj vnitřní buňky je řízen třetí „zapomínací“ (forget) bránou. Čili zavřením této brány je zabráněno uchovávání kontextu. Dokud je vstupní brána zavřená a „zapomínací“ brána zavřená, nedochází ke změně obsahu buňky. Znázornění jedné buňky LSTM je na obrázku 2.10. Ilustrace řízení takové buňky pomocí bran potom naleznete na obrázku 2.11b.

K učení LSTM lze použít algoritmus BPTT, popsaný v kapitole 2.5.1. Využití tento typ sítě nachází především v predikci časových řad, robotice, rozpoznávání řeči, kompozici hudby a učení rytmů počítačem, učení gramatik, ale v neposlední řadě také rozpoznávání ručně psaného písma [35].



Obrázek 2.10: Paměťový blok LSTM. Buňka (červěně) má rekurentní spoj s pevnou váhou 1,0. Tři brány (zeleně) sbírají informace ze zbytku sítě a kontrolují stav buňky pomocí multiplikativních jednotek (modře). Funkce g a h jsou zvolené aktivační funkce. Vnitřní propojení mezi buňkou a branami se nazývají *peephole weights*. Převzato z [9].

2.7 Hopfieldova síť

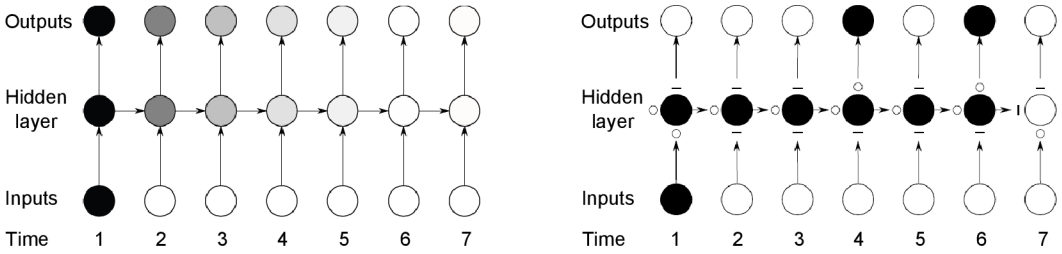
Vynález Hopfieldovy sítě *(Hopfield neural network – HNN)* se přikládá pánům McCullohovi a Pittsovi a datuje se do roku 1943. Nicméně známou se stala až díky Hopfieldovi na počátku 80. let dvacátého století. Hopfield při její analýze využil analogie s fyzikální teorií magnetických materiálů a rozpracoval využití energetické funkce, která je nutná pro správné fungování sítě. Z ní jsou odvozena pravidla pro učení a vybavování. V současnosti existuje mnoho variant tohoto modelu, které se snaží zlepšit jeho chování. Jedná se o druh autoasociativní paměti. [24, 22]

2.7.1 Topologie sítě

Hopfieldova síť je druh cyklické plně rekurentní sítě s n neurony, kde každý neuron je zároveň vstupní i výstupní. Neuronů je právě tolik, kolik je vstupů. Výstupy jednoho neuronu jsou zároveň vstupy všech ostatních. Váhu mezi i-tým a j-tým neuronem označme jako w_{ij}. Jelikož váhy jsou stejné v obou směrech (tj. $w_{ij} = w_{ji}$) a na vstupy neuronů nejsou vedeny jejich vlastní výstupy (tedy $w_{ii} = 0$), vzniká tak symetrická matice vah s nulami na diagonále:

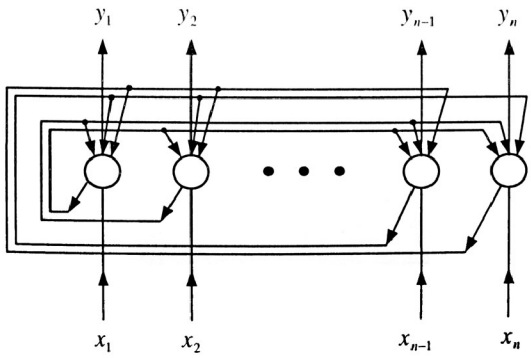
$$\mathbf{W} = \begin{bmatrix} 0 & w_{12} & w_{13} & \dots & w_{1n} \\ w_{12} & 0 & w_{23} & \dots & w_{2n} \\ w_{13} & w_{23} & 0 & \dots & w_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ w_{1n} & w_{2n} & w_{3n} & \dots & 0 \end{bmatrix} \tag{2.25}$$

Ilustrace této topologie je na obrázku 2.12. Každý neuron má jeden externí vstup – jsou



(a) (b)

Obrázek 2.11: (a) vanishing gradient effect, (b) Ukázka zachování informace o kontextu u LSTM. Diagram zachycuje stav sítě s jedním paměťovým blokem v čase. Aktivace input, output a forget gate jsou zobrazeny pod, nad, resp. vlevo od paměťového bloku (○ zcela otevřená, – zcela zavřená). Převzato z [9].



Obrázek 2.12: Topologie Hopfieldovy sítě. Převzato z [24].

označeny $x_1, x_2, \ldots, x_n$ a jeden externí výstup – ty jsou označeny $y_1, y_2, \ldots, y_n$. Po skončení vybavovací fáze jsou na těchto výstupech odpovídající výstupy sítě. Hodnoty vstupů, stavů i výstupů jsou bipolární, tj. nabývají hodnot z množiny $\{-1; 1\}$. Stejně, jak tomu je u jiných druhů neuronů, i v tomto případě má neuron svůj vnitřní potenciál ξ. Ten se počítá jako vážený součet vstupů, a protože jsou váhy celočíselné, bude i hodnota potenciálu celočíselná. Neuron aktivuje svůj výstup pomocí aktivační funkce f, kterou nejčasteji bývá ostrá nelinerialita. Její předpis je téměř totožný jako 2.3, až na hodnoty výstupů:

$$f(\xi) = \begin{cases} +1 & \text{pro } \xi \geq 0 \\ -1 & \text{pro } \xi < 0. \end{cases} \tag{2.26}$$

2.7.2 Proces učení

Narozdíl od dopředné sítě a učení pomocí zpětného šíření chyby, je v případě Hopfieldovy neuronové sítě učení neiteračním procesem. Pro každý vzor vytvoříme dílčí matici o velikosti $n \times n$. Ta bude obsahovat prvky vzniklé vynásobením i-tého a j-této vstupu. Násobíme tedy každý prvek s každým, s výjimkou totožných vstupů ($i = j$). Stejně jako matice vah, je i tato matice symetrická s nulami na diagonále. Její prvky jsou však z množiny $\{-1; 1\}$. Sečtením všech těchto s dílčích matic jednotlivých vzorů dostaneme výslednou matici vah W. Nastavení vah na základě vstupních vzorů se řídí tzv. *Hebbovým zákonem*. Ten říká, že

pokud jsou dva spolu propojené neurony aktivní, pak jsou jejich vazby zesíleny a naopak. Tento zákon je zahrnut ve vztahu pro výpočet vah mezi neurony:

$$w_{ij} = \begin{cases} \sum_{k=1}^{s} x_{ki} x_{kj} & \text{pro } i \neq j \\ 0 & \text{pro } i = j, \end{cases} \tag{2.27}$$

kde x_{ki}, resp. x_{kj} jsou i-té, resp. j-té prvky k-tého trénovacího vzoru. Vzor je taktéž binární, čili $x_{ki}, x_{kj} \in \{-1, 1\}$.

2.7.3 Vybavování v Hopfieldově síti

Vybavování je iterační proces. Nejprve jsou v čase $t = 0$ hodnoty všech výstupů nastaveny na hodnoty odpovídajících vstupů. Stavy jsou dále v každém kroku aktualizovány dle vzahu

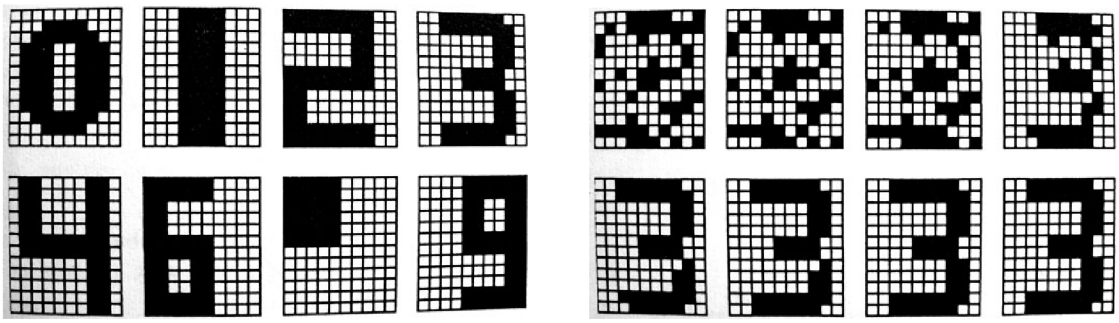
$$y_j(t+1) = f\left(\sum_{i=1}^{n} w_{ij} y_j(t)\right), \quad j = 1, 2, \ldots, n\,, \tag{2.28}$$

kde f je uvedená aktivační funkce 2.26. V každé iteraci jsou výstupy přepočítány a slouží znovu jako vstupy sítě. Tento děj opakujeme, dokud se stavy neuronů mění. Tedy v momentě, kdy jsou při dvou po sobě jdoucích iteracích stavy všech neuronů shodné, vybavovací fázi ukončíme. Získané výstupy pak odpovídají vybavenému vzoru.

2.7.4 Vlastnosti sítě

Pokud chceme Hopfieldovu síť používat jako autoasociativní paměť, musíme myslet na její omezení. Jedním z nich je, že kapacita paměti je určena poměrem počtu trénovacích vzorů s a počtu neuronů n. Pokud síť naučíme na příliš mnoho vzorů, může během vybavování konvergovat k nějakému zvláštnímu obrazci, na který naučena nebyla. K vyvarování se této chyby je nutné dodržet pravidlo $s \leq 0,138n$. Pro 10 vzorů je to minimálně 73 bodů, z čehož plyne velikost váhové matice $73^2 = 5\,329$ vah. Dalším omezením je nutnost volit trénovací vzory s co nejvyšší vzájemnou Hammingovou vzdáleností (tj. aby se lišily na co nejvíce pozicích a byly si tak co nejméně podobné). Naopak výhodou tohoto druhu neuronové sítě je, že automaticky zná všechny inverzní vzory k těm, na něž byla natrénována.

2.7.5 Příklad použití



Obrázek 2.13: Příklad trénovacích vzorů pro Hopfieldovu síť (vlevo) a jednotlivé fáze při vybavování vzoru (vpravo). Převzato z [24].

Využití Hopfieldovy sítě si můžeme demonstrovat na následujícím příkladu. Mějme 8 trénovacích vzorů z obrázku 2.13 vlevo. Velikost každého z nich je 10×12 bodů, což nám dává 120 neuronů v síti a $120^2 = 14\,400$ vah mezi nimi. Je vidět, že vzory byly zvoleny tak, aby se od sebe co nejvíce lišily. Vstupy sítě byly sestaveny po řádcích obrazce, přičemž bílá barva reprezentuje hodnotu $+1$, černá naopak hodnotu -1. Po naučení sítě byl vybrán obrázek číslice „3“, který byl znehodnocen 25% šumem. Čili 30 náhodných pixelů bylo invertováno. Takto poškozený obrazec byl předložen na vstupy sítě. Jednotlivé iterace vybavovací fáze jsou vidět na obrázku 2.13 vpravo. Během sedmi provedených iterací postupně docházelo ke zlepšování výstupu sítě, až byl ve finále rozpoznán původní vzor.

Kapitola 3

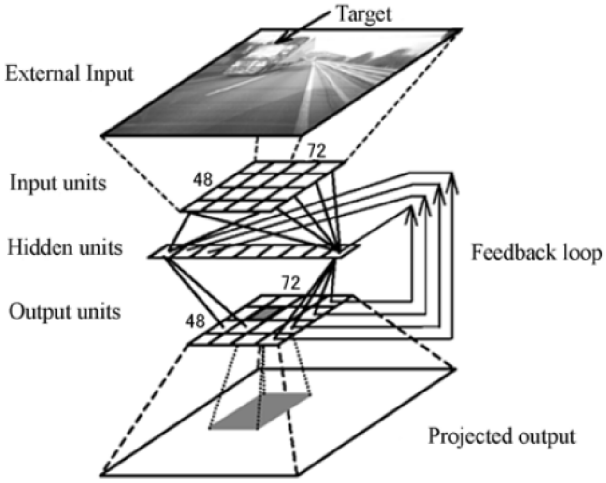
Využití rekurentních neuronových sítí v počítačovém vidění

V této kapitole jsou prezentovány některé ze zajímavých aplikací rekurentních neuronových sítí v počítačovém vidění. Jedná se o krátký přehled problémů, kde RNN našly svoje uplatnění. Snahou je demonstrovat možné přístupy k řešení takových úloh a ukázat místa, kde mohou být RNN s výhodou použity. Každý z předložených příkladů je založen na jiném modelu RNN. Pro hlubší pochopení problematiky jsou uvedeny odkazy na původní práce.

3.1 Detekce jedoucího vozidla

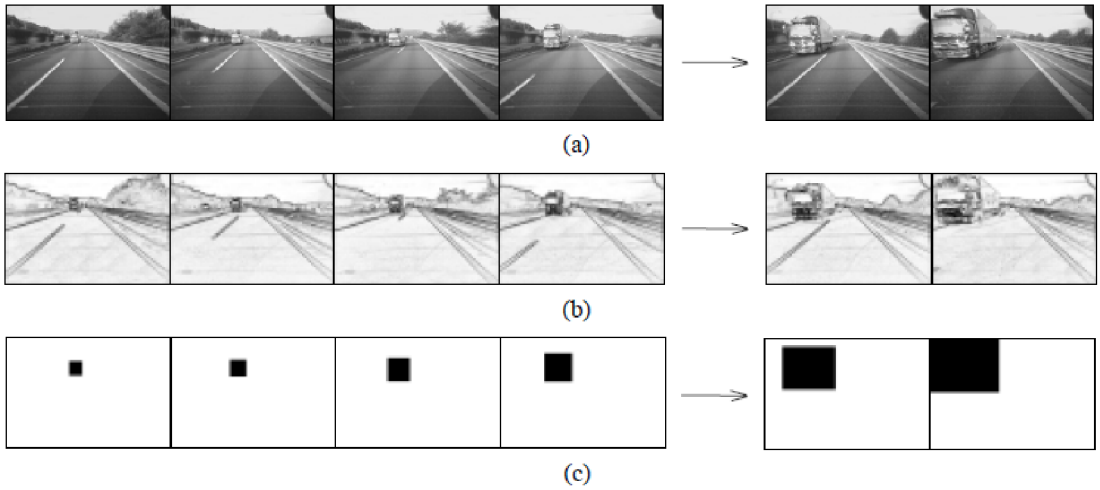
Pánové Inagaki, Sato a Umezaki z Japonska v roce 2003 publikovali článek *„A Recurrent Neural Network Approach to Rear Vehicle Detection Which Consider State Dependency“* [13]. Představují v něm nový přístup k detekci pohybujícího se vozidla ve videu pořízeném ze zadního okna automobilu jedoucího před ním. U postupů založených na počítačovém vidění vývojáři často musí čelit nejrůznějším změnám v obraze, jako je počasí, perspektiva, osvětlení apod. Všechny tyto aspekty samozřejmě komplikují správnou detekci. Hodně přístupů používá k detekci pouze jeden snímek. Spolehlivost těchto metod je poměrně nejistá, pokud jsou zatíženy reálnými vlivy, tedy hlavně oslněním sluncem a odlesky, kdy se sledované vozidlo na chvíli ztratí z dohledu. Narozdíl od běžných metod se tedy autoři snaží využít souvislosti mezi jednotlivými snímky a z tohoto důvodu využívají právě rekurentní neuronové sítě.

Architektura použité sítě je vidět na obrázku 3.1. Stavy neuronů výstupní vrstvy jsou přivedeny na vstupy vrstvy skryté, což odpovídá architektuře **Jordanovy sítě**. Jestli ale byla nějak upravena, autoři bohužel neuvádí. Jako učící algoritmus byl použit Backpropagation through time. Při učení byly na vstupy přikládány po sobě jdoucí snímky, z nichž v každém byly nejprve Sobelovým operátorem zvýrazněny horizontální a vertikální hrany a následně byl zmenšen na rozměr 72×48 pixelů. Z toho plyne, že velikost vstupní a výstupní vrstvy sítě byla 3456 neuronů. Skrytá vrstva obsahovala během experimetování 48, 64, 96 nebo 128 neuronů. Učení probíhalo na 12 pořízených scénách, testování potom na dalších 5. Data obsahovala odlišné barvy vozidel, různé změny jejich trajektorie, předjíždění a proměnlivou vzdálenost sledovaného auta od kamery. V 11% snímků byly automobily nezřetelně viditelné kvůli špatným světelným podmínkám, resp. přílišnému slunečnímu svitu. Snímky byly pořízeny CCD kamerou na japonských silnicích za denního světla. Obraz byl ve 256 stupních šedi velikosti 720×480 pixelů. Očekávaným výstupem RNN byl binární



Obrázek 3.1: Architektura použité sítě.

obraz, kde bylo bílou barvou označeno pozadí a černou oblast automobilu (viz obrázek 3.2).



Obrázek 3.2: Ukázka trénovacích dat: (a) pořízená sekvence snímků, (b) obrázky s detekovanými hranami, (c) očekávané výstupy sítě.

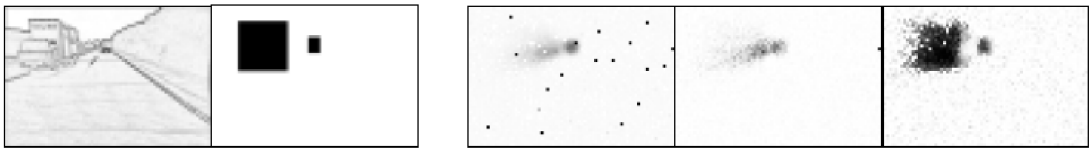
Z trénovacích dat byl vždy vybrán náhodný interval, který byl opakovaně předkládán síti. Vždy, když byla přiložena nová scéna, byly váhy zpětnovazebních spojů uvedeny do původního stavu. Ukončení učícího procesu bylo podmíněno následující rovnicí:

$$E_t = \frac{1}{N} \sum_{t=0}^{N} \frac{1}{2} (R_{out}(t) \oplus R_{cri}(t)), \tag{3.1}$$

kde R_{out} značí oblasti, kde jsou výstupy $y_i \geq 0.2$, R_{cri} požadovaná oblast ve výstupu sítě a N je počet snímků. Když E_t překročilo zvolený práh, bylo učení skončeno. Při následné evaluaci výsledků detekce vozidla bylo použito kritérium

$$R_{eval} = \frac{R_{out} \bigcap R_{cri}}{R_{cri}} \tag{3.2}$$

Každý snímek, kde poměr R_{eval} dosáhl prahu 0.5, byl vyhodnocen jako správně určený. Jak je vidět na obrázku 3.3, po 1 000 epochách učení je na výstupu několik izolovaných pixelů. Síť reagovala na oblasti s vysokou hustotou hran na vstupu, ale nikoliv již na ty s hustotou nízkou. Po 5 000 epochách izolované pixely zmizely, avšak síť pořád nedávala patřičné výstupy v oblastech s nízkou hustotou hran. Teprve po 10 000 epochách byl pozorován odpovídající výstup i na tyto oblasti. Výstupní obraz se velmi podobal vzoru, na který byla síť naučena.



Obrázek 3.3: Výstupy sítě během trénování. Vlevo vstup a očekávaný výstup, vpravo výstupy sítě po 1 000, 5 000 a 10 000 epochách učení.

Výsledky ukazují, že se rekurentní síť dokázala naučit detekovat jedoucí automobily tak, jak bylo požadováno. Navíc se velice dobře vypořádává i s velkými změnami v jasu a okolním osvětlení. Je to tím, že bere v potaz i předchozí snímky, jejichž vliv zůstáva zakódován ve vnitřním stavu sítě. Odpověď sítě tedy není vždy na jeden aktuální snímek, ale je ovlivněna i předchozím stavem sítě. Kdy ale popsaná metoda selhává, jsou situace, kdy sledované auto náhle mění svoji polohu vůči předem natrénované trajektorii.

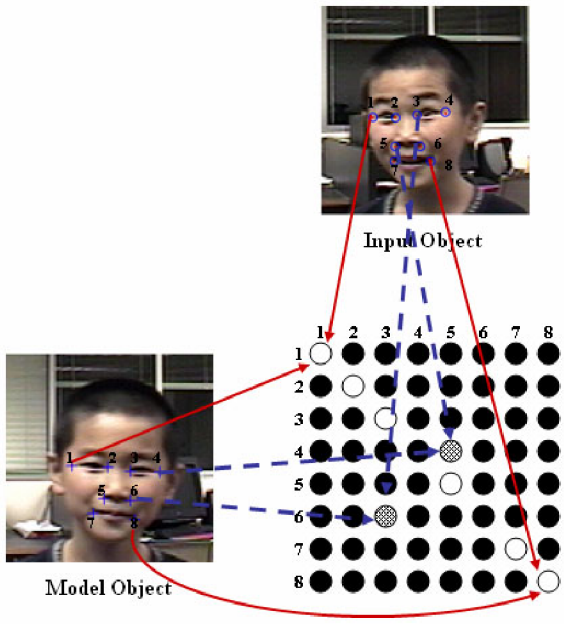
Autoři se ve svém článku věnují i srovnání přístupu využívajícím RNN s klasickými vícevrstvými perceptrony *(MLP)*. Zjistili, že na stejných datech, na kterých síť nebyla naučena, dosahuje RNN úspěšnosti 93,4%, narozdíl od sítě typu MLP, která dosahuje úspěšnosti pouze 80%. Vícevrstvý perceptron často dával výstup, kde byla chybně označena vozidla jedoucí opačným směrem. Tímto neduhem rekurentní neuronová síť netrpí, což bude opět vlivem souvislosti mezi následujícími snímky, kterou si tato síť uchovává. Klasická dopředná síť má totiž v jednom časovém okamžiku informace pouze o aktuálním obraze.

3.2 Detekce a trekování obličeje s verifikací pomocí RNN

V roce 2006 byl publikován článek *„Recurrent Neural Network Verifier for Face Detection and Tracking“*, jehož autory jsou S. Yoon, G. Hur a J. Kim [36]. Autoři se ve své práci věnují detekování obličeje ve videosekvekci a jeho následnému trekování. Popisují robustní algoritmus, kde nejprve využívají barvy kůže a gradientu k označení kandidátských oblastí v obraze, odpovídajícím obličeji, a následně geometrikých příznaků, jež jsou předloženy **Hopfieldově neuronové síti**. Ta potom provádí verifikaci, zda se opravdu jedná o obličej. Neuronová síť tedy řeší část úlohy, kdy dochází k „potvrzení“ výsledků předchozích kroků algoritmu.

U aplikací počítačového vidění, jako je toto, je velmi důležité, aby příznaky získáváné z obrazu, byly invariantní vůči posunutí, rotaci a změně měřítka. K extrakci geometrických příznaků byla oblast získaná předchozí segmentací, v níž by se měl obličej vyskytovat,

převedena na binární obraz. K tomu bylo použito dynamického prahování. Následně byly detekovány okrajové (rohové) body jednotlivých částí obličeje. Konkrétně to bylo následujících osm: vnější a vnitřní koutky pravého a levého oka, okraje nosu a koutky úst. Vychází z toho, že nejvíce informací je uloženo právě v bodech, kde dochází k největšímu zakřivení tvarů. Algoritmus provádějící tuto detekci autoři publikovali již dříve. Odkaz na článek je uveden přimo v [36]. Mezi takto získanými body můžeme hledat další příznaky – úhly a vzdálenosti mezi nimi.

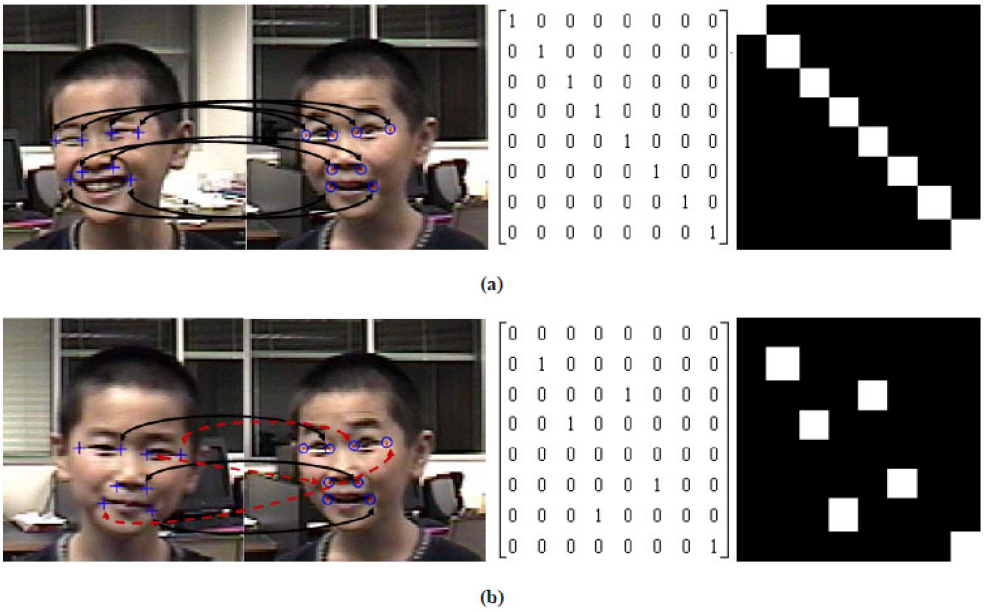


Obrázek 3.4: Znázornění vstupních dat pro verifikaci.

Z popsaných příznaků byl sestaven modelový graf. Každý uzel tak nesl vlastní, lokální příznak a příznaky ve vztahu k ostatním uzlům. Ke znázornění procesu verifikace bylo vytvořeno dvourozměrné pole uzlů. Řádky odpovídaly významným bodům v modelovém (referenčním) obraze a sloupce zase těm v obraze vstupním. Stav každého uzlu odpovídá míře, jakou se od sebe liší odpovídající si body v porovnávaných obrazech. Např. bílou barvou je na obrázku 3.4 znázorněna shoda obou bodů. Šedá neopak značí neshodu. Obecně i-tému bodu v modelovém obraze a k-tému bodu ve vstupním obraze náleží uzel s indexy (i, k). Toto pole je tak vlastně analogií pro zapojení neuronů v Hopfieldově síti. Pokud tedy mají být oba objekty navzájem verifikovány, měly by všechny uzly na hlavní diagonále mít po ustálení stavu sítě bílou barvu (hodnotu 1). Problém verifikace je tak popsán jako optimalizační úloha, která je vhodná k řešení neuronovými sítěmi.

Obrázek 3.5 znázorňuje dva možné výsledky verifikace. Nahoře je ukázka úspěšné verifikace, kdy byly všechny body sesouhlaseny. Dole je naopak výsledek neúspěšné verifikace. Tři páry příznakových bodů si odpovídaly, tedy č. 2 (vnitřní koutky pravých očí), 6 (levé okraje nosů) a 8 (levé koutky úst) – tj. uzly o souřadnicích $(2, 2)$, $(6, 6)$ a $(8, 8)$. Jednička mimo diagonálu značí nesprávné spárování. Třeba jednička na souřadnicích $(3, 5)$ je v důsledku toho, že vnitřní koutek oka na modelovém objektu byl označen jako pravý okraj nosu objektu vstupního. Je zřejmé, že uzel má hodnotu 0, když významnému bodu v modelu nebyl přiřazen žádný bod ze vstupního obrazu. Verifikace v tomto případě selhává, jelikož pro po-

zitivní verifikaci je potřeba minimálně pěti jedniček na diagonále (tedy sesouhlasení alespoň 5 z 8 párů bodů).



Obrázek 3.5: Ukázka úspěšné verifikace (a) a chybné verifikace (b).

Autoři ve své práci používají dvou rozdílných modelů pro popis obličeje. Jsou to *three–face reference model (TFRM)*, kdy je tvář popsána třemi obrazy – zepředu, zleva a zprava a *single–face reference model (SFRM)*, kdy je obličej popsán jediným snímkem zepředu. V případě, že byl při testování použit TFRM, byl vstupní obrázek porovnán se všemi třemi modelovými a stačilo, když došlo ke shodě alespoň u jednoho z nich. Testovací sada obsahovala 400 sekvencí snímků. Každý obrázek obsahoval stejný obličej, ale vždy v jiné pozici, měřítku a jiným výrazem ve tváři. Z této databáze byly vybírány jak referenční, tak testovací obrázky. Nejprve je v článku uvedeno srovnání SFRM a TFRM v případě, kdy byly pro verifikaci vybírány náhodné obrázky z databáze. Model SFRM dokázal úspěšně verifikovat až 94,14%, TFRM až 100% objektů ze všech 17. V dalším testu již nebyly modelové snímky vybírány náhodně, nýbrž porovnání proběhlo vždy mezi dvěma následujícími snímky v sekvenci. Zmíněný algoritmus v tomto případě dosáhl ještě lepších výsledků, nicméně selhával ve chvílích, kdy došlo ke špatné segmentaci obličeje na jednom z obrázků. V třetím testu byly simulovány situace, kdy 2 nebo 3 z významných bodů kvůli chybné detekci vůbec ve snímcích nebyly zastoupeny. Výsledky ukazují, že i v tomto případě byl TFRM odolnější, správné verifikace dosahoval až ve 74,51% případů oproti SFRM, jehož úspěšnost byla pouze 68,63%. Navíc model TFRM nebyl tolik náchylný na změny v nastavení tolerance. Poslední z prezentovaných testů popisuje stabilitu systému ve chvílích, kdy byly jednotlivé příznakové body posunuty mimo svou původní polohu, opět kvůli jejich špatné detekci. Oba modely dosahovaly úspěšnosti až 100% v závislosti na zvolené toleranci. Síla TFRM se projevuje v menší náchylnosti na tyto změny.

Z provedených testů vyplývá, že HNN je schopna úspěšně plnit úlohu verifikace při trekování obličejů ve videosekvenci. Autoři zvolili přístup využívající rekurentní neuronovou síť pouze z důvodu zvýšení přesnosti při řešení tohoto problému. Pokud jsou vizuální příznaky, tedy barva, gradienty a geometrie zkoumaného objektu vhodně zkombinovány, lze

díky vybranému modelu neuronové sítě dosáhnout kvalitních výsledků. Bohužel v publikovaném článku nejsou uvedena konkrétní srovnání v přístupu bez a s využitím popsaného verifikátoru. Dále autoři uvádějí, že budoucí vývoj by mohl být místo verifikace obličejů zaměřen na jejich rozpoznávání.

3.3 Rozpoznávání obličeje ze sekvence snímků

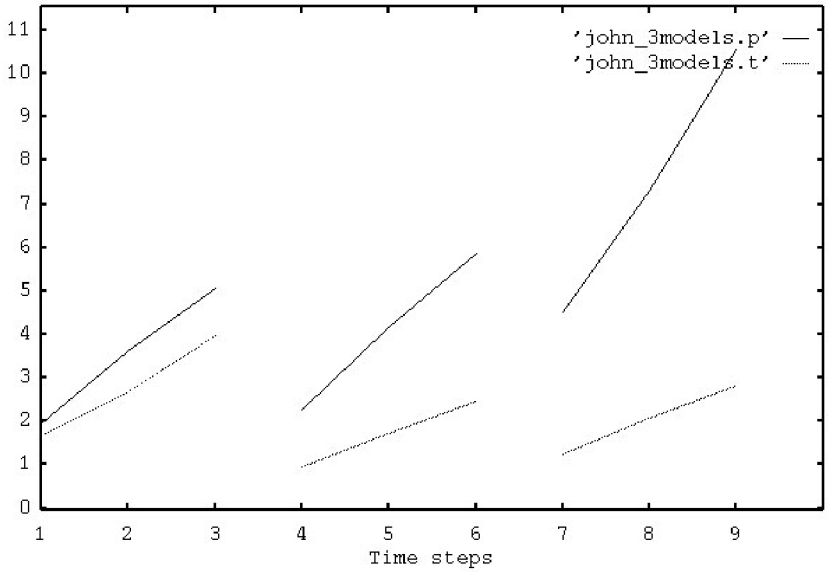
Další možné využití RNN v oblasti počítačového vidění lze nalézt v práci od S. Gonga a kolektivu autorů s názvem „*Tracking and Recognition of Face Sequences*“ [8]. Autoři se zde zabývají problematikou sledování obličeje ve videosekvenci a jejich následné identifikaci. K identifikaci používali dopřednou neuronovou síť, ale v závěru článku, konkrétně v podkapitole „*Learning Temporal Signatures in Face Sequence*“, uvádí i experimenty s **Elmanovou rekurentní neuronovou sítí**.

Z videosekvence bylo vybráno 5 snímků tváře, z nichž každý odpovídal jinému směru pohledu (měnícího se po 45°) Z obrázků byly metodou eigenfaces (viz podkapitola 4.3.1) extrahovány příznakové vektory. Dále byla zkoumána změna těchto příznakových vektorů v čase, kterou autoři definovali jako euklidovskou vzdálenost vektorů dvou po sobě jdoucích snímků. Jelikož byla PCA analýza prováděna pro každou trénovanou osobu zvlášť, byl příznakový prostor rozdělen do podprostorů, kde každý z nich odpovídal právě jednomu člověku. Pro 3 různé obličeje tedy byly natrénovány 3 nezávislé neuronové sítě. Vstupem byl příznakový vektor a požadovanýn výstupem vektor následujícího snímku v sekvenci. Trénování proběhlo algoritmem backpropagation. Síť se tedy měla naučit předpovídat další snímek v řadě.

V popsaném experimentu použili pro každou ze 3 osob 10 sekvencí obrázků tváří, každou o 5 snímcích. Následně mohlo být pro každý podprostor vypočítáno 50 příznakovách vektorů, přičemž z celkového počtu 49 eigenfaces bylo vzato pouze prvních 20. Na těchto datech byly natrénovány tři RNN, každá pro jednu osobu. Vstupní a výstupní vrstva měly 20 neuronů, skrytá a kontextová potom 30. Učící proces trval 6 000 epoch. Poté bylo cílem ověřit, zda natrénované sítě dokáží správně rozpoznat sekvenci obrázků od neznámého člověka. Vzali tedy novou sekvenci snímků od jedné z oněch 3 osob, z nichž promítnutím do třech podprostorů obličejů vypočítali nové 3 příznakové vektory. Ty pak byly přiloženy na vstupy odpovídajících sítí. V grafu na obrázku 3.6 jsou zaneseny euklidovské vzdálenosti mezi po sobě jdoucími vstupy sítě a po sobě jdoucími výstupy sítě. Je možno pozorovat, že dvojice křivek těchto vzdáleností jsou si nejblíže na obrázku úplně vlevo. Právě ty odpovídají příznakovým vektorům identifikované osoby. Lze tedy usoudit, že neuronová síť úspěšně předpovídala následující snímky. Z toho důvodu se jejich příznaky měnily nejpodobněji právě u té sítě, patřící neznámé osobě. Znamená to také, že změna příznakových vektorů obrázků tváří v čase popisuje druh temporálních příznaků v dynamicky se měnící scéně. Článek se touto problematikou bohužel zabývá jen okrajově, takže chybí další obsáhlejší testy a experimenty.

3.4 Rozpoznávání ručně psaného písma

Zajímavou oblastí počítačového vidění je bezesporu rozpoznávání písma. Zvláště pak, jde-li o písmo ručně psané. Tímto tématem se zabývají autoři A. Graves, M. Liwicki, S. Fernández a kol. ve svém článku „*A Novel Connectionist System for Unconstrained Handwriting Recognition*“ z roku 2009.



Obrázek 3.6: Euklidovské vzdálenosti mezi po sobě jdoucími příznakovými vektory.

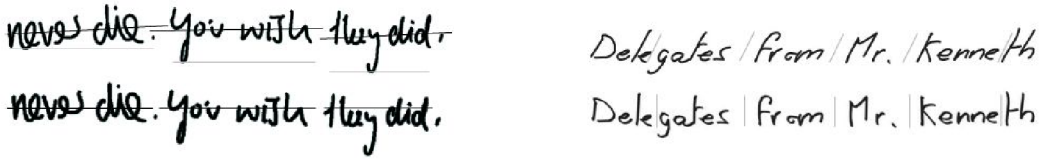
Rozpoznávání ručně psaného písma obvykle rozdělujeme do dvou skupin – online a offline zpracování. Při online zpracování máme k dispozici posloupnost souřadnic pera v čase, zatímco při tom offlinovém je vstupem pouze samotný obrázek. Díky jednodušší extrakci příznaků přináší online přístup obvykle lepší výsledky. Další velký rozdíl je potom v rozpoznávání izolovaných písmen nebo slov a celých řádků textu. Druhá varianta je samozřejmě o mnoho obtížnější. Posledním kritériem, které je nutno zohlednit, jsou potom omezení, která klademe na podobu vstupních dat našeho systému. Pokud bude schopen přijímat např. pouze čislice nebo tiskací písmo, nepůjde o tak těžký úkol, jako když vstupní data budou neomezená. Tak jako tak, vytvoření spolehlivého univerzálního systému pro zpracování a rozpoznávání ručně psaného textu je dodnes otevřeným problémem.

Autoři se v článku věnují neomezenému spojitému rozpoznávání ručně psaného písma a to jak online, tak offline variantě. Místo, k tomuto účelu zřejmě nejpoužívanějších, skrytých Markovových modelů (*Hidden Markov Models - HMM*[1]) se rozhodli vyzkoušet rekurentní neuronovou síť typu bidirectional long short–term memory. Jedním z důvodů je ten, že u HMM výstupní pravděpodobnost závisí pouze na aktuálním stavu. Je tedy poněkud těžké zaznamenávat nějaký kontext. Model LSTM byl použit také proto, že pro trénování klasických RNN je nutné mít oddělená trénovací data pro každý vstup a to je činí použitelné pouze pro rozpoznávání jednotlivých izolovaných znaků.

3.4.1 Zpracování online dat

Online data pro experimenty byla pořízena na tabuli se speciálním zařízením eBeam pro snímání pohybu pera. Výstupem tohoto zařízení byly souřadnice bodů (x, y) společně s časovými značkami. Záznam byl prováděn pouze v době, kdy pero bylo v pohybu a v kontaktu s tabulí. Tyto jednotlivé úseky autoři nazvali *strokes* (do češtiny možno přeložit jako tah nebo úhoz). Po krátké úpravě chybějících a šumových bodů byla data uložena ve formátu XML se vzorkovací frekvencí 30–70 bodů za sekundu.

[1] http://en.wikipedia.org/wiki/Hidden_Markov_model



(a) upravení směru textu „do kopce“ nebo „z kopce“

(b) upravení sklonu písma

Obrázek 3.7: Ukázka předzpracování rozpoznávaného textu. Převzato z [9].

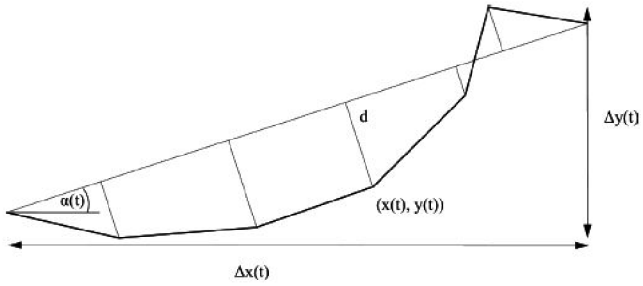
Jelikož každé písmo má jinou výšku, sklon a směr, byla prvním krokem předzpracování normalizace trénovacích dat – viz ukázka na obrázku 3.7. Autoři také zmiňují vypozorovaný fakt, že se styl písma psaného na tabuli liší od písma psaného na papír. I to museli vzít při normalizaci v potaz. Stejně tak bylo nutné normalizovat rychlost psaní. Souřadnice tedy byly transformovány tak, aby byly rovnoměrně rozmístěny vždy se stejnou mezerou mezi dvěma sousedními body.

Následovala extrakce příznaků. Příznaků pro online zpracování bylo celkem 25 pro každou souřadnici (x, y) a byly rozděleny do dvou skupin. Zde jsou vypsány pouze některé z nich:

1. Temporální příznaky získáné z předchozích a následujících sousedních bodů v čase:
   - binární údaj, zda se pero dotýkalo tabule či nikoliv
   - rychlost v daném bodě
   - x-ová souřadnice po filtraci horní propustí
   - y-ová souřadnice po normalizaci
   - kosinus a sinus úhlu mezi úséčkami k předchozímu a následujícímu bodu sekvence (zakřivení)
   - poměr $\frac{\Delta y(t) - \Delta x(t)}{\Delta y(t) + \Delta x(t)}$ (viz obrázek 3.8)
   - kosinus a sinus úhlu α mezi horizontální osou řádku a úsečkou mezi prvním a posledním bodem sledovaného okolí
   - délka úsečky mezi prvním a posledním bodem sledovaného okolí vydělená $max\left(\Delta x(t), \Delta y(t)\right)$
   - průměrný čtverec vzdálenosti d^2 každého bodu sledovaného okolí od úsečky proložené jeho prvním a posledním bodem
2. Prostorové příznaky získané z obrázku písma (matice bodů):
   - počet bodů nad *corpus line* a pod *baseline* (přímky určující horní, resp. dolní okraj malých písmen) ve svislém okolí zkoumaného bodu
   - počet černých bodů v okolí 3×3 daného bodu

3.4.2 Zpracování offline dat

Pro extrakci příznaků byla u tohoto přístupu použita metoda klouzavého okna. Z každého řádku textu tak vznikla posloupnost 9 prvkých vektorů; šířka okna byla 1 pixel. Opět jsou zmíněny pouze některé vybrané příznaky:



Obrázek 3.8: Sledované okolí bodu $(x(t), y(t))$. Na tomto obrázku jsou uvažovány 3 předchozí a 3 následující sousední body. Převzato z [9].

- průměrná intenzita pixelu
- těžiště pixelů
- pozice nejvyššího a nejnižšího pixelu + rychlost jejich změny s ohledem na sousední okna
- počet přechodů z černé do bílé barvy a poměr černých pixelů mezi nejvyšším a nejnižším pixelem

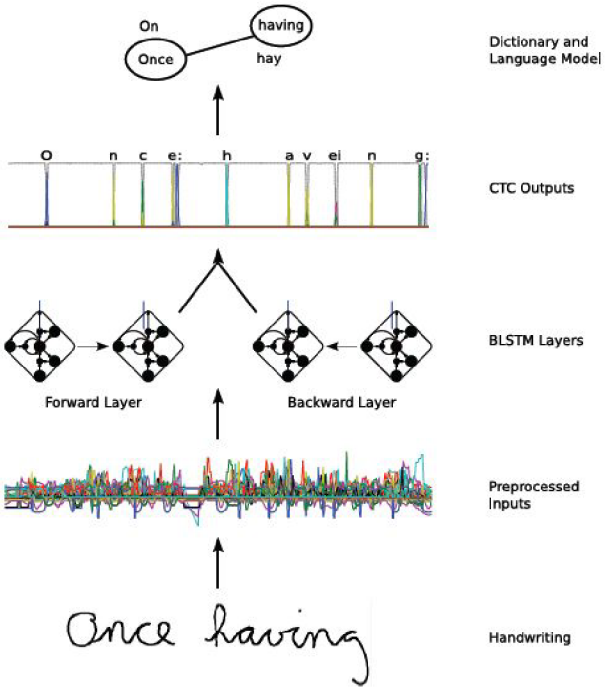
3.4.3 Proces rozpoznávání

Jak již bylo zmíněno v úvodu této podkapitoly, autoři ve svém díle použili RNN typu **bidirectional long short–term memory**. Síť typu LSTM již byla popsána v kapitole 2.6. Pro mnoho úloh je dobré mít informace o kontextu nejenom předešlém, ale také budoucím. Kupříkladu u rozpoznávání písma chceme znát okolí písmene jak vpravo, tak vlevo od něj. Právě proto vznikly obousměrné rekurentní sítě *(bidirectional recurrent neural networks – BRNN)* umožňující přístup ke kontextu z obou stran. Obsahují dvě skryté vrstvy. Jedna zpracovává vstupní sekvenci odpředu dozadu, druhá pak naopak. Obě jsou potom napojeny na jednu výstupní vrstvu. Kombinací BRNN a LSTM dostáváme bidirectional LSTM, tedy BLSTM.

Tradiční RNN vyžadují pro jejich trénování předsegmentovaná data – množinu dvojic *(vstupní vektor, výstupní vektor)*. U aplikací, jakou je např. rozpoznávání ručně psaného písma se ale potýkáme se zásadním problémem – taková data je obtížné správně segmentovat. Navíc je potřeba nezávislé výstupní vektory podrobit nějakému dalšímu zpracování *(post-processing)* a převést je tak do požadové formy, v tomto případě posloupnosti znaků nebo slov. Speciálně k účelu označování sekvencí *(sequence labelling)* byl navržen nový typ výstupní vrstvy RNN, označovaný jako *connectionist temporal classification (CNC)*. Ten nevyžaduje předsegmentovaná data a na svůj výstup dává pravděpodobnosti jednotlivých značek. Vrstva obsahuje právě tolik výpočetních jednotek, kolik má být systém schopen rozeznávat různých značek (znaků) plus dvě navíc – *„blank“* a *„no label“*. Součet výstupních pravděpodobností je 1. Pro získání výsledných sekvencí slov bylo zjištěné rozložení pravděpodobnosti jednotlivých znaků předloženo slovníkovému a jazykovému modelu. Pro podrobnější informace odkazuji na původní článek [9].

Diagram 3.9 ilustruje fungování celého systému. Konkrétní model RNN, který byl zmíněn v tomto článku, obsahoval 100 paměťových bloků v obou skrytých vrstvách LSTM.

Výstupní CNC vrstva potom měla 81 uzlů. Počet vstupů závisel na typu zpracování dat – 25 pro online a 9 pro offline přístup.



Obrázek 3.9: Schéma systému pro rozpoznávání ručně psaného písma, popis v textu. Převzato z [9].

3.4.4 Dosažené výsledky

Při vyhodnocování výsledků představeného systému se autoři věnovali především srovnání se systémem založeným na HMM. Při zpracování online i offline dat vykazuje jejich přístup o mnoho lepší výsledky – viz tabulka 3.1. Dokonce i ve srovnání s komerční aplikací Microsoft tablet PC handwriting recognizer si vede lépe. Dosažené výsledky se samozřejmě lišily v závislosti na velikosti použitého slovníku a dalších parametrech. Pro detailní popis provedených testů opět odkazuji na původní článek. Zvolený model rekurentní neuronové sítě se každopádně ukázal jako velice vhodný pro tento typ úlohy. Testování probíhalo na databázích IAM Handwriting Database[2] a IAM On-Line Handwriting Database[3].

	Online data		**Offline data**	
Systém	**Přesnost slov**	**Přesnost znaků**	**Přesnost slov**	**Přesnost znaků**
HMM	65,0 %	–	64,5 %	–
RNN	79,7 % ± 0,3 %	88,5 % ± 0,05 %	74,1 % ± 0,8 %	81,8 % ± 0,6 %

Tabulka 3.1: Výsledky systému pro rozpoznávání ručně psaného písma založeného na BLSTM ve srovnání s HMM.

[2] http://www.iam.unibe.ch/fki/databases/iam-handwriting-database
[3] http://www.iam.unibe.ch/fki/databases/iam-on-line-handwriting-database

Kapitola 4

Rozpoznávání obličejů

V této kapitole jsou vysvětleny pojmy detekce, identifikace a verifikace obličeje a jsou vzpomenuty aplikace těchto technik v různých odvětvích lidské činnosti. Pobrobněji je vysvětlena metoda eigenfaces pro rozpoznávání obličejů.

4.1 Využití metod detekce a identifikace obličejů

V dnešní době existuje mnoho oblastí využití metod detekce a rozpoznávání lidské tváře v obraze, popř. ve videu. Je to především z důvodu masivního rozšíření snímacích a výpočetních zařízení mezi širokou veřejnost. Téměř každý digitální fotoaparát již disponuje funkcí pro detekci obličeje nebo rovnou detekci úsměvu. Jde také o zábavné aplikace typu Picasa od spol. Google, které touto funkcí uživatelům usnadňují třídění a označování fotografií v jejich albech. Někteří výrobci dodávají ke svým webkamerám software, který umožňuje nejrůznější žertovné hrátky s obličejem uživatele – přidání vousů, masek či třeba klobouku. Velkým přínosem jsou tyto techniky i v biometrii. Po otiscích prstů je to totiž nejpoužívanější biometrická vlastnost. Existují zařízení provádějící ověřování uživatelů právě podle obličeje – mohou to být přístupové terminály v továrnách, ale i programy pro usnadnění přihlašování např. do operačního systému počítače. Využití nacházíme i v kriminalistice ke sledování nebo rozpoznávání podezřelých osob nebo na různých úřadech k ověření totožnosti osoby, která se může domáhat jistých dávek nebo vstupu do země. Některé automobilky dokonce již nabízí moderní systémy do jejich vozů, které jsou mimojiné schopny odhalit zavřené oči a upozornit tak řidiče na možný mikrospánek za volantem.

V případě detekce obličeje mluvíme o určení oblasti obrazu, která tento obličej obsahuje. Rozlišujeme ale i pojmy *identifikace (rozpoznávání)* a *verifikace* obličeje. Identifikace je porovnávací proces (1:N), kdy z databáze vybíráme takovou tvář, která je té hledané nejpodobnější. V případě verifikace jde o ověření (1:1), zda si dvě zadané tváře odpovídají, resp. zda obě patří stejnému jedinci. [23, 5]

4.2 Detekce obličeje

Základním předpokladem pro úspěšné rozpoznání obličeje je jeho korektní detekce v obraze. Ne vždy totiž máme k dispozici přímo fotografii s portrétem osoby. Častěji jde spíše o surové fotografie, kde se kromě samotné tváře nachází i další objekty. Jde tedy o proces, kdy se snažíme určit, zda se v daném obraze obličej vyskytuje, či nikoliv. Pokud ano, lze tuto oblast lokalizovat a následně postoupit dalšímu zpracování. Nemusí však jít jen o de-

tekci obličeje jako celku. Můžeme chtít označit i jeho jednotlivé části, tedy oči, ústa a nos. Rozšířením může být i určení směru pohledu.

Přístupů k lokalizaci lidské tváře v obraze existuje mnoho. Znalostní metody využívají definovaných pravidel, které popisují „typickou tvář“. Víme, že každý obličej se skládá ze dvou očí, které jsou symetrické. Mezi nimi se nachází nos a pod ním ústa. Mohou být také definovány relativní vzdálenosti mezi jednotlivými částmi. Máme také metody využívající invariatních rysů, tedy rysů, které nepodléhají proměnným podmínkám osvětlení nebo natočení obličeje. Může jít např. o zkoumání distribuce šedé barvy v obraze. I když dva jedinci mají rozdílný vzhled, tak oblast očí bude vždy tmavší než čelo atp. Další často používanou technikou je detekce obličeje na základě barvy kůže. Pokud obraz převedeme do barevného modelu YCbCr[1], můžeme pozorovat, že barva lidské kůže v tomto modelu zaujímá určitý kompaktní podprostor. Ohraničením tohoto podprostoru lze klasifikovat jednotlivé oblasti obrazu.

Všechny tyto algortimy se musí umět vypořádat s řadou proměnných vlastností obrazů. Jsou jimi natočení obličeje, rozličné typy pozadí, výraz ve tváři, nehomogenní osvětlení, ale i celkové osvětlení scény. Svou roli pak také hraje to, zda osoba na fotografii nosí brýle či má vousy. Některé z těchto neduhů můžeme eliminovat např. ekvalizací histogramu (změna kontrastu obrazu) nebo některými druhy filtrace. [32, 11]

4.3 Metody používané pro identifikaci

Metod pro extrakci příznaků z obrazu tváře a jejich následné využití pro účely rozpoznávání obličejů existuje celá řada. Zmiňme např. klasifikátor AdaBoost, statistické modely AAM *(Active Appearance Model)* a ASM *(Active Shape Model)*, analýzu nezávislých komponent (ICA) nebo analýzu hlavních komponent (PCA) [29, 30]. Jelikož tato práce není primárně o rozpoznávání obličejů, ale o rekurentních neuronových sítích, podrobněji si popíšeme pouze poslední zmíněnou metodu, tedy PCA a techniku eigenfaces, která bude použita v praktické části práce.

4.3.1 Eigenfaces

V této podkapitole bude popsán princip metody analýzy hlavních komponent (PCA) a její využití pro rozpoznávání obličejů. Pro uvedení do kontextu budou vysvětleny také pojmy vlastní vektor a vlastní číslo, spadající do oblasti lineární algebry.

Vlastní vektory, vlastní čísla

Vlastním vektorem *(eigenvector)* čtvercové matice A je označován takový nenulový vektor v, jehož směr se po vynásobení maticí nemění. Koeficient λ, o který se mění velikost vektoru, se potom nazývá vlastní číslo *(eigenvalue)*.

$$Av = \lambda v \tag{4.1}$$

Předpona *eigen-* vzešla z německého slova „eigen“, což znamená vlastní. Od toho také název eigenfaces, tedy vlastní obličeje.

Rovnici 4.1 můžeme zapsat také jako $(A - \lambda E)v = 0$, kde E značí jednotkovou matici řádu n. Determinant $A(\lambda) = |A - \lambda E|$ nazýváme **charakteristický polynom matice** A.

[1] http://en.wikipedia.org/wiki/YCbCr

Tento polynom stupně n má v oboru komplexních čísel n_λ kořenů, kde $1 \leq n_\lambda \leq n$. Rovnice 4.2 se pak nazývá **charakteristická rovnice matice** A a jejími kořeny jsou právě vlastní čísla matice A.

$$det(A - \lambda E) = 0 \tag{4.2}$$

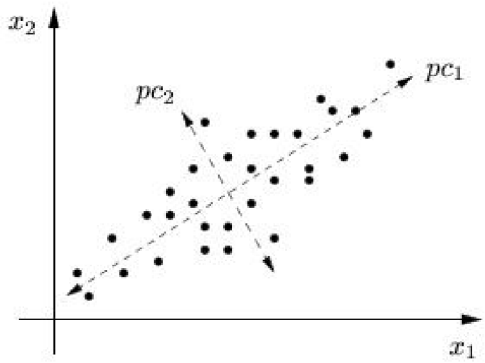
Dle předchozí rovnice vypočteme všechna vlastní čísla λ_i. Výpočet vlastního vektoru v_i už je jednoduchým řešením homogenní soustavy rovnic

$$(A \ - \lambda_i E)v_i = \mathbf{o} \tag{4.3}$$

Ke každému vlastnímu číslu λ_i existuje m_i lineárně nezávislých vlastních vektorů, kde $1 \leq m_i \leq n_i$ je geometrická násobnost vlastního čísla λ_i a n_i označujeme jako algebraickou násobnost vlastního čísla, tj. násobnost kořene λ_i charakteristického polynomu matice. [34, 21]

Analýza hlavních komponent

Analýza hlavních komponent (angl. *Principal Component Analysis – PCA*) je statistická metoda pro redukci dimenze příznakového prostoru tak, aby zůstalo zachováno co největší množství informace. Byla objevena v roce 1901 K. Pearsonem. Cílem je nalezení ortogonální báze, která bude co nejlépe popisovat distribuci dat v původním prostoru. Jedná se tedy o lineární transformaci, která převádí korelované vstupní proměnné na nové, nekorelované proměnné, nazvané **hlavní komponenty**. Přičemž počet hlavních komponent je menší nebo roven počtu původních proměnných a jsou seřazeny dle rozptylu, od největšího k nejmenšímu. V novém prostoru, tvořeném onou ortogonální bází, bude tedy na první souřadnici ležet první hlavní komponenta, která popisuje největší variabilitu dat v prostoru původním. Ukázka PCA analýzy 2D dat je na obrázku 4.1. Pomocí této metody můžeme tedy i z dat ve vysokodimenzionálním prostoru vyextrahovat jakési vzory určující jejich vzájemnou podobnost a rozdíly. [33, 29]



Obrázek 4.1: Ukázka PCA s hlavními komponentami pc_1 a pc_2. Převzato z [12].

Použití analýzy hlavních komponent si ukážeme přímo pro účely rozpoznávání obličejů, ikdyž její využití je samozřejmě daleko obecnější.

Identifikace obličeje pomocí PCA

Obrázky obličejů můžeme jednoduše reprezentovat jako sloupcové vektory $\Gamma_1, \Gamma_2, \ldots, \Gamma_M$. Délku vektoru označme N, je rovna šířce krát výšce obrázku. Samozřejmě podmínkou je, že všechny obrázky mají stejné rozměry. Vypočítáme průměrný obličej jako aritmetický průměr těchto vektorů:

$$\Psi = \frac{1}{M} \sum_{i=1}^{M} \Gamma_i \tag{4.4}$$

Odečtením průměrného obličeje od původních zjistíme, jak se od sebe liší. Tedy

$$\Phi_i = \Gamma_i - \Psi \tag{4.5}$$

Z takto vypočítaných vektorů Φ_i můžeme určit kovarianční matici C:

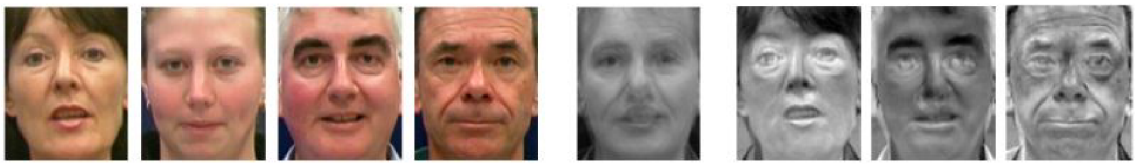
$$C = \frac{1}{M} \sum_{i=1}^{M} \Phi_i \Phi_i^T = AA^T, \tag{4.6}$$

kde matice $A = [\Phi_1 \Phi_2 \ldots \Phi_M]$ má rozměr $N \times M$. Matice AA^T má tedy dimenzi N^2, což však bývá příliš mnoho (např. pro obrázky o rozměrech 256×256 pixelů má dimenzi 65536). Z ní potřebujeme spočítat vlastní vektory, což by však bylo výpočetně velice náročné. Obličejů ale typicky bývá daleko méně, než je dimenze prostoru ($M \ll N$). Označme v_i vlastními vektory matice $A^T A$, jejíž dimenze je pouze $M \times N$. Potom $(A^T A)v_i = \lambda_i v_i$, kde λ_i jsou vlastní čísla. Z toho $A(A^T A)v_i = A(\lambda_i v_i)$, což znamená $(AA^T)(Av_i) = \lambda_i(Av_i)$ a Av_i jsou vlastní vektory matice $C = AA^T$. Nemusíme tedy vlastní vektory kovarianční matice počítat přímo z ní, ale můžeme použít o mnoho menší matici $A^T A$, která má stejná vlastní čísla. Výsledné vlastní vektory u_i matice C jsou tedy

$$u_i = Av_i = [\Phi_1 \Phi_2 \ldots \Phi_M] \begin{bmatrix} v_1^i \\ v_2^i \\ \vdots \\ v_M^i \end{bmatrix} = \sum_{k=1}^{M} v_k^i \Phi_k \tag{4.7}$$

Dále z toho plyne jeden důležitý závěr, že vlastních vektorů příslušejícím kladným vlastním hodnotám je $M - 1$.

Eigenfaces nejsou nic jiného, než znormalizované vlastní vektory, jejichž hodnoty převedeme do intervalu $\langle 0, 255 \rangle$ tak, aby mohly být zobrazeny. Obrázek 4.2 ukazuje pro 4 vstupní obrázky obličejů průměrný obličej a všechny 3 eigenfaces.

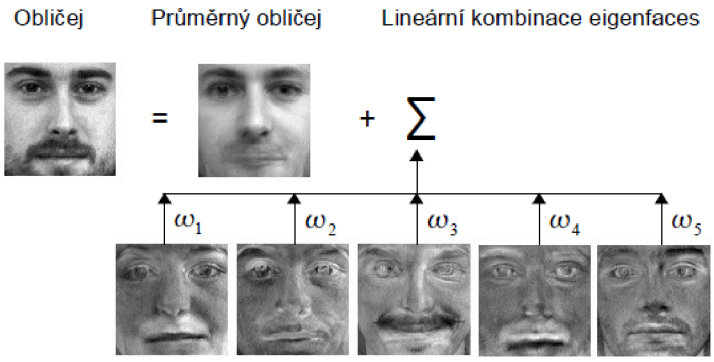


Obrázek 4.2: Vstupní obrázky obličejů (vlevo), průměrný obličej (uprostřed) a eigenfaces (vpravo).

Každý vstupní vektor Γ můžeme promítnout do redukovaného prostoru obličejů *(face-space)* dle následujícího vzorce:

$$\omega_k = \frac{u_k^T(\Gamma - \Psi)}{\lambda_k}, \quad \text{pro } k \ = 1, \ldots, M'. \tag{4.8}$$

K projekci nemusíme použít všechny vektory, proto $M' \leq M$. Obvykle se volí pouze několik prvních vlastních tváří. Další už totiž tak dobře nepopisují distribuci dat v původním prostoru a spíše odpovídají šumu. Vektor $\Omega = [\omega_1 \omega_2 \ldots \omega_{M'}]$ obsahuje váhy jednotlivých eigenfaces jakožto míru jejich příspěvku do obrázku tváře. Tento vektor vah používáme jako příznakový vektor při rozpoznávání obličejů. Jak probíhá složení obrázku tváře z průměrného obličeje a eigenfaces je názorně ukázáno na obrázku 4.3.



Obrázek 4.3: Složení obrázku tváře z průměrného obličeje a vlastních obličejů. Převzato z [30].

Princip metody eigenfaces k identifikaci obličejů je tedy takový, že provedeme analýzu vlastních kompoment na sadě snímků známých obličejů a uložíme si průměrný obličej, vlastní vektory, čísla a příznakové vektory těchto obličejů. Následně, když obdržíme obrázek neznámého obličeje, provedeme jeho projekci do prostoru těchto obličejů, čímž získáme nový příznakový vektor. Dále spočítáme Euklidovské vzdálenosti ke všem uloženým vektorům příznaků Ω_k:

$$\epsilon_k = ||\Omega - \Omega_k||. \tag{4.9}$$

Nalezení příslušejícího obličeje je otázkou nalezení nejbližšího souseda mezi těmito vektory, tedy vektoru s nejmenší euklidovskou vzdáleností. [4, 31, 8]

Kapitola 5

Návrh systému pro rozpoznávání obličeje ze sekvence snímků pomocí RNN

Praktická část diplomové práce je věnována rozpoznávání obličeje ze sekvence obrázků pomocí Elmanovy jednoduché rekurentní sítě (kap. 2.5). Inspirace vzešla z článku „*Tracking and Recognition of Face Sequences*“ již popsaného v kapitole 3.3. Nápad prezentovaný jeho autory nás zaujal, a tak jsme se rozhodli na něj navázat. Experimenty popsané v původním článku nejsou nijak rozsáhlé, byly totiž prováděny pouze na velmi malé testovací sadě. Podrobností o implementaci zde také není příliš mnoho, což jsou další z důvodů, proč se tímto tématem dále zabývat.

V této kapitole je popsán princip metody. Celý postup je rozdělen do tří fází – získání trénovacích dat, trénování a nakonec samotné rozpoznávání.

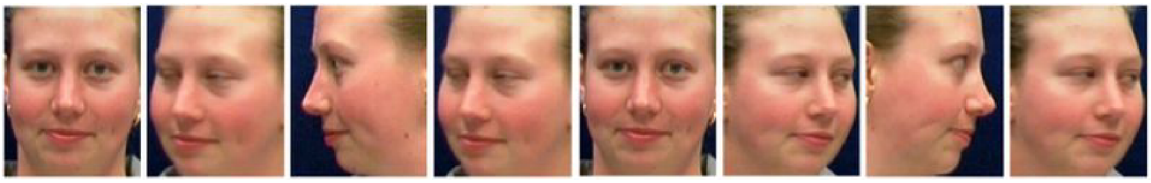
5.1 Princip metody

Většina běžně používaných metod pro identifikaci obličejů využívá pouze jednoho snímku nebo několika snímků nezávisle na sobě. Námi navržený systém využívá celou sekvenci obrázků (viz. ukázka na obrázku 5.1) a hledá v nich temporální příznaky. Díky tomuto přístupu by teoreticky mohl dosahovat lepších výsledků. Ve více obrázcích lze totiž najít více směrodatných informací než v jednom. Navíc můžeme využít souvislosti mezi jednotlivými snímky. Princip metody si popíšeme v bodech, které budou dále popsány podrobněji.

1. Pořídíme videonahrávky lidských obličejů vykonávajících předem určený pohyb.
2. Vybereme z nich několik snímků s určitým natočením hlavy a detekujeme oblast obličeje.
3. Nad obrázky provedeme PCA analýzu (eigenfaces) a získáme tak příznakové vektory. Takto si postupně vytvoříme celou trénovací množinu.
4. Pro každého jedince natrénujeme rekurentní neuronovou síť schopnou predikovat následující snímek v sekvenci.
5. Při rozpoznávání dostaneme sekvenci snímků neznámé osoby. Příznakové vektory získáme projekcí těchto obrázků do prostoru známých obličejů.

6. Tyto vektory postupně předložíme všem natrénovaným sítím.

7. Předpokládáme, že síť naučená konkrétně pro rozpoznávanou osobu, bude predikce schopna lépe než sítě ostatní. Takto můžeme identifikovat neznámého jedince.



Obrázek 5.1: Ukázka sekvence obrázků obličejů.

5.2 Sady trénovacích a testovacích dat

K provádění experimentů je nutné si pořídit vhodná testovací a trénovací data. Na internetu je k dispozici spousta volně dostupných databází obličejů. Bohužel pouze málo z nich obsahuje jejich sekvence, které by navíc byly použitelné pro tuto práci. Jednou z nich je databáze VidTIMIT.

5.2.1 Databáze VidTIMIT

Plným názvem *„The VidTIMIT Audio-Video Dataset“*[1] je databáze audiovizuálních nahrávek od 43 dobrovolníků (z toho 19 žen a 24 mužů). Vznikla na Griffith University v Queenslandu v Austrálii. Natáčení probíhalo na tři části, od každého jedince jsou zde tedy k dispozici tři záznamy. Mezi první a druhou části byla pauza přibližně 7 dní, mezi druhou a třetí asi 6 dní. Díky tomu můžeme pozorovat změny v hlase, oblečení, či líčení jednotlivých lidí. Nastavení přiblížení kamery bylo v průběhu také lehce náhodně měněno. Všechny sekvence zachycují pohyb hlavy zepředu doprava, doleva, zpět na střed, nahoru a dolů. Během toho lidé odříkávají naučené věty. Jelikož se tato práce nezabývá zpracováním řeči, můžeme audio složku ignorovat. Průměrná sekvence je dlouhá 4,25 s a je složena ze 106 snímků.

Pozadí scény je homogenní a má modrou barvu. Osvětlení proběhlo běžnými zářivkami a jednou lampou přímo před snímanou osobou. Všechny světelné zdroje byly z důvodu rozptýlení světla překryty bílým kancelářským papírem. Videa jsou uložena jako sekvence očíslovaných obrázků ve formátu JPG s rozlišením 512×384 pixelů (na šířku) a 90% úrovní komprese. Na obrázku 5.2 je výběr z jedné z obsažených videosekvencí. [27]

5.3 Fáze učení

Tato podkapitola popisuje jednotlivé části trénovací fáze algoritmu. Schéma můžete nalézt na obrázku 5.5.

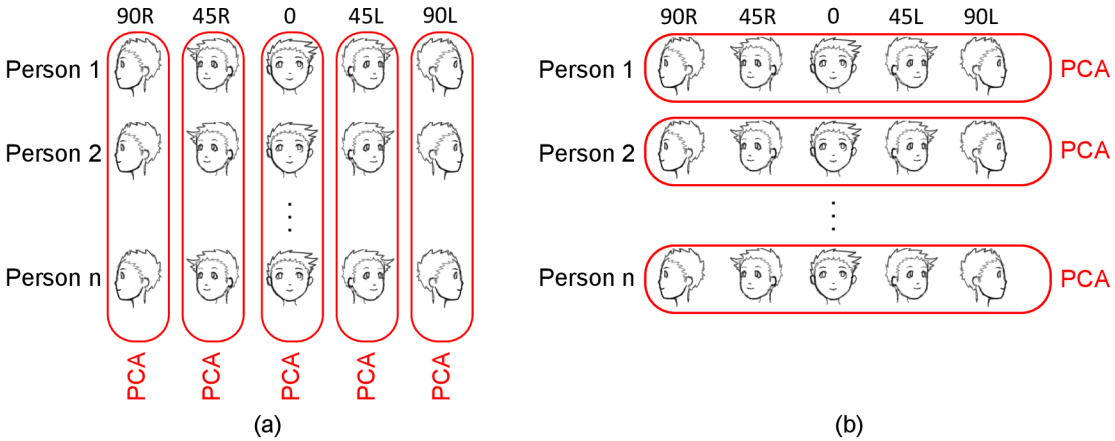
[1] http://archive.itee.uq.edu.au/~conrad/vidtimit/

Obrázek 5.2: Výběr snímků jedné ze sekvencí databáze VidTIMIT.

5.3.1 Extrakce příznakových vektorů

Výběr odpovídajících snímků z videí a lokalizace obličejů probíhala ručně. Ukázku několika vybraných snímků najdete v příloze A. K tomuto účelu byl použit nástroj ViPER–GT (kap. 6.5). Následně byly všechny obrázky normalizovány zmenšením na stejnou velikost a upravením kontrastu ekvalizací histogramu. Pro získání příznaků z obrázků byla zvolena technika **eigenfaces**. Zde existují dva možné přístupy.

První z přístupů kopíruje zmíněný článek. Tedy že PCA analýza byla prováděna pro snímky obličeje jedné osoby, **nezávisle na směru pohledu**. Pokud tedy máme k dispozici data od 10 osob a pro každou z nich 5 sekvencí o 5 snímcích, bude analýza probíhat celkem 10x.



Obrázek 5.3: Dva možné způsoby, jak provádět PCA analýzu. (a) V závislosti na směru pohledu, (b) nezávisle na směru pohledu, pro každou osobu zvlášť.

Takovéto použití metody vlastních obličejů je poměrně netradiční. Eigenfaces totiž popisují rysy společné pro všechny zkoumané objekty. Rozumnější se tedy zdá rozdělit množinu všech dostupných obrázků **dle směru pohledu** a provést PCA zvlášť pro každou jednu podmnožinu. Pro stejnou datovou sadu jako v předchozím případě zde analýza proběhne pouze 5x, jelikož máme 5 rozdílných směrů pohledu. I tento způsob byl vyzkoušen. Rozdíl mezi oběma přístupy je demonstrován na obrázku 5.3.

Pro každou osobu a každý snímek obdržíme z PCA analýzy jeden příznakový vektor. U snímků známe směr pohledu, takže je můžeme seřadit do sekvence. Z pěti vybraných obrázků vytvoříme následující sekvenci o 8 obrázcích:

$$\texttt{0} \rightarrow \texttt{45R} \rightarrow \texttt{90R} \rightarrow \texttt{45R} \rightarrow \texttt{0} \rightarrow \texttt{45L} \rightarrow \texttt{90L} \rightarrow \texttt{45L}$$

Čísla zde značí úhel a `R`, resp. `L` pravou, resp. levou stranu (z pohledu natočené osoby). Ukázka takové sekvence je na obrázku 5.1. Jde samozřejmě jen o ilustraci, ve skutečnosti nejsou použity samotné obrázky, ale jim odpovídající příznakové vektory. Vycházíme z toho, že v případě reálného nasazení implementovaného rozpoznávacího systému by uživatel přistoupil čelem ke kameře a následně provedl pohyb hlavy doprava, poté doleva a nakonec by zase skončil pohledem do kamery. Sekvence extrahovaných příznakových vektorů nám tvoří naši trénovací množinu.

5.3.2 Umělé rozšíření datové sady

Jelikož máme k dispozici pouze tři nahrávky od každé osoby, je možné si datovou sadu uměle rozšířit. Tedy ručně vybrané snímky z těchto sekvencí „rozhýbat“ a vytvořit tak další, mírně posunuté v čase a v prostoru. U každého obrázku známe jeho pořadí a oblast obličeje. Není nic jednoduššího, než během generování trénovacích dat do trénovací sady zahrnout i obrázky s trochu jinak natočeným a detekovaným obličejem. Zvolenou oblast tváře můžeme posunout horizontálně, vertikálně, ale taktéž ji lze roztáhnout, či změnšit. Příklad takto uměle vytvořených snímků je na obrázku 5.4.

Obrázek 5.4: Uměle vytvořené (posunuté) snímky obličeje – úplně vlevo je originální snímek, zbytek obrázků byl dogenerován.

5.3.3 Normalizace vstupů

Surové vstupní vektory, které obdržíme z PCA analýzy, je nutné ještě před předložením neuronové síti normalizovat do určitého intervalu. V našem případě je to ideálně $\langle 0.1, 0.9 \rangle$ [10]. Bylo vypozorováno, že hodnoty všech položek příznakových vektorů mají přibližně normální (Gaussovo) rozdělení pravděpodobnosti. Proto můžeme využít pravidlo tří sigma. To nám říká, že pravděpodobnost, že se hodnota náhodné veličiny bude od střední hodnoty lišit o trojnásobek směrodatné odchylky, je vetší nebo rovna 99 %. Spočítáme tedy směrodatnou odchylku:

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2} \tag{5.1}$$

kde $\overline{x}$ je aritmetický průměr

$$\overline{x} = \frac{1}{N}\sum_{i=1}^{N} x_i. \tag{5.2}$$

Následně můžeme provést normalizaci hodnot:

$$a' = \frac{a - min_x}{|max_x - min_x|} * (h - l) + l, \tag{5.3}$$

kde $min_x = \overline{x} - 3\sigma$, $max_x = \overline{x} + 3\sigma$, l je spodní a h horní hranice zvoleného intervalu. Hodnoty min_x a max_x je nutné si uložit pro použití v rozpoznávací fázi. Položky vstupních vektorů zde totiž budeme opět normalizovat stejným způsobem v závislosti na trénovacích datech.

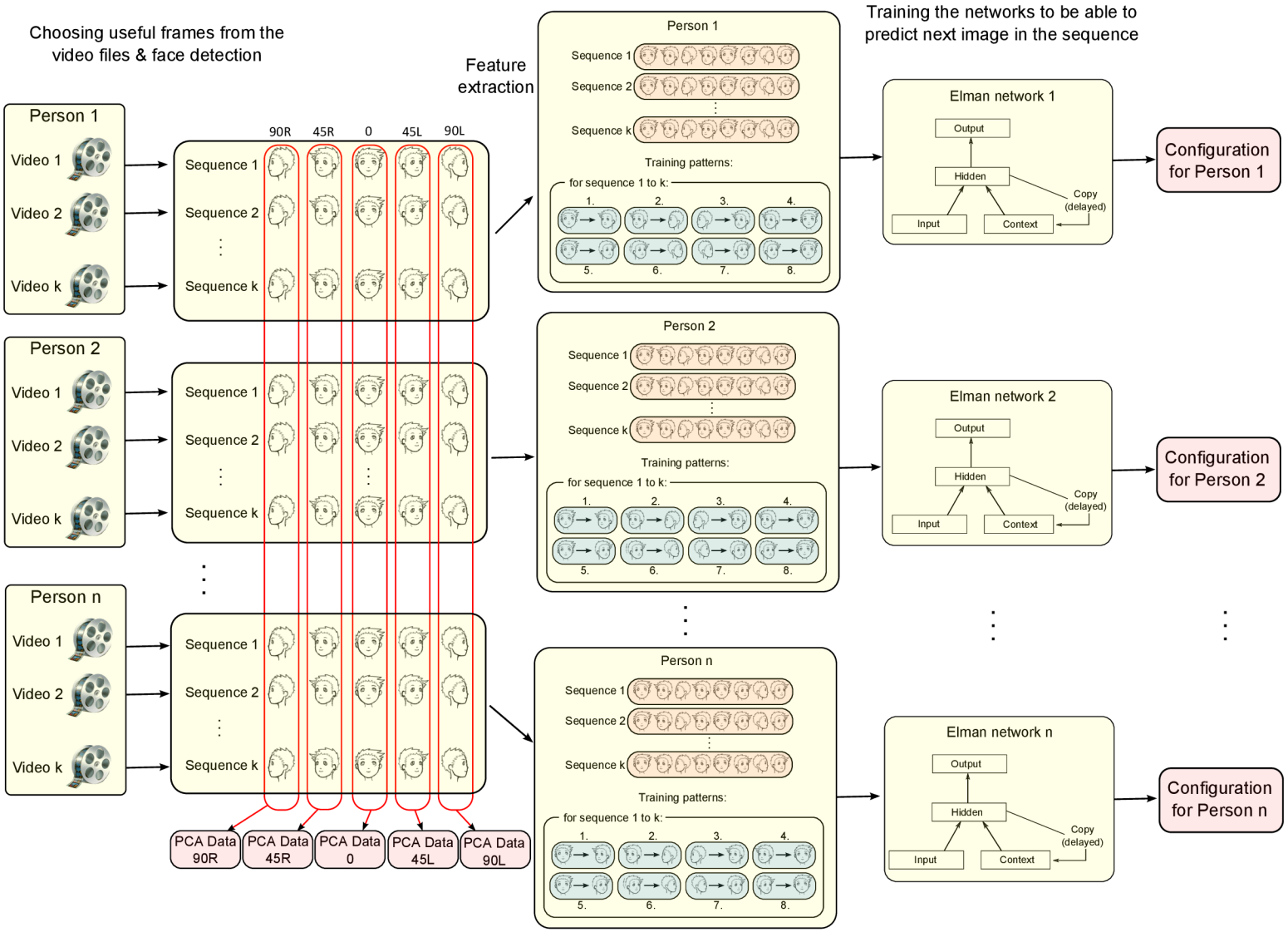
5.3.4 Trénování neuronových sítí

Elmanova síť nám slouží pro predikci dalšího snímku v sekvenci. Trénovací sada se tedy skládá z dvojic – vstupu a požadovaného výstupu sítě, kde vstupem je vždy příznakový vektor jednoho snímku a výstupem vektor snímku následujícího.

Elmanovu rekurentní neuronovou síť lze učit buď klasickým algoritmem backpropagation nebo algoritmem backpropagation through time. Oba již byly popsány v kapitole 2. Samozřejmě musíme mít na paměti kontinualitu vstupů. U dopředné sítě na pořadí trénovacích vzorů v trénovací množině nezáleží. U Elmanovy RNN je ale výstup v jednom okamžiku závislý i na předchozím vstupu, proto přikládáme vektory vždy ve správném pořadí.

Díky tomu, že nám poslední snímek v sekvenci predikuje opět ten první, můžeme u trénování pomocí BP každou sekvenci vstupních/výstupních vektorů v trénovací množině několikrát cyklicky zopakovat. V případě BPTT po posledním vzoru každé sekvence nesmíme zapomenout vymazat obsah kontextové vrstvy. Porovnání obou variant trénovacího algoritmu bude provedeno v kapitole 7.

Jak již bylo zmíněno dříve, není nutné a většinou ani příhodné, používat příznakový vektor celý. Bereme tedy v úvahu pouze jeho několik prvních prvků. Empiricky bylo zjištěno, že ideální je použít jich přibližně 10 %.



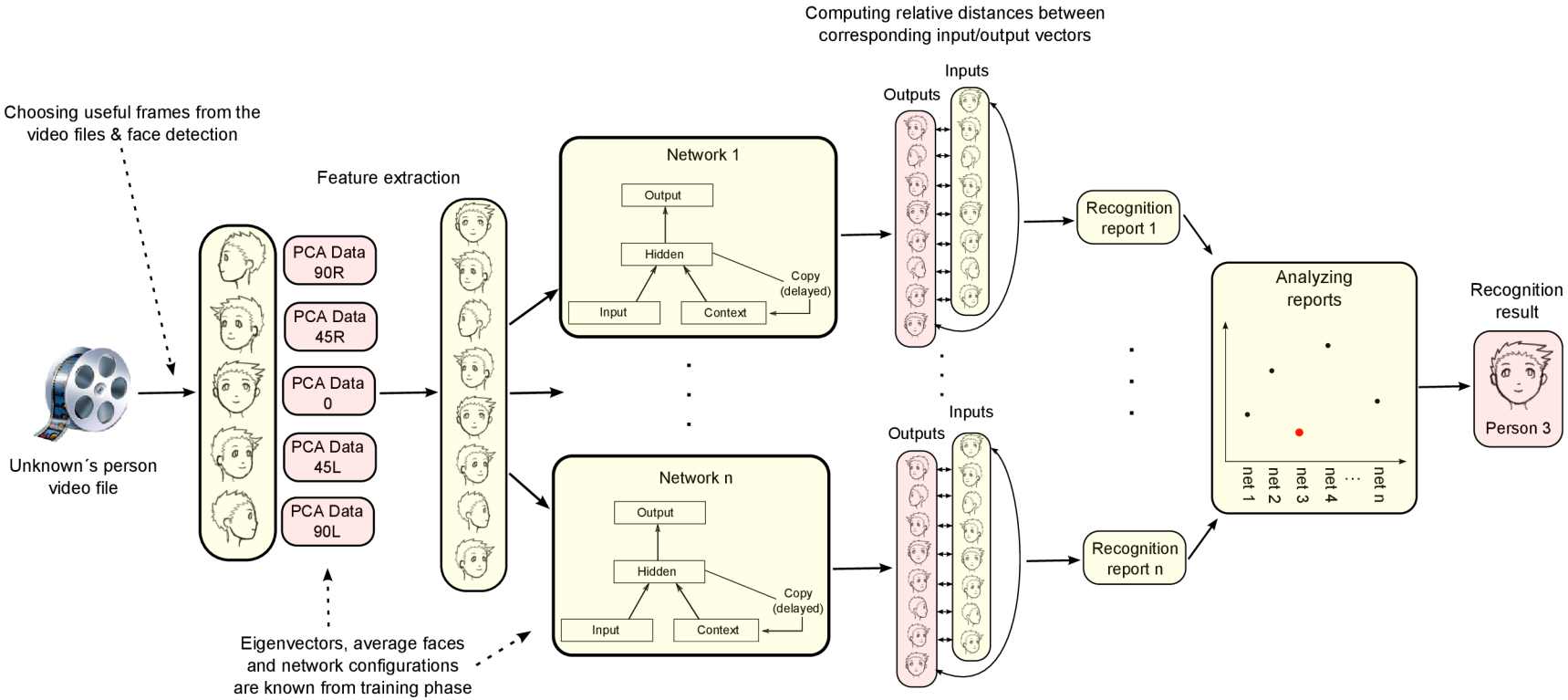
Obrázek 5.5: Schéma trénovací fáze algoritmu.

5.4 Fáze rozpoznávání

Z trénovací fáze máme uloženy výsledky PCA analýzy, tedy průměrné obličeje a vlastní vektory, konfigurace neuronových sítí a minimální a maximální hodnoty příznakových vektorů pro normalizaci vstupů. Narozdíl od klasické metody eigenfaces nebudeme během rozpoznávání potřebovat váhy jednotlivých vlastních obličejů osob z trénovací množiny.

Na vstupu opět obdržíme videonahrávku pohybu hlavy od neznámého člověka. Naším cílem je určit, která osoba z původní množiny je na ní zachycena. Vybereme tedy 5 snímků s potřebným směrem pohledu a detekujeme v nich obličeje. Obrázky je samozřejmě nutné zmenšit, resp. zvětšit na určenou konstantní velikost a provést ekvalizaci histogramu k vyrovnání konstrastu. Z obrázků složíme sekvenci 8 snímků, vše stejně jako při trénování. Příznakové vektory obdržíme projekcí obrázků do redukovaného prostoru obličejů – viz rovnice 4.8. Připomeňme, že pokud proběhla PCA analýza v závislosti na směru pohledu, budeme projekci provádět pouze 5x a stejné příznaky budou použity pro všechny sítě. Pokud jsme ale PCA dělali nezávisle na směru pohledu, tedy pro každou osobu zvlášť, budeme muset zvlášť provést i projekci pro každého jedince, resp. „jeho“ síť. Po normalizaci jsou vstupní vektory postupně předkládány všem neuronovým sítím a jsou zaznamenávány jejich výstupy.

V původním článku autoři samotnou identifikaci osoby prováděli tak, že porovnali průběhy Euklidovských vzdáleností mezi dvěma po sobě jdoucími vstupy a průběhy vzdáleností mezi dvěma výstupy. Udávají, že u sítě natrénované na datech rozpoznávané osoby jsou tyto průběhy nejvíce podobné (obr. 3.6).Dalším možným způsobem je sledovat vzdálenosti mezi výstupním vektorem a korespondujícím vstupem. Tedy pokud nám síť pro pohled 45° vpravo predikuje směr 90° vpravo, porovnáme vzdálenosti aktuálního vstupu a předchozího výstupu. Předpokládáme, že síť našeho neznámého jedince bude predikci provádět nejlépe, a tedy vzdálenosti mezi vektory budou nejmenší. Obě varianty budou opět zkoumány v kapitole 7. Schéma celé rozpoznávací fáze naleznete na obr. 5.6.



Obrázek 5.6: Schéma rozpoznávácí fáze algoritmu.

Kapitola 6

Implementace

6.1 Knihovna RNNFlib

Výsledkem praktické části diplomové práce je především knihovna RNNFlib – *Recurrent Neural Networks based Face Recognition Library.* Obsahuje třídy pro přípravu trénovacích dat z videí anotovaných prostřednictvím nástroje ViPER (viz. dále), tvorbu sekvencí obličejových snímků, předzpracování obrázků a extrakci příznaků. Zahrnuje dva modely Elmanovy RNN – jeden z externí knihovny Encog s učícím algoritmem backpropagation a jeden vlastní, jež k trénování používá algoritmus backpropagation through time. Jelikož se jedná o experimentální projekt, je zde i podpora pro tvorbu reportů s výsledky rozpoznávání s možností exportu do CSV formátu a automatickým generováním grafů. Ke generování grafů byl použit program gnuplot[1]. Samozřejmostí je logování průběhu jednotlivých algoritmů včetně délky jejich trvání pro ulehčení analýzy programů apod.

Knihovna byla implementována v jazyce C# v prostředí .NET Framework 4 za využití knihoven Encog a Emgu CV. Podrobnější informace lze nalézt v programové dokumentaci na přiloženém DVD.

6.2 Programy RNNTrainDataCreator, RNNFaceTrainer a RNNFaceRecognizer

Všechny tři tyto programy jsou založeny na knihovně RNNFlib. Jak již napovídá samotný název, **RNNTrainDataCreator** slouží k tvorbě trénovacích dat. Na vstupu jsou videa se záznamy pohybů hlavy a soubor s metadaty z Viper–GT toolkitu. Na výstupu potom sekvence předzpracovaných obrázků obličejů. Program **RNNFaceTrainer** byl vytvořen k trénování rekurentních neuronových sítí. Ze vstupních dat vytvoří trénovací sadu a natrénuje sítě prostřednictvím algoritmu BP nebo BPTT. Všechna potřebná data dokáže uložit pro pozdější použití v programu **RNNFaceRecognizer**. Ten se stará o samotné rozpoznávání. Ze sekvencí snímků od neznámých osob extrahuje příznaky, předloží je naučeným sítím a analyzuje výsledky rozpoznávání. Zároveň generuje reporty ve formátu CSV a grafy ve formátech PNG a EPS. Do grafické podoby lze převést i výstupy sítí. Každému neznámému jedinci je přiřazen nejpravděpodobnější, resp. nejpodobnější kandidát z původní databáze známých osob.

Ve všech třech případech se jedná o konzolové aplikace. Nastavení všech parametrů probíhá prostřednictvím konfiguračních souborů.

[1] http://www.gnuplot.info/

6.3 Knihovny OpenCV a Emgu CV

OpenCV[2] je open source knihovna pro práci s obrazem původně vyvinuta společností Intel. Je zaměřena především na počítačové vidění *(CV – Computer Vision)*. Je napsaná v jazyce C a C++, což znamená, že běží pod operačními systémy Windows, Linux a Mac OS X. Existují i rozhraní pro jazyky Python, Ruby aj. Velký důraz je kladen na real-time aplikace a díky optimalizacím dokáže využít předností vícejádrových procesorů. Nabízí přes 500 funkcí pro zpracování obrazu, detekci objektů, strojové učení, práci s kamerou a mnoho dalších. Dnešní aktuální verze je 2.2. [3]

Protože je naše aplikace napsána v jazyce C#, bylo nutné použít nějaký wrapper knihovny OpenCV pro .NET Framework. Jedním z nich je EmguCV. Umožňuje volání funkcí OpenCV z prostředí .NET, přičemž podporovanými jazyky jsou C#, Visual Basic, Visual C++, IronPython a další. Knihovna může být zkompilována v prostředí frameworku Mono[3], což ji činí spustitelnou i pod operačními systémy Linux a Mac OS X. Použití EmguCV přináší spoustu výhod, zejména objektově orientovaný přístup programování, garbage collector a propojení se všemi funkcemi MS .NET Frameworku. Kromě možnosti volání nativních funkcí OpenCV jsou zde implementovány dvě základní třídy – `Image` a `Matrix`, nabízející plné využití předností technologie .NET. [7]

6.4 Encog Java and DotNet Neural Network Framework

Encog je pokročilý framework pro práci s neuronovými sítěmi a strojové učení. Od roku 2008 je vyvíjena společností Heaton Research Inc., v jejímž čele stojí p. Jeff Heaton. Poskytuje třídy k vytvoření mnoha druhů neuronových sítí s nejrůznějšími typy trénovacích algoritmů, stejně tak jako třídy pro zpracování, normalizaci, ukládání a načítání dat. Pro zvýšení výkonu je zde možnost využít k výpočtům jazyk OpenCL a procesor grafické karty. Existují verze pro Javu, .NET Framework a Silverlight. Pro snadnější modelování a trénování neuronových sítí je zde i grafické uživatelské rozhraní. Knihovna je dobře dokumentovaná, dodávaná se spoustou názorných příkladů. Díky aktivnímu vývoji má i slušnou technickou podporu na stránkách projektu. Je šířitelná pod licencí LGPL. Jediné, co zde pro naše účely schází, je implementace algoritmu BPTT. [10]

6.5 ViPER: Ground Truth Editor

The Video Performance Evaluation Resource Kit's Ground Truth tool nebo zkráceně *ViPER–GT* [18] je grafický nástroj pro anotaci videa. Video lze označkovat metadaty. Kromě data pořízení videa nebo klíčových slov popisujících jeho obsah lze označovat i jednotlivé snímky nebo oblasti v nich. Uživatel si může vytvořit vlastní schéma, skládající se z tzv. deskriptorů. Deskriptorem může být např. obličej. Ten je pak tvořen několika atributy. V případě obličeje to může být jméno osoby, obdélník ohraničující oblast obličeje, směr natočení hlavy, poloha očí apod. K dispozici je několik geometrických atributů (bod, obdélník, kružnice, elipsa, polygon) a několik nevizuálních (řetězec, číslo, boolovská hodnota nebo výčtový typ). Desktiptory potom lze označit samostatné snímky nebo jejich sekvence.

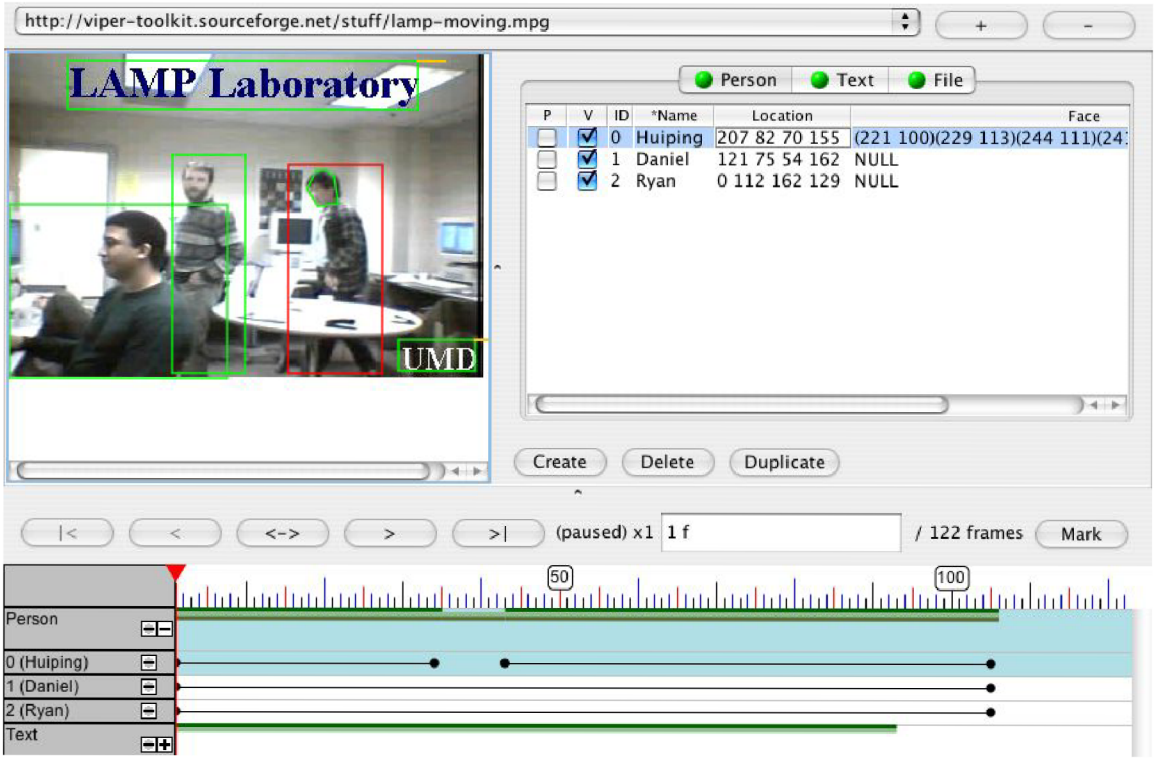
Práce s tímto nástrojem je velice snadná, uživatelské rozhraní je poměrně přívětivé. Na obrázku 6.1 je ukázka ze samotné aplikace během anotování. Program je napsán v jazyce

[2] http://opencv.willowgarage.com/
[3] http://www.mono-project.com

Java.

Produktem anotace je soubor ve formátu XML s metadaty dle vytvořených schémat. Následuje ukázka části kódu popisujícího jeden snímek s obličejem. Prezentované schéma bylo použito v této práci.

```
<object framespan="1:1 27:27 82:82 188:188 243:243" id="0" name="FACE">
  <attribute name="RECT">
      <data:bbox framespan="1:1" height="226" width="207" x="154" y="92"/>
      <data:bbox framespan="27:27" height="220" width="200" x="143" y="97"/>
      <data:bbox framespan="82:82" height="216" width="197" x="102" y="104"/>
      <data:bbox framespan="188:188" height="222" width="204" x="162" y="92"/>
      <data:bbox framespan="243:243" height="217" width="198" x="213" y="100"/>
  </attribute>
  <attribute name="DIR">
      <data:lvalue framespan="1:1" value="0"/>
      <data:lvalue framespan="27:27" value="45R"/>
      <data:lvalue framespan="82:82" value="90R"/>
      <data:lvalue framespan="188:188" value="45L"/>
      <data:lvalue framespan="243:243" value="90L"/>
  </attribute>
</object>
```



Obrázek 6.1: Ukázka z aplikace ViPER–GT. Převzato z [18].

Kapitola 7

Testování a výsledky

V této kapitole si uvedeme výsledky provedených experimentů a srovnáme různé varianty jednotlivých částí navrženého systému. Mimo jiné mezi sebou porovnáme algoritmy backpropagation a backpropagation through time, zhodnotíme dva možné přístupy, jakými lze provádět PCA analýzu a porovnáme náš systém využívající rekurentních neuronových sítí s běžným rozpoznáváním založeným na eigenfaces a nalezení nejbližšího souseda. Přehled výsledků vybraných testů je prezentován v podkapitole 7.7.

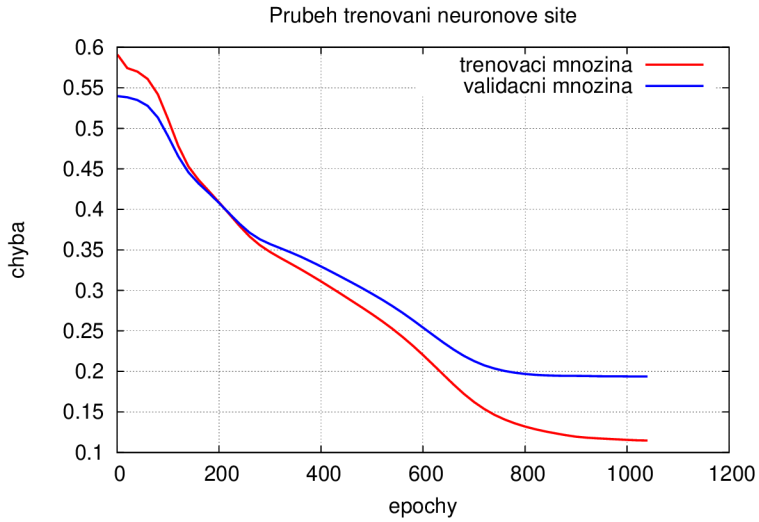
7.1 Průběh trénování

Jak jsme uvedli výše, z databáze VidTIMIT jsme měli k dispozici 3 videonahrávky od každé ze 43 osob. Ve všech testech byly pro účely trénování použity 2 z nich a na zbylé, třetí, se provádělo testování systému. Avšak pouhé dvě trénovací sekvence jsou pro kvalitní učení neuronových sítí poněkud málo. Proto jsme využili dvou způsobů rozšíření trénovací množiny. Jeden z nich byl popsán v kapitole 5.3.2. Jedná se o jakési „rozhýbání“ ručně vybraných snímků. Z každé ze dvou sekvencí jsme vytvořily další 3 „umělé“. Tak nám vzniklo celkem 8 trénovacích sekvencí, z nichž některé byly využívány pro crossvalidaci. Stejným způsobem samozřejmě můžeme zvětšit i testovací sadu, čímž lze lépe ověřit schopnost generalizace našeho systému a jeho odolnost vůči posunutým či jinak odlišným vstupním obrázkům. Druhým, častěji používaných způsobem, bylo ze dvou videí ručně vytvořit ne 2, ale 3 trénovací sekvence. Ruční výběr zaručil, že vždy byly vybrány mírně odlišné snímky.

Pro samotné trénování jsme použili Elmanovy rekurentní sítě se sigmoidálními aktivačními funkcemi ve skrytých a výstupních vrstvách. Během trénování byl dynamicky měněn parametr učení (learning rate). Momentum bylo fixně nastaveno na hodnotu $0,9$. Pro ukončení učení jsme stanovili tři podmínky:

1. **maximální počet epoch**
2. **maximální požadovaná chyba** – pokud chyba sítě klesla pod tuto konstantu, trénování bylo ukončeno
3. **minimální přírustek** – pokud byla po několik epoch chyba sítě menší, než stanovená konstanta, učení bylo ukončeno

Velikost parametru BPTT, tedy „paměti“ sítě, byla předmětem testů. Ukázku průběhu chyby sítě (rovnice 2.18) na trénovací a validační množině naleznete na obrázku 7.1. Průměrné počty epoch a časy potřebné k učení budou uvedeny dále.



Obrázek 7.1: Průběh chyby během učení

7.2 Popis testovacích dat

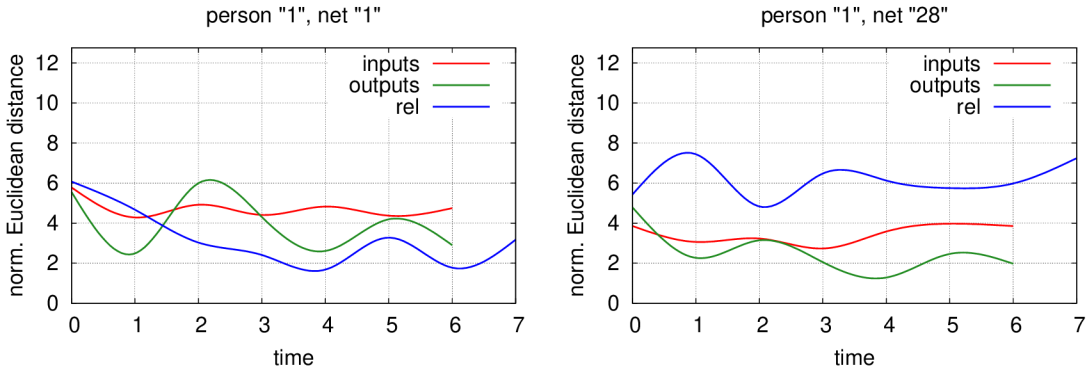
U všech testů jsme používali dvě různé množiny testovacích dat. Abychom to nemuseli u každého konkrétního testu uvádět znovu, popišme si zde, jak testování probíhalo. V prvním případě byla v testovací sadě pouze 1 sekvence obrázků pro každou osobu. Obrázky byly z videonahrávek vybrány ručně. V druhém případě jsme z této jedné sekvence „rozhýbáním“ vytvořili další 3, čímž pádem jsme testovací sadu rozšířili na 4 sekvence. Stejně jako u trénovací množiny jsme toto dělali z důvodu lepšího ověření schopnosti generalizace u natrénovaných neuronových sítí. Pokud nebude explicitně uvedeno jinak, byly pro všechny následující testy použity tyto dva druhy testovacích množin. Každý test byl spuštěn 4x a výsledky byly zprůměrovány.

7.3 Analýza výstupů sítí

Jednou z nejdůležitějších částí celého rozpoznávacího systému je zajisté správné vyhodnocení výstupů sítí. Jde tedy o samotnou identifikaci neznámého jedince. Zde se ukázalo, že postup, který použili autoři v referenčním článku [8], je naprosto nevhodný. Způsob autorů aplikovaný na naši trénovací a testovací sadu dosahoval úspěšnosti okolo 0 %.

V grafu 7.2a vidíme průběhy vzdáleností u sítě, která odpovídá rozpoznávané osobě (byla na ni naučena) a v grafu 7.2b potom odezvy sítě, jež byla naterénována na jiném člověku. Dle uvedeného článku by červená křivka (vzdálenosti mezi následujícími vstupy) a zelená křivka (vzdálenosti mezi následujícími výstupy) měly mít podobnější průběh v grafu vlevo. Jak ale vidíme, je tomu přesně naopak. Srovnání průběhu podobností těchto vzdáleností tedy nic nevypovídá o schopnosti predikce, kterou jsme u sítí očekávali. Pro úplnost, modrá křivka v těchto grafech značí vzdálenosti mezi výstupem sítě a jejím předchozím vstupem.

Experimentováním jsme přišli na způsob, jak rozpoznávání provádět o mnoho lépe. Schopnost predikovat následující snímky v sekvenci lze změřit spočtením vzdálenosti mezi výstupem sítě a předchozím vstupem. Měříme tak vzdálenost mezi tím, co očekáváme, a tím,



(a) síť natrénovaná na danou osobu

(b) síť natrénovaná na jinou osobu

Obrázek 7.2: Průběhy vzdáleností mezi vstupem (červně), výstupem sítě (zeleně) a vzdáleností mezi výstupem sítě a jejím předchozím vstupem (modře).

co dostaneme. Jedna sekvence sestává z 8 obrázků, takže získáme 8 relativních vzdáleností. Pro výpočet vzdálenosti mezi dvěma vektory lze využít Euklidovskou metriku, nicméně lepších výsledků jsme dosáhli použitím normalizované Euklidovské vzdálenosti:

$$d(\vec{x}, \vec{y}) = \sqrt{\sum_{i=1}^{n} \frac{(x_i - y_i)^2}{\sigma_i^2}}, \tag{7.1}$$

kde σ je směrodatná odchylka (rov. 5.1). Dle zjištěných vzdáleností musíme vybrat síť s nejlepší predikcí. Nabízí se spočítat sumu nebo aritmetický průměr vzdáleností a vybrat tu síť, kde tato hodnota bude nejmenší. Opět se nám během experimentů osvědčilo použití harmonického průměru:

$$H = \frac{n}{\sum_{i=1}^{n} \frac{1}{n_i}}. \tag{7.2}$$

Celý proces identifikace obličeje z výstupů sítí můžeme popsat následujícími rovnicemi. Pokud I_t, resp. O_t je vstupní, resp. výstupní vektor sítě v čase t, potom vzdálenost mezi výstupy a odpovídajícími vstupy sítě i v čase t zapišme jako

$$d_t(i) = d(\vec{I_t}, \vec{O_{t-1}}). \tag{7.3}$$

Průměrná vzdálenost je jejich harmonický průměr:

$$\overline{d(i)} = H(d_t(i)), \tag{7.4}$$

kde $i = 1, \ldots, N$, $t = 1, \ldots, 8$ a N je celkový počet neuronových sítí. Ze všech hodnot průměrů vybereme identifikovanou osobu R jako

$$R = \arg\min_i \overline{d(i)}. \tag{7.5}$$

Průměr hodnot pro každou ze sítí si můžeme zanést do grafu. V grafu 7.3a vidíme příklad úspěšného rozpoznávání, v grafu 7.3b potom výsledek rozpoznávání chybného.

Projekcí výstupních vektorů do prostoru obličejů lze získat vlastní vektory a ty pak vizualizovat jako obrázky (eigenfaces). Takto můžeme názorně demonstrovat schopnost sítě predikovat další snímek v sekvenci. Na obrázku 7.5 vidíme nahoře obrázky, z nichž byly extrahovány příznakové vektory a uprostřed a dole odezvy sítí na tyto vstupy. Obličeje zobrazené uprostřed nám předpověděla síť, jež byla na tuto osobu natrénována. Obličeje dole odpovídají výstupům sítě naučené na obrázcích jiné osoby.

Zajímavým způsobem, jak se přesvědčit o tom, jak dobře dokáží natrénované sítě předpovídat následující snímky, je cyklicky využít jejich výstup znovu jako vstup. Pokud to takto provedeme 8x, měli bychom z pouhého jednoho vstupního obrázku obržet celou sekvenci. Ukázku vidíme na obr. 7.6.

7.4 Srovnání algoritmů BP a BPTT

V tomto testu porovnáme dva použité trénovací algoritmy, tedy klasický backpropagation a backpropagation through time. Implementaci Elmanovy sítě a BP nám nabízí knihovna Encog. Algoritmus BPTT byl implementován v rámci diplomové práce a je součástí knihovny RNNFLib.

Srovnávat můžeme dle dvou kritérií – úspěšnosti rozpoznávání a rychlosti učení. Co se týče úspěšnosti, pokud byla síť dostatečně velká, dával nám klasický BP lepší výsledky. Pro sítě s menším počtem neuronů se nicméně ukázal výhodnější algoritmus BPTT. V rychlosti trénování jednoznačně vede BP. Je to samozřejmě dáno principielně – během učení zpětným šíření chyby v čase je potřeba daleko více výpočtů. Velkou roli zde hraje i to, že námi implementovaný algoritmus, narozdíl od BP z knihovny Encog není příliš optimalizovaný. Encog např. dokáže učení provádět paralelně ve více vláknech.

V tabulce 7.1 vidíme konkrétní výsledky všech testů. V testech 1. – 8. byl použit trénovací algoritmus BPTT. Klasický BP byl použit v případě testů 9. – 12. (tj. tam, kde je hodnota parametru $\tau = 0$). Vidíme zde také vliv počtu neuronů ve vstupní a skryté vrstvě a velikosti parametru τ. Poslední zmíněný parametr nám udává, kolik kroků do minulosti si síť uchovává kontext během trénovací fáze.

Největší úspěšnosti 95,35 % (tedy 41 ze 43 rozpoznaných osob) jsme dosáhli v případě, kdy PCA analýza probíhala v závislosti na směru pohledu, dále 30 ze všech 128 eigenvektorů bylo použito k trénování, ve skrytých vrstvách jsme měli 45 neuronů a sítě jsme učili algoritmem backpropagation. Trénování na 3 sekvencích si průměrně vyžádalo 396 epoch a zabralo 8 min 40 s. Se sítěmi se stejnými parametry, ale trénovanými pomocí BPTT s $\tau = 5$, jsme docílili velice podobné úspěšnosti, a to 93,02 %. Zde bylo ovšem průměrně potřeba 1 587 epoch a učení zabralo více, než 4x delší dobu, tedy 35 min 53 s.

U varianty s crossvalidací (zopakujme, že v tomto případě byly použity i uměle vytvořené sekvence) jsme dosáhli maximální úspěšnosti 88,37 %. Bylo to opět s algoritmem BP. Vstupní vrstvy obsahovaly 34 neuronů, skryté pak 51. K učení bylo průměrně potřeba 269 epoch a trvalo 11 min 18 s. Algorimus BPTT tady s úspěšností 87,79 % zůstává jen lehce pozadu. Doba potřebná pro natrénování takovýchto sítí ovšem byla opět mnohonásobně větší – celých 48 min 13 s.

Z provedených testů tedy vyplyvá, že pro trénování Elmanových sítí s dostatečným počtem neuronů v našem případě stačí klasický algoritmus BP. Dosahuje lepší úspěšnosti i rychlosti. Mimo jiné se ukázalo, že u BPTT při vyšší hodnotě τ dochází ke konvergenci chyby rychleji. Pokud vhodně rozšíříme trénovací množinu, stačí nám k učení pouze 2 videosekvence od každého jedince.

7.5 Vliv typu dat použitých pro PCA

Krátce zhodnoťme i dvě varianty, jakými lze provádět PCA analýzu. Prvním způsobem, převzatým z referenčního článku, je analyzovat vždy najednou obrázky od jednoho člověka nezávisle na směru pohledu. Druhým způsobem je celou datovou sadu rozdělit dle směru pohledu (v našem případě je to pět směrů) a na každé takové podmnožině provést PCA. Už při prvotní analýze problému se nám druhá z variant jevila jako příhodnější. A testy naši domněnku potvrdily. V tabulce 7.1 si můžete prohlédnout výsledky testů 5., 6. a 10., kdy při analýze nebyl úhel pohledu zohledněn. Nejlepší úspěšnosti, jaké jsme dosáhli, byla 65,12 % u algoritmu BPTT. Naproti tomu, když jsme úhel pohledu vzali v úvahu, dosáhli jsme průměrné úspěšnosti až 95,35 %. Jak vypadají průměrné obličeje v případě obou variant PCA, se můžete podívat na obrázku 7.4.

Obrázek 7.4: Průměrné obličeje dvou variant PCA. Vlevo probíhala analýza v závislosti na směru pohledu, vpravo nikoliv.

7.6 Srovnání s běžným rozpoznáváním založeným na eigenfaces

Abychom zjistili, jak je na tom náš systém v porovnání s jinými způsoby rozpoznávání, vyzkoušeli jsme naše data podrobit rozpoznávání pomocí klasických eigenfaces.

PCA analýzu jsme provedli v závislosti na směru pohledu. Tím jsme dostali 5 různých obličejových prostorů. Stejně jako u prezentovaného systému využívajícího RNN jsme každý z pěti vybraných snímků sekvence promítli do příslušného prostoru obličejů, čímž jsme získali příznakové vektory. Pro každý z nich byl potom nalezen nejbližší soused (pomocí euklidovské metriky) z databáze a ten byl označen jako rozpoznávaná osoba. Pokud byl výsledek správný alespoň pro 3 z 5 obličejů, vyhodnotili jsme tuto jednu konkrétní sekvenci jako správně rozpoznanou.

Testování probíhalo na stejných datech jako u RNN systému. V případě, kdy byly v databázi 3 sady obrázků od každého jedince, vykazovalo rozpoznávání založené na eigenfaces úspěšnosti 90,7 %. Ve chvíli, kdy jsme do databáze obličejů přidali i „umělé“ obrázky (a testovali na 4 sekvencích), docílili jsme úspěšnosti dokonce 95,35 %. Na menší trénovací sadě tedy vítězí náš systém s úspěšností 93,02 %. Na větší, rozšířené sadě, má potom klasické rozpoznávání pomocí klasických eigenfaces stejnou úspěšnost jako systém využívající RNN.

7.7 Přehled dosažených výsledků

Kombinací všemožných parametrů, kterými lze rozpoznávání ovlivnit, je velké množství. V následující tabulce uveďme přehled výsledků vybraných testů, které jsme provedli. Většina z nich byla okomentována v rámci této kapitoly.

Testování probíhalo na počítači s procesorem Intel Core i5-450M 2.4 GHz s 4GB RAM a operačním systémem MS Windows 7 s .NET Frameworkem verze 4.

č. testu	záv. na směru pohledu	eigenvektorů	trénovacích. sekv. / os.	validačních sekv. / os.	testovacích sekv. / os.	vstupních neuronů	skrytých neuronů	BPTT (τ)	Ø úspěšnost	medián epoch trén.	Ø epoch trénování	Ø čas trén. [mm:ss]
1.	záv.	128	3	0	1	13	20	3	89,53 %	2393	2361,1	35:51
2.	záv.	128	3	0	1	13	20	5	91,28 %	1568,5	1590,6	23:25
3.	záv.	128	3	0	1	30	45	3	93,02 %	2372	2405,1	41:28
4.	záv.	128	3	0	1	30	45	5	93,02 %	1569,5	1586,8	35:53
5.	nezáv.	14	3	0	1	14	21	3	65,12 %	2215	2390,4	22:44
6.	nezáv.	14	3	0	1	10	15	3	53,49 %	2143,5	2309,8	19:40
7.*	záv.	343	6	2	4	34	51	5	87,79 %	387	1346,1	48:13
8.*	záv.	343	6	2	4	20	30	5	82,56 %	211	1143,1	30:46
9.**	záv.	128	3	0	1	13	20	0	80,23 %	932	1066,5	06:26
10.**	nezáv.	14	3	0	1	14	21	0	53,49 %	2286,5	2467,6	15:31
11.**	záv.	128	3	0	1	30	45	0	95,35 %	389,5	396	08:40
12.**	záv.	343	6	2	4	34	51	0	88,37 %	261	268,7	11:18

Tabulka 7.1: Souhrn dosažených výsledků. V databázi bylo 43 různých osob. Barevně jsou zvýrazněny nejlepší dosažené výsledky.

* Během trénování nebylo použito omezující kritérium maximální chyby.
** Byla použita implementace Elmanovy sítě s algoritmem BP z knihovny Encog.

7.8 Zhodnocení výsledků

Navržený systém využívající rekurentních neuronových sítí se ukázal být vhodným nástrojem k řešení problému rozpoznávání obličejů ze sekvence obrázků. Elmanova síť díky své architektuře splnila svoji úlohu predikce následujících snímků v sekvenci.

Kromě algoritmů prezentovaných v původním článku jsme vyzkoušeli jiný způsob přístup k PCA analýze, kdy jsme zohlednili směr pohledu osob na obrázcích. Dosáhli jsme tak až o 30 % lepší úspěšnosti. K trénování rekurentních sítí jsme použili algoritmy zpětného šíření chyby a zpětného šíření chyby v čase. V úspěšnosti se ve většině případů varianta s klasickým BP ukázala jako vhodnější. S BPTT jsme docílili lepších výsledků pouze u sítí s menším počtem neuronů. U učení pomocí BP hraje ale velkou výhodu doba potřebná k učení, která je řádově 4x kratší, než u BPTT.

V nejlepším případě se nám podařilo dosáhnout průměrné úspěšnosti až 95,35 %. Testování proběhlo celkem 4x u 43 osob. Pro každou z nich byla k dispozici 1 testovací sekvence. Správně jsme tedy rozpoznali 41 lidí ze 43.

U výsledků testů je samozřejmě nutné brát v úvahu povahu dat, na nichž byl celý systém testován. Všechna videa pocházejí z jedné databáze. Jsou natočena proti jednolitému modrému pozadí, jsou téměř rovnoměrně osvětlená, nejsou rozmazaná, snímky byly vybírány ručně apod. Pro důkladnější otestování by bylo nutné jej konfrontovat s reálnějšími daty.

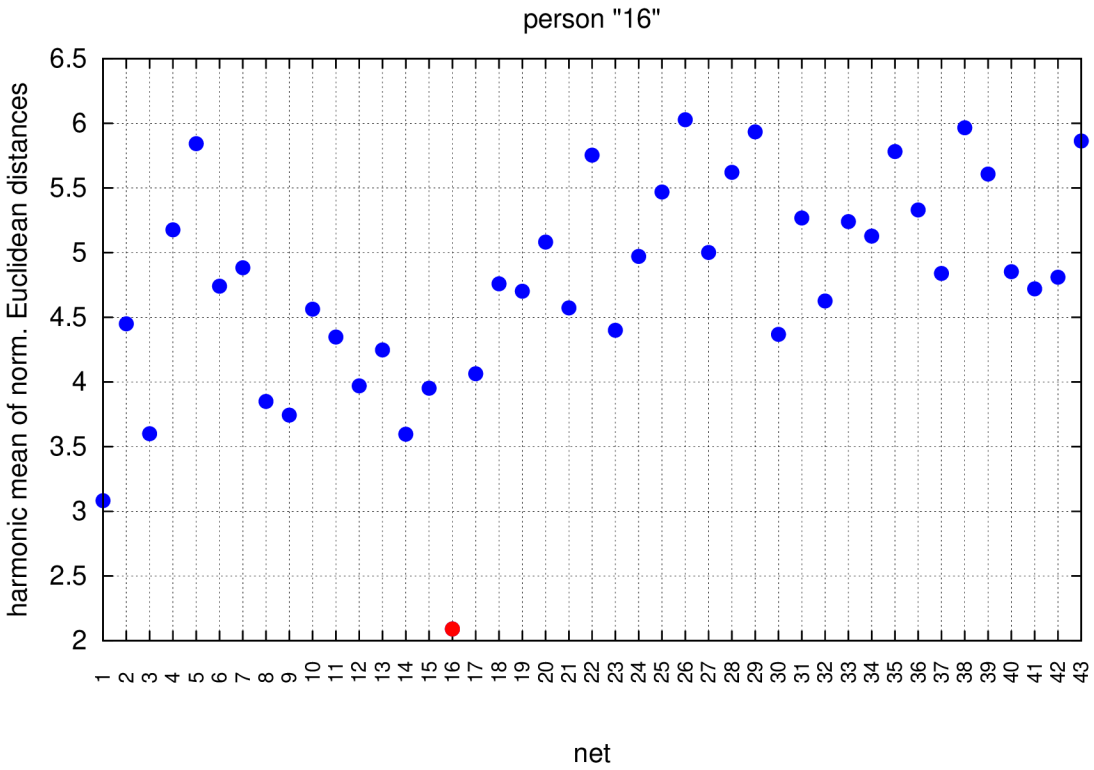
Otázkou také zůstává, zda se vyplatí takový způsob v praxi využívat. Ve srovnání s běžnou metodou rozpoznávání pomocí eigenfaces totiž dosahuje téměř shodných výsledků, nicméně za cenu větší složitosti, paměťové náročnosti a dlouhé doby potřebné k natrénování RNN.

7.9 Možnosti budoucího vývoje

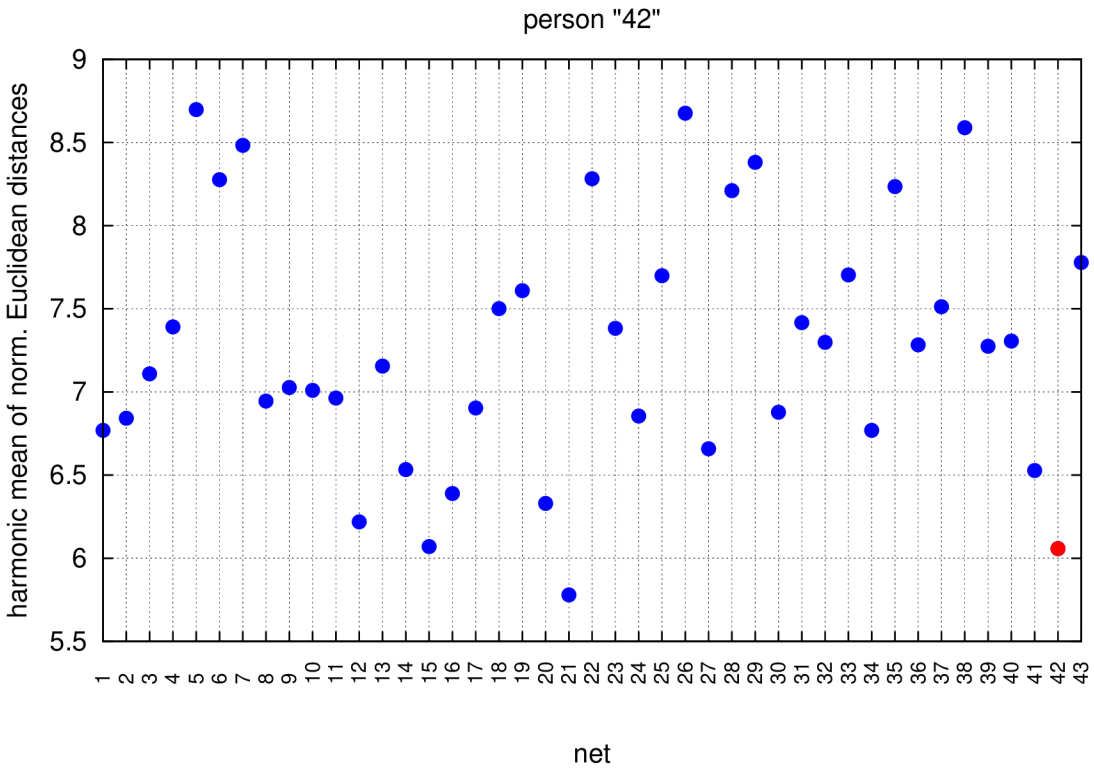
Jak jsme uvedli, byl implementovaný systém testován v „laboratorních podmínkách“. Bylo by tedy určitě velice vhodné ověřit jeho fungování na rozsáhlejší datové sadě obsahující reálná data. Zajímavým pokračováním práce by byl automatický výběr snímků z videonahrávek a zároveň automatická detekce obličejů. Rozpoznávací fáze algoritmu by určitě šla realizovat i v *real–time*.

Stálo by za to vyzkoušet z našich obrázků extrahovat jiné příznaky, než jsou váhy vlastních vektorů u metody eigenfaces. Bohužel není úplně lehké vybrat vhodný typ příznaků, aplikovatelný na snímky obličeje ve všech směrech natočení. Kromě jiných příznaků bychom také mohli k trénování použít jiný typ rekurentních sítí. Z prezentovaných se nabízí long short–term memory, která v mnohých aplikacích svými výsledky převyšuje ostatní typy NN.

Jelikož se jednalo o experimentální téma, nezajímali jsme se příliš o rychlost použitých algoritmů. Na poli optimalizace by šlo provést mnohá vylepšení. Už jenom samotným přepsáním trénovacího algoritmu BPTT, např. do jazyka C, bychom zajisté získali velké zrychlení výpočtů.



(a) správné rozpoznání



(b) chybné rozpoznání

Obrázek 7.3: Ukázka vyhodnocení rozpoznávání. V grafu jsou zaneseny harmonické průměry vzdáleností mezi souvisejícími vektory.

Obrázek 7.5: Vizualizace odezev sítě na zvolené vstupy. Nahoře obrázky použité ke získání příznakových vektorů, uprostřed výstupy sítě, jež byla naučena pro tuto osobu a dole výstupy sítě natrénované na datech jiného jedince.

Obrázek 7.6: Demonstrace schopnosti predikce RNN. Síti byl předložen jeden vstupní vektor a zbytek snímků sekvence byl predikován přivedením výstupu zpět na vstup. Z výstupních vektorů byly získány eigenfaces, na kterých lze tuto schopnost vhodně ukázat.

Kapitola 8

Závěr

Cílem diplomové práce bylo seznámit se s problematikou umělých neuronových sítí se zaměřením na rekurentní architektury a prozkoumat možné aplikace RNN v oblasti počítačového vidění. Dále vybrat úlohu vhodnou k řešení za pomoci RNN, zvolené řešení implementovat a provést experimenty. Všechny body zadání práce byly splněny.

V teoretické části byla mimo jiné popsána síť typu vícevrstvý perceptron, Jordanova a Elmanova síť, long short–term memory a Hopfieldova síť. Z trénovacích algoritmů byly prezentovány backpropagation, backpropagation through time a učící a vybavovací fáze u Hopfieldovy sítě. Byly vybrány čtyři zajímavé úlohy, které se jejich autoři s úspěchem rozhodli řešit pomocí RNN. Popisu těchto problémů a principům jejich řešení byla věnována celá jedna kapitola. Jedna z teoretických kapitol se také věnuje rozpoznávání obličejů, jímž se potom zabývá celá praktická část práce.

Během studia byl objeven článek *„Tracking and Recognition of Face Sequences“* [8], kde se autoři mimo jiné zabývají rozpoznáváním obličeje ze sekvence snímků. Naše řešení bylo do značné míry tímto přístupem inspirováno. Využívá se tady techniky eigenfaces ke získání příznakových vektorů ze vstupních obrázků a Elmanových rekurentních sítí k predikci následujících snímků v sekvenci.

Z provedených testů vyplývá, že námi navržený a implementovaný systém je na použité testovací sadě schopen dosáhnout velice dobrých výsledků. V nejlepším případě jsme docílili úspěšnosti přes 95 %. Co se týče úspěšnosti rozpoznávání, je na tom náš systém vždy stejně nebo lépe, než běžné rozpoznávání založené na metodě eigenfaces a hledání nejbližšího souseda. Kde ale náš systém pokulhává, je rychlost. Natrénování neuronových sítí si totiž většinou vyžádalo i několik desítek minut.

Elmanova RNN splnila naše očekávání a ukázala, že je díky svojí rekurentní architektuře a vnitřní paměti schopna správně predikovat následující obrázky v sekvenci. Toho se nám s výhodou podařilo využít při řešení problému identifikace osob dle snímků obličeje.

Literatura

[1] BERNACKI, M.; WŁODARCZYK, P.; GOLDA, A.: Principles of training multi-layer neural network using backpropagation [online]. 2005 [cit. 2011-01-02], Akademia Górniczo Hutnicza w Krakowie, Katedra Elektroniki.
URL http://home.agh.edu.pl/~vlsi/AI/backp_t_en/backprop.html

[2] BODÉN, M.: A guide to recurrent neural networks and backpropagation. *Electrical Engineering*, ročník 2001, č. 2: s. 1–10.

[3] BRADSKI, G.; KAEHLER, A.: *Learning OpenCV*. O'Reilly Media Inc., 2008, ISBN 978-0-596-51613-0.

[4] BURIÁN, P.: *Rozpoznání lidské tváře*. Bakalářská práce, FIT VUT v Brně, 2010.

[5] DRAHANSKÝ, M.: Slajdy k předmětu Biometrické systémy. 2005, FIT VUT v Brně.

[6] DRÁBEK, O.; SEIDL, P.; TAUFER, I.: Umělé neuronové sítě – základy teorie a aplikace. *CHEMagazín*, ročník XVI, č. 1, 2006: s. 12–14, ISSN 1210-7409.

[7] *Emgu CV: OpenCV in .NET [online]*. 2008 [cit. 2011-01-10].
URL http://www.emgu.com

[8] GONG, S. et al.: Tracking and Recognition of Face Sequences. In *European Workshop on Combined Real and Synthetic Image Processing for Broadcast and Video Production*, Hamburg, Germany, 1994, s. 97–112.

[9] GRAVES, A.; LIWICKI, M.; FERNÁNDEZ, S. et al.: A Novel Connectionist System for Unconstrained Handwriting Recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, ročník 31, 2009: s. 855–868, ISSN 0162-8828.

[10] Heaton Research: *Encog Java and DotNet Neural Network Framework [online]*. 2008 [cit. 2011-01-10].
URL http://www.heatonresearch.com/encog

[11] HENDRYCH, P.: *Lokalizace obličeje pomocí neuronové sítě*. Diplomová práce, FIT VUT v Brně, 2008.

[12] Illinois Network Design and Experimentation Group: Towards building a high–performance, interference-mitigated and robust wireless mesh network [online]. 2006 [cit. 2010-12-28], Department of Computer Science, University of Illinois at Urbana – Champaign.
URL http://lion.cs.uiuc.edu/projects/wmn.html

[13] INAGAKI, K.; SATO, S.; UMEZAKI, T.: A Recurrent Neural Network Approach to Rear Vehicle Detection Which Considered State Dependency. *The Journal on Systemics, Cybernetics and Informatics (JSCI)*, ročník 1, č. 4, 2003.

[14] KŘEPSKÝ, J.: *Detekce hran pomocí neuronové sítě*. Bakalářská práce, FIT VUT v Brně, 2009.

[15] KOŠČAK, J.: *Experimentálna analýza algoritmu backpropagation through time pre učenie rekurentných neurónových sietí*. Diplomová práca, TU FEI Košice, 2005.

[16] KOUTNÍK, J.: Elektronická příloha ke skriptu pro předmět Neuronové sítě a neuropočítače [online]. 2002 [cit. 2011-01-02], Katedra počítačů FEL ČVUT v Praze. URL http://neuron.felk.cvut.cz/courseware/

[17] KRÖSE, B.; SMAGT, P.: *An Introduction to Neural Networks*. Amsterdam, The Netherlands: University of Amsterdam, 8. vydání, 1996.

[18] Language and Media Processing Laboratory: *ViPER: The Video Performance Evaluation Resource [online]*. 2003 [cit. 2011-05-11], University of Maryland, USA. URL http://viper-toolkit.sourceforge.net/

[19] LEE, S.; SONG, H.: A New Recurrent Neural-Network Architecture for Visual Pattern Recognition. *IEEE Transactions on Neural Networks*, ročník 8, č. 2, 1997: s. 331–340, ISSN 1045-9227.

[20] LOMOVCIV, P.: *Predikce pomocí neuronové sítě*. Diplomová práce, FIT VUT v Brně, 2005.

[21] LÁVIČKA, M.: Vlastní čísla, vlastní vektory (pomocný text k předmětu Geometrie I) [online]. 2004 [cit. 2010-12-27], Západočeská univerzita v Plzni. URL http://home.zcu.cz/~lavicka/subjects/G1/texty/pomoctext_eigen.pdf

[22] ŠÍMA, J.; NERUDA, R.: *Teoretické otázky neuronových sítí*. Praha: Matfyzpress, 1996, ISBN 80-85863-18-9.

[23] MACENAUER, O.: *Identifikace obličeje*. Diplomová práce, FIT VUT v Brně, 2010.

[24] MAŘÍK, V.; ŠTĚPÁNKOVÁ, O.; LAŽANSKÝ, J. et al.: *Umělá inteligence (4. díl)*. Praha: Academia, 2003, ISBN 80-200-1044-0.

[25] MEHROTRA, K.; MOHAN, C.; RANKA, S.: *Elements of Artificial Neural Networks*. USA: MIT Press, 1996, ISBN 0262133288.

[26] ŠNOREK, M.; JIŘINA, M.: *Neuronové sítě a neuropočítače*. Praha: Vydavatelství ČVUT, 1996, ISBN 80-01-01455-X.

[27] SANDERSON, C.: VidTIMIT Dataset Documentation. 2010 [cit. 2011-01-10]. URL http://archive.itee.uq.edu.au/~conrad/vidtimit/zips/vidtimit_documentation.pdf

[28] SANDERSON, C.; PALIWAL, K. K.: Polynomial Features for Robust Face Authentication. In *IEEE International Conference on Image Processing (ICIP)*, ročník 3, 2002, s. 997–1000.

[29] ČÁSTEK, P.: *Identifikace obličeje*. Diplomová práce, FIT VUT v Brně, 2008.

[30] SVOBODA, P.: *Vyhledávání osob ve fotografii*. Diplomová práce, FIT VUT v Brně, 2009.

[31] TURK, M.; PENTLAND, A.: Eigenfaces for Recognition. *Journal of Cognitive Neuroscience*, ročník 3, č. 1, 1991.

[32] VLACH, J.; PŘINOSIL, J.: Lokalizace obličeje v obraze s komplexním pozadím. *Elektrorevue - Internetový časopis (http://www.elektrorevue.cz)*, ročník 4, 2007: s. 1–12, ISSN 1213-1539.

[33] WALCZYSKO, M.: *Analýza EEG signálu pomocí analýzy hlavních komponent (PCA)*. Bakalářská práce, FEKT VUT v Brně, 2008.

[34] Wikipedia – The Free Encyclopedia: Eigenvalues and eigenvectors [online]. 2005 [cit. 2010-12-27].
URL http://en.wikipedia.org/wiki/Eigenvalues_and_eigenvectors

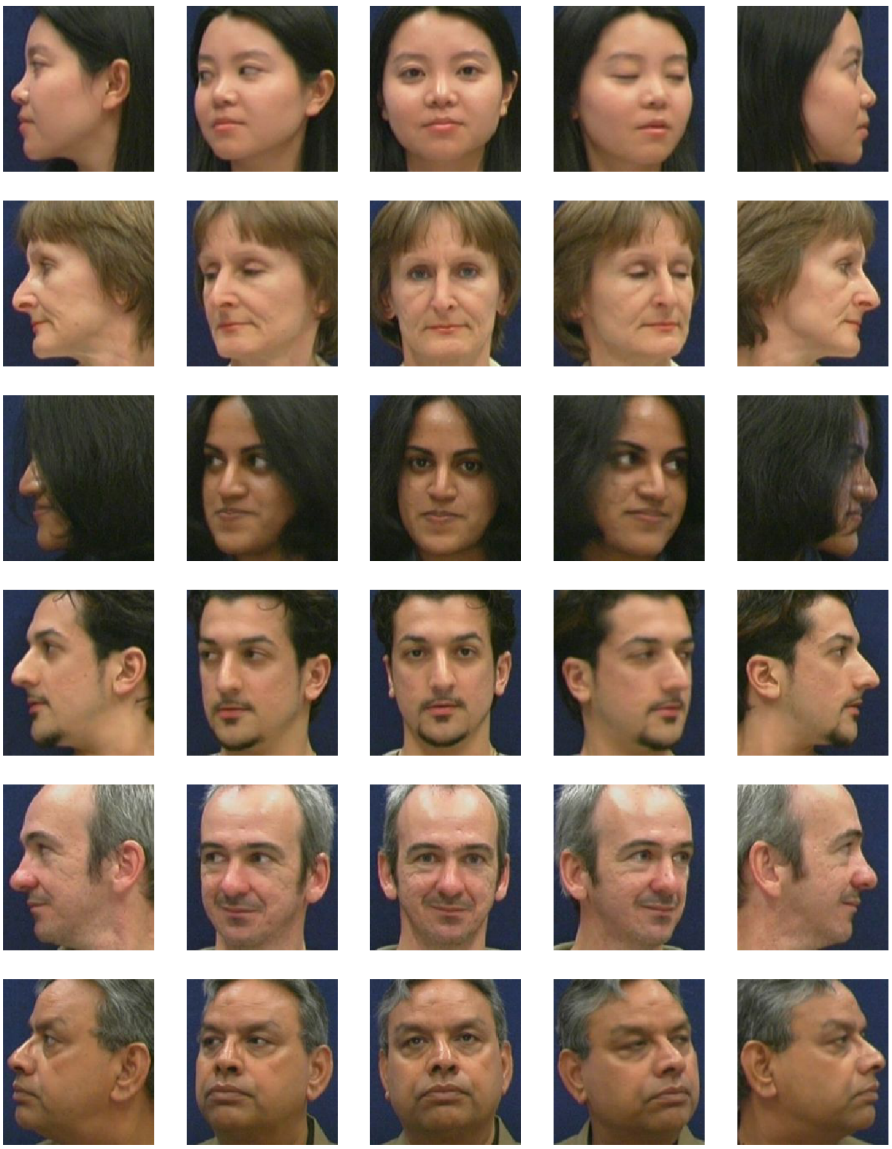
[35] Wikipedia – The Free Encyclopedia: Long short term memory [online]. 2007 [cit. 2011-05-01].
URL http://en.wikipedia.org/wiki/Long_short_term_memory

[36] YOON, S. H.; HUR, G. T.; KIM, J. H.: Recurrent Neural Network Verifier for Face Detection and Tracking. In *IEA/AIE*, 2006, s. 488–499.

[37] ZBOŘIL, F. V.: Slajdy k předmětu Soft Computing. 2010, FIT VUT v Brně.

Příloha A

Ukázka trénovacích snímků vybraných z databáze VidTIMIT



Obrázek A.1: Několik vybraných obrázků obličejů z trénovací sady.