Algoritmy pro rozpoznání ručně psaných znaků

Algorithms for handwritten characters recognition

Václav Růžek

Bakalářská práce
2010

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2009/2010

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Václav RŮŽEK**
Osobní číslo:	**A07205**
Studijní program:	**B 3902 Inženýrská informatika**
Studijní obor:	**Informační a řídicí technologie**
Téma práce:	**Algoritmy pro rozpoznání ručně psaných znaků**

Zásady pro vypracování:

Cílem práce je vytvoření přehledu algoritmů navržených pro rozpoznávání ručně psaných znaků (písmen a číslic) a jejich vzájemné porovnání.

1. **Sestavte přehled základních typů algoritmů používaných pro rozpoznávání ručně psaných znaků a vytvořte jejich klasifikaci.**
2. **Vytvořte seznam zdrojů (internetových stránek), kde leze získat detailní popisy jednotlivých algoritmů, případně zdrojové kódy odpovídajících programů.**
3. **Shromážděte dostupné programy, pomocí nichž lze jednotlivé algoritmy otestovat on-line (internetová stránka) nebo off-line (spustitelný soubor).**
4. **Otestujte a porovnejte shromážděné programy.**

Rozsah bakalářské práce:

Rozsah příloh:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **Bishop, Christopher M.. Pattern recognition and machine learning /. New York : Springer, 2006. 738 s. : ISBN 978-0387-31073-2.**
2. **Fuzzy models and algorithms for pattern recognition and image processing. New York : Springer Science + Business Media, 2005. 776 s. : ISBN 978-0-387-24515-7.**
3. **Handwriting recognition [online]. 2009 , 1 December 2009 [cit. 2009-12-08]. Dostupný z WWW: <http://en.wikipedia.org/wiki/Handwriting_Recognition>.**
4. **Wu, Muguel Po-Hsien. Handwritten Character Recognition [online]. 2003 [cit. 2009-12-08]. Dostupný z WWW: <http://innovexpo.itee.uq.edu.au/2003/exhibits/s804636/thesis.pdf>.**
5. **Chalupa, Petr. Programové vybavení pro vizuální vyhodnocování testů. Zlín, 1999. 55 s. Vysoké učení technické v Brně. Vedoucí diplomové práce Ing. Tomáš Sysala.**

Vedoucí bakalářské práce: **Ing. Petr Chalupa, Ph.D.**
Ústav řízení procesů

Datum zadání bakalářské práce: **5. března 2010**

Termín odevzdání bakalářské práce: **1. června 2010**

Ve Zlíně dne 5. března 2010

L.S.

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky

prof. Ing. Vladimír Vašek, CSc.
děkan

doc. Ing. Ivan Zelinka, Ph.D.
ředitel ústavu

ABSTRAKT

Práce se zabývá procesem rozpoznání ručně psaných znaků. Popisuje jednotlivé techniky sloužící k zpracování vstupních dat. Hlavní náplní práce je popis jednotlivých algoritmů a metod používaných pro rozpoznání ručně psaných znaků. Zabývá se klasifikátory na principu minimální vzdálenosti, neuronovými sítěmi a pravděpodobnostními Markovovými modely. V praktické části jsou zmíněny informační zdroje a programy sloužící pro detailní popis zmíněných algoritmů.

Klíčová slova: Rozpoznání ručně psaných znaků, klasifikátory minimální vzdálenosti, neuronové sítě, Neocognitron, skryté Markovovy modely.

ABSTRACT

This bachelor work examines the process of handwritten characters recognition. It describes different methods used for processing of input data. The main scope is description of algorithms and methods used for handwritten characters recognition. It deals with the minimum distance classifiers, neural networks and stochastic Markov models. Information sources and programs containing detailed descriptions of these algorithms are mentioned in the practical part.

Keywords: Handwritten characters recognition, minimum distance classifiers, neural networks, Neocognitron, hidden Markov models.

Tímto bych chtěl poděkovat své rodině a přátelům za podporu, díky které jsem se mohl plně soustředit na tvorbu bakalářské práce, dále Ing. Petru Chalupovi, Ph.D. za cenné rady a připomínky k této práci.

„Vzdělání má hořké kořeny, ale sladké ovoce.“

Démokritos z Abdér

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně 27.5.2010 …….…….

podpis diplomanta

OBSAH

ÚVOD

Rozpoznání ručně psaných znaků je činnost, která provází lidstvo již od kresby prvních piktogramů přes vznik písma až do současnosti. Lidé se naučili pomocí znaků zaznamenávat určitou skutečnost, s čímž šla ruku v ruce i schopnost naučit se tyto znaky rozpoznávat. Rozpoznání spočívá ve schopnosti znakům přiřadit určitou vlastnost, nebo-li zařadit je do určité skupiny charakterizující tyto vlastnosti. Této schopnosti se učíme již od dětství a využíváme ji poté naprosto automaticky v nepřeberném množství úkonů během každodenního života.

S rozvojem výpočetní techniky došlo v polovině minulého století ke snaze naučit této schopnosti i stroje. Vznikly první systémy, jež umožnily třídění poštovních zásilek dle směrovacího čísla apod. Jelikož se nejedná o elementární problém, vzhledem k různým druhům rukopisů, sklonu, velikosti apod., tak se také rozpoznávací techniky založené na triviálních principech vyvinuly do mnohem sofistikovanějších podob, kdy se například některé systémy inspirují činností lidského mozku - neuronové sítě apod. Samotný princip rozpoznání, tedy zařazení znaků do určitých skupin, nebo-li tříd, však zůstává neměnný.

I když je patrné, že v dnešní době dochází k částečnému útlumu používání ručně psaných znaků, přesto však stále existují obory, kde je jejich význam nenahraditelný. Systémy pro rozpoznání ručně psaných znaků tak mají široké pole působnosti všude tam, kde se zpracovávají velké objemy ručně psaných dat. Příkladem jsou poštovní služby, bankovnictví, zpracování dotazníků, převod rukopisů do digitální podoby apod. Objevují se i zcela nové oblasti, kde nacházejí tyto systémy využití. Příkladem je on-line rozpoznání znaků, které je dnes běžnou součástí veškerých dotykových zařízení jako jsou přenosné počítače, mobilní telefony apod., což předpovídá slibný rozvoj těchto systémů i v budoucnosti.

I. TEORETICKÁ ČÁST

1 ROZPOZNÁNÍ TEXTU

Rozpoznání ručně psaného textu je specifickým oborem vědní disciplíny optického rozpoznávání znaků (*Optical Character Recognition*). OCR je v podstatě technologie, která slouží k zpracování a převedení textu z bitmapového formátu do digitální formy. Jedná se tedy o digitalizaci dat.

1.1 Historie

Technologie rozpoznání znaků je poměrně mladým oborem, avšak určité náznaky můžeme pozorovat již na počátku 20. století. Mezi první zmínky patří rok 1914, kdy byl v USA vydán patent na mechanický přístroj sloužící mimo jiné i k rozpoznávání rukopisů. Dále byly ve 30. letech zaznamenány patenty na mechanické přístroje, které pomocí šablony a fotodetektoru rozpoznávaly určité znaky. Tyto přístroje však mají s dnešním oborem OCR jen pramálo společného.

Skutečné počátky bychom měli datovat do 60. let 20. století, které souvisí s rozvojem výpočetní techniky. Je zde snaha o rychlé a efektivní zpracování bankovních dokumentů jako jsou šeky, cenné papíry, kreditní karty, dále zpracování poštovních zásilek atd. Úplně první systém vytvořila společnost Reader's Digest již v roce 1955. Ten sloužil pro převod ručně psaného textu na děrové štítky, pro další zpracování počítačem. V roce 1965 vzniká v USA standart písma OCR-A, což je v podstatě první zjednodušené písmo, umožňující strojové čtení [1]. V témže roce začala využívat technologii OCR americká poštovní společnost United States Postal Service na třídění zásilek. Vznikají také první slavné systémy jako IMB 1287, který byl představen na světové výstavě v New Yorku 1965.

V 70. letech se objevují první komerční systémy a dochází k postupnému rozvoji OCR. Avšak až do 90. let je díky vysoké pořizovací ceně jejich využití omezeno pouze na několik málo velikých společností. V posledních 20 letech došlo v důsledku prudkého rozvoje výpočetní techniky jednak ke zvýšení efektivity systémů, a také především ke snížení jejich ceny, což umožnilo masové rozšíření OCR systémů mezi běžné uživatele. Díky tomu dnes existuje celá škála oborů, kde je využíváno těchto systémů.

1.2 Struktura

1.2.1 Vstupní data

Vstupem celého rozpoznávacího systému jsou různé druhy vstupních dat, které má systém zpracovávat. U rozpoznání ručně psaného textu se vstupní data dělí do dvou základních skupin.

On-line data – text je zapisován přímo v digitální formě pomocí speciálních zařízení jako jsou digitalizační tablety, dotykové displeje. Rozpoznání on-line dat zaznamenává v posledních letech raketový rozvoj, díky rozmachu zařízení jako jsou kapesní počítače, dotykové mobilní telefony a další přístroje s dotykovým displejem.

Off-line data – jsou klasická data napsaná na papíře. Pro jejich zpracování je tedy nejprve nutné tyto data převést do digitální formy pomocí skeneru.

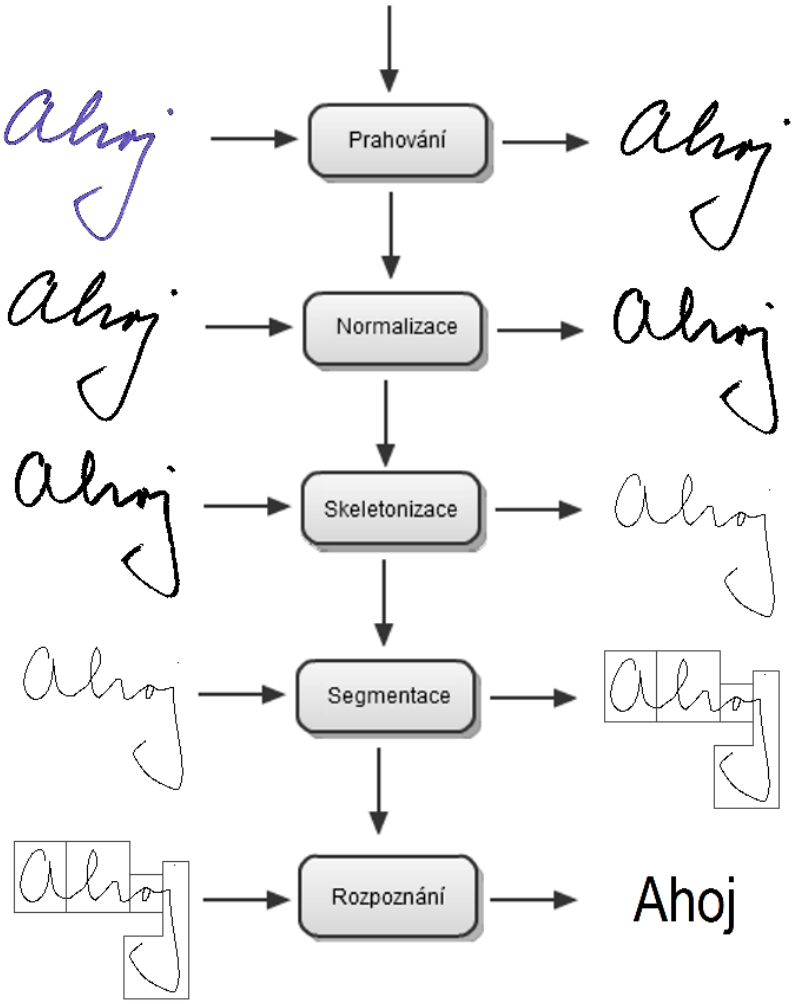
1.2.2 Předzpracování dat

Přímá vstupní data však mají stále množství nedostatků, které stěžují jejich rozpoznání příslušnými algoritmy. Na řadu tedy musí přijít tzv. ***preprocesní zpracování,*** které vstupní data upraví do takové podoby, která je příhodná pro rozpoznávací algoritmy. Preporocesní zpracování má větší váhu u dat získaných off-line, avšak některé jeho části najdou uplatnění i u zpracovávání on-line dat. Mezi techniky preprocesního zpracování patří jednoduché techniky ***prahování*** sloužící k oddělení znaků od pozadí. Techniky ***segmentace*** pro izolaci jednotlivých znaků. Dále ***skeletonizace*** pro vypreparování kostry daných znaků a v neposlední řadě ***normalizační*** techniky pro výslednou úpravu znaků do požadované podoby. Tímto předzpracováním vstupních dat značně ulehčíme práci samotných algoritmů a dosáhneme vyšší účinnosti rozpoznání.

1.2.3 Rozpoznávací algoritmy

Jádrem OCR systémů jsou již samotné metody pro rozpoznání. Dnes se nejběžněji používají algoritmy založené na neuronových sítích, skrytých Markovových modelech, klasifikátorech minimální vzdálenosti apod. Tyto metody mají obvykle velikou úspěšnost rozpoznání u dobře napsaných dat. U ručně psaných dat je rozpoznání komplikovanější díky značné rozmanitosti jednotlivých rukopisů. Při zpracování tedy může dojít u různých

systémů k chybné klasifikaci znaku nebo k neschopnosti znak rozpoznat, proto je zde potřeba určité následné úpravy získaných dat.



Obr. 1. Schéma ideálního preprocesního zpracování a rozpoznání dat.

1.2.4 Poprocesní zpracování

Tímto se dostáváme k další části systémů, kterou je poprocesní zpracování dat. Zde pracujeme se znalostí kontextu výsledného textu. Mezi nejběžnější nástroje patří kontrola pravopisu (*spell checking*). K chybné kvalifikaci může dojít jednak díky zhoršení kvality dat. Některé znaky jsou si však velmi podobné, proto je může algoritmus rozpoznání snadno zaměnit. Typickým příkladem jsou např. „5“ a „S“ [2] nebo „rn“ a „m“ aj. Tato snadno vzniknutelná záměna je díky znalosti kontextu také snadno identifikovatelná a opravitelná. Některé znaky je bez znalosti kontextu nemožné efektivně rozpoznat např. „0“ a „O“. Dnešní robustní systémy se bez poprocesního zpracování neobejdou.

1.3 Využití

Možností využití systémů pro rozpoznání znaků je dnes nepřeberné množství. Klasické rozpoznání znaků OCR nachází značné využití při digitalizaci textů, čehož se hojně využívá například v knihovnách apod. Specifičtější rozpoznání ručně psaného textu můžeme také využít k digitalizaci textu jako jsou rukopisy, což je zde však spíše okrajová záležitost. Hlavní pole působnosti těchto systémů je především tam, kde se zpracovává velké množství ručně psaných dat, což znamená např. na poštách pro identifikaci PSČ, dále v bankovnictví pro identifikaci šeků, v pojišťovnictví, pro zpracování různých dotazníků apod.

U rozpoznání on-line dat zaznamenávají tyto systémy v posledních letech masivní rozšíření a lze předpokládat značný rozvoj i v budoucnosti. Systémy pro rozpoznání ručně psaného textu se stávají dnes již běžným standardem u tabletových PC, kapesních počítačů, mobilních telefonů a v podstatě u většiny zařízení vybavených dotykovým displejem. Tyto systémy jsou dnes běžnou součástí balíčků operačních systémů, což je masově rozšiřuje ke koncovým uživatelům.

2 PREPROCESNÍ ZPRACOVÁNÍ

V procesu rozpoznávání znaků má preprocesní zpracování neodmyslitelné místo. Vstupní obraz je potřeba různými procesy zjednodušit, normalizovat a usnadnit tak práci samotným algoritmům rozpoznání znaků. Metody a postupy jsou různé, dle charakteru vstupních dat. Část preprocesního zpracování bude logicky obsáhlejší v případě naskenovaných off-line dat, než u dat zadaných on-line.

2.1 Prahování

Prahování (*thresholding*) je nejjednodušší metodou segmentace obrazu. Na základě jasové nebo barevné složky transformuje vstupní obraz na dvoubarevný obraz. Matematicky lze prahování vyjádřit jako funkci [3]:

$$g(i,j) = \begin{cases} 1 & pro \quad f(i,j) \geq T \\ 0 & pro \quad f(i,j) < T \end{cases} \tag{1}$$

kde:

f je vstupní funkce,

(i, j) jednotlivé pixely obrazu,

T konstantní hodnota tzv. práh.

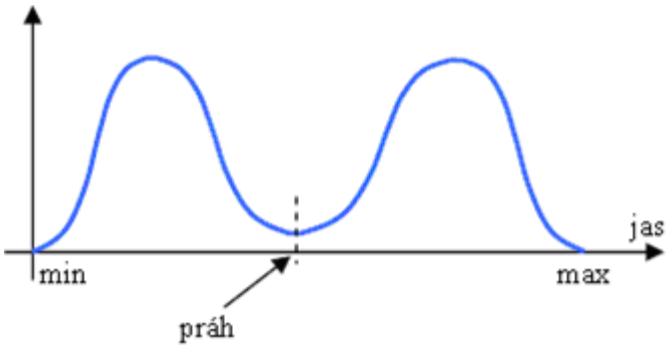
Na základě hodnoty prahu tedy algoritmus porovná všechny pixely vstupního obrazu a nahradí je logickou hodnotou 0 nebo 1. Tím dosáhneme jednak transformace barevné složky obrazu na obraz černobílý, a také oddělení popředí od pozadí. Stěžejní je zde samotná hodnota prahu. Určování prahu může být:

- interaktivní – práh je určován obsluhou
- automatické – pomocí některé z metod prahování.

Nejjednodušší metodou určování prahu je ***procentní prahování***. Tuto metodu můžeme využít za předpokladu, že známe poměr ploch objektu a pozadí. Na základě této znalosti určíme hodnotu prahu tak, aby právě požadované procento plochy mělo nižší resp. vyšší hodnotu než práh.

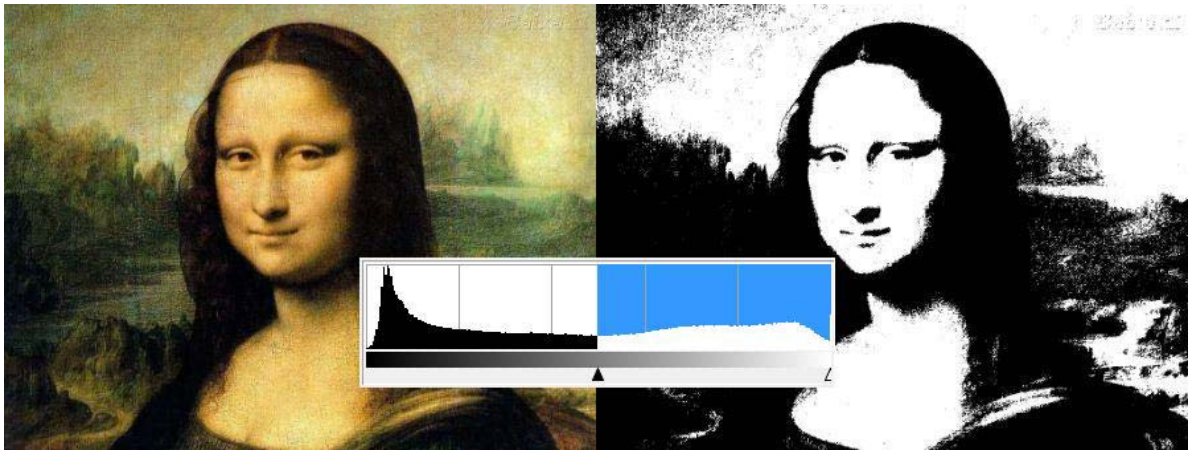
Další metody se opírají o znalosti ***histogramu***. Histogram je grafem četnosti jasové nebo barevné složky obrazu. Pokud jsou v obraze objekty podobného jasu jasově odlišeny

od pozadí, má histogram jasu právě dva vrcholy (*bimodální*). Jeden vrchol odpovídá hodnotě jasu objektů a druhý jasu pozadí. Hodnota jasu prahu je poté minimem četnosti mezi těmito dvěma vrcholy.



Obr. 2. Ukázka určení prahu z ideálního histogramu.

Histogram může obsahovat i více než dva vrcholy, poté je nutné provést segmentaci s více prahy. V praxi často není možné efektivně určit vrcholy a minima histogramu, někdy je potřeba výsledný obraz rozdělit a pro každou část analyzovat vlastní práh.



Obr. 3. Příklad aplikace prahování s analýzy histogramu.

Díky své jednoduchosti je prahování stále jednou z nejpoužívanějších metod zpracování obrazu. Jeho největší předností je hardwarová nenáročnost a rychlost zpracování. Nevýhodou je nemožnost od sebe odlišit jasově podobné složky. Při zpracování textu však většinou analyzujeme tmavý text na světlém pozadí, což zaručuje dostatečné jasové odlišení. Problémy mohou nastat, pokud se na pozadí vyskytují výrazné vzory, které není možné touto metodou efektivně odstranit.

2.2 Normalizace

Je typickou preprocesní metodou při zpracování ručně psaného textu. Základním úkolem normalizace je korekce vstupních znaků dle požadovaných proporcí, tzn. srovnání šikmo psaného textu, transformaci velikosti znaků atd.

Využíváme zpravidla lineárních transformací obrazu jako otočení, zkosení, změnu měřítka. Každá tato operace je vyjádřena transformační maticí A, která má pro 2D prostor rozměry 3x3. Jednotlivé transformace se mohou skládat a to tak, že jsou postupně aplikovány na zadaný bod. Výslednou transformaci lze vyjádřit jako matici, která vznikne násobením matic jednotlivých transformací zprava, dle zadaného pořadí [4].

2.2.1 Otočení obrazu

Jedná se o transformaci, při které otáčíme obraz kolem daného bodu o určitý úhel. V případě otáčení kolem počátku souřadnic můžeme tuto transformaci vyjádřit vztahy:

$$x' = x.\cos\alpha - x.\sin\alpha \tag{2}$$

$$y' = y.\sin\alpha - y.\cos\alpha \tag{3}$$

kde:

x', y' jsou souřadnice transformovaného bodu,

x, y souřadnice původního bodu,

α úhel otočení.

Maticové vyjádření otočení je:

$$A_R = \begin{bmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4}$$

Při zpracování diskrétního obrazu se mohou objevit v transformovaném obraze efekty, které se nevyskytují v původním obraze. Toto je způsobeno nepřesným přepočtem souřadnic při transformaci rastru obrazu a tento jev nazýváme alias [5].

2.2.2 Změna měřítka

Změna měřítka (*zoom*) je operací změny velikosti objektu ve směru os. Matematicky lze tuto transformaci vyjádřit vztahem:

$$x' = x.s_x \tag{5}$$

$$y' = y.s_y \tag{6}$$

kde:

s_x, s_y je koeficient změny měřítka v dané ose.

Základním prvkem při změně měřítka je tedy koeficient změny měřítka. Na velikosti koeficientu závisí i povaha změny měřítka. Pokud je koeficient v intervalu (0,1) dochází ke zmenšení. Je-li větší než 1 dochází ke zvětšení (prodloužení). Pro záporné hodnoty koeficientu nastává převrácení, nebo-li transformace v opačném směru. Maticí je změna měřítka vyjádřena jako:

$$A_S = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{7}$$

2.2.3 Zkosení

Další z často využívaných transformačních operací, např. při korekci šikmo psaného textu. Matematicky můžeme zkosení vyjádřit rovnicemi:

$$X' = X + SH_X \cdot Y \tag{8}$$

$$Y' = SH_Y \cdot X + Y \tag{9}$$

kde:

SH_X, SH_Y jsou koeficienty zkosení příslušných os.

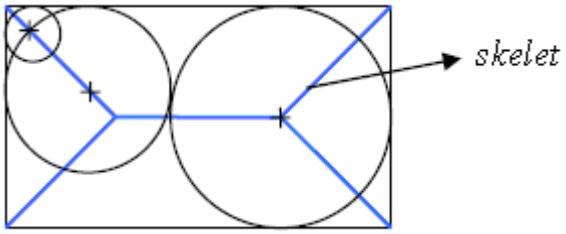
Matice pro transformaci zkosení má tvar:

$$A_{Sh} = \begin{bmatrix} 1 & SH_Y & 0 \\ SH_X & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{10}$$

2.3 Skeletonizace

Skeletonizace je dekompozice uzavřené oblasti (*2D entit*) na posloupnost křivek (*1D entit*). Jedná se o proces, při kterém dochází k redukci informace a celkovému zjednodušení objektu, při zachování jeho tvarové charakteristiky [6]. Výsledkem procesu skeletonizace je topologická kostra tzv. ***skeleton***.

Pro danou oblast představuje skeleton posloupnost křivek. V praxi je mnohdy výhodnější diskrétní reprezentace skeletonu pomocí úseček s konečným množstvím vrcholů. Základním a nejjednodušším typem skeletonu je tzv. ***střední osa*** (*medial axis*). Topologickou kostru S(X) poté můžeme vyjádřit jako množinu všech bodů – středů kružnic, které jsou obsaženy v oblasti X a zároveň se dotýkají hranice oblasti alespoň ve 2 bodech. Tyto kružnice jsou nazývány maximálním kruhem, jelikož se dotýkají hranice ve 2 a více bodech, a proto je již není možné rozšířit.



Obr. 4. Skeleton obdélníku určený metodou střední osy.

2.3.1 Sekvenční ztenčování

Sekvenční ztenčování je metoda skeletonizace využívaná při zpracování obrazu pro rozpoznání. Skeleton se ve výsledku skládá pouze z čar tloušťky 1 a izolovaných bodů.

Operace ztenčování je formálně definována jako:

$$X \oplus B = X \setminus (X \otimes B) \tag{11}$$

kde:

X je obraz,

B strukturní element.

Obvykle se ztenčování používá opakovaně. Sekvenční ztenčování pro čtvercový rastr může být vyjádřeno jako posloupnost 8 strukturních elementů:

$$X \oplus \{B_{(i)}\} = \left(\left(\left(X \oplus B_{(1)}\right) \oplus B_{(2)}\right) \ldots . \oplus B_{(n)}\right) \tag{12}$$

Ztenčování konverguje do konečného stavu, který je indikován tím, že dvě po sobě následující iterace mají shodný výsledek. Jako náhrada skeletu se využívá strukturní element L, který dle Golayovy abecedy nabývá hodnot [7]

$$L_1 = \begin{bmatrix} 0 & 0 & 0 \\ * & 1 & * \\ 1 & 1 & 1 \end{bmatrix}, \; L_2 = \begin{bmatrix} * & 0 & 0 \\ 1 & 1 & 0 \\ * & 1 & * \end{bmatrix}, \; L_3 = \begin{bmatrix} 1 & * & 0 \\ 1 & 1 & 0 \\ 1 & * & 0 \end{bmatrix}, \; \ldots\ldots \; L_7 = \begin{bmatrix} 0 & * & 1 \\ 0 & 1 & 1 \\ 0 & * & 1 \end{bmatrix}, \; L_8 = \begin{bmatrix} 0 & 0 & * \\ 0 & 1 & 1 \\ * & 1 & * \end{bmatrix}.$$

Homotopická náhrada skeletu pomocí elementu L může mít velkou rozeklanost. Tento nedostatek se nahradí elementem E Golayovy abecedy, kde

$$E_1 = \begin{bmatrix} * & 1 & * \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \; E_2 = \begin{bmatrix} 0 & * & * \\ 0 & 1 & * \\ 0 & 0 & 0 \end{bmatrix}, \; E_3 = \begin{bmatrix} 0 & 0 & * \\ 0 & 1 & 1 \\ 0 & 0 & * \end{bmatrix}, \; \ldots\ldots \; E_7 = \begin{bmatrix} * & 0 & 0 \\ 1 & 1 & 0 \\ * & 0 & 0 \end{bmatrix}, \; E_8 = \begin{bmatrix} * & * & 0 \\ * & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

Většinou využíváme sekvenční ztenčování elementem L až do konečného stavu a poté výsledek upravíme pomocí elementu E.

Obr. 5. Příklad sekvenčního ztenčování s iterací po 10 krocích.

Díky sekvenčnímu ztenčování tedy docílíme značného snížení objemu dat. Dále můžeme také v některých případech napravit některé nepravidelnosti ve zpracovávaných znacích.

2.4 Segmentace obrazu

Vedle metody prahování se při preprocesním zpracování obrazu využívají i další metody segmentace. Úkolem těchto metod je rozdělení obrazu na jednotlivé segmenty, v ideálním případě obsahujícím jednotlivé znaky. Tato procedura je velmi důležitá pro další zpracování rozpoznávacími algoritmy, kde je potřeba, abychom byli schopni přivést na jejich vstup izolovaně jednotlivé znaky.

2.4.1 Segmentace na základě detekce hran

Hrana je místo obrazu, kde se náhle mění jeho jasová funkce. Nalezené hrany jsou spojeny do řetězců, ze kterých získáváme informaci o hranicích jednotlivých obrazových objektů. Většina metod detekce hran pracuje s derivací jasové funkce, jelikož gradient nám udává změnu tvaru funkce.

Gradient je vektor, který určuje směr a strmost růstu funkce. Pro spojitou obrazovou funkci $f(x,y)$ je velikost gradientu $|\nabla f(x,y)|$ a jeho směr ψ dány vztahy:

$$|\nabla f(x,y)| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2} \tag{13}$$

$$\psi = \arg\left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right) \tag{14}$$

Detektory hran tedy mohou pracovat s první derivací funkce. Ta nabývá svého maxima v místě hrany [8]. Nejjednodušším případem je stav, kdy derivace určíme pro jednotlivé sloupce pro každý pixel od shora dolů. Nevýhodou je zde veliká náchylnost na šum obrazu.

Poněkud spolehlivější jsou metody založené na práci s druhou derivací. Využívá se detekce průchodu nulou, jelikož druhá derivace nabývá v místě hrany nulovou hodnotu. Definujeme zde lineární Laplaceův obraz tzv. ***Laplacián***, který pro funkci $f(x,y)$ určíme jako:

$$\Delta f(x,y) = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2} \tag{15}$$

Nevýhodou metod založených na druhé derivaci je značné vyhlazení obrazu, což může znamenat ztrátu ostrých rohů a náchylnost k vytváření uzavřených smyček hran.

2.4.2 Segmentace narůstáním oblasti

Uplatňuje se v případech, kdy je obraz výrazně zašuměn a není možné využít detekci hran. Základem je rozdělení obrazu na segmenty dle zvoleného parametru ***homogenity***. Kritérium homogenity je založeno na jasových vlastnostech nebo komplexnějších způsobech popisu obrazu [3]. Pro homogenní oblasti vyžadujeme splnění dvou podmínek:

- $H(R_i) = TRUE \qquad pro \quad i = 1,2,...,I$
- $H(R_i \cup R_j) = FALSE \quad pro \quad i, j \leq 1,2,...,I \quad i \neq j \quad Ri\ sousedí\ s\ R_j$, kde

I je počet oblastí,

R_i jednotlivé oblasti,

$H(R_i)$ dvouhodnotové vyjádření kritéria homogenity.

Typickým příkladem je metoda spojování oblastí. Obraz je na počátku rozdělen na malé oblasti tzv. ***obrazové elementy***. Velikost těchto elementů jsou řádově jednotky pixelů. Dále spojujeme dvě sousední oblasti, za předpokladu splnění podmínek. Spojením těchto dvou oblastí vzniká oblast nová. Proces končí v okamžiku, kdy nejdou spojit žádné dvě sousední oblasti.

3 KLASIFIKÁTORY MINIMÁLNÍ VZDÁLENOSTI

Klasifikace je metoda, pomocí které zařazujeme rozpoznávané vzory do odpovídajících tříd, na základě určitých vlastností pro danou třídu charakteristických. Pro obraz x je dán ukazatel na příslušnou třídu y, dle funkce $y = f(x)$. Základem klasifikačních metod je rozhodující pravidlo pro zařazování. Klasifikátory minimální vzdálenosti zařazují vzory do příslušných tříd dle kritéria minimální vzdálenosti. Jedná se o jednoduchou a přímočarou formu klasifikace bez učení.

3.1 Metoda nejbližšího souseda

Je základní formou klasifikátoru minimální vzdálenosti. Každá jedna třída je charakterizována určitým typickým prvkem tzv. ***etalonem.*** Klasifikovaný objekt poté porovnáme s etalony jednotlivých tříd a vybíráme etalon našemu obrazu nejpodobnější, což znamená, že objekt zařadíme do třídy, od jejíhož etalonu má nejmenší vzdálenost. Jelikož klasifikovaný obraz i etalony jsou n-rozměrné vektory, definujeme podobnost obvykle jako Euklidovskou vzdálenost mezi nimi. Pro dvourozměrný prostor určujeme vzdálenost dvou bodů, která je vyjádřena jako:

$$d_{ij} = \sqrt{(x_{1i} - x_{1j})^2 + (x_{2i} - x_{2j})^2} \tag{16}$$

Kdy pro n-rozměrný prostor můžeme vzdálenost zobecnit na:

$$d_{ij} = \sqrt{\sum_{k=1}^{n} (x_{ki} - x_{kj})^2} \tag{17}$$

Ne vždy je použití Euklidovské metriky výhodné, její použití je závislé na konkrétním případě. Někdy je vhodnější využít ***tangentovou metriku***, která je odolnější proti geometrickým transformacím jako je rotace, posun, zkosení apod. Její využití avšak zvyšuje náročnost výpočtů algoritmu. Obdobně je tomu při nesymetrickém rozložení shluků jednotlivých tříd, kdy je výhodnější počítat ***Mahalanobisovu vzdálenost***, která při výpočtu vzdálenosti bere v úvahu rozptyl jednotlivých shluků [9].

Klasifikátor je tedy určený množinou etanolů μ_y, které reprezentují příslušnou třídu y. Hledanou třídu pro prvek x poté určíme jako:

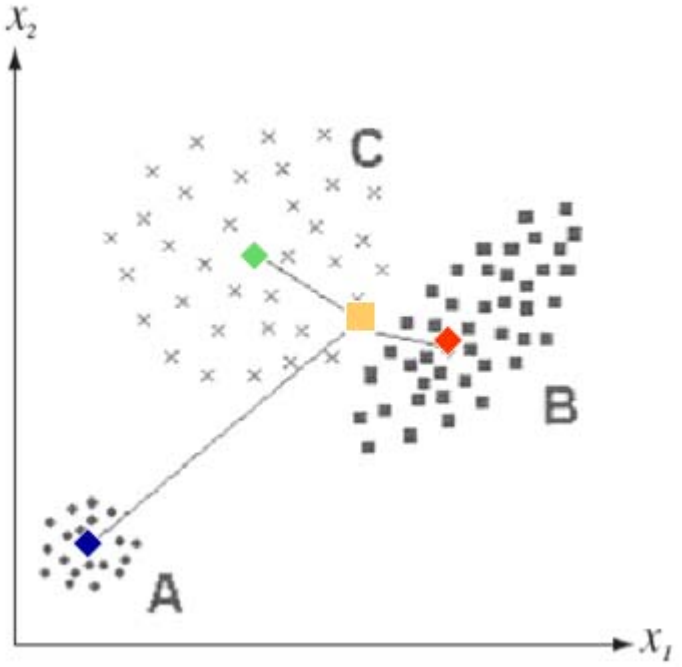
$$f(x) = \min \left\| x - \mu_y \right\|_{1,\cdots,K} \tag{18}$$

Nejdůležitějším faktorem pro správnou klasifikaci je zvolení vhodného reprezentanta třídy. Nejjednodušším způsobem zvolení etalonu je spočítáním průměrné hodnoty vstupních vzorů v jednotlivých třídách. Formálně výpočet zapíšeme jako:

$$\mu_y = \frac{1}{\left|T_y\right|} \sum_{i \in T_y} x^i \tag{19}$$

kde:

$T_y = \left\{ i \mid i \in \{1,\ldots,m\} \wedge y^j = y \right\}$ značí množinu trénovacích vzorů třídy y [10].



Obr. 6. Příklad klasifikace dle minimální vzdálenosti.

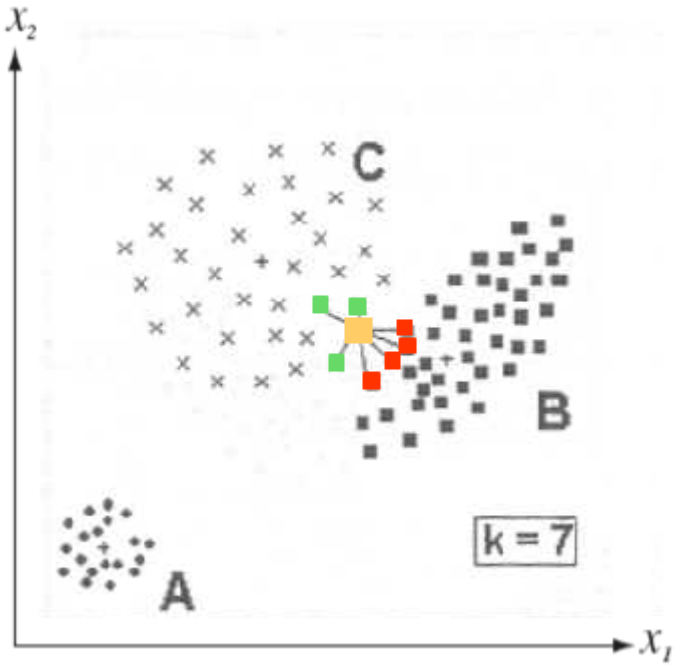
Hlavní výhodou této metody je jednoduchost a rychlost klasifikace. Pokud prvky jednotlivých tříd tvoří symetrické shluky, vzájemně snadno oddělitelné, tak můžeme jednotlivé třídy příslušným etalonem efektivně popsat (Obr. 6). Klasifikace poté dosahuje vysoké úspěšnosti. Problém nastává při nerovnoměrném rozdělení prvků jednotlivých tříd,

kdy je nemožné efektivně rozdělit shluky jednotlivých tříd pomocí jediného etalonu. U těchto případů pak dochází ke značné chybě klasifikace.

3.2 Metoda k-nejbližších sousedů

Je určitou modifikací předchozí metody, kdy je z trénovací množiny vybrán větší počet reprezentantů. Tito reprezentanti by měli co nejlépe charakterizovat danou třídu. Pro klasifikaci je nutné vypočítat vzdálenost rozpoznávaného vzoru od všech reprezentantů [9]. Poté najdeme k-počet nejbližších reprezentantů zadaného vzoru a zařadíme jej do třídy, která má mezi těmito reprezentanty maximální zastoupení. Může však nastat situace, kdy dojde k shodě v počtu zastoupených reprezentantů, což má za následek neschopnost klasifikace.

Řešením je odlišná verze této metody, kdy jsou postupně procházeni nejbližší reprezentanti až do doby, dokud není nalezen k-počet reprezentantů stejné třídy. Vzor je poté přiřazen právě této třídě. Velikost parametru k má podíl na úspěšnosti kvalifikace, ale také na zvýšení náročnosti celého algoritmu. Jeho volba je tedy zásadní, avšak závisí především na charakteru konkrétní množiny prvků.



Obr. 7. Ukázka metody k-nejbližších sousedů.

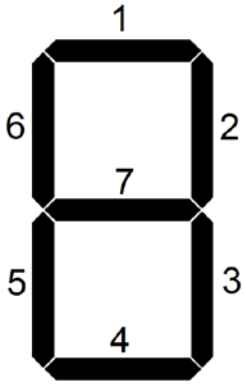
Parametr k by měl nabývat hodnot v rozsahu od 1 do 10. Při vyšších hodnotách než 10 se na výsledku objevuje šum [11].

Tato metoda dosahuje oproti předchozí v určitých případech vyšší úspěšnosti rozpoznání, avšak dochází zde k narůstání množství zpracovávaných informací, což má za následek zpomalení celého algoritmu. Jelikož se jedná o metodu bez učení, v průběhu klasifikace již není možné zlepšovat dané výsledky.

3.3 Využití

Metody klasifikace minimální vzdálenosti, především její modifikace, mají velmi široké pole působnosti v různých oborech lidské činnosti. Jedním z nich je i počítačové vidění, kde jsou využívány v systémech rozpoznávání znaků [11]. Pro potřeby rozpoznání ručně psaných znaků, však není tato metoda úplně ideální, ale v určitých konkrétních případech může dosahovat uspokojivých výsledků. Základem pro úspěšnou klasifikaci je kvalitní předzpracování vstupních znaků, jelikož tato metoda není příliš odolná proti jakýmkoliv tvarovým změnám, jako jsou posunutí, zkosení, a také zašumění jednotlivých znaků. Pro názornost naznačím základní princip implementace metody nejbližšího souseda na rozpoznání znaků.

Pro jednotlivé etalony, které reprezentují jednotlivé klasifikační třídy, tedy přímo i jednotlivé rozpoznávané znaky, je potřeba vhodně zvolit jednotlivé příznaky. Ty je nutné volit tak, abychom z nich získali informace o oblastech, které jednotlivé znaky odlišují. Již toto je velmi obtížné, a tak budeme pro zjednodušení uvažovat množinu rozpoznávaných znaků pouze arabské číslice 0 až 9. Pro nastínění základního principu této metody si zjednodušíme i tvar těchto číslic, který budeme uvažovat v podobě digitální číslice (Obr. 8).



Obr. 8. Rozdělení segmentů digitálních číslic.

Tímto rozvržením získáme 7 příznaků, shodných s jednotlivými segmenty digitální číslice. Etanoly jsou poté vektory, které nesou informaci o stavu příznaku pro jednotlivé znaky, v tomto případě pouze binární informaci, je-li daný příznak aktivní. Konkrétně etanol pro znak 0 bude vektor (1,1,1,1,1,1,0), pro znak 1 poté bude (0,1,1,0,0,0,0) atd.

Samotný postup klasifikace je násldující. Nejprve je pro klasifikovaný znak určen příznakový vektor, extrakcí zadaných příznaků. Od tohoto vektoru jsou odečteny etanoly jednotlivých tříd. Následně určíme absolutní hodnotu rozdílu vektorů, což opakujeme pro všechny třídy. Z těchto absolutních hodnot je vyhledána minimální hodnota, podle které zařadíme znak do příslušné třídy. V ideálním případě je hodnota rozdílu vektoru klasifikovaného znaku s vektorem reprezentanta třídy rovna 0, což znamená, že jsou tyto znaky totožné. Problémem u této metody je častá shoda minimálních vzdáleností pro více etanolů a nemožnost poté určit vhodnou třídu pro daný znak.

Pro komplexnější rozpoznání je tedy potřeba tuto metodu poněkud inovovat, avšak základní princip zde nastíněný zůstává nezměněn. Důležité je tedy vhoné rozložení vstupního znaku na příslušné příznaky dle jejich charakteru tak, abychom co nejlépe popsali tvarové rozdíly zástupců jednotlivých tříd. Na druhou stranu je nutné uvažovat o výpočetní náročnosti celé operace, tak aby etanol zůstal pokud možno co nejjednodušší.

4 NEURONOVÉ SÍTĚ

Neuronové sítě vycházejí svojí podstatou z biologických struktur. Neuronové sítě můžeme rozdělit na biologické neuronové sítě, které tvoří základ všech částí biologického informačního systémů. Základním stavebním prvkem je neuron. Příkladem biologického systému může být například lidský mozek, který tvoří jednu z nejsložitějších soustav vůbec. Tato struktura obsahuje až 100 miliard navzájem propojených neuronů, vzájemně na sebe působících. Tato struktura a její vzájemné vazby se neustále vyvíjí.

Definice neuronové sítě není jednoznačná, jedna z interpretací je následovná: „Neuronová síť je systém sestávající z mnoha jednoduchých procesů, pracujících paralelně, jejichž funkce je determinována strukturou sítě, intenzitou propojení a zpracováním ve výpočetních elementech nebo uzlech [12].“

4.1 Umělé neuronové sítě

Umělé neuronové sítě (*Artificial Neural Network*) jsou matematickými modely biologických neuronových sítí. Funkci UNS lze tedy chápat jako transformaci f vstupního signálu u na výstupní signál $y = f(u)$. Formálně definujeme umělou neuronovou síť jako orientovaný graf s dynamicky ohodnocenými hranami a vrcholy, nebo-li uspořádanou pětici $[V, E, \varepsilon, w, y]$, kde:

V je množina vrcholů (neuronů),

E množina hran (synapsí),

ε zobrazení incidence hran s vrcholy ($\varepsilon : E \to V \times V$),

w dynamické ohodnocení hran ($w : \varepsilon(E) \times T \to R$),

y dynamické ohodnocení vrcholů ($y : V \times t \to R$),

T, t časové osy [13].

Základním stavebním prvkem UNS je také neuron, nejrozšířenějším modelem je tzv. formální neuron (Obr. 9). Neuron zpracovává vstupní údaje podle vztahu [14]:

$$y = f(\sum_{i=1}^{N} w_i x_i + \Theta) \tag{20}$$

kde:

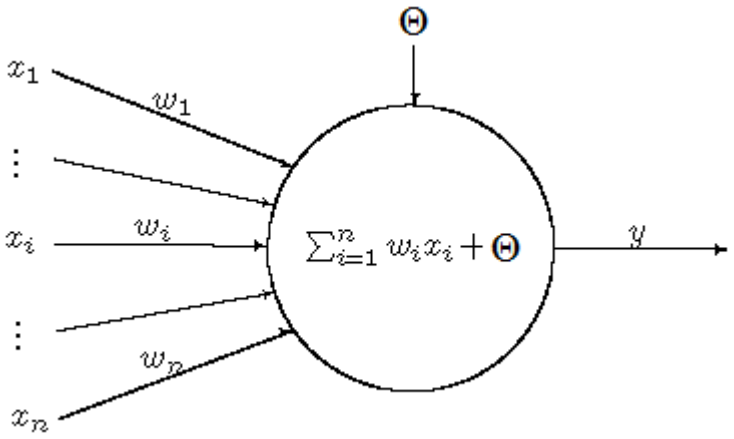
y je výstup, nebo-li aktivita neuronu,

x_i i-tý vstup neuronu,

w_i hodnota i-té synaptické váhy,

f přenosová funkce neuronu,

Θ prahová hodnota.



Obr. 9. Model McCulloch-Pittsova neuronu.

Pro jednotlivé vstupy x_i představuje w_i příslušnou synaptickou váhu, která je vyjádřením zkušeností neuronu. Při učení hledáme odpovídající hodnotu vektoru synaptických vah, abychom na výstupu získali požadovanou hodnotu. Prahová hodnotaΘ nám určuje, kdy bude neutron aktivní. Neutron je aktivní tehdy, pokud je vážená suma všech vstupů vyšší než prahová hodnota [12].

4.1.1 Přenosová funkce

Aktivační, neboli přenosová funkce neuronu, označená f (rovnice 20) může mít různý tvar. Obecně tato funkce bývá lineární, spojitá nebo diskrétní. Nejčastější přenosové funkce používané v modelech neuronových sítí jsou následující:

sinoida $$f(u) = \frac{1}{1+e^{-u/T}} \tag{21}$$

hyperbolická tangenta $$f(u) = \tanh\left(\frac{u}{T}\right) \tag{22}$$

skoková funkce $$f(u) = \begin{cases} 1 & u \geq 0 \\ 0 & u < 0 \end{cases} \tag{23}$$

lineární $$f(u) = au + b \tag{24}$$

prahová funkce $$f(u) = \begin{cases} u & u \geq 0 \\ 0 & u < 0 \end{cases} \tag{25}$$

Výběr funkce závisí na typu úlohy, již hodláme řešit. Také na náročnosti výpočtů a času vyhrazeného trénování těchto sítí. Neuronové sítě se mohou nacházet ve dvou fázích [14]. V adaptivní fázi se síť učí, v aktivní vykonává svoji činnost - vybavuje.

4.1.2 Učení neuronové sítě

Učení se provádí nastavováním jednotlivých synaptických vah a prahů. Na počátku se nastaví startovací hodnoty, které mohou být náhodně zvolené. Poté se na vstup přivede tzv. trénovací vstup. Neuronová síť poskytne výstup a odezvu. Zde se učení dělí na dva základní typy.

Učení s učitelem, kdy se síť učí srovnáním aktuálního výstupu s požadovaným výstupem. Poté změní nastavení synaptických vah tak, aby se rozdíl mezi aktuální a požadovanou výstupní hodnotou snížil. Tento postup se provádí pomocí učícího algoritmu. Učení obvykle probíhá do doby, než je splněna určitá podmínka. Příkladem může být pokles hodnoty globální chyby pod stanovenou mez.

Učení bez učitele, kdy je celé vyhodnocování založeno pouze na informacích získaných během procesu učení. Není zde rozhodovací kritérium, algoritmus hledá určité společné vzory ve vstupních datech.

4.1.3 Využití

Jak je zřejmé, neuronové sítě mají široké pole působnosti. Jejich využití je v úlohách predikce, optimalizace, aproximace, asociace a rozpoznání. Nás v souvislosti s rozpoznáním ručně psaného textu zajímá oblast rozpoznávání, neboli klasifikace, kdy jsou jednotlivé vstupní vektory vyhodnocovány a zařazovány do příslušných tříd.

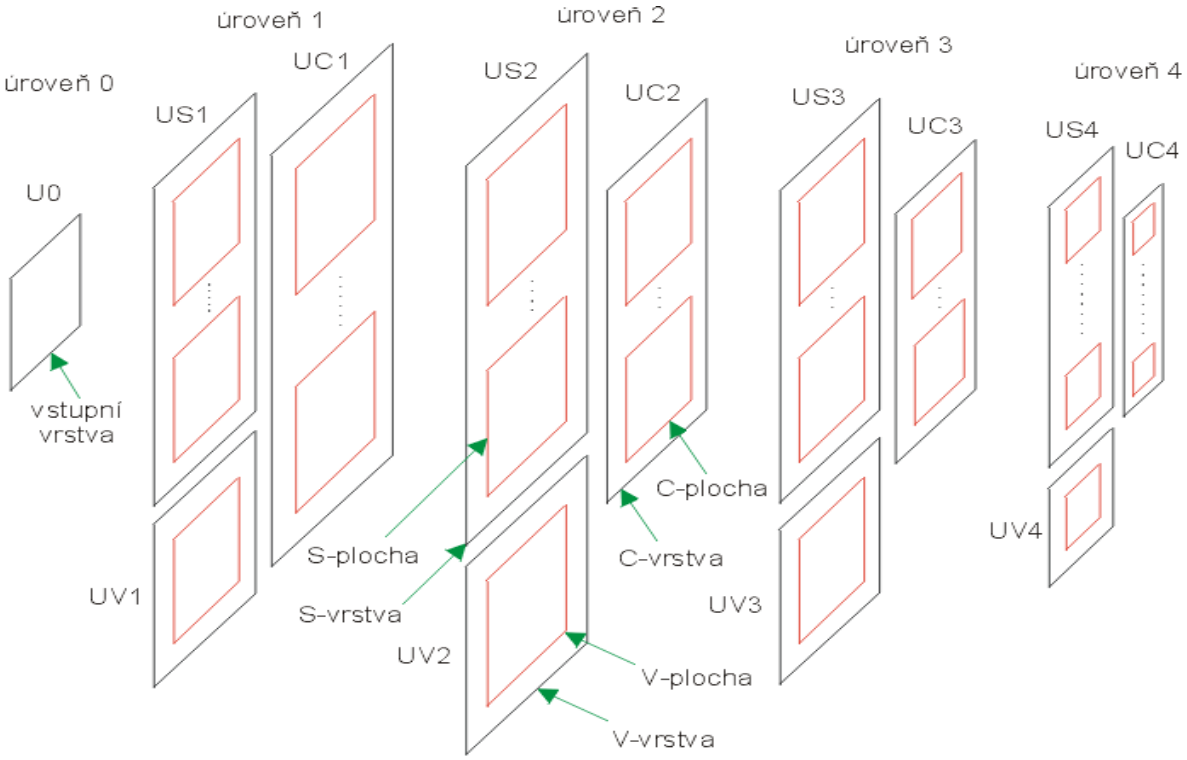
4.2 Neocognitron

Neocognitron je typ vícevrstvé hierarchické neuronové sítě, které se využívá pro rozpoznávání ručně psaných znaků. Síť navrhl japonský profesor ***Kunihiko Fukushima*** v roce 1979. Během dlouhé doby od jejího uveřejnění, prošla síť mnohými modifikacemi a dnes existuje několik různých verzí. Původní verze byla bez učitele, později však byla vytvořena verze s učitelem, pro kterou je nutno vytvořit speciální soubor učících znaků [14]. Dále se budu zabývat právě touto modifikovanou verzí s učitelem.

4.2.1 Struktura sítě

Jak bylo uvedeno, jedná se o hierarchickou síť. Tato hierarchie spočívá v tom, že v nižších úrovních síť detekuje nejjednodušší příznaky. S každou další úrovní jsou tyto příznaky složitější. Jednotlivé úrovně obsahují vždy tři vrstvy: S-vrstvu, C-vrstvu, V-vrstvu. Výjimkou je pouze nultá úroveň obsahující pouze vstupní vrstvu, která ukládá vstupní informace. Jednotlivé vrstvy se skládají z určitého počtu ploch, které jsou dle příslušné vrstvy buď S, C nebo V. Jednotlivé plochy tvoří dvojrozměrné pole buněk[1]. Síť Neocognitron obsahuje čtyři základních typy buněk: S-buňky, C-buňky, V-buňky a receptorové buňky. Základní stavební prvky, tedy jednotlivé buňky, již pracují s reálnými nezápornými hodnotami.

[1] V literatuře se místo pojmu neuron používá pojem buňka [14].



Obr. 10. Struktura sítě Neocognitron.

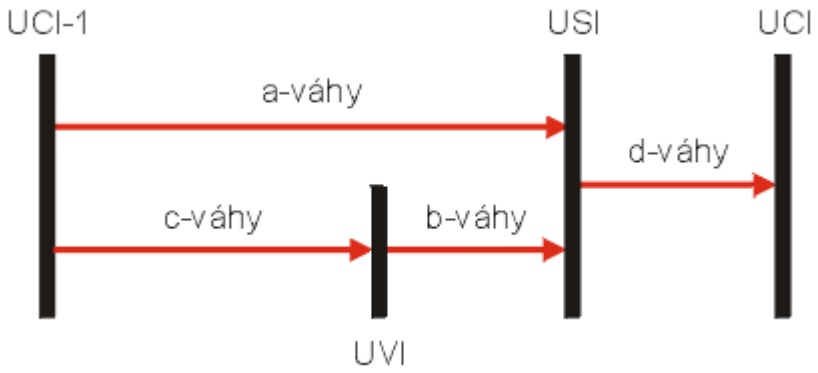
Všechny plochy obsažené v jedné vrstvě mají stejně veliké pole buněk. Stejně tak jsou si rozměrově totožné V-plochy a S-plochy stejné úrovně. Příklad počtu ploch a příslušných buněk pro vrstvy jednotlivých úrovní je zobrazen v následující tabulce (Tab. 1). Tento konkrétní příklad je pro klasickou aplikaci neuronové sítě Neocognitron na rozpoznávání ručně psaných znaků, konkrétně číslic 0 až 9 a písmen velké abecedy A až Z.

Tab. 1. Počet buněk a ploch pro jednotlivé vrstvy u reálné aplikace neuronové sítě Neocognitron [14].

Vrstva	Rozměr pole buněk	Počet ploch	Počet buněk
U0	19 x 19	-	361
US1	19 x 19	12	4 332
UV1	19 x 19	1	361
UC1	21 x 21	8	3 528
US2	21 x 21	80	35 280
UV2	21 x 21	1	441
UC2	13 x 13	33	5 577
US3	13 x 13	97	16 393
UV3	13 x 13	1	169
UC3	7 x 7	64	3 136
US4	3 x 3	46	414
UV4	3 x 3	1	9
UC4	1 x 1	35	35
Celkem	-	380	70 045

4.2.2 Propojení sítě

Sítě Neocognitron se vyznačují vysokou hustotou spojů mezi jednotlivými buňkami. Každá buňka je spojena se skupinou buněk, tzv. ***připojovací oblastí***, předchozí vrstvy. Tato připojovací oblast má nejčastěji rozměr 3x3 nebo 5x5 buněk. Buňky vstupní vrstvy nebo C-vrstvy předchozí úrovně jsou propojeny s buňkami na S-vrstvě a V-vrstvě. Dále jsou jednotlivé V-buňky spojeny s S-buňkami příslušné vrstvy. Nakonec jsou S-buňky spojeny s C-buňkami. Každé propojení v síti má určitou váhu. Zde existují čtyři typy vah: a-váhy, b-váhy, c-váhy a d-váhy.



Obr. 11. Váhy jednotlivých spojení.

a-váhy

Jsou modifikovány učením. Spojují připojovací oblast C-ploch předešlé úrovně s příslušnými S-buňkami následující úrovně. Váhy jsou sdílené, což znamená, že všechny S-buňky v jedné S-ploše mají stejnou hodnotu a-váhy.

b-váhy

Také jsou modifikovány učením. Spojují V-buňky s příslušnými S-buňkami. B-váhy jsou také sdílené, takže všem S-buňkám stejné S-plochy přísluší stejná hodnota b-váhy. K jedné S-buňce přísluší pouze jedna b-váha, jelikož je počet V a S-buněk v jedné ploše totožný.

c-váhy

Hodnota c-vah je určena pevně při konstrukci sítě. Spojují připojovací oblast C-buněk předchozí úrovně s V-buňkami úrovně následující. Tyto váhy bývají nejčastěji nastaveny tak, že nejméně omezují přenos informací ze středu připojovací oblasti a postupně k jejím okrajům tento přenos tlumí [15].

d-váhy

Jsou spojením mezi připojovací oblastí S-buněk a příslušnou C-buňkou. Váhy jsou pevně nastaveny při konstrukci a stejně jako c-váhy tlumí přenos informace směrem k okrajům připojovací oblasti.

4.2.3 Buňky

Jak již bylo zmíněno, v neuronových sítích Neocognitron jsou čtyři druhy buněk. Prvními a nejjednoduššími buňkami jsou ***receptorové buňky***, které se nachází ve vstupní vrstvě. Jejich funkcí je uchovávat informaci o jednom pixelu vstupního obrazu. Počet receptorových buněk nám tady udává maximální možné rozlišení vstupního vzoru.

S-buňky

Dalším druhem buněk jsou S-buňky. Jejich úkolem je detekce příznaků na předem daných pozicích ve vrstvě. Excitační informaci získává každá S-buňka ze svých připojovacích oblastí C-ploch předešlé úrovně a dále získává také inhibiční informaci od příslušné V-buňky. Tato informace udává průměrnou aktivitu v příslušné oblasti.

Formálně se výstupní hodnota S-buňky vypočítá dle vztahu:

$$u_{S\ell}(n,k) = r_l(k) \cdot \varphi\left[\frac{1+\sum_{\kappa=1}^{K_{C\ell-1}}\sum_{v\in Al} a_\ell(v,\kappa,k)\cdot u_{C\ell-1}(n+v,k)}{1+\frac{r_\ell(k)}{1+r_\ell(k)}\cdot b_\ell(k)\cdot u_{V\ell}(n)}\right] \tag{26}$$

kde:

ℓ je číslo úrovně,

n souřadnice buňky,

k číslo plochy v rámci vrstvy,

v souřadnice buňky v rámci připojovací oblasti,

$K_{C\ell}$ počet C-ploch v C-vrstvě,

A_ℓ připojovací oblast S-buňky,

r_ℓ selektivita,

φ prahová přenosová funkce (rovnice 25),

a_ℓ a-váha,

b_ℓ b-váha,

$u_{V\ell}$ výstupní hodnota V-buňky.

Důležitým je zde především parametr r_ℓ, neboli ***selektivita***. Její velikost lineárně ovlivňuje význam inhibiční vstupní složky, čímž se ovlivňuje schopnost rozlišovat deformované vzory. Při vysoké hodnotě selektivity se zvýší přesnost rozpoznání, avšak sníží se schopnost reagovat na odlišnější vzory. Naopak snižováním selektivity roztřídíme i deformované vzory, avšak sníží se přesnost roztřídění. Každá S-vrstva může mít specifickou hodnotu selektivity. Správné nastavení této hodnoty je nezbytné pro optimální funkci neuronové sítě.

V-buňky

Hlavním úkolem je předat informaci o průměrné aktivitě připojovací oblasti C-ploch, příslušné S-buňce. Výstupní hodnota V-buňky je popsána vztahem:

$$u_{V\ell}(n) = \sqrt{\sum_{\kappa=1}^{K_{C\ell-1}} \sum_{\nu \in A\ell} c_\ell(\nu) \cdot u_{C\ell-1}^2(n+\nu, \kappa)} \tag{27}$$

kde:

c_ℓ je c-váha.

C-buňky

Úkolem C-buněk je zajistit odolnost proti natočení a posunutí vzorů. Aktivita buňky je přímo úměrná d-váze a hodnotě S-buněk v připojovací oblasti. Připojovací oblasti jednotlivých C-buněk na S-vrstvě se překrývají, takže jedna S-buňka ovlivňuje více C-buněk. Tímto je zaručena určitá odolnost, jelikož výsledkem C-plochy je rozmazaný obraz vzoru z S-plochy. Formálně je výstupní hodnota C-buňky popsána vztahem:

$$u_{C\ell}(n,k) = \psi\left[\sum_{\kappa \in K_{S\ell}} \sum_{\nu \in D\ell} d_\ell(\nu) \cdot u_{S\ell}(n+\nu, \kappa)\right] \tag{28}$$

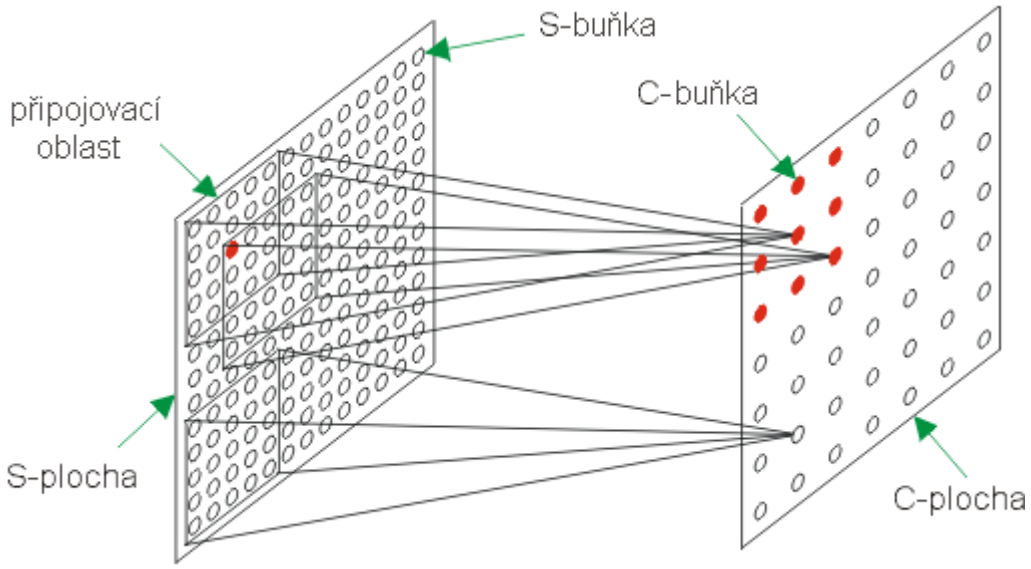
kde:

d_ℓ je d-váha,

$K_{S\ell}$ počet S-ploch v S-vrstvě,

D_ℓ připojovací oblast C-buňky,

ψ přenosová funkce.



Obr. 12. Ukázka překrytí připojovací oblastí C-buněk.

4.2.4 Učení Neocognitronu

Učení spočívá v předkládání trénovacích vzorů a nastavování modifikovaných vah (a-vah, b-vah) tak, aby síť byla schopna úspěšně rozpoznat tyto vzory. Pro učení je tedy potřeba vytvořit trénovací vzory, které by měly přímo odpovídat rozpoznávaným znakům.

Na začátku učení jsou vynulovány veškeré a-váhy a b-váhy. Učení probíhá od nejnižších vrstev. Nejprve se tedy učí jednotlivé S-plochy S-vrstvy první úrovně. Ve vybrané S-ploše zvolíme jednu buňku tzv. ***semínko*** (*seed*) a na vstup se vloží požadovaný vzor. Pro semínko určíme příslušné váhy. Jelikož jsou váhy v jedné S-ploše sdílené, automaticky se dle vah semínka nastaví ostatní buňky S-plochy. Obdobně pokračujeme pro další S-plochy. Poté co jsou naučeny všechny S-plochy dané S-vrstvy, pokračuje se na S-vrstvu vyšší úrovně.

Pro každou S-plochu je obvykle jeden trénovací vzor, ale může jich být i více. Pro každou úroveň obsahuje trénovací vzor jen určitý příznak, charakteristickou část daného vzoru, kterou bude síť v daném kroku rozpoznávat. Příznaky by měly být nastaveny tak, aby pro danou úroveň detekovaly to, co jednotlivé znaky rozděluje.

$$\Delta a_\ell(v,\kappa,\hat{k}) = q_\ell \cdot c_\ell(v) \cdot u_{C\ell-1}(\hat{n}+v,k) \tag{29}$$

$$\Delta b_\ell(\hat{k}) = q_\ell \cdot u_{V\ell}(\hat{n}) \tag{30}$$

kde:

q_ℓ je učící koeficient,

$\hat{n}$ souřadnice semínka,

$\hat{k}$ číslo plochy, kde je semínko [15].

4.2.5 Vybavování Neocognitronu

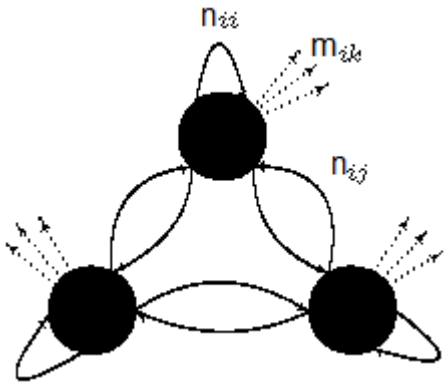
Vybavování je samotným procesem rozpoznávání předložených vzorů. Výsledkem je určení kategorie, do které daný vzor patří. Nejprve tedy vložíme vzor na receptorové buňky vstupní vrstvy. Následně se určí hodnota V-buněk V-vrstvy první úrovně. Poté mohou být vypočteny hodnoty S-buněk, které detekují nejjednodušší příznaky. Dále dojde k zpracování C-vrstvou, která obraz rozmaže a poté se opět postup analogicky opakuje ve vyšší vrstvě. Výstupem C-buněk nejvyšší vrstvy je míra podobnosti předloženého vzoru s kategorií, již daná C-buňka reprezentuje.

5 SKRYTÉ MARKOVOVY MODELY

Využití Markovových modelů má dlouhou historii. Poprvé, na počátku minulého století, použil ruský matematik Andrej Markov tento typ stochastického modelu pro analýzu charakteru posloupnosti. Po něm nesou tyto stavové modely svůj název. Skryté Markovovy modely (*Hidden Markov Models*) byly popsány v šedesátých letech Leonardem Baumem a jeho spolupracovníky. Tyto modely byly zpočátku používané pro rozpoznání řeči. Od 70. let prošel tento vědní obor značným vývojem a dnes HMM v oblasti rozpoznání řeči jednoznačně dominují. Postupem času začaly HMM pronikat i do dalších oborů jako je bio informatika a rozpoznání obrazů. Od 90. let se začalo s výzkumem pro rozpoznání ručně psaného textu. Během několika posledních let došlo ke značnému pokroku a dnes nachází HMM uplatnění při detekci on-line i off-line rukopisů a mají slibné vyhlídky do budoucnosti [16].

5.1 Princip

Skrytý Markovovův model je pravděpodobnostní stavový model s konečným počtem stavů, který přechází mezi jednotlivými stavy na základě pravděpodobnosti. Zvenčí není možné přesně zjistit stav, ve kterém se nachází, dostáváme však informaci o výstupu, který nastane s určitou pravděpodobností. Pro každý krok je na výstupu určitý symbol. Pro každý stav je dána pravděpodobnost pro výskyt onoho konkrétního symbolu na výstupu [9].



Obr. 13. Schematické zobrazení skrytého Markovova modelu [9].

Formálně jej můžeme definovat jako uspořádanou pětici $G = (Q, V, N, M, \pi)$, kde

$Q = \{q_1, ..., q_{|Q|}\}$ je množina stavů,

$V = \{v_1, ..., v_{|V|}\}$ je množina výstupních symbolů,

$N = \{n_{ij}\}$ je přechodová matice mezi jednotlivými stavy o velikosti $|S| \times |S|$, kde každý prvek udává pravděpodobnost přechodu ze stavu q_i v čase t, do stavu q_j v čase $t+1$,

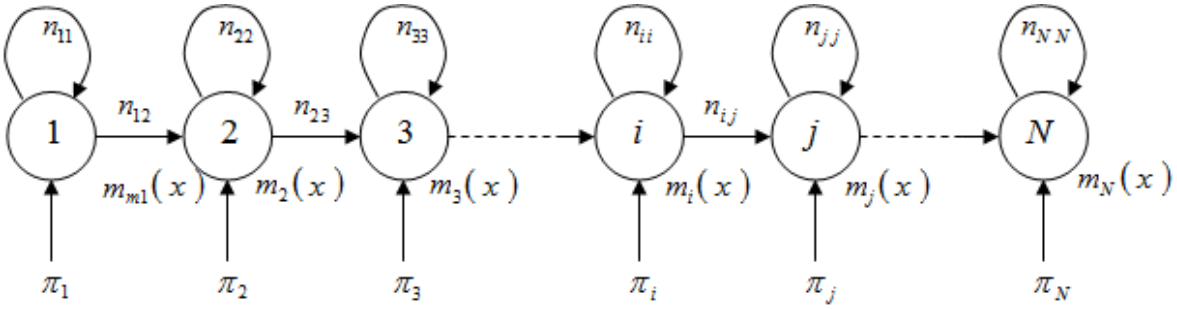
$M = \{m_{jk}\}$ je matice o velikosti $|S| \times |V|$, jejíž prvek určuje pravděpodobnost, že výstupním symbolem je v_k pro příslušný stav q_j,

$\pi = \{\pi_i\}$ je vektor, který udává pravděpodobnost, že příslušné q_i je počáteční stav.

5.1.1 Určení pravděpodobnosti

Jelikož se jedná o pravděpodobnostní model, je pro nás nezbytné určit pravděpodobnost, se kterou tento model přechází mezi jednotlivými stavy. Parametry Markovova modelu jsou trojice $\lambda = (N, M, \pi)$. Jelikož se jedná o pravděpodobnostní model musí být součet jednotlivých řádků všech parametrů roven 1, pak:

$$\sum_j n_{ij} = \sum_k m_{jk} = \sum_i \pi_i = 1 \tag{31}$$



Obr. 14. N-stavový skrytý Markovovův model.

Pokud se omezíme pouze na HMM prvního stupně, pak má na pozorování v čase t vliv pouze pozorování v čase $t-1$. Pravděpodobnost vygenerování požadovaného symbolu můžeme určit dle vzoru [17]:

$$P(O \mid \lambda) = \sum_{Q} P(O \mid Q, \lambda) P(Q \mid \lambda) = \sum_{q_1 \ldots q_T} \pi_{q_1} m_{q_1}(o_1) n_{q_1 q_2} m_{q_2}(o_2) \ldots n_{q_{T-1} q_T} m_{q_T}(o_T) \quad (32)$$

Pro výpočet pravděpodobnosti můžeme využít metodu tzv. ***dopředný výpočet***, kde pravděpodobnost, že při generování posloupnosti $\{o_1, ..., o_n\}$ se dostaneme do stavu q_i je rovna:

$$\alpha_1(i) = \pi_i m_i(o_1) \quad (33)$$

Výslednou pravděpodobnost můžeme určit rekurzivně jako:

$$P(O \mid \lambda) = \sum_{i=1}^{N} \alpha_T(i) \quad (34)$$

Další metodou je tzv. ***zpětný výpočet***, kde pravděpodobnost generování posloupnosti $\{o_1, ..., o_n\}$ a stavu q_i je dána proměnnou $\beta_1(i)$, jejíž hodnotu lze vypočítat rekurzivně dle vzorce:

$$\beta_1(i) = \sum_{j=1}^{N} n_{ij} m_j(o_{t+1}) \beta_{t+1}(j) \quad (35)$$

Výsledná pravděpodobnost je poté dána vztahem:

$$P(O \mid \lambda) = \sum_{i=1}^{N} \pi_i m_i(o_1) \beta_1(i) \quad (36)$$

V praxi se často používá k určení pravděpodobnosti ***Viterbiova algoritmu*** [18], který využívá techniky dynamického programování.

5.1.2 Učení modelu

Učení je proces nastavování parametrů $\lambda = (N, M, \pi)$. Nejběžněji používáme ***Baum-Welchův algoritmus*** [18], který hledá maximalizující parametry. Při procesu učení jsou tedy předkládány postupně vzory jednotlivých tříd. Proces učení provádíme opakovaně až do doby, kdy se hodnota změny nového a starého parametru dostane pod předem danou mez. Výpočet nových hodnot prvků jednotlivých nových matic N* a M* provádíme pomocí vztahů:

$$n^*_{ij} = \frac{\left[\sum_{s=1}^{S}\left[P\left(O^{(s)} \mid \lambda\right)\right]^{-1}\sum_{t=!}^{T_{S-1}}\alpha_t{}^{(s)}(i)n_{ij}m_j(o^{(s)}{}_{t+1})\beta^{(s)}{}_{t+1}(j)\right]}{\left[\sum_{s=1}^{S}\left[P\left(O^{(s)} \mid \lambda\right)\right]^{-1}\sum_{t=!}^{T_{S-1}}\alpha_t{}^{(s)}(i)\beta_t{}^{(s)}(j)\right]} \tag{37}$$

$$m^*{}_j(l) = \frac{\left[\sum_{s=1}^{S}\left[P\left(O^{(s)} \mid \lambda\right)\right]^{-1}\sum_{t=1,o_l=v_l}^{T_{S-1}}\alpha_t{}^{(s)}(j)\beta_t{}^{(s)}(j)\right]}{\left[\sum_{s=1}^{S}\left[P\left(O^{(s)} \mid \lambda\right)\right]^{-1}\sum_{t=1}^{T_{S-1}}\alpha_t{}^{(s)}(j)\beta_t{}^{(s)}(j)\right]} \tag{38}$$

kde:

$s = 1,\ldots,S$ je počet vzorů jedné třídy klasifikátoru.

5.1.3 Vybavování modelu

Vybavování modelu se provádí na principu maximální věrohodnosti. Pro neznámý zpracovávaný znak O jsou určeny pro všechny λ jednotlivé pravděpodobnosti $P(O \mid \lambda)$. Neznámy znak poté klasifikujeme příslušné třídě na základě maximální hodnoty příslušné pravděpodobnosti. Pravděpodobnost pro rozhodovací kritérium je určena vztahem [18]:

$$P\left(O^{(r)} \mid \lambda\right) = \prod_{s=!}^{S_r} P\left(O^{(rs)} \mid \lambda_r\right) \tag{39}$$

kde:

r je počet naučených modelů.

Z těchto pravděpodobností vybereme maximální pravděpodobnost pomocí vztahu:

$$\varpi_r - \arg\max_r P\left(O^{(x)} \mid \lambda_r\right) \tag{40}$$

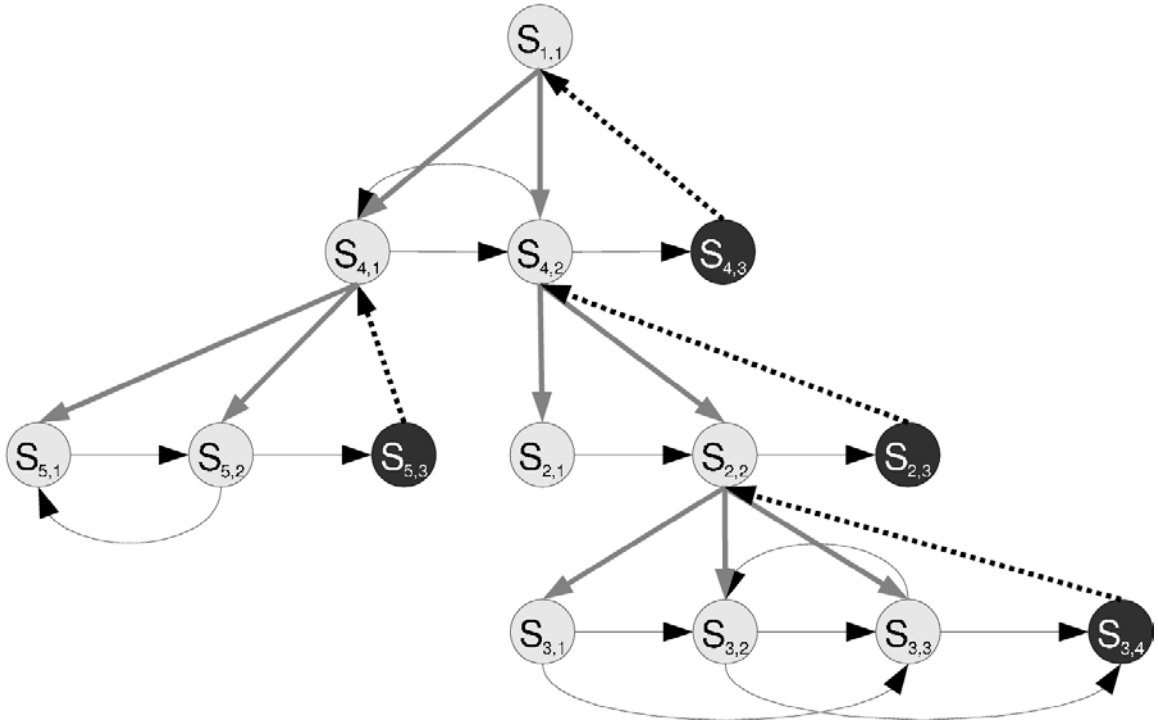
5.2 Hierarchické skryté Markovovy modely

Hierarchické HMM jsou specifickou skupinou Markovových modelů, pro něž je příznačné víceúrovňové uspořádání. Hierarchický HMM je model, kde na každé úrovni je samostatný HMM model, emitující nižší úroveň rekurzivním způsobem. Nejnižší úroveň jsou obvykle jednoduché jednorozměrné HMM. Tyto jednotlivé HMM tvoří stav pro

HMM nadřazené úrovně [9]. Výslednou hodnotou pozorování je pozorovací vektor. Pozorování v čase t je závislé na stavu proměnných v čase t na všech úrovních hierarchie $Q_t = Q_t^1 \ldots Q_t^L$, kde L je číslo nejvyšší úrovně.

5.2.1 Struktura

Čísla stavového popisu hierarchického HMM udávají počet stavů vnořených HMM. Vnořené řádky zpracovávají příslušné řádky vstupu. Nadřazené celky poté zpracovávají data ve vertikálním směru, tzn. přepínají mezi jednotlivými vnořenými HMM. U hierarchických modelů není možné přepínat mezi stavy dvou různých vnořených HMM, čímž docílíme menšího počtu spojů a relativního zjednodušení celé struktury [19]. Důležitým parametrem je právě počet úrovní, který je třeba vyvodit dle charakteru zpracovávaných dat a požadavků kladených na jejich zpracování.



Obr. 15. Příklad struktury hierarchického HMM.

Každý jednotlivý stav je tedy samostatným pravděpodobnostním modelem. Pokud je tedy stav aktivní, uskuteční svůj pravděpodobnostní model, čímž může aktivovat další stav. Tyto stavy, které přímo nevyjadřují symboly se nazývají ***vnitřní stavy***, zobrazené jsou jako světle šedé kruhy (Obr. 15). Proces se opakuje až do bodu aktivace tzv. ***výrobního stavu*** [20]. Výrobní stav je reprezentován černým kruhem (Obr. 15). To je stav, který nám

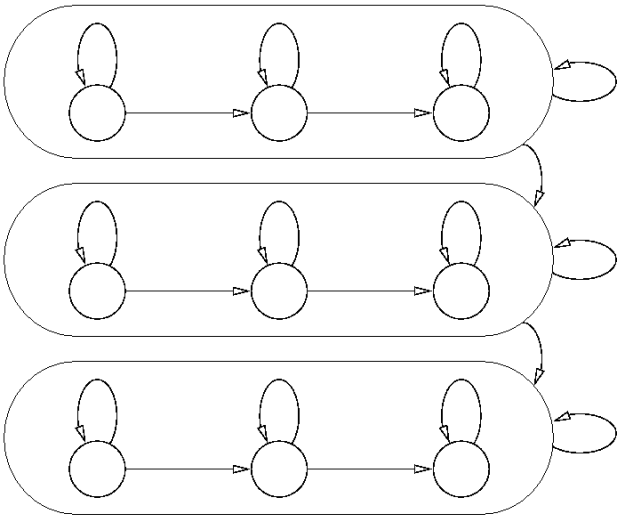
vydává pozorovatelný symbol. Aktivace vnitřních stavů se nazývá ***vertikální přechod***, který je reprezentován šedou čárou (Obr. 15). Po ukončení vertikálního přechodu dochází k tzv. ***horizontálnímu přechodu*** v rámci stejné úrovně viz. tmavá tečkovaná čára (Obr. 15). Při horizontálním přechodu dochází k vrácení se do stavu o úroveň výše v hierarchii, než byla úroveň poslední vertikálního přechodu.

5.2.2 Aplikace

Důležitou složkou pro vstup do HMM je vhodné rozvržení vstupního znaku na jednotlivé příznaky. Nejjednodušším způsobem je rozložení vstupního obrazu přímo na jednotlivé pixely, kdy příznakový vektor, zde konkrétně příznaková matice obsahuje hodnoty jednotlivých pixelů. Budeme uvažovat, že obraz prošel procesem prahování, takže prvky matice nabývají pouze binárních hodnot. I přesto je patrné, že takovýto příznakový vektor, je značně objemný. Vhodným způsobem pro HMM obecně je využití diskrétní kosinové transformace, kdy dochází ke ztrátové kompresi a tím zanedbání nepotřebných informací, čímž se značně zvýší jednoduchost a efektivita systému [20].

Abychom získali funkční klasifikátor, je potřeba náš model naučit, tzn. předložit vhodnou trénovací množinu znaků a dle ní určit přechodovou a pravděpodobnostní matici. Opět zde platí stejné pravidlo jako pro ostatní klasifikační metody, kdy je potřeba vhodně zvolit tuto množinu znaků pro jednotlivé třídy. Pro každou jednotlivou třídu se trénuje specifický model. Následné hledání třídy pro rozpoznávaný znak probíhá dle vyhledání maximální pravděpodobnosti. U hierarchických HMM můžeme jednotlivé stavy chápat jako samostatné modely, proto by mělo vyhodnocování probíhat rekurzivně [21].

Pokud budeme uvažovat nejjednodušší model umožňující klasifikovat znaky, měl by dostačovat dvouúrovňový hierarchický HMM. Na nejnižší úrovni jsou samostatné modely pracující na principu zleva doprava. Tato architektura umožňuje skok z jednotlivého stavu buď na stav vpravo nebo na sebe sama. Tyto jednotlivé HMM nejnižší úrovně zpracovávají příslušné řádky vstupního znaku a tvoří stav nadřazeného HMM, který zde slouží pro vertikální zpracování dat. Ukázka na obrázku (Obr. 16), kde první úroveň tvoří tři vnořené HMM typu zleva doprava a na druhé úrovni jeden jim nadřazený model.



Obr. 16. Dvouúrovňový hierarchický HMM.

Hierarchické HHM jsou ve skutečnosti jen zobecněním HHM, a tak mají stejně jako ony rozsáhlé pole působnosti. V praxi dosahují vysoké efektivity v oblasti rozpoznání obrazů, nevyjímaje rozpoznání ručně psaného textu, z čehož plyne i jejich využití. V rámci HMM se jedná o jednu z nejrozšířenější strukturu využívanou pro rozpoznání rukopisů [21].

II. PRAKTICKÁ ČÁST

6 ON-LINE PROGRAMY

Cílem v této části práce bylo nelézt různé programy, pomocí níž lze otestovat jednotlivé algoritmy on-line, což znamená přímo na internetu. Většinou se jedná o jednoduché Java programy, jež jsou umístěny na webových stránkách různých výukových projektů. Jejich cílem je tedy přiblížit uživateli základní principy jednotlivých metod a demonstrovat jejich praktické využití. Z technického hlediska jsou možnosti těchto programů částečně limitovány již samotnou podstatou, kterou je implementace na webové stránky.

6.1 Klasifikátor minimální vzdálenosti

I když na internetu existuje spoustu programů, pomocí kterých lze otestovat klasifikační metody minimální vzdálenosti, pro konkrétní úlohu rozpoznávání znaků jich mnoho není. Naštěstí jsem objevil jeden, který je ideální pro ozkoušení a pochopení základních mechanismů rozpoznání znaků touto metodou. Tento jednoduchý program je umístěn na adrese [22], kde je součástí výukového programu „Klasifikace a rozpoznávání“ spadající pod Katedru informatiky a výpočetní techniky Západočeské University v Plzni.

Program je určený pro rozpoznání digitálních číslic, které klasifikuje do příslušných tříd pro znaky 0 až 9. Vstupem tohoto simulátoru je grafické okno o rozměrech 5x19 pixelů, do kterého nakreslíme klasifikovaný znak. Tato bitmapa je rozdělená na 7 příznaků. Tyto jednotlivé příznaky se skládají z trojice pixelů a jsou rozvrženy tak, aby pokryly jednotlivé segmenty digitální číslice. Příslušný příznakový vektor pro rozpoznávaný znak je tedy uspořádaná sedmice čísel 0 až 3, dle počtu aktivních pixelů v oblasti jednotlivých příznaků.

Program má tři základní funkce. První z nich je extrakce jednotlivých příznaků a vypočtení příznakového vektoru pro námi nakreslený znak. Další funkcí je výpočet jednotlivých rozdílů příznakového vektoru s reprezentanty jednotlivých tříd. A konečně třetí funkci, kterou je samotná klasifikace, kdy ve vektoru vzdáleností vyhledá nejmenší hodnoty. Pokud je nejmenší hodnota jediná, tak vypíše příslušnou třídu.

Na tomto programu je velmi názorně demonstrován základní princip této metody, jež je ve své podstatě totožný i pro využití metody k rozpoznání libovolných ručně psaných znaků.

6.2 Simulátor sítě Neocognitron

Hierarchická neuronová síť Neocognitron, kterou jsem se detailně zabýval v teoretické části, je s množstvím svých modifikací jednou z nejpoužívanějších metod pro rozpoznání ručně psaného textu. Díky jejímu značnému rozšíření, vzniklo pro vědecké a studijní účely mnoho simulátorů této sítě. Vhodným zástupcem této množiny programů je simulátor, jež je součástí výukového pásma „Neuronová síť Neocognitron“, vytvořeného jako diplomová práce Tomášem Velínským, pod záštitou fakulty elektrotechnické ČVUT v Praze. Program je dostupný jako Java aplikace na [23].

Jedná se o velmi jednoduchý simulátor této sítě, který pouze nastiňuje základní principy a dovoluje pouze omezené nastavení jednotlivých parametrů. Konkrétně zde šest různých vzorů klasifikujeme do dvou tříd. Pro jednotlivé vrstvy jsou zobrazeny příslušné S-plochy a C-plochy. Při kliknutí na tyto plochy se zobrazí příslušné připojovací oblasti a receptivní pole, jež této ploše náleží. Výstupy jednotlivých ploch jsou reprezentovány různou intenzitou barvy. Pro jednotlivé nadefinované znaky nám tento simulátor zobrazí zjednodušený postup klasifikace sítě Neocognitron [15].

I přes omezené možnosti nastavení program velmi srozumitelně demonstruje celou metodu rozpoznání znaků pomocí této neuronové sítě.

6.3 Simulátor Markovova modelu

Vzhledem k velmi úzké specifikaci, kterou je využití HMM pro rozpoznání ručně psaného textu, se mi nepodařilo najít vhodný on-line simulátor HMM sloužící k tomuto konkrétnímu účelu, tedy se zaměřením na rozpoznání rukopisů. Budeme se tedy muset spokojit s programy sloužícími pro prezentaci základních všeobecných principů této metody. Jedním takovýmto příkladem je Java program, který je dostupný na [24]. Jedná se o jednoduchý simulátor Markovova modelu. Simulátor graficky zobrazuje přechody mezi stavy pro model, jež zadáme pomocí pravděpodobnostní matice. Markovovův model může být maximálně čtyř stavový. Na této aplikaci je názorně prezentován základní princip tohoto pravděpodobnostního modelu.

7 OFF-LINE PROGRAMY

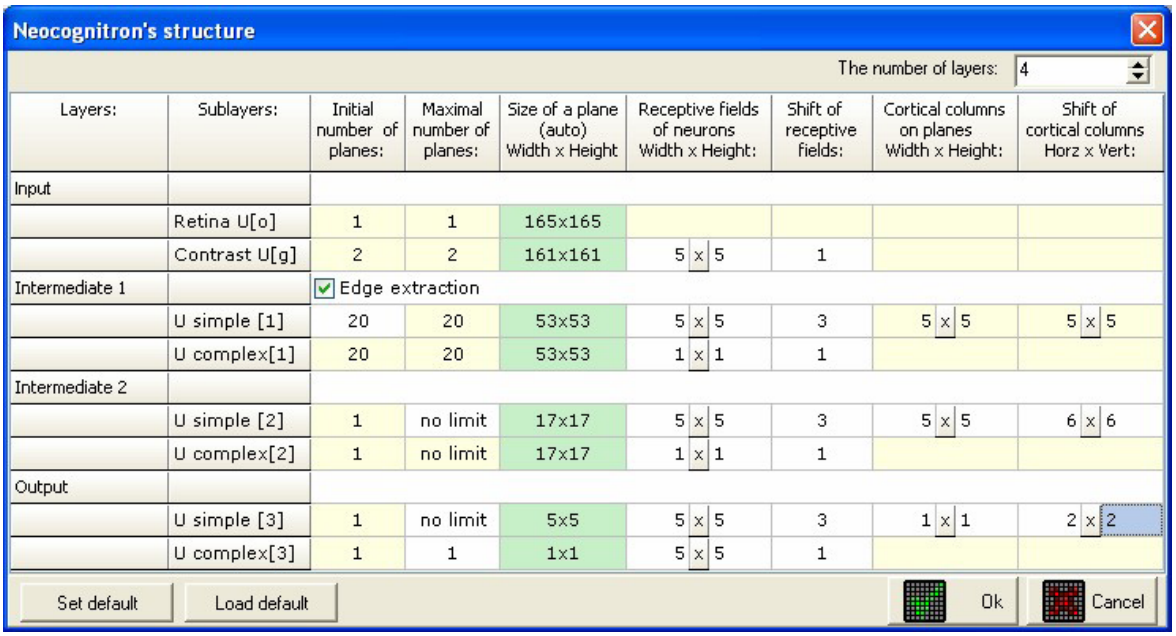
V této části jsem se měl zaměřit na off-line programy, tedy spustitelné programy, jež slouží pro testování a simulaci algoritmů pro rozpoznání ručně psaných znaků. Hledal jsem tedy na internetu různý freeware software určený pro tyto studijní a vědecké účely. Oproti programům běžících přímo na webových stránkách, jež jsem zkoumal v minulé kapitole, mají tyto programy daleko větší technické možnosti, což se logicky projeví v jejich využití, kde kromě demonstrativních účelů již částečně slouží i pro účely navrhování, zkoumání apod. Pro názornost si tedy detailněji rozebereme jeden takovýto program nazvaný ***Beholder***.

7.1 Beholder

Program Beholder verzi 2.0 vytvořil v roce 2004 Ilia Korjoukov pod záštitou The Netherlands Ophthalmic Research Institute. Softwarový simulátor je určený pro pokročilé studium neuronové sítě Neocognitronu. Poskytuje flexibilní nástroj pro práci s touto neuronovou sítí. Program může být použit bezplatně pro vzdělávací a výzkumné účely [25]. Aplikaci je možné stáhnout na [26].

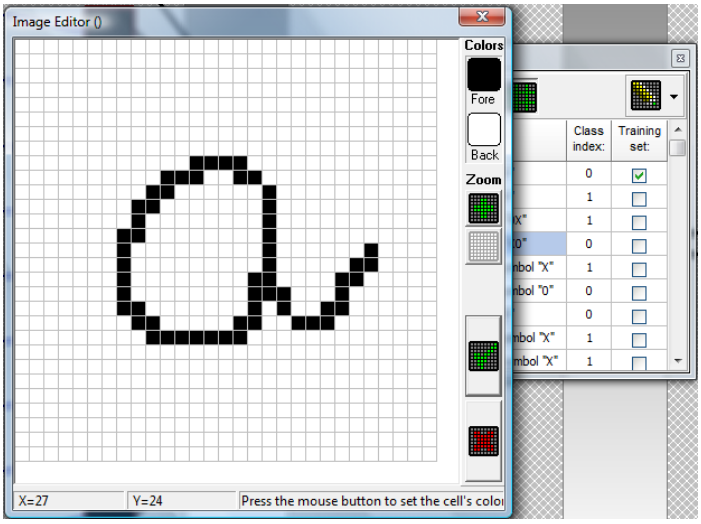
7.1.1 Popis programu

Samotný program nám umožňuje vytvářet libovolně složité topologie hierarchické sítě. Umožní nám nastavit počet vrstev, počet jednotlivých ploch ve vrstvách, rozměry jednotlivých buněk, rozměry připojovací oblasti pro každou následující vrstvu. Dále program umožňuje nastavit překrytí jednotlivých receptivních polí. Automaticky sám dopočítá zbylé parametry, včetně celkového počtu neuronů. Další možností pro tvorbu sítě je nastavení pevných C a D vah pro jednotlivé vrstvy. Dále také nastavení hodnoty selektivity a především počtu učících cyklů.



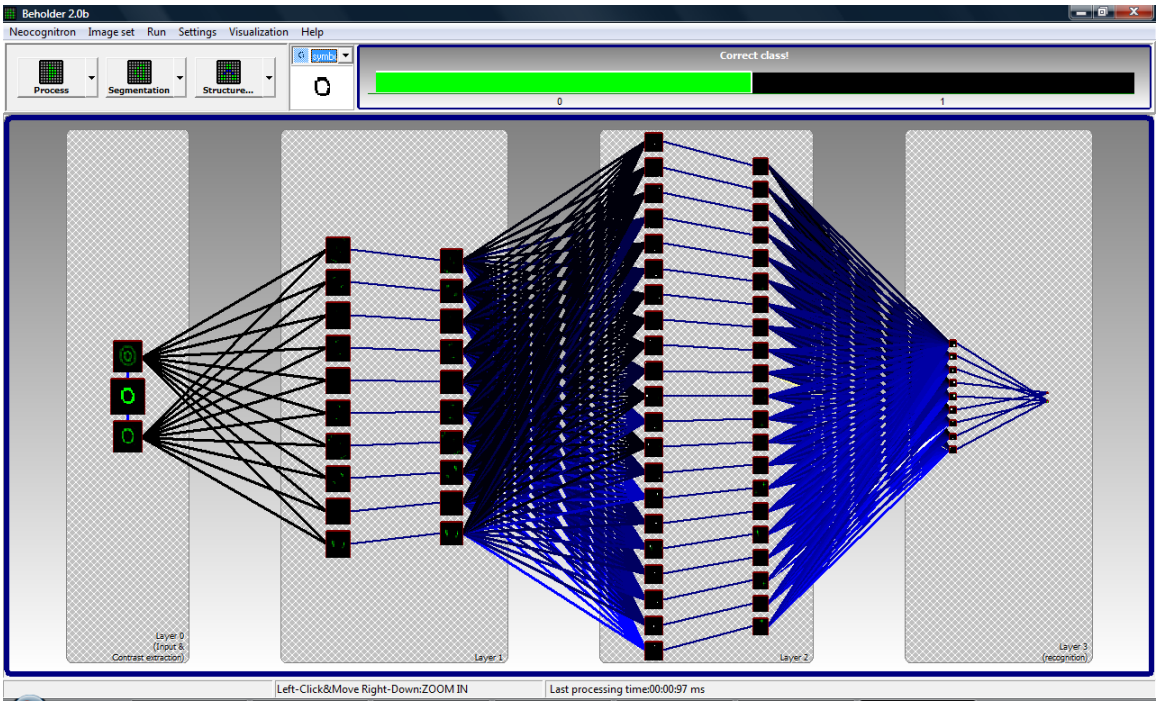
Obr. 17. Okno pro nastavení struktury sítě.

Po vytvoření struktury hierarchické sítě a zadání jejich parametrů je potřeba aplikovat trénovací množinu znaků tak, abychom neuronovou síť mohli učit tzn. nastavit modifikovatelné A a B váhy jednotlivých spojení. Pro vytvoření jednotlivých znaků slouží v programu jednoduchý grafický editor, pomocí něhož můžeme nakreslit jednotlivé znaky, které můžeme poté uložit jako samostatný soubor. Skupinu znaků lze též načíst ze samostatného souboru. Obdobně se postupuje i při tvorbě rozpoznávané množiny. Kromě používání editoru lze nahrávat znaky z externího souboru v bitmapovém formátu.



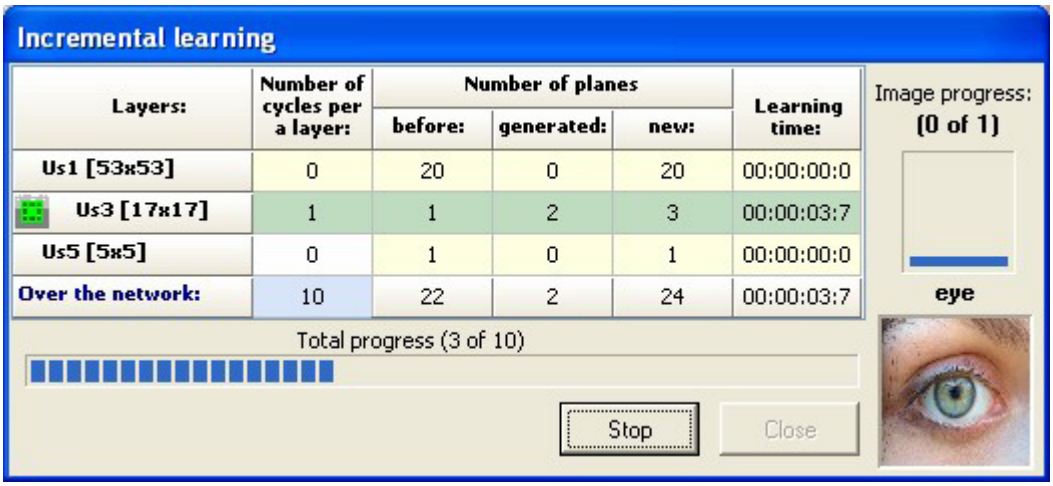
Obr. 18. Jednoduchý editor znaků simulátoru.

Další důležitou částí je vizualizace, která nám graficky zobrazuje topologii celé sítě, a také slouží jako víceúrovňový přístup k datům. Jsou zde zobrazeny jednotlivé S a C plochy, které jsou aktualizovány vždy při procesu učení či klasifikace. Dále jsou zde zobrazeny propojení jednotlivých ploch. Po kliknutí myší na požadované spojení se graficky zobrazí příslušná váha spoje, která je dána intenzitou barvy. Vizualizační okno umožňuje zoom konkrétní částí, což hojně využijeme při práci s rozsáhlejší sítí.



Obr. 19. Ukázka vizualizace sítě v prostředí Beholder v2.0.

Pro správnou funkci sítě je rozhodující proces učení. Jednotlivé trénovací vzory můžeme předkládat jednotlivě pro každou vrstvu nebo celkově pro celou síť. Pro proces učení obsahuje program speciální okno, které slouží pro nastavení parametrů učení, a také pro vizualizaci aktuálního stavu procesu, včetně času potřebného pro naučení a počtu cyklů.



Obr. 20. Okno pro proces učení [25].

Následně již můžeme síti předkládat znaky pro klasifikaci. Celý proces rozpoznání můžeme sledovat graficky pomocí vizualizace. Výsledné zařazení do jednotlivých tříd je zobrazeno zvlášť ve výsledkovém okně. Program navíc obsahuje statistické okno, kde jsou uvedeny detailní informace o aktuální síti, včetně počtu neuronů, spojení, využívané paměti apod. Toto okno je možné uložit jako xls dokument.

7.1.2 Vytvoření sítě Neocognitron

Abych detailně prozkoumal program *Beholder V2.0* a ozkoušel jsi teoretické znalosti o návrhu, učení a rozpoznávání neuronové sítě Neocognitron, vytvořil jsem v tomto simulátoru jednoduchou síť, která by měla rozlišovat ručně psané arabské číslice 0 až 9. Takováto síť by mohla sloužit např. při rozpoznání PSČ v poštovních úřadech.

7.1.2.1 Struktura sítě

Základní krokem při návrhu hierarchické sítě je vytvoření její struktury. Po rozvaze jsem se rozhodl pro čtyřvrstvou jednoduchou síť, která bude předkládané znaky klasifikovat do deseti tříd. Vstupní vrstva má rozměr 29×29 receptorových buněk, či-li vstupem do systému jsou bitmapové obrázky s rozměrem 29×29 pixelů. Následná S-vrstva obsahuje 10 S-ploch a stejný počet C-ploch má i příslušná C-vrstva. V-vrstva první úrovně obsahuje jednu V-plochu. Na druhé úrovní má S-vrstva 28 S-ploch a obdobně má 28 C-ploch i C-vrstva. Nejvyšší úroveň obsahuje 10 S-ploch a stejný počet C-ploch. Detailně uvedeno v tabulce (Tab. 2).

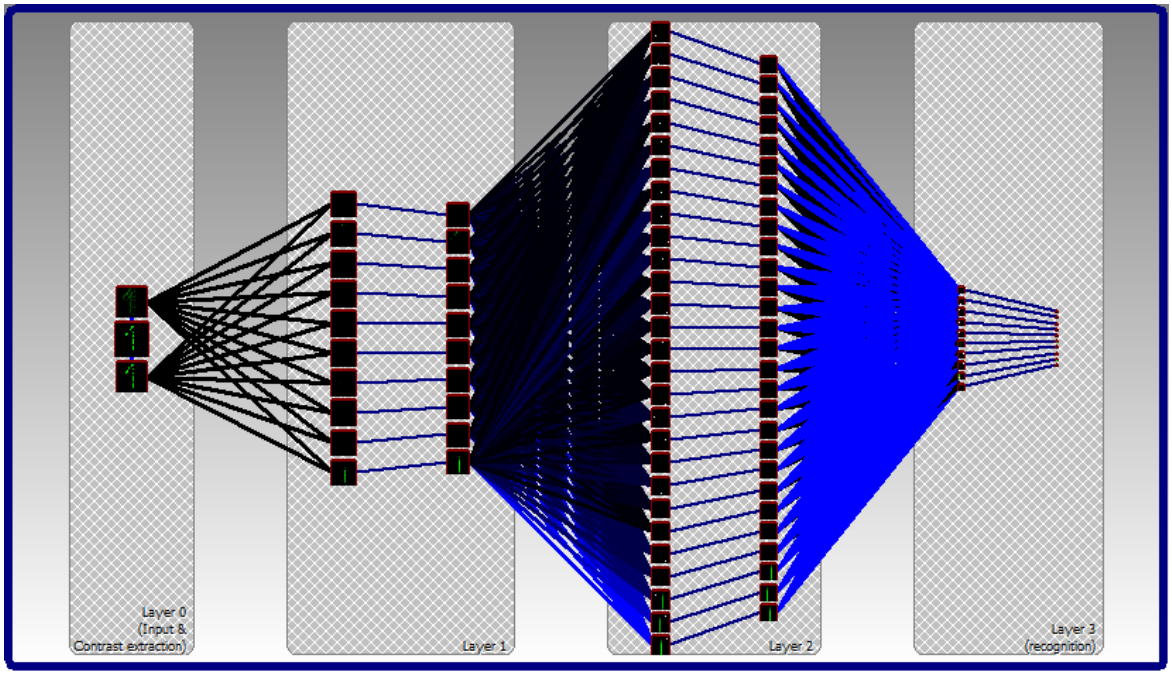
Tab. 2. Počet buněk a ploch pro jednotlivé vrstvy navrhnuté sítě.

Vrstva	Rozměr pole buněk	Počet ploch	Počet buněk
U0	29 x 29	-	841
US1	21 x 21	10	4 410
UV1	19 x 19	1	361
UC1	19 x 19	10	3 610
US2	15 x 15	28	6 300
UV2	15 x 15	1	225
UC2	13 x 13	28	4 732
US3	5 x 5	10	250
UV3	5 x 5	1	25
UC3	1 x 1	10	10
Celkem	-	99	20 764

Navrhnutá síť se tedy bude skládat přibližně z 20 tisíc neuronů, které je potřeba vhodně propojit tak, aby byla zajištěna požadovaná funkčnost celého systému.

7.1.2.2 Propojení

Důležitým faktorem při propojení je zvolení velikosti připojované oblasti pro buňky následující vrstvy. Velikost připojovací oblasti pro S-buňky na první a druhé úrovni jsem zvolil 5x5 buněk a pro S-buňky nejvyšší úrovně 9x9. Připojovací oblast C-buněk pak 3x3 pro první a druhou vrstvu a 5x5 pro vrstvu na nejvyšší úrovni. Dalším krokem je nastavení hodnoty pevně daných C a D vah, což jsou váhy spojů mezi V a C-buňkami resp. C a S-buňkami následující úrovně. Oboje váhy by měly být nastaveny tak, aby postupně tlumily informaci z připojovací oblasti od středu k jejím okrajům. Pro nastavení těchto vah jsem využil defaultního nastavení, které simulátor umožňuje. Zbylé A a B váhy jsou nastaveny při samotném procesu učení sítě.



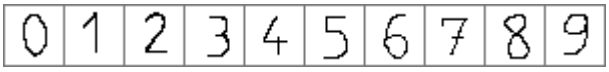
Obr. 21. Topologie navrhnuté sítě.

Posledním důležitým parametrem, který je potřeba nastavit je tzv. selektivita. Selektivita ovlivňuje velikost inhibiční části vstupu S-buněk. Při nízké hodnotě selektivity se snižuje schopnost S-buněk rozlišovat příznaky, což znamená, že i více deformované příznaky považují za správné. Naopak při vysoké hodnotě se zvýší přesnost, avšak sníží se schopnost reakce na odlišné vzory. Matematicky je selektivita dána poměrem excitační a inhibiční složky. Správné určení selektivity je tedy důležité pro úspěšnost rozpoznání. Důkladným ozkoušením jsem nakonec hodnotu selektivity pro S-vrstvy první a druhé úrovně nastavil na hodnotu 0,5 a pro S-vrstvu nejvyšší úrovně na hodnotu 0,45. V tuto chvíli máme hotovou strukturu sítě, která obsahuje 693 538 vzájemných propojení mezi jednotlivými neurony. Pro její funkčnost je potřeba síť podrobit procesu učení.

7.1.2.3 Učení sítě

Učení sítě spočívá v předložení trénovacích vzorů a dle nich nastavení modifikovatelných vah spojení tak, aby tyto vzory dokázala síť úspěšně klasifikovat do příslušných tříd. Pro jednotlivé úrovně postupně předkládáme jednotlivé vzory od jednoduchých příznaků pro nejnižší úroveň až po komplexní vzory na úrovni nejvyšší. Simulátor nám zde zjednoduší práci, jelikož stačí předložit kompletní znak pro jednotlivou třídu, který program rozloží na jednotlivé příznaky pro jednotlivé úrovně. V tomto případě

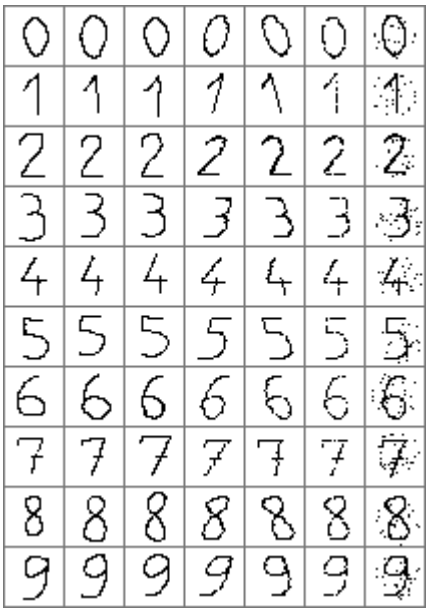
jsem pro jednoduchost zvolil pro každou třídu pouze jeden vzor. Při samotném procesu učení již pouze v simulátoru nastavíme počet cyklů.



Obr. 22. Trénovací množina pro jednotlivé třídy.

7.1.2.4 Vybavování sítě

Vybavování je již samotný proces, kdy síť klasifikuje předkládané znaky do příslušných tříd. Abych ověřil funkčnost navrhnuté sítě, vytvořil jsem množinu 70 zkušebních znaků. Tato množina obsahuje 7 znaků pro každou třídu, tedy pro každou číslici. Z těchto 7 znaků jsou 3 znaky klasicky napsány a zbylé 4 jsou podrobeny jednoduché úpravě sklonu a zašumění.



Obr. 23. Testovací množina znaků.

Testovací množinu jsem tedy po jednotlivých znacích předkládal vstupní vrstvě neuronové sítě. Výsledky klasifikace jsou uvedeny v tabulce (Tab. 3).

Tab. 3. Vyhodnocení klasifikace testovací množiny.

Číslice	**Počet správně klasifikovaných znaků**	**Počet chybně klasifikovaných znaků**	**Počet neklasifikovaných znaků**
0	7	-	-
1	7	-	-
2	5	2	-
3	5	1	1
4	3	4	-
5	5	2	-
6	4	3	-
7	4	3	-
8	3	4	-
9	4	3	-
Celkem	48 (67,14 %)	22 (31,42 %)	1 (1,43 %)

7.1.3 Zhodnocení

Neuronová síť úspěšně klasifikovala znaky přibližně v 67 % případů. Toto relativně nízké číslo je dáno několika faktory. Prvním z nich je samotná nedokonalost navrhnuté sítě Neocognitron, dáno samotnou topologií sítě a nastavením jednotlivých pevných vah a parametrů. Tomuto faktoru můžeme přičíst chybné klasifikace, které souvisí s podobností příznaků předkládaných vzorů, kdy docházelo k záměnám číslic 5 za 6, 8 za 3, 7 za 1, 9 za 0 apod. Řešení se nabízí v podobě vytvoření dokonalejší struktury a vyladěním konstant. To však vyžaduje praxi a zkušenosti s navrhováním neuronových sítí, což není přímo náplní této práce.

Při detailnějším zkoumání výsledků, však objevíme i jiné skutečnosti, které do jisté míry ovlivňují korektní klasifikaci. Největší chyba klasifikace pro vytvořenou síť Neocognitron byla pro zkosené testovací znaky, což je celkem logické, vzhledem

k posunutí detekovaných příznaků. U reálných systémů se tímto problémem zabývá tzv. preprocesní část, která upravuje zmíněné nedostatky jako zkosení, posunutí, zašumění apod. Pokud by znaky, před předložení neuronové síti, prošly některými z preprocesních metod jako např. korekcí zkosení, znamenalo by to značné zvýšení celkové úspěšnosti rozpoznání pro konkrétní množinu. Naopak při detekci posledních dvou testovacích znaků pro každou třídu, kde bylo částečně simulováno zašumění znaku, klasifikovala síť stejně jako u běžně napsaných znaků bez zašumění. Tedy lehké zašumění předkládaných znaků nepředstavuje pro síť větší obtíže v klasifikaci.

Dalším faktorem může být i omezený počet trénovacích znaků, kdy pro každou třídu byl použit pouze jeden. Po přezkoušení jsem zjistil, že v tomto omezeném případě nemá velikost trénovací množiny na funkčnost zásadní vliv. Důvodem je jednotný rukopis předkládané testovací množiny. Jinak je tomu u reálných systémů, kde je potřeba počítat se značnou rozmanitostí rozpoznávaných rukopisů. Vytvoření trénovací množiny je pak značně obtížné, jelikož ta musí obsahovat vhodný vzorec z různých typů rukopisů, tak aby dokázala najít shody v jednotlivých příznacích.

V neposlední řadě se na chybné klasifikaci podílí i samotný program, který automaticky nastavoval mnohé parametry sítě, včetně jednotlivých příznaků a pevných vah. Po detailním prozkoumání tohoto programu je patrné, že simulátor je vhodný především pro vzdělávací účely, kde se jedná o velmi názornou pomůcku. Pro návrh a vývoj neuronové sítě Neocognitron má již určité nedostatky, avšak z volně dostupného softwaru, který jsem našel na internetu, mi právě tento simulátor připadá nejvhodnější.

7.2 Zdrojové kódy

Jednou z dalších položek, kterou jsem měl vyhledat na internetu, byly zdrojové kódy programů pro příslušné typy algoritmů. Pro detailní rozbor je ideální prozkoumat právě zdrojový kód. Pro názornost se zde zmíním alespoň o některých zdrojích, kde je možné najít jednotlivé algoritmy.

Neocognitron je velmi populární neuronová síť, což se projevuje jednak rozšířením v systémech pro rozpoznání ručně psaných znaků, a také v množství informací, programů, výukových materiálů včetně filmu apod., které jsou volně dostupné na internetu. O programech zabývajících se simulací této sítě, jsem se zmínil v předešlých kapitolách. Kompletní program sloužící pro rozpoznání ručně psaných číslic napsaný v jazyce C je pro

nekomerční účely volně dostupný na webové adrese [27]. Součástí programu je i návod a celý zdrojový kód je pomocí komentářů detailně popsán. Autorem tohoto zdrojového kódu je Kunihiko Fukushima, který je i tvůrcem sítě Neocognitron. Díky tomuto kódu získáváme naprosto detailní popis celé této metody.

Klasifikátory minimální vzdálenosti, konkrétně algoritmus nejbližších k-sousedů, je na internetu zastoupen množstvím programů a zdrojových kódů. Najít implementaci tohoto kódu přímo na rozpoznání ručně psaných znaků je už poněkud obtížnější. Pro příklad zde uvedu některé webové adresy, kde je možné získat zdrojový kód tohoto klasifikačního algoritmu. Algoritmus napsaný v jazyce C++ je dostupný na [28]. Zdrojový kód naspaný pro Matlab je volně dostupný na [29].

HMM jakožto pravděpodobnostní modely mají velmi široké pole působnosti. Využívají se v burzovních analýzách, pro rozpoznání řeči, v genetice apod. Často se tedy na internetu vyskytují celé knihovny, jež sdružují jednotlivé algoritmy a metody v rámci HMM. Příklad takovéto knihovny, napsané v jazyce Java, je možné stáhnout na adrese [30]. Obdobnou nástrojovou sadu napsanou pro Matlab je možné stáhnout na [31].

Pro jednotlivé metody je možné najít značné množství zdrojových kódů napsaných v rozličných jazycích pro konkrétní klasifikační úlohy. Pro představu jsem se zmínil jen o některých z nich.

ZÁVĚR

V této práci jsem se snažil teoreticky pokrýt celý proces, který je potřebný k rozpoznání ručně psaných znaků. Základem každého systému pro rozpoznání znaků je úprava vstupních dat před klasifikací samotným algoritmem. Rozpracovávám zde techniky prahování, skeletonizace, normalizace a segmentace znaků. Kvalitní zpracování a úprava vstupních dat je velmi důležitou částí v procesu rozpoznání a velmi rapidně ovlivňuje úspěšnost samotných systémů.

Hlavní část rozpoznání závisí na použité metodě klasifikace. Nejjednodušší metody rozpoznání jsou založené na principu minimální vzdálenosti, kde je nejpoužívanější algoritmus k-nejbližších sousedů. Díky relativní jednoduchosti jsou tyto metody stále využívány v praxi. Nejpoužívanější algoritmy pro rozpoznání znaků jsou neuronové sítě, které vychází ze samotné podstaty biologických struktur a snaží se napodobit proces rozpoznání lidského mozku. Detailněji se zde zabývám neuronovou sítí Neocognitron, která je vyvinutá přímo na aplikaci rozpoznání ručně psaných znaků. Dále zde rozebírám pravděpodobnostní metody, konkrétně skryté Markovovy modely. Tyto modely jsou nejčastěji využívány k rozpoznání řeči, avšak jsou celkem úspěšně implementovatelné i na rozpoznání znaků.

Nakonec se zaměřuji na jednotlivé informační zdroje volně dostupné na internetu. Jednak prezentuji jednoduché simulátory jednotlivých algoritmů, pak také některé programy sloužící pro detailnější zkoumání algoritmů. Příkladem je program Beholder, který umožňuje vytvořit a otestovat neuronovou síť Neocognitron.

RESUME

In this work, I tried to theoretically cover the entire process which is required for handwritten characters recognition. The base of all systems for characters recognition is pre-processing of input data before classification algorithm. I analyse methods of thresholding, skeletonization, normalization and segmentation of the characters. Quality pre-processing of input data is a very important part in the detection process and it affects very significantly success of these systems.

The main part of the recognition depends on the method which is used for classification. The simplest method of detection is based on the principle of minimum distance classification, which is a typical k-nearest neighbours algorithm. These methods have been already used in practice for its simplicity. Most widely used algorithms for character recognition are neural networks based on the nature of biological structures and it tries to imitate the process of human brain. I analyse artificial neural network Neocognitron in detail. This network was developed for handwritten character recognition. Furthermore, I analyze the stochastic methods namely hidden Markov models. These models are often used for speech recognition, but they are quite successfully implemented on handwritten character recognition.

Finally, I focus on the individual information sources freely available on the Internet. First I present the simple simulator of these algorithms, then some programs serving for more detailed study of the algorithms. An example is a program named Beholder, which allows you to create and test the neural network Neocognitron.

SEZNAM POUŽITÉ LITERATURY

[1] EKVIL, Line. *Optical Character Recognition* [online]. 1993 [cit. 2010-04-24]. Dostupný z WWW: <http://www.nr.no/~eikvil/OCR.pdf>.

[2] HABIBALLA, Hashim. *Umělá Inteligence* [online]. 2004 [cit. 2010-04-24]. Dostupný z WWW: <http://www.volny.cz/habiballa/publ/umint.pdf>.

[3] ŽELEZNÝ, Miloš. *Zpracování digitálního obrazu* [online]. 2006 [cit. 2010-03-27]. Dostupné z WWW: < http://artin.zcu.cz/courses/zdo/ZDO_aktual_060217.pdf>.

[4] MICHALÍK, Marek. *Algoritmy pro rozpoznání obličeje*. Zlín, 2008. 52 s. Bakalářská práce. UTB.

[5] ŠONKA, Milan; HLAVÁČ, Václav. *Počítačové vidění*. Praha : Grada, 1992. 252 s. ISBN 8085424673.

[6] BAYER, Tomáš; ŠÍP, Marcel. *Aplikace skeletonu při automatizované kartografické generalizaci* [online]. 2008 [cit. 2010-03-27]. Dostupné z WWW: <http://gis.vsb.cz/GIS_Ostrava/GIS_Ova_2008/sbornik/Lists/Papers/035.pdf>.

[7] HLAVÁČ, Václav; SEDLÁČEK, Miloš. *Zpracování signálů a obrazů*. Vyd. 2., přeprac. Praha : Vydavatelství ČVUT, 2005. 255 s. ISBN 8001031101.

[8] KRATOCHVÍLOVÁ, Anna. *Parciální diferenciální rovnice ve zpracování obrazu* [online]. Praha, 2007. 61 s. ČVUT. Dostupné z WWW: <http://kmlinux.fjfi.cvut.cz/~oberhtom/studenti/07-kratochvilova-anna-pde-in-image-processing.pdf>.

[9] VALA, Tomáš. *Rozpoznávání SPZ z jednoho snímku*. 2006. 83 s. Diplomová práce. Univerzita Karlova, Matematicko-fyzikální fakulta.

[10] FRANC, Vojtěch; PAJDLA, Tomáš; SVOBODA, Tomáš. *Úloha - rozpoznávání číslic* [online]. 2007 [cit. 2010-05-15]. Dostupné z WWW: <http://labe.felk.cvut.cz/~tkrajnik/kui2/cviceni/lab05/X33KUI-Recognition-student-version.pdf>.

[11] IOANNIDIS, Yannis; NOVIKOV, Boris; RACHEV, Boris. *Advances in databases and information systems : 11th East European Conference, ADBIS 2007*. Berlin : Springer, 2007. 375 s. ISBN 9783540751847.

[12] TUČKOVÁ, Jana. *Úvod do teorie a aplikací umělých neuronových sítí*. Vyd. 1. Praha : Vydavatelství ČVUT, 2003. 103 s. ISBN 8001028003.

[13] KŘIVAN, Miloš. *Úvod do umělých neuronových sítí*. Praha : Nakladatelství Oeconomica, 2008. 42 s. ISBN 978-80-245-1321-8.

[14] ŠNOREK, Miroslav. *Neuronové sítě a neuropočítače.* Praha : Vydavatelství ČVUT, 2002. 156 s. ISBN 80-01-02549-7.

[15] VELÍNSKÝ, Tomáš. *Neuronová síť Neocognitron* [online]. 2000 [cit. 2010-03-22]. Dostupné z WWW: <http://neuron.felk.cvut.cz/courseware/data/chapter/velinsky2000/>.

[16] PLÖTZ, Thomas; FINK, Gernot. *Markov models for offline handwriting recognition*. Berlín : Springer, 2009. s. 269-298.

[17] BLUNSOM, Phil. *Hidden Markov Model* [online]. 2004 [cit. 2010-04-23]. Dostupné z WWW: <http://ww2.cs.mu.oz.au/460/2004/materials/hmm-tutorial.pdf>.

[18] BARDOŇOVÁ, Jana; PROVAZNÍK, Ivo; SZABÓ, Zoltán. *Rozpoznávání izolovaných slov pomocí Markovových modelů*. 2000 [cit. 2010-04-28]. Dostupné z WWW: <http://radio.feld.cvut.cz/AES/ATP2000/proc/atp00psp5.pdf>.

[19] HELLER, Katherine A. *Infinite Hierarchical Hidden Markov Models* [online]. 2009 [cit. 2010-05-01]. Dostupné z WWW: <http://www.gatsby.ucl.ac.uk/~heller/ihhmm.pdf>.

[20] BALLOT, Johan S.S. *Face recognition using Hidden Markov Models* [online]. 2005 [cit. 2010-05-16]. Dostupné z WWW: <https://etd.sun.ac.za/bitstream/10019/963/1/Ballot,%20J%20S%20S.pdf>.

[21] FINE, Shai; SINGER, Yoram; TISHBY, Naftali . *The Hierarchical Hidden Markov Model: Analysis and Applications* [online]. Sprinter Netherlands, 2004 [cit. 2010-05-01]. Dostupné z WWW: <http://www.springerlink.com/content/h7630r4u78j0xhu1/>.

[22] *Rozpoznání digitálních číslic* [online]. [cit. 2010-05-16]. Dostupné z WWW: <http://www.kiv.zcu.cz/studies/predmety/uir/KRO2/Digits/Digits.html>.

[23] *Simulátor sítě Neocognitron* [online]. [cit. 2010-05-16]. Dostupné z WWW: <http://neuron.felk.cvut.cz/courseware/data/chapter/velinsky2000/cz/priklad5.html>.

[24] *Markov System Simulation* [online]. [cit. 2010-05-16]. Dostupné z WWW: <http://people.hofstra.edu/Stefan_Waner/markov/markov.html>.

[25] KORJOUKOV, Ilia. *A Cognitive Model of Human Drawing* [online]. 2004 [cit. 2010-05-16]. Dostupný z WWW: <http://www.science.uva.nl/research/scs/papers/archive/Korjoukov2004a.pdf >.

[26] *Beholder v2.0* [online]. [cit. 2010-05-16]. Dostupné z WWW: < http://www1.nin.knaw.nl/~korjouko/projects/beholder2/index.htm>.

[27] *Neocognitronu for handwritten digit recogniton == Program in C language* [online]. [cit. 2010-05-21]. Dostupné z WWW: <http://visiome.neuroinf.jp/modules/xoonips/detail.php?item_id=375>.

[28] *Nearest Neighbour on KD-Tree in C++ and Boost* [online]. [cit. 2010-05-21]. Dostupné z WWW: <http://codingplayground.blogspot.com/2010/01/nearest-neighbour-on-kd-tree-in-c-and.html>.

[29] *K-Mean Clustering Code in Matlab* [online]. [cit. 2010-05-21]. Dostupné z WWW: <http://people.revoledu.com/kardi/tutorial/kMean/matlab_kMeans.htm>.

[30] *Jahmm - An implementation of Hoden Markov Models in Java* [online]. [cit. 2010-05-21]. Dostupné z WWW: <http://code.google.com/p/jahmm/downloads/detail?name=jahmm-0.6.1-src.zip&can=2&q=>.

[31] *HMMall* [online]. [cit. 2010-05-21]. Dostupné z WWW: <http://www.cs.ubc.ca/~murphyk/Software/HMM/hmm_download.html>.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

OCR Optical Character Recognition

UNS Umělá Neuronová Síť

HMM Hidden Markov Model

SEZNAM OBRÁZKŮ

SEZNAM TABULEK