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

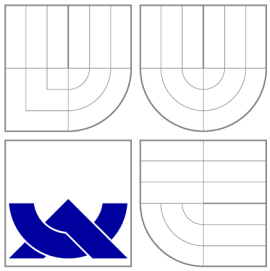
ROZPOZNÁNÍ RUČNĚ PSANÝCH ČÍSLIC

DIPLOMOVÁ PRÁCA
MASTER'S THESIS

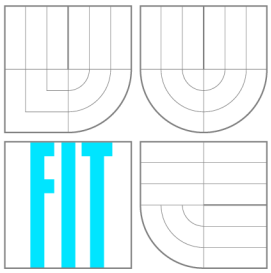
AUTOR PRÁCE
AUTHOR

Bc. MIROSLAV ŠTRBA

BRNO 2010



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

ROZPOZNÁNÍ RUČNĚ PSANÝCH ČÍSLIC

THESIS TITLE

DIPLOMOVÁ PRÁCA
MASTER'S THESIS

AUTOR PRÁCE — Bc. MIROSLAV ŠTRBA
AUTHOR

VEDÚCI PRÁCE — Ing. ADAM HEROUT, Ph.D.
SUPERVISOR

BRNO 2010

Abstrakt

Rozpoznávání ručně psaných číslic je problém, který se dá použít jako modelová úloha pro vícetřídní rozpoznávání vzorů v obraze. Tato práce zkoumá různé druhy algoritmů (Samo-organizující se mapy, Stromové klasifikátory a AdaBoost) a metody pro zvyšování úspěšnosti klasifikace pomocí fúze (většinové rozhodování, průměrování logaritmických pravděpodobnostních hodnot, lineární logistická regrese). Metody fúze byly využité na kombinaci klasifikátorů s identickými parametry trénování, s rozdílnými trénovacími metodami a s podvzorkovaným vstupním vzorem.

Abstract

Recognition of handwritten digits is a problem, which could serve as model task for multiclass recognition of image patterns. This thesis studies different kinds of algoritms (Self-Organizing Maps, Randomized tree and AdaBoost) and methods for increasing accuracy using fusion (majority voting, averaging log likelihood ratio, linear logistic regression). Fusion methods were used for combine classifiers with indentical train parameters, with different training methods and with multiscale input.

Kľúčové slová

Stromový klasifikátor, Rozhodovací strom, Samo-organizující se mapy, SOM, AdaBoost, Fúze klasifikátorů, Multirozlíšení, MNIST

Keywords

Tree classifier, Randomized Tree, Self-organizing map, SOM, AdaBoost, Classfier fusion, Multiresolution, MNIST

Citácia

Miroslav Štrba: Rozpoznání ručně psaných číslic, diplomová práca, Brno, FIT VUT v Brně, 2010

Rozpoznání ručně psaných číslic

Prehlásenie

Prehlasujem, že som diplomovú prácu vypracoval samostatne pod vedením Ing. Adama Herouta, Ph.D.

.....
Miroslav Štrba
24. mája 2010

Poďakovanie

Poďakovanie patrí Ing. Adamovi Heroutovi, Ph.D. za cenné rady, konštruktívnu kritiku a pomoc pri tvorbe tejto práce. Za poskytnutie výsledkov AdaBoost klasifikátorov sa chcem poďakovať Michalovi Hradišovi. Veľké poďakovanie patrí rodičom, ktorí ma podporovali počas štúdia na škole.

Obsah

Kapitola 1

Úvod

Rozpoznávanie ručne písaných číslic je v súčasnosti jeden z problémov, ktorým sa zaoberá množstvo ľudí. Jedným z dôvodov prečo je to tak je potencionálne využitie v praxi. Dnes sa rozpoznávanie využíva pri triedení poštových zásielok podľa PSČ a čiastočne aj ako podúloha pri OCR. Ďalším možným dôvodom popularity tohto problému môže byť aj jeho použiteľnosť ako ukážkový problém pri testovaní úspešnosti rozpoznávacích algoritmov.

Úloha by sa dala zaradiť od vedného oboru nazývaného *počítačové videnie*, pričom sa využívajú najmä algoritmy z umelej inteligencie. Samotným problémom pri rozpoznávaní je nemožnosť odhadnúť/upraviť výsledok rozhodovania na základe kontextu a rôznorodosť písania číslice ľuďmi. Pravdepodobnosť, že dvaja ľudia napíšu všetkých desať číslic rovnako je veľmi malá. Pravdepodobnosť toho, že tá istá osoba napíše číslice rovnako je síce väčšia, ale aj to závisí od nálady písajúceho (či je v kľude alebo sa niekam ponáhľa), prípadne či osoba pri písaní sedí alebo stojí.

Na zvýšenie úspešnosti učiaceho algoritmu sa v súčasnosti s obľubou používa fúzia klasifikátorov. V článku [3] L.A. Alexandre úspešne odskúšal fúziu klasifikátorov na úlohe rozpoznávania pohlavia osoby, kde rôzne klasifikátory boli získané pomocou zmeny veľkosti vstupného obrázka.

Motivácia

Hlavným cieľom rozpoznávania objektov je automatické spracovávanie. Rozpoznávanie ručne písaných číslic je vhodné na spracovávanie dokumentov, ktoré majú presne určenú pozíciu, kde sa číslice môžu nachádzať. Takýmto dokumentom sú rôzne formuláre. Formuláre, ktoré by mohli byť automaticky spracované pomocou prístroja a výsledky uložené do počítača určite ušetria množstvo času.

Keďže existuje množstvo rôznych implementácii založených na rôznych spôsoboch učenia (support vector machine, neurónové siete, AdaBoost ...), mojim cieľom bude vytvoriť viacero rozpoznávacích systémov. Tie by mali spolu dokázať rozpoznávať číslice lepšie ako každý systém samostatne. Pre každú implementáciu musí platiť, aby jej vyhodnocovanie bolo založené na inej podstate. Aby sa eliminoval vplyv učiacej metódy, budú jednotlivé systémy natrénované zakaždým inou metódou. Následne sa budem snažiť úspešnosť systémov vylepšiť rôznymi fúzovacími algoritmami podobne ako v článkoch [26], [15] a využívať pri trénovaní jednotlivých systémov rôzne príznaky. Podobne ako L.A. Alexandre tiež vyskúšam fúziu rovnakých klasifikátorov s rôznymi veľkosťami vstupných obrázkov.

Štruktúra práce

Práca je rozdelená do viacerých hlavných kapitol. V kapitole 2 sú popísané rozpoznávacie systémy. V časti 2.1 je popísaná metóda rozpoznávania založená na stromových klasifikátoroch. V jednotlivých častiach tejto kapitoly je vysvetlený princíp fungovania metódy a popis tzv. *tagov*, ktoré sú podstatou rozpoznávacieho systému založeného na stromových klasifikátoroch. V časti 2.2 je rozpísaná metóda rozpoznávania, ktorá je naučená pomocou samo-organizujúcich sa máp (ďalej len SOM). Časť 2.3 objasňuje postup učenia klasifikátora pomocou metódy AdaBoost využívajúca príznaky Multi-Block Local Binary Pattern. V ďalšej časti sa venujem databázam, ktoré sa používajú na ohodnocovanie úspešnosti jednotlivých metód, špeciálne databáze MNIST. Na záver kapitoly sú popísané tri fúzovacie metódy použité na zvýšenie úspešnosti jednotlivých klasifikátorov. V kapitole 3 sa nachádzajú informácie o vytvorených implementáciach, ktoré boli použité na rozpoznávanie číslic. V kapitole 4 sú implementované systémy otestované a vyhodnotené. Následne sú popísané experimenty so zvyšovaním úspešnosti jednotlivých rozpoznávacích systémov pomocou fúzovacích metód. Kapitola 5 obsahuje informácie o konferencii, na ktorú bol odoslaný článok, ktorý vychádza s tejto práce. Posledná kapitola obsahuje zhrnutie práce, najmä percentuálne výsledky úspešnosti rozpoznávania samostatných systémov, ako aj výsledky fúzovacích metód.

Kapitola 2

Rozpoznávanie ručne písaných číslic

Kapitola obsahuje popis troch rozpoznávacích systémov založených na rôznych metódach. Keďže popisované systémy budú použité na odskúšanie úspešnosti fúzovacích metód boli vybrané také učiace algoritmy, ktoré budú používať rozličné typy príznakov. Kombinácia príznakov by mala popisovať číslice komplexnejšie. Vybrané učiace algoritmy boli:

Stromové klasifikátory – využívajú malé okolia dvoch pixelov a smeru medzi nimi ako rozhodovacie kritérium (časť 2.1),

Samo-organizujúce mapy – neurónová sieť, ktorá využíva ako vstup priamo hodnoty pixelov (časť 2.2) a

AdaBoost – metóda využívajúca multi-block local binary patterns (časť 2.3).

Po popise učiacich algoritmov nasleduje časť, ktorá sa venuje databázam na trénovanie a testovanie algoritmov. Najviac využívanejšou databázou v súčasnosti je MNIST databáza, a preto bude sekcia orientovaná práve na túto databázu.

V záverečnej sekcii sú rozpísané podmienky kladené na jednotlivé algoritmy, aby mohla byť vykonávaná fúzia klasifikátorov. Ďalej sú naznačené možné spôsoby zabezpečenia požadovaných podmienok. Následne sú popísané tri fúzovacie metódy:

- Väčšinové rozhodovanie – angl. majority voting (časť 2.5.1),
- Priemerovanie logaritmických pravdepodobnostných hodnôt – angl. averaging log likelihood (časť 2.5.2) a
- Lineárna logistická regresia – angl. linear logistic regression (časť 2.5.3).

Každá časť obsahuje jednu z metód, popisuje princíp fúzie a spôsob vyhodnocovania fúzie (zaradení prvku z testovacej sady do triedy). Prvé dve metódy (väčšinové rozhodovanie a priemerovanie logaritmických pravdepodobnostných hodnôt) sú nenáročné metódy zatiaľ, čo fúzia pomocou lineárnej logistickej regresie je metóda sofistikovanejšia, a preto je implementačne výrazne náročnejšia.

2.1 Stromový klasifikátor

V tejto časti sa budem venovať rozpoznávaniu ručne písaných číslic pomocou stromových klasifikátorov. Práca bude vychádzať z článku [12] a bude nadväzovať na prácu [7].

Základom algoritmu je nájdenie vzťahu medzi významnými oblasťami, ktoré by boli unikátne pre jednotlivé druhy číslic a zároveň, aby číslice patriace do rovnakej skupiny tieto vzťahy obsahovali. Významnou oblasťou je okolie pixelu, v ktorej dochádza k zmene farby. V prípade šedotónového obrázku je za významnú oblasť považovaná každá časť číslice, ktorá obsahuje aspoň jeden pixel inej farby ako všetky ostatné pixely. Významné oblasti sa preto vyskytujú na hranách číslice a poskytujú informáciu o tvare číslice. Každé okolie sa potom zaklasifikuje do určitej triedy. Zaklasifikovaná oblasť sa bude nazývať *tag*. Vzťah medzi tagmi sa bude nazývať pravidlo. Pomocou sady pravidiel je potom možné určiť aký typ číslice sa na obrázku nachádza.

2.1.1 Tagovací strom

Aby mohol byť zostavený tagovací strom, je nutné najskôr vyhľadať významné oblasti. Keďže databáza (popísaná v kapitole 2.4) bola upravená prahovaním, za významnú oblasť budeme považovať časť obrázku, ktorá bude obsahovať súčasne aspoň jeden čierny a aspoň jeden biely pixel.

Počet oblastí okrem hrúbky čiary číslice ovplyvňuje ich veľkosť. Čím väčšia veľkosť, tým je aj počet nájdených oblastí viacej (samozrejme pokiaľ berieme do úvahy, že existuje dostatočné okolie číslice). V článku [12] bol za optimálnu veľkosť považovaný rozmer 4×4. Keďže autori pracovali s rovnakou dátovou sadou, ďalej sa budeme zaoberať tagmi len s touto veľkosťou aj keď pre iné rozlíšenie obrázkov môže byť vhodná iná veľkosť tagov.

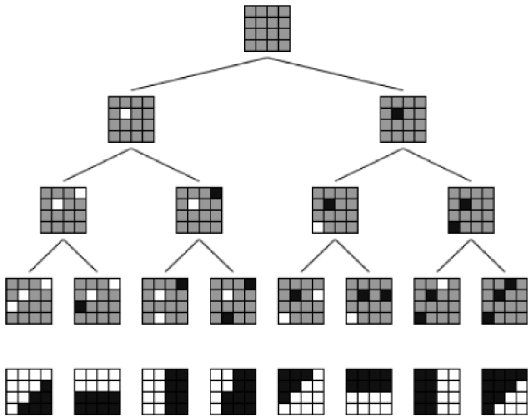
Ďalším faktorom, ktorý ovplyvňuje počet oblastí je typ číslice. Pre jednotlivé typy je príznačné, že obsahujú menej významných oblasti. Jedným zo spomínaných typov je číslica 1, ktorá v priemere môže obsahovať až o polovicu menej významných oblastí ako typ 0. Nasledujúca tabuľka obsahuje informácie o priemernom počte oblastí v jednotlivých typoch ručne písaných číslic:

číslica	0	1	2	3	4	5	6	7	8	9
počet oblastí	366	186	330	329	289	314	298	261	320	273

Tabuľka 2.1: Priemerný počet nájdených oblastí

Po získaní všetkých oblastí je možné začať s klasifikáciou. Klasifikácia sa uskutočňuje pomocou stromu, ktorý sa bude nazývať *tagovací strom*. Ukážka tagovacieho stromu je na obrázku 2.1. Tagovací strom má za úlohu zatriediť oblasti do skupín.

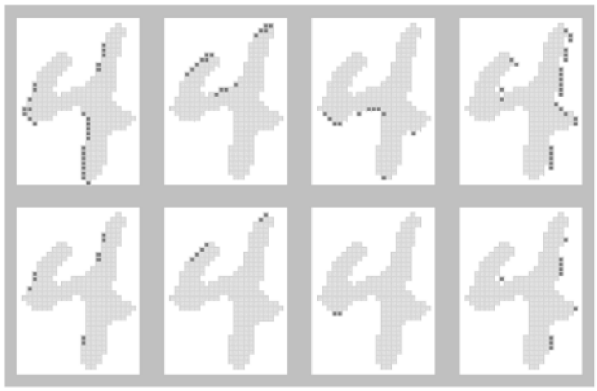
Tagovací strom v každom uzle delí významné oblasti na dve časti v závislosti na danom pixeli. V prvej úrovni má na výber pre veľkosť 4×4 16 pixelov, podľa ktorých sa môžu oblasti deliť. V jednom podstrome sa potom budú nachádzať oblasti, ktoré majú biely pixel na danej pozícii (*"biely podstrom"*) a v druhom podstrome (*"čierny podstrom"*) sa budú nachádzať oblasti s čiernym pixelom na danej pozícii. V ďalšej fáze delenia sa významných oblastí sa spracováva len tá časť, ktorá vyhovela všetkým predkom. To znamená, že "biely podstrom" uzla, ktorý rozdelil významné oblasti podľa pixelu na pozícii $[x, y]$, bude rozhodovať o deliacom pixely len z množiny nájdených významných oblastí, ktoré na danej pozícii obsahujú biely pixel.



Obrázok 2.1: Ukážka tagovacieho stromu s hĺbkou 3. Pod listami stromu sú znázornené významné oblasti patriace do danej skupiny tagov. Prevzaté z [12].

Rozdelenie významných oblastí by malo byť najrovnomernejšie, aby bolo možné jednotlivé významné oblasti účinné zaklasifikovať. Preto je výhodné preskúmať všetky varianty rozdelenia v danom uzle. Potom bude uzol obsahovať taký deliaci pixel, ktorý rozdeľuje významné oblasti na približne rovnaké časti.

Typ oblasti je definovaný uzlami, ktoré sa nachádzajú na ceste k listu stromu. Každý uzol reprezentuje tag. Každá skupina, ktorú zastupujú jednotlivé uzly má určité spoločné rysy. Počet spoločných rysov je závislý od hĺbky uzlu v strome. Za spoločný rys sa považujú pixely rovnakej hodnoty na rovnakej pozícii. Na obrázku 2.1 je vidieť ako jednotlivé pixely začleňujú oblasti do podobných skupín. Keďže tagy reprezentujú oblasť len určitým počtom pixelov zachovávajú aj určitú mieru voľnosti.



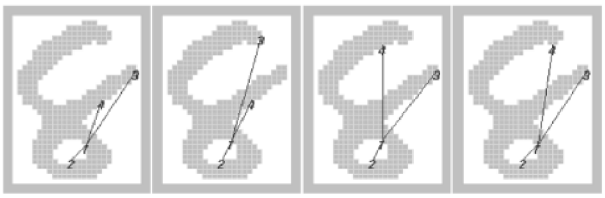
Obrázok 2.2: Ukážka rozloženia tagov stromu s hĺbkou 3. Prevzaté z [12].

Oblasti, ktoré boli označené rovnakým tagom je vidieť na obrázku 2.2. Tie vytvárajú v obrázku zhluky. Pomocou zhlukov a ich pozícií je možné číslice reprezentovať. Z uvedeného dôvodu budú pravidlá založené na rozložení zhlukov v obrázku.

2.1.2 Rozhodovací strom

Rozhodovací strom je určený na samotné rozpoznávanie číslice. Jeho podstatou je využívanie pravidiel na triedenie číslic do podstromov až listov. Ako bolo spomenuté pravidlá reprezentujú vzťah medzi tagmi. Vzťah je v tomto prípade pozícia jedného tagu voči druhému. Pravidlo teda pozostáva z troch častí: z dvoch tagov a smeru, ktorý je medzi týmito tagmi.

Tagy sú zaklasifikované oblasti pomocou tagovacích stromov. Smer medzi tagmi je určený polohou medzi tagmi. Kvôli určitému stupňu voľnosti bol atribút smeru rozdelený do ôsmych skupín. Pre lepšiu predstavu je možné ich pomenovať podľa kompasu: "sever", "severovýchod", ... Dva tagy sú vo vzťahu, ak druhý tag sa nachádza v príslušnom $\pi/4$ kvadrante od prvého tagu. Keďže v pravidle sa vyskytuje iba informácia o smere medzi tagmi a nie o vzdialenosti, existuje viacero variant ako môže byť pravidlo splnené. Tieto možnosti sú ukázané na obrázku 2.3.



Obrázok 2.3: Ukážka splnenia pravidla rôznymi spôsobmi pre rovnakú číslicu. Prevzaté z článku [12].

Z obrázku 2.3 je vidno, ako sa pomocou pravidiel dá popísať tvar číslice. V prípade pravidla, ktorý definuje daný typ číslice je možné, že pravidlo bude môcť byť splnené viacerými spôsobmi.

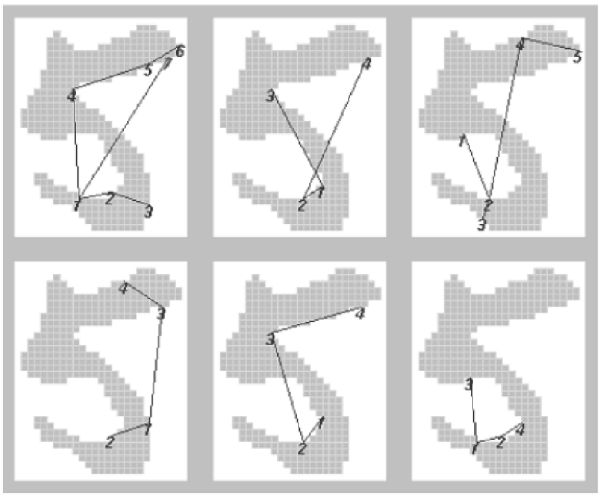
Rozhodovací strom využíva pravidlá na delenie číslic do listov. V listoch sa nachádza informácia o tom, koľko akých typov číslic patrí danému listu. Táto informácia je podstatná pre rozhodnutie s akou pravdepodobnosťou číslica, ktorá dospela do daného uzlu, je určitého typu. Optimálny rozhodovací strom rozdelí číslice spôsobom, že v listoch sa budú nachádzať iba číslice jedného typu.

Delenie číslic prebieha na základe informácie, či sa v obrázku vyskytuje aspoň jedna dvojica tagov spĺňajúca pravidlo. Ak sa dvojica v obrázku nachádza je zaradená do jedného podstromu (*"áno podstrom"*). Inak je zaradená do druhého podstromu (*"nie podstrom"*). V prípade nižšie položených uzlov v "áno podstrome" sa vychádza z už nájdených tagov. Na ďalšie vytvorenie "áno podstromu" musí teda platiť, že existuje tag, ktorý je prvou časťou pravidla, ktoré boli označené ako splnené. Pravidlá tak vytvárajú rozvetvený strom. Splnenie troch pravidiel je možné vidieť na obrázku 2.4.

Metrika

Výber najvhodnejšieho pravidla je najdôležitejšou časťou zostavenia stromu pre rozhodovanie o ručne písaných číslicach. Ohodnocovanie pravidiel vychádza z práce [7]. Najlepšie pravidlo minimalizuje hodnotu M, ktorá sa vypočíta z rovnice 2.1.

$$M = K_1 * N * \left(x_\phi - \frac{P}{2N} \right)^2 - K_2 * \left(\sum_{i=0}^{N-1} (x_i - x_\phi)^2 \right) \tag{2.1}$$



Obrázok 2.4: Ukážka pravidiel, ktoré boli vytvorené rôznymi rozhodovacími stromami. Prevzaté z článku [12].

Časť za konštantou K_1 ohodnocuje vyváženosť rozhodovacieho stromu a časť za konštantou K_2 oddeľuje jednotlivé typy číslic od seba. Obe konštanty udávajú pomer v akom sa majú brať do úvahy jednotlivé zložky. V práci [7], kde bola metrika zavedená a otestovaná sa ako optimálny pomer ukázal byť pomer $7:1$. Ďalším konštantným parametrom je počet tried N. Tento parameter má v prípade ručne písaných číslic hodnotu 10. Nasledujúci parameter P je už závislý na konkrétnom umiestnení uzlu. Parameter udáva počet vzorov v danom uzle. Je logické, že sa bude hodnota znižovať s hĺbkou stromu. Koreňový uzol rozhodovacieho stromu bude mať P rovné počtu vzoriek v trénovacej sade. V optimálnom prípade by synovské uzly mali hodnotu P rovnú jednej polovici zo všetkých trénovacích vzoriek. Hodnoty x_ϕ a x_i sú už závislé na konkrétnej variante pravidla, ktoré má byť metrikou ohodnotené. Hodnota x_i udáva počet vzoriek, ktoré patria do určitej skupiny a spĺňajú vyhodnocované pravidlo. Hodnota x_ϕ udáva priemerný počet vzorov spĺňajúcich pravidlo t.j. $x_\phi = \frac{1}{N}\sum_{i=0}^{N} x_i$.

2.2 Rozpoznávanie pomocou SOM

Táto kapitola popisuje metódu založenú na neurónových sieťach, keďže samotné rozpoznávanie by malo byť rýchle a čo možno najpresnejšie. Neurónové siete patria medzi metódy, ktoré dosahujú výborné výsledky a ich rýchlosť vyhodnocovania je tiež dostatočná. Ako spôsob usporiadania neurónov boli zvolené samo-organizujúce sa mapy (SOM). Autorom modelu usporiadania neurónov do SOM je T. Kohonen (SOM definoval v práci [16]). Preto sa SOM niekedy nazývajú *Kohonenovymi mapami*. Základom rozhodnutia využiť práve samo-organizujúcu mapu je fakt, že SOM nepotrebujú žiadnu skrytú vrstvu a tak jediný parameter, ktorý ovplyvňuje schopnosť mapy sa naučiť resp. nenaučiť rozpoznávať číslice je jej veľkosť. Ďalším argumentom pre výber SOM ako zástupcu neurónového učenia je spôsob upravovania váh neurónom. Metóda rozpoznávania ručne písaných číslic pomocou SOM bola otestovaná v článku [4], kde sa snažili implementovať SOM do hardvéru.

SOM je druh neurónovej siete, ktorej učenie prebieha podobne ako pri jednoduchej

súťaživej sieti. Pri oboch typoch sa vyberie neurón, ktorý pomocou svojich váh dáva najlepšiu odovzu na vstup. Takýto neurón sa nazýva víťazný a jeho váhy sú modifikované aby dávali ešte lepšiu odozvu na ten istý vstup. Rozdiel medzi jednoduchou súťaživou sieťou a SOM je v tom, že SOM nemodifikuje váhy len víťaznému neurónu ale aj neurónom, ktoré sa nachádzajú v jeho okolí. Upravovanie váh svojmu okoliu spôsobuje rýchlejšie učenie siete. Táto vlastnosť je užitočná pri trénovaní.

Algoritmus 2.2.1 trénovanie 2D Kohonenovej mapy

Vstup: trénovacia množina $T = \langle i_1, \ldots, i_P \rangle$ s veľkosťou $i \times j$
číslo K udávajúce počet iterácií trénovania
učiaca konštanta η_k a topologická vzdialenosť $D(k)$ pre každú iteráciu K

Inicializácia: $w_{ij} = \langle 0 \ldots 1 \rangle$ pre každý n_{xy}
pre: $k = 1, 2, \ldots, K$

1. Urči učiacu konštantu η_k a topologickú vzdialenosť $D(k)$
2. Výber trénovací vstup i_p z trénovacej množiny T
3. Nájdi neurón n_b, ktorého váhy w sú najbližšie k i_p
4. Prepočítaj váhy w všetkých neurónov n_{xy} v topologickej vzdialenosti $D(k)$ od n_b. Použi pravidlo :

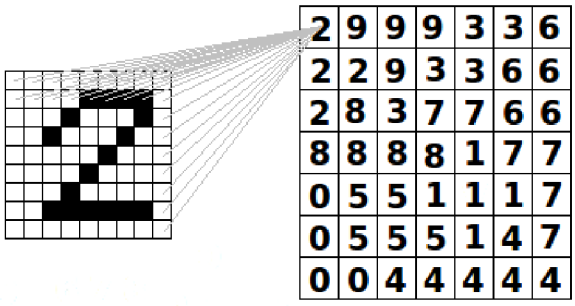
$$w_{ij}(k+1) = w_{ij}(k) + \eta(k)\,(i_p(k) - w_{ij}(k)) \tag{2.2}$$

Výstup: natrénovaná SOM s $x \times y$ neurónmi n a $i \times j$ váhami w

---

Hlavným znakom SOM je teda usporiadanie neurónov v pravidelnej štruktúre (reťaz, mriežka, kocka ...). Dôležité je, aby bolo jasne definovateľné okolie neurónu. Preto je možné, aby mriežka mohla mať štvorcový tvar, kde neurón má štyri respektíve osem susedných neurónov, alebo môže mať šesťuholníkový tvar, kde počet susedných neurónov je šesť. Okolie neurónu využíva funkcia $D(k)$, ktorá v každom kroku k určí veľkosť okolia. Funkcia $D(k)$ by sa mala so zvyšujúcim sa krokom klesať až k nule. Ak sa hodnota funkcie rovná nule znamená to, že víťazný neurón Kohonenovej mapy neovplyvňuje svoje okolie. Upravovanie váh okolitých neurónov spôsobí zhlukovanie číslic rovnakého typu. Malý počet neurónov v SOM sa chová podobne ako K-means, ale so zvyšujúcim sa počtom neurónov SOM dokáže reorganizovať dáta tak, aby zodpovedali svojej triede.

Na obrázku 2.5 je zobrazená ukážka jednej Kohonenovej mapy. Na obrázku je naznačené prepojenie jedného neurónu so všetkými pixelmi na vzore pomocou šedých čiar. Čiary nie sú pre prehľadnosť zakreslené všetky, ale iba vrchné dva riadky a pravý stĺpec. Tieto prepojenia prepájajú každý neurón so všetkými pixelmi. To znamená, že v prípade mapy o veľkosti 7×7 a o veľkosti vstupného vzoru 9×9 je počet prepojení 3 969. Každé prepojenie má určenú váhu, ktorou neurón reaguje na pixel na danej pozícii. Aby neurón mohol rozhodovať o klasifikácii vzoru do určitej triedy, musí okrem váh prepojení obsahovať informáciu o počte typov vzoriek, pre ktoré sa stal víťazný. Na základe informácie o rozložení typov je možné podľa rôznych kritérií rozhodovať o vzorke na vstupe siete v procese testovania a tak vyhodnocovať úspešnosť rozpoznávania.



Obrázok 2.5: Ukážka natrénovanej samo-organizovacej mapy. Šedé čiary naznačujú prepojenie všetkých pixelov s každým neurónom.

2.3 AdaBoost

AdaBoost [27, 9] je skrátený anglický názov pre *adaptive boosting*. Tento algoritmus dokáže výrazne znížiť chybu ľubovoľného klasifikátora (slabý klasifikátor, angl. *weak classifier*), ktorý dokáže dávať výsledky lepšie ako náhodné tipovanie. AdaBoost je adaptabilný z dôvodu využívania viacerých slabých klasifikátorov. Pri učení sa využíva množina predtým zle zaradených objektov na natrénovanie nasledujúcich klasifikátorov. Výsledným výtvorom je jeden robustný klasifikátor (silný klasifikátor, angl. *strong classifier*). Metóda AdaBoost bola použitá na rozpoznávanie ručne písaných číslic M. P. Carterom v článku [6].

Podstatou prispôsobovania sa zle klasifikovaným objektom je algoritmus, v ktorom sa upravujú váhy jednotlivých klasifikátorov. V prípade, že sa objekt zle zaklasifikuje, je váha klasifikátora, ktorý tento objekt zdetekoval upravená, čím sa dosiahne toho, že v nasledujúcom trénovaní algoritmus dokáže svoje rozhodnutia opraviť.

Samotný algoritmus je citlivý na všetky objekty. To znamená, že sa snaží dosiahnuť, aby každý objekt bol zaradený do správnej množiny. To aj v prípade, že je nesprávne klasifikovaných prvkov výrazne menej. Táto vlastnosť nie je dobrá pre generalizáciu, ale vzhľadom na to, že je AdaBoost odolný voči pretrénovaniu (pri vysokom počte prvkov v trénovacej sade nemôže výsledný klasifikátor dávať horšie výsledky, ako keby bolo v trénovacej sade menej prvkov) je táto vlastnosť výrazne potlačená.

Postup trénovania

Popis fungovania AdaBoostu je vysvetlený v dokumente [10], kde je popísaná základná varianta algoritmu 2.3.1. Dokument [11] obsahuje dve rozšírenia pre varianty, keď je potrebné zatriediť objekty do viacerých skupín.

Algoritmus 2.3.1 popisuje fungovanie AdaBoostu

Vstup: sekvencia vstupov $\langle (x_1, y_1), \ldots, (x_m, y_m) \rangle$
trénovací algoritmus na slabé klasifikátory v príklade nazývaný ako **LearnAlg**
číslo T udávajúce počet iterácií trénovania

Inicializácia: $D_1(i) = 1/m$ pre každé i
pre: $t = 1, 2, \ldots, T$

1. Urči pomocou **LearnAlg** a s využitím D_t hodnotu vrátenú každým klasifikátorom.

2. Vyber slabý klasifikátor $h_t = X \to \{-1, +1\}$, ktorý minimalizuje chybu s ohľadom na $D_t(i)$ pomocou rovnice 2.3:

$$h_t \colon \epsilon_t = \sum_{i:h_t(x_i) \neq y_i} D_t(i) \tag{2.3}$$

$$\text{Ak } \epsilon_t = \Pr_{i \sim D_t}[h_t(x_i) \neq y_i] \tag{2.4}$$

3. Vyber $\alpha_t = \frac{1}{2} \ln \left(\frac{(1-\epsilon_t)}{\epsilon_t} \right)$.

4. Obnov

$$D_t \colon D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{ak } h_t(x_i) = y_i \\ e^{\alpha_t} & \text{ak } h_t(x_i) \neq y_i \end{cases} \tag{2.5}$$

$$= \frac{D_t(i) \, \exp(-\alpha_t y_i h_t(x_i))}{Z_t}, \tag{2.6}$$

   kde Z_t je normalizačná konštanta zvolená tak, aby funkcia D_{t+1} zostala pravdepodobnostným rozložením.

Výstup: hypotéza $h_{fin}(x) = \text{sign} \left(\sum_{t-1}^{T} \alpha_t h_t(x) \right)$

Výsledkom je funkcia, ktorá je váhové ohodnotenie slabých klasifikátorov.

WaldBoost

V predchádzajúcej časti bol popísaný algoritmus AdaBoost, ktorého nevýhodou je, že nie je možné rozhodnúť, či daná prehľadávaná oblasť patrí alebo nepatrí do určitej skupiny skôr, ako sa prepočítajú hodnoty zo všetkých slabých klasifikátorov, aj keď je výsledok zrejmý už počas vyhodnotenia časti klasifikátorov. Na riešenie spomenutého nedostatku vznikla modifikácia AdaBoostu nazvatá *WaldBoost*, ktorá je popísaná v [30]. Klasifikátor natrénovaný pomocou WaldBoostu nemusí byť vyhodnotený celý. V prípade, že počas vyhodnocovania slabým klasifikátorom sa prekročí určená spodná alebo vrchná hranica, oblasť je zaradená do danej skupiny. V opačnom prípade sa skúmaná oblasť predá nasledujúcemu klasifikátoru.

Algoritmus 2.3.2 popisuje fungovanie WaldBoostu

Vstup: sekvencia vstupov $\langle (x_1, y_1), \ldots, (x_m, y_m) \rangle$; $x_m \in X, y_m \in \{-1, +1\}$

Inicializácia: váha $w_1(x_i, y_i) = 1/m$ pre každé i
horná hranica $A = \frac{1-\beta}{\alpha}$ a dolná hranica $B = \frac{\beta}{1-\alpha}$

pre: $t = 1, 2, \ldots, T$

1. Vyber h_t pomocou rovnice $h^{T+1} = \frac{1}{2} \log \frac{P(y=+1|x, w^{(T)}(x,y))}{P(y=+1|x, w^{(T)}(x,y))}$.

2. Odhadni pravdepodobnosť $R_t(x) = \frac{p(h^1, h^2, \ldots, h^t(x)|y=-1)}{p(h^1, h^2, \ldots, h^t(x)|y=+1)}$.

3. Nájdi hraničné hodnoty θ_A^t a θ_B^t.

4. Odstráň vzorky z trénovacej sady pre ktoré platí $H_t \geq \theta_B^t$ alebo $H_t \leq \theta_A^t$.

5. Zaraď vzorky do trénovacej sady.

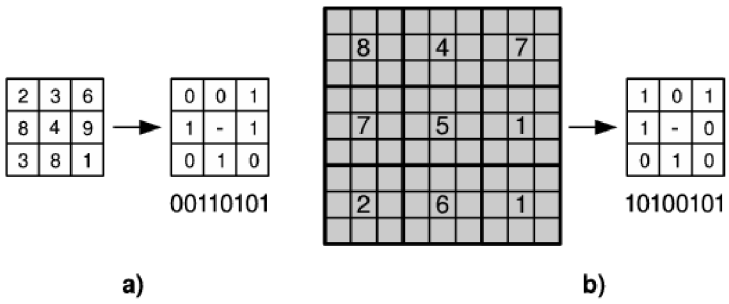
Výstup: silný klasifikátor H_t a hraničné hodnoty θ_A^t a θ_B^t

---

Intuitívne sa dá predpokladať, že rýchlosť WaldBoostu je ovplyvnená počtom slabých klasifikátorov. Z prieskumu v [30] je pri počte 300 klasifikátorov priemerná rýchlosť $\bar{T}_s = 9.57$ a pri počte 600 je hodnota $\bar{T}_s = 13.92$. To znamená, že na vyhodnotenie pridaných 300 klasifikátorov stačí polovica pôvodného času.

2.3.1 Multi-Block Local Binary Pattern

Multi-Block Local Binary Pattern sú príznaky, ktoré vychádzajú z pôvodných Local Binary Patterns (LBP) [25]. LBP boli preukázané ako efektívny spôsob popisovania tvárí pre ich následné rozpoznávanie v práci [1]. Multi-Block Local Binary Pattern (MB-LBP) sú takisto úspešne využívané na rozpoznávanie ľudských tvári [23]. V článku boli MB-LBP definované a prvýkrát použité namiesto pôvodne využívaných LBP. MB-LBP majú oproti LBP niekoľko výhod. Medzi najdôležitejšie patrí ich robustnosť a schopnosť popísať makro aj mikro štruktúru nachádzajúcu sa na obrázku.



Obrázok 2.6: a) Prahovanie LBP a konverzia do binárnej podoby. b) Prahovanie MB-LBP a konverzia do binárnej podoby.

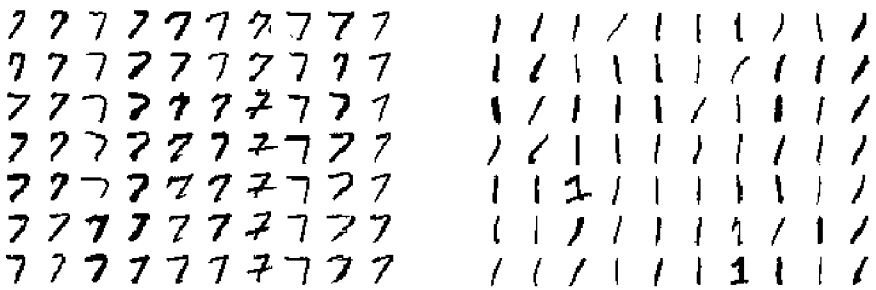
LBP popisuje oblasť zvyčajne o veľkosti 3×3 pixelov, kde hodnota sa počíta pre stredný pixel. Po vyprahovaní okolitých pixelov hodnotou stredného pixela, sú hodnoty pixelov konvertované do binárnej podoby. Proces prahovania a binarizácie je zobrazený na obrázku 2.6a. Binarizácia vytvára hodnotu od ľavého horného rohu v smere hodinových ručičiek. U MB-LBP je hodnota jedného pixela nahradená priemernou hodnotou úrovne šedej v regióne. Každý región je štvorcová oblasť obsahujúca susedné pixely. Oblasti sú potom prahované obdobne ako u LBP a ich hodnoty sú rovnakým spôsobom zakódované do binárnej podoby 2.6b. V prípade veľkosti regiónu 3×3 pixelov je veľkosť MB-LBP 9×9 pixelov.

2.4 Dátová sada

Keďže problematika rozpoznávania číslic je rozšírená [17, 18, 20] existuje viacero dátových sád, ktoré sa používajú na trénovanie a testovanie vytvorených systémov ([22, 28, 29], ...). Najznámejšou a najpoužívanejšou dátovou sadou ručne písaných číslic je dátová sada MNIST.

MNIST dátová sada vznikla premiešaním dvoch dátových sád (Special Database 1 a Special Database 3) zostavených americkým úradom pre štandardy a technológie (**N**ational **I**nstitute of **S**tandards and **T**echnology) ďalej len NIST. Pôvodne bola jedna databáza (SD-3) určená na trénovanie a druhá na testovanie (SD-1) rozpoznávacích systémov, ale keďže obe databázy boli zozbierané v rozdielnych podmienkach vznikol problém pri testovaní úspešnosti. Ako riešenie sa ukázalo vhodné vytvoriť databázu, ktorá bude obsahovať zmiešané vzorky z oboch databáz.

Dátová sada pochádzajúca z USA má nevýhody pri reálnom nasadení rozpoznávacích systémov v európskych krajinách. Dajú sa poznať určité rozdiely napísaných číslic v USA voči čísliciam písaných v Európe. Ako je vidieť na obrázku 2.7 pochádzajúceho z [7] to čo by sa dalo v Európe považovať za číslicu 1 sa v USA pokladá za číslicu 7. Rozdiel je spôsobený tým, že Američania píšu číslicu 1 ako zvislú čiaru (bez zalomenej vrchnej časti) alebo so zalomenou vrchnou časťou, ale celú číslicu ešte podtrhnutú.



Obrázok 2.7: Rozdiel medzi číslicami 1 a 7

Vzory číslic NIST databázy boli najskôr prepočítané na rozmery maximálne 20×20 pixelov, pričom bol zachovaný pomer strán. SD-1 a SD-3 obsahovali vzory číslic len binárne a tak pri prepočítavaní rozmerov vznikli vzory číslic šedotónové. Následne boli vzory umiestené do štvorca o veľkosti 28×28 v závislosti na ťažisku pixelov.

Dátová sada MNIST obsahuje 60 000 trénovacích vzoriek a 10 000 testovacích vzoriek. Sada je popísaná v [22], kde je možnosť aj databázu stiahnuť. Jednotlivé časti sa skladajú z dvoch súborov. Jeden obsahuje obrázky číslic uložených vo viacrozmernej matici. Prvým rozmerom je poradie číslice. Druhým a tretím rozmerom je šírka (28 pixelov) a výška (28 pixelov) obrázka. V súbore sa nachádza 28×28 pixelov v odtieňoch šedej pre každú vzorku. Druhý súbor obsahuje označenia pre obrázok. Tento súbor obsahuje jednorozmernú maticu hodnôt od 0 do 9. Hodnota na pozícii udáva informáciu o tom, aká číslica je zobrazená na obrázku, ktorý sa nachádza na tej istej pozícii v druhom súbore.

Dátová sada MNIST bude použitá na testovanie jednotlivých systémov. V prípade stromových klasifikátorov bude databáza opäť prevedená do binárnej podoby. Využije sa pri tom metóda prahovania, kde prah bude nastavený na hodnotu 127. Obdobne sa vykoná binarizácia aj pri rozpoznávaní pomocou samo-organizujúcich máp.

2.5 Fúzia klasifikátorov

Fúzia klasifikátorov, v prípade rozpoznávania ručne písaných číslic popísaná v článku [8], môže dosahovať oveľa väčšiu úspešnosť ako jednotlivé klasifikátory. Preto existuje množstvo rozličných fúzovacích algoritmov [26, 15]. V tejto časti budú popísané dva jednoduchšie postupy fúzie klasifikátorov (väčšinové rozhodovanie 2.5.1 a priemerovanie logaritmických pravdepodobnostných hodnôt 2.5.2) a jedna sofistikovanejšia metóda (lineárna logistická regresia 2.5.3).

Fúzia pomocou väčšinového rozhodovania a fúzia pomocou priemerovania logaritmických pravdepodobnostných hodnôt sú jednoduchšie metódy kombinácie klasifikátorov hlavne preto, lebo na vyhodnocovanie testovacej sady nie sú potrebné žiadne ďalšie informácie okrem odoziev klasifikátorov na jednotlivé prvky sady. Lineárna logistická regresia je náročnejšia metóda fúzie, keďže vyhodnocovanie na testovacej sade musí predchádzať nájdenie optimálnych parametrov fúzie. Na určenie parametrov je vhodné využiť algoritmus umelej inteligencie.

Použitie fúzie ako nástroja na zlepšenie úspešnosti vyhodnocovania je možné, ak učiaci algoritmus nie je deterministický (môže produkovať rozdielne klasifikátory), alebo ak je použitých viacero klasifikačných metód. Učiaci algoritmus produkuje rozdielne klasifikátory, ak vo svojom algoritme využíva náhodný generátor čísel, prípadne v trénovacej časti sú vstupom rozdielne dáta, alebo má rozdielne nastavené parametre.

Generátor náhodných čísel sa často využíva na inicializáciu iteratívneho trénovacieho procesu, alebo môže byť využitý na výber len určitých prvkov ako v prípade klasifikátora založeného na stromových štruktúrach, kde sa využíva na výber najlepšieho pravidla, keďže vyhodnocovanie metriky je časovo náročné a nie je možné, aby sa pre všetky vzory vyhodnocovali všetky možné pravidlá. Rozdielne vstupy pre klasifikátory v trénovacom procese môžu byť dosiahnuté rozdelením trénovacej sady na menšie časti, alebo prevzorkovaním vstupných obrázkov do viacerých rozlíšení. Je tiež možné použiť identický učiaci algoritmus, ale učenie vykonávať s rozdielnymi príznakmi, ako v prípade [3]. Vykonávanie fúzie klasifikátorov s rozdielnymi príznakmi spôsobí, že rozpoznávací systém môže ohodnotiť jednotlivé vzorky komplexnejšie a na základe získaných hodnôt vykonať presnejšie vyhodnotenie.

2.5.1 Väčšinové rozhodovanie

Väčšinové rozhodovanie je jedna z jednoduchších metód fúzie klasifikátorov popísaná v článkoch [19, 13]. V prípade rozhodnutia väčšinou, každý klasifikátor urobí rozhodnutie $a_{k,i}(x)$ o danom vzore x pre každú triedu i (v prípade rozpoznávania ručne písaných číslic je počet tried desať).

Hodnota $\delta_{k,i}(x)$ v rovnici 2.7 na základe rozhodnutia od klasifikátora priradí pravdepodobnosť 1 tej triede, ktorá bola označená ako najpravdepodobnejší kandidát pre danú vzorku. Ostatným triedam sa priradí hodnota 0. Hodnota $\delta_{k,i}(x)$ reprezentuje volebný hlas. Každý klasifikátor svojim hlasom volí danú (z pohľadu klasifikátora najpravdepodobnejšiu) triedu, pričom všetky klasifikátory majú rovnakú rozhodovaciu váhu.

$$\delta_{k,i}(x) = \begin{cases} 1 & \text{ak } a_{k,i}(x) = \max\limits_{i=0}^{9} a_{k,i}(x) \\ 0 & \text{inak} \end{cases} \tag{2.7}$$

Rovnica (2.8) popisuje funkciu pre rozhodovanie na základe väčšiny, ktorá využíva volebné hlasy reprezentujúce triedy a vyhodnocuje skupinovú odozvu všetkých N klasifikátorov.

M_i na základe toho, že každý klasifikátor obsahuje hodnotu 1 práve v jednom prípade a v ostatných prípadoch obsahuje 0, reprezentuje okrem počtu hlasov aj počet klasifikátorov, pre ktoré je vzorka x najviac pravdepodobná triede i.

$$M_i(x) = \sum_{k=0}^{N} \delta_{k,i}(x) \tag{2.8}$$

Vzor na obrázku je rozpoznaný ako typ číslice, ktorý dosiahol najviac hlasov. Z rovnice 2.8 sa hľadá argument i, pre ktoré je $M_i(x)$ maximálne. Ak je maximálna hodnota dosiahnutá pre dve alebo viacej tried, rozhodnutie je urobené náhodným výberom jednej z týchto tried.

2.5.2 Priemerovanie logaritmických pravdepodobnostných hodnôt

Priemerovanie logaritmických pravdepodobnostných hodnôt (ďalej uvádzané ako „log likelihood“ pôvodom z angličtiny) je ďalšia z jednoduchých a rýchlych metód. Metóda fúzie klasifikátorov, na rozdiel od rozhodovania väčšiny, berie do úvahy všetky hodnoty pre jednotlivé triedy a nielen najlepšie ohodnotenú triedu z každého klasifikátora. Odozvy z klasifikátora $a_{k,i}(x)$ reprezentujú hodnotu log likelihoodu a nie pravdepodobnosť. Keďže, ale niektoré klasifikačné postupy môžu udávať ako výstup práve pravdepodobnosť, je dobré poznať vzťah medzi hodnotou log likelihood a pravdepodobnosťou. Konverzia jednej hodnoty na druhú je založená na rovnici 2.9:

$$p_{k,i}(x) = \frac{1}{1 - e^{a_{k,i}(x)}}, \tag{2.9}$$

kde $p_{k,i}(x)$ je pravdepodobnosť, že vzor číslice x patrí do triedy i podľa klasifikátora k. Fúzia klasifikátorov používajúca metódu priemerovania logaritmických pravdepodobnostných hodnôt môže byť vyjadrená ako:

$$M_i(x) = \sum_{k=0}^{N} a_{k,i}(x). \tag{2.10}$$

Finálne rozhodnutie je urobené na základe nájdenia maximálnej hodnoty. Trieda i, ktorá ma odozvu fúzie najväčšiu, je považovaná za rozpoznaný typ vzoru číslice x. Keďže sa pre rozhodovanie o zatriedení vzoru hľadá maximum z hodnôt $M_i(x)$, nie je dôležitá presná hodnota, a preto sa pravá strana rovnice 2.10 nemusí deliť počtom klasifikátorov.

2.5.3 Lineárna logistická regresia

Na rozdiel od predchádzajúcich dvoch metód hlavná myšlienka lineárnej logistickej regresie je nastavenie váh rozhodnutiam pre jednotlivé klasifikátory. Váhy sú nastavované v závislosti na úspešnosti vyhodnocovania trénovacích vzoriek klasifikátormi. Jeden klasifikátor obsahuje toľko váhovacích parametrov, koľko je tried na rozpoznávanie (v prípade ručne písaných číslic klasifikátor obsahuje 10 váh).

Lineárna logistická regresia využíva obdobný postup ako priemerovanie logistických hodnôt popísané v predchádzajúcej časti 2.5.2. Fúzia pozostáva z nastavenia váhovacích parametrov pre všetky klasifikátory. Váhovacie parametre sa určujú na základe výstupov klasifikátora, ktorým sú opäť logaritmické pravdepodobnostné hodnoty. Výstup z jedného

klasifikátora k pre rozpoznávanie ručne písanej číslice x môžeme matematicky zapísať ako vektor:

$$\vec{l_k}(x) = (a_{k,0}(x), a_{k,1}(x), \ldots, a_{k,9}(x)). \tag{2.11}$$

Funkcie $a_{k,0}$ až $a_{k,9}$ ohodnocujú vzor číslice. V tomto prípade sú to logaritmické pravdepodobnostné hodnoty. Pre vyhodnocovanie sú hodnoty všetkých vektorov zfúzované do jedného logaritmického pravdepodobnostného ohodnotenia vyjadreného ako $\vec{l'}$ použitého na konečné rozhodnutie [5]:

$$\vec{l'}(x) = \sum_{k=1}^{N} \alpha_k \vec{l_k}(x) + \beta. \tag{2.12}$$

V rovnici 2.12 je fúzia klasifikátorov vykonávaná sumáciou ohodnotenia klasifikátorov, ktoré sú vynásobené váhovacími koeficientmi α a parametrom β, ktoré transformujú pravdepodobnosť vyhodnocovania. Cieľom lineárnej logistickej regresie je nájdenie optimálnych hodnôt pre koeficientov fúzie. Optimálne hodnoty musia spĺňať podmienku vyjadrenú v rovnici:

$$(\alpha_1, \alpha_2, \ldots, \alpha_N, \beta) = \arg\max C'_{llr}, \tag{2.13}$$

kde C'_{llr} je hodnota úspešnosti rozpoznávania pre sfúzovaný vektor $\vec{l'}()$ nad trénovacou databázou. Pre účel nájdenia koeficientov regresie musela byť trénovacia databáza rozdelená na časť určenú na trénovanie jednotlivých klasifikátorov a na časť určenú na trénovanie fúzie pomocou lineárnej logistickej regresie [24].

Rozdelenie sady bolo v pomere 90 % ku 10 %. Tento pomer sa ukázal ako vhodný, keďže 10 % zo 60 000 bol dostatočný počet vzorov pre natrénovanie parametrov a pritom pre jednotlivé klasifikátory zostalo väčšie množstvo trénovacích vzoriek.

Kapitola 3

Implementácia systémov

Táto kapitola obsahuje popis implementácie jednotlivých systémov. Prvý implementovaný systém, vychádzajúci s popisu v časti 2.1, je popísaný v časti 3.2. Keďže sa jedná o zložitejší proces je popis rozdelený na fázu vytvárania tagovacieho stromu a na fázu vytvárania rozhodovacieho stromu. Ako druhý systém bude popísaný systém založený na neurónových sieťach. V časti 2.2 je teoretický popis metódy učenia neurónovej siete bez učiteľa, ktorá je založená na princípe samo-organizujúcich sa máp. Časť 3.3 bude obsahovať popis práve tohto spôsobu rozpoznávania ručne písaných číslic. V časti 3.4 sa mal nachádzať popis implementácie rozpoznávacieho systému založený na metóde AdaBoost. Táto metóda nebola implementovaná a bol využitý systém, ktorý vytvoril Ing. Michal Hradiš.

Pri všetkých systémoch bol ako implementačný jazyk zvolený jazyk C++. Vzhľadom na charakteristiku úlohy bolo vhodné rozdeliť všetky systémy na dve časti: na trénovaciu a na rozpoznávaciu. Trénovacia časť má za úlohu vytvoriť klasifikátor, ktorý je možné použiť na vyhodnocovanie vzoriek vo fáze testovania. Výstupom trénovacej časti je teda klasifikátor. Ako vhodná forma reprezentácie klasifikátora sa ukázala možnosť XML súboru. Časť, zaoberajúca sa spracovaním XML súborov sa nachádza v časti 3.1.

3.1 XML

XML je rozšírený formát. Jeho nespornou výhodou je veľmi čitateľný spôsob zápisu štruktúrovaných informácií, čo v prípade klasifikátorov je splnené. Nevýhodou je, že informácia uložená v takomto tvare obsahuje množstvo redundantných údajov a veľkosť XML súboru môže byť výraznejšie väčšia, ako keby sa zvolil kompaktnejší spôsob uloženia. V mojom prípade som zvolil možnosť využiť prehľadnejší spôsob zápisu na úkor vyššieho nároku na uloženie klasifikátorov.

Vďaka veľkej popularite XML formátu existuje veľké množstvo knižníc, ktoré dokážu XML súbory spracovávať nebolo nutné vytvárať vlastný modul. Hlavným kritériom na výber vhodnej knižnice bola voľná dostupnosť a jednoduchosť. Knižnica, ktorú som si vybral má názov XMLparser[1].

Knižnica XMLparser je zložená len z jedného hlavičkového a jedného zdrojového súboru. Ich celková veľkosť nie je viac ako 128 KB. Knižnica je multiplatformová a veľký dôraz je kladený na jednoduchosť. Všetky uzly XML stromu majú jednotnú triedu `XMLNode`, takže neexistujú žiadne špeciálne triedy. Takisto knižnica využíva inteligentné ukazovatele, ktoré sú náhradou za neexistujúci garbage kolektor. Výhodou využívania inteligentných

[1] bližšie informácie sú na www stránke `http://www.applied-mathematics.net/tools/xmlParser.html`

ukazovateľov je jednoduchosť správy pamäte, ktorá sa prejaví najmä pri presúvaní uzlov so XML stromu. Takisto je výhodou spracovávanie veľkých XML súborov, ktoré podľa autora knižnice F. V. Berghena je veľmi efektívne.

Hlavnou nevýhodou využitia malej knižnice je limitujúci počet funkcií na spracovávanie XML súboru. V prípade ukladania rozpoznávacích klasifikátorov sa ukázalo vhodné, keby použitá knižnica obsahovala funkciu, ktorá by umožňovala ukladať/načítať polia do/z jedného atribútu XML uzla.

3.2 Systém založený na stromových klasifikátoroch

Implementácia systému založeného na stromových klasifikátoroch bola urobená s ohľadom na rýchlosť trénovania a rozpoznania v jazyku C++. Aj keď jazyk je objektovo orientovaný, v návrhu nie sú žiadne prvky objektového programovania a procesy trénovania a rozpoznávania sú založené na princípoch funkcionálneho programovania. Implementácia nevyužíva okrem štandardnej knižnice jazyka C++ a knižnice XMLparser (časť 3.1) žiadne ďalšie externé moduly.

Samotný algoritmus bol v prípade stromových klasifikátorov rozdelený do troch častí. Prvé dve časti sa používajú pri trénovaní a tretia časť uskutočňuje rozpoznávanie. Jedna časť používaná pri trénovaní vytvára tagovací strom. Druhá časť trénovania pomocou tagovacieho stromu vytvorí rozhodovací strom. Rozhodovací strom obsahuje pravidlá, ktoré sú uplatnené v procese rozpoznávania. Proces trénovania a rozpoznávania bude detailnejšie popísaný v nasledujúcich častiach tejto kapitoly.

Na ukladanie stromov (tagovacích aj rozhodovacích) je zvolená knižnica XMLparser, ktorá bola popísaná v časti 3.1. Knižnica splňuje všetky požiadavky, ktoré sú kladené na načítanie a ukladanie XML súborov. Bez problémov sa dokáže vysporiadať aj s veľkými XML súbormi, ktoré môžu vzniknúť.

3.2.1 Trénovanie

Proces trénovania ma za úlohu vytvoriť rozhodovací strom, ktorý by svojimi pravidlami dokázal popísať číslice. Predpokladom vytvorenia rozhodovacieho stromu je tagovací strom. Úlohou tagovacieho stromu je vytvoriť tagy z významných oblastí.

Tagovacie strom

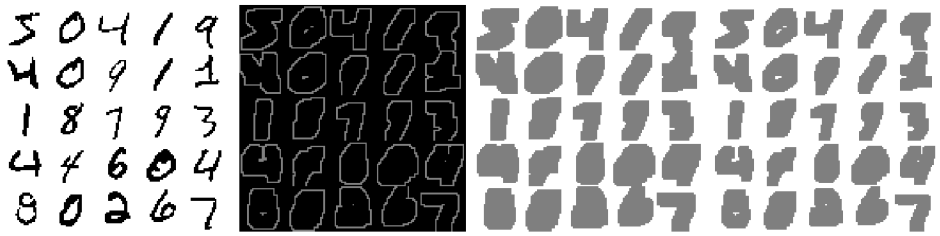
Tagovací strom využíva významné oblasti. Ako bolo popísané v kapitole 2.1.1 významné oblasti obsahujú minimálne jeden čierny a jeden biely pixel a za najlepšiu veľkosť oblasti je považovaný rozmer 4×4. Kvôli efektívnosti sa v článku [12] popisuje možnosť využiť iba 4 stredové pixely. Tento prístup obmedzí počet nájdených oblastí. Zo skupiny úspešne vyhľadaných sa pritom odstránia len tie, ktorých poloha bola zaznamenaná už inými významnými oblasťami. Týmto spôsobom sa obmedzí prekrývanie oblastí. Ako je vidieť v tabuľke 3.1 počet nájdených oblastí sa u číslic znížil približne o tretinu. Na obrázku 3.1 je znázornená zmena veľkosti vyhľadaných významných oblastí. Zmena nie je veľmi výrazná a dokazuje výrazné prekrývanie oblastí.

Po nájdení významných oblastí sa môže zostaviť tagovací strom. Zostavovanie tagovacieho stromu spočíva v dvoch krokoch.

V prvom kroku sa inicializuje koreň stromu a vytvoria sa ukazovatele na všetky oblasti. Každá oblasť je pritom popísaná svojou polohou a ukazovateľom na hodnoty pixelov.

číslica	0	1	2	3	4	5	6	7	8	9
všetky pixely	366	186	330	329	289	314	298	261	320	273
vnútorné 4px	233	110	209	209	183	199	196	164	217	182

Tabuľka 3.1: Priemerný počet nájdených oblastí



Obrázok 3.1: Ukážka nájdených významných oblastí. Prvý obrázok sú originálne číslice. Druhý rozdiel medzi nájdenými oblasťami. Tretí obrázok znázorňuje nájdené oblasti pri porovnávaní všetkých pixelov. Štvrtý obrázok znázorňuje nájdené oblasti pri porovnávaní iba vnútorných pixelov.

Druhý krok je vystavanie stromu. Vystavanie je rekurzívna funkcia, ktorej ukončenie je dané hĺbkou stromu. Najskôr sa vyberie pixel, ktorý bude rozdeľovať oblasti do dvoch skupín. Výber najvhodnejšieho pixelu spočíva v spočítaní bielych a čiernych pixelov pre každú pozíciu. Počet, ktorý sa najviac približuje polovici počtu oblastí je najvýhodnejší, lebo rozdeľuje oblasti na približne rovnaké skupiny. Po určení pixelu môže prebehnúť samotné rozdelenie. Rozdelenie prebieha pomocou ukazovateľov na oblasti. Ukazovatele sa zatriedia pomocou triediaceho algoritmu, kde biele pixely majú menšiu hodnotu ako väčšie. V uzloch stromu potom netreba ukladať celý zoznam oblastí, ale stačí ukladať informáciu o tom, ktorá časť z ukazovateľov danému uzlu patrí a počet bielych pixelov. Okrem týchto hodnôt sa v uzle musí uložiť pozícia pixelu, ktorý je považovaný za deliaci.

Výsledný strom sa potom uloží do XML formátu. Nižšie sa nachádza ukážka, ako XML súbor obsahujúci tagovací strom vyzerá. Hodnoty atribútov boli nahradené typom používaným v jazyku C/C++. Podstatou tagovacieho stromu je zatriedenie oblastí do skupín a preto sa v uzloch ukladajú len informácie o deliacich pixeloch. V prípade listov sa deliaci pixel už neukladá.

```
<?xml version="1.0" encoding="utf-8"?>
<tagTree typeCnt="int" tagSize="int" DividePixelPoxX="int" DividePixelPoxY="int">
    <left DividePixelPoxX="int" DividePixelPoxY="int">
        ...
            <left DividePixelPoxX="int" DividePixelPoxY="int">
                <left type="int"/>
                <right type="int"/>
            </left>
            <right DividePixelPoxX="int" DividePixelPoxY="int">
                <left type="int"/>
                <right type="int"/>
            </right>
        ...
```

```
    </left>
    <right DividePixelPoxX="int" DividePixelPoxY="int">
        ...
    </right>
</tagTree>
```

V koreňovom uzle pomenovaný ako `tagTree` sa okrem pozície ukladá aj veľkosť oblasti reprezentovaný atribútom `tagSize` (v tejto implementácii vždy hodnota 4) a počet kategórií reprezentovaný atribútom `typeCnt`. Počet typov zároveň udáva aj počet listov, keďže každý list určuje jeden typ tagu. V liste sa preto ukladá typ tagu.

Rozhodovací strom

Rozhodovacie stromy sú podstatou celého algoritmu. Ich tvorba je časovo najnáročnejšia. Zostavovanie prebieha podobne ako pri tagovacích stromoch. Najskôr prebehne inicializácia a potom sa rekurzívne vytvára strom.

Inicializácia stromu pozostáva s vytvorenia zoznamu číslic. Tento zoznam sa bude triediť na dve skupiny podľa toho, či daná číslica obsahuje alebo neobsahuje pravidlo. Číslica, ktorá obsahuje pravidlo, bude mať pri triedení nižšiu hodnotu ako tá, ktorá pravidlo neobsahuje. Bude sa teda nachádzať v ľavej časti zoznamu.

Druhý krok je samotné vytvorenie stromu. Najskôr sa vyberie pravidlo, podľa ktorého sa budú číslice deliť do dvoch častí. Výber pravidla je najdôležitejší krok. Ako už bolo spomínané v časti 2.1.2 pravidlo sa skladá z troch častí: prvý typ tagu, druhý typ tagu a smer. Pokiaľ sa uzol nachádza v ľavej strane stromu resp. podstromu, je nutné brať do úvahy uplatnené pravidlá. V takomto prípade je nutné, aby prvou časťou pravidla bol typ tagu, ktorý bol uplatnený v predchádzajúcich uzloch. Pokiaľ sa uzol nenachádza v ľavej časti, je treba vybrať ako prvý typ tagu jeden zo všetkých existujúcich typov. Výber druhého typu tagu nie je ovplyvnený pravidlami, ktoré sa už uplatnili, keďže smer medzi týmito tagmi môže byť odlišný. Samozrejme je nemožné, aby sa v rodičovskom uzle a uzle potomka nachádzalo ako deliace kritérium rovnaké pravidlo. Táto možnosť je vylúčená v procese vyhodnotenia úspešnosti pravidla metrikou. Na ohodnotenie sa v súčasnej implementácii využíva metrika 2.1 popísaná v časti 2.1.2, ktorá vychádza z práce [7]. Pre výber najvhodnejšieho pravidla je nutné zmerať ako dobre pravidlo rozdelí číslice a v prípade už uplatneného pravidla sa všetky uzly zaradia do jedného podstromu. Hodnota x_ϕ v rovnici 2.1 (priemerný počet číslic obsahujúcich pravidlo) bude preto maximálna možná. Výsledná hodnota metriky spomínaného prípadu bude vysoká a ľubovolné iné kritérium by bolo lepšie (malo by menšiu hodnotu).

Po výbere pravidla sa usporiadajú číslice v danom zozname. Do uzlu sa opäť uloží informácia o začiatku a konci časti zoznamu, ktorá uzlu patrí. Tiež sa uloží počet číslic spĺňajúcich pravidlo a samozrejme aj pravidlo, ktoré číslice rozdelilo. V prípade, že uzol sa má stať listom tak sa uloží aj štatistika o tom, koľko percent akých typov číslic sa do listu dostalo. Uzol sa môže stať listom v troch prípadoch:

1. uzol sa nachádza v maximálnej výške stromu – Obmedzenie sa dá odôvodniť rýchlosťou implementácie. V prípade malej výšky stromu je vysoká pravdepodobnosť zlej klasifikácie. V prípade príliš veľkej hodnoty výšky prehľadávanie stromu môže byť časovo náročné.

2. počet číslic dosiahlo minimum – Ak sa v uzle nachádza málo číslic výpovedná hodnota listu bude malá.

3. typ číslice je dominantný – Ak sa stane uzol listom z tohto dôvodu, znamená to, že číslica, ktorá sa pri rozpoznávaní prepracuje do tohto listu, bude s percentuálnou hodnotou rovnajúcej sa dominancii typu označená za daný typ. Ukončenie rekurzie s tohto dôvodu je najvýhodnejšie.

Po nájdení všetkých pravidiel a vytvorení stromu nasleduje jeho uloženie do XML súboru. Ukážka rozpoznávacieho stromu je nižšie. Koreňový uzol má názov `randomizedTree` a oproti zvyšným uzlom obsahuje atribút `recognitionCnt`. Tento atribút udáva počet znakov, ktoré sa rozpoznávajú, kedže algoritmus nemusí byť použitý iba na rozpoznávanie číslic. V prípade rozpoznávania číslic je hodnota atribútu vždy 10.

```
<?xml version="1.0" encoding="utf-8"?>
<randomizedTree recognitionCnt="int" decisionTagType1="int"
                decisionTagType2="int" direction="int">
   <left decisionTagType1="int" decisionTagType2="int" direction="int">
      ...
         <left decisionTagType1="int" decisionTagType2="int" direction="int">
            <left char???Cnt="float" ... char???Cnt="float"/>
            <right char???Cnt="float" ... char???Cnt="float"/>
         </left>
         <right decisionTagType1="int" decisionTagType2="int" direction="int">
            <left char???Cnt="float" ... char???Cnt="float"/>
            <right char???Cnt="float" ... char???Cnt="float"/>
         </right>
      ...
   </left>
   ...
</randomizedTree>
```

Všetky uzly obsahujú pravidlo. Ako bolo niekoľkokrát spomenuté skladá sa z troch častí. Tie sú v uzle uložené v troch atribútoch: `decisionTagType1`, `decisionTagType2` a `direction`. Hodnoty typu tagu sú zhodné s jednou z hodnôt typu listu tagovacieho stromu. Smer (atribút `direction`) môže nadobúdať hodnoty od 0 do 7. Hodnota 0 reprezentuje sever. Zvyšujúca sa hodnota je posun na kompase v smere hodinových ručičiek o $\pi/4$.

V listoch sa nachádza štatistika udávajúca percentuálne zastúpenie typov znakov zo všetkých znakov v liste. Počet atribútov `char???Cnt` je daný hodnotou `recognitionCnt` v koreňovom uzle stromu. Na miesto `???` je doplnená hodnota 0 až `recognitionCnt`-1.

3.2.2 Rozpoznávanie

Proces rozpoznávania je časovo oveľa menej náročný. Na začiatku je nutné načítať testovaciu sadu a tagovací strom. Z týchto súborov sa vytvorí testovacia množina. Množina už obsahuje vyhľadané tagy. Číslice sú teda pripravené na rozpoznávanie. Načíta sa rozpoznávací strom a pomocou pravidiel sa jednotlivé vzorky zadeľujú do listov. Podľa toho do akého listu číslica dospela sa rozhodne o jej type. Typ je určený pomocou štatistiky, ktorá sa v liste nachádza. Štatistika udáva percentuálny podiel zastúpených typov číslic počas trénovania. Najväčšia hodnota percentuálneho podielu potom určuje typ číslice a pravdepodobnosť jej správneho určenia.

Počas rozpoznávania sa zbierajú rôzne štatistické informácie. Najdôležitejšou informáciou je úspešnosť rozpoznávania. Okrem úspešnosti rozpoznávania nás môžu zaujímať aj

iné vlastnosti siete napr., ktorá číslica je najčastejšie rozpoznávaná úspešne alebo aké typy číslic sa medzi sebou najviac pletú. Tieto údaje sú obsiahnuté v matici zámen.

Matica zámen pre prípad rozpoznávania ručne písaných číslic je tabuľka s 10 riadkami a 10 stĺpcami. V riadkoch sa nachádza informácia o jednotlivých typoch číslic, ktoré boli na vstupe. Súčet všetkých stĺpcov v riadku musí byť 100 %. V stĺpcoch je obsiahnutá informácia, za akú číslicu bol daný vstup prehlásený. Optimálny klasifikátor by mal na diagonále obsahovať 100 %. Počet percent pre neúspešne rozpoznané číslice je teda suma všetkých hodnôt v riadku okrem hodnoty nachádzajúcej sa na diagonále. Jednoduchý klasifikátor, ktorý bude na každý vstup odpovedať napr. hodnotou `0` bude mať v stĺpci `0` všade 100 %, ale len pre riadok `0` bude na 100 % úspešný.

	0	1	2	3	4	5	6	7	8	9
0	**0.72**	0.01	0.02	0.01	0.01	0.01	0.15	0.02	0.03	0.02
1	0.00	**0.95**	0.01	0.00	0.01	0.00	0.01	0.00	0.01	0.01
2	0.03	0.01	**0.58**	0.15	0.02	0.05	0.02	0.08	0.03	0.03
3	0.01	0.01	0.15	**0.70**	0.01	0.05	0.01	0.03	0.02	0.01
4	0.02	0.21	0.01	0.00	**0.58**	0.01	0.04	0.06	0.01	0.06
5	0.03	0.01	0.05	0.11	0.01	**0.65**	0.02	0.02	0.06	0.02
6	0.09	0.05	0.01	0.01	0.05	0.01	**0.73**	0.00	0.04	0.01
7	0.01	0.02	0.04	0.03	0.03	0.01	0.00	**0.80**	0.01	0.05
8	0.06	0.04	0.04	0.05	0.03	0.08	0.09	0.03	**0.52**	0.06
9	0.03	0.04	0.03	0.01	0.05	0.03	0.02	0.17	0.03	**0.59**

Tabuľka 3.2: Ukážka matice zámen stromového klasifikátora.

3.2.3 Príprava na fúzovacie metódy

Ako bolo popísané v úvode časti o fúzovacích metódach 2.5, aby bolo možné zvýšiť úspešnosť rozpoznávacích systémov, je nutné, aby bola implementácia nedeterministická. V prípade systému rozpoznávania ručne písaných číslic je táto nedeterminickosť zahrnutá v procese výberu najlepšieho pravidla. Keďže nie je časovo možné, aby boli vyhodnotené metrikou všetky varianty. Ďalším prvkom, ktorý prispieva k vzniku rozdielnych klasifikátorov je náhodný výber oblastí v procese tvorby rozhodovacieho stromu, ktoré sa majú preskúmať ako významné oblasti. Tento počet nie je síce tak časovo náročný, ale pri uchovávaní informácie o možných pravidlách, ktoré sa môžu ešte uplatniť vo vzore, by vznikalo extrémne množstvo možných kombinácií.

3.3 Systém samo-organizujúcich máp

Kvôli kompatibilite s predchádzajúcim programom bol ako implementačný jazyk zvolený jazyk C++. Ako bolo spomínané v úvode kapitoly trénovanie a rozpoznávanie sú nezávislé časti, preto boli aj v prípade samo-organizujúcich máp vytvorené dva samostatne spúšťateľné programy *train* a *test*. Na ukladanie samo-organizujúcich sa máp bolo opäť s výhodou využitá knižnica XMLparser (časť 3.1).

3.3.1 Trénovanie

V tejto časti je popísaný samotný postup učenia siete. Úlohou programu na trénovanie je vytvorenie neurónovej siete s lepšou schopnosťou rozpoznávania ako tá predchádzajúca. Výstupom je preto vždy neurónová sieť. Trénovací program má dve varianty:

1. vytvorenie novej siete a
2. pokračovanie v trénovaní existujúcej siete.

Najskôr sa vykonáva inicializácia podľa algoritmu 2.2.1. Vytvorí sa dvojrozmerná matica s neurónmi. Počet váhovaných vstupov pre každý neurón je rovnaký ako počet pixelov obrázka. Pri vytváraní novej siete sa hodnoty váh inicializujú uniformným rozložením od 0 do 1. Pri pokračovaní v trénovaní sa váhy načítajú zo súboru. Pre každé trénovacie kolo sa SOM upravuje nasledujúcim spôsobom. Sekvenčne sa prechádzajú vstupné obrázky číslic a pre každý vstup sa vyberie víťazný neurón. Neurónu a jeho okoliu sa následne modifikujú váhy. Na rozdiel od pôvodnej rovnice na modifikovanie váh 2.2 je spôsob výpočtu váh jemne odlišný. Berie sa do úvahy aj vzdialenosť od víťazného neurónu.

$$w_{ij}(k+1) = w_{ij}(k) + \eta(k)\left(i_p(k) - w_{ij}(k)\right)/d \tag{3.1}$$

Vzdialenosť d je Mahnattanská vzdialenosť od víťazného neurónu. Táto úprava dokázala výrazne zlepšiť trénovací proces.

Počas trénovacieho procesu sa ďalej ku každému víťaznému neurónu ukladá aj informácia o tom, pre aký typ číslice je daný neurón víťazný. Neurón je potom reprezentáciou typu číslice, ktorá sa počas trénovania najčastejšie stala vstupom pre daný víťazný neurón (ukážka natrénovaných neurónov v mape je vidieť v tabuľke 3.3). Po natrénovaní sa SOM uloží do súboru.

Aby bolo možné s jednotlivými SOM pracovať, rozhodol som sa znovu využiť možnosti formátu XML a výsledné siete ukladať práve do súborov s príponou .xml.

Súbor, ktorý obsahuje uloženú Kohonenovu sieť vyzerá nasledovne:

```
<?xml version="1.0" encoding="utf-8"?>
<network round="int" inputSize="int">
   <row>
      <neuron type="float ...  float" weight="float ...  float"/>
      <neuron type="float ...  float" weight="float ...  float"/>
   </row>
   ...
   <row>
      <neuron type="float ...  float" weight="float ...  float"/>
      <neuron type="float ...  float" weight="float ...  float"/>
   </row>
</network>
```

Hodnoty atribútov sú nahradené dátovým typom používaným v jazyku C/C++. Z obsahu súboru je jasné, že sa jedná o sieť, ktorá bola natrénovaná v kole udanom hodnotou atribútu `round`. Hodnota atribútu `inputSize` udáva počet vstupov tzn. *šírka* × *výška* vstupného obrázka. Táto hodnota je zároveň aj počet hodnôt typu *float* v atribúte `weight`. Samotná sieť je rozdelená do elementov `row`. Každý element `row` potom obsahuje element `neuron`, ktorý reprezentuje samotný neurón siete. Neurón má dva atribúty. Jednym sú už spomínané

váhy a druhým je vlastnosť nazývaná *type*. Pod týmto pojmom sa rozumie rozdelenie pravdepodobnosti pre jednotlivé triedy, ktorú neurón zastupuje.

Výstupom SOM je trieda, ktorá má najvyššiu hodnotu spomínanej vlastnosti type. Výstup z jedného klasifikátora môže vyzerať napríklad ako v tabuľke 3.3.

7	7	7	7	9	9	1	1	1	1	2	2	2	1	1
7	7	7	7	7	4	7	1	1	1	2	2	2	1	1
7	7	2	7	7	4	2	2	1	1	1	8	3	1	1
8	8	2	7	4	4	2	2	2	2	2	2	6	2	1
8	8	8	9	9	9	3	2	2	2	2	2	6	6	5
8	8	8	3	9	9	9	2	2	2	2	3	5	5	5
8	8	8	5	9	9	9	9	7	3	3	3	3	3	2
5	5	5	5	8	4	9	9	9	4	3	3	3	3	3
0	0	2	5	6	6	9	4	4	4	3	3	3	5	3
0	0	0	6	6	6	6	4	4	4	3	3	3	5	3
0	0	0	6	6	6	6	4	4	4	3	5	5	5	3
0	0	0	0	6	6	6	6	4	4	7	4	4	8	8
0	0	0	6	6	6	6	6	4	9	9	7	4	4	8
0	0	0	6	6	5	6	4	4	9	3	7	7	5	4
0	0	3	5	5	5	5	4	4	7	7	7	7	7	4

Tabuľka 3.3: Ukážka výstupných hodnôt SOM.

3.3.2 Rozpoznávanie

Proces rozpoznávania je oproti trénovaniu výrazne jednoduchší. Najskôr sa načíta Kohonenova mapa. K načítanej sieti sa potom prikladá položka z dátovej sady. Následne sa určí víťazný neurón a podľa jeho typu je vstup prehlásený za danú číslicu. Najdôležitejším údajom rozpoznávacieho procesu je percentuálny podiel správne klasifikovaných vzorov. Číslica je úspešne rozpoznaná, ak sa typ víťazného neurónu zhoduje z hodnotou získanou z referenčného súboru.

Pre porovnanie výsledkov rozpoznávania pomocou SOM a stromového klasifikátora je v tabuľke 3.4 zobrazená matica zámen pre samo-organizovaciu mapu.

	0	1	2	3	4	5	6	7	8	9
0	**0.98**	0.00	0.00	0.00	0.00	0.01	0.01	0.00	0.00	0.00
1	0.00	**0.99**	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00
2	0.10	0.04	**0.74**	0.01	0.01	0.00	0.01	0.05	0.03	0.00
3	0.10	0.02	0.05	**0.73**	0.00	0.03	0.00	0.02	0.04	0.01
4	0.10	0.02	0.01	0.01	**0.76**	0.00	0.01	0.02	0.01	0.06
5	0.16	0.02	0.01	0.13	0.05	**0.56**	0.01	0.02	0.02	0.02
6	0.12	0.01	0.01	0.00	0.05	0.04	**0.77**	0.00	0.00	0.00
7	0.05	0.03	0.02	0.01	0.01	0.00	0.00	**0.85**	0.01	0.02
8	0.13	0.02	0.05	0.06	0.02	0.07	0.01	0.05	**0.58**	0.01
9	0.06	0.01	0.04	0.01	0.16	0.01	0.00	0.07	0.01	**0.63**

Tabuľka 3.4: Ukážka matice zámen SOM.

Táto matica dosahuje celkovej úspešnosti v priemere okolo 76 %. Z matíc je vidno, že najlepšie výsledky dosahuje v prípade číslic 0 a 1. Najmenej úspešne rozpoznávanie bolo pri číslicach 5 a 8. Nesprávne zaklasifikovanie sa dá pripísať roztrúsenosti víťazných neurónov po Kohonenovej mape. Nevýhodou Kohonenových máp je aj to, že jednotlivé matice zámen sú rôznorodejšie ako v prípade stromových klasifikátorov. To znamená, že u niektorých máp sú výrazne odlišné úspešnosti rozpoznávania typov číslic ako v prípade SOM, použitej pri vytváraní matice zámen 3.4.

Výstupom testovacieho programu je súbor, ktorého názov je odvodený od XML súboru s uloženou SOM. Tieto súbory majú príponu `.out`. Každý takýto súbor obsahuje pravdepodobnosť príslušnosti vzoriek do všetkých tried. Informácia o počte úspešne klasifikovaných číslic sa vypisuje naštandardný výstup.

3.3.3 Príprava na fúzovacie metódy

Keďže jednotlivé Kohonenove mapy sa nedokázali dobre vysporiadať s veľkým množstvom trénovacích vzoriek a veľmi rýchlo sa pretrénovali, existovalo jediné rozumné riešenie a to rozdeliť trénovacie dáta do menších skupín a natrénovať viacej klasifikátorov. Druhou možnosťou bolo zvoliť trénovacie konštanty dostatočne malé, ale trénovanie by sa stalo extrémne časovo náročným.

Pri trénovaní teda vznikne určitý počet klasifikátorov založených sa SOM. Tieto klasifikátory budeme označovať ako *slabé*. Zlúčením väčšieho počtu slabých klasifikátorov vznikne jeden *silný*, ktorého úspešnosť rozoznávať číslice je lepšia.

Keďže slabé klasifikátory vznikli rovnakým spôsobom, je pravdepodobné, že úspešnosť ich rozpoznávania bude tiež rovnaká. Z tohto dôvodu sa silný klasifikátor netrénuje žiadnou ďalšou metódou strojového učenia.

3.4 AdaBoost

Táto kapitola by mala obsahovať implementáciu systému na rozpoznávanie ručne písaných číslic pomocou metódy AdaBoost. Keďže som tento spôsob rozpoznávania neimplementoval, tak sa chcem poďakovať M. Hradišovi, ktorý sa na FIT VUT venuje rozpoznávacím a detekčným systémom založených na AdaBoost metódach, za vyhodnotenie odoziev nad MNIST databázou. Jeho implementácia využíva princíp popísaný v kapitole 2.3 a ako príznaky pre rozpoznávanie boli použité Multi-Block Local Binary Pattern, ktoré sú tiež popísané v danej kapitole.

Kapitola 4

Testovanie

Testovanie systémov je veľmi dôležitá časť práce, lebo práve z výsledkov pri testoch je možné posúdiť, či je daný systém účinný a ako sa chová pri zmene jednotlivých parametrov.

Táto kapitola je zameraná hlavne na nastavenie jednotlivých parametrov pri rozpoznávacích systémoch. V časti 4.1 sú popísané grafy, ktoré objasňujú vývoj úspešnosti klasifikátora pri zmene jednotlivých parametrov, ako sú výška tagovacieho a rozhodovacieho stromu, ukončovacie podmienky generovania rozhodovacieho stromu (minimálny počet uzlov v uzle a dominantné zastúpenie jedného typu číslice) a parametre ako počet vyhľadávaných tagov.

Druhá časť (sekcia 4.2) je zameraná na testovanie samo-organizujúcich sa máp. V prípade SOM je počet parametrov výrazne nižší. Základom je určenie veľkosti mapy.

Sekcia 4.3 sa zaoberá rôznymi spôsobmi, ako zvýšiť úspešnosť klasifikátorov. Ide hlavne o fúzovacie metódy, ktorých princípom je dodanie ďalších informácií do rozhodovacieho procesu.

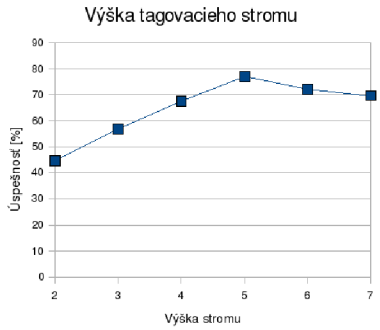
4.1 Systém založený na stromových klasifikátoroch

V tejto časti budú popísané niektoré vlastnosti rozhodovacieho stromu a ich vplyv na úspešnosť rozpoznávania. Testovanie bude rozdelené do dvoch častí. V oboch častiach sa bude testovať úspešnosť rozpoznávania pre tagy o rozmeroch 4×4. Prvý test sa bude zaoberať tagovacím stromom. Zvyšné testy budú skúmať vlastnosti rozhodovacieho stromu, kde základnou otázkou bude, či vyhľadávať všetky významné oblasti, alebo len oblasti, ktoré obsahujú čierny aj biely pixel vo vnútorných štyroch pixeloch. Zároveň je skúmaná aj ďalšia podstatná vlastnosť rozhodovacích stromov a to je ich výška.

4.1.1 Výška tagovacieho stromu

Jednou z prvých otázok pri rozpoznávaní ručne písaných číslic pomocou stromových klasifikátorov je výška tagovacieho stromu. Tagovací strom je zodpovedný za klasifikáciu významných oblastí do tried. Čím je strom vyšší, tým je viac tried. Viac tried znamená aj menšiu možnosť označenia rozličných významných oblastí rovnakým tagom a tagy sa stávajú špecifickejšie.

Na grafe na obrázku 4.1 je znázornená závislosť úspešnosti vyhodnocovania číslic na výške použitého tagovacieho stromu. Je zreteľne vidieť, že pokiaľ sa využijú len tagy, ktoré sú reprezentované listami tagovacieho stromu, je optimálna výška stromu 5. Pre hodnoty 2, 3 a 4 sú tagy veľmi obecné a určené pravidlá sú ľahko splniteľné. V prípade výšky stromu 6 a 7 úspešnosť rozhodovania je nižšia. Príčinou je väčšie množstvo tagov a rozhodovací strom

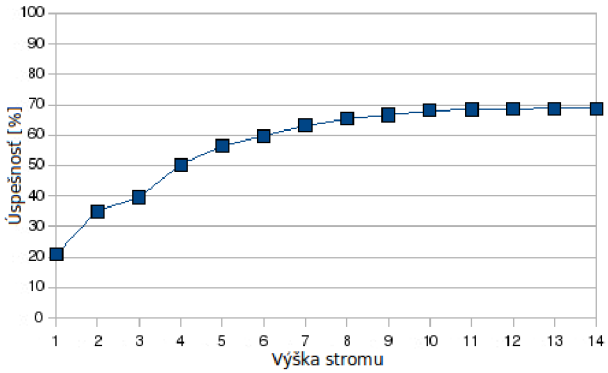


Obrázok 4.1: Úspešnosť rozpoznávania číslic v závislosti na výške tagovacieho stromu.

o hĺbke 14, ktorý bol použitý pre testovanie úspešnosti, nedokázal vytvoriť dostatočný počet rozhodnutí pre číslice (obsahoval málo listov). Následné pravidla boli veľmi špecializované a v listoch sa nachádzali ešte neroztriedené číslice.

4.1.2 Všetky oblasti

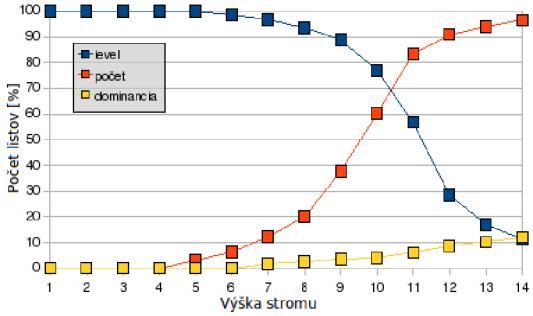
Faktor, ktorý pri rozpoznávaní výrazne ovplyvňuje úspešnosť je výška rozhodovacieho stromu. Ako konštantné parametre pri vytváraní stromu boli určené minimálny počet číslic a dominancia typu číslice. Minimálny počet bol stanovený na 30 vzoriek a dominancia bola zvolená 98 %.



Obrázok 4.2: Úspešnosť rozpoznávania v závislosti na výške stromu.

Úspešnosť takto nastaveného stromu dosahuje hranice 70 %. V grafe 4.2 je vidieť, že nárast úspešnosti do výšky 8 je výrazný. Následne je správne rozhodnutie o type číslice približne rovnaké, aj keď stále mierne stúpa. Za optimálnu výšku rozhodovacieho stromu sa dá považovať hodnota 13 prípadne 14.

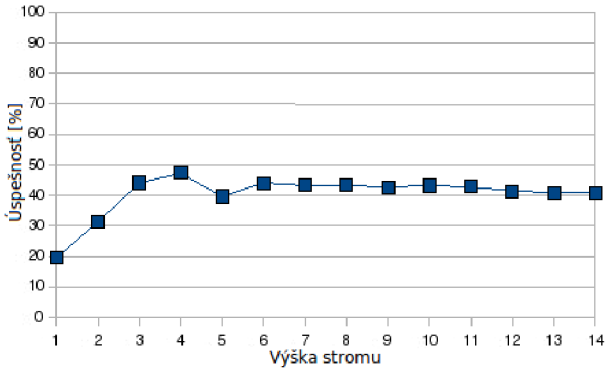
Pre zaujímavosť bol vytvorený graf, ktorý popisuje dôvod ukončenia uzlu. Uzol sa môže stať listom kvôli trom dôvodom spomínaných v časti 3.2.1. Pre jeden uzol, ale môže platiť viacero dôvodov súčasne. Veľmi často napríklad platí, že ak je uzol ukončený kvôli dominancii jedného typu v uzle, tak je zároveň aj ukončený z dôvodu nízkeho počtu vzorov. Aby boli výsledky z jednotlivých stromov porovnateľné, v grafe 4.3 je zaznačený počet ukončení listov s daným dôvodom ku celkovému počtu listov stromu.



Obrázok 4.3: Percentuálne zastúpenie ukončenia listov v závislosti na výške stromu.

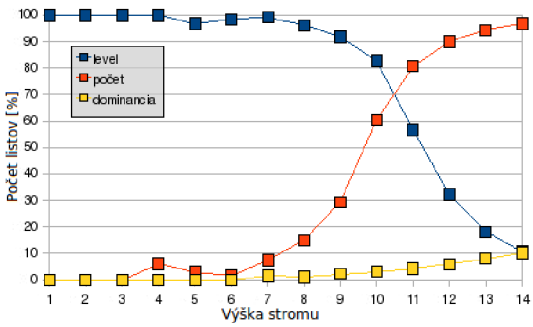
4.1.3 Oblasti so zmenou hodnoty vo vnútorných pixeloch

V tejto časti je obdobný test ako v predchádzajúcej sekcii. Nastavenie parametrov zostalo zachované. Zmenený bol algoritmus na vyhľadávanie významných oblastí v obrázku. Namiesto porovnávania všetkých 16 pixelov na rovnakú farbu boli porovnávané iba štyri vnútorné pixely.



Obrázok 4.4: Úspešnosť rozpoznávania v závislosti na výške stromu.

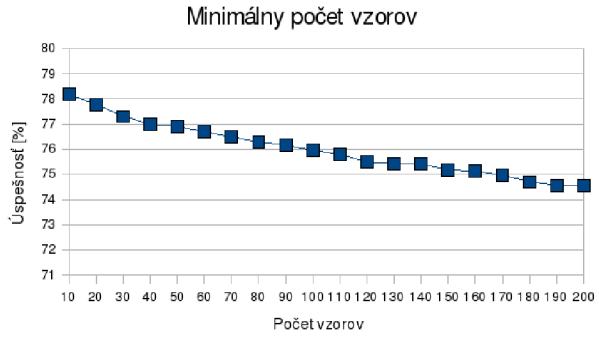
V grafe 4.4 je vidieť, že úspešnosť stromu klesla na hranicu 47 %. Maximálna hodnota úspešnosti je dosahovaná pri výške stromu 4. Podľa grafu 4.5 je, výška 4 nedostačujúca, keďže všetky listy boli ukončené kvôli dosiahnutiu maximálnej výšky. Pri vyšších hodnotách výšky stromu hodnota úspešnosti klesne a následne sa drží na úrovni 43 %. Tento počet sa dá považovať za reálny výsledok rozpoznávania.



Obrázok 4.5: Percentuálne zastúpenie ukončenia listov v závislosti na výške stromu.

4.1.4 Minimálny počet vzorov v uzle

Minimálny počet vzorov v uzle je jedným z parametrov, ktorý ovplyvňuje ukončenie tvorby rozhodovacieho stromu. Ak počet vzorov číslic v uzle rozhodovacieho stromu počas trénovania klesne pod udávanú hodnotu je proces generovania stromu ukončený a pre uzol nie je vytvorený žiadny podstrom. Z uzla sa stáva list. Všetky vzory následne slúžia na zostavenie histogramu, ktorý sa využije pri rozhodovaní o číslicach počas samotného rozpoznávania.

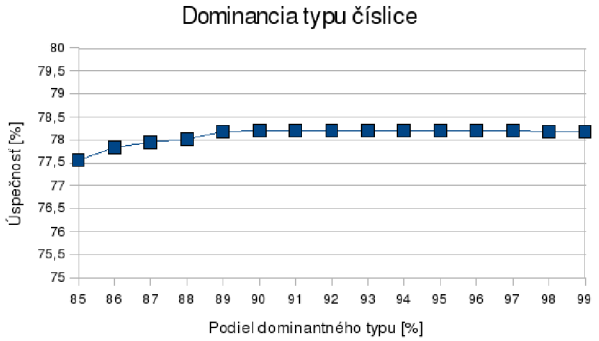


Obrázok 4.6: Vplyv minimálneho počtu vzorov v uzle na úspešnosť vyhodnocovania.

Na grafe 4.6 je vidieť závislosť minimálneho počtu vzorov v uzloch na úspešnosť vyhodnocovania. Je zjavné, že čím je minimálny počet vyšší, tým je úspešnosť rozhodovania menšia. Zistené chovanie je aj logicky správne, keďže v prípade, že je číslic v uzle málo, stále sa dokážu pomocou metriky rozdeliť tak, aby vybrané pravidlo rozdelilo vzory podľa typu. Uzol môže ďalej pokračovať v testovaní vlastností vzoru a potom zaklasifikovať číslice do viacerých tried namiesto jednej, do ktorej by musel zaradiť všetky vzory, ak by bol listom.

4.1.5 Dominantné zastúpenie typu vzorov v uzle

Posledným z troch možných dôvodov ukončenia generovania rozhodovacieho stromu z uzla je dominantné zastúpenie jedného typu číslic v uzle. Tento spôsob ukončenia je najviac žiadaný. V prípade, že sa tak v procese tvorby stromu udeje je možné povedať, že pravdepodobnosť, že číslica je daného typu, je väčšia ako zadaná hodnota zastavenia generovania. Ak napríklad uzol obsahoval 95 % číslic typu 1 je 95 % pravdepodobnosť, že vzor na vstupe je jednotka.



Obrázok 4.7: Závislosť ukončenia listov pre dominantné zastúpenie typu vzorov na úspešnosť rozhodovania.

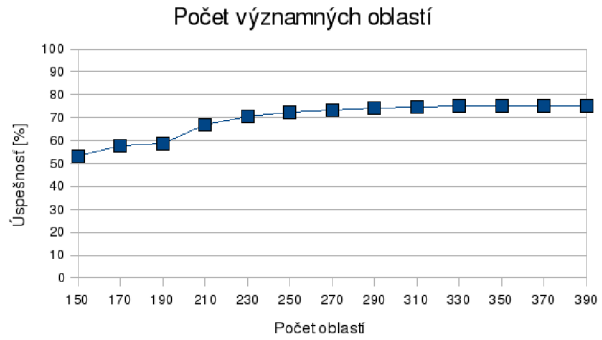
Na grafe na obrázku 4.7 je znázornené ako vplýva zmena dominantného zastúpenia typu

vzoru v uzle. Prekvapujúco sa zmeny hodnôt odrážajú v úspešnosti len v rozmedzí jedného percenta. Dôvodom môže byť vplyv minimálneho počtu číslic na tento spôsob ukončenia. Keďže metrika použitá pri vytváraní stromu využíva pomer vyváženia stromu a separácie jednotlivých typov číslic 7 : 1 je možnosť väčšinového zastúpenia jedného typu v uzle pri vyššom počet vzorov (200 kusov) nízka. V prípade nižšieho počtu vzorov v uzle nie je nutné presne určiť minimálnu hodnotu dominancie. Na grafe je vidieť, že úspešnosť sa od hodnoty 90 % už nezvyšuje. Za optimálne nastavenie parametru dominancie typu v uzle sa dá považovať spomínaná hodnota 90 %. Testovací strom mal ako zvyšné dve podmienky ukončenia nastavené minimálne 30 vzorov v jednom uzle a maximálnu výšku stromu 14.

4.1.6 Počet tagov vo vzore

Ako jeden zo spôsobov ako urýchliť rozpoznávanie ručne písaných číslic pomocou rozhodovacích stromov je nespracovávať všetky významné oblasti. Menší počet oblastí výrazne znižuje čas potrebný na tvorbu rozhodovacieho stromu. Urýchlenie sa deje tam, kde je nutné vytvárať zoznamy možných pravidiel, ktoré obsahujú tagy spĺňajúce všetky predchádzajúce pravidlá.

Významné oblastí pri veľkosti vzoru 28×28 a veľkosti tagov 4×4 sa môžu vyskytovať na 576 pozíciách. Ako už bolo spomenuté v 3.1 priemerný počet tagov v čísliciach sa pohybuje od 186 do 366 kusov. Keďže niektoré číslice obsahujú viac a niektoré menej významných oblastí ako priemerný počet, tak rozmedzie testovacieho počtu bolo zvolené od 150 do 390.

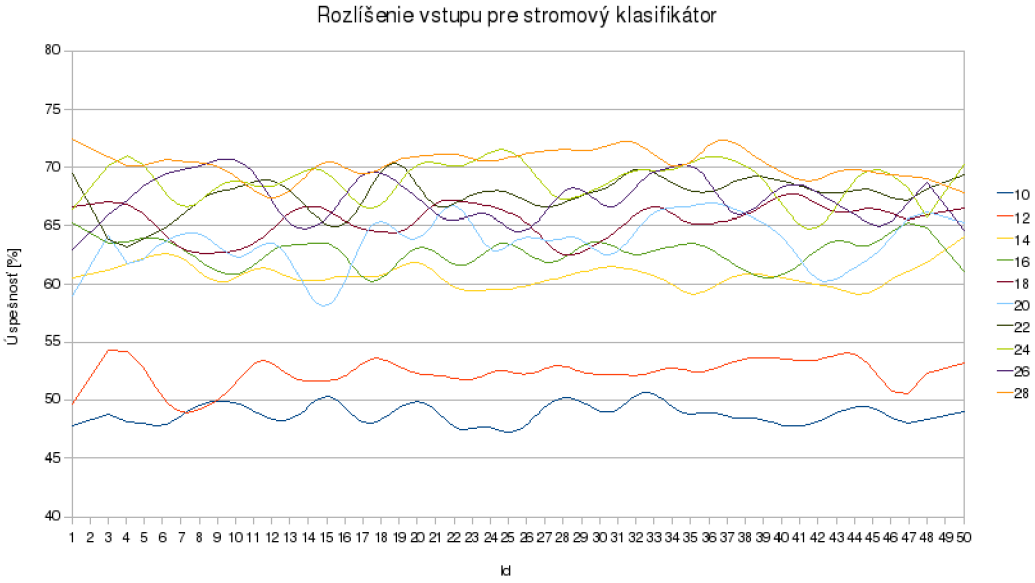


Obrázok 4.8: Úspešnosť rozpoznávania ručne písaných číslic v závislosti na počte vyhľadávaných tagov.

Ako je vidieť na grafe 4.8 úspešnosť rozpoznávania rastie so zvyšujúcim sa počtom významných oblastí. Pre hodnoty vyššie ako 300 významných oblastí sa krivka na grafe vyrovnáva. Nezvyšovanie úspešnosti je spôsobené tým, že nie vo všetkých vzoroch sa nachádza dostatočný počet oblastí, ktoré môžu byť považované za významné. V takomto prípade sa na ďalšie spracovanie využije menší počet oblastí ako bol zadaný. Graf teda znázorňuje maximálny počet tagov použitých na ďalšie spracovanie pri rozhodovacom strome.

4.1.7 Rozlíšenie vstupného obrázka

Príčinou testovania rozlíšenia vstupného obrázka bolo nájsť najlepšie rozlíšenie vstupu ak zostane zachovaný rozmer významných oblastí a tagov (4×4px). Očakávaný výsledok, že najlepšie rozpoznávanie číslic bude na nezmenšených obrázkoch sa potvrdil, takisto sa potvrdilo, že najhorší výsledok dosiahnú klasifikátory s najmenšou dĺžkou strany.



Obrázok 4.9: Úspešnosť jednotlivých stromových klasifikátorov, ktoré boli trénované na podvzorkovaných vstupoch.

Na grafe 4.9 sú zobrazené aproximované hodnoty pre jednotlivé stromové klasifikátory. Je evidentné, že najhoršie dopadli klasifikátory s rozlíšením 10×10 a 12×12. Vo zvyšných prípadoch existujú klasifikátory, ktoré svojou úspešnosťou dokážu prekonať klasifikátory s iným rozlíšením. Preto je pre určenie najlepšieho rozlíšenia vstupu vhodné brať do úvahy priemernú hodnotu správnosti vyhodnotenia. Priemerné hodnoty klasifikátorov natrénovaných na rozdielnych rozlíšeniach vstupného obrázka sú znázornené v tabuľke 4.1.

dĺžka strany [px]	10	12	14	16	18	20	22	24	26	28
úspešnosť [%]	48,87	52,46	60,72	62,76	65,52	63,69	67,64	68,69	67,35	70,27

Tabuľka 4.1: Priemerná úspešnosť klasifikátorov natrénovaných na vstupe o danej dĺžke

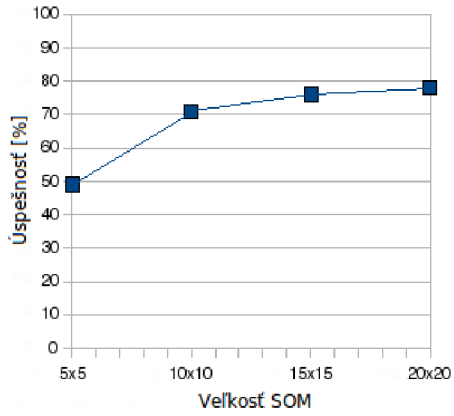
Z tabuľky 4.1 vyplýva, že jedine klasifikátor s pôvodným rozlíšením 28×28 dosahuje priemernej úspešnosti nad 70 %. Zaujímavosťou je, že klasifikátory 24×24 a 18×18 dosahujú o niečo lepšie výsledky ako okolité klasifikátory.

4.2 Rozpoznávanie pomocou SOM

Neurónové siete nie sú žiadnou výnimkou a aj tu je množstvo parametrov, ktoré môžu ovplyvniť výslednú úspešnosť rozpoznávania ručne písaných číslic. V prípade voľby SOM je tento počet výrazne nižší, keďže nie je nutné rozhodovať o tom, koľko vrstiev neurónovej siete je potrebné. Jediným parametrom tak zostáva veľkosť samo-organizujúcej sa mapy. Keďže sa zistilo, že kvôli veľkej vstupnej dátovej sade dochádza k rýchlemu pretrénovaniu, tak bola sada rozdelená na menšie skupiny. Počet prvkov jednotlivých skupín je druhý testovaný parameter, ktorý ovplyvňuje úspešnosť rozpoznávania SOM.

4.2.1 Veľkosť SOM

Základný parameter, ktorý ovplyvňuje rýchlosť rozpoznávania a aj samotnú úspešnosť je veľkosť Kohonenovej mapy. Testovať som sa rozhodol 4 veľkosti SOM, ktorých rozostup bol zakaždým +5 výška a +5 šírka.

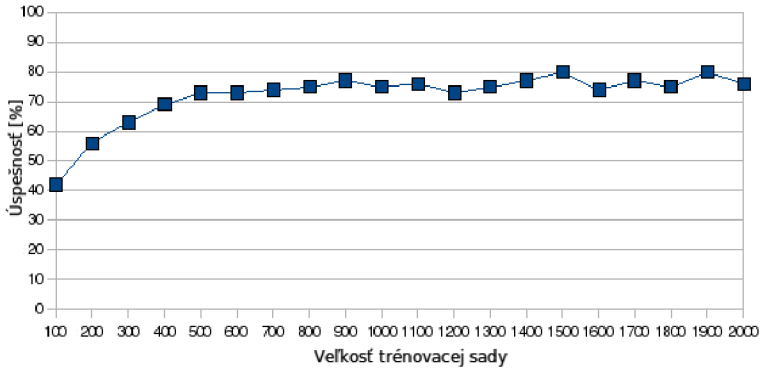


Obrázok 4.10: Úspešnosť samo-organizujúcich sa máp v závislosti na veľkosti SOM.

Na grafe 4.10 je vidieť úspešnosť klasifikátora, ktorý sa trénoval na 1 000 prvkovej vstupnej množine. Ako minimálna veľkosť bola použitá neurónová sieť s celkovým počtom 25 neurónov. Keďže sieť 5×5 nedosahuje úspešnosti ani 50 % je na rozpoznávanie nevhodná a je nutné sa zamerať na väčšie siete. Pri veľkosti 10×10 sa úspešnosť rozpoznávanie zvýšila oproti SOM 5×5 skoro o polovicu. Pri väčších sieťach už úspešnosť nestúpa tak výrazne, ale naopak rapídne narastá čas, ktorý je potrebný na rozpoznanie.

4.2.2 Veľkosť trénovacej množiny

Vzhľadom na fakt, že trénovanie je viacprechodový proces je nutné zladiť faktory trénovania s veľkosťou trénovacej množiny.



Obrázok 4.11: Úspešnosť neurónového klasifikátora v závislosti na veľkosti trénovacej sady.

Pri testovaní veľkosti trénovacej množiny na úspešnosť rozpoznávania bol zvolený štvorprechodový cyklus. V prvom kole bol nastavený parameter okolia víťazného neurónu na

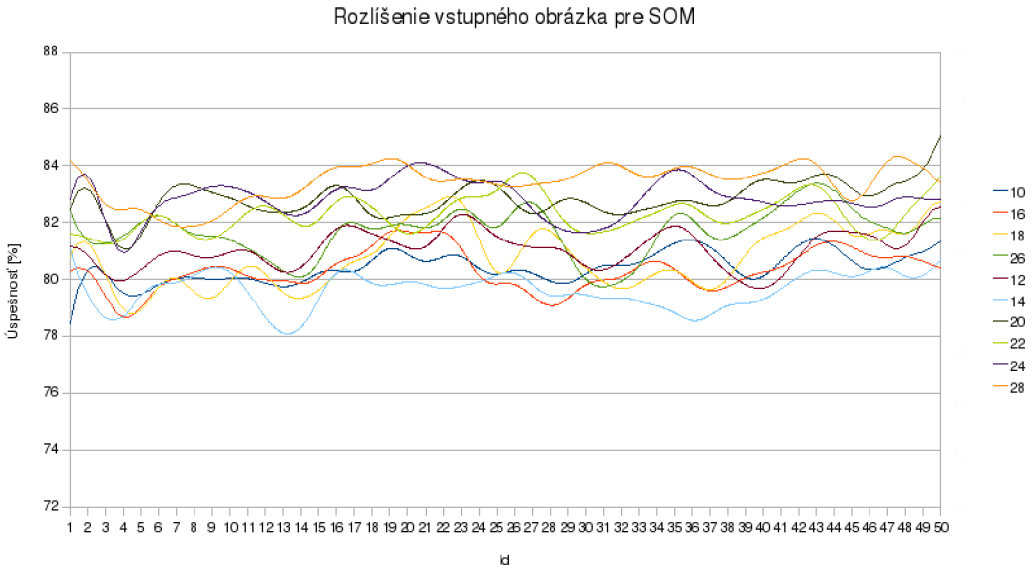
hodnotu 3 a v každom prechode sa postupne znižoval o 1. Takisto sa znižovala aj učiaca konštanta, ktorá bola v prvom kole nastavená na hodnotu 0,3. Znižovanie učiacej konštanty bolo vykonávané po kroku 0,1, pričom v poslednom kole sa hodnota nemenila.

Na základe predchádzajúceho testovania bola zvolená veľkosť neurónovej siete 15×15. Táto veľkosť bola zvolená z dôvodu dobrých výsledkov a výrazne rýchlejšieho vyhodnocovania ako sieť s veľkosťou 20×20.

Výsledok testovania je zaznamenaný v grafe 4.11. Z testovania vyplynulo, že pre trénovanie klasifikátora je rozumné rozdeliť trénovaciu sadu na časti, ktoré budú obsahovať aspoň 500 vzoriek. Pri hodnotách väčších ako je 1 300 neurónová sieť už začína zle generalizovať vstup a výsledky sú rozkmitané.

4.2.3 Rozlíšenie vstupného obrázka

Podobne ako v prípade rozpoznávania pomocou stromových klasifikátorov, tak aj v prípade samo-organizujúcich sa máp bol vykonaný experiment s nájdením optimálneho rozlíšenia vstupu. Keďže metóda pracuje na globálnej úrovni dalo sa očakávať, že výsledky rozpoznávania nebudú tak výrazne klesať ako v prvom prípade.



Obrázok 4.12: Úspešnosť samo-organizujúcich sa máp v závislosti na veľkosti trénovacej sady.

Na grafe 4.12 je znázornená úspešnosť klasifikátorov natrénovaných na rôznych veľkostiach vstupnej sady. Je vidieť, že väčšina výsledkov sa pohybuje medzi 79 % a 84 % a úspešnosť jednotlivých klasifikátorov je vyrovnaná. Najhoršie dopadli klasifikátory s rozlíšením 14×14, ktoré boli najčastejšie najmenej úspešné. Najlepší výsledok dosiahol klasifikátor 20×20, ktorý ako jediný prekonal hranicu 85 %.

dĺžka strany [px]	10	12	14	16	18	20	22	24	26	28
úspešnosť [%]	80,47	81,06	79,71	80,33	82	82,9	82,34	82,87	82,44	83,44

Tabuľka 4.2: Priemerná úspešnosť klasifikátorov natrénovaných na vstupe o danej dĺžke

Viac ako úspešnosť jednotlivých klasifikátorov je dôležitejšia priemerná úspešnosť klasifikátorov natrénovaných rovnakým spôsobom. Hodnoty priemernej úspešnosti sú zobrazené v tabuľke 4.2. Z tabuľky je vidieť, že najlepšiu úspešnosť dosahuje klasifikátor s rozlíšením 28×28 a nie klasifikátor 20×20. Ďalej sa dá vyčítať, že úspešnosť rozpoznávania so znižovaním rozlíšenia nepatrne klesá.

4.3 Fúzia klasifikátorov

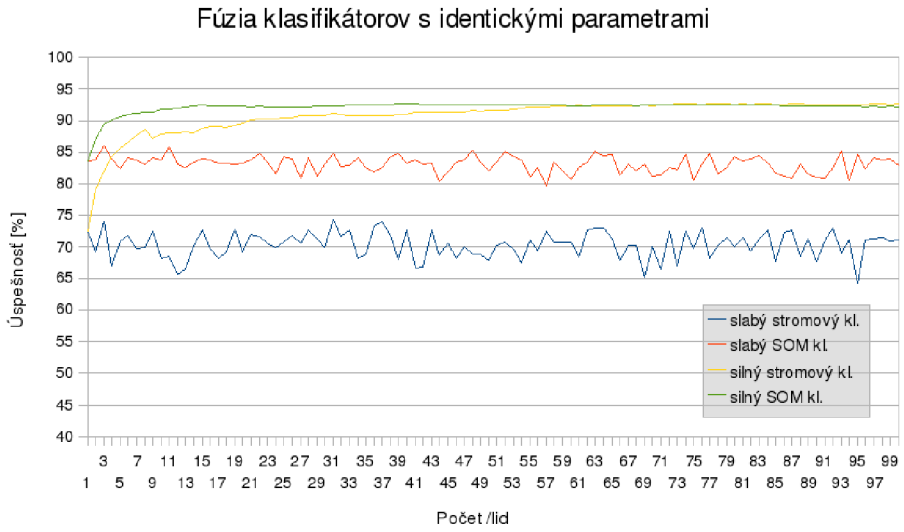
Táto časť práce obsahuje experimenty na zistenie výhod a nevýhod fúzovacích metód pri využití systémov založených na rozdielnych princípoch a rozdielnych rozlíšeniach vstupných vzoriek. Ako bolo už spomínané, jedna vzorka klasifikátora sa bude nazývať *slabý klasifikátor* a fúzia skupiny klasifikátorov bude označovaná ako *silný klasifikátor*. Najskôr bude popísaná fúzia jednotlivých klasifikátorov natrénovaných s identickými parametrami. Výsledky a dôvody pre využitie tejto metódy zvyšovania úspešnosti rozpoznávania sú popísané v časti 4.3.1. Časť 4.3.2 opisuje fúziu klasifikátorov naučených rozdielnymi metódami, ktoré využívajú rozdielne príznaky. Rozdielne príznaky by mali obsahovať rozličné informácie o číslici a tým dodávať do rozhodovacieho procesu väčšie množstvo informácií ako by sa uvažovalo, ak by sa rozhodovalo na základe výstupov klasifikátorov len s jedným typom príznakov. V ďalšej časti 4.3.3 je popísaná fúzia klasifikátorov, ktoré boli natrénované s rovnakými parametrami, ale ich vstupy boli podvzorkované. Prioritou je overenie hypotézy z článku [3], že klasifikátory natrénované na podvzorkovaných obrázkoch môžu pridať dodatočnú informáciu o klasifikovaných vzoroch.

Na záver sekcie venovanej testovaniu fúzií klasifikátorov bude spomenutá fúzia všetkých spôsobov. To znamená, že sa budú kombinovať výsledky viacerých klasifikátorov natrénovaných viacerými učiacimi metódami, pričom niektoré klasifikátory budú natrénované na podvzorkovaných vstupoch.

4.3.1 Fúzia klasifikátorov s identickými parametrami

Fúzia klasifikátorov s identickými parametrami požaduje, aby algoritmus, ktorý bol použitý na vytváranie klasifikátorov, produkoval rozdielne inštancie klasifikátorov. Testované boli dva typy rozpoznávacích systémov: klasifikátory založené na rozpoznávacích stromoch a klasifikátory založené na samo-organizujúcich sa mapách. Systém založený na stromových klasifikátoroch generuje rozdielne klasifikátory hlavne z dôvodu nepreskúmavania všetkých možných pravidiel pri zostavovaní stromu a výbere len určitého počtu tagov, kde ich pozície sú volené náhodne. SOM klasifikátor využíva na vytváranie rozdielnych klasifikátorov trénovaciu sadu rozdelenú na menšie časti a náhodnú inicializáciu váh jednotlivých neurónov. Tretí typ rozpoznávacieho systému (založený na metóde AdaBoost) nebol na testovanie fúzie vhodný, lebo podstata AdaBoost metódy je konštruovanie silného klasifikátora zo slabých (v tomto prípade multi-block local binary pattern) a samotné zostavovanie klasifikátora sa vykonáva podľa deterministických pravidiel.

Graf na obrázku 4.13 znázorňuje ako pridávanie slabých klasifikátorov má pozitívny vplyv na úspešnosť rozpoznávacieho systému. Na získanie výsledkov fúzie bola použitá metóda voľby väčšiny. Je vidieť, že úspešnosť jedného stromového klasifikátora sa pohybuje medzi 66.2 % a 75.5 %. Fúzia klasifikátorov tohto typu do jedného silného klasifikátora dokáže zvýšiť úspešnosť rozpoznávania až na 92 %. Počet stromových klasfikátorov pre fúziu by sa mal pohybovať okolo 60 kusov.



Obrázok 4.13: Fúzia klasifikátorov s identickými parametrami.

Podobné výsledky je možné dosiahnuť aj v prípade fúzie klasifikátorov založených na samo-organizujúcich sa mapách, kde minimálna úspešnosť jedného klasifikátora je 74.7 % a maximálna úspešnosť je 83.1 %. Silný SOM klasifikátor dosahuje úspešnosti až 91,4 %. Na testovanie sa použili slabé klasifikátory, ktorých veľkosť bola 10×10. Trénovacia sada bola rozdelená na časti po 1 000 vzorkov.

Z výsledkov znázornených v grafe 4.13 sa dá povedať, že vhodným počtom SOM klasifikátorov je 15. Následné zvyšovanie počtu klasifikátorov má len minimálny vplyv na úspešnosť. Takéto zvýšenie ale vedie k väčšej časovej náročnosti, keďže daný vstup sa musí vyhodnotiť pre každý slabý klasifikátor. Samotný výsledok dosahuje úspešnosti 91,4 %. V článku [2] bola použitá úspešnosť 90 % na porovnávanie výsledkov, takže v prípade SOM je úspešnosť o niečo vyššia. Porovnanie úspešnosti medzi jednotlivými typmi neurónových sietí je dobre popísané v [21], kde sa úspešnosť jednotlivých typov pohybuje od 88 % do 98 %.

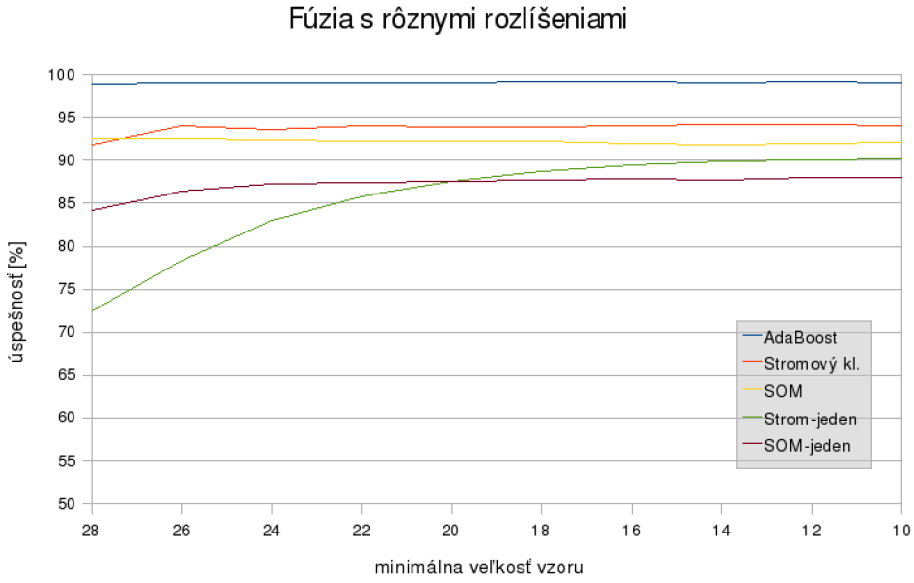
Keďže v oboch typoch silných klasifikátoroch sa úspešnosť priblížila hranici 92 % bol vykonaný experiment, ktorý mal overiť, či výsledky nie sú ovplyvnené binarizáciou vstupu, ktorá bola SOM a stromovych klasifikátoroch uskutočnená. Experiment spočíval v natrénovaní AdaBoost klasifikátora na vyprahovaných dátach. AdaBoost klasifikátor dosiahol úspešnosti 98,69 %. Preto sa dá povedať, že binarizácia vstupu nespôsobila zastavenie zvyšovania úspešnosti na spomínanej hranici a približne rovnaká úspešnosť silných klasifikátorov je náhodná.

4.3.2 Fúzia klasifikátorov s podvzorkovaním vstupných vzorov

V prípade, že učiaci algoritmus rozpoznávacieho systému nedokáže generovať rôzne inštancie klasifikátorov je možné klasifikátory vytvárať pomocou iných vstupov. Ak má trénovacia sada limitujúci počet vzoriek, kvôli ktorému je rozdelenie sady na časti nevhodné, tak riešením môže byť zníženie rozlíšenia položkám dátovej sady, čím sa vytvorí nová trénovacia sada o rovnakom počte vzoriek. Využitie klasifikátorov natrénovaných na podvzorkovaných obrázkoch je princípom fúzie popísanej v tejto časti. Jedinou podmienkou, ktorá zaručuje zvyšovanie úspešnosti je, aby algoritmus pracoval lokálne. Ak je požadovaná podmienka

splnená, podvzorkovanie obrázkov spôsobí rozšírenie lokálnych oblastí vzoru a pri následnej fúzii sa do rozhodovania dodá ďalšia informácia.

Podvzorkovanie trénovacích dát pravdepodobne spôsobí zníženie úspešnosti rozpoznávania pre jeden klasifikátor. Použitím fúzie klasifikátorov s podvzorkovaním vstupných vzorov môže v konečnom dôsledku výsledok rozpoznávania zlepšiť.



Obrázok 4.14: Fúzia klasifikátorov s podvzorkovaním vstupov. "Strom-jeden" a "SOM-jeden" je fúzia klasifikátorov, kde je zastúpený z každého rozlíšenia jeden klasifikátor, zatiaľ čo v prípade "Stromový kl." a "SOM" je fúzia konštruovaná z viacerých klasifikátorov rôzneho rozlíšenia tak, aby bol zachovaný počet slabých klasifikátorov.

Na grafe na obrázku 4.14 sú znázornené výsledky dosiahnuté spriemerovaním logaritmických pravdepodobnostných hodnôt z jednotlivých klasifikátorov. Učiace metódy AdaBoost a SOM pracujú viac-menej na globálnej úrovni a tak efekt fúzie nie je tak výrazný. V prípade SOM s konštantným počtom slabých klasifikátorov (SOM krivka) fúzia nemá skoro žiadny vplyv na úspešnosť rozpoznávania, lebo silný klasifikátor obsahuje dostatočné množstvo slabých klasifikátorov a ako bolo spomínané, nebola ani splnená podmienka o lokálnosti metódy. U krivky SOM-jeden, kde sa počet slabých klasifikátorov zvyšuje je vidieť ako sa zvyšuje, aj úspešnosť rozpoznávania. V tomto prípade je ukázané, že v prípade nemožnosti vytvárania rôznorodých klasifikátorov je možné využiť fúziu s podvzorkovaním na zlepšenie rozpoznávacieho systému.

U stromových klasifikátorov ("Stromový kl." a "Strom-jeden") je splnená podmienka, ktorá zaručuje zvýšenie úspešnosti rozpoznávania (lokálne pracujúce klasifikátory). V porovnaní so SOM klasifikátormi, krivka znázorňujúca úspešnosť rastie v oboch prípadoch, keďže sú dodané nové informácie o vzore.

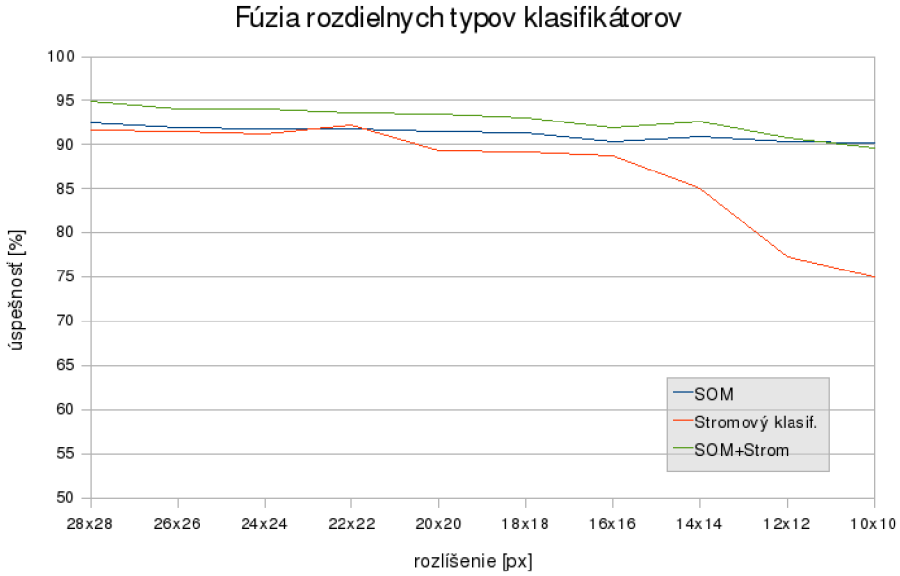
AdaBoost metóda s rozlíšením vstupných vzorov 28×28 pixelov dosahovala 98.87 % úspešnosť. Použitím fúzie s podvzorkovanými vstupmi sa úspešnosť zvýšila nad hranicu 99 % (presné hodnoty sú zobrazené v tabuľke 4.3). Hoci sa krivka na grafe 4.14 zdá plochá, fúzia dokázala eliminovať viac ako 16 % zle klasifikovaných vzorov. Fúzia s podvzorkovanými vstupmi sa preto javí ako relatívne jednoduchá metóda zvýšenia úspešnosti už aj tak výkonných klasifikátorov.

dĺžka strany [px]	28	26	24	22	20	18	16	14	12	10
úspešnosť	9 887	9 898	9 894	9 896	9 901	9 905	9 907	9 904	9 905	9 903

Tabuľka 4.3: Počet úspešne rozpoznaných vzoriek pomocou fúzie klasifikátorov natrénovaných metódou AdaBoost s rozdielnou minimálnou veľkosťou (fúzované bolo po jednom klasifikátore z každého rozlíšenia od 28×28 do minimálnej veľkosti, odstupňované po dvoch pixeloch). Počet vzoriek na rozpoznávanie bolo 10 000 kusov.

4.3.3 Fúzia klasifikátorov natrénovanými rôznymi systémami

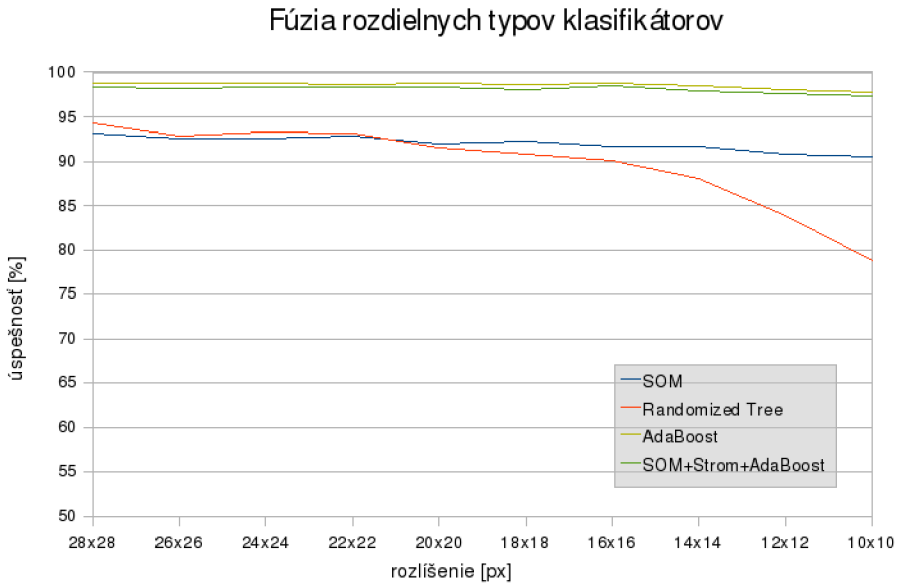
Fúzia rozdielnych klasifikátorov využíva výsledky klasifikátorov pracujúcich s rozdielnymi príznakmi, aby mohla vyhodnotiť vzor presnejšie. Základnou myšlienkou je opäť dodať do rozhodovania ďalší druh informácie. V prípade fúzie rôznych systémov je informácia úplne iného charakteru. Charakter dodanej informácie závisí od použitých príznakov rozpoznávacími systémami.



Obrázok 4.15: Fúzia klasifikátorov natrénovaných rozličnými metódami (SOM a stromové klasifikátory) s rovnakým rozlíšením vstupných vzoriek.

Na grafe 4.15 je zobrazená fúzia SOM klasifikátorov a stromových klasifikátorov pomocou spriemerovania logaritmických pravdepodobnostných hodnôt. Je vidieť, že vyhodnotenie vzorov pre jednotlivé podvzorkované trénovacie databázy je úspešnejšie ako pre samostatné silné klasifikátory (okrem fúzie pre rozlíšenie 10×10, kde úspešnosť SOM systému je vyššia lebo pridaný stromový systém má výrazne horšie výsledky a tak negatívne ovplyvnil výsledky). Z hodnôt pre rozlíšenie 10×10 sa dá usúdiť, že fúzia systémov s výrazne rozdielnymi výsledkami úspešnosti spôsobí degradovanie lepšieho systému. Príčinou je zanesenie nepresností horším systémom. Na grafe 4.15 je vidieť, že táto degradácia nie je výrazná oproti lepšiemu systému.

Degradácia systému je zreteľná na grafe 4.16, ktorý znázorňuje fúziu všetkých troch rozpoznávacích systémov. Pre dosiahnutie najlepších výsledkov sú na grafe znázornené výsledky fúzie klasifikátorov pomocou lineárnej logistickej regresie. Úspešnosť fúzie je horšia (98.4 %), ako najlepší dosiahnutý výsledok jedného systému (AdaBoost, 98.87 %) a to

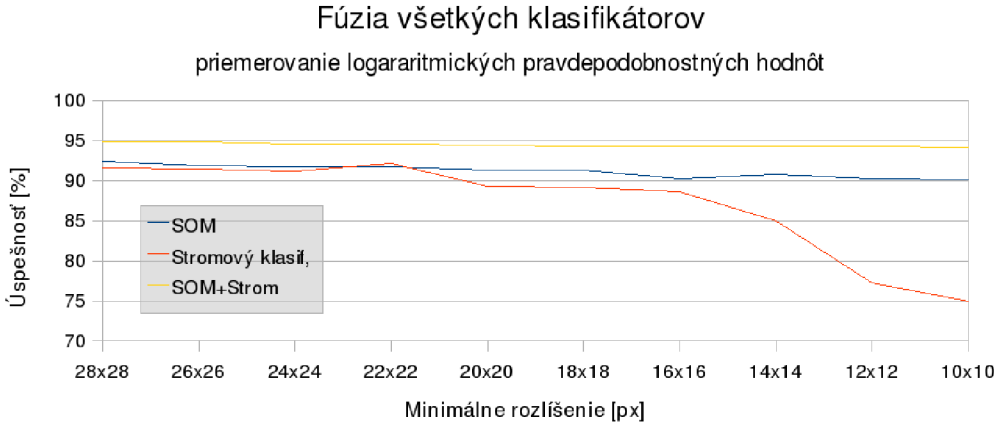


Obrázok 4.16: Fúzia klasifikátorov netrénovaných rôznymi metódami (SOM, stromové a AdaBoost klasifikátory) s rovnakým rozlíšením vstupných vzoriek.

pre všetky fúzovacie metódy. Toto je opäť spôsobené fúzovaním klasifikátorov s výrazne odlišnou úspešnosťou rozoznávania.

4.3.4 Fúzia všetkých klasifikátorov

Posledný skúšaný spôsob zlepšenia rozpoznávania bola fúzia výsledkov všetkých typov klasifikátorov. Fúzia sa vykonávala pomocou lineárnej logistickej regresie na SOM, stromových klasifikátoroch a AdaBoost klasifikátoroch. Jednotlivé typy obsahovali varianty podvzorkovaných vstupov od veľkosti 28×28 po veľkosť 10×10 zmenšované po dvoch pixeloch z každej strany. Počet SOM a stromových klasifikátorov pre rovnaké rozlíšenie bolo 50 kusov. Pri metóde AdaBoost bol klasifikátor len jeden pre jedno rozlíšenie. Celkovo bolo teda použitých 500 SOM klasifikátorov, 500 stromových klasifikátorov a 10 AdaBoost klasifikátorov. Výsledná fúzia dosiahla úspešnosť 97.6 %.



Obrázok 4.17: Fúzia SOM a stromových klasifikátorov s rôznym rozlíšením vstupov.

Keďže všetky medzisystémové fúzie zhoršovali výsledky AdaBoost klasifikátorov, tak som sa rozhodol otestovať úspešnosť fúzií len na stromových klasifikátoroch a SOM. Výsledky fúzie pomocou priemerovania logaritmických pravdepodobnostných hodnôt sú znázornené na grafe 4.17. Počet stromových a SOM klasifikátorov bol rovnaký ako v predchádzajúcej časti t.j. celkovo 500 kusov pre jeden typ.

Graf úspešnosti fúzie 4.17 sa podobá na fúziu rozdielnych typov (4.15). Rozdiel je, že v prípade fúzie rozdielnych typov je každá hodnota úspešnosti získaná zo 100 výsledkov klasifikátorov a u fúzie všetkých klasifikátorov sa ku klasifikátorom pridávajú tie s menším rozlíšením vstupov a tak sa ich celkový počet zvyšuje. Keďže na grafe je krivka fúzie pomerne rovná, výsledky sú zobrazené aj v tabuľke 4.4.

dĺžka strany [px]	28	26	24	22	20	18	16	14	12	10
úspešnosť [%]	94,91	94,83	94,62	94,65	94,52	94,34	94,36	94,38	94,29	94,2

Tabuľka 4.4: Percentuálny podiel úspešne rozpoznaných vzoriek pomocou fúzie klasifikátorov natrénovaných metódou stromových klasifikátorov a SOM s rozdielnou minimálnou veľkosťou (fúzované bolo po jednom klasifikátore z každého rozlíšenia od 28×28 do minimálnej veľkosti, odstupňované po dvoch pixeloch).

Z tabuľky je vidieť, že hodnoty úspešnosti mierne klesajú. Dá sa teda povedať, že fúzia viacerých spôsobov nie je vhodná v prípade priemerovania logaritmických pravdepodobnostých hodnôt. Jednou z príčin môže byť počet klasifikátorov, kde pridávaním ďalších sa neuplatní relevantná informácia, ale zanesie sa do rozhodovania "šum".

Kapitola 5

Konferencia ACIVS 2010

Zo základov diplomovej práce bol vypracovaný článok, ktorý bol odoslaný 1.mája na konferenciu ACIVS 2010. Obsah článku je uvedený v prílohe B.

ACIVS (Advanced Concepts for Intelligent Vision Systems) je konferencia zameraná na techniky pre vytváranie adaptívnych, intelignentých, bezpečných a bezpečnostných systémov spracovávajúcich obrazové dáta. Hlavnými témami tohto ročníka sú:

- Kamerové systémy, vrátane multikamerových systémov,
- Spracovanie obrazu a videa (lineárna a nelineárna filtrácia, segmentácia, multirozlíšenia, Markovské techniky, spracovanie farieb, modelovanie, analýza, interpolačné a priestorové transformácie, pohyb, fraktály a multifraktály),
- Analýza vzorov (analýza tvaru, dátová a obrazová fúzia, zhodovanie vzorov, neurónové siete, počítačové učenie) and záchrana dát z obrazu na základe obsahu,
- Diaľkové snímanie (techniky pre filtráciu, kompresiu, zobrazovanie a analýzu optických, infračervených, radarových, multi- a hyperspektrálnych leteckých a satelitných snímkov),
- Kódovanie a prenos statického obrazu a videa (kódovanie obrazu/videa, hybridné kódovanie, meranie kvality, ochrana obrazu a videa, databázy obrazu a videa, vyhľadávanie v obrazoch a triedenie, indexácia videa, multimediálne aplikácie),
- Architektúra a vyhodnocovanie výkonu (implementácia algoritmov, GPU implementácia, testovanie záťaže, kritéria hodnotenia).

Články z konferencie budú publikované konzorciom Springer Verlag v Lecture Notes in Computer Science. Zároveň bude vydaná aj knižná podoba. Elektronická verzia sa bude nachádzať na stránkach http://www.springerlink.com.

Výsledky budú známe do 30. júla a tak v súčasnosti výsledky o prijatí resp. neprijatí článku nepoznám.

Kapitola 6

Záver

Cieľom, ktorý bol v zadaní vytýčený bolo vytvoriť rozpoznávanie založené na stromových klasifikátoroch. Systém bol rozdelený do troch častí. Fáza vytvorenia rozhodovacieho stromu trvá výrazne dlhšie ako zvyšné časti. Úspešnosť jedného rozpoznávania sa pohybuje na hranici 70 %.

Následne bolo zadanie rozšírené o vytvorenie systémov rozpoznávania založených na iných princípoch a natrénovaných inými učiacimi metódami. Z týchto systémov bol mnou implementovaný iba jeden. Ten je založený na rozpoznávaní pomocou neurónových sietí, konkrétne samo-organizujúcich sa máp. Systém ako príznaky využíval priamo hodnoty pixelov vstupného vzoru a úspešnosť SOM klasifikátora sa pohybovala na hranici 80 %.

Tretí systém rozpoznávania ručne písaných číslic, ktorý je spomínaný v mojej práci, bol vytvorený učiacou metódou AdaBoost. Systém využíva ako príznaky Multi-Block Local Binary Patterns. Úspešnosť AdaBoost systému pri rozlíšení vstupných obrázkov 28×28 bola 98,82 %.

Ďalšia časť práce obsahuje snahu o zvýšenie úspešnosti jednotlivých systémov. Ako prvá možnosť bola u stromových klasifikátoroch a SOM klasifikátoroch fúzia na základe viacnásobného vyhodnocovania vzoru pomocou klasifikátorov vytvorených rovnakým postupom (keďže proces tvorby klasifikátorov nie je deterministický). U oboch systémoch sa podarilo zvýšenie úspešnosti na hranicu 92,5 %.

Ďalším spôsobom zvýšenia úspešnosti bolo využitie podvzorkovaných vstupov obdobne ako v práci L.A. Alexandreho [3]. U SOM klasifikátoroch sa úspešnosť v podstate nezmenila. U stromových klasifikátoroch sa opäť zvýšila hodnota úspešnosti až na úroveň 94,4 % čo bolo u tohto systému maximum. U metódy AdaBoost bola maximálna úspešnosť 99,1 %, ktorá sa stala aj maximom pre celú prácu.

Na záver bola odskúšaná fúzia rôznych rozpoznávacích systémov kde kombinácia SOM a stromových klasifikátorov dosiahla úspešnosť 94,9 %. Po pridaní AdaBoost klasifikátorov sa úspešnosť zvýšila na 98,4 %, ale samotná metóda AdaBoost dosahovala vyššiu úspešnosť (98.82 %).

Pomocou rôznych experimentov sa ukázalo, že metódam výrazne pomôže, ak sa k vyhodnocovaniu pridajú aj iné informácie. Najvýhodnejšie sa ukázalo využiť fúziu klasifikátorov natrénovaných na rôznom rozlíšení vstupných dát a nevyužívať len klasifikátory natrénované na jednom optimálnom rozlíšení. Takýto spôsob zvyšovania úspešnosti fungoval na všetkých metódach aj keď v určitých prípadoch nebola zmena taká výrazná.

Medzisystémová fúzia sa ukázala ako nevýhodná, lebo vytvorené systémy dosahujú výrazne odlišné úspešnosti. Ak sú však kombinované systémy s porovnateľnou úspešnosťou, je aj medzisystémová fúzia rozumný spôsob, ako znížiť chybovosť pri rozpoznávaní. V prí-

pade zaradenia metódy AdaBoost, ktorá výrazne prekonávala ostatné dva systémy, to bola prvá možnosť. Jednou z príčin môže byť využitie binarizácie vstupu, pri ktorej mohlo dôjsť k strate dôležitej informácie. V ďalšej práci by mohli byť zostavené klasifikátory, ktoré využívajú šedotónový vstup, čo je možné tak ako v prípade SOM, tak aj v prípade stromových klasifikátoroch (postup je spomenutý v [12]).

Po vyskúšaní rôznych variant fúzii sa pre MNIST databázu ukázalo ako najlepšie riešenie využiť fúziu iba s podvzorkovanými vstupmi pre metódu AdaBoost. Na fúziu sa ukázalo využiť len rozlíšenia 16×16, 18×18, 20×20, 22×22, 24×24, 26×26, a 28×28. Aby bola množina variant rozumne malá do úvahy sa brali len párne násobky, pričom bol zachovávaný pomer strán. V tomto prípade úspešnosť dosiahla 99,1 %.

Výsledky všetkých systémov sú primerané k času vzniku odborných článkov, z ktorých som vychádzal. V prípade fúzií je úspešnosť jednotlivých systémov ešte lepšia a dosahuje tak vyšších hodnôt, ako boli v článkoch spomínané. V porovnaní so súčasnými metódami, ale nedosahujú takých kvalitných výsledkov. Najúspešnejší súčasný rozpoznávací systém vytvoril Daniel Keysers [14]. Princípom je kombinácia štyroch state-of-the-art techník a dosahovaná chybovosť je 0,35 %.

Literatúra

[1] Ahonen, T.; Hadid, A.; Pietikainen, M.: Face recognition with local binary patterns. *Proceedings of the European Conference on Computer Vision*, 2004: s. 469–481.

[2] Al-Omari, S. A. K.; Sumari, P.; Al-Taweel, S. A.; aj.: Digital Recognition using Neural Network. *Journal of Computer Science*, 2009: s. 427–434.

[3] Alexandre, L. A.: Gender recognition: a multiscale decision fusion approach. *Pattern Recognition Letters*, 2010, doi:10.1016/j.patrec.2010.02.010.

[4] Appiah, K.; Hunter, A.; Meng, H.; aj.: A binary self-organizing map and its FPGA implementation. *IEEE International Joint Conference on Neural Networks*, 2009.

[5] Brummer, N.: FoCal Multi-class: Toolkit for Evaluation, Fusion and Calibration of Multi-class Recognition Scores. Manual to SW Package, June 2007, http://niko.brummer.googlepages.com/focalmulticlass.
URL http://niko.brummer.googlepages.com/focalmulticlass

[6] Carter, M. P.: Boosting a Simple Weak Learner For Classifying Handwritten Digits. Technická Zpráva PCS-TR98-341, Dartmouth College, 1998.

[7] Dobrovolný, M.: Rozpoznání ručne psaných číslic. Bakalárska práca, Vysoké učení technické v Brne, Fakulta informačných technológií, 2009, ved. Adam Herout.

[8] Ebrahimpour, R.; Hamedi, S.: Hand Written Digit Recognition by Multiple Classifier Fusion based on Decision Templates Approach. *The International Conference on Computer, Electrical, and Systems Science, and Engineering*, 2009: s. 245–250.

[9] Freund, Y.; Schapire, R. E.: A decision-theoretic generalization of on-line learning and an application to boosting. In *EuroCOLT '95: Proceedings of the Second European Conference on Computational Learning Theory*, London, UK: Springer-Verlag, 1995, ISBN 3-540-59119-2, s. 23–37.

[10] Freund, Y.; Schapire, R. E.: Experiments with a New Boosting Algorithm. 1996.

[11] FREUND, Y.; SCHAPIRE, R. E.: A short introduction to boosting. *J. Japan. Soc. for Artif. Intel.*, ročník 14, č. 5, september 1999: s. 771–780.

[12] Geman, D.; Amit, Y.; Wilder, K.: Joint Induction of Shape Features and Tree Classifiers. *IEEE Trans. PAMI*, ročník 19, 1997.

[13] Kant, S.; Sharma, V.; Dass, B. K.: On recognition of cipher bit stream from different sources using majority voting fusion rule. *Scientific Analysis Group*, 2006: s. 90–111.

[14] Keysers, D.: Comparison and Combination of State-of-the-art Techniques for Handwritten Character Recognition: Topping the MNIST Benchmark. *CoRR*, ročník abs/0710.2231, 2007.

[15] Kittler, J.: A Framework for Classifier Fusion: Is It Still Needed? In *Proceedings of the Joint IAPR International Workshops on Advances in Pattern Recognition*, London, UK: Springer-Verlag, 2000, ISBN 3-540-67946-4, s. 45–56.

[16] Kohonen, T.: The self-organizing map. *Neurocomputing*, ročník 21, 1998: s. 1–6, ISSN 0925-2312.

[17] Kussul, E.; Baidyk, T.: Improved Method of Handwritten Digit Recognition.

[18] Labusch, K.; Barth, E.; Martinetz, T.: Simple method for high-performance digit recognition based on sparse coding. *Neural Networks, IEEE Transactions*, ročník 19, 2008: s. 1985–1989, ISSN 1045-9227.

[19] Lam, L.; Suen, C.: Application of Majority Voting to Pattern Recognition: An Analysis of Its Behavior and Performance. *SMC*, ročník 27, č. 5, 1997: s. 553–568.

[20] Lauer, F.; Suen, C. Y.; Bloch, G.: A trainable feature extractor for handwritten digit recognition.

[21] LeCun, Y.; Bottou, L.; Bengio, Y.; aj.: Gradient-Based Learning Applied to Document Recognition. 1998, s. 2278–2324.

[22] LeCun, Y.; Cortes, C.: THE MNIST DATABASE of handwritten digits. 2007, [Online; cit. 2009-12-21].
URL http://yann.lecun.com/exdb/mnist/

[23] Liao, S.; Zhu, X.; Lei, Z.; aj.: Learning Multi-scale Block Local Binary Patterns for Face Recognition. In *Proceedings of ICB 2007, LNCS*, LNCS, Springer-Verlag, 2007, s. 282–837.

[24] Minka, T. P.: A comparison of numerical optimizers for logistic regression. 2003, uRL http://research.microsoft.com/ minka/papers/logreg/.

[25] Ojala, T.; Pietikäinen, M.; Mäenpää, T.: Multiresolution Gray-Scale and Rotation Invariant Texture Classification with Local Binary Patterns. *IEEE Trans. Pattern Anal. Mach. Intell.*, ročník 24, č. 7, 2002: s. 971–987, ISSN 0162-8828, doi:http://dx.doi.org/10.1109/TPAMI.2002.1017623.

[26] Ruta, D.; Gabrys, B.: An Overview of Classifier Fusion Methods. *Computing and Information Systems*, ročník 7, 2000: s. 1–10.

[27] Schapire, R. E.; Singer, Y.: Improved Boosting Algorithms Using Confidence-rated Predictions. *Mach. Learn.*, ročník 37, 1999: s. 297–336.

[28] Seewald, A. K.: Digits - A Dataset for Handwritten Digit Recognition. 2005.

[29] Vaquero, D. A.: Handwritten Digit Images. 2005.

[30] ŠOCHMAN, J.; MATAS, J.: WaldBoost - Learning for Time Constrained Sequential Detection. IEEE, jún 2005, ISBN 0-7695-2372-2, s. 150–156.

Dodatok A

Obsah CD

- klasifikator
  - som - priečinok obsahuje niekoľko SOM klasifikátorov.
  - tree - priečinok obsahuje stromové klasifikátory
- plagat - plagát prezentujúci diplomovú prácu
- prezentacia - prezentácia
- sprava - technická správa a zdrojové súbory v LaTeXu
- src
  - dataset - datová sada MNIST v rôznych rozlíšeniach. Trénovacia časť rozdelená v pomere 9:1 na trénovaciu sadu a validačnú.
  - fuzia - program na vyhodnocovanie výsledkov klasifikátorov pomocou priemerovania
  - konvertor - program na vytvorenie podvzorkovanej dátovej sady MNIST
  - som - zdrojové súbory pre trénovanie a testovanie samoorganizujúcich sa máp
  - tree - zdrojové súbory pre trénovanie a testovanie stromových klasifikátorov
- vysledky
  - adaboost - priečinok obsahuje odozvy AdaBoost klasifikátov
  - som - priečinok obsahuje odozvy SOM
  - tree - priečinok obsahuje odozvy stromových klasifikátorov

Dodatok B

Článok určený na konferenciu ACIVS 2010

Handwritten Digits Recognition Improved by Multiresolution Classifier Fusion

Miroslav Štrba and Adam Herout

Graph@FIT
Brno University of Technology, Faculty of Information Technology
Brno, Czech Republic
herout@fit.vutbr.cz
http://www.fit.vutbr.cz/research/groups/graph/

Abstract. Recognition of handwritten digits is a well-studied problem of multiclass pattern recognition. One common approach to construction of highly accurate classifiers is fusion of several weaker classifiers into a compound one, which (when meeting some constraints) outperforms all the individual fused classifiers. This paper studies the possibility of fusion classifiers of different kinds (Self-Organizing Maps, Randomized Trees, and AdaBoost with MB-LBP weak hypotheses) constructed on training sets resampled to different resolutions. While it is common to select one resolution of the input samples as the "ideal one" and fuse classifiers constructed for it, this paper shows that the accuracy of classification can be improved by fusing information from several scales.

Key words: Digit Recognition, Classifier Fusion, Multiresolution, SOM, Randomized Trees, AdaBoost

1 Introduction

Recognition of handwritten digits is a popular basic task of computer vision and machine learning. This can be explained both by its application potential (automatic processing of various forms etc.) and by its properties from the point of view of machine learning. In contrast to recognition of letters and especially continuous text, in the case of digits, the number of classes is reasonably low and no further syntactic/semantic analysis helps recognition of separate samples, so recognition of handwritten digits is a good (however rather simple) benchmarking task in the field of pattern recognition and computer vision.

Fusion of classifiers is a popular approach to improving performance of a machine learning system [1], [2]. Typically, different classification principles or classifiers using different feature extractors are used (e.g. [3]). Recently, Luís A. Alexandre explored the possibility to improve classification performance on the task of gender recognition [4]. A common approach to face detection, recognition, and related tasks is to identify an "ideal" image resolution, resample the images to it and use it for processing. Alexandre points out that although one of

the resolutions can be identified as the best performing, the classification performance can be improved by using several different resolutions. The gain by fusion of classifiers of different resolution sampling seems to be even higher than the gain of fusion classifiers using different feature sets (histograms of oriented gradients and local binary patterns were used). The best (by far) result is obtained by fusing different feature extractors and different resolutions at the same time. Note that the fusion principle was the simplest possible: decision by majority of classifiers. This paper reports a set of experiments using multiple resolutions to improve recognition of handwritten digits.

The text is organized as follows: Section 2 summarizes methods of handwritten digits recognition based on neural networks / self-organizing maps, randomized trees (forests), and recognition based on adaptive boosting. These classification principles were used for the experimental evaluation of the benefit of multiresolution classifier fusion. Section 3 revises three fusion methods which are commonly used in machine learning and which were used in the evaluation. Section 4 gives the experimental results and their discussion and the paper is concluded with Section 5.

2 Handwritten Digits Recognition Background

The problem of handwritten digits recognition is popular [5], [6], [7] and therefore more databases of samples exist ([8], [9], [10], ...), which are used for training and testing the recognition systems. The most popular and most frequently used database is the MNIST database of handwritten digits [8]. This database was constructed by mixing two NIST (National Institute of Standards and Technology) databases. The database includes two datasets: the training dataset comprises 60 000 patterns and the testing dataset 10 000 patterns, 28×28 pixels in greyscale for each sample. The second file contains a label for each sample at corresponding location within the first file.

This paper evaluates several recognition systems based on different methods. Each method should be based on a different classification principle and use different features to describe the pattern, in order to have the fused system describe the pattern in a more complex way. The chosen methods are:

Self-Organizing Maps – a type of neural network using all pixels in the sample (see Section 2.1 for details),

Randomized Trees – using two small neighbourhood descriptors and direction between them as the decision criteria (see Section 2.2), and

AdaBoost – using multi-block local binary patterns as the image feature (Section 2.3).

The evaluations use the MNIST database for both training and testing. In the case of the randomized tree method, the database is binarized by thresholding in the same manner as in [11]. The threshold was set to value 128. Binarization was also used in the case of self-organizing maps.

2.1 Self-Organizing Maps

Self-Organizing Maps (SOM) [12] were chosen as a method based on neural networks, where the arrangement of neurons is defined exactly and it does not have any hidden layer. Another reason for this decision was the fact that the winning neuron of SOM affects the surrounding neurons and thus the net is learning the input set of patterns faster because neurons directly form clusters for the classes to be recognized. Usage of SOM to recognition of handwritten digits on the MNIST database was based on the work of Appiah et al. [13].

The essence of the training process is iterative: a new neural network is created based on the previous one, with better recognition ability. During training, each neuron adapts its weight and collects information about the count of digits for which it is winning over the other neurons. The winning neuron denotes such a neuron whose weights best represent the input sample. Each learned neuron represents the digit type which became most common input during training. Experiments show that the best SOM size for MNIST database is 10×10. This constitutes a compromise between accuracy and time/spatial complexity. The training set was divided into smaller parts (~ 1000 samples) because the SOM are overtraining quickly for a huge set of samples. This dividing, together with random weight initialization for each neuron in the net, cause variability required by fusion.

The recognition process is simpler than training: every sample from the testing dataset is applied as the input for each neuron in the map. The neuron with the best response is proclaimed as winning and the pattern is resolved using neuron's histogram of classes of digits.

2.2 Randomized Trees

The algorithm [11, 14] is based on finding a relationship between areas of interest that would be unique for each class of digits. A potential area of interest is a neighbourhood of pixels (e.g. 4×4) which contains both values possible in the binarized image. These areas appear on the edges and provide information about the shape of digits. Each neighbourhood is classified into a certain class called a *tag*. The relationship of two particular tags and their mutual direction (out of 8 possible directions: north, north-east, east, south-east, ...) is called a rule. The decision tree is constructed with decision rules in each non-leaf node.

Training can be divided into two main phases: the first phase is finding important areas; the second phase is classification of these areas. The classification of neighbourhoods is carried out by a binary decision tree. This tree is called the *tag tree*. The criterion at each non-leaf node of the tag tree is the value of one pixel whose position is stored in the node. The tag tree is illustrated in Figure 1; below the tree are examples of areas that belong to the classes (*tags*).

The last phase of training is assembling the *decision tree* using the tags assigned by the tag tree. The decision tree uses rules for sorting patterns into the subtrees and leafs. Since it is impossible to investigate all rules during training for all nodes, only a subset of possible rules is selected for training each node

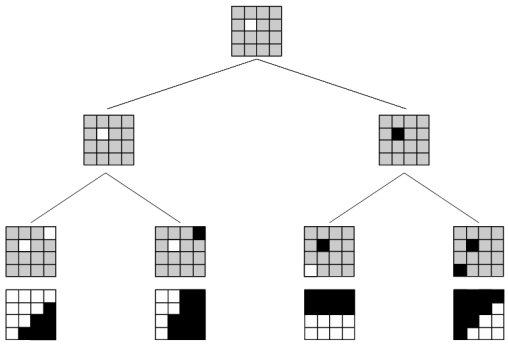


Fig. 1. Tag tree with depth 2.

randomly. This randomness determines the variability among the trees trained on the same training set – similarly to the SOM, this is important for later fusion.

The process of recognition is much less time-consuming compared to the training process. The samples of the test set are binarized, potential areas of interest are classified by the *tag tree* and the set of obtained tags is further classified by the decision tree. The leaf of the decision tree contains probabilities of digit classes, computed from statistics gathered during the training process.

2.3 Adaptive Boosting

Adaptive boosting [15, 16] is a machine learning algorithm, which can significantly reduce the error of another learning algorithm which gives better results than random results. Its input is a set *weak classifiers* for learning and it uses a set of previously wrongly classified objects to select and train new classifiers. The result is the robust *strong classifier*. Adaptive boosting was used for recognition of hanwritten digits by Carter [17].

The root of adaptation to wrongly classified object is the weighting mechanism: in the case that object is wrongly classified, the weights of classifiers are (optimally) updated to obtain a corrected decision in the next training round. Thanks to this, the algorithm is sensitive to all objects, which means that it is trying to classify each object into the right set. This property is not good for generalization, but the fact that AdaBoost is resistant to overtraining (large number of elements in the training set bring better accuracy than in the case of a classifier trained on a smaller training set) is eliminating this disadvantage.

The recognition method used in our system is similar to a method previously used for face recognition [18]. In this article, the Multi-Block Local Binary Pattern (MB-LBP) were defined and used instead of the original Local Binary Patterns (LBP) [19] that has been proved to be effective for face recognition by

Ahonen et al. [20]. MB-LBP has several advantages: they are more robust than LBP and they encode both the micro and macro structure.

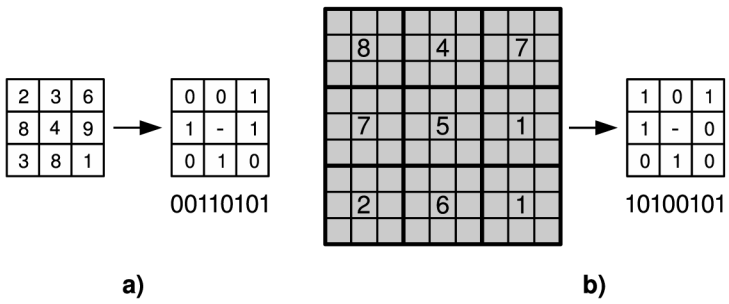


Fig. 2. a) LBP thresholding area and conversion to binary value. b) MB-LBP thresholding area and conversion to binary value.

This area usually has the size of 3×3 pixels and the pixel of interest is in the middle. After the surrounding pixels are thresholded by the value of the middle pixel, the distribution of values is converted to a binary value. The process of thresholding and binariation is illustrated by Figure 2a. In MB-LBP, the value of single pixel is replaced by the average of gray-level values of a sub-region. Each sub-region is a square area containing neighbouring pixels. In the (common) case that the sub-region size is 3×3, MB-LBP size is 9×9 pixels.

3 Multiresolution Classifier Fusion

Classifier fusion – particularly in the case of recognition of handwritten digits [21] – can generate more accurate classification then each of the basic classifiers. Various algorithms for classifier fusion exist [1, 2] – from simple (Voting by Majority, Secion 3.1 or Averaging, Section 3.2) to more sophisticated ones (Linear Logistic Regression, Section 3.3).

Use of fusion is only reasonable if the learning algorithm is not deterministic and therefore it can produce different classifiers or using at least two different classification algorithms. One learning algorithm can produce different classifiers due to different algorithm parameters or using randomness or various inputs for training. Random number generators are often used for initialization of iterative training procedures or in the recognition system based on randomized trees in selection of the best rule, because the evaluation metrics is time-consuming and/or it is impossible to evaluate all patterns and select the best one. Various inputs of the training process may be achieved by division of the training set to smaller parts, by a mechanism of bootstrapping the training set, or by rescaling input images to different scales. It is also possible to use identical learning

algorithm but with different feature sets extracted from the input samples [4]; fusing classifiers operating on different feature sets can observe the samples in a more complex ways.

3.1 Majority Voting

Majority voting is one of the simplest method of classifier fusion [22, 23]. In decision by majority, each classifier k outputs a decision $a_{k,i}(x)$ about the given pattern x for each class i. Value $\delta_{k,i}(x)$ in Equation (1) identifies the class with the best response (in the case of handwritten digits recognition, ten possible classes i exist):

$$\delta_{k,i}(x) = \begin{cases} 1 \text{ if } a_{k,i}(x) = \max\limits_{i=0}^{9} a_{k,i}(x) \\ 0 \text{ otherwise} \end{cases} \tag{1}$$

Each classifier votes for one class and all classifiers share equal weight 1. Equation (2) describes the function for majority voting, which uses these representations of classes to evaluate the ensemble response by all N classifiers.

$$M_i(x) = \sum_{k=0}^{N} \delta_{k,i}(x) \tag{2}$$

The pattern is recognized as belonging to the class which received the most votes. If the maximal value is achieved for two or more classes, the decision is made by randomly selecting one of them.

3.2 Averaging Log Likelihood Ratio

Averaging *log likelihood ratio* is a fast method, which considers all values for different classes and not only the best selected class from each classifier. The classifier responses $a_{k,i}(x)$ are the log likelihood ratio; however, some classification principles output directly the probability of each class (randomized trees, SOM). The conversion to LLR can be done based on Equation (3):

$$p_{k,i}(x) = \frac{1}{1 - e^{a_{k,i}(x)}} \tag{3}$$

Fusion of classifiers using averaging log likelihood ratio can be expressed as

$$M_i(x) = \sum_{k=0}^{N} a_{k,i}(x); \tag{4}$$

note that since the maximal value of $M_i(x)$ is looked for, the right side of Equation 4 is not divided by the number of classifiers. The final decision is found as the class i with maximal response of $M_i(x)$.

3.3 Fusion by Linear Logistic Regression

As in the previous Section 3.2, the log likelihood ratio vector

$$\boldsymbol{l}_k(x) = (a_{k,0}(x), a_{k,1}(x), \ldots, a_{k,9}(x)) \tag{5}$$

is obtained from the individual classifiers and its values are fused into one log likelihood ratio used for the final decision $\boldsymbol{l}'$ [24]:

$$\boldsymbol{l}'(x) = \sum_{k=1}^{N} \alpha_k \boldsymbol{l}_k(x) + \beta. \tag{6}$$

The fusion coefficients are found as

$$(\alpha_1, \alpha_2, \ldots, \alpha_N, \beta) = \arg\,\max C'_{llr}, \tag{7}$$

where C'_{llr} is calculated for the fused $\boldsymbol{l}'()$ over a supervised training database. For this purpose, the training set is split into a part used for training the individual classifiers and a part used for training the fusion by linear logistic regression [25].

4 Experimental Results

This section reports the experiments carried out to find out the benefits of fusion of different classification principles and different resolutions of the input samples. One instance of any classifier is referred to as a *weak classifier* and a fused group of weak classifiers is denoted as a *strong classifier*. Firstly, fusion of randomized classifiers with identical training parameters was tested (Section 4.1). Fusion improves the accuracy by eliminating classifiers which make a wrong decision based on specific properties of the pattern and/or overtraining. Then, the fusion of classifiers trained using different learning methods was tested (Section 4.2) – the fusion collects information for different feature types and thus it can recognize patterns more accurately. Finally, the multiscale fusion for different classification methods was tested (Section 4.3) – in this case we intended to verify the hypothesis suggested by [4] that classifiers trained on subsampled images can add further information about the classified patterns.

4.1 Fusion of Classifiers of Identical Parameters

Fusion of classifiers of identical parameters requires algorithm which produces different instances classifiers. We tested two recognition systems: classifiers based on randomized trees and classifiers based on self-organizing maps.

Figure 3 shows how the addition of further weak classifiers has positive effect to the accuracy on the recognition system. The accuracy of one randomized tree classifier is between 66.2 % and 75.5 %. Fusion of these classifiers into one increases recognition accuracy to 92 %. In this case, majority voting was used for fusion.

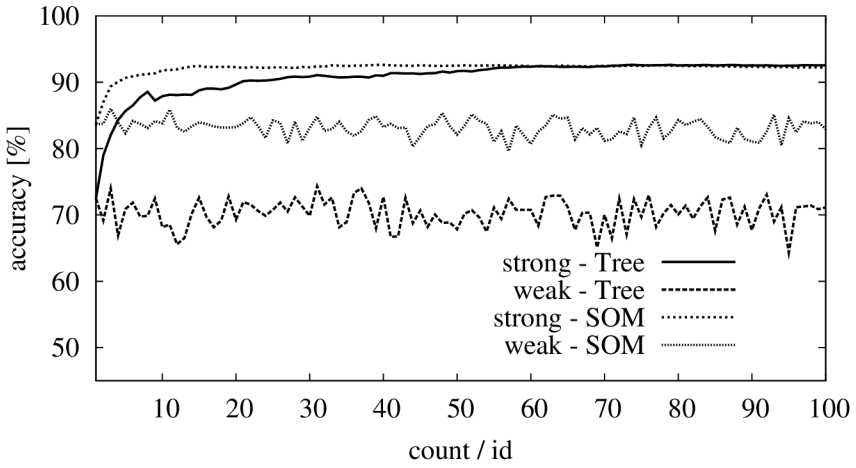


Fig. 3. Fusion of randomized tree classifiers.

Similar results are achieved for fusion of SOM classifiers: one SOM classifier has a 74.7 % accuracy minimum and maximum accuracy was 83.1 %; the strong SOM classifier reached the maximum accuracy of 91,4 %.

AdaBoost was not evaluated in this way because its base idea is building a strong classifier from weak ones (MB-LBP in this case) already, and for a given dataset of reasonable size (as is the case of the MNIST dataset), the classifier construction is deterministic.

4.2 Fusion of Different Classifiers

Fusion of different classifiers uses classifiers with different features to evaluate pattern precisely. Figure 4 reports the fusion of the SOM and randomized tree classifiers. Fusion accuracy is higher than the accuracy of the separate classifiers except for 10×10 resolution, where the accuracy of the randomized tree classifiers is significantly worse and the fused classifier is influenced by them. Fusion of systems with significantly different accuracy results in degradation of the successful system by the "noise" introduced by the worse one.

Degradation of the system by fusion is shown clearly in Figure 5, which shows results of fusion of three systems. Fusion accuracy is worse (98.4 %) than the best performing system (AdaBoost, 98.82 %) for all fusion methods tested. This, again, is caused by fusing classifiers with significantly different accuracy.

4.3 Multiscale Fusion for Different Classification Methods

This fusion method is expected to be useful if the learning algorithm is working locally. If the system fulfils this condition, the fusion will be successful because subsampling the pattern widens the local area. In such a case, the output classifier for a system with subsampled input will by less accurate than for full

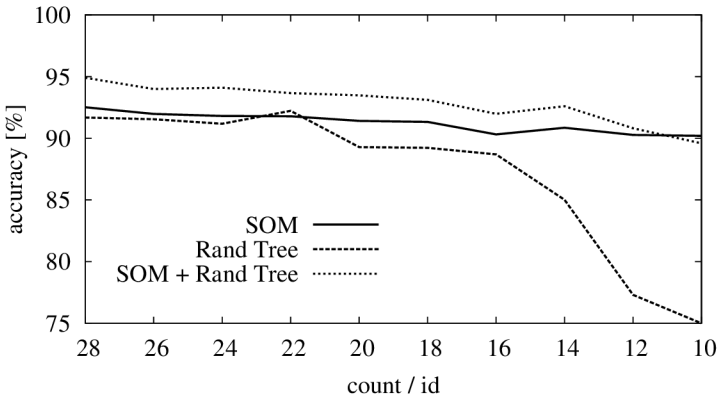


Fig. 4. Fusion for different methods (SOM and Randomized Trees) with the same input resolution.

resolution, but using multiscale fusion, the final result will be improved. If the learning algorithm is deterministic and therefore produces only one classifier for one given training set, the multiscale fusion could increase the accuracy even if the learning algorithm operates globally over the sample area.

Figure 6 shows the gain achieved by this mode of fusion. AdaBoost and SOM learning methods are working (more-or-less) globally, so the gain is not so clear. SOM with a constant number of weak classifier ("SOM" curve) does not have any effect on the recognition accuracy because there was enough classifiers working globally, but using only one classifier ("SOM – One") for each resolution caused accuracy increase. On the randomized tree recognition system ("Rand Tree"), which has inverse properties (many classifiers work in local area) compared to the SOM, fusion worked well. The area of interest was enlarged and the system could incorporate new information.

patterns count	28	26	24	22	20	18	16	14	12	10
10 000	9 887	9 898	9 894	9 896	9 901	9 905	9 907	9 904	9 905	9 903

Table 1. Fusion of AdaBoost classifiers with different minimal size (all sizes from the minimal one to 28 are fused, one classifier per resolution). The original sample size is 28×28px, i.e. the images were downsampled. Reported are the numbers of correctly recognized samples.

The AdaBoost method with input pattern resolution 28×28 has accuracy 98.8 %. Using multiscale fusion, the accuracy was increased over 99 % (refer, please, to Table 1 for exact numbers). Though the curve in the graph appears flat, the accuracy improvement means ellimination of over 16 % of wrongly classified

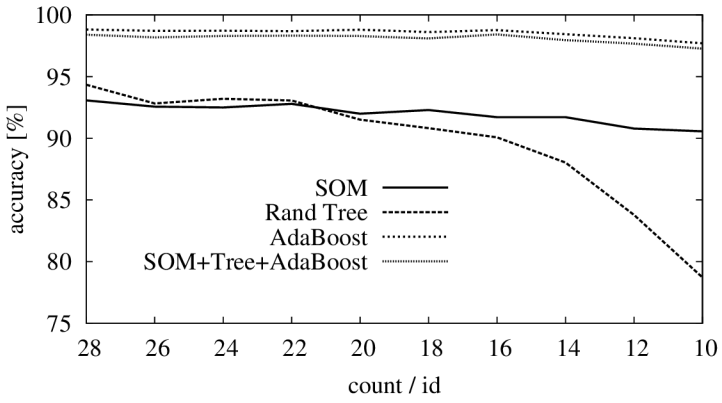


Fig. 5. Fusion for different methods with the same input resolution.

samples. Multiresolution fusion therefore appears as a relatively simple method of improving accuracy of an already well-performing classifier.

5 Conclusions

The purpose of the research described in this paper is to consider and evaluate the possibility of fusing classifiers trained on different image resolutions to obtain a superior performance. The experiments carried out show that fusing different scales helps the accuracy of classification; this means that the commonly used approach – to select one best resolution and use it – is not optimal. However, the gain of this method is not as high as in the case of gender recognition reported by Alexandre [4].

The classification principles evaluated in the mentioned experiments were Randomized Trees, Self-Organizing Maps and AdaBoost with MB-LBP image features. AdaBoost outperformed the other classification techniques by far; among other possible explanations, the other methods use binarized image and thus drop an important portion of information content. For future experiments we plan to involve other classification methods and use the multiresolution and multi-classifier approach to construct a compound classifier outperforming the state-of-the-art solutions.

Acknowledgements

This work was partially supported by the BUT FIT grant FIT-10-S-2 and the research plan MSM0021630528.

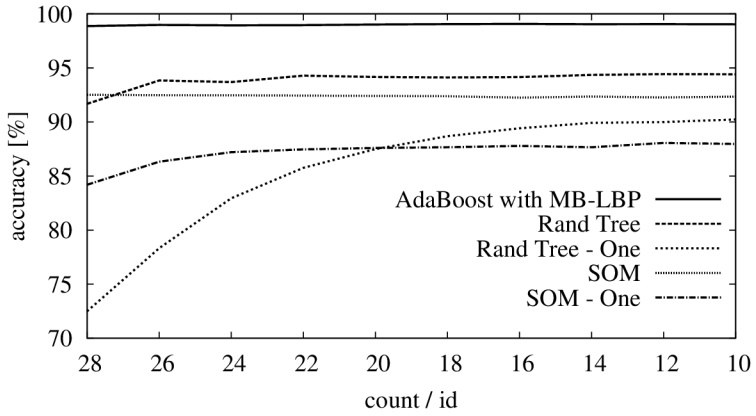


Fig. 6. Multiscale fusion for different methods. "Rand Tree - One" and "SOM - One" stand for fusion of single instances of the classifier on a given resolution, while "Rand Tree" and "SOM" are compound classifiers constructed from a number of weak ones for each resolution.

References

1. Ruta, D., Gabrys, B.: An overview of classifier fusion methods. Computing and Information Systems **7** (2000) 1–10
2. Kittler, J.: A framework for classifier fusion: Is it still needed? In: Proceedings of the Joint IAPR International Workshops on Advances in Pattern Recognition, London, UK, Springer-Verlag (2000) 45–56
3. Chang, S.F., He, J., Jiang, Y.G., Khoury, E.E., Ngo, C.W., Yanagawa, A., Zavesky, E.: Columbia university/VIREO-CityU/IRIT TRECVID2008 high-level feature extraction and interactive video search. In: NIST TRECVID Workshop. (2008)
4. Alexandre, L.A.: Gender recognition: a multiscale decision fusion approach. Pattern Recognition Letters (2010) doi:10.1016/j.patrec.2010.02.010.
5. Labusch, K., Barth, E., Martinetz, T.: Simple method for high-performance digit recognition based on sparse coding. Neural Networks, IEEE Transactions **19** (2008) 1985–1989
6. Lauer, F., Suen, C.Y., Bloch, G.: A trainable feature extractor for handwritten digit recognition. Pattern Recognition **40**(6) (June 2007) 1816–1824 doi:10.1016/j.patcog.2006.10.011.
7. Kussul, E.M., Baidyk, T.: Improved method of handwritten digit recognition tested on mnist database. Image Vision Comput. **22**(12) (2004) 971–981
8. LeCun, Y., Cortes, C.: The mnist database of handwritten digits (2007) [Online; cit. 2009-12-21] http://yann.lecun.com/exdb/mnist/.
9. Seewald, A.K.: Digits - a dataset for handwritten digit recognition. Technical Report TR-2005-27, Austrian Research Institut for Artificial Intelligence (2005)
10. Vaquero, D.A., Barrera, J., Jr., R.H.: A maximum-likelihood approach for multiresolution W-operator design. Brazilian Symposium on Computer Graphics and Image Processing **0** (2005) 71–78
11. Geman, D., Amit, Y., Wilder, K.: Joint induction of shape features and tree classifiers. IEEE Trans. PAMI **19** (1997)

12. Kohonen, T.: The self-organizing map. Neurocomputing **21** (1998) 1–6
13. Appiah, K., Hunter, A., Meng, H., Yue, S., Hobden, M., Priestley, N., Hobden, P., Pettit, C.: A binary self-organizing map and its FPGA implementation. IEEE International Joint Conference on Neural Networks (2009)
14. Breiman, L., Schapire, E.: Random forests. In: Machine Learning. (2001) 5–32
15. Schapire, R.E., Singer, Y.: Improved boosting algorithms using confidence-rated predictions. Mach. Learn. **37** (1999) 297–336
16. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. In: EuroCOLT '95: Proceedings of the Second European Conference on Computational Learning Theory, London, UK, Springer-Verlag (1995) 23–37
17. Carter, M.P.: Boosting a simple weak learner for classifying handwritten digits. Technical Report PCS-TR98-341, Dartmouth College (1998)
18. Liao, S., Zhu, X., Lei, Z., Zhang, L., Li, S.Z.: Learning multi-scale block local binary patterns for face recognition. In: Proceedings of ICB 2007, LNCS. LNCS, Springer-Verlag (2007) 282–837
19. Ojala, T., Pietikäinen, M., Mäenpää, T.: Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. IEEE Trans. Pattern Anal. Mach. Intell. **24**(7) (2002) 971–987
20. Ahonen, T., Hadid, A., Pietikainen, M.: Face recognition with local binary patterns. Proceedings of the European Conference on Computer Vision (2004) 469–481
21. Ebrahimpour, R., Hamedi, S.: Hand written digit recognition by multiple classifier fusion based on decision templates approach. The International Conference on Computer, Electrical, and Systems Science, and Engineering (2009) 245–250
22. Lam, L., Suen, C.: Application of majority voting to pattern recognition: An analysis of its behavior and performance. SMC **27**(5) (1997) 553–568
23. Kant, S., Sharma, V., Dass, B.K.: On recognition of cipher bit stream from different sources using majority voting fusion rule. Scientific Analysis Group (2006) 90–111
24. Brummer, N.: Focal multi-class: Toolkit for evaluation, fusion and calibration of multi-class recognition scores. Manual to SW Package (June 2007) http://niko.brummer.googlepages.com/focalmulticlass.
25. Minka, T.P.: A comparison of numerical optimizers for logistic regression. Technical report (2003) URL http://research.microsoft.com/ minka/papers/logreg/.