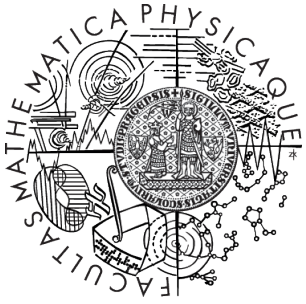
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

DIPLOMOVÁ PRÁCE



Jiří Ondrušek

Neuronové sítě vyšších řádů pro rozpoznávání vzorů

Katedra softwarového inženýrství
Vedoucí diplomové práce: *RNDr. Jana Štanclová, Ph.D.*
Studijní program: *Teoretická informatika, Neprocedurální programování a umělá inteligence*

Chtěl bych poděkovat RNDr. Janě Štanclové, Ph.D. ze vedení mé diplomové práce, rady a připomínky, které mi pomohly práci dokončit.

Dále bych chtěl poděkovat své přítelkyni a rodině za jejich podporu.

Prohlašuji, že jsem svou diplomovou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce.

V Praze dne 14.12.2007

Jiří Ondrušek

Obsah

Název práce: Neuronové sítě vyšších řádů pro rozpoznávání vzorů
Autor: Jiří Ondrušek
Katedra: Katedra softwarového inženýrství
Vedoucí diplomové práce: RNDr. Jana Štanclová, Ph.D.
E-mail vedoucího: Jana.Stanclova@mff.cuni.cz
Abstrakt:

Diplomová práce se zabývá studiem umělých neuronových sítí vyšších řádů pro rozpoznávání vzorů. Cílem práce je ověřit a otestovat modely neuronových sítí vyšších řádů použitelné pro rozpoznávání znaků české abecedy s ohledem na různé deformace (otočení, změnu velikosti, posunutí, ...). Jsou navrženy také modifikace jednotlivých studovaných modelů s cílem zlepšit rozpoznávací vlastnosti modelů. Získané experimentální výsledky jsou shrnuty a vzájemně porovnány. Součástí práce je i zhodnocení získaných výsledků.

Klíčová slova: neuronová síť vyššího řádu, rozpoznávání vzorů, rozpoznávání znaků

Title: High order neural networks for pattern recognition
Author: Jiří Ondrušek
Department: Department of Software Engineering
Supervisor: RNDr. Jana Štanclová, Ph.D.
Supervisor's e-mail address: Jana.Stanclova@mff.cuni.cz
Abstract:

This thesis is concerned with study of high order neural networks for character recognition. The goal of the thesis is to verify and test models of high order neural networks, which are applicable to character recognition with regard to various deformations (rotation, scaling, translation ...). Our modifications of these models are designed to improve classification abilities of the models. Obtained experimental results are summarized and compared with each other. The thesis also includes an evaluation of the results.

Keywords: high order neural network, pattern recognition, character recognition

Kapitola 1

Úvod

Pro člověka přirozená schopnost rozpoznávat zvuky, obličeje, tištěné znaky a mnoho dalších je ve světě počítačů věc složitá. Tato problematika označovaná jako rozpoznávání vzorů (pattern recognition) spadá do oblasti umělé inteligence a strojového učení. Jedná se o mladou a rychle se rozvíjející disciplínu, která nachází praktické využití v řadě odvětví. K typickým problémům rozpoznávání vzorů patří například automatické rozpoznávání řeči, rozpoznávání psaného textu, klasifikace textu (filtrování spamu) nebo identifikace lidské tváře.

Stejně jako existuje řada problémů řešitelných pomocí rozpoznávání vzorů, existuje také mnoho technik a způsobů aplikovatelných na tuto problematiku - například statistické metody, neuronové sítě, techniky strojového učení, bayesovské sítě a další [2]. Jednotlivé techniky a jejich kombinace dosahují v dnešní době relativně vysoké úspěšnosti, zatím však nesrovnatelně nižší než člověk sám. Důvodů, proč lidský mozek dosahuje vyšší úspěšnosti, je celá řada. Především stále není dostatečně znám princip fungování lidského mozku. Navíc se jedná o příliš složitý a rozsáhlý systém, jehož počítačová simulace je v dnešní době téměř nemožná. Největší superpočítače stále nedosahují potřebného výkonu pro simulaci celého lidského mozku. Neuronové sítě vzhledem k jejich obecnosti, podobnosti s biologickými systémy a vysoké úspěšnosti patří k nejčastěji užívaným technikám.

1.1 Rozpoznávání znaků

Rozpoznávání znaků je spjato již s počátky rozpoznávání vzorů. Rozpoznávání vzorů je oblast strojového učení, která klasifikuje vzory na základě vybraných vlastností nebo jiných informací získaných se vzoru. Složitější variantou rozpo-

znávání znaků je rozpoznávání znaků, které je invariantní vůči vybraným deformacím (např. posunutí, otočení, změna velikosti). Cílem je rozpoznat znaky, které podstoupily vybrané deformace (případně kombinace deformací), se stejnou úspěšností jako při rozpoznávání nedeformovaných znaků. Techniky umožňující rozpoznávat znaky invariantně vůči některým deformacím nabízejí výrazně širší okruh použití v praxi. Většina úloh pro rozpoznávání znaků v praxi naráží na nutnost vypořádat se s některými deformacemi, které se mohou u rozpoznávaných znaků vyskytovat (např. pootočení naskenovaných znaků při rozpoznávání textu). Na obrázku 1.1 je možné vidět znak „A“ a jeho dvě deformace (vzniklé otočením a změnou velikosti).

Obrázek 1.1: Znak „A“ a jeho deformace

Jednou z možností, jak tuto problematiku řešit, jsou právě biologické systémy. Lidský mozek dokáže rozpoznávat znaky s ohledem na invariance se stoprocentní úspěšností. Tedy lidská schopnost rozpoznávání znaků nám udává stoprocentní měřítko, se kterým srovnáváme jednotlivé implementované techniky. Jednou z technik vycházejících z biologických systémů jsou neuronové sítě. Neuronová síť je výpočetní model založený na biologických systémech. Pro výpočet využívá svoji strukturu, která je tvořena umělými neurony a vazbami mezi nimi.

V této práci se zaměříme na jednu oblast neuronových sítí, tzv. neuronové sítě vyšších řádů, a pokusíme se je aplikovat na řešení problematiky rozpoznávaní znaků s ohledem na invarianci vůči vybraným deformacím.

1.2 Cíle práce

Samotné rozpoznávání znaků (bez ohledu na invarianci) může použitím různých metod dosáhnout relativně vysoké úspěšnosti. Řada metod je však téměř nepoužitelná již při nepatrných deformacích rozpoznávaného znaku. Návrhů řešení problematiky rozpoznávání znaků s ohledem na invarianci vůči deformacím existuje celá řada (např. [8] a [16]). Jejich výsledky ve srovnání s lidským mozkem dosahují nižší úspěšnosti.

Pro náš výzkum budou klíčové tři práce využívající neuronové sítě vyšších řádů [7], [1] a [3]. První dvě práce vycházejí ze základních myšlenek neuronových sítí vyšších řádů. Třetí práce přináší novou datovou strukturu pro neuronové sítě vyšších řádů. Při řešení obecných problémů z oblasti rozpoznávání vzorů dosa-

huje rovněž vysoké úspěšnosti [3], není však speciálně navržena pro rozpoznávání deformovaných znaků.

Jak vyplývá z předešlých odstavců, problém rozpoznávání znaků s ohledem na invariance není stále úspěšně vyřešen. Náš výzkum se zaměří na existující práce zabývající se touto problematikou [7], [1], [3]. Cíle našeho výzkumu jsou následující:

- Práce [7], [1], [3] prostudujeme a experimentálně ověříme při rozpoznávání tištěného písma české abecedy. V experimentech se zaměříme na úspěšnosti jednotlivých modelů při rozpoznávání s ohledem na vybrané invariance (otočení, změnu velikosti, ...). Výsledky všech modelů vzájemně porovnáme a vyhodnotíme.
- Pokusíme se navázat na práce jednotlivých autorů [7], [1], [3] a modely modifikovat s cílem dosažení lepších výsledků. Pokusíme se pomocí vzájemné kombinace modelů a znalostí z oboru neuronových sítí vytvořit modely úspěšnější.
- Implementujeme aplikaci pro testování úspěšností modelů při rozpoznávání znaků. Aplikace bude navržena tak, aby se dala jednoduše rozšiřovat o další modely.

1.3 Struktura práce

Práce bude strukturována následovně:

V kapitole 2 připomeneme základní pojmy potřebné k porozumění neuronových sítí. Představíme algoritmus učení neuronových sítí a některé jeho modifikace, které budou použity při zkoumání jednotlivých modelů [7], [1], [3].

V kapitole 3 popíšeme model neuronové sítě vyššího řádu, jak jej prezentují ve článku [7]. Navrhneme jeho možné úpravy pro zvýšení úspěšnosti rozpoznávání znaků. Srovnáme experimentální výsledky původního modelu i námi navržené modifikace.

V kapitole 4 popíšeme model vyššího řádu prezentovaný v [1]. Navrhneme jeho možné úpravy pro zvýšení úspěšnosti rozpoznávání vzorů. Srovnáme získané výsledky námi modifikované verze s původním modelem.

V kapitole 5 představíme víceúrovňovou hierarchickou síť prezentovanou v článku [3]. Navrhneme její možné úpravy s ohledem na zvýšení úspěšnosti rozpoznávání vzorů. Srovnáme naměřené výsledky původního i námi upraveného modelu.

V kapitole 6 srovnáme všechny zkoumané modely a srovnáme získané výsledky experimentů.

V kapitole 7 zhodnotíme celou práci a navrhneme možné směry pro pokračování výzkumu.

Kapitola 2

Neuronové sítě

Tato kapitola je věnována základním teoretickým pojmům. Popíšeme podstatu neuronových sítí, jejich strukturu a algoritmus učení a vybavování.

2.1 Historie

Neuronové sítě jsou inspirovány principy biologických systémů. Oblast neuronových sítí se snaží modelovat činnost lidského mozku na základě známých faktů. Lidský mozek obsahuje kolem 10^{11} neuronů a každý z nich je spojen přibližně s 10^4 dalšími neurony [11]. K přenosu informací dochází řádově v časech kolem 10^{-3} sekund. Ačkoliv jsme schopni dosáhnout u počítačů rychlosti přenosu řádově kolem 10^{-10} sekundy, je lidský mozek schopen řešit komplexní problémy rychleji. Například rozpoznání obličeje známé osoby netrvá déle než 10^{-1} sekundy [11]. To odpovídá přibližně několika stovkám iterací mezi neurony v lidském mozku. Z toho plyne, že lidský mozek dokáže zpracovávat informace paralelně a při zpracování používá velice složité metody předávání a kódovaní informací. Umělé neuronové sítě se s těmi biologickými v současné době nedají srovnávat. I přes výrazně vyšší rychlost přenosu informací mezi neurony nejsou schopny umělé sítě řešit komplikované problémy ve srovnatelných časech a se srovnatelnými výsledky jako lidský mozek.

Počátek výzkumu v oblasti neuronových sítí se datuje do 2. poloviny 40. let 20. století, kdy Warren McCulloch a Walter Pitts prezentovali jednoduchý model neuronu [9]. V roce 1957 Frank Rosenblatt zobecnil model neuronu na tzv. perceptron [13], který na rozdíl od neuronu McCullocha a Pittse počítal s reálnými čísly. Dále navrhl učící algoritmus, který dokázal nalézt v konečném čase hodnoty vah perceptronu. Na základě těchto prací byla v roce 1957 a 1958 vytvo-

řena první neuronová síť schopná rozpoznávat znaky. Na konci 70. let matematik Marvin Minsky dokázal ve své práci [10], že jednovrstevný perceptron není schopen řešit lineárně neseparabilní problémy, čímž zapříčinil pokles zájmu o oblast neuronových sítí. V roce 1986 David E. Rumelhart a James L. McClelland publikovali práci [15], ve které prezentovali model vrstevnaté neuronové sítě. V témže roce publikovali David E. Rumelhart, Geoffrey E. Hinton a Ronald J. Williams článek [14], ve kterém představili učící algoritmus Back propagation. Jedná se o jednoduchý algoritmus pro učení vrstevnatých neuronových sítí aplikovatelný na širokou škálu problémů. Od té doby zájem o neuronové sítě neustále roste a neuronové sítě se udržují v popředí zájmu.

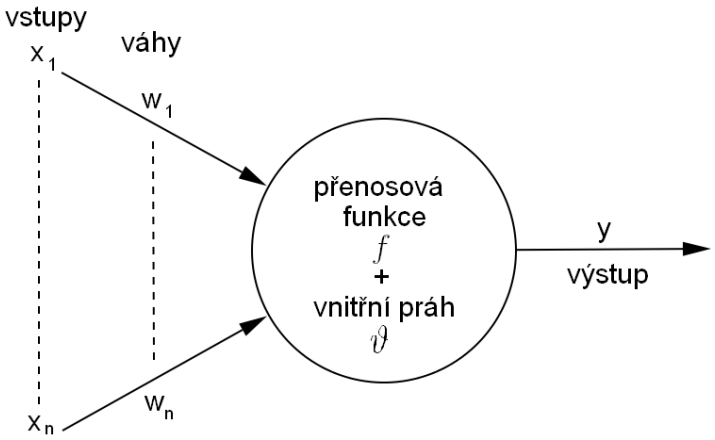
V dnešní době můžeme mluvit o dvou směrech výzkumu v oblasti neuronových sítí. Jeden proud se snaží pomocí neuronových sítí studovat a modelovat biologické sítě. Tento proud úzce spolupracuje s biologií a snaží se aplikovat nejnovější poznatky o biologických sítích do stavby umělých sítí. Druhý proud se snaží o nalezení vysoce efektivního algoritmu učení neruonových sítí s využitím technik strojového učení bez kopírování technik z biologických systémů. Snaží se vylepšit současné algoritmy a najít nové způsoby učení pro různé kategorie řešených problémů.

2.2 Neuron

Základními stavebními prvky neuronových sítí jsou jednotlivé neurony. V biologickém světě jsou jednotlivé neurony tvořeny buňkami. Tyto buňky se skládají z několika částí se specifickou funkčností.

- Tělo (soma) - zzpracování signálu, který přichází do neuronu.
- Dendrity - vlákna, kterými do neuronu přichází signály.
- Axon - jediný výstup neuronu, který je bohatě rozvětvený (do synapsí).
- Synapse - přechod z axonu k dendritu jiného neuronu. Synapse mohou signál zeslabovat či zesilovat.

Základní model (umělého) neuronu vychází z podstaty biologického neuronu. Zjednodušený model (umělého) neuronu je zobrazen na obrázku 2.1. Neuron přijímá od ostatních neuronů signál v podobě vstupního vektoru $\vec{x} = (x_1, \ldots, x_n)$. Příchozí signály mohou být zesíleny či zeslabeny pomocí vnitřních vah neuronu $w_1, \ldots, w_n$. V těle neuronu se pomocí tzv. přenosové funkce f a vnitřního prahu neuronu ϑ spočítá výstup neuronu y. Tento výstup je předán všem navazujícím neuronům.



Obrázek 2.1: Formální model neuronu

Definice 2.2.1. *Neuron s váhovým vektorem* $\vec{w} = (w_1, \ldots, w_n) \in \mathbb{R}^n$*, vnitřním prahem* $\vartheta \in \mathbb{R}$ *a přenosovou funkcí* $f : \mathbb{R}^{n+1} \times \mathbb{R}^n \to \mathbb{R}$ *počítá pro libovolný vstup* $\vec{x} \in \mathbb{R}^n$ *svůj výstup* $y \in \mathbb{R}$ *jako hodnotu přenosové funkce* $y = f[\vec{w}, \vartheta](\vec{x})$*.* $\mathbb{R}$ *značí množinu reálných čísel.*

Existuje celá řada (umělých) neuronů, které se liší použitou přenosovou funkcí. Přenosová funkce ve většině případů pracuje s tzv. potenciálem neuronu.

Definice 2.2.2. *Potenciál neuronu* ξ *je definován jako*

$$\xi = \sum_{i=1}^{n} w_i x_i + \vartheta,$$

kde $\vec{w} = (w_1, \ldots, w_n)$ *je váhový vektor neuronu,* ϑ *je vnitřní práh neuronu a* $\vec{x} = (x_1, \ldots, x_n)$ *je vstupní vektor neuronu.*

Jednou z nejpoužívanějších přenosových funkcí je tzv. skoková přenosová funkce:

$$f[\vec{w}, \vartheta](\vec{x}) = f(\xi) = \begin{cases} 1 & \text{pokud } \xi \geq 0 \\ 0 & \text{jinak.} \end{cases}$$

Tato přenosová funkce určuje, že neuron je aktivní pouze tehdy, pokud jeho vstupní signály přesáhnou zápornou hodnotu prahu. V této práci budeme používat tzv. sigmoidální přenosovou funkci.

Definice 2.2.3. *Sigmoidální funkce* $f[\vec{w}, \vartheta](\vec{x})$ *je definována jako*

$$f[\vec{w}, \vartheta](\vec{x}) = f(\xi) = \frac{1}{1 + e^{-\lambda\xi}},$$

kde λ je strmost funkce, $\vec{w} = (w_1, \ldots, w_n)$ je váhový vektor neuronu, ϑ je vnitřní práh neuronu, $\vec{x} = (x_1, \ldots, x_n)$ a e je Eulerova konstanta.

Tato funkce je často používaná právě v kombinaci s vrstevnatými neuronovými sítěmi a učícím algoritmem Back propagation (viz dále). Sigmoidální přenosová funkce je používána ve spojitosti s algoritmem Back propagation především pro jednoduchost výpočtu hodnoty své vlastní derivace [11].

2.3 Vrstevnaté neuronové sítě

Stejně jako biologické neurony i umělé neurony jsou spojovány do složitějších struktur - neuronových sítí. Teprve neuronové sítě jsou schopny vykonávat složitější operace. Neuronová síť je definována následovně.

Definice 2.3.1. *Neuronová síť M je uspořádaná 6-tice $M = (N, C, I, O, w, t)$, kde*

- *N je konečná neprázdná množina neuronů,*
- *$C \subseteq N \times N$ je neprázdná množina orientovných spojů mezi neurony,*
- *$I \subseteq N$ je neprázdná množina vstupních neuronů,*
- *$O \subseteq N$ je neprázdná množina výstupních neuronů,*
- *$w : C \to \mathbb{R}$ je váhová funkce,*
- *$t : N \to \mathbb{R}$ je prahová funkce.*

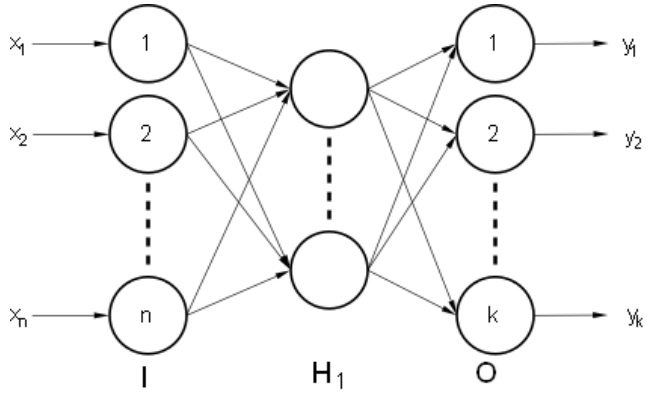
Při našem zkoumání se zaměříme především na vrstevnaté neuronové sítě. Jedná se o neuronové sítě, ve kterých jsou jednotlivé neurony uspořádány do tzv. vrstev. Jednotlivé neurony jsou spojeny pouze s neurony s předcházející a následující vrstvy. Formální definice vrstevnatých sítí je následující.

Definice 2.3.2. *Vrstevnatá neuronová síť s L skrytými vrstvami je neuronová síť tvořená orientovaným acyklickým grafem spojů mezi neurony. Množina neuronů je tvořena posloupností $L + 2$ vzájemně disjunktních podmnožin zvaných vrstvy. Graf spojů obsahuje pouze hrany z i-té do $(i + 1)$-ní vrstvy.*

- *První vrstva zvaná vstupní vrstva je množina všech vstupních neuronů sítě. Tyto neurony nemají v grafu spojů žádné předchůdce. Jejich vstupní hodnota je rovna jejich výstupní hodnotě.*

- *Poslední vrstva zvaná výstupní vrstva je množina všech výstupních neuronů sítě. Tyto neurony nemají v grafu spojů žádné následníky.*
- *Všechny ostatní neurony zvané skryté neurony jsou obsaženy ve zbylých L vrstvách zvaných skryté vrstvy.*

Na obrázku 2.2 je zobrazena jednoduchá vrstevnatá neuronová síť s jednou skrytou vrstvou H_1. Vstupní vrstvě I s n neurony jsou předkládány hodnoty vstupního vektoru $\vec{x} = (x_1, \ldots, x_n)$ tak, že vstupní hodnotou neuronu na pozici i je hodnota x_i. Výstupy neuronů ve výstupní vrstvě O s k neurony vytváří výstupní vektor $\vec{y} = (y_1, \ldots, y_k)$.



Obrázek 2.2: Vrstevnatá neuronová síť

Vzhledem k charakteru našeho výzkumu budou vstupy pro zkoumané neuronové sítě tvořeny maticemi s hodnotami 0 a 1. Tyto matice reprezentují černobílé bitmapy, kde 0 reprezentuje barvu bílou a 1 reprezentuje barvu černou. Matici budeme převádět na vektor tak, že výsledný vektor bude tvořen posloupností jednotlivých řádků matice.

2.4 Učení a vybavování neuronových sítí

V této kapitole se zaměříme na problém, jak naučit vrstevnaté sítě klasifikovat (rozpoznávat) jednotlivé vstupy a přiřazovat jim správnou výstupní hodnotu. Tedy popíšeme proces učení a následné vybavování vrstevnatých neuronových sítí.

Pro učení neuronových sítí existuje mnoho různých přístupů a algoritmů. V našem zkoumání se zaměříme na tzv. *učení s učitelem*. Tento způsob učení je podmíněn existencí tzv. trénovací množiny, kdy ke každému vstupnímu vzoru máme k dispozici také požadovaný výstup.

Definice 2.4.1. *Trénovací množina T je množina p ($p > 0$) uspořádaných dvojic tvaru vstupní vzor/požadovaný výstup:* $T = \{[\vec{x_1}, \vec{d_1}], \ldots, [\vec{x_p}, \vec{d_p}]\}$

Proces učení vrstevnatých neuronových sítí je ve většině případů založený na minimalizaci rozdílů skutečných výstupů sítě ve srovnání s požadovanými výstupy - minimalizaci chyby. V následující kapitole popíšeme učící algoritmus Back propagation, který jsme použili při našem zkoumání.

2.4.1 Back propagation

Jedním z nejrozšířenějších algoritmů pro učení vrstevnatých neuronových sítí je Back propagation nebo-li algoritmus zpětného šíření. Algoritmus zpětného šíření byl publikován v druhé polovině 80 let [14]. Vznikl modifikací algoritmu gradientního hledání lokálního minima [11]. Back propagation je založen na minimalizaci chybové funkce na trénovací množině. Chybová funkce pro Back propagation odpovídá čtvercové odchylce mezi skutečnými výstupy sítě a požadovanými výstupy. Její definice je následující:

Definice 2.4.2. *Chybová funkce E na trénovací množině* $T = \{[\vec{x_1}, \vec{d_1}], \ldots, [\vec{x_p}, \vec{d_p}]\}$ *je definována jako*

$$E = \frac{1}{2} \sum_t \sum_j (y_{j,t} - d_{j,t})^2,$$

kde $y_{j,t}$ je hodnota výstupního neuronu j a $d_{j,t}$ je požadovaná hodnota výstupního neuronu j při předložení t-tého vzoru z T.

Cílem algoritmu Back propagation je najít takové nastavení vah sítě, kde skutečný výstup neuronové sítě bude stejný jako její požadovaný výstup. V tomto případě by síť dosáhla nulové hodnoty chybové funkce.

Průběh algoritmu Back propagation je následující:

1. Inicializace všech vah sítě w_{ij} na náhodné hodnoty.
2. Předložení trénovacího vzoru ve tvaru $[\vec{x}, \vec{d}]$ síti.
3. Výpočet skutečného výstupu sítě: Aktivita neuronu j v každé vrstvě je dána vztahem:
$$y_j = f(\xi_j) = \frac{1}{1 + e^{-\lambda \xi_j}}.$$
Symbol ξ_j se spočte ze vztahu:
$$\xi_j = \sum_i y_i w_{ij}(t),$$

Symbol $w_{ij}(t)$ je váha vedoucí z neuronu i do neuronu j v čase t, čas t odpovídá počtu provedených iterací a λ udává strmost přenosové funkce. Takto vyjádřené výstupy pak tvoří vstup následující vrstvy.

4. Adaptace vah: Při úpravě vah se postupuje směrem od výstupní vrstvy ke vstupní. Váhy se adaptují podle následujícího pravidla:

$$w_{ij}(t+1) = w_{ij}(t) + \alpha\delta_j y_i + \alpha_m(w_{ij}(t) - w_{ij}(t-1)), \tag{2.1}$$

kde α je parametr učení $\alpha \in < 0, 1 >$, α_m značí moment učení $\alpha_m \in < 0, 1 >$, δ_j je chyba neuronu j. Pro výstupní neurony se δ_j spočte ze vztahu:

$$\delta_j = (d_j - y_j)y_j(1 - y_j)\lambda \tag{2.2}$$

Pro skryté vrstvy se δ_j spočte podle:

$$\delta_j = \lambda y_j(1 - y_j)\sum_k \delta_k w_{jk}, \tag{2.3}$$

kde k označuje neurony z následující vrstvy, λ strmost přenosové funkce.

5. Pokud není splněna podmínka ukončení učení, přechod ke kroku 2.

Pro dosažení nízké hodnoty chybové funkce je třeba provést adaptaci vah mnohokrát. V našich experimentech docházelo k adaptacím vah při učení řádově stotisíckrát a více. V definici algoritmu zpětného šíření nebyla definována podmínka ukončení. V našich experimentech jsme v mnoha případech použili nejjednodušší řešení - předem nadefinovaný počet cyklů učícího algoritmu. Jiný způsob je sledovat hodnotu chybové funkce a při její stagnaci algoritmus učení ukončit (tento přístup jsme využili u modelu popsaného v kapitole 5).

2.4.2 Vybavování neuronové sítě

Vybavování neuronové sítě je proces, ve kterém je naučené neuronové síti předložen vzor a ten je sítí klasifikován. Algoritmus vybavování popisují kroky 2 a 3 z algoritmu Back propagation. Síti je předložen vzor $\vec{x}$ (krok 2). Spočítají se výstupy všech neuronů v síti (krok 3). Výstupní vektor $\vec{y}$ je výsledkem vybavování pro vzor $\vec{x}$.

Kapitola 3

Neuronová síť vyššího řádu

Pro rozpoznávání vzorů lze využít různé techniky a postupy. V této kapitole se zaměříme na jeden z vybraných modelů neuronových sítí použitelných pro rozpoznávání vzorů. Jedná se o model neuronové sítě vyššího řádu, který publikovali Kollias a spol. v [7].

Pro snazší orientaci v následujícím textu budeme model neuronové sítě [7] označovat zkratkou *POHONN* (zkratka je odvozena z názvu článku *Performance of Higher Order Neural Networks in Invariant Recognition*).

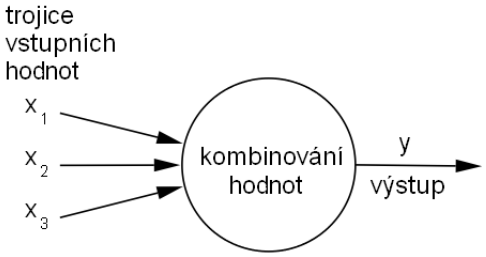
V kapitole 3.1 popíšeme model neuronové sítě vyššího řádu [7]. V kapitole 3.2 navrhneme vlastní modifikace modelu, abychom zlepšili vlastnosti modelu při rozpoznávání znaků. V kapitole 3.3 prezentujeme výsledky experimentů s ohledem na různé konfigurace modelu a srovnáme jejich úspěšnosti.

3.1 Základní model

Neuronové sítě vyšších řádů vycházejí z vrstevnatých neuronových sítí. Liší se v definici vstupní vrstvy neuronové sítě. Doposud neuronová síť obsahovala vstupní vrstvu délky odpovídající délce vstupního vektoru. Vstupní hodnotou neuronu ve vstupní vrstvě byla hodnota ze vstupního vektoru. U neuronových sítí vyšších řádů je ze vstupního vektoru vybráno několik (N) hodnot, které jsou předány do jednoho neuronu vstupní vrstvy. Ve vstupních neuronech probíhá kombinování hodnot. Počet vstupních neuronů odpovídá počtu všech N-tic hodnot ze vstupního vektoru.

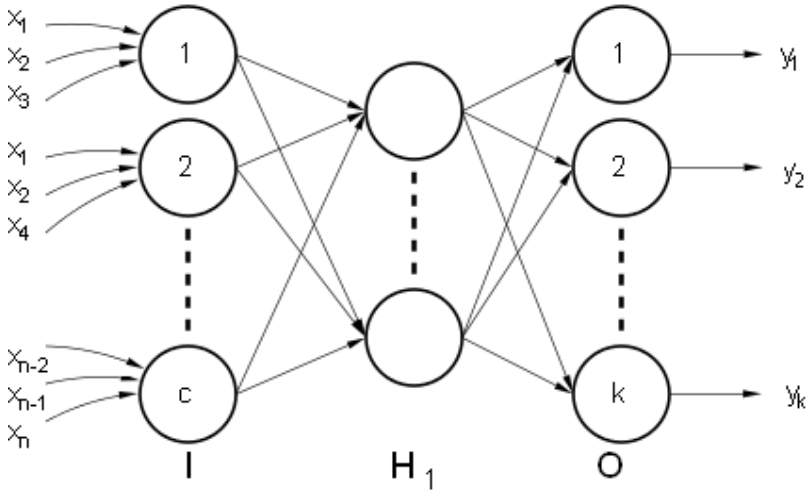
Zavedeme *řád vstupního neuronu*. Tento řád udává počet hodnot, které se vzájemně kombinují uvnitř vstupního neuronu. Základní způsob kombinování hodnot

ve vstupním neuronu je násobení (vynásobení všech hodnot dohromady). Na obrázku 3.1 je zobrazen vstupní neuron řádu tři.



Obrázek 3.1: Vstupní neuron řádu tři

Struktura neuronové sítě vyššího řádu vychází ze struktury vrstevnaté neuronové sítě. Liší se pouze vstupní vrstvou. Vstupní vrstva neuronové sítě řádu N obsahuje vstupní neurony řádu N. Jednoduchá neuronová síť řádu tři s jednou skrytou vrstvou je zobrazena na obrázku 3.2. Podobnost struktury neuronových sítí vyššího řádu s vrstevnatými neuronovými sítěmi dovoluje pro sítě vyššího řádu použít stejné techniky a algoritmy pro učení.



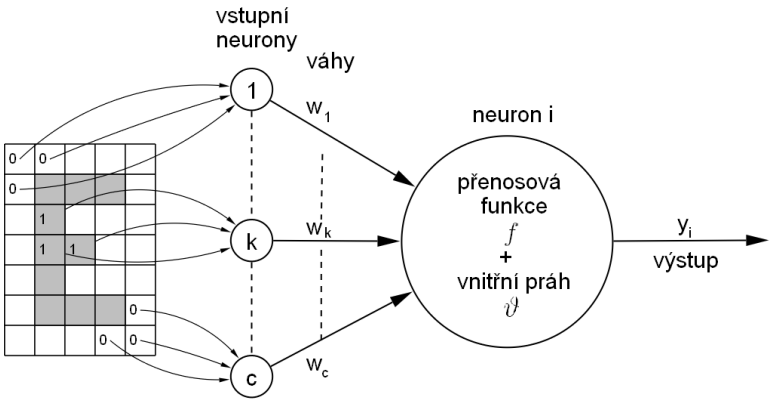
Obrázek 3.2: Neuronová síť řádu tři

Základní myšlenka modelu *POHONN* je následující. Při rozpoznávání znaků, které má být invariantní vůči vybraným deformacím, lze tyto deformace zahrnout do stavby sítě. Dle [5] lze upravit způsob kombinace hodnot ve vstupních neuronech sítě tak, že síť je schopna rozpoznávat také znaky deformované. Správně zvolený způsob kombinace vstupních hodnot ve vstupních neuronech zajistí, že výstup vstupní vrstvy bude stejný pro původní i deformovaný vzor. Neuronová síť proto oba vzory klasifikuje stejně. Tento postup vychází z biologických systémů. Podobná činnost probíhá také v lidském mozku [9].

V následujících podkapitolách popíšeme vybrané transformace a odvodíme způsoby kombinace hodnot ve vstupních neuronech [5].

3.1.1 Invariance vzhledem k transformacím

Abychom mohli definovat *invariantnost vůči transformaci*, zavedeme si označení $y_i(\vec{x})$ jako výstup i-tého neuronu ve druhé vrstvě neuronové sítě řádu N (tj. neuronu, který přijímá hodnoty od vstupních neuronů řádu N) pro vstupní vektor $\vec{x}$. Na obrázku 3.3 je vidět příklad takového neuronu pro řád tři.



Obrázek 3.3: Neuron ve druhé vrstvě

Pro neuronovou síť M řádu N a vstupní vektor $\vec{x} = (x_1, \ldots, x_n)$ lze vypočítat hodnotu $y_i(\vec{x})$ dle následujícího vztahu:

$$y_i(\vec{x}) = \sum_{i_1,\ldots,i_N \in (1,\ldots,n)} w_i(i_1, \ldots, i_N)\ x_{i_1} \ldots x_{i_N} + \vartheta, \tag{3.1}$$

kde $i_1, \ldots, i_N$ je N-tice indexů ve vstupním vektoru, ϑ je vnitřní práh neuronu a $w_i(i_1, \ldots, i_N)$ je váha mezi vstupním vektorem, který kombinuje hodnoty vstupního vektoru $x_{i_1}, \ldots, x_{i_N}$ na pozicích $i_1, \ldots, i_N$, a neuronem na pozici i.

Definice 3.1.1. *Výstup neuronu y_i je invariantní vůči transformaci t, pokud*

$$y_i(t[\vec{x}]) = y_i(\vec{x}),$$

kde $y_i(\vec{x})$ je výstup i-tého neuronu a $\vec{x}$ je vstupní vektor.

Nejjednodušší transformací pro výpočet je posunutí, na kterém podrobněji popíšeme odvozování pravidel pro modifikaci neuronové sítě vyššího řádu [5].

Posunutí vzoru

Nechť t je posunutí vstupního vektoru o m pozic. Transformace t je definována vztahem:

$$t(x_i) = \begin{cases} x_{i+m} & \text{pokud } 0 \leq i+m \leq n \\ 0 & \text{jinak,} \end{cases} \tag{3.2}$$

kde $\vec{x} = (x_1, \ldots, x_n)$ je vstupní vektor.

Aby byl výstup neuronu invariantní vůči posunutí, musí dle vztahu 3.1 platit následující rovnosti (pro řády sítě 1 a 2):

$$\begin{aligned} \sum_{i_1 \in (1,\ldots,n)} w_i(i_1) x_{i_1} &= \sum_{i_1 \in (1,\ldots,n)} w_i(i_1) x_{i_1+m} \\ \sum_{i_1,i_2 \in (1,\ldots,n)} w_i(i_1,i_2) x_{i_1} x_{i_2} &= \sum_{i_1,i_2 \in (1,\ldots,n)} w_i(i_1,i_2) x_{i_1+m} x_{i_2+m} \end{aligned} \tag{3.3}$$

Použitím substitucí $i_1 \to i_1 - m$, $i_2 \to i_2 - m$ do rovností 3.3 získáme

$$\begin{aligned} \sum_{i_1 \in (1,\ldots,n)} w_i(i_1) x_{i_1} &= \sum_{i_1 \in (1,\ldots,n)} w_i(i_1-m) x_{i_1} \\ \sum_{i_1,i_2 \in (1,\ldots,n)} w_i(i_1,i_2) x_{i_1} x_{i_2} &= \sum_{i_1,i_2 \in (1,\ldots,n)} w_i(i_1-m,i_2-m) x_{i_1} x_{i_2}. \end{aligned} \tag{3.4}$$

Z rovnosti 3.4 vyplývá následující:

$$\begin{aligned} w_i(i_1) &= w_i(i_1-m) \\ w_i(i_1,i_2) &= w_i(i_1-m,i_2-m) \end{aligned} \tag{3.5}$$

Z rovností 3.5 plyne dle [5], že invarianci nelze řešit pomocí vstupních neuronů řádu 1, protože řešením první rovnosti je konstantní funkce. Řešením druhé rovnosti (pro řád 2) je funkce $w_i(i_1,i_2) = w_i(i_1 - i_2)$. Toto řešení přiřazuje vahám 2. řádu váhy 1. řádu. Výsledné vztahy nám určují, jakým způsobem má být zkonstruována neuronová síť vyššího řádu schopná rozpoznávat vzory invariantně vůči transformaci posunutí. V tomto případě bude přičten výstup vstupního neuronu řádu 2 pro kombinaci pozic i_1 a i_2 do výstupu vstupního neuronu řádu 1 pro pozici $i_1 - i_2$. Změna struktury sítě přináší následující výhody:

- Vstupní neurony pro řád 1 jsou nezávislé na pozici vstupní hodnoty (i_1), tedy je lze vyřadit ze struktury sítě.
- Protože vstupní neurony řádu 2 odpovídají vstupním neuronům řádu 1, výrazně se snižuje velikost sítě. Dochází k výraznému snížení počtu vstupních neuronů sítě a výraznému urychlení trénování a vybavování sítě ve srovnání se základním modelem.

Pro ostatní typy transformací představíme pouze výsledné vztahy, které odvodili autoři modelu [7].

Změna velikosti vzoru

Obdobným způsobem lze odvodit i vztahy pro *změnu velikosti vzorů*. Pro změnu velikosti vzorů uvažujeme jako vstup matici $x = \{x_{ij}\}$. Odvozené vztahy jsou dle [7] následující pravidla:

$$\begin{aligned} w_i(i_1, j_1, i_2, j_2) &= w_i(i_1/j_1, i_2/j_2) \\ w_i(i_1, j_1, i_2, j_2) &= w_i(i_1/i_2, j_1/j_2) \\ w_i(i_1, j_1, i_2, j_2) &= w_i((i_1 - i_2)/(j_1 - j_2)), \end{aligned} \tag{3.6}$$

kde i_1 a j_1 jsou indexy vstupní hodnoty $x_{i_1 j_1}$ a i_2 a j_2 jsou indexy vstupní hodnoty $x_{i_2 j_2}$.

Otočení vzoru

Při odvozování vztahu pro *otočení* se počítá s polárními souřadnicemi r, Θ. Hodnota r udává vzdálenost bodu od počátku souřadnic a Θ je úhel spojnice bodu a počátku souřadnic od zvolené osy ležící v rovině (typicky osa x kartézských souřadnic). Odvozená pravidla jsou dle [7] následující:

$$\begin{aligned} w_j(r_{i_1}, \Theta_1) &= w_j(r_{i_1}) \\ w_j(r_{i_1}, r_{i_2}, \Theta_{i_1}, \Theta_{i_2}) &= w_j(r_{i_1}, r_{i_2}, \Theta_{i_1} - \Theta_{i_2}) \end{aligned} \tag{3.7}$$

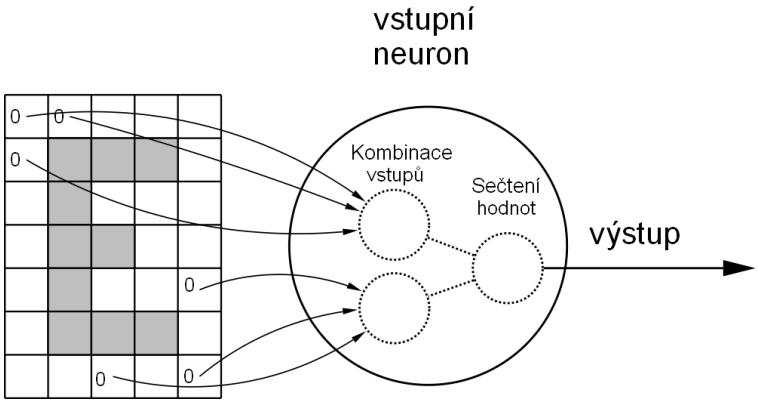
Kombinace transformací

Pro dosažení invariantnosti vůči všem transformacím i jejich kombinací je třeba dle [7] vstupních neuronů řádu 3. Dvě trojice vstupních hodnot (x_i, y_i a z_i) a (x_k, y_k a z_k) jsou přiřazeny jednomu vstupnímu neuronu, právě když

$$\alpha_i = \alpha_k \wedge \beta_i = \beta_k \wedge \gamma_i = \gamma_k, \tag{3.8}$$

kde α_i, β_i a γ_i jsou vnitřní úhly trojúhelníku tvořeného pozicemi bodů x_i, y_i a z_i ve vstupní matici, α_j, β_j a γ_j jsou vnitřní úhly trojúhelníku tvořeného pozicemi bodů x_j, y_j a z_j ve vstupní matici.

Na obrázku 3.4 jsou vidět dvě trojice bodů, které mají stejné vnitřní úhly a proto jsou hodnoty přiřazeny do stejného vstupního neuronu. Obě trojice bodů jsou zpracovány samostatně (kombinování hodnot pomocí násobení) a potom obě výsledné hodnoty sečteny. Hodnoty vnitřních úhlů trojúhelníku tvořeného trojicí indexů ve vstupní matici určují vstupní neuron, který přijme jejich hodnoty. Pokud nebudeme zohledňovat pořadí vnitřních úhlů, pak dle [7] získáme také invariantnost vůči zrcadlovému obrazu.



Obrázek 3.4: Kombinování trojic vstupů se stejnými vnitřními úhly

Protože různých trojic vnitřních úhlů vstupních trojúhelníků může být relativně velké množství, zavedli jsme si pro implementaci konstantu c_a (*míra úhlu*). Hodnota udává míru zaokrouhlení při porovnávání hodnot vnitřních úhlů. Před přiřazením vstupního neuronu trojici bodů jsou jejich vnitřní úhly vydělen konstantou c_a (např. hodnota $c_a = 1$ odpovídá zaokrouhlování na celá čísla). S větší hodnotou c_a klesá počet vstupních neuronů. Velké zaokrouhlování může způsobit snížení úspěšnosti modelu, protože trojúhelníky s různými vnitřními úhly budou síti interpretovány jako stejné trojúhelníky.

3.1.2 Algoritmus pro učení

Pro učení tohoto typu neuronové sítě vyššího řádu doporučují autoři [7] použít algoritmus Back propagation. V rámci tohoto algoritmu navrhují použít některé urychlující metody.

Modifikuje se výpočet hodnoty váhy v závislosti na parametru učení α (vztah 2.1). Tento způsob vychází z práce R. Jacobse [6]. V původním algoritmu Back propagation je parametr učení jediná konstanta pro celou síť. Urychlovací metoda definuje tento parametr specificky pro každou váhu sítě. Pokud při učení dochází ke zvyšování nebo snižování hodnoty váhy, je lokální parametr učení zvýšen, čímž dojde k urychlení učení. Naopak pokud se střídá zvyšování a snižování hodnoty váhy, je lokální parametr učení snížen. V tomto případě je vztah 2.1 v algoritmu Back propagation modifikován podle vztahu:

$$w_{ij}(t+1) = w_{ij}(t) + \alpha_{ij}(t)\delta_j y_i + \alpha_m(w_{ij}(t) - w_{ij}(t-1))$$
$$\alpha_{ij}(t) = d_m \alpha_{ij}(t-1),$$

kde $\alpha_{ij}(t)$ je specifický parametr učení pro váhu w_{ij} při iteraci t a d_m udává velikost změny parametru učení. Hodnotu d_m doporučují autoři článku [7] zvolit následovně

$$d_m = \begin{cases} k & \text{pokud } \mathrm{w}_{ij}(t) > w_{ij}(t-1) \\ \frac{1}{k} & \text{jinak,} \end{cases}$$

kde $k \in < 1,1; 1,3 >$.

3.2 Modifikace modelu

Protože jsou hodnoty ve vstupních neuronech kombinovány pomocí násobení, pak libovolná kombinace hodnot obsahující hodnotu 0 dá jako výsledek hodnotu 0. Proto v následujícím textu a v definicích budeme jako vstupní trojúhelníky uvažovat pouze trojice bodů ze vzoru, které neobsahují hodnotu 0.

Nyní popíšeme námi navržené modifikace modelu, které umožní rozpoznávat vzory s ohledem na kombinaci všech dříve zmíněných transformací. Dle původního návrhu síť považuje vstupní trojúhelníky se stejnými vnitřními úhly, ale odlišnou velikostí, jako stejné vstupní hodnoty. Myslíme si, že zohledněním velikosti vstupních trojúhelníků lze pozitivně ovlivnit úspěšnost modelu. Původní model nedokáže rozpoznat vzory, které maji vstupní trojúhelníky stejné vnitřní úhly a liší se pouze obsahem. V naší modifikaci rozšíříme definici 3.8 o velikost vstupního trojúhelníku. Tuto hodnotu však musíme brát relativně vůči celkové velikosti vzoru, aby vztah fungoval i pro rozpoznávání s ohledem na změnu velikosti.

Přidáním obsahu trojúhelníku do vztahu 3.8 se zvýší počet různých kombinací vstupních hodnot a tím i velikost vstupní vrstvy. Zavedeme proto konstantu s_n, která omezí tento růst. Pro skupinu vstupních trojúhelníků se stejnými vnitřními úhly udává s_n maximální počet kombinací, které z této skupiny mohou vzniknout přidáním obsahu do vztahu 3.8. Tyto různé kombinace označíme jako *třídy obsahu* (s_c). Například $s_n = 2$ udává, že skupina vstupních trojúhelníků bude rozdělena do dvou tříd obsahu - na „menší“ a „větší“ trojúhelníky.

Vztah 3.8 modifikujeme následovně:

$$\alpha_i = \alpha_k \wedge \beta_i = \beta_k \wedge \gamma_i = \gamma_k \wedge s_{c_i} = s_{c_k},$$

kde s_{c_i} a s_{c_k} jsou třídy obsahu trojúhelníků.

Navrhli jsme dvě varianty výpočtu třídy obsahu. První varianta jednoduše odvozuje třídu obsahu trojúhelníku dle poměru obsahu trojúhelníku k obsahu maximálního vstupního trojúhelníku ve vzoru.

Definice 3.2.1. *Třída obsahu s_c pro vstupní trojúhelník ze vzoru je rovna*

$$s_c = (S/s_{max})s_n,$$

kde S je obsah vstupního trojúhelníku, s_n je maximální počet tříd obsahu a s_{max} je maximální hodnota obsahu ze všech vstupních trojúhelníků ve vzoru.

První varianta by měla dle našeho soudu výrazně zvýšit úspěšnost rozpoznávání celého modelu, protože umožní síti přijímat nové informace o velikosti vstupních trojúhelníků. Obáváme se však snížení úspěšnosti při rozpoznávání poškozených vzorů. I nepatrné poškození vzoru může změnit hodnotu s_{max} a tím ovlivnit třídy obsahu všech vstupních trojúhelníků. Proto jsme navrhli také druhou variantu, které se tento problém netýká. Druhá varianta odvozuje třídu obsahu relativně k mediánu obsahů všech vstupních trojúhelníků ze vzoru.

Definice 3.2.2. *Třída obsahu s_c pro trojúhelník ze vzoru je rovna*

$$s_c = \begin{cases} s_n & \textit{pokud } S > 2s_{med} \\ S/(2s_{med})s_n & \textit{jinak,} \end{cases}$$

kde S je obsah vstupního trojúhelníku, s_n je maximální počet tříd obsahu a s_{med} je medián hodnot obsahů všech vstupních trojúhelníků ve vzoru.

Použití mediánu ve druhé variantě by mělo snížit negativní vlastnosti modelu při rozpoznávání poškozených vzorů. Pokud ve vzoru po poškození vzniknou nové příliš velké nebo příliš malé vstupní trojúhelníky, nebude změna mediánu všech vstupních trojúhelníků tak výrazná. Odvozené třídy obsahu jednotlivých vstupních trojúhelníků zůstanou téměř stejné.

3.3 Srovnání

V následující kapitole srovnáme vlastnosti původního modelu [7] s modifikovaným modelem, který jsme navrhli. Zaměříme se na srovnání úspěšnosti modelů při učení s ohledem na jednotlivé transformace a kombinace transformací. Ověříme schopnosti modelu rozpoznat vzory poškozené (zašumělé).

Měření jsou provedena na skupině dat odpovídající velkým písmenům české abecedy bez diakritiky (ukázka v Příloze E). Pro porovnání výkonu modelů vzhledem k velikosti vstupních vzorů jsou měření provedena na datech v různých rozlišeních. Jde o vzory s rozlišením 10x10, 18x18 a 30x30 pixelů.

Pro každé rozlišení je provedeno více měření s různými hodnotami parametrů. Pro každé ověřování vlastností modelu s ohledem na určitou transformaci jsou tyto kombinace parametrů stejné. Celkově se jedná o 35 různých kombinací parametrů pro původní model *POHONN* a 110 kombinací parametrů pro námi modifikovaný model *POHONN* (pro každou transformaci). V práci prezentujeme pouze nejlepší výsledky pro dané rozlišení a zkoumanou transformaci. Kompletní výsledky měření se nacházejí na přiloženém CD (viz Příloha A).

Následující podkapitoly jsou věnovány jednotlivým deformacím a kombinacím deformací. Na závěr zhodnotíme naměřené výsledky.

3.3.1 Úspěšnost s ohledem na rotaci

Nejprve se zaměříme na vlastnosti modelu při rozpoznávání s ohledem na pootočení vzorů. Testovací množinu tvoří 286 vzorů s různými stupni otočení (5°, 10°, 25°, 45°, 80°, 95°, 100°, 150°, 250° a 280°). Pro trénování využijeme základní data obsahující 26 vzorů. V tabulce 3.1 jsou shrnuty nejlepší dosažené výsledky původního modelu, jak jej navrhli autoři v článku [7].

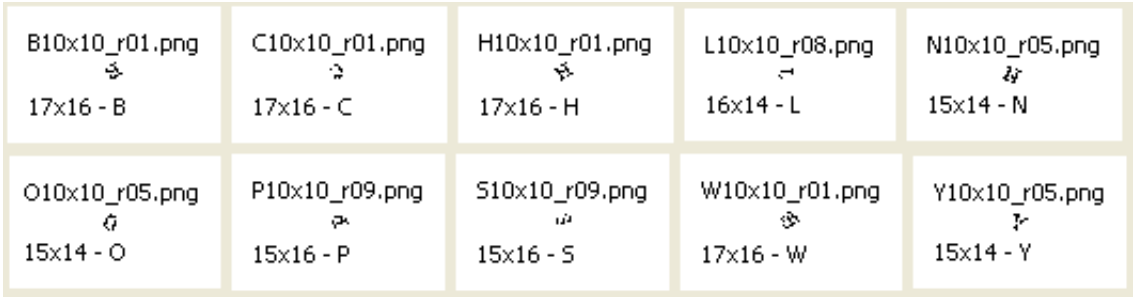
Význam jednotlivých atributů je popsán v Příloze D. Trénovací úspěšnost udává procentuální úspěšnost rozpoznávání na trénovací množině, testovací úspěšnost udává procentuální úspěšnost na testovací množině.

Výsledky pro nízká rozlišení (10x10) dosahují dle našeho soudu přijatelných hodnot. Výsledky jsou relativně dobré, i když je část testovacích vzorů při tomto rozlišení obtížně klasifikovatelná lidským okem. Příklady takových vzorů se nacházejí na obrázku 3.5. Výsledky pro vyšší rozlišení dosahují velice nízkých hodnot i přesto, že pro lidské oko jsou snadněji rozpoznatelné než při nízkém rozlišení (viz Příloha E). Domníváme se, že nízká úspěšnost modelu při rozpoznávání vzorů ve vyšším rozlišení je způsobena vynecháním informací o velikosti vstupních trojúhelníků. Ve vzorech s vyšším rozlišením je celkově větší počet vstupních

Rozliš. vzorů	Koef. eta	Koef. alfa	Koef. d	Max d	Míra úhlu	Pořadí úhlů	Počet iterací	Trénovací úspěšnost	Testovací úspěšnost
10x10	0,3	0,7	1,02	2	1	ne	$1x10^5$	100,00%	**40,21%**
10x10	0,3	0,7	1,02	2	2	ne	$1x10^5$	100,00%	**40,21%**
10x10	0,3	0,7	1,02	2	3	ne	$1x10^5$	100,00%	**40,21%**
18x18	0,3	0,7	1,02	2	1	ne	$1x10^5$	100,00%	**24,48%**
18x18	0,3	0,7	1,02	2	2	ne	$1x10^5$	100,00%	**23,43%**
18x18	0,3	0,7	1,02	2	1,5	ne	$1x10^5$	100,00%	**22,73%**
30x30	0,3	0,7	1,02	2	1	ne	$1x10^5$	80,77%	**20,63%**
30x30	0,3	0,7	1,02	2	1,5	ne	$1x10^5$	65,38%	**17,48%**
30x30	0,3	0,7	1,02	2	1	ano	$1x10^5$	100,00%	**12,59%**

Tabulka 3.1: Výsledky původního modelu *POHONN* - rotace

trojúhelníků. Díky zaokrouhlování a následnému přiřazování vstupního neuronu na základě vnitřních úhlů dochází k častému splývání vstupních trojúhelníků. Mnoho vstupních trojúhelníků různých velikostí je síti předkládáno jako stejný vstup. Právě u vzorů ve vyšším rozlišení je těchto „stejných“ trojúhelníků více. Tím dochází k větší ztrátě informací, které jsou potřebné pro správnou klasifikaci vzoru, a tím i snižování úspěšnosti rozpoznávání.



Obrázek 3.5: Ukázka hůře rozpoznatelných vzorů

V tabulce 3.2 jsou zobrazeny nejlepší dosažené výsledky námi navržené modifikace modelu. Úspěšnosti pro nízká rozlišení jsou mírně lepší než u původního modelu. Rozpoznávání vzorů ve vyšším rozlišení je výrazně úspěšnější. Z výsledků je patrné, že nejúspěšnější konfigurace modifikovaného modelu pro vyšší rozlišení nedokázaly rozpoznat celou trénovací množinu. Je pravděpodobné, že při vysokém zaokrouhlování (míra úhlu) jsou některé trénovací vzory sítí klasifikovány jako stejné. Počet takto chybně rozpoznaných vzorů při testování je však nižší než počet chybně rozpoznaných vzorů při menším zaokrouhlování. Při malém zaokrouhlování je sice rozpoznána správně celá trénovací množina, ale je výrazně nižší úspěšnost na testovacích datech.

Rozliš. vzorů	Koef. eta	Koef. alfa	Koef. d	Max d	Typ obsahu	Míra úhlu	Třída obsahu	Počet iterací	Trénovací úspěšnost	Testovací úspěšnost
10x10	0,3	0,7	1,02	1,2	s_{max}	6	2	1x10^5	100,00%	**42,31%**
10x10	0,3	0,7	1,02	1,2	s_{max}	6	2	1x10^5	100,00%	**41,96%**
10x10	0,3	0,7	1,02	1,2	s_{max}	3	2	1x10^5	100,00%	**40,91%**
18x18	0,3	0,7	1,02	1,2	s_{max}	10	6	1x10^5	96,15%	**38,46%**
18x18	0,3	0,7	1,02	1,2	s_{max}	20	8	1x10^5	73,08%	**37,76%**
18x18	0,3	0,7	1,02	1,2	s_{max}	6	6	1x10^5	100,00%	**36,71%**
30x30	0,3	0,7	1,02	1,2	s_{max}	6	10	1x10^5	92,31%	**48,60%**
30x30	0,3	0,7	1,02	1,2	s_{max}	6	20	1x10^5	100,00%	**44,76%**
30x30	0,3	0,7	1,02	1,2	s_{max}	3	2	1x10^5	92,31%	**44,41%**

Tabulka 3.2: Výsledky námi modifikovaného modelu *POHONN* - rotace

Z výsledků modifikovaného modelu je vidět, že úspěšné konfigurace modelu měly nastavenu míru úhlu na vyšší hodnoty. U původního modelu se však konfigurace s vyššími mírami úhlů nebyly schopny naučit úspěšně ani trénovací množinu (viz kompletní výsledky měření na přiloženém CD). Můžeme usuzovat, že u původního modelu dochází nezapočítáním obsahů vstupních trojúhelníků ke snižování úspěšnosti rozpoznávání. Různé vzory mohou totiž obsahovat podobné trojúhelníky, které se liší pouze velikostí (obsahem). Původní model není schopen tyto vzory rozlišit.

3.3.2 Úspěšnost s ohledem na posunutí

Jak je zřejmé z definice modelu *POHONN*, úspěšnost s ohledem na posunutí je u všech testovaných modelů 100%. Při předzpracování vzoru jsou veškeré informace ovlivněné posunutím vzoru ve vstupní matici ignorovány. Tedy síť je schopna rozpoznat každý naučený vzor bez ohledu na umístění ve vstupní matici.

Také jsme ověřili, že model při rozpoznávání vzorů s ohledem na některou deformaci dosahuje stejných výsledků jako při kombinaci posunutí a původní deformace. Proto se při následujících měřeních nebudeme zabývat úspěšností ostatních deformací v kombinaci s posunutím.

3.3.3 Úspěšnost s ohledem na změnu velikosti

Nyní se zaměříme na úspěšnost modelu při rozpoznávání s ohledem na změnu velikosti. Pro trénování jsme zvolili základní množinu vzorů obsahující 26 velkých písmen české abecedy ve třech různých velikostech - 10x10, 18x18 a 30x30. Testovací množina obsahuje 364 vzorů. Jedná se o trénovací vzory zmenšené (případně

zvětšené) o 5% - 30%. V tabulce 3.3 jsou shrnuty nejlepší dosažené výsledky původního modelu, jak jej navrhli autoři [7].

Rozliš. vzorů	**Koef. eta**	**Koef. alfa**	**Koef. d**	**Max d**	**Míra úhlu**	**Pořadí úhlů**	**Počet iterací**	**Trénovací úspěšnost**	**Testovací úspěšnost**
10x10	0,3	0,7	1,02	2	8	ano	$1x10^5$	100,00%	**26,67%**
10x10	0,3	0,7	1,02	2	5	ano	$1x10^5$	100,00%	**24,62%**
10x10	0,3	0,7	1,02	2	9	ano	$1x10^5$	100,00%	**24,36%**
18x18	0,3	0,7	1,02	2	5	ano	$2x10^5$	96,15%	**23,85%**
18x18	0,3	0,7	1,02	2	4	ano	$2x10^5$	100,00%	**22,56%**
18x18	0,3	0,7	1,02	2	2	ano	$1x10^5$	100,00%	**19,74%**
30x30	0,3	0,7	1,02	2	2	ano	$1x10^5$	100,00%	**27,44%**
30x30	0,3	0,7	1,02	2	1,5	ano	$1x10^5$	100,00%	**27,18%**
30x30	0,3	0,7	1,02	2	3	ano	$1x10^5$	80,77%	**25,64%**

Tabulka 3.3: Výsledky původního modelu *POHONN* - změna velikosti

Výsledky pro jednotlivé velikosti (rozlišení) vzorů jsou podobné. Konfigurace pro jednotlivá rozlišení vzorů se liší pouze hodnotami míry úhlu - s rostoucím rozlišením vzorů se snižuje míra úhlu. Vzory s vyšším rozlišením obsahují více vnitřních trojúhelníků, jejichž vnitřní úhly se liší, než vzory s nižším rozlišením. Díky většímu počtu odlišných hodnot vnitřních úhlů pak model s nižší mírou úhlu umožňuje klasifikovat jednotlivé vzory přesněji.

Rozliš. vzorů	**Koef. eta**	**Koef. alfa**	**Koef. d**	**Max d**	**Typ obsahu**	**Míra úhlu**	**Tříd obsahu**	**Počet iterací**	**Trénovací úspěšnost**	**Testovací úspěšnost**
10x10	0,3	0,7	1,02	1,2	s_{max}	10	2	$1x10^5$	100,00%	**25,90%**
10x10	0,3	0,7	1,02	1,2	s_{max}	10	2	$1x10^5$	100,00%	**25,38%**
10x10	0,3	0,7	1,02	1,2	s_{max}	3	2	$1x10^5$	100,00%	**25,13%**
18x18	0,3	0,7	1,02	1,2	s_{max}	10	6	$1x10^5$	96,15%	**26,41%**
18x18	0,3	0,7	1,02	1,2	s_{max}	6	2	$1x10^5$	92,31%	**25,64%**
18x18	0,3	0,7	1,02	1,2	s_{max}	10	2	$1x10^5$	80,77%	**25,64%**
30x30	0,3	0,7	1,02	1,2	s_{max}	3	2	$1x10^5$	92,31%	**48,72%**
30x30	0,3	0,7	1,02	1,2	s_{max}	3	2	$1x10^5$	92,31%	**48,21%**
30x30	0,3	0,7	1,02	1,2	s_{max}	6	2	$1x10^5$	76,92%	**42,56%**

Tabulka 3.4: Výsledky námi modifikovaného modelu *POHONN* - změna velikosti

V tabulce 3.4 se nachází nejlepší dosažené výsledky námi navržené modifikace modelu. Je vidět, že pro rozlišení 10x10 jsou výsledky původního modelu ve srovnání s námi modifikovaným modelem téměř stejné. Pro rozlišení 18x18 jsou výsledky modifikovaného modelu mírně lepší a pro 30x30 výrazně lepší. Vztah míry obsahu vzhledem k rozlišení vzorů je u modifikovaného modelu podobný jako u původního. Z výsledků je vidět, že nejúspěšnější konfigurace modifikovaného

modelu mají konstantu s_n (počet tříd obsahu) rovnu 2. Vstupní trojúhelníky jsou rozdělovány pouze do 2 tříd podle obsahu, aby bylo dosaženo maximální úspěšnosti při rozpoznávání.

3.3.4 Úspěšnost s ohledem na kombinace

Nyní ověříme vlastnosti modelu při rozpoznávání vzorů, na které byly aplikovány všechny předešlé deformace - rotace, posunutí a změna velikosti. Trénovací množina je stejná jako u předešlých měření (26 velkých písmen české abecedy ve třech různých velikostech - 10x10, 18x18 a 30x30). Testovací množina je pro každé rozlišení sestavena z 260 vzorů, na které byly aplikovány kombinace následujících deformací - rotace o 0° - 270°, posunutí o 1 - 6 pixelů libovolným směrem a zmenšení (zvětšení) o 5% - 40%. Nejlepší dosažené výsledky původního modelu jsou zobrazeny v tabulce 3.5.

Rozliš. vzorů	**Koef. eta**	**Koef. alfa**	**Koef. d**	**Max d**	**Míra úhlu**	**Pořadí úhlů**	**Počet iterací**	**Trénovací úspěšnost**	**Testovací úspěšnost**
10x10	0,3	0,7	1,02	2	5	ne	$1x10^5$	100,00%	**20,98%**
10x10	0,3	0,7	1,02	2	3	ne	$1x10^5$	100,00%	**20,63%**
10x10	0,3	0,7	1,02	2	4	ne	$1x10^5$	100,00%	**20,63%**
18x18	0,3	0,7	1,02	2	1	ne	$1x10^5$	100,00%	**22,03%**
18x18	0,3	0,7	1,02	2	2	ne	$1x10^5$	100,00%	**21,68%**
18x18	0,3	0,7	1,02	2	1,5	ne	$1x10^5$	100,00%	**21,33%**
30x30	0,3	0,7	1,02	2	1	ne	$1x10^5$	80,77%	**18,18%**
30x30	0,3	0,7	1,02	2	1,5	ne	$1x10^5$	65,38%	**15,03%**
30x30	0,3	0,7	1,02	2	1,5	ano	$1x10^5$	100,00%	**12,59%**

Tabulka 3.5: Výsledky původního modelu *POHONN* - kombinace změn

Dosažené výsledky jsou svými hodnotami i vlastnostmi konfigurací podobné výsledkům u předešlých měření. Výsledky potvrzují, že původní model dosahuje nejhorších výsledků u vzorů s nejvyšším rozlišením (30x30). Pro srovnání jsou v tabulce 3.6 zobrazeny nejlepší dosažené výsledky námi modifikovaného modelu.

Výsledky u modifikovaného modelu potvrzují předešlá měření. Modifikovaný model dosahuje lepších výsledků než model původní. Nejúspěšnější konfigurace modelu pro nejvyšší rozlišení nedokázaly rozpoznat celou trénovací množinu.

Rozliš. vzorů	Koef. eta	Koef. alfa	Koef. d	Max d	Typ obsahu	Míra úhlu	Tříd obsahu	Počet iterací	Trénovací úspěšnost	Testovací úspěšnost
10x10	0,3	0,7	1,02	1,2	s_{max}	10	2	1x10^5	100,00%	**22,38%**
10x10	0,3	0,7	1,02	1,2	s_{med}	10	2	1x10^5	100,00%	**21,68%**
10x10	0,3	0,7	1,02	1,2	s_{max}	10	2	1x10^5	100,00%	**21,33%**
18x18	0,3	0,7	1,02	1,2	s_{max}	3	2	1x10^5	100,00%	**24,13%**
18x18	0,3	0,7	1,02	1,2	s_{max}	3	2	1x10^5	100,00%	**23,78%**
18x18	0,3	0,7	1,02	1,2	s_{max}	1	2	1x10^5	100,00%	**22,03%**
30x30	0,3	0,7	1,02	1,2	s_{max}	6	6	1x10^5	96,15%	**39,16%**
30x30	0,3	0,7	1,02	1,2	s_{max}	20	6	1x10^5	76,92%	**37,76%**
30x30	0,3	0,7	1,02	1,2	s_{max}	6	10	1x10^5	92,31%	**37,06%**

Tabulka 3.6: Výsledky námi modifikovaného modelu *POHONN* - kombinace změn

3.3.5 Vliv šumu

Poslední srovnání ověří vlastnosti modelu při rozpoznávání vzorů poškozených. Trénovací množina je stejná jako u předešlých měření. Testovací množina je tvořena 156 vzory, které vznikly z trénovací množiny přidáním náhodného šumu v rozmezí 1% - 5%. Hodnota každého pixelu vstupního vzoru byla změněna na opačnou s danou pravděpodobností (1% - 5%). Nejlepší dosažené výsledky pro původní model jsou zobrazeny v tabulce 3.7. Výsledky pro nízká rozlišení jsou relativně vysoká, bohužel stejně jako v předešlých měřeních jsou výsledky pro rozlišení 30x30 výrazně horší.

Rozliš. vzorů	Koef. eta	Koef. alfa	Koef. d	Max d	Míra úhlu	Pořadí úhlů	Počet iterací	Trénovací úspěšnost	Testovací úspěšnost
10x10	0,3	0,7	1,02	2	1,5	ano	1x10^5	100,00%	**64,74%**
10x10	0,3	0,7	1,02	2	1	ano	1x10^5	100,00%	**63,46%**
10x10	0,3	0,7	1,02	2	2	ano	1x10^5	100,00%	**63,46%**
18x18	0,3	0,7	1,02	2	1	ano	1x10^5	100,00%	**89,74%**
18x18	0,3	0,7	1,02	2	1,5	ano	5x10^4	100,00%	**85,90%**
18x18	0,3	0,7	1,02	2	2	ano	1x10^5	100,00%	**82,05%**
30x30	0,3	0,7	1,02	2	1,5	ano	1x10^5	100,00%	**21,79%**
30x30	0,3	0,7	1,02	2	1,5	ne	1x10^5	65,38%	**14,74%**
30x30	0,3	0,7	1,02	2	4	ano	1x10^5	42,31%	**10,90%**

Tabulka 3.7: Výsledky původního modelu *POHONN* - vliv šumu

V tabulce 3.8 jsou zobrazeny výsledky námi modifikovaného modelu. Ve srovnání s původní verzí modelu došlo pro všechna rozlišení ke snížení úspěšnosti až o 60%. Z výsledků je patrné, že naše modifikace výrazně negativně ovlivnila schopnost modelu rozpoznávat vzory poškozené. Zhoršení je pravděpodobně způsobeno tím, že po přidání šumu vzniknou ve vzoru nové velké trojúhelníky, které

výrazně změní všechny vstupní hodnoty. Vstupní hodnoty jsou udávány relativně k největšímu trojúhelníku (nebo mediánu trojúhelníků), proto je velké množství vstupních hodnot jiných a dosažené úspěšnosti jsou tak nízké.

Rozliš. vzorů	Koef. eta	Koef. alfa	Koef. d	Max d	Typ obsahu	Míra úhlu	Tříd obsahu	Počet iterací	Trénovací úspěšnost	Testovací úspěšnost
10x10	0,3	0,7	1,02	1,2	s_{max}	1	2	1x10^5	100,00%	**41,67%**
10x10	0,3	0,7	1,02	1,2	s_{max}	1	2	1x10^5	100,00%	**41,03%**
10x10	0,3	0,7	1,02	1,2	s_{max}	3	2	1x10^5	100,00%	**37,18%**
30x30	0,3	0,7	1,02	1,2	s_{max}	1	2	1x10^5	100,00%	**30,13%**
30x30	0,3	0,7	1,02	1,2	s_{max}	1	2	1x10^5	100,00%	**30,13%**
30x30	0,3	0,7	1,02	1,2	s_{max}	3	2	1x10^5	100,00%	**26,92%**
30x30	0,3	0,7	1,02	1,2	s_{med}	3	2	1x10^5	100,00%	**20,51%**
30x30	0,3	0,7	1,02	1,2	s_{max}	3	2	1x10^5	92,31%	**19,87%**
30x30	0,3	0,7	1,02	1,2	s_{max}	3	20	1x10^5	100,00%	**19,87%**

Tabulka 3.8: Výsledky námi modifikovaného modelu *POHONN* - vliv šumu

3.4 Shrnutí

Experimentální ověření potvrdilo, že model *POHONN* dokáže rozpoznávat znaky s ohledem na různé invariance. Dosažené hodnoty nejsou příliš vysoké, avšak ve srovnání s nespecializovanými modely a neuronovými sítěmi jde o výrazné zlepšení. Praktická měření ukázala, že model dosahuje lepších výsledků pro vzory s nízkým rozlišením. Nejlepších výsledků model *POHONN* dosahuje při rozpoznávání vzorů s ohledem na rotaci. Model dosahuje dobrých výsledků i při rozpoznávání vzorů poškozených náhodným šumem.

Z naměřených výsledků je vidět, že rozdílné výsledky při zohledňování (nezohledňování) pořadí vstupních úhlů jsou závislé na typu deformace, kterou testovací vzory podstoupily. Nezohledňování pořadí vstupních úhlů přináší lepší výsledky při rozpoznávání s ohledem na změnu velikosti a při rozpoznávání poškozených vzorů. Pokud jsou vzory deformovány otočením, zohledňování pořadí výrazně snižuje úspěšnost rozpoznávání.

Námi navržená modifikace modelu dosáhla téměř při všech měřeních lepších výsledků. Nejkladněji hodnotíme výrazně lepší výsledky při rozpoznávání vzorů ve vyšším rozlišení. Narozdíl od původního modelu, kdy výsledky pro vyšší rozlišení jsou řádové o několik procent horší, u modifikovaného modelu jsou až o desítky procent lepší. Slabou stránkou námi modifikovaného modelu je snížená schopnost správně klasifikovat vzory poškozené náhodným šumem.

Z obou variant výpočtu třídy obsahu se osvědčila pouze varianta odvozování

od maximálního obsahu vstupních trojúhelníků (Definice 3.2.1). Varianta s použitím mediánu (Definice 3.2.2) dopadla hůře téměř při každém měření. Při rozpoznávání poškozených vzorů jsou rozdíly mezi oběma variantami relativně malé, ale celkově je varianta s použitím mediánu horší. Pravděpodobně je medián obsahu vstupních trojúhelníků příliš malý, protože většina vstupních trojúhelníků má malý obsah (trojice bodů často leží blízko sebe). Takto nízký medián proto pomocí námi definovaného výpočtu nerozlišuje obsah vstupních trojúhelníků správně.

Kapitola 4

Modifikovaná síť vyššího řádu

Tato kapitola je zaměřena na model vyšší neuronové sítě, který představili ve své práci E. Artymov a O. Yadid-Petch [1].

Nejprve představíme model a popíšeme jeho teoretické základy. Shrneme jeho klady a slabiny. Dále představíme naše modifikace s cílem zvýšit úspěšnost modelu při rozpoznávání. Oba modely experimentálně ověříme. Na závěr zhodnotíme vlastnosti původního modelu i modelu s námi navrženými modifikacemi a tyto výsledky srovnáme.

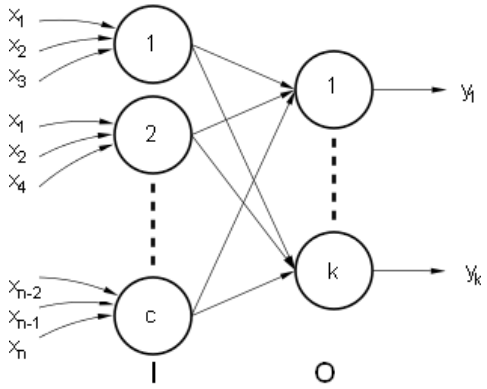
4.1 Základní model

Model dle [1] budeme označovat zkratkou *MHONN* (z názvu článku Modified High-Order Neural Network ...). Model *MHONN* vychází ze základního modelu neuronových sítí vyššího řádu (viz kapitola 3). Jedná se o neuronovou síť řádu 3, která neobsahuje žádnou skrytou vrstvu. Na obrázku 4.1 je zobrazena struktura modelu *MHONN*. Obsahuje pouze vstupní vrstvu I a výstupní vrstvu O.

Autoři [1] pozměnili způsob předkládání a zpracování vzorů a navrhli vlastní algoritmus učení sítě. Model *MHONN* by měl dle slov autorů přinést kromě lepších rozpoznávacích schopností také významné urychlení výpočetního času, snížení paměťových nároků a snížení počtu vstupních neuronů sítě.

4.1.1 Předzpracování vzorů

Jednou z negativních vlastností neuronových sítí vyšších řádů je exponenciální růst počtu vstupních neuronů sítě v závislosti na řádu sítě. Dle základní definice



Obrázek 4.1: Modifikovaná neuronová síť řádu tři

neuronových sítí vyšších řádů odpovídá každé kombinaci vstupních bodů ve vzoru odlišný vstupní neuron sítě. Protože u modelu *MHONN* je řád sítě 3, jsou vstupní hodnoty sítě trojice bodů tzv. trojúhelníky. V tabulce 4.1 je vidět růst počtu trojúhelníků ve vzoru v závislosti na rozměrech vstupního vzoru [1]. Je vidět, že s pomocí základního modelu neuronových sítí vyšších řádů již u sítí řádů tři nelze rozpoznávat vzory větších rozměrů dostatečně rychle.

Rozměr vzoru	Počet vah
10x10	161700
18x18	5616324
30x30	726571800
50x50	2601042500
100x100	166616670000

Tabulka 4.1: Růst počtu trojúhelníků ve vzoru v závislosti na rozměru vzoru.

Kromě výpočetní složitosti, která je dána množstvím vah, je třeba vzít v úvahu také paměťové nároky rozpoznávání. Například při předzpracování vzoru o rozměrech 40x40 pixelů jsou spočteny všechny trojice pozic a uloženy ukazatele k těmto hodnotám. Každý ukazatel potřebuje minimálně dva byty paměti (pro něj samotného a pro jemu příslušnou hodnotu). Výsledkem je minimálně $1,3628\text{x}10^9$ bytů, což jsou hodnoty vskutku enormní [1].

Arymov a Yadid-Petch [1] navrhli model *MHONN* tak, aby efektivně sdílel váhy aproximativně podobných trojúhelníků (trojic vstupních pozic) a tím výrazně snižoval negativní vlastnosti neuronových sítí vyšších řádů popsané v předešlých odstavcích. Autoři uvádějí, že jejich model je schopen efektivně rozpoznávat vzory o velikostech větších než 100x100 pixelů bez předešlého zpracování, vynechání nebo zjednodušení jakékoliv informace vzoru.

V [1] je navrženo několik způsobů jak zefektivnit rozpoznávání vzorů a sní-

žit časové a paměťové nároky. Tyto jednotlivé způsoby popíšeme v následujících podkapitolách. Na závěr popíšeme algoritmus pro učení modelu, který představili autoři současně s jejich modelem.

4.1.2 Aproximativně podobné trojúhelníky

Jak již bylo řečeno v předešlých kapitolách, vstupních trojúhelníků pro vstupy s většími rozměry je veliké množství. Nejdůležitějším krokem ke snížení počtu vstupních trojúhelníků je vytváření tzv. *aproximativně podobných trojúhelníků*, které byla navrženo v [12]. Hlavní myšlenkou je přiřazení podobných trojúhelníků jedinému vstupnímu neuronu sítě a tím snížení počtu vstupních neuronů sítě.

Definice 4.1.1. *Dva vstupní trojúhelníky T_i a T_j jsou si podobné, pokud náleží do stejné třídy podobnosti určené kladnými čísly k, l a m. Hodnoty k, l a m pro trojúhelník T jsou určeny následujícími nerovnostmi:*

$$(k-1)c_a \leq \alpha < kc_a$$
$$(l-1)c_a \leq \beta < lc_a$$
$$(m-1)c_s \leq S < mc_s,$$

kde c_a je konstanta určující přesnost pro třídu úhlu, c_s je konstanta určující přesnost pro třídu obsahu, α je nejmenší vnitřní úhel trojúhelníka T, β je druhý nejmenší vnitřní úhel trojúhelníka T a S je obsah trojúhelníka T.

Nahrazením vstupních trojúhelníků za trojice hodnot k, l a m v závislosti na konstantách c_a a c_s dostaneme nižší počet různých vstupních hodnot. Různé vstupní trojúhelníky mohou patřit do stejné třídy podobnosti a tím se zredukuje počet odlišných vstupních hodnot. Čím větších hodnot budou nabývat konstanty c_a a c_s, tím menší počet různých vstupních hodnot získáme.

4.1.3 Snižování paměťových nároků

Arymov a Yadid-Petch [1] navrhli způsob ukládání dat, který v kombinaci s předešlými návrhy výrazně snižuje paměťovou náročnost sítě. Vstupní vzory jsou předzpracovány dle následující definice.

Definice 4.1.2. *Nechť $\vec{t_1}, \vec{t_2}, \ldots, \vec{t_n}$ jsou trénovací vzory. Potom předzpracované vzory $\vec{p_1}, \vec{p_2}, \ldots, \vec{p_z}$ získáme následujícím způsobem:*

(1) Všechny trojúhelníky z trénovacích vzorů převedeme na trojici hodnot k, l a m přiřazující trojúhelníku příslušnou třídu podobnosti (viz definice 4.1.1).

(2) Všechny možné kombinace hodnot k, l a m očíslujeme hodnotami $1, \ldots, z$.

(3) Každý trénovací vzor $\vec{t_i}$ převedeme na vektor $\vec{p_i} = (p_{i1}, \ldots, p_{ij})$ tak, že hodnota p_{ik} ($1 \leq k \leq j$) je rovna počtu trojúhelníků dané třídy podobnosti (označené indexem k), které se nacházejí ve vzoru $\vec{t_i}$.

Předzpracování dle předešlé definice výrazně sníží paměťové nároky pro učení. Místo všech trojúhelníků z trénovacích vektorů stačí uchovávat v paměti kombinace hodnot k, l, m a předzpracované vektory. Uchovávaná data reprezentuje tabulka 4.2.

Třída podobnosti	Kombinace k, l a m			Vzor $\vec{t_1}$	Vzor $\vec{t_2}$	...	Vzor $\vec{t_n}$
1	k_1	l_1	m_1	p_{11}	p_{21}		p_{n1}
2	k_2	l_2	m_2	p_{12}	p_{22}		p_{n2}
⋮							
z	k_z	l_z	m_z	p_{1z}	p_{2z}		p_{nz}

Tabulka 4.2: Data uchovávaná v paměti

Sloupec „Třída podobnosti“ určuje index ve vstupním vektoru předzpracovaného vzoru. Sloupec „Kombinace k, l a m“ určují kombinaci hodnot k, l a m pro daný index. Hodnoty p_{ij} udávají počet vstupních trojúhelníků ze vzoru $\vec{t_i}$ pro danou kombinaci hodnot k, l a m. Velikost z (počet různých kombinací k, l a m) závisí na velikostech konstant přesnost pro třídu úhlu (c_a) a přesnost pro třídu obsahu (c_s).

4.1.4 Trénovací algoritmus

Současně s modelem byl navržen v [12] i algoritmus optimalizovaný pro rychlejší trénování sítě.

Průběh algoritmu je následující:

1. Inicializace všech prvků váhové matice $W = \{w_{ij}\}$ velikosti $z \times k$ nulovými hodnotami. Hodnota z je počet tříd podobnosti vstupních trojúhelníků z definice 4.1.2 a k je velikost výstupního vektoru.

2. Spočtení výstupu y_j j-tého neuronu ve výstupní vrstvě dle následujícího vzorce:

$$y_j = f(\sum_{i=1}^{z} w_{ij} p_{ki}),$$

kde i je index neuronu ve vstupní vrstvě, w_{ij} váha sítě mezi i-tým vstupním a j-tým výstupním neuronem a p_{ki} je předzpracovaná hodnota pro k-tý vzor na pozici i.

3. Adaptace vah podle následujících pravidel.

$$w_{ij} = \begin{cases} \pi(d_j - y_j) & \text{pokud } p_{ki} > 0 \\ 0 & \text{jinak} \end{cases}$$

kde d_j je očekávaný výstup sítě na pozici j, y_j je skutečný výstup sítě na pozici i a π je parametr učení.

4. Opakovat kroky 2 a 3, dokud nedojde k ukončení učení.

Algoritmus, jak jej popsali autoři v článku [12], představuje zjednodušenou verzi algoritmu Back propagation. Algoritmus není navržen pro neuronovou síť se skrytými vrstvami a nemá definovaný způsob ukončení. Pro náš výzkum jsme jako ukončovací podmínku použili dosažení předem definovaného počtu iterací. Dle našeho názoru bude učení pomocí tohoto algoritmu probíhat rychleji než pomocí Back propagation, protože bude docházet k větší změna vah při adaptaci.

4.2 Modifikace modelu

V následující kapitole modifikujeme původní model z [1] tak, aby dosahoval lepších výsledků při rozpoznávání znaků.

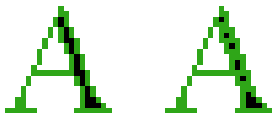
Prováděné úpravy lze rozdělit do dvou skupin. Úpravy týkající se předzpracování dat a úpravy týkající se definice aproximativně podobných trojúhelníků (viz kapitola 4.1.2).

4.2.1 Předzpracování dat

Všechny dříve uvedené metody modifikace sítě pro naučení určité transformace (kombinace transformací) využívají vstupní neurony vyššího řádu. Ze vztahu 3.1 je zřejmé, že ve vstupním vzoru jsou podstatné pouze nenulové hodnoty. Jakákoliv kombinace vstupních hodnot obsahující alespoň jednu nulovou hodnotu je díky součinu rovna nule. Při předzpracování dat lze veškeré nulové hodnoty ignorovat. V článku [1] autoři navrhli možnost ignorovat také všechny vstupní hodnoty (pixely) nesousedící s okrajem znaku. Síti by byly předkládány pouze obrysy znaků. Klasifikace obrysu znaku lidským okem dosahuje stejné úspěšnosti jako klasifikace

znaků celých. Tedy obrys znaku nese postačující informace pro jeho klasifikaci. Ignorování neokrajových pixelů může výrazně snížit počet vstupních trojúhelníků a tím i velikost vstupní sítě vrstvy. V [1] bohužel neuvedli přesnou definici, jak se určí pixel sousedící s okrajem znaku.

V našich experimentech prověříme možnost ignorování/neignorování neokrajových pixelů znaku a její dopad na úspěšnost rozpoznávání sítě. Obrázek 4.3 ilustruje všechny možnosti výběru relevantních pixelů pro ověřování úspěšnosti sítě, které budeme testovat. Zelenou barvou jsou označeny pixely vybrané k dalšímu zpracování. První obrázek představuje výběr pixelů znaku, které sousedí s okrajem znaku alespoň v jednom ze čtyř základních směrů. Jedná se o pixely s hodnotou 1, které sousedí pravým, levým, horním nebo dolním okrajem s pixelem s hodnotou 0. Druhý obrázek představuje výběr pixelů znaku sousedících s okrajem znaku v libovolném směru (tedy i diagonálním).



Obrázek 4.2: Výběr pixelů

4.2.2 Aproximativně podobné trojúhelníky

Zavedení aproximativně podobných trojúhelníků má negativní vliv na rozpoznávání s ohledem na změnu velikosti [1]. Pomocí námi navržených modifikací se pokusíme tuto negativní vlastnost zmírnit.

V našem prvním návrhu modifikujeme vztah 4.1.1. Nová definice podobnosti trojúhelníků *s relativním obsahem* je následující.

Definice 4.2.1. *Dva vstupní trojúhelníky T_i a T_j vzoru jsou si podobné (s relativním obsahem), pokud náleží do stejné třídy podobnosti určené kladnými čísly k, l a m. Hodnoty k, l a m pro trojúhelník T jsou dány následujícími nerovnostmi:*

$$(k-1)c_a \leq \alpha < kc_a$$
$$(l-1)c_a \leq \beta < lc_a$$
$$(m-1)(s/S_t) \leq S < m(s/S_t),$$

kde c_a je konstanta určující přesnost pro třídu úhlu, c_s je konstanta určující přesnost pro třídu obsahu, α je nejmenší vnitřní úhel trojúhelníka T, β je druhý nejmenší vnitřní úhel trojúhelníka T, S je obsah trojúhelníka T a S_t je obsah rozpoznávaného vzoru (tj. počet pixelů vzoru s hodnotou 1).

Zavedením hodnoty S_t do vztahu pro určení podobnosti trojúhelníků se bude obsah jednotlivých trojúhelníků započítávat relativně vzhledem k obsahu celého vzoru. Tedy při změně velikosti vzoru bude tento poměr zachován a bude možné dosáhnout vyšší úspěšnosti vzhledem k rozpoznávání s ohledem na změnu velikosti.

V naší druhé modifikaci jsme se rozhodli zcela vynechat vztah obsahu trojúhelníků k určení jeho podobnosti. Vynecháním posledního vztahu z definice 4.2.1 snížíme počet tříd podobnosti trojúhelníků. Tím se sníží velikost vstupní vrstvy sítě a urychlí se doba potřebná pro učení sítě.

4.3 Srovnání

Nyní se zaměříme na experimentální ověření modelu při rozpoznávání znaků s ohledem na invariance. Budeme postupovat obdobným způsobem jako u předešlého modelu (viz kapitola 3.3).

Pro každé rozlišení je provedeno více měření s různými hodnotami parametrů. Pro původní model *MHONN* se jedná o 56 různých kombinací parametrů a pro námi modifikovaný model *MHONN* jde o 112 různých kombinací (pro každou transformaci). V následujících podkapitolách jsou prezentovány pouze nejlepší výsledky pro dané rozlišení a testovanou transformaci. Kompletní výsledky měření se nacházejí na přiloženém CD (viz Příloha A).

Srovnání modelu *MHONN* s dříve zkoumaným modelem *POHONN* bude věnována samostatná kapitola 6. V následujících odstavcích se zaměříme pouze na vlastnosti modelu *MHONN* a srovnáme jej s námi modifikovanými variantami.

4.3.1 Úspěšnost s ohledem na rotaci

Trénovací množina je tvořena z 26 velkých písmen české abecedy (bez diakritiky) v rozlišeních 10x10, 18x18 a 30x30. Testovací množina je tvořena 286 vzory pro každé rozlišení, které vznikly otočením vzorů z trénovací množiny o 5°, 10°, 25°, 45°, 80°, 95°, 100°, 150°, 250° a 280°.

V tabulce 4.3 jsou zaznamenány nejúspěšnější konfigurace původního modelu *MHONN* pro rozpoznávání s ohledem na rotaci. Z výsledků je patrný růst úspěšnosti modelu s rostoucím rozlišením testovacích vzorů. Tento jev si vysvětlujeme tím, že pootočení zdeformuje vzor při nízkém rozlišení více než při vysokém. Pokud dojde k posunu pixelu při deformaci v rozlišení 10x10, posune se minimálně

o 1 pixel, což je $\frac{1}{10}$ velikosti. Při rozlišení 30x30 je minimální vzdálenost posunu $\frac{1}{30}$ velikosti, což umožní přesněji vytvořit deformovaný vzor. Na druhou stranu se nejúspěšnější sítě pro vyšší rozlišení nedokázaly naučit celou trénovací množinu. Nejúspěšnější konfigurace pro rozlišení 18x18, která se naučila kompletní trénovací množinu, dosáhla na testovací množině úspěšnosti pouhých $36,06\%$ (viz kompletní výsledky na přiloženém CD). Tato hodnota je o několik procent nižší než nejlepší výsledky při rozlišení 10x10.

Rozlišení vzorů	**Parametr učení**	**Míra úhlu**	**Míra obsahu**	**Počet iterací**	**Trénovací úspěšnost**	**Testovací úspěšnost**
10x10	0,3	0,5	6	$1x10^5$	100,00%	**41,61%**
10x10	0,3	1	1	$1x10^5$	100,00%	**40,56%**
10x10	0,3	15	3	$1x10^5$	100,00%	**40,56%**
18x18	0,3	10	20	$1x10^5$	96,15%	**48,25%**
18x18	0,3	20	6	$1x10^5$	92,31%	**46,50%**
18x18	0,3	10	15	$1x10^5$	96,15%	**44,76%**
30x30	0,3	20	15	$5x10^4$	100,00%	**67,83%**
30x30	0,3	20	20	$5x10^4$	92,31%	**60,84%**
30x30	0,3	15	20	$5x10^4$	100,00%	**59,79%**

Tabulka 4.3: Výsledky původního modelu *MHONN* - rotace

Tabulka 4.4 ukazuje výsledky námi modifikovaného modelu. Sloupec „Typ okraje“ ukazuje způsob výběru pixelů při předzpracování (kapitola 4.2.1). Hodnota „hr“ představuje první variantu - pixely sousedící s okrajem v některém ze čtyř základních směrů. Hodnota „hr, vr“ reprezentuje druhou variantu - pixel které mohou sousedit i diagonálně. Sloupec „Def. troj.“ udává varianty definici aproximativně podobných trojúhelníků (kapitola 4.2.2). Hodnota „ro“ značí první variantu (viz definice 4.2.1), hodnota „bo“ značí druhou variantu - vynechání obsahu z definice.

Výsledky pro rozlišení 10x10 jsou téměř shodné v porovnání s původním modelem. Pro rozlišení 18x18 a 30x30 dostáváme výsledky lepší téměř o 4% - 5%. Jak je vidět z výsledků, způsob výběru okrajových bodů při rozpoznávání s ohledem na rotaci přinesl pouze malé rozdíly ve výsledcích (kolem 1% až 2%). Nejúspěšnější konfigurace sítí pro každé rozlišení používali první variantu výběru pixelů. Modifikace definice aproximativně podobných trojúhelníků se projevila mnohem výrazněji. U rozlišení 10x10 jsou výsledky obdobné pro obě modifikace definice podobností trojúhelníků. U vyššího rozlišení nejúspěšnější konfigurace s definicí bez obsahu dosáhla úspěšnosti pouhých $27,6\%$ u 18x18 a $20,28\%$ u 30x30, což je nesrovnatelně horší výsledek (viz kompletní výsledky experimentů na přiloženém CD).

Rozliš. vzorů	Par. učení	Míra úhlu	Míra obsahu	Počet iterací	Typ okraje	Def. troj.	Trénovací úspěšnost	Testovací úspěšnost
10x10	0,3	1	15	$1x10^5$	hr	ro	100,00%	**41,26%**
10x10	0,3	1	-	$1x10^5$	hr, vr	bo	100,00%	**40,91%**
10x10	0,3	3	-	$1x10^5$	hr	bo	100,00%	**40,91%**
18x18	0,3	20	20	$1x10^5$	hr	ro	96,15%	**52,10%**
18x18	0,3	20	20	$1x10^5$	hr, vr	ro	96,15%	**50,70%**
18x18	0,3	20	15	$1x10^5$	hr	ro	96,15%	**50,35%**
30x30	0,3	20	15	$5x10^4$	hr	ro	100,00%	**73,08%**
30x30	0,3	20	15	$5x10^4$	hr, vr	ro	100,00%	**71,33%**
30x30	0,3	15	10	$5x10^4$	hr, vr	ro	100,00%	**69,58%**

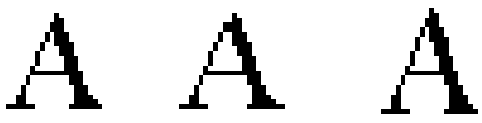
Tabulka 4.4: Výsledky námi modifikovaného modelu *MHONN* - rotace

4.3.2 Úspěšnost s ohledem na posunutí

Při rozpoznávání s ohledem na posunutí jsme došli ke stejným výsledků jako u předešlého modelu *POHONN*. Model *MHONN* dosahuje 100% úspěšnosti při klasifikaci posunutých vzorů. Předzpracování ze vzorů odstraňuje informace o přesné poloze pixelů, tedy síť je schopna rozpoznat každý naučený vzor bez ohledu na umístění ve vstupní matici.

4.3.3 Úspěšnost s ohledem na změnu velikosti

Úspěšnost s ohledem na změnu velikosti ověříme podobným způsobem jako v případě rotace. Testovací množina obsahuje vzory zvětšené (zmenšené) v rozmezí 5% - 30% a vzory zvětšené pouze v jednom směru o 10% (tj. rozšířené pouze vertikálně nebo pouze horizontálně - viz obrázek 4.3.3). Celkový počet testovacích vzorů je 364.



Obrázek 4.3: Znak „A“ a jeho deformace vzniklé zvětšením o 10% ve vertikálním a horizontálním směru

Úspěšnost původního modelu *MHONN* je zobrazena v tabulce 4.5. Námi získané výsledky potvrzují upozornění [1] o nízké úspěšnosti modelu při rozpoznávání s ohledem na změnu velikosti. Paradoxně vyšších úspěchů dosahuje model pro nižší rozlišení. Tento jev vysvětlujeme několika fakty:

- Již z výsledků rozpoznávání s ohledem na rotaci bylo patrné, že nejúspěšnější konfigurace modelů pracují s nízkou mírou pro určování úhlu. Tedy ve výsledku se síť naučí rozpoznávat především jedinečné úhly pro jednotlivé vzory. Při změně velikosti část těchto úhlů je změněna jen minimálně, díky čemuž je model pro nižší rozlišení úspěšnější.

- Nejúspěšnější konfigurace modelů pro vyšší rozlišení využívají vysokou míru úhlu a tím zpodobňují velké množství vstupních trojúhelníků. Síť se učí rozpoznávat místo jedinečných trojúhelníků spíše vzájemné poměry tříd podobnosti trojúhelníků. Při změně několika trojúhelníků dochází ke změně poměrů jednotlivých tříd podobností a tím snížení úspěšnosti.

Rozlišení vzorů	**Parametr učení**	**Míra úhlu**	**Míra obsahu**	**Počet iterací**	**Trénovací úspěšnost**	**Testovací úspěšnost**
10x10	0,3	1	1	$1x10^5$	100,00%	**23,08%**
10x10	0,3	2	3	$1x10^5$	100,00%	**23,08%**
10x10	0,3	2	15	$1x10^5$	100,00%	**23,08%**
18x18	0,3	2	1	$1x10^5$	100,00%	**16,41%**
18x18	0,3	2	3	$1x10^5$	100,00%	**15,90%**
18x18	0,3	3	1	$1x10^5$	100,00%	**15,90%**
30x30	0,3	0,5	3	$5x10^4$	100,00%	**19,49%**
30x30	0,3	0,5	1	$5x10^4$	100,00%	**18,46%**
30x30	0,3	2	15	$1x10^5$	100,00%	**16,92%**

Tabulka 4.5: Výsledky původního modelu *MHONN* - změna velikosti

Pro srovnání jsou v tabulce 4.6 zobrazeny výsledky námi modifikovaného modelu.

Rozliš. vzorů	**Par. učení**	**Míra úhlu**	**Míra obsahu**	**Počet iterací**	**Typ okraje**	**Def. troj.**	**Trénovací úspěšnost**	**Testovací úspěšnost**
10x10	0,3	20	25	$1x10^5$	hr, vr	ro	84,62%	**25,90%**
10x10	0,3	20	15	$1x10^5$	hr, vr	ro	96,15%	**25,64%**
10x10	0,3	20	20	$1x10^5$	hr, vr	ro	96,15%	**25,64%**
18x18	0,3	20	15	$1x10^5$	hr	ro	96,15%	**32,31%**
18x18	0,3	20	15	$1x10^5$	hr, vr	ro	96,15%	**32,31%**
18x18	0,3	20	25	$1x10^5$	hr	ro	96,15%	**30,00%**
30x30	0,3	20	15	$5x10^4$	hr	ro	100,00%	**49,74%**
30x30	0,3	15	15	$5x10^4$	hr, vr	ro	100,00%	**49,23%**
30x30	0,3	20	10	$5x10^4$	hr	ro	100,00%	**47,95%**

Tabulka 4.6: Výsledky námi modifikovaného modelu *MHONN* - změna velikosti

Výsledky pro rozlišení 10x10 se příliš neliší od výsledků původního modelu. Výrazného úspěchu dosahuje modifikovaný model u rozlišení 18x18 a 30x30. Dvojnásobné a vyšší úspěšnosti dosahuje modifikovaný model využívající novou definici aproximativně podobných trojúhelníků s relativním obsahem (4.2.1). Výsledky potvrzují, že modifikovaná definice podobnosti trojúhelníků výrazně zvyšuje úspěšnost celého modelu při rozpoznávání s ohledem na změnu velikosti vzoru. V naší definici jsou obsahy vstupních trojúhelníků započítávány relativně vzhledem k celkové velikosti znaku. Při rozpoznávání s ohledem na změnu velikosti proto nedochází k přiřazení jiné třídy obsahu jako u modelu původního.

4.3.4 Úspěšnost s ohledem na kombinace

Nyní srovnáme úspěšnosti modelů s ohledem na kombinaci deformací - rotace, posunutí a změna velikosti vzoru. Testovací množina pro každé rozlišení (10x10, 18x18 a 30x30) obsahuje 286 různých vzorů. Výsledky původního modelu *MHONN* jsou zobrazeny v tabulce 4.7.

Rozlišení vzorů	**Parametr učení**	**Míra úhlu**	**Míra obsahu**	**Počet iterací**	**Trénovací úspěšnost**	**Testovací úspěšnost**
10x10	0,3	20	6	$1x10^5$	92,31%	**22,03%**
10x10	0,3	10	10	$1x10^5$	88,46%	**21,68%**
10x10	0,3	20	1	$1x10^5$	100,00%	**21,68%**
18x18	0,3	10	20	$1x10^5$	96,15%	**25,87%**
18x18	0,3	10	10	$1x10^5$	100,00%	**25,17%**
18x18	0,3	15	10	$1x10^5$	96,15%	**24,83%**
30x30	0,3	10	25	$5x10^4$	96,15%	**27,62%**
30x30	0,3	15	10	$5x10^4$	100,00%	**27,62%**
30x30	0,3	15	15	$5x10^4$	100,00%	**27,62%**

Tabulka 4.7: Výsledky původního modelu *MHONN* - kombinace změn

Úspěšnosti se mírně zvyšují v závislosti na růstu rozlišení. Na konfiguracích nejúspěšnějších modelů pro nízká rozlišení je zajímavé, že všechny obsahují vysokou míru pro určení úhlu, ačkoliv v předešlých měřeních tomu bylo opačně. Většina nejúspěšnějších konfigurací modelů nebyla schopna se naučit kompletní trénovací množinu. Pro srovnání obsahuje tabulka 4.8 výsledky měření s námi modifikovaným modelem.

Výsledky potvrzují předešlá měření. Námi modifikované modely jsou schopny lépe rozpoznat vzory s ohledem na otočení i na změnu velikosti vzoru. Proto při rozpoznávání s ohledem na kombinace deformací dosahují výrazně lepších výsledků. Konfigurace úspěšných modelů podobně jako u původního modelu

Rozliš. vzorů	Par. učení	Míra úhlu	Míra obsahu	Počet iterací	Typ okraje	Def. troj.	Trénovací úspěšnost	Testovací úspěšnost
10x10	0,3	20	20	$1x10^5$	hr, vr	ro	96,15%	**24,48%**
10x10	0,3	20	20	$1x10^5$	hr	ro	96,15%	**23,78%**
10x10	0,3	20	15	$1x10^5$	hr, vr	ro	96,15%	**22,03%**
18x18	0,3	20	15	$1x10^5$	hr	ro	96,15%	**33,92%**
18x18	0,3	20	15	$1x10^5$	hr, vr	ro	96,15%	**33,92%**
18x18	0,3	20	20	$1x10^5$	hr	ro	96,15%	**32,87%**
30x30	0,3	20	25	$5x10^4$	hr	ro	96,15%	**50,70%**
30x30	0,3	20	15	$5x10^4$	hr	ro	100,00%	**49,65%**
30x30	0,3	20	25	$5x10^4$	hr, vr	ro	96,15%	**49,65%**

Tabulka 4.8: Výsledky námi modifikovaného modelu *MHONN* - kombince změn

MHONN obsahují vysokou míru pro určení úhlu a ve většině případů nejsou schopny se naučit kompletní trénovací množinu.

4.3.5 Vliv šumu

Poslední srovnání ověří vlastnosti modelu při rozpoznávání s ohledem na poškození vzoru. Do testovacích vzorů byl přidán náhodný šum v rozmezí 1% - 5%. Na testovací vzory nebyly aplikované žádné jiné deformace. Počet testovacích vzorů je 156. Výsledky pro původní model jsou shrnuty v tabulce 4.9.

Rozliš. vzorů	Parametr učení	Míra úhlu	Míra obsahu	Počet iterací	Trénovací úspěšnost	Testovací úspěšnost
10x10	0,3	0,5	3	$1x10^5$	100,00%	**41,67%**
10x10	0,3	0,5	10	$1x10^5$	100,00%	**39,10%**
10x10	0,3	0,5	25	$1x10^5$	100,00%	**38,46%**
18x18	0,3	1	20	$1x10^5$	100,00%	**30,77%**
18x18	0,3	1	6	$1x10^5$	100,00%	**29,49%**
18x18	0,3	1	1	$1x10^5$	100,00%	**28,85%**
30x30	0,3	0,5	3	$5x10^4$	100,00%	**28,21%**
30x30	0,3	1	3	$1x10^5$	100,00%	**26,28%**
30x30	0,3	0,5	6	$5x10^4$	100,00%	**25,00%**

Tabulka 4.9: Výsledky původního modelu *MHONN* - vliv šumu

Výsledky dokazují neschopnost modelu *MHONN* poradit si s poškozením vzoru. Pro vyšší rozlišení a tím i větší počet poškozených pixelů úspěšnost modelu výrazně klesá narozdíl od předešlých měření. Výsledky námi modifikovaného modelu jsou zobrazeny v tabulce 4.10.

Rozliš. vzorů	Par. učení	Míra úhlu	Míra obsahu	Počet iterací	Typ okraje	Def. troj.	Trénovací úspěšnost	Testovací úspěšnost
10x10	0,3	1	-	$1x10^5$	hr, vr	bo	100,00%	**35,26%**
10x10	0,3	1	25	$1x10^5$	hr	ro	100,00%	**35,26%**
10x10	0,3	3	20	$1x10^5$	hr, vr	ro	100,00%	**35,26%**
18x18	0,3	1	20	$1x10^5$	hr, vr	ro	100,00%	**28,21%**
18x18	0,3	1	10	$1x10^5$	hr	ro	100,00%	**26,92%**
18x18	0,3	1	3	$1x10^5$	hr	ro	100,00%	**26,28%**
30x30	0,3	1	25	$5x10^4$	hr	ro	100,00%	**23,72%**
30x30	0,3	1	15	$5x10^4$	hr	ro	100,00%	**22,44%**
30x30	0,3	2	3	$5x10^4$	hr, vr	ro	100,00%	**22,44%**

Tabulka 4.10: Výsledky námi modifikovaného modelu *MHONN* - vliv šumu

Naše modifikace snížila úspěšnost rozpoznávání s ohledem na poškození vzoru. Domníváme se, že tento jev je způsoben použitím obsahu vzoru (počtu černých pixelů) při určování podobnosti tříd. Vzhledem k charakteristice vzorů před deformací, kdy počet bílých pixelů je větší než počet černých pixelů, zvýší provedení deformace počet černých pixelů (dojde k více změnám bílé na černou než opačně). Tím dojde ke zvýšení obsahu vzoru (počtu černých pixelů) a odlišnému přiřazování tříd podobnosti pro vstupní trojúhelníky.

4.4 Shrnutí

Experimentální ověření potvrdilo, že model *MHONN* dosahuje relativně dobrých výsledků při rozpoznávání znaků s ohledem na různé deformace. Ve většině případů roste úspěšnost rozpoznávání vzhledem k velikosti rozlišení trénovacích a testovacích vzorů.

Námi navržené modifikace modelu dosahují téměř ve všech klíčových oblastech lepších výsledků. Tyto rozdíly se zvyšují vzhledem k růstu rozlišení vzorů (u rozlišení 30x30 dosahují až 30%). Modifikovaný model dosahuje nejvyššího zlepšení vůči původnímu modelu při rozpoznávání s ohledem na změnu velikosti vzoru a kombinaci deformací. K mírnému zhoršení dochází při rozpoznávání s ohledem na poškození vzoru. Zhoršení je pravděpodobně způsobeno charakterem rozpoznávaných vzorů, ve kterých se nachází více pixelů s hodnotou 0 než 1. Aplikování náhodného šumu proto zvýší počet pixelů s hodnotou 1 a tím pozmění přiřazenou třídu podobnosti pro velké množství vstupních trojúhelníků, protože třída podobnosti je závislá na počtu pixelů s hodnotou 1.

Kapitola 5

Adaptivní stromy

Článek [3] představuje nový typ neuronové sítě pro rozpoznávání vzorů. Jedná se o hierarchickou víceúrovňovou neuronovou síť. Autoři tuto síť nazvali Adaptive High Order Neural Tree (*AHONT*). Model *AHONT* vychází ze základních neuronových sítí, které jsou uspořádány do stromové topologie.

V průběhu učení dochází k rozšiřování stromové struktury v závislosti na úspěšnosti jednotlivých uzlů. Model *AHONT* současně využívá možností neuronových sítí vyššího řádu. Souběžně s růstem stromové struktury dochází ke zvyšování řádu jednotlivých uzlů.

Podle autorů [3] by měl model *AHONT* poskytnout širší možnosti učení, snížit dobu učení a zvýšit úspěšnost rozpoznávání.

5.1 Základní model

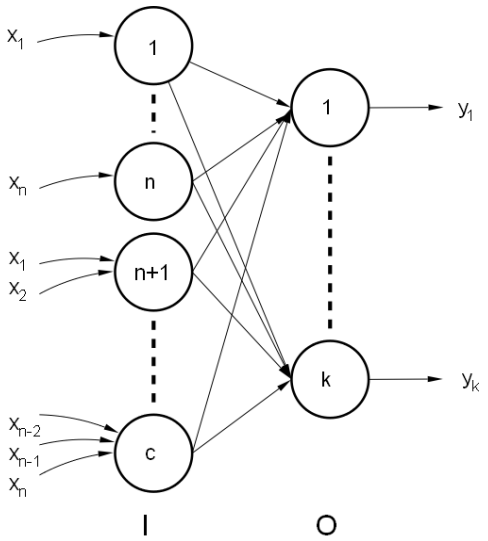
Model *AHONT* je tvořen neuronovou sítí se stromovou topologií. Jedná se o adaptivní model, kdy struktura sítě není předem dána a vytváří se během učení. Jednotlivé uzly jsou tvořeny tzv. *vícenásobnými neurony.*

Vícenásobný neuron řádu N je modifikovaná neuronová síť řádu N bez skryté vrstvy. Od základní neuronové sítě řádu N se liší tím, že vstupní kombinace nejsou tvořeny pouze N-ticemi hodnot ze vstupního vektoru, ale také (N-1)-ticemi, (N-2)-ticemi, ..., samotnými hodnotami (1-ticemi). Výstup y_k k-tého výstupního neuronu se spočte dle následujícího vztahu:

$$y_k = f(\sum_{j=1}^{n} w_{kj}x_j + \sum_{j=1}^{n}\sum_{i=1}^{n} w_{kji}x_j x_i + \sum_{j=1}^{n}\sum_{i=1}^{n}\sum_{l=1}^{n} w_{kjil}x_j x_i x_l + \dots), \qquad (5.1)$$

kde w_{kj}, w_{kji}, ... jsou vnitřní váhy neuronu, f je přenosová funkce, n je délka vstupního vektoru a x_i je i-tá hodnota vstupního vektoru $\vec{x}$.

Na obrázku 5.1 je možné vidět strukturu vícenásobného neuronu řádu tři. Do vstupní vrstvy vstupují jak jednotlivé hodnoty vstupního vektoru, tak také dvojice a trojice hodnot.



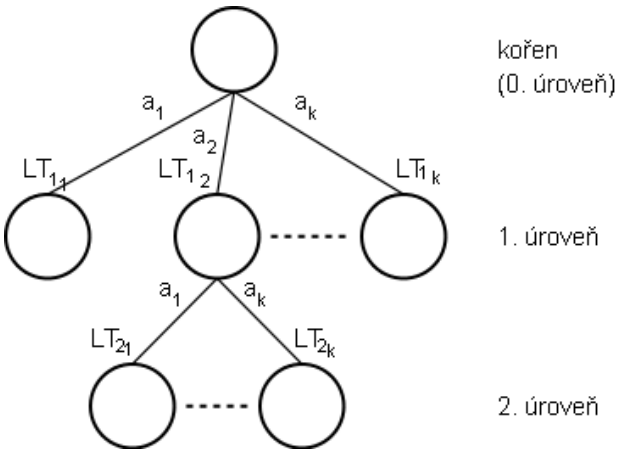
Obrázek 5.1: Vícenásobný neuron řádu tři

Vícenásobný neuron odpovídá celé neuronové síti. Model *AHONT* je tvořen stromem neuronových sítí. Tohoto faktu se pokusíme využít při modifikaci modelu *AHONT* za cílem dosažení vyšší úspěšnosti.

5.1.1 Stavba stromu

Model *AHONT* je tvořen stromovou strukturou s vícenásobnými neurony jako uzly. Při učení na trénovací množině může docházet k zvyšování řádu jednotlivých uzlů, případně k růstu samotné stromové struktury. Na obrázku 5.2 je vidět příklad adaptivního stromu. Jednotlivé úrovně stromu očíslujeme. Číslování začíná v kořeni - 0. úroveň. Každý uzel v adaptivním stromu obsahuje c vstupních neuronů a může být napojen až na k uzlů v nižší úrovni stromu. LT_{ij} je trénovací množina pro vícenásobný neuron v úrovni i na pozici j.

Pro učení sítě modelu *AHONT* je třeba definovat počáteční řád n_{min} a maximální řád n_{max} vícenásobných neuronů. Tyto konstanty udávají rozmezí pro zvyšování řádu jednotlivých uzlů. Typicky bývá n_{min} nastaven jako 1. Počáteční trénovací množinu označíme T a její velikost K.



Obrázek 5.2: Adaptivní strom

Průběh učení pro model *AHONT*:

1. Na počátku učení obsahuje model *AHONT* pouze jeden uzel (tzv. kořen) řádu n_{min}. Uzel je natrénován na množině T (Kapitola 5.1.2).

2. Pokud kořen dokáže všechny vzory z trénovací množiny správně rozpoznat, je učení ukončeno. V opačném případě učení pokračuje krokem 3.

3. Trénovací množina T je rozdělena do k disjunktních skupin ($1 < k \leq K$). Označme je jako lokální trénovací množiny $LT_1, \ldots, LT_k$. Vzory $\vec{x_1}, \ldots, \vec{x_K}$ z trénovací množiny T jsou rozřazeny do lokálních trénovacích množin $LT_1, \ldots, LT_k$ dle aktivačních hodnot (první pozice ve výstupním vektoru s nejvyšší hodnotou y_{max}). Vzor $\vec{x_i}$ je přiřazen množině LT_j, pokud y_j je nejvyšší aktivační hodnota ve výstupním vektoru. Pokud vznikla pouze jedna lokální trénovací množina, uzel se dále nezpracovává. V opačném případě je ve stromové struktuře vytvořen nový uzel pro každou neprázdnou lokální trénovací množinu.

4. Každý nový uzel je naučen na základě příslušné trénovací množiny. Pokud jsou všechny vzory z lokální trénovací množiny rozpoznány správně, uzel se dále nezpracovává. V opačném případě učení pokračuje krokem 3.

5.1.2 Učení vícenásobného neuronu

Vzhledem k rozdílnosti vícenásobného neuronu od standardního neuronu a neuronových sítí nelze použít přímo algoritmus Back propagation na jeho učení. Autoři [3] navrhli vlastní algoritmus, který využívá vlastnosti vícenásobného neuronu, především umožňuje růst vnitřního řádu. Algoritmus navržený [3] je následující:

1. Počáteční řád N vícenásobného neuronu je 1.

2. Chyba učení $E(N)$ vícenásobného neuronu pro řád N se spočte dle následujícího vztahu:

$$E(N) = \frac{1}{Kn} \sum_{k=i}^{K} \sum_{i=1}^{n} (d_i^k - y_i^k),$$

   kde d_i^k je požadovaný výstup pro k-tý trénovací vzor na pozici i, y_i^k je skutečný výstup i-tého neuronu ve výstupní vrstvě pro k-tý vzor.

3. Vícenásobný neuron řádu N je trénován, dokud nedojde ke stagnaci chyby. K té dojde, pokud po určitý počet iterací τ je chyba v rozmezí intervalu $< -er, er >$. Parametr er nabývá nízkých hodnot (např. 10^{-6}).

4. Pokud je vnitřní řád vícenásobného neuronu nižší než maximální řád n_{max} a buď $N = 1$ nebo $E(N) - E(N-1) < \Theta_{err}$, pak

   (a) řád vícenásobného neuronu je zvýšen na $N + 1$

   (b) přechod zpět ke Kroku 2.

5. Konec učení.

Hodnota prahu Θ_{err} nabývá dle [3] hodnot $< 10^{-4}; 10^{-1} >$. Algoritmus nedefinuje přesný způsob učení vícenásobného neuronu pro libovolný řád N. Vzhledem ke způsobu výpočtu chyby jednoho vícenásobného neuronu jsme se rozhodli pro učení vícenásobného neuronu pro libovolný řád N použít algoritmus Back propagation.

5.1.3 Klasifikace vzorů

Nyní zbývá popsat postup, jak rozvětvený strom modelu *AHONT* rozpoznává jednotlivé vzory. Postup pro klasifikaci vzoru je následující:

1. Vzor $\vec{x}$ je předložen kořenovému uzlu a posléze propagován směrem k listům stromu.

2. Po předložení vzoru vypočítá uzel (ve vrstvě l) příslušný výstupní vektor $\vec{y}$. Pozice první nejvyšší hodnoty ve výstupním vektoru $i_{y_{max}}$ určí odpovídající uzel ve vrstvě $l + 1$ a pokud takový uzel neexistuje, tak představuje $i_{y_{max}}$ hodnotu výstupu neuronového stromu pro vzor $\vec{x}$.

5.2 Modifikace modelu

Samotný model *AHONT* není navržen speciálně pro rozpoznávání znaků (či jiných objektů) vzhledem k různým invariancím. Model je navržen obecně pro libovolný problém. Dá se předpokládat, že jeho úspěšnost při rozpoznávání znaků s ohledem na invariance nebude příliš vysoká.

Na druhou stranu nabízí model *AHONT* široké možnosti pro rozšiřování. Jak jsme již zmínili dříve, návrh jednoho vícenásobného neuronu odpovídá dle našich definic vrstevnaté neuronové síti. Použitím jiných sítí jako základních stavebních prvků adaptivního stromu, je možné výrazně zvýšit úspěšnost modelu *AHONT*.

Z tohoto důvodu jsme se rozhodli modifikovat model *AHONT* takovým způsobem, aby mohl pracovat s jinými neuronovými sítěmi. Námi modifikovaný model *AHONT* bude jiné neuronové sítě organizovat do stromové struktury a zajišťovat předávání trénovacích vzorů a výstupů sítě. Algoritmus pro *AHONT* pozměníme minimálně. Jednotlivé uzly stromové struktury budou tvořeny modely *POHONN* a *MHONN*. Pro každou úroveň stromové struktury je možné nadefinovat jinou konfiguraci modelu *POHONN* nebo *MHONN*.

5.2.1 Modifikace algoritmu

Vzhledem k charakteru našeho výzkumu je možné, že algoritmus stavby stromu (viz kapitola 5.1.1) bude vytvářet nadbytečné uzly ve stromové struktuře. Například pokud kořenový uzel rozpozná správně 25 znaků z 26, v 1. úrovni stromu se vytvoří 25 uzlů (25 různých klasifikací) a k nim příslušné trénovací množiny. 24 z těchto trénovacích množin budou obsahovat pouze jeden trénovací vzor. 25. množina bude obsahovat vzory dva - špatně rozpoznaný vzor a správně rozpoznaný vzor, jehož klasifikace odpovídá chybné klasifikaci špatně rozpoznaného vzoru.

Pro odstranění těchto přebytečných uzlů jsme se rozhodli mezi krok 3 a krok 4 algoritmu pro stavbu stromu modelu vložit nový krok 3b:

3b Pokud jsou všechny vzory z lokální trénovací množiny rozpoznány správně (uzlem, který vytváří tuto lokální trénovací množinu), nový uzel pro tuto lokální trénovací množinu je ze stromu odstraněn.

Zařazení kroku 3b mezi krok 3 a krok 4 zajistí, že pokud uzel správně rozpozná některý vzor z trénovací množiny, tak tento vzor nevytvoří lokální trénovací množinu o jednom prvku. Tím dojde k celkovému urychlení algoritmu, protože nebude docházet k vytváření tolika uzlů stromu.

5.2.2 Spolupráce s ostatními modely

Použití modelů *POHONN* a *MHONN* umožní stavět adaptivní strom *AHONT* ze sítí specializovaných na rozpoznávání vzorů s ohledem na různé invariance. Na druhou stranu model připravíme o adaptivní růst vnitřních řádů jednotlivých uzlů, protože u modelů *POHONN* a *MHONN* nelze měnit vnitřní řády. Proto je důležité, které modely budeme do modelu *AHONT* zapojovat, aby došlo ke zvyšování úspěšnosti modelu.

Zapojením úspěšných konfigurací modelů *POHONN* nebo *MHONN* do adaptivního stromu nemusí vždy dojít ke zvýšení úspěšnosti. Příliš úspěšný uzel dokáže všechny vzory klasifikovat sám, nedojde k vytvoření lokálních trénovacích množin a vytvoření nových uzlů ve stromu. Pro výběr konfigurací vhodných do modifikovaného modelu *AHONT* použijeme výsledků našich experimentů, které jsme prezentovali v kapitolách 3.3 a 4.3.

Vzhledem ke znalostem trénovacích a testovacích množin předešlých modelů a výsledků jednotlivých konfigurací, můžeme vybrat vhodné kandidáty pro model *AHONT*. Zavedeme tzv. *koeficient úspěšnosti*, který bude udávat schopnost modelu (konfigurace) rozpoznat správně část trénovací množiny.

Definice 5.2.1. *Nechť Tr_e je procentuální úspěšnost rozpoznávání na trénovací množině a Te_e je procentuální úspěšnost rozpoznávání na testovací množině. Koeficient úspěšnosti q_e je definován jako*

$$q_e = \frac{Te_e}{Tr_e}$$

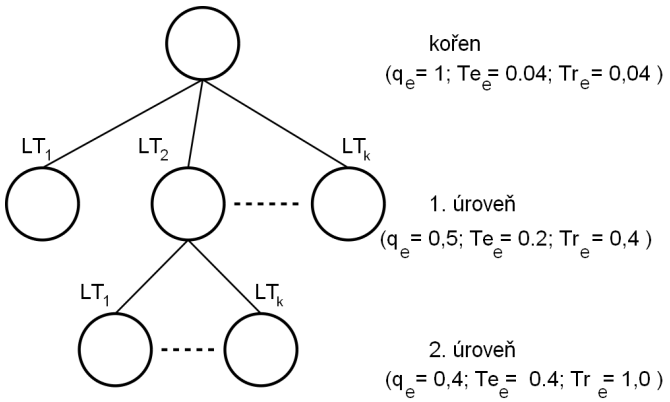
Námi navržený výběr nejvhodnějších konfigurací modelů *POHONN* nebo *MHONN* pro použití s modifikovaným modelem *AHONT* je založen na následujících vlastnostech:

- Vybraná konfigurace musí dosahovat vyššího koeficientu úspěšnosti než konfigurace s nejvyšší úspěšností.
- Vybrané konfigurace nesmí rozpoznat kompletní trénovací množinu.

Takto vybraná konfigurace se dokáže naučit část trénovací množiny a s vysokou úspěšností ji klasifikovat. Zbytek trénovací množiny předá dalším uzlů ve stromě k naučení. Pro dosažení maximální úspěšnosti použijeme pro jednotlivé úrovně adaptivního stromu různé konfigurace. Pro nejvyšší úrovně (úrovně 0, 1, ..) použijeme konfigurace s vysokým koeficientem úspěšnosti a nízkou úspěšností na trénovací množině. Pro ostatních vrstvy použijeme konfigurace s nejvyšší úspěšností

jak na trénovací tak na testovací množině. Důvody pro námi definovaný výběr demonstrujeme na příkladu.

Na obrázku 5.3 je vidět adaptivním strom s úrovněmi 0, 1 a 2, ve kterém je pro každou úroveň použita jiná konfigurace modelu. Ke každé konfiguraci jsou zobrazeny hodnoty - koeficient úspěšnosti q_e, procentuální úspěšnost na trénovací množině Tr_e a procentuální úspěšnost na testovací množině Te_e.



Obrázek 5.3: Výběr konfigurací pro adaptivní strom

Kořen rozpozná 4% ($Tr_e = 0,04$) trénovacích vzorů. Protože v trénovací i testovací množině je zachován poměr počtu vzorů jedné třídy k celkovému počtu vzorů, vyplývá z 4% úspěšnosti na testovací množině ($Te_e = 0,04$), že 4% vzorů jsou rozpoznány přibližně se 100% úspěšností ($q_e = 1$). Obdobně 1. úroveň rozpozná 40% vzorů ($Tr_e = 0,4$) z trénovací množiny až s 50% úspěšností ($q_e = 0,5$) a 2. úroveň rozpozná 100% vzorů ($Tr_e = 1$) až s 40% úspěšností ($q_e = 0,4$).

Nyní spočteme výslednou úspěšnost celého adaptivního stromu. 4% vzorů jsou rozpoznána až s 100% úspěšností, 20% až s 50% úspěšností a ostatní vzory až s 40% úspěšností. Maximální dosažená úspěšnost je až 42% ($0.04 * 1 + 0.2 * 0.5 + 0.76 * 0.4$), což je o 2% více než maximální úspěšnost jednotlivých úrovní.

Námi uvedený příklad ukazuje, že pro správně vybrané konfigurace modelů do uzlů stromu může dojít ke zvýšení úspěšnosti modelu *AHONT*.

5.3 Srovnání

Při ověřování modelu se zaměříme pouze na námi modifikovanou verzi modelu *AHONT*. Původní verze modelu není specializované na rozpoznávání znaků s ohledem na různé invariance a její výsledky jsou proto ve srovnání s ostatními modely nesrovnatelně horší (viz kompletní výsledky na přiloženém CD).

Ověřování modifikovaného modelu provedeme pouze s modelem *MHONN*. Model *POHONN* dosahuje ve srovnání s modelem *MHONN* výrazně horších výsledků a ani při maximálním možném zlepšení (viz kapitola 5.2.2) nemůže dosahovat lepších výsledků než model *MHONN*.

Pro každé rozlišení a transformaci je provedeno více měření s různými kombinacemi parametrů. Pro námi modifikovaný model *AHONT* ve spolupráci s původním modelem *MHONN* se jedná o 16 různých kombinací a pro námi modifikovaný model *AHONT* ve spolupráci s námi modifikovaným modelem *MHONN* jde o 49 různých kombinací. V následujících podkapitolách se nachází pouze nejlepší výsledky pro dané rozlišení a testovanou transformaci. Kompletní výsledky měření se nacházejí na přiloženém CD (viz Příloha A).

Ověřování provádíme stejným způsobem a na stejných datech jako ověřování předešlých modelů (kapitoly 3.3 a 4.3). Trénovací množina obsahuje 26 velkých písmen české abecedy bez diakritiky a testovací množiny obsahují vzory vzniklé aplikováním deformací na trénovací množinu. Ukázky testovacích a trénovacích vzorů je možné vidět v Příloze E.

5.3.1 Úspěšnost s ohledem na rotaci

Pro ověřování používáme stejné testovací a trénovací množiny jako u předešlých modelů. Testovací množinu tvoří 286 vzorů s různými stupni otočení (5° - 280°). Trénovací množina obsahuje 26 znaků.

V tabulce 5.1 jsou zobrazeny výsledky modifikovaného modelu *AHONT*, kde uzly tvoří původní modelem model *MHONN*, při rozpoznávání vzorů s ohledem na rotaci.

Ve srovnání s výsledky původního modelu *MHONN* došlo ve všech rozlišení ke zlepšení o 2-4%. Pro rozlišení 10x10 dosahuje model *AHONT* s původním modelem *MHONN* vyšší úspěšnosti než modifikovaný model *MHONN*. Pro ostatní rozlišení jsou výsledky o 1-2% horší.

Úspěšnosti modelu *AHONT* ve spolupráci s námi modifikovaným modelem *MHONN* lze vidět v tabulce 5.2. Dosažená úspěšnost pro rozlišení 10x10 odpovídá úspěšnosti samostatného modifikovaného modelu *MHONN*. U rozlišení 18x18 a 30x30 již došlo ke zlepšení o více než 1%. Ačkoliv je dosažené zlepšení relativně malé, kombinace modelu *AHONT* s modifikovaným modelem *MHONN* dosahuje nejvyšších úspěšností ze všech zkoumaných modelů při rozpoznávání s ohledem na otočení (pro rozlišení 18x18 a 30x30).

Rozliš. vzoru	Úroveň	Model uzlů	Param. učení	Míra úhlu	Míra obsahu	Počet iterací	Trénovací úspěšnost	Testovací úspěšnost
10x10	0.	původní	0,3	15,0	15,0	$5x10^5$	100,00%	**44,06%**
	1.	*MHONN*	0,3	15,0	3,0			
10x10	0.	původní	0,3	20,0	15,0	$5x10^5$	100,00%	**41,96%**
	1.	*MHONN*	0,3	2,0	3,0			
10x10	0.	původní	0,3	15,0	15,0	$5x10^5$	100,00%	**41,61%**
	1.	*MHONN*	0,3	1,0	1,0			
18x18	0.	původní	0,3	15,0	20,0	$5x10^5$	96,15%	**51,05%**
	1.	*MHONN*	0,3	15,0	3,0			
18x18	0.	původní	0,3	10,0	25,0	$5x10^5$	100,00%	**48,25%**
	1.	*MHONN*	0,3	1,0	10,0			
18x18	0.	původní	0,3	10,0	25,0	$5x10^5$	96,15%	**47,90%**
	1.	*MHONN*	0,3	15,0	10,0			
30x30	0.	původní	0,3	20,0	15,0	$5x10^5$	100,00%	**71,33%**
	1.	*MHONN*	0,3	1,0	1,0			
30x30	0.	původní	0,3	20,0	15,0	$5x10^5$	100,00%	**70,63%**
	1.	*MHONN*	0,3	2,0	3,0			
30x30	0.	původní	0,3	15,0	15,0	$5x10^5$	100,00%	**70,28%**
	1.	*MHONN*	0,3	15,0	3,0			

Tabulka 5.1: Výsledky námi modifikovaného modelu *AHONT* (s původním modelem *MHONN*) - rotace

Rozliš. vzoru	Úroveň	Model uzlů	Par. učení	Míra úhlu	Míra obsahu	Typ okraje	Def. troj.	Počet iterací	Trénovací úspěšnost	Testovací úspěšnost
10x10	0.	modif.	0,3	6,0	-	hr, vr	bo	$3x10^5$	100,00%	**41,61%**
	1.	*MHONN*	0,3	15,0	15,0	hr	ro			
10x10	0.	modif.	0,3	20,0	20,0	hr, vr	ro	$2x10^5$	100,00%	**41,26%**
	1.	*MHONN*	0,3	3,0	-	hr	bo			
10x10	0.	modif.	0,3	6,0	-	hr	bo	$3x10^5$	100,00%	**40,91%**
	1.	*MHONN*	0,3	20,0	10,0	hr, vr	ro			
18x18	0.	modif.	0,3	20,0	20,0	hr	ro	$2x10^5$	100,00%	**53,15%**
	1.	*MHONN*	0,3	15,0	20,0	hr	ro			
18x18	0.	modif.	0,3	20,0	15,0	hr	ro	$2x10^5$	100,00%	**51,05%**
	1.	*MHONN*	0,3	15,0	15,0	hr	ro			
18x18	0.	modif.	0,3	15,0	-	hr, vr	bo	$2x10^5$	100,00%	**50,35%**
	1.	*MHONN*	0,3	1,0	1,0	hr	ro			
30x30	0.	modif.	0,3	20,0	15,0	hr	ro	$2x10^5$	100,00%	**74,13%**
	1.	*MHONN*	0,3	15,0	15,0	hr	ro			
30x30	0.	modif.	0,3	20,0	15,0	hr	ro	$2x10^5$	100,00%	**70,28%**
	1.	*MHONN*	0,3	20,0	10,0	hr, vr	ro			
30x30	0.	modif.	0,3	6,0	-	hr	bo	$2x10^5$	100,00%	**69,58%**
	1.	*MHONN*	0,3	20,0	15,0	hr, vr	ro			

Tabulka 5.2: Výsledky námi modifikovaného modelu *AHONT* (s námi modifikovaným modelem *MHONN*) - rotace

5.3.2 Úspěšnost s ohledem na posunutí

Rozpoznávání s ohledem na posunutí jsme neprováděli, protože jak původní tak modifikovaný model *MHONN* dosahují stoprocentních úspěšností při rozpoznávání. Tato úspěšnost zůstane zachována i při kombinaci s modelem *AHONT*.

5.3.3 Úspěšnost s ohledem na změnu velikosti

Změnu velikosti ověřujeme na stejných datech jako u předešlých modelů. Pro každé rozlišení se trénovací množina skládá z 26 znaků (velká písmena české abecedy bez diakritiky). Testovací množina obsahuje 364 vzorů s různými stupni zvětšení a zmenšení.

Při rozpoznávání s ohledem na změnu velikosti dosahuje původní model *MHONN* nejhorších výsledků. Úspěšnosti při kombinaci s modelem *AHONT* jsou zachyceny v tabulce 5.3.

Rozliš. vzoru	**Úroveň**	**Model uzlů**	**Param. učení**	**Míra úhlu**	**Míra obsahu**	**Počet iterací**	**Trénovací úspěšnost**	**Testovací úspěšnost**
10x10	0.	původní	0,3	20,0	15,0	$5x10^5$	100,00%	**24,62%**
	1.	*MHONN*	0,3	1,0	1,0			
10x10	0.	původní	0,3	15,0	15,0	$5x10^5$	100,00%	**24,62%**
	1.	*MHONN*	0,3	1,0	1,0			
10x10	0.	původní	0,3	15,0	15,0	$5x10^5$	100,00%	**24,10%**
	1.	*MHONN*	0,3	15,0	3,0			
18x18	0.	původní	0,3	20,0	25,0	$9x10^5$	100,00%	**17,44%**
	1.	*MHONN*	0,3	0,5	10,0			
18x18	0.	původní	0,3	20,0	25,0	$9x10^5$	100,00%	**16,67%**
	1.	*MHONN*	0,3	1,0	1,0			
18x18	0.	původní	0,3	10,0	25,0	$5x10^5$	100,00%	**15,90%**
	1.	*MHONN*	0,3	1,0	10,0			
30x30	0.	původní	0,3	20,0	20,0	$3x10^5$	100,00%	**16,67%**
	1.	*MHONN*	0,3	0,5	10,0			
30x30	0.	původní	0,3	20,0	20,0	$3x10^5$	100,00%	**15,90%**
	1.	*MHONN*	0,3	0,5	3,0			
30x30	0.	původní	0,3	20,0	25,0	$3x10^5$	100,00%	**15,90%**
	1.	*MHONN*	0,3	1,0	1,0			

Tabulka 5.3: Výsledky námi modifikovaného modelu *AHONT* (s původním modelem *MHONN*) - změna velikosti

Výsledné úspěšnosti jsou opět relativně malé. Pro rozlišení 10x10 a 18x18 došlo ke zlepšení. U rozlišení 30x30 jsou však naměřené úspěšnosti horší přibližně o 3%. Domníváme se, že příčinou neúspěchu pro rozlišení 30x30 je rozložení správně kla-

sifikovaných vzorů v testovací množině. Konfigurace z 0. i 1. úrovně rozpoznávají nejúspěšněji tytéž vzory, ale konfigurace v 0. úrovni s nižší úspěšností. Přesto má vyšší koeficient úspěšnosti, protože dokáže rozpoznat pouze tyto „jednoduché“ vzory. Bohužel výsledkem takovéto kombinace konfigurací je snížení úspěšnosti modelu.

Výsledky kombinace námi modifikovaného modelu *AHONT* s námi modifikovaným modelem *MHONN* jsou zobrazeny v tabulce 5.4. Výsledky pro všechna rozlišení jsou lepší než u samotného modifikovaného modelu *MHONN*. K největšímu zlepšení došlo u rozlišení 10x10, kde jsou výsledky lepší o 3%. Vzhledem k možnostem modelu *AHONT* nemůžeme lepší výsledek ani očekávat. Pro rozlišení 18x18 činí zlepšení přibližně 1,5% a u rozlišení 30x30 došlo k zlepšení o méně než 1%. Ačkoliv zlepšení modelu *AHONT* v kombinaci s modifikovaným modelem *MHONN* je relativně nízké, přesto se jedná o doposud nejúspěšnější model pro rozpoznávání vzorů s ohledem na změnu velikosti.

Rozliš. vzoru	**Úroveň**	**Model uzlů**	**Par. učení**	**Míra úhlu**	**Míra obsahu**	**Typ okraje**	**Def. troj.**	**Počet iterací**	**Trénovací úspěšnost**	**Testovací úspěšnost**
10x10	0.	modif.	0,3	20,0	25,0	hr	ro	$5x10^5$	96,15%	**28,97%**
	1.	*MHONN*	0,3	10,0	15,0	hr	ro			
10x10	0.	modif.	0,3	20,0	25,0	hr, vr	ro	$5x10^5$	100,00%	**28,72%**
	1.	*MHONN*	0,3	15,0	15,0	hr	ro			
10x10	0.	modif.	0,3	20,0	25,0	hr, vr	ro	$5x10^5$	100,00%	**27,69%**
	1.	*MHONN*	0,3	1,0	20,0	hr, vr	ro			
18x18	0.	modif.	0,3	20,0	15,0	hr	ro	$2x10^5$	100,00%	**33,59%**
	1.	*MHONN*	0,3	15,0	15,0	hr	ro			
18x18	0.	modif.	0,3	20,0	25,0	hr, vr	ro	$2x10^5$	100,00%	**31,54%**
	1.	*MHONN*	0,3	15,0	15,0	hr	ro			
18x18	0.	modif.	0,3	20,0	15,0	hr	ro	$2x10^5$	100,00%	**31,54%**
	1.	*MHONN*	0,3	20,0	10,0	hr, vr	ro			
30x30	0.	modif.	0,3	20,0	15,0	hr	ro	$1x10^5$	100,00%	**50,00%**
	1.	*MHONN*	0,3	20,0	10,0	hr, vr	ro			
30x30	0.	modif.	0,3	6,0	-	hr	bo	$2x10^5$	100,00%	**49,23%**
	1.	*MHONN*	0,3	20,0	15,0	hr, vr	ro			
30x30	0.	modif.	0,3	20,0	15,0	hr	ro	$1x10^5$	100,00%	**49,23%**
	1.	*MHONN*	0,3	15,0	15,0	hr	ro			

Tabulka 5.4: Výsledky námi modifikovaného modelu *AHONT* (s námi modifikovaným modelem *MHONN*) - změna velikosti

5.3.4 Úspěšnost s ohledem na kombinace

Trénovací množina zůstává stejná jako u předešlého měření - 26 znaků v rozlišení 10x10, 18x18 a 30x30. Testovací množina je pro každé rozlišení složena z 260

vzorů, které podstoupily kombinace všech předešlých deformací.

V tabulce 5.5 je možné vidět výsledky námi modifikovaného modelu *AHONT* v kombinaci s původní verzí modelu *MHONN*. U výsledků pro rozlišení 10x10 došlo ke zlepšení asi o 1% ve srovnání s původním modelem *MHONN*. Pro vyšší rozlišení došlo ke zhoršení v řádech desetin procent. Příčina zhoršení je dle našeho mínění stejná jako u výsledku rozpoznávání s ohledem na změnu velikosti. Přestože má konfigurace modelu v kořeni vyšší koeficient úspěšnosti, dosahuje na vzorech, které dokáže klasifikovat, horších výsledků než konfigurace modelu v 1. úrovni.

Rozliš. vzoru	**Úroveň**	**Model uzlů**	**Param. učení**	**Míra úhlu**	**Míra obsahu**	**Počet iterací**	**Trénovací úspěšnost**	**Testovací úspěšnost**
10x10	0.	původní	0,3	20,0	15,0	$5x10^5$	100,00%	**23,08%**
	1.	*MHONN*	0,3	1,0	1,0			
10x10	0.	původní	0,3	15,0	15,0	$5x10^5$	100,00%	**22,73%**
	1.	*MHONN*	0,3	15,0	3,0			
10x10	0.	původní	0,3	20,0	15,0	$5x10^5$	100,00%	**22,73%**
	1.	*MHONN*	0,3	2,0	1,0			
18x18	0.	původní	0,3	20,0	25,0	$9x10^5$	100,00%	**25,57%**
	1.	*MHONN*	0,3	0,5	10,0			
18x18	0.	původní	0,3	15,0	20,0	$5x10^5$	96,15%	**25,52%**
	1.	*MHONN*	0,3	15,0	3,0			
18x18	0.	původní	0,3	20,0	25,0	$9x10^5$	100,00%	**25,52%**
	1.	*MHONN*	0,3	1,0	1,0			
30x30	0.	původní	0,3	20,0	15,0	$1x10^5$	100,00%	**27,27%**
	1.	*MHONN*	0,3	2,0	3,0			
30x30	0.	původní	0,3	20,0	15,0	$1x10^5$	100,00%	**26,92%**
	1.	*MHONN*	0,3	1,0	1,0			
30x30	0.	původní	0,3	20,0	20,0	$3x10^5$	100,00%	**26,92%**
	1.	*MHONN*	0,3	0,5	3,0			

Tabulka 5.5: Výsledky námi modifikovaného modelu *AHONT* (s původním modelem *MHONN*) - kombinace změn

Zajímavější jsou výsledky modelu *AHONT* s námi modifikovaným modelem *MHONN* (tabulka 5.6). Naměřené výsledky jsou pro všechna rozlišení lepší přibližně o 1% než u modifikovaného modelu *MHONN*. Toto zvýšení úspěšnosti odpovídá dle našeho odhadu předpokládanému zlepšení, které je dosažitelné pro koeficienty úspěšnosti modifikovaného modelu *MHONN*. Výsledky potvrzují, že kombinace modelu *AHONNT* s modifikovaným modelem *MHONN* dosahuje nejlepších výsledků ze všech zkoumaných modelů.

Rozliš. vzoru	Úroveň	Model uzlů	Par. učení	Míra úhlu	Míra obsahu	Typ okraje	Def. troj.	Počet iterací	Trénovací úspěšnost	Testovací úspěšnost
10x10	0.	modif.	0,3	20,0	25,0	hr, vr	ro	$5x10^5$	100,00%	**25,17%**
	1.	*MHONN*	0,3	15,0	15,0	hr	ro			
10x10	0.	modif.	0,3	20,0	25,0	hr, vr	ro	$5x10^5$	100,00%	**24,83%**
	1.	*MHONN*	0,3	1,0	20,0	hr, vr	ro			
10x10	0.	modif.	0,3	20,0	20,0	hr, vr	ro	$2x10^5$	100,00%	**24,48%**
	1.	*MHONN*	0,3	3,0	-	hr	bo			
18x18	0.	modif.	0,3	20,0	15,0	hr	ro	$2x10^5$	100,00%	**34,97%**
	1.	*MHONN*	0,3	20,0	10,0	hr, vr	ro			
18x18	0.	modif.	0,3	20,0	15,0	hr	ro	$2x10^5$	100,00%	**34,62%**
	1.	*MHONN*	0,3	15,0	15,0	hr	ro			
18x18	0.	modif.	0,3	20,0	15,0	hr	ro	$2x10^5$	100,00%	**33,57%**
	1.	*MHONN*	0,3	15,0	20,0	hr	ro			
30x30	0.	modif.	0,3	20,0	25,0	hr	ro	$2x10^5$	100,00%	**51,75%**
	1.	*MHONN*	0,3	20,0	15,0	hr	ro			
30x30	0.	modif.	0,3	20,0	25,0	hr, vr	ro	$2x10^5$	100,00%	**50,35%**
	1.	*MHONN*	0,3	20,0	15,0	hr	ro			
30x30	0.	modif.	0,3	20,0	25,0	hr, vr	ro	$2x10^5$	100,00%	**50,00%**
	1.	*MHONN*	0,3	1,0	20,0	hr, vr	ro			

Tabulka 5.6: Výsledky námi modifikovaného modelu *AHONT* (s námi modifikovaným modele *MHONN*) - kombinace změn

5.3.5 Vliv šumu

Na závěr ověříme vlastnosti modelu při rozpoznávání poškozených vzorů. Trénovací množina je stejná jako u předešlých měření. Testovací množina je pro každé rozlišení tvořena 156 vzory, které vznikly z trénovací množiny přidáním náhodného šumu v rozmezí 1% - 5%. Hodnota každého pixelu testovacího vzoru byla přepsána na opačnou s danou pravděpodobností.

Výsledky původního modelu *MHONN* byly relativně nízké. Výsledky modelu *AHONT* v kombinaci s původním modelem *MHONN* jsou shrnuty v tabulce 5.7. Bohužel výsledky modelu *AHONT* jsou výrazně horší, v některých případech došlo ke snížení úspěšnosti až o 8%. Je pravděpodobné, že původní model *MHONN* dokáže správně klasifikovat jen malý počet vzorů po poškození. Tyto vzory, s nějakým charakteristickým rysem, však dokáže rozpoznat s vysokou úspěšností. Námi nadefinovaný keoficient úspěšnosti bohužel upřednostnil do kořene konfiguraci modelu, která na těchto vzorech dosahovala nízké úspěšnosti. Proto celkově úspěšnější konfigurace v úrovni 1 již nedokázala zvýšit celkovou úspěšnost.

Rozliš. vzoru	Úroveň	Model uzlů	Param. učení	Míra úhlu	Míra obsahu	Počet iterací	Trénovací úspěšnost	Testovací úspěšnost
10x10	0.	původní	0,3	20,0	20,0	$4x10^5$	100,00%	**38,46%**
	1.	*MHONN*	0,3	0,5	3,0			
10x10	0.	původní	0,3	15,0	15,0	$5x10^5$	100,00%	**34,62%**
	1.	*MHONN*	0,3	15,0	3,0			
10x10	0.	původní	0,3	20,0	20,0	$4x10^5$	100,00%	**34,62%**
	1.	*MHONN*	0,3	0,5	10,0			
18x18	0.	původní	0,3	20,0	15,0	$5x10^5$	100,00%	**24,36%**
	1.	*MHONN*	0,3	1,0	1,0			
18x18	0.	původní	0,3	20,0	15,0	$5x10^5$	100,00%	**24,36%**
	1.	*MHONN*	0,3	2,0	3,0			
18x18	0.	původní	0,3	20,0	20,0	$5x10^5$	100,00%	**23,72%**
	1.	*MHONN*	0,3	0,5	3,0			
30x30	0.	původní	0,3	20,0	15,0	$1x10^5$	100,00%	**20,51%**
	1.	*MHONN*	0,3	1,0	1,0			
30x30	0.	původní	0,3	15,0	20,0	$1x10^5$	100,00%	**20,51%**
	1.	*MHONN*	0,3	20,0	6,0			
30x30	0.	původní	0,3	20,0	20,0	$3x10^5$	100,00%	**20,51%**
	1.	*MHONN*	0,3	0,5	3,0			

Tabulka 5.7: Výsledky námi modifikovaného modelu *AHONT* (s původním modelem *MHONN*) - vliv šumu

Kombinace s modifikovaným modelem *MHONN* (tabulka 5.8) také přinesla ve většině případů snížení úspěšnosti. Pro rozlišení 10x10 zůstávají hodnoty podobné jako u modifikovaného modelu *MHONN*. Pro rozlišení 18x18 a 30x30 dochází ke snížení přibližně o 2%. Předpokládáme, že příčina snížení úspěšnosti je stejná jako u kombinace s původním modelem *MHONN*.

5.4 Shrnutí

Experimentálním ověřením jsme dospěli k závěru, že model *AHONT* bez jakékoliv úpravy dosahuje ve srovnání s ostatními modely velice nízké úspěšnosti a není schopen rozpoznávat vzory s ohledem na různé kombinace.

Na druhou stranu lze model *AHONT* jednoduše modifikovat a v kombinaci se specializovanými modely pak dosahuje relativně dobrých výsledků. Při kombinacích s původním modelem *MHONN* dochází v některých případech ke zvýšení úspěšnosti. Jedná se především o rozpoznávání s ohledem na rotaci a změnu velikosti. Při rozpoznávání s ohledem na kombinaci změn zůstávají výsledky podobné. K výraznému snížení úspěšnosti dochází při rozpoznávání poškozených vzorů.

Kombinace s modifikovaným modelem *MHONN* přináší téměř ve všech měře-

Rozliš. vzoru	Úro-veň	Model uzlů	Par. učení	Míra úhlu	Míra obsahu	Typ okraje	Def. troj.	Počet iterací	Trénovací úspěšnost	Testovací úspěšnost
10x10	0.	modif.	0,3	20,0	-	hr	bo	$3x10^5$	100,00%	**35,26%**
	1.	*MHONN*	0,3	1,0	15,0	hr	ro			
10x10	0.	modif.	0,3	15,0	-	hr	bo	$2x10^5$	100,00%	**33,97%**
	1.	*MHONN*	0,3	2,0	-	hr	bo			
10x10	0.	modif.	0,3	6,0	-	hr	bo	$3x10^5$	100,00%	**33,78%**
	1.	*MHONN*	0,3	20,0	15,0	hr, vr	ro			
18x18	0.	modif.	0,3	15,0	-	hr, vr	bo	$2x10^5$	100,00%	**26,28%**
	1.	*MHONN*	0,3	1,0	-	hr, vr	bo			
18x18	0.	modif.	0,3	10,0	-	hr	bo	$2x10^5$	100,00%	**25,64%**
	1.	*MHONN*	0,3	1,0	15,0	hr	ro			
18x18	0.	modif.	0,3	20,0	-	hr	bo	$2x10^5$	100,00%	**25,00%**
	1.	*MHONN*	0,3	1,0	20,0	hr	ro			
30x30	0.	modif.	0,3	3,0	-	hr, vr	ro	$2x10^5$	100,00%	**20,51%**
	1.	*MHONN*	0,3	20,0	10,0	hr	ro			
30x30	0.	modif.	0,3	20,0	-	hr	ro	$2x10^5$	100,00%	**20,51%**
	1.	*MHONN*	0,3	20,0	10,0	hr	ro			
30x30	0.	modif.	0,3	6,0	-	hr	ro	$2x10^5$	100,00%	**20,51%**
	1.	*MHONN*	0,3	20,0	15,0	hr, vr	ro			

Tabulka 5.8: Výsledky námi modifikovaného modelu *AHONT* (s námi modifikovaným modelem *MHONN*) - vliv šumu

ných kategoriích zlepšení. Pro některá rozlišení jde o zlepšení až o 3%. Ke zhoršení dochází pouze při rozpoznávání poškozených vzorů. Z naměřených výsledků je patrné, že kombinace modelu *AHONT* s modifikovaným modelem *MHONN* dosahuje nejlepších výsledků ze všech zkoumaných modelů.

Příčinou snížení úspěšnosti v některých oblastech rozpoznávání je nerovnoměrné rozložení správně klasifikovaných vzorů z testovací množiny. Některé konfigurace dosahují pro několik specifických vzorů nižší úspěšnosti, ačkoliv jejich koeficient úspěšnosti je vyšší. Tento problém nastává převážně v kombinaci s původním modelem *MHONN*.

Kapitola 6

Srovnání

Následující kapitola je věnována porovnání všech námi zkoumaných modelů. Zhodnotíme významné vlastnosti jednotlivých modelů a vzájemně je porovnáme. Jejich vlastnosti a úspěšnosti porovnáme také s námi navrženými modifikacemi modelů.

6.1 Charakteristiky modelů

Postupně se budeme věnovat jednotlivým modelům a budeme je porovnávat s ostatními. Zaměříme se úspěšnosti modelů při rozpoznávání znaků s ohledem na jednotlivé deformace (otočení, změna velikosti vzoru, ...).

6.1.1 POHONN

Model *POHONN* je historicky nejstarším ověřovaným modelem, čemuž odpovídají nejnižší výsledky ve srovnání s ostatními modely. Model dosahuje nízké úspěšnosti při rozpoznávání znaků ve vyšších rozlišeních (18x18 a 30x30). Narozdíl od ostatních modelů, kde dochází u vyšších rozlišeních k výraznému růstu úspěšnosti při rozpoznávání, model *POHONN* dosahuje v mnoha případech dokonce horších výsledků než u rozlišení 10x10. Výhodou tohoto modelu je, že se dokáže relativně dobře vypořádat s poškozenými vzory.

Nízkou dosaženou úspěšnost pro vyšší rozlišení si vysvětlujeme ignorováním velikosti vstupních trojúhelníků při předzpracování vzoru. Protože nebyla zohledňována velikost vstupních trojúhelníků, dochází u vzorů s větším rozlišením ke splynutí mnoha vstupních trojúhelníků, které se liší pouze velikostí. Pravděpodobně díky tomuto jevu dochází ke ztrátě klíčových informací, které umožní

identifikovat vzor správně.

Navržením našich modifikací jsme dosáhli nepatrného zvýšení úspěšnosti při rozpoznávání vzorů s rozlišením 10x10 (kolem 2%). Pro vyšší rozlišení je nárůst úspěšnosti výraznější (5%-30%). Negativně se projevily námi navržené modifikace při rozpoznávání poškozených vzorů, kde došlo k snížení úspěšnosti až o 20%.

Nejvýraznějším úspěchem námi modifikované verze modelu je právě nárůst úspěšností pro vyšší rozlišení. Přidání informace o velikosti vstupního trojúhelníku ke vstupním údajům sítě umožňuje lépe rozlišit charakteristické rysy jednotlivých znaků a zvyšuje dosahované úspěšnosti. Tato vlastnost se však stává negativní při rozpoznávání poškozených vzorů. Přidání šumu do rozpoznávaných vzorů odstraní mnoho starých a přidá mnoho nových trojúhelníků. V původní verzi modelu dochází mezi mnohými odstraněnými a novými trojúhelníky ke splynutí, protože se liší pouze velikostí. Bohužel u námi modifikované verze nedochází ke splývání nových a starých trojúhelníků a výsledná úspěšnost je proto výrazně nižší.

6.1.2 MHONN

Model *MHONN* je historicky mladší model ve srovnání s modelem *POHONN*. Jeho základ vychází z podobných principů jako model *POHONN*, avšak model *MHONN* odstraňuje některé slabé stránky modelu *POHONN*. Především je navržen pro práci se vzory ve vyšších rozlišeních a snaží se minimalizovat paměťové nároky bez ovlivnění výkonnosti modelu.

Řady testů s různými konfiguracemi modelu *MHONN* potvrdily rychlejší práci se vzory ve vyšších rozlišeních i snížení paměťových nároků. Ve srovnání s modelem *POHONN* dochází ve většině případů ke zvyšování úspěšnosti s růstem rozlišení testovacích vzorů. K drobnému snížení úspěšnosti ve srovnání s modelem *POHONN* dochází při rozpoznávání vzorů po změně velikosti (ve všech rozlišeních). Právě při rozpoznávání s ohledem na změnu velikosti dosahuje model *MHONN* nejnižších úspěšností ve srovnání s ostatními modely. Výrazné snížení úspěšnosti vůči modelu *POHONN* se projevuje při rozpoznávání poškozených vzorů.

Výsledky původní verze modelu *MHONN* odpovídají našim představám. Přidáním informace o velikosti vstupních trojúhelníků do vstupních hodnot sítě pozitivně ovlivňuje rozpoznávání při vyšších rozlišeních a negativně ovlivňuje rozpoznávání poškozených vzorů. Na celkově nízkou dosahovanou úspěšnost při rozpoznávání s ohledem na změnu velikosti upozorňovali již autoři modelu [1] a naše experimenty ji potvrdily.

Aplikací našich modifikací pro model *MHONN* jsme dosáhli výrazně lepších výsledků (o 5% - 30%) ve všech oblastech, kromě rozpoznávání s ohledem na poškození vzoru. U rozpoznávání poškozených vzorů došlo ke snížení úspěšnosti o 2% - 4% pro všechna rozlišení. Kladně hodnotíme odstranění nízké úspěšnosti při rozpoznávání znaků s ohledem na změnu velikosti vzoru.

6.1.3 AHONT

U modelu *AHONT* nemůžeme mluvit přímo o jeho výsledcích. Model *AHONT* přinesl adaptivní stromovou strukturu navrženou pro neuronové sítě vyšších řádů. Jeho návrh (jak jej definovali Foresti a spol. [3]) nepodporuje přímo rozpoznávání znaků s ohledem na různé invariance. Model si proto nedokáže poradit s většinou zkoumaných invariancí.

Obecná definice modelu *AHONT* však dovoluje použít adaptivní stromovou strukturu ve spojením s ostatními námi zkoumanými modely a tím částečně zlepšit jejich výsledky. Především v kombinaci s modifikovaným modelem *MHONN* dosahuje model *AHONT* nejlepších výsledků ze všech zkoumaných modelů. Kombinace těchto modelů zvyšuje nejlepší dosažené úspěšnosti ve většině případů o 1% - 3%. Pouze při rozpoznávání poškozených vzorů dosahují nejlepších výsledků samostatné modely.

Zvýšení úspěšnosti kombinace modifikované verze modelu *AHONT* s modifikovaným modelem *MHONN* je dáno především vlastnostmi modelu *MHONN*. U modelu *MHONN* existují konfigurace, které se dokáží naučit pouze část trénovací množiny a zároveň dosahují vyšší úspěšnosti na testovací množině v poměru k počtu naučených vzorů. Vložením podobné konfigurace do kořene stromové struktury modelu *AHONT* a použitím celkově nejúspěšnějších konfigurací do úrovní vyšších, můžeme dosáhnou lepších výsledků při rozpoznávání. Bohužel tento jev neplatí pro rozpoznávání poškozených vzorů. Většina konfigurací modelu *MHONN* dokáže rozpoznat s vysokou úspěšností pouze některé poškozené vzory. Spolupráce různých konfigurací pak zajistí, že tyto vzory rozpoznává pouze kořen stromu, a celkově dochází ke snižování úspěšnosti.

Kapitola 7

Závěr

Cílem naší práce bylo prozkoumat řešitelnost úlohy rozpoznávání znaků s ohledem na některé invariance (např. posunutí, otočení, změna velikosti, ...). Úlohu rozpoznávání znaků jsme řešili použitím vybraných modelů neuronových sítí vyšších řádů. Implementovali jsme vybrané modely a otestovali je při rozpoznávání znaků. Získané výsledky jsme diskutovali a vzájemně porovnali.

7.1 Zhodnocení cílů

Hlavními cíli práce bylo implementování a experimentální ověření modelů použitelných pro problematiku rozpoznávání znaků s ohledem na různé invariance. Zaměřili jsme se na oblast neuronových sítí vyšších řádů. Dále jsme se pokusili vybrané modely modifikovat ve snaze dosáhnout lepších výsledků při rozpoznávání znaků. Pro všechna praktická ověřování a měření jsme vybrané modely implementovali a vytvořili tak samostatnou aplikaci umožňující zkoumání neuronových sítí vyšších řádů pro rozpoznávání znaků.

Cíle diplomové práce definované v kapitole 1.2 byly dosaženy.

- Podařilo se nám nastudovat, implementovat a otestovat všechny vybrané modely neuronových sítí vyšších řádů (kapitoly 3, 4, 5). Experimentálně jsme ověřili vlastnosti modelů s ohledem na vybrané invariance (kapitoly 3.3, 4.3, 5.3). Modely jsme vzájemně porovnali (kapitola 6).
- Všechny modely jsme modifikovali za cílem dosažení lepších rozpoznávacích vlastností modelů s ohledem na vybrané invariance (kapitoly 3.2, 4.2, 5.2). Experimentálně jsme ověřili vlastnosti námi navržených modifikací s ohledem na vybrané invariance. Získané výsledky jsme diskutovali a srovnali je

s výsledky původních modelů. Ve většině případů se nám podařilo dosáhnout zvýšení úspěšnosti.

- Pro výzkum jsme implementovali jednoduchou aplikaci umožňující učení a testování modelů vyšších neuronových sítí (Příloha B).

Při ověřování jednotlivých modelů jsme dosáhli relativně dobrých výsledků. Námi navržené modifikace modelů se osvědčily. Přesto existuje celá řada možností, jak by bylo možné náš výzkum dále rozšířit. V následující kapitole představíme některé z těchto možností.

7.2 Další výzkum

Při implementaci a testování zkoumaných modelů jsme narazili na směry, kterými by mohly být modely v budoucnu rozšiřovány a vylepšovány.

7.2.1 Rozšíření experimentů

Při experimentálním ověřování jsme i přes relativně vysoký počet testovaných konfigurací jednotlivých modelů nemohli zahrnout všechny možné varianty. Jedním z možných rozšíření naší práce by mohlo být experimentnální ověření dalších konfigurací jednotlivých modelů (např. různé kombinace skrytých vrstev v modelech).

7.2.2 Ošetření šumu

Výsledky při testování modelů na rozpoznávání poškozených vzorů nedosahovaly většinou příliš dobrých hodnot. Jde o typický problém mnoha modelů. I nepatrné množství šumu může zapříčinit chybnou klasifikaci snadno rozpoznatelného vzoru. Potlačení negativního dopadu šumu na úspěšnosti modelů by mohlo významně ovlivnit celkovou použitelnost modelů v praxi.

7.2.3 Stupně šedi

Všechny modely byly navrženy pro rozpoznávání černo-bílých bitmap (vzorů). Právě u černo-bílých vzorů způsobují deformace (změna velikost, rotace, ...) velké

poškození. Při nízkých rozlišeních vzorů nejsou některé deformované vzory rozpoznatelné ani lidským okem. Pokud by se podařilo uzpůsobit modely pro rozpoznávání vzorů s odlišenými stupni šedi, lze očekávat výsledky mnohem lepší. Deformace pro takové vzory by již nezpůsobily tak velké poškození a byly by snáze rozpoznatelné lidským okem.

7.2.4 Zapojení dalších modelů

Vzhledem k obecným možnostem neuronových sítí při rozpoznávání vzorů a jejich širokému využití existuje celá řada jiných modelů a návrhů použitelných na problematiku rozpoznávání vzorů s ohledem na invariance. Využitím těchto nových myšlenek a návrhů ve spojitosti s námi zkoumanými modely by mohly vzniknout nové směry, kterými by bylo možné pokračovat ve výzkumu. Možnou variantou by mohly být například neuronové sítě typu *neocognitron* [4].

Literatura

[1] Artymov E., Yadid-Petch O. (2004): *Modified High-Order Neural Network for Invariant Pattern Recognition. Pattern Recognition Letters, 26, 843-851.*

[2] Duda R. O., Hart P. E. Hart, Stork D. G. (1997): *Pattern Classification (2nd ed.). Wiley-Interscience, New York.*

[3] Foresti G. L., Christian M., Snidaro L. (2002): *Adaptive High Order Neural Trees for Pattern Recognition. Proc. of the 16th International Conference on Pattern Recognition, 2, 877-880.*

[4] Fukushima K. (1980): *Neocognitron: A Self-Organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position. Biological Cybernetics, 193-202.*

[5] Gilles C. L., Griffin R. D., Maxwell T. (1988): *Encoding Geometric Invariances in Higher-Order Neural Networks. Neural Information Processing Systems, American Institute of Physics Conference Proceedings, 301-309.*

[6] Jacobs R. (1988): *Increase Rates of Convergence Through Learning Rate Adaptations. Neural Networks, 1, 295-307.*

[7] Kollias S., Stafylopatis A., Tirakis A. (1991): *Performance of Higher Order Neural Networks in Invariant Recognition. Neural Networks: Advances and Applications, 79-108.*

[8] LeCun Y., Bottou L., Bengio Y., Haffner P. (1998): *Gradient-Based Learning Applied to Document Recognition. Proceedings of the IEEE, 86(11), 2278-2324.*

[9] McCulloch, W., Pitts, W. (1943): *A Logical Calculus of the Ideas Immanent in Nervous Activity. Bulletin of Mathematical Biophysics, 5, 115-133.*

[10] Minsky M., Papert S. (1969): *Perceptrons. MIT Press, Cambrige.*

[11] Mitchell T., McGraw H. (1997): *Machine Learning. McGraw-Hill College, 81-126.*

[12] Perantonis S.J., Lisboa P.J.G (1992): *Translation, Rotation and Scale Invariant Pattern Recognition by Higher-Order Neural Networks and Moment Classifiers. Pattern Recognition, 25, 975-985.*

[13] Rosenblatt F. (1958): *The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. Cornell Aeronautical Laboratory, Psychological Review, 65, 6, 386-408.*

[14] Rumelhart D. E., Hinton G. E., Williams R. J. (1986): *Learning Internal Representations by Error Propagation. MIT Press, Cambridge, 318-362.*

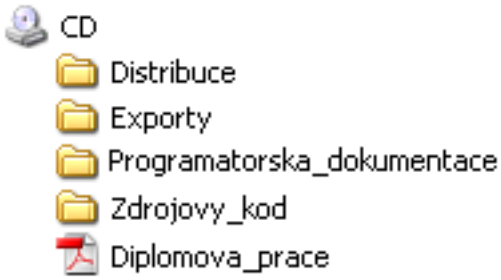
[15] Rumelhart D. E., McClelland J. L. (1986): *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. MIT Press, Cambrige.*

[16] Simard P. Y., LeCun Y., Denker J. S., Victorri B. (2001): *Transformation Invariance in Pattern Recognition - Tangent Distance and Tangent Propagation. International Journal of Imaging Systems and Technology, 11(3), 181-194.*

[17] *Javadoc Tool. http://java.sun.com/j2se/javadoc/.*

Příloha A - Obsah CD

Součástí diplomové práce je přiložené CD se souvisejícími materiály. Přesnějšímu popisu obsahu CD je věnována tato kapitola. Na obrázku 7.1 je možné vidět strukturu přiloženého CD.



Obrázek 7.1: Obsah přiloženého CD

V kořenovém adresáři se nachází soubor *Diplomova_prace.pdf*. Jde o celou diplomovou práci uloženou ve formátu *pdf*.

Distribuce

Adresář *Distribuce* obsahuje vše potřebné pro nainstalování naší aplikace pro testování neuronových sítí vyšších řádů. Obsáhlý popis instalace a hardwarových požadavků obsahuje Příloha B.

Exporty

Adresář *Exporty* obsahuje veškeré exporty výsledků z našeho zkoumání. V adresářích odpovídajících zkratkám jednotlivých modelů jsou uloženy výsledky jednotlivých modelů. Exporty jsou uloženy v souborech **.xls* (Microsoft Excel).

Exporty obsahují všechny výsledky daného modelu seřazené od nejúspěšnějších. Názvy a pořadí atributů odpovídá výsledkům prezentovaným v kapitolách věnovaných jednotlivým modelům (Kapitoly 3, 4, 5).

Programátorská dokumentace

Adresář *Programatorska_dokumentace* obsahuje vygenerovaný javadoc [17] pro nejdůležitější třídy celé aplikace. Jedná se výstup softwarové pomůcky pro jazyk Java pro automatické generování dokumentace. Výsledkem jsou html stránky s úvodní stránkou *index.html*.

Zdrojové kódy

Adresář *Zdrojove_kody* obsahuje zdrojové kódy naší aplikace pro testování neuronových sítí vyšších řádů.

Příloha B - Instalace a spuštění aplikace

Tato kapitola je věnována podrobnému popisu instalace naší aplikace a obeznámení s hardwarovými nároky.

Hardwarové a softwarové požadavky

Naše aplikace byla vytvořena za účelem testování neuronových sítí vyšších řádů. Její hardwarové a softwarové požadavky vyplývají jednak z programovacího jazyka Java, ve kterém byla celá aplikace napsána, jednak ze samotné náročnosti problematiky rozpoznávání vzorů. Hardwarové a softwarové požadavky jsou následující:

- Instalace *JRE/JDK* (verze min 1.4.2).
- Doporučená operační paměť min. 512MB (doporučená 1024MB).

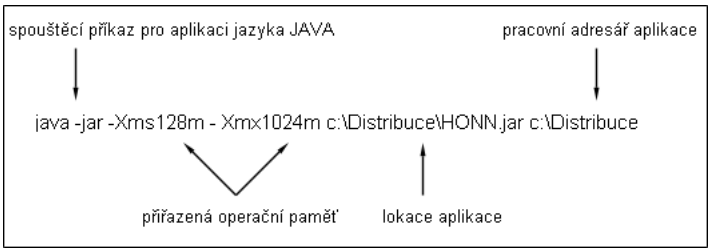
Instalace

Aplikace je distribuována jako samospouštěcí jar (executable jar) - *HONN.jar*. Tento soubor se nachází v adresáři *Distribuce*. Pro rychlé spuštení lze použít přiložené CD. Příkaz pro spuštění je vidět na obrázku 7.2.

Doporučujeme však zkopírovat obsah adresáře *Distribuce* na lokální disk, aby bylo možné ukládat změny v předvyplněných modelech. Jedná se o pracovní adresáře aplikace s předvyplněnými základními daty (např. námi použité vzory pro testování, uložené konfigurace testovaných modelů, ...).

Pro jednodušší spouštění jsme připravili dávková soubor *HONN.bat*, ve kterém je třeba upravit spouštěcí atributy aplikace tak, aby parametry odpovídaly

skutečným lokacím na disku. Vysvětlení jednotlivých parametrů je na obrázku 7.2.



Obrázek 7.2: Spouštěcí příkaz pro aplikaci.

Potřebná operační paměť souvisí se způsobem práce s aplikací a s testovanými konfiguracemi. Pro práci s obsáhlými plány, případně konfiguracemi vytvářející rozsáhlé neuronové sítě (např. velmi malé hodnoty pro míry úhlu o obsahu), je potřeba větší množství paměti. Přednastavené hodnoty poskytují typicky dostatek paměti pro většinu operací.

Adresářová struktura aplikace

Na obrázku 7.3 je možné vidět strukturu pracovních adresářů aplikace.

Obrázek 7.3: Adresářová struktura aplikace.

Význam jednotlivých adresářů je následující:

- *images* - adresář pro ukládání vzorů, ze kterých se tvoří trénovací a testovací množiny.
- *plans* - adresář pro ukládání plánů - seznamů konfigurací pro trénování či rozpoznávání.
- *saves* - adresář pro ukládání jednotlivých sítí.
- *sets* - adresář pro ukládání trénovacích a testovacích množin.

Spuštění

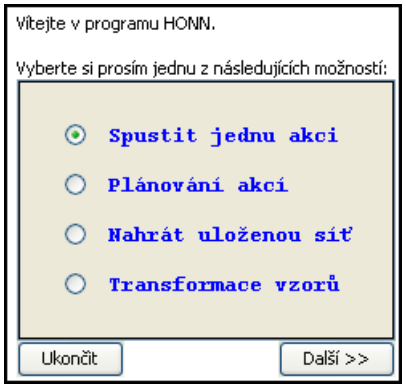
Aplikace se spouští souborem *HONN.bat*.

Příloha C - Uživatelské rozhraní aplikace

Následující kapitola popisuje uživatelské rozhraní naší aplikace pro zkoumání a testování neuronových sítí vyšších řádů. Popíšeme nejdůležitější úkony spojené s učením neuronových sítí, následnou klasifikací vzorů a exportováním výsledků. Upozorníme na některé důležité, avšak méně nápadné možnosti našeho programu. Podrobně popíšeme složitější operace (např. vytváření plánů).

Úvodní obrazovka

Po spuštění aplikace se zobrazí úvodní rozcestník (obrázek 7.4) se základními skupinami operací, které aplikace poskytuje. Zvolením jedné z možností a následným potvrzením tlačítkem *Další*, se přejde do části aplikace odpovídající zvolené skupině operací.



Obrázek 7.4: Úvodní obrazovka

Popisu jednotlivých skupin operací jsou věnovány následující kapitoly. Stručný popis jednotlivých operací následuje.

- *Spustit jednu akci* - obsahuje úkony pro práci s jedním modelem. V rámci této skupiny úloh lze konfigurovat, trénovat a následně ukládat jednotlivé modely vyšších neuronových sítí. Současně tato část aplikace obsahuje administraci trénovacích a testovacích množin.
- *Plánování akcí* - obsahuje úkony pro práci s více modely najednou (tzv. plány). Jedná se o vytváření/úpravy/rušení jednotlivých plánů, jejich trénování, testování a následné exportování do přehledných tabulek.
- *Nahrát uloženou síť* - umožňuje načíst dříve uloženou konfiguraci modelu a otestovat ji na zvolené testovací množině.
- *Transformace vzorů* - poskytuje nástroje pro aplikaci jednotlivých invariancí do zvolených vzorů (obrázků).

Nyní se zaměříme na operace dostupné zvolením možnosti *Spustit jednu akci*.

Jedna akce

Pro provádění libovolné operace je nutné zvolit jeden z implementovaných modelů. Na obrázku 7.5 je možné vidět modely nabízené v naší aplikaci. Jedná se o modely, které jsme zkoumali v předešlých kapitolách. Pořadí jednotlivých modelů odpovídá pořadí kapitol, ve kterých byly modely představeny (Kapitoly 3, 4 a 5). Na výběr jsou poskytnuty modely včetně námi modifikovaných verzí.

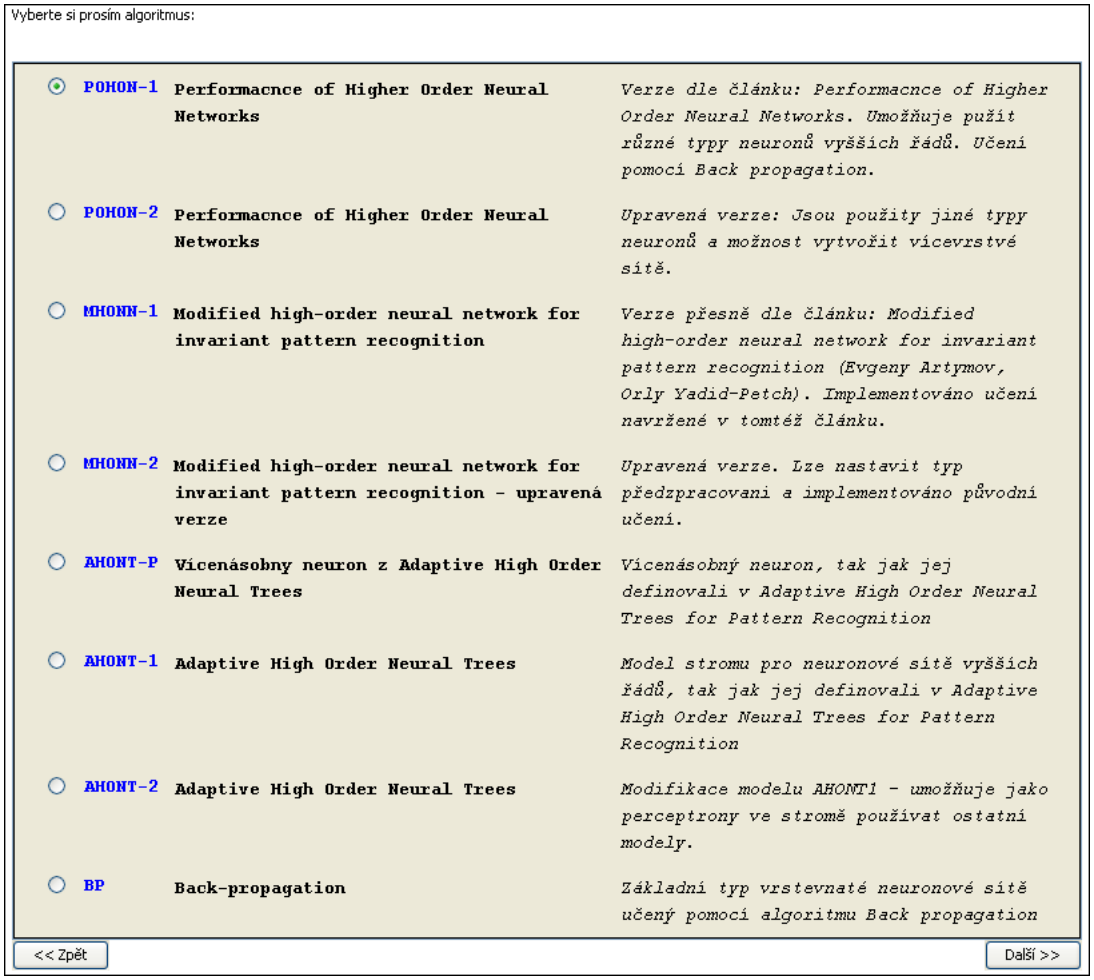
Výjimkou jsou modely *AHONT-P* a *BP*.

- Model *AHONT-P* je součástí implementace modelu *AHONT* (kapitola 5). Jedná se o samostatný model vícenásobného neuronu.
- Model *BP* představuje základní vrstevnatou neuronovou síť (viz kapitola 2.1) trénovanou pomocí algoritmu Back propagation.

Výběrem modelu a následným potvrzení tlačítkem *Další* se přejde ke konfiguraci modelu a administraci/výběru trénovacích množin.

Konfigurace modelu

Levá část panelu je věnován konfiguraci zvoleného modelu, pravá část je věnována administraci/výběru trénovacích množiny (obrázek 7.6).



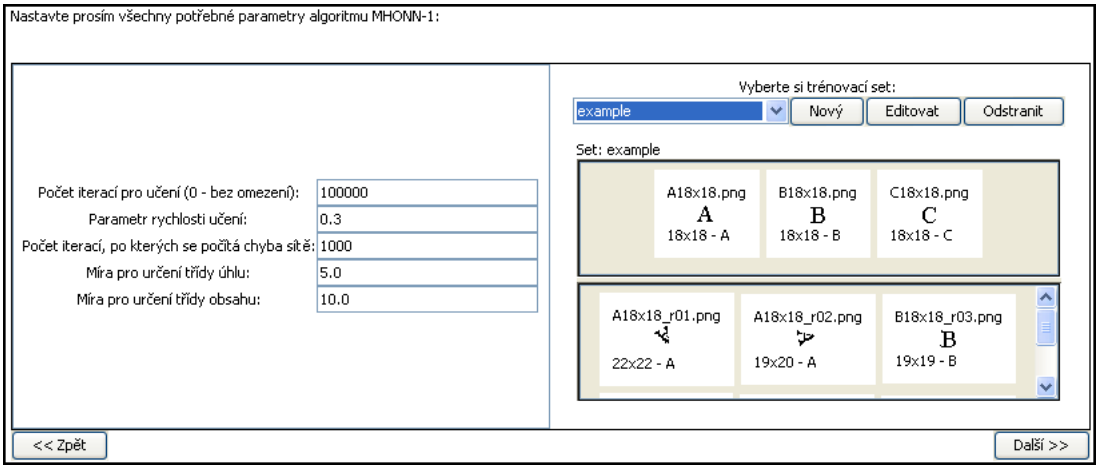
Obrázek 7.5: Výběr modelu pro jednu akci

Konfigurace modelu je předvyplněna parametry tak, aby se dalo přejít rovnou k trénování. Přesný popis jednotlivých parametrů konfigurací pro jednotlivé modely je popsán v Příloze D. Pro výběr trénovací množiny slouží výklopná nabídka v levé horní části panelu. Vybraná trénovací množina může být zároveň upravována.

Administrace trénovacích množin

Trénovací (testovací) množina je reprezentována dvěma skupinami vzorů - *trénovací* a *testovací*.

- *Trénovací skupina vzorů* slouží k trénování sítě, jedná se o vzory předkládané síti v průběhu učení.



Obrázek 7.6: Konfiguace modelu MHONN

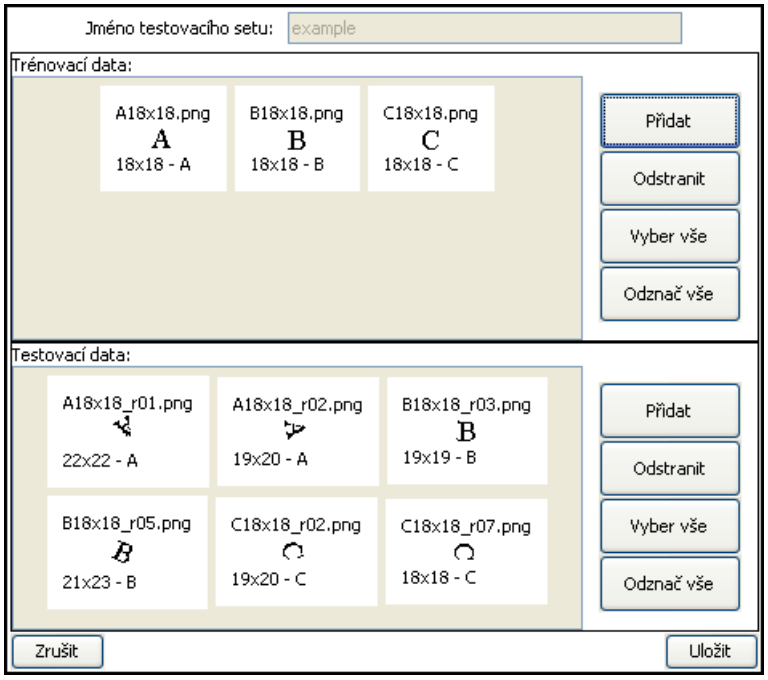
- *Testovací skupina vzorů* slouží ve fázi trénování k výpočtu globální chyby, avšak vzory nejsou předkládány síti při učení.

Při testování (klasifikaci) sítě, lze zvolit, na které skupině bude docházet k rozpoznávání. Při učení sítí jsme testovací skupinu většinou nechali prázdnou, protože její vyplněnost nemá vliv na výsledek trénování sítě a zároveň výrazně zpomaluje trénovací fázi. Použití testovací skupiny je vhodné v počátečních krocích výzkumu při práci s jednotlivými modely sítí. Díky testovací skupině je již v průběhu učení vidět schopnosti sítě poznávat vzory neobsažené v trénovací množině.

Pravá část panelu je věnované administraci těchto množin. Pomocí tlačítka *Nový* nebo *Editovat* se zobrazí detail množiny s možností přidávání/odebírání vzorů (viz obrázek 7.7). Tlačítky na pravé straně okna lze měnit obsah jednotlivých skupin vzorů. Změny se potvrzují tlačítkem *Uložit*.

Jednotlivé trénovací množiny a jejich obsah je dán obsahem adresáře *sets* v adresářové struktuře aplikace (kapitola 7.2.4). Každá trénovací množina je reprezentována adresářem odpovídajícím jménu trénovací množiny. Každá trénovací množina obsahuje dva podadresáře *training* a *testing*, ve kterých jsou soubory reprezentující vzory daných skupin. Vzory jsou reprezentovány soubory typu *PNG* s příponou *.png*.

Potvrzením konfigurace a výběru trénovací množiny přejdeme k samotnému trénování modelu.



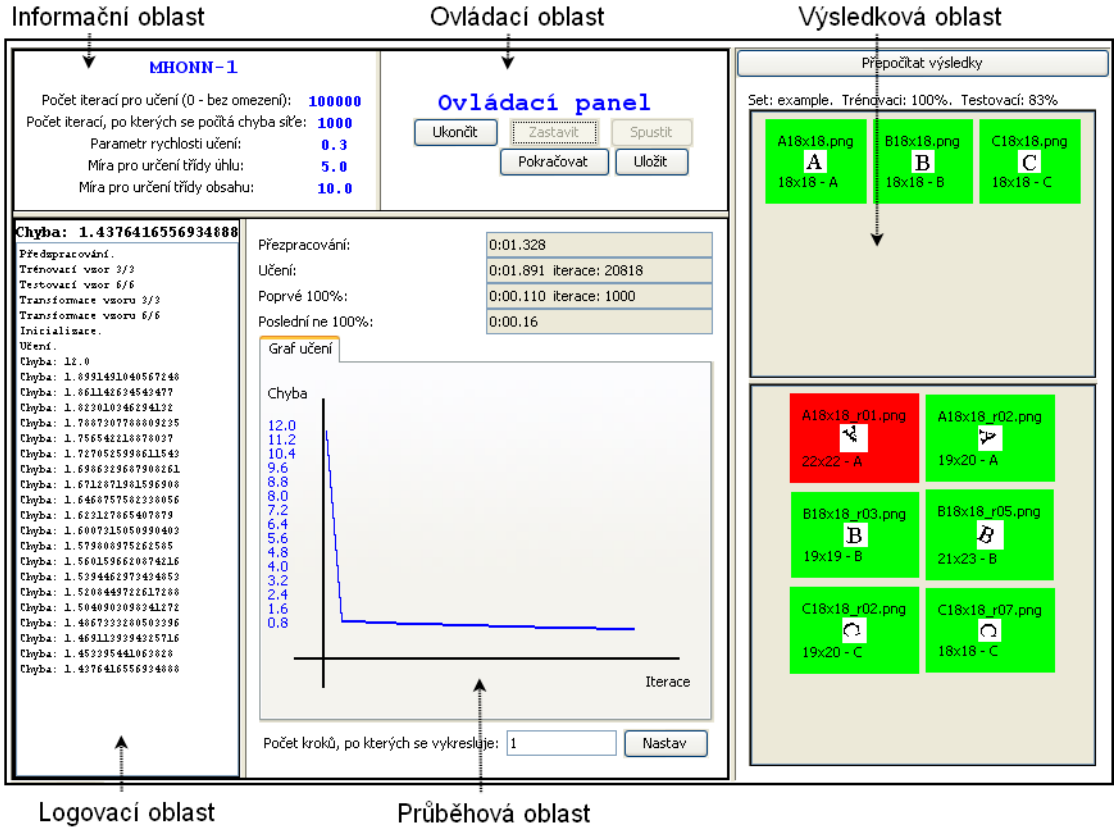
Obrázek 7.7: Editace trénovací množiny

Trénování modelu

Panel pro trénování modelu je rozdělen do několika logických oblastí. Na obrázku 7.8 je vidět ukázka panelu z průběhu trénování.

Jednotlivé oblasti a jejich význam je následující:

- *Informační oblast* - obsahuje informace o modelu. Nachází se zde jméno zvoleného modelu a jeho aktuální konfigurace (nastavená v předešlém kroku).
- *Ovládací oblast* - jedná se nejdůležitější oblast panelu. Obsahuje řídící prvky pro trénování. Jde o tlačítka *Spustit*, *Zastavit*, *Pokračovat*, *Ukončit* a *Uložit*, kterými se řídí trénování modelu.
- *Logovací oblast* - jde o textové záznamy provedených operací při trénování.
- *Průběhová oblast* - nachází se zde graf chyby učení a časy jednotlivých fází. U modelů *AHONT-1* a *AHONT-2* lze graf přepnout na zobrazení hierarchické struktury sítě.
- *Výsledková oblast* - zobrazuje trénovací a testovací množinu. Po spuštění učení jsou barevně odlišeny správně naučené vzory (zelená barva) a chybně naučené vzory (červená barva). V záhlaví se nachází procentuální vyjádření úspěšnosti učení.



Obrázek 7.8: Panel pro učení

V *Průběhové oblasti* se nachází časy a iterace jednotlivých částí trénování. Jejich význam je následující:

- *Předzpracování* - čas trvání fáze předzpracování.
- *Učení* - čas a počet iterací samotného učení, od ukončení fáze předzpracování.
- *Poprvé 100%* - nejnižší čas a číslo iterace, kdy síť byla schopna rozpoznat celou trénovací množinu.
- *Poprvé ne 100%* - nejvyšší čas a číslo iterace, kdy síť nebyla schopna rozpoznat celou trénovací množinu.

Ukládat model lze pouze při pozastaveném nebo ukončeném trénování. Je vhodné ukládat model pouze ve fázi trénování nebo po jejím ukončení, ne při předzpracování. Model uložený při předzpracování nemůže nikdy správně klasifikovat jakékoliv vzory. Při ukládání je uživateli nabídnut adresář *save* z adresářové struktury aplikace pro uložení. Pro větší přehlednost doporučujeme ukládat jednotlivé

modely do podsložek odpovídajících jednotlivým modelů. Modely jsou ukládány do souborů ve formátu *XML*.

Po stisknutí tlačítka *Zastavit* dojde k pozastavení trénování, trénování lze opětovně spustit tlačítkem *Pokračovat*. Po stisku tlačítka *Ukončit* dojde k definitivnímu ukončení trénování. Nelze již znovu spustit tlačítkem *Pokračovat*.

Tlačítkem *Nastav* v dolní části uprostřed lze v kombinaci s kladným celým číslem snížit počet úseček, ze kterých je sestaven graf průběhu učení.

Plánování akcí

Plánování akcí umožňuje práci s více modely/konfiguracemi modelů najednou. Následující část aplikace umožňuje vkládat jednotlivé modely/konfigurace modelů do plánů. Takto vytvořený plán lze trénovat, nechat klasifikovat a výsledky exportovat.

Napřed je třeba vysvětlit, jak plány fungují a jakým způsobem se ukládají. Plán je v podstatě seznam jednotlivých modelů, jejichž trénování a testování lze spustit najednou. Jednotlivé modely v plánu jsou označeny jedinečnými jmény. Pro správné pochopení je třeba rozumět způsobu ukládání jednotlivých plánů.

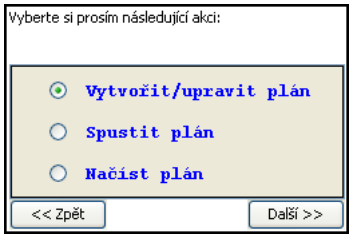
Uložení plánů vytvoří specifické soubory v adresářové struktuře aplikace (podobně jako při ukládání trénovacích množin). Každý plán je reprezentovaný hlavním souborem odpovídajícího jména. V tomto souboru je uložen seznam jmen jednotlivých modelů v plánu, názvy trénovacích a testovacích množin, trénovacích a testovacích úspěšností. Jednotlivé modely pro plán jsou uloženy samostatně v adresáři jména uloženého plánu a přípony „items“.

Rozlišujeme dva typy plánů - *Základní* a *Výsledkové*.

- *Základní plány* jsou obyčejné plány umožňující provádět fázi trénování i testování.
- *Výsledkové plány* jsou kopiemi základních plánů. Slouží pouze k otestování základních, již naučených, plánů. Nelze u nich provádět samostatné trénování. Uložený plán je reprezentován pouze jediným (hlavním) souborem.

Po spuštění plánování akcí se zobrazí rozcestník (viz obrázek 7.9) pro výběr následující akce. Význam a účel jednotlivých akcí je následující:

- *Vytvořit/upravit plán* - umožňuje spustit administraci plánů (vytváření/ukládání/rušení plánů).



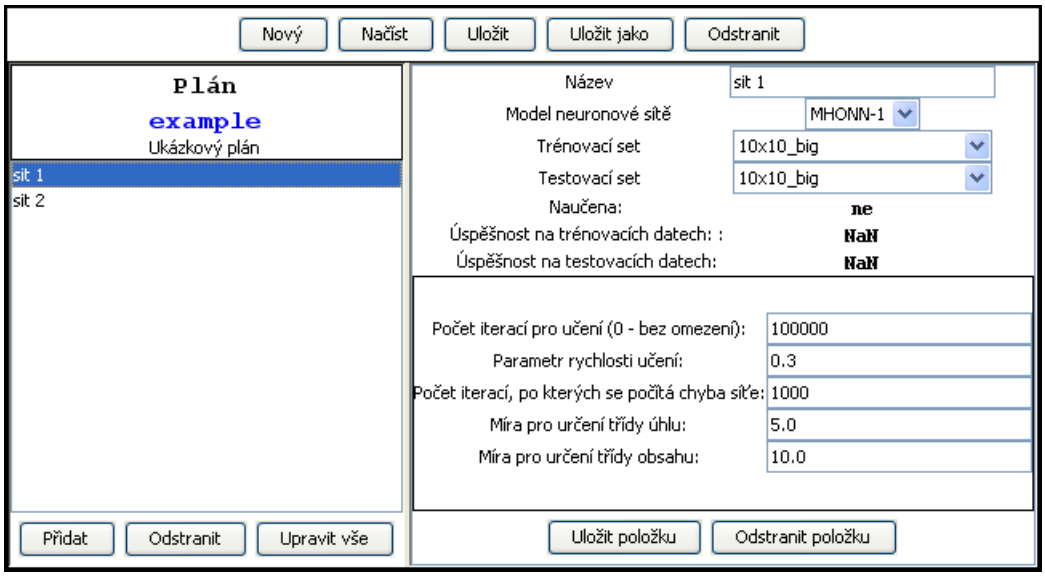
Obrázek 7.9: Možnosti pro plánování akcí

- *Spustit plán* - umožňuje spustit jeden či více plánů.
- *Načíst plán* - umožňuje načíst uložený plán. Výsledky načteného plánu lze exportovat. Jednotlivé modely načteného plánu lze testovat na různých testovacích množinách.

V následujících kapitolách se zaměříme na jednotlivé oblasti operací s plány.

Tvorba, úprava plánu

Základní činnost je tvorba a následná úprava plánu. Tyto akce se ovládají z panelu pro administraci plánů (viz obrázek 7.10).



Obrázek 7.10: Úprava plánu

Horní část panelu obsahuje ovládací prvky pro načítání, ukládání a rušení celého modelu. Levá část panelu reprezentuje samostatný plán. Horní část zobrazuje

informace o plánu - název a popis. Seznam všech položek plánu - jednotlivých modelů v plánu. Výběrem jednoho z modelů se v pravé části zobrazí detail zvoleného modelu.

Zaměříme se nyní na detaily jednotlivých modelů. Detail obsahuje následující atributy.

- *Název* - označení prvku v plánu. Musí být v rámci celého plánu jedinečné.
- *Model neuronové sítě* - jde o jeden z modelů popsaných v předešlých kapitolách 3, 4 a 5.
- *Trénovací set* - trénovací množina, na které probíhá trénování modelu. Vzory z množiny jsou předkládány modelu při učení.
- *Testovací set* - testovací množina, na které probíhá testování modelu. Vzory jsou klasifikovány modelem a úspěšnost klasifikace udává úspěšnost modelu.
- *Naučena* - hodnota „Ano“ reprezentuje stav, kdy model je již naučen. Hodnota „Ne“ označuje stav, kdy model ještě nebyl trénován.
- *Úspěšnost na trénovacích datech* - pokud je model již naučen, obsahuje toto políčko procentuální úspěšnost na trénovacích datech.
- *Úspěšnost na testovacích datech* - pokud je model již naučen a proběhlo rozpoznávání na testovací množině, obsahuje toto políčko procentuální úspěšnost na testovacích datech.
- *Konfigurace modelu* - dolní část okna umožňuje nastavit konfiguraci modelu. Konfiguracím jednotlivých modelů je věnován Příloha D.

Jakákoliv provedená změna v detailu modelu musí být potvrzena stiskem tlačítka *Uložit položku*. Pokud není potvrzena, změny se neuloží do plánu.

Tlačítka pod seznamem položek v plánu umožňují pracovat s jednotlivými položkami plánu. Ve většině případů umožňují pracovat s více položkami současně. Jejich význam je následující:

- *Přidat* - přidání nové položky do plánu. Po přidání se zobrazí prázdný detail položky, který je třeba vyplnit.
- *Kopírovat* - k označeným položkám se vytvoří kopie, které lze následně upravovat.
- *Nahoru* - posune označené položky v seznamu nahoru.

- *Upravit* - otevře okno po úpravu více položek najednou. Tímto oknem lze změnit trénovací a testovací množinu přidělenou k jednotlivým položkám. Při změně trénovací množiny naučený model přijde o všechny informace získané trénováním a ztratí veškeré výsledky. Při změně testovací množiny zůstává naučený model naučen, pouze přijde o výsledky na testovacích datech.
- *Odstranit* - odstraní zvolené položky z plánu.
- *Dolů* - posune označené položky v seznamu dolů.

Pro rozsáhlé testování doporučujeme několik následujících rad:

- Je dobré tvořit plány s přiměřeným počtem položek (řádově desítky). Práce s rozsáhlými plány je výpočetně náročná.
- Pro přehlednost doporučujeme dávat do plánů pouze modely jednoho typu. Dodržování tohoto pravidla usnadní práci s exporty.
- Při nutnosti otestovat naučený plán na jiných datech doporučujeme následující postup.
  - Načíst naučený model.
  - Pomocí tlačítka *Uložit jako* a následné zaškrtnutí možnosti *Pouze pro výsledky* vytvořit *Výsledkový plán*
  - Označit všechny položky plánu a pomocí tlačítka *Upravit* nastavit novou testovací množinu.
  - Spustit nově vytvořený plán.

Spuštění plánu

Při spuštění plánu je třeba vybrat plán. Všechny nenaučené nebo neotestované položky z vybraného plánu se přidají do seznamu ke zpracování. Pomocí tlačítka *Přidat plán* lze připojit položky dalších plánů.

Tlačítkem *Spustit* lze spustit trénování (testování) položek plánů. Dle seznamu se zpracovávají jednotlivé položky. Pokud položka není naučena, spustí se učení. Pokud položka není otestována, spustí se testování. Tímto způsobem se zpracují všechny položky v seznamu.

Při běhu plánu dochází k automatickému ukládání plánu v následujících krocích.

- Po ukončení trénování právě zpracované položky dojde k uložení celého plánu.
- Po ukončení testování právě zpracované položky dojde k uložení celého plánu.

Načtení plánu

Pomocí tlačítka *Načíst* lze načíst uložený plán. Zobrazí se seznam všech položek plánu. Vybráním položky plánu dojde k zobrazení detailu položky s možností otestovat položku na libovolné testovací množině. Obsluha tohoto rozpoznávání se provádí stejně jako při načtení jednoho modelu (kapitola 7.2.4).

Načtený plán lze exportovat do tabulky pomocí tlačítka *Exportovat výsledky*. Tato akce vyvolá okno s tabulkou obsahující výsledky. Tabulka obsahuje informace o konfiguracích modelu a dosažené výsledky z trénování a testování. Pravým tlačítkem na myši dojde k vyvolání kontextové nabídky s možností otevřít výsledky v příslušném programu (typicky *Microsoft Excel*). Dojde k vygenerování souboru *export.xls* a jeho následovnému otevření. Operace export může u složitých plánů trvat řádově až minuty.

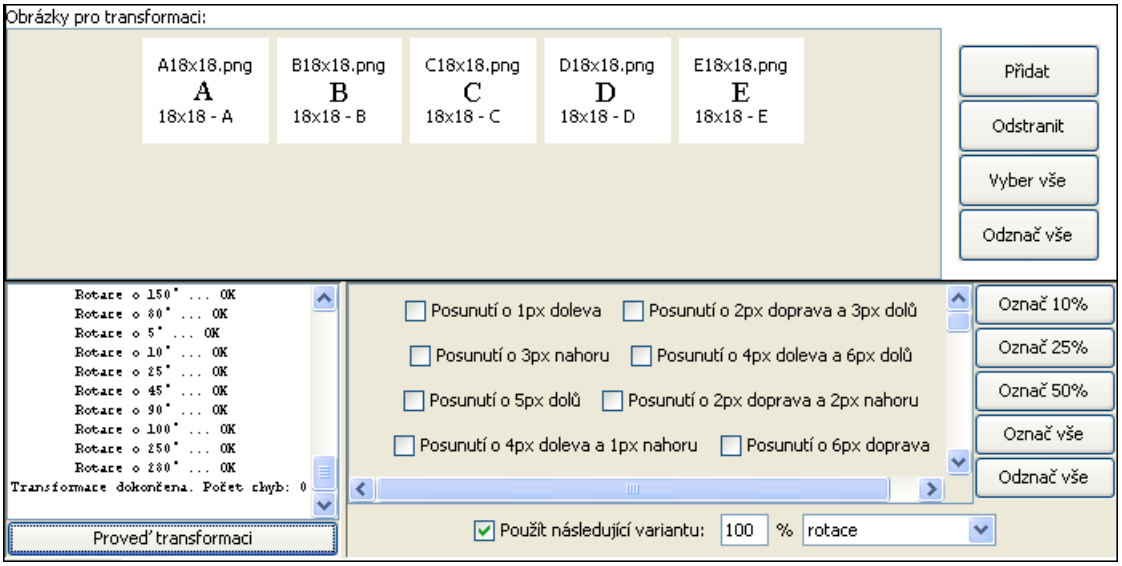
Načtení uloženého modelu

V panelu pro načítání uložených sítí lze pomocí tlačítka *Načíst neuronovou síť* nahrát uložený model. V levé části panelu se zobrazí informace o načteném modelu (název, typ, konfigurace, ...). Pomocí výklopných nabídek lze vybrat novou testovací množinu. Tlačítkem *Rozpoznej* se spustí testování nahraného modelu na vybrané testovací množině. Po dokončení testování se zobrazí panel s procentuálními výsledky.

Transformace vzorů

Poslední oblast aplikace poskytuje nástroje pro transformace trénovacích (testovacích) vzorů. Panel pro ovládání lze vidět na obrázku 7.11.

Horní část panelu umožňuje výběr transformovaných vzorů. Pomocí tlačítek *Přidat* a *Odstranit* lze vybrat skupinu vzorů, na kterých budou provedeny vybrané transformace.



Obrázek 7.11: Transformace vzorů

Dolní část panelu slouží k výběr aplikovaných transformací. Aplikace poskytuje dva způsoby, jak lze vybírat transformace.

- Pokud je označeno políčko *Použít následující variantu* v dolním řádku panelu, výběr udává procentuální hodnota společně s výklopnou nabídkou ve spodním řádku. Kombinace vybere typ prováděných transformací, procentuální hodnota určí, jaké množství transformací bude provedeno (kolik procent ze všech transformací daného typu).
- V ostatních případech jsou použity označené deformace ve střední části panelu. Tlačítka v pravé části panelu umožňují náhodně vybírat deformace ze střední části panelu.

Stiskem tlačítka *Proveď transformaci* se spustí transformace vybraných vzorů. V dolní levé části se zobrazuje průběh operací. Na závěr je třeba vybrat adresář, kam se mají modifikované vzory uložit.

Příloha D - Konfigurace modelů

V následujícím textu podrobně vysvětlíme všechny parametry pro konfigurace jednotlivých modelů, které se dají v naší aplikaci nastavit. Konfigurace modelů jsou popisovány v pořadí, ve kterém jsme jednotlivé modely zkoumali.

POHONN

Nyní popíšeme nastavení konfigurace pro model z kapitoly 3. Jedná se o původní verzi modelu.

- *Typ aproximativně podobných trojúhelníků* umožňuje vybrat způsob klasifikace aproximativně podobných trojúhelníků do tříd podobnosti pro kombinaci všech invariancí (viz kapitola 3.1.1). Lze vybrat, zda klasifikace bude závislá na pořadí trojúhelníků či nikoliv.
- *Míra pro určení třídy úhlu* - umožňuje nastavit konstantu c_a pro zaokrouhlování hodnot vnitřních úhlů vstupních trojúhelníků. Při nastavení parametru c_a na velmi nízké hodnoty (< 1), vzrůstají výrazně paměťové a výpočetní nároky pro trénování modelu.
- *Počet iterací pro učení* může nabývat nezáporných celých hodnot. Udává počet iterací, které se mají vykonat při trénování sítě. Hodnota 0 znamená, že nebude omezen počet iterací pro učení. Učení bude probíhat, dokud jej uživatel nezastaví.
- *Hodnota alfa* určuje hodnotu konstanty α pro algoritmus back-propagation (viz kapitola 2.4.1). Doporučená hodnota je z intervalu $\langle 0; 1 \rangle$.
- *Hodnota lambda* určuje hodnotu konstanty λ pro algoritmus back-propagation (viz kapitola 2.4.1).

- *Koeficient růstu lambdy* určuje konstantu d ($d_1 = d$, $d_2 = \frac{1}{d_1}$) z rozšíření algoritmu back-propagation pro model *POHONN* (viz kapitola 3.1.2). Doporučená hodnota je z intervalu $\langle 1,1; 1,3 \rangle$.

- *Maximální hodnota lambdy* umožňuje nastavit maximální hodnotu konstanty λ, pokud je umožněn růst konstanty ($d \neq 1$). Hodnota 0 znamená bez omezení. V praxi se nám nejlépe osvědčila hodnota 2.

- *Počet iterací pro výpočet chyby* udává počet iterací, po kterých má dojít k výpočtu celkové chyby sítě na trénovacích datech. Zároveň dojde k vykreslení aktualizovaného grafu chyby a aktuální úspěšnosti rozpoznávání v grafickém prostředí aplikace.

Modifikovaná verze modelu *POHONN* obsahuje všechny atributy původní verze. Jeden atribut však nabývá jiných hodnot a další dva jsou nově.

- *Typ neuronů* umožňuje vybrat způsob klasifikace aproximativně podobných trojúhelníků do tříd podobnosti (viz kapitola 3.2). Lze vybrat ze způsobu výpočtu obsahu možnost dle maximálního obsahu nebo dle mediánu.

- *Počet tříd relativního obsahu* umožňuje nastavit parametr s_n pro určení podobnosti aproximativně podobných trojúhelníků (viz kapitola 3.2). Při nastavení parametru s_n na velmi vysoké hodnoty vzrůstají výrazně paměťové a výpočetní nároky pro trénování modelu.

- *Vnitřní skryté vrstvy* nastavují počet a velikosti vnitřních skrytých vrstev neuronové sítě (viz kapitola 2.3.2). Velikosti jednotlivých vrstev se zapisují pomocí číslic. Jednotlivé vrstvy se oddělují znakem ",". Příklad pro síť se dvěma skrytými vrstvami o velikostech 5 a 6 je "5,6"'.

MHONN

Nyní se zaměříme na model popsaný v kapitole 4. Konfigurace původního modelu obsahuje následující parametry.

- *Počet iterací pro učení* může nabývat nezáporných celých hodnot. Udává počet iterací, které se mají vykonat při trénování sítě. Hodnota 0 znamená, že nebude omezen počet iterací pro učení. Učení bude probíhat, dokud jej uživatel nezastaví.

- *Parametr rychlosti učení* udává hodnotu konstanty π použitou v algoritmu pro trénování modelu (viz kapitola 4.1.4). Hodnoty jsou kladná reálná čísla. Doporučené hodnoty jsou z intervalu $\langle 0,2; 0,5\rangle$.

- *Počet iterací pro výpočet chyby* udává počet iterací, po kterých má dojít k výpočtu celkové chyby sítě na trénovacích datech. Zároveň dojde k vykreslení aktualizovaného grafu chyby a aktuální úspěšnosti rozpoznávání v grafickém prostředí aplikace.

- *Míra pro určení třídu úhlu* udává hodnotu konstanty c_a použitou ve vztazích pro určovní aproximativně podobných trojúhelníků (viz kapitola 4.1.2). Povolené hodnoty jsou kladná reálná čísla. Při nastavení parametru c_a na velmi nízké hodnoty (< 1), vzrůstají výrazně paměťové a výpočetní nároky pro trénování modelu.

- *Míra pro určení třídu obsahu* udává hodnotu konstanty c_s použitou ve vztazích pro určování aproximativně podobných trojúhelníků (viz kapitola 4.1.2). Povolené hodnoty jsou kladná reálná čísla. Při nastavení parametru c_s na velmi nízké hodnoty (< 1), vzrůstají výrazně paměťové a výpočetní nároky pro trénování modelu.

Modifikované verze modelu *MHONN* obsahuje tytéž parametry jako verze původní. Je však rozšířena o dva nové parametry.

- *Typ výběru okrajových pixelů* umožňuje zvolit způsob výběru okrajových pixelů (viz kapitola 4.2.1). Možnosti jsou následující.
  - *hranou* - jsou vybrány pouze pixely sousedící s hranou.
  - *hranou nebo vrcholem* - jsou vybrány pouze pixely sousedící s hranou nebo vrcholem.
  - *všechny* - jsou vybrány všechny pixely.

- *Typ aproximativně podobných trojúhelníků* umožňuje vybrat typ tvorby aproximativně podobných trojúhelníků (viz kapitola 4.2.2). Závisí na něm způsob započítání obsahu při určování třídy aproximativně podobných trojúhelníků.

AHONT

Nyní se zaměříme na model popsaný v kapitole 5. Konfigurace původního modelu obsahuje následující parametry.

- *Počet cyklů pro kontrolu ukončení učení* udává počet cyklů výpočtu chyby pro kontrolu stagnace chyby (viz kapitola 5.1.2). Tedy tato hodnota násobena hodnotou parametru *Počet iterací pro výpočet chyby* udává délku intervalu pro kontrolu stagnaci chyby.

- *Interval chyby pro kontrolu ukončení učení* nastavuje hodnotu e z algoritmu pro učení vícenásobného neuronu (viz kapitola 5.1.2). Jde o rozmezí chyby pro kontrolu stagnace.

- *Počet iterací pro učení* může nabývat nezáporných celých hodnot. Udává počet iterací, které se mají vykonat při trénování sítě. Hodnota 0 znamená, že nebude omezen počet iterací pro učení. Učení bude probíhat, dokud jej uživatel nezastaví.

- *Hodnota alfa* určuje hodnotu konstanty α pro algoritmus back-propagation (viz kapitola 2.4.1). Doporučená hodnota je z intervalu $\langle 0; 1 \rangle$.

- *Hodnota lambda* určuje hodnotu konstanty λ pro algoritmus back-propagation (viz kapitola 2.4.1).

- *Koeficient růstu lambdy* určuje konstantu d ($d_1 = d$, $d_2 = \frac{1}{d_1}$). Doporučená hodnota je z intervalu $\langle 1,1; 1,3 \rangle$.

- *Maximální hodnota lambdy* umožňuje nastavit maximální hodnotu konstanty λ, pokud je umožněn růst konstanty ($d \neq 1$). Hodnota 0 znamená bez omezení. V praxi se nám nejlépe osvědčila hodnota 2.

- *Počet iterací pro výpočet chyby* udává počet iterací, po kterých má dojít k výpočtu celkové chyby sítě na trénovacích datech. Zároveň dojde k vykreslení aktualizovaného grafu chyby a aktuální úspěšnosti rozpoznávání v grafickém prostředí aplikace.

- *Vnitřní skryté vrstvy* nastavují počet a velikosti vnitřních skrytých vrstev neuronové sítě (viz kapitola 2.3.2). Velikosti jednotlivých vrstev se zapisují pomocí číslic. Jednotlivé vrstvy se oddělují znakem ",". Příklad pro síť se dvěma skrytými vrstvami o velikostech 5 a 6 je "5,6"'.

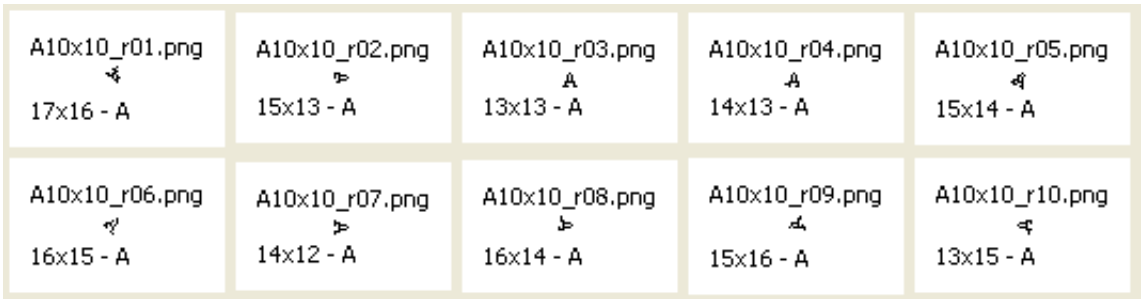
- *Startovací řád* nastavuje počáteční vnitřní řád perceptronů, ze kterých se skládá adaptivní strom. Doporučujeme hodnotu 1 nebo 2. Vyšší hodnoty výrazně zvyšují paměťové a výpočetní nároky.

- *Maximální řád* nastavuje atribut n_{max} z algoritmu pro učení perceptronu (viz kapitola 5.1.2). Jde o nejvyšší povolený vnitřní řád uzlu v adaptivním stromu.

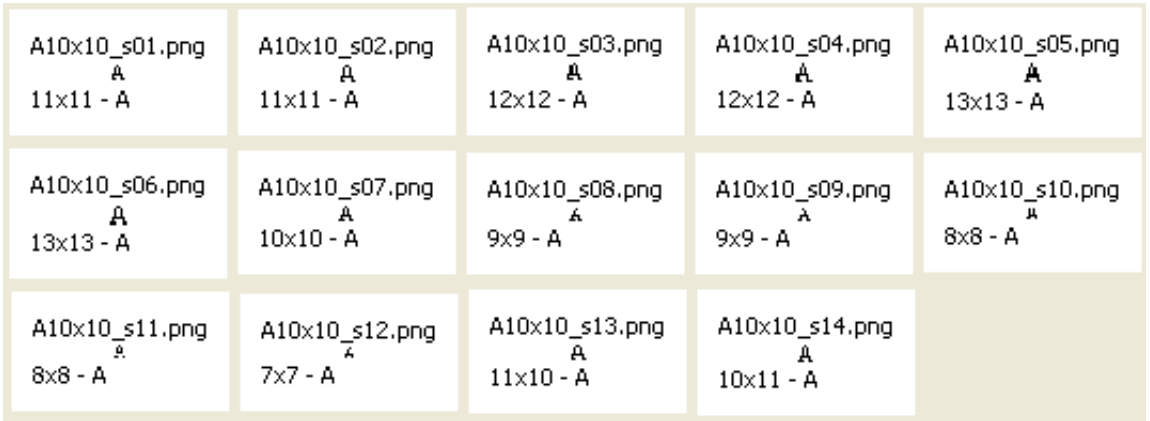
Všechny parametry modifikovaného modelu *AHONT* se shodují s původním. Výklopnou nabídkou se volí jeden z předešlých modelů pro použití uvnitř modelu *AHONT* (viz kapitola 5.2.2). Pomocí tlačítek *Přidat úroveň* a *Odstranit poslední úroveň* lze nastavit různé konfigurace pro jednotlivé úrovně stromu.

Příloha E - Ukázky vzorů

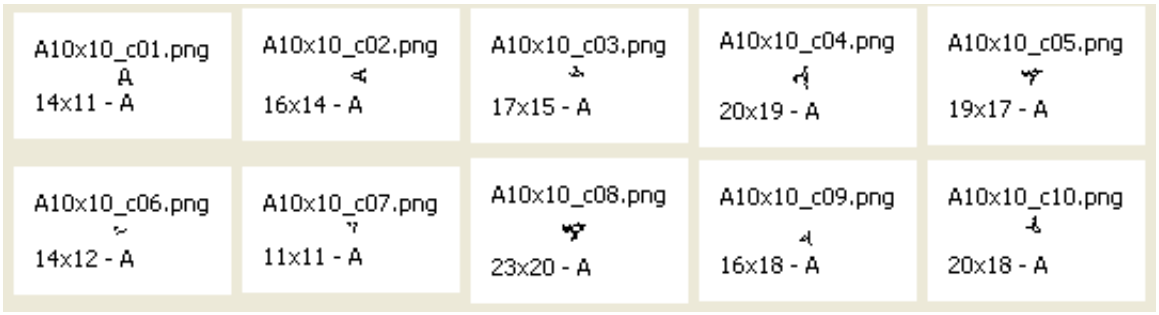
V následující příloze je možné vidět ukázky vzorů, které byly jednotlivým modelům předkládány pro testování. Na následujících obrázcích jsou vzory vzniklé deformací znaku „A“ v rozlišeních 10x10, 18x18 a 30x30.



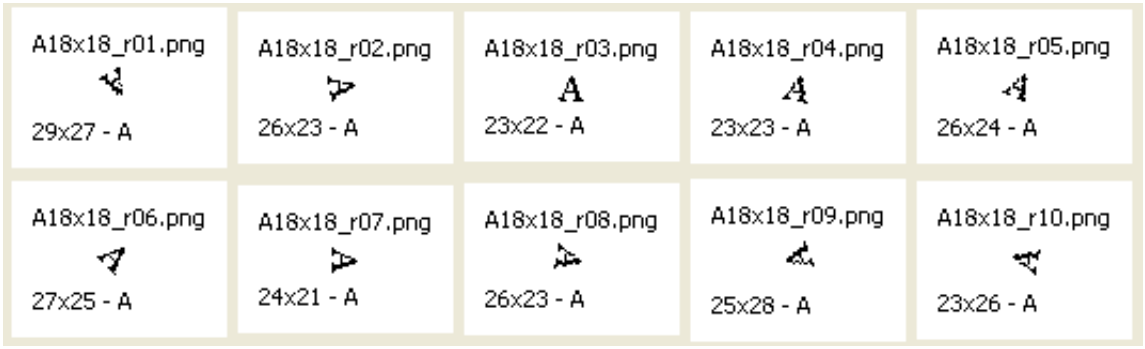
Obrázek 7.12: Vzory vzniklé otočením vzoru „A“ v rozlišení 10x10



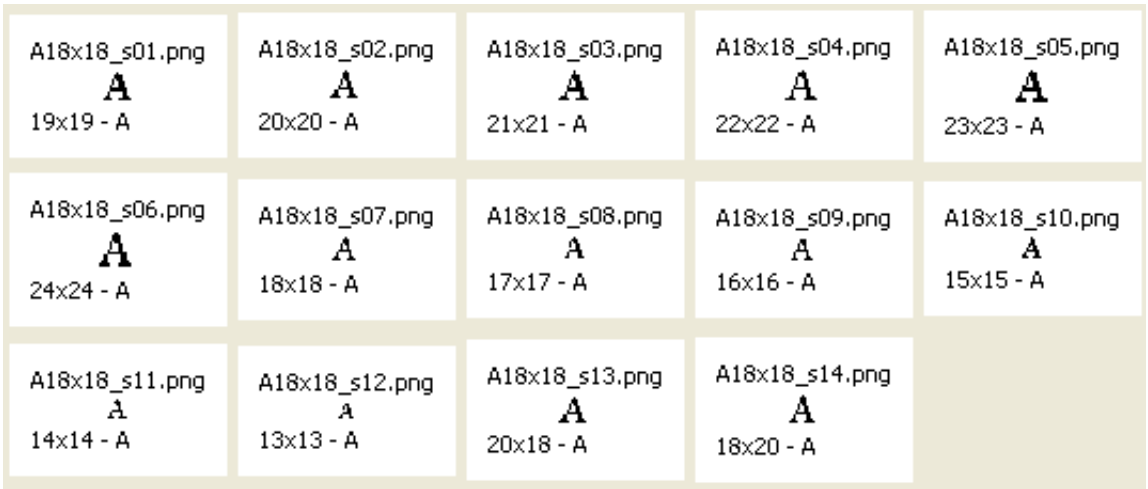
Obrázek 7.13: Vzory vzniklé změnou velikosti vzoru „A“ v rozlišení 10x10



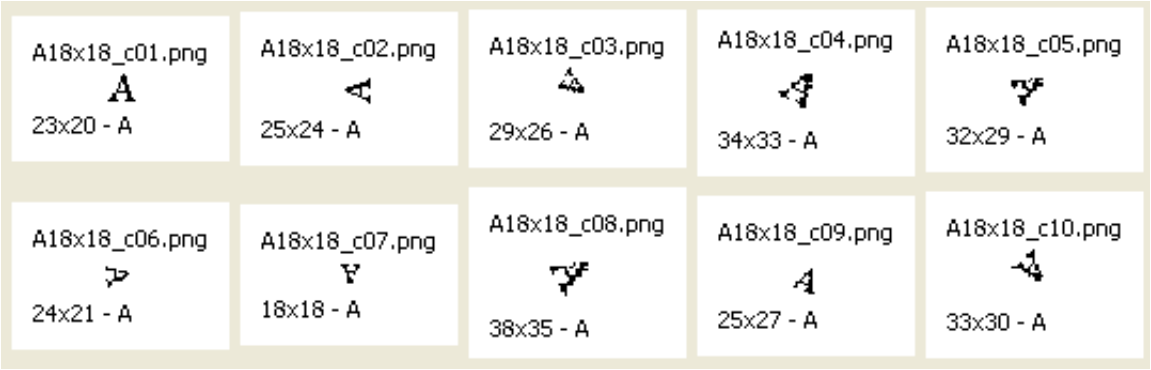
Obrázek 7.14: Vzory vzniklé kombinací deformací vzoru „A“ v rozlišení 10x10



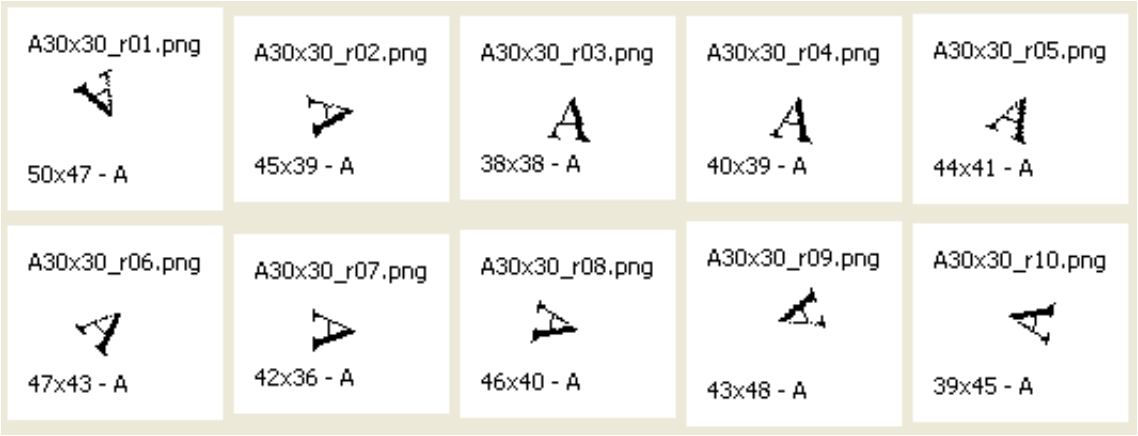
Obrázek 7.15: Vzory vzniklé otočením vzoru „A“ v rozlišení 18x18



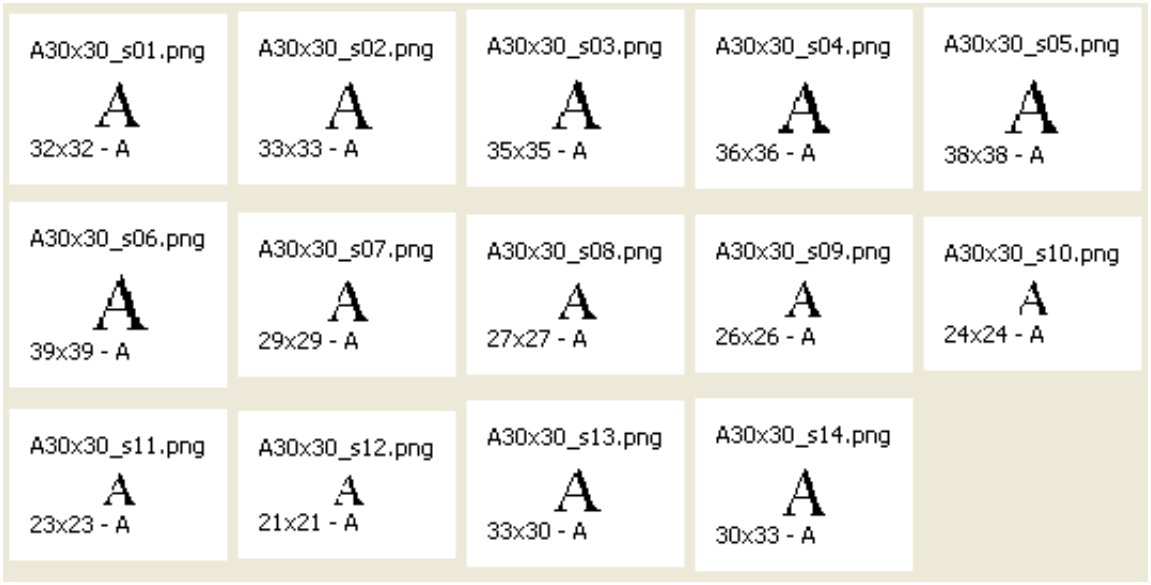
Obrázek 7.16: Vzory vzniklé změnou velikosti vzoru „A“ v rozlišení 18x18



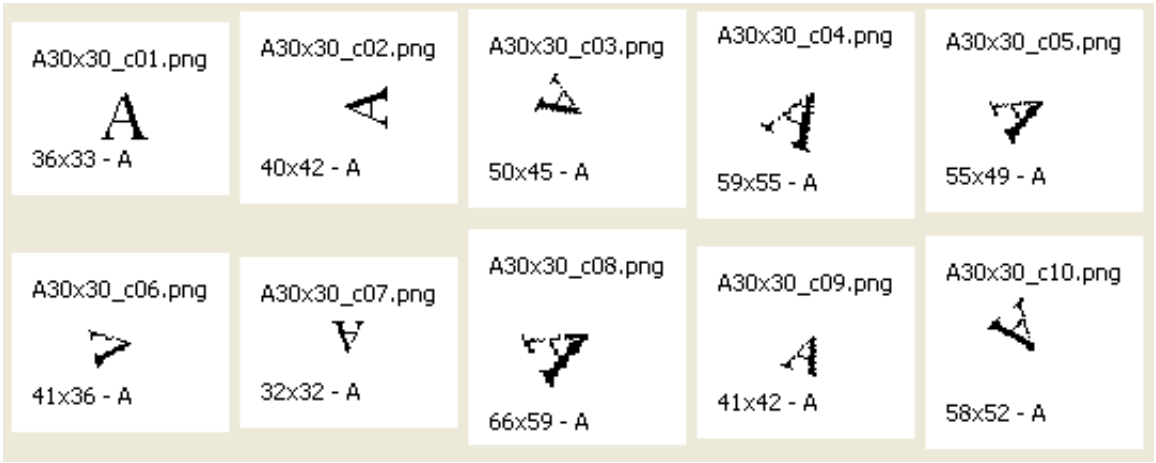
Obrázek 7.17: Vzory vzniklé kombinací deformací vzoru „A“ v rozlišení 18x18



Obrázek 7.18: Vzory vzniklé otočením vzoru „A“ v rozlišení 30x30



Obrázek 7.19: Vzory vzniklé změnou velikosti vzoru „A“ v rozlišení 30x30



Obrázek 7.20: Vzory vzniklé kombinací deformací vzoru „A“ v rozlišení 30x30