VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

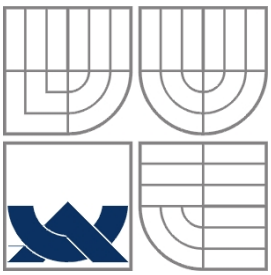
ROZPOZNÁVÁNÍ OBJEKTŮ POMOCÍ NEURONOVÝCH SÍTÍ

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

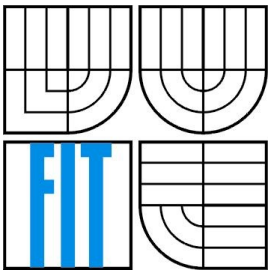
AUTOR PRÁCE
AUTHOR

JAROSLAV MARÁK

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

ROZPOZNÁVÁNÍ OBJEKTŮ POMOCÍ NEURONOVÝCH SÍTÍ

OBJECT RECOGNITION BY NEURAL NETWORKS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

Jaroslav Marák

VEDOUCÍ PRÁCE
SUPERVISOR

Doc. Ing. František V. Zbořil, CSc.

BRNO 2010

Abstrakt

Tato práce se zaměřuje na neuronové sítě a jejich klasifikační schopnosti při rozpoznávání objektů v obraze. Pro rozpoznávání je zde použito vícevrstvé dopředné neuronové sítě, trénovatelné pomocí algoritmu zpětného šíření chyby - Back Propagation. V práci je zmíněna problematika volby topologie takovéto sítě a významných parametrů ovlivňujících průběh učení dopředné sítě. Pomocí série experimentů rozpoznávání objektů v různých úlohách jsou prezentovány dosažené výsledky spolu se zhodnocením úspěšnosti.

Abstract

This thesis is focused on neural networks and their classification capability in object recognition tasks. For recognition is there used neural networks with feedforward architecture which is learned by Back Propagation algorithm. We discusses about problems which appear while a choosing topology of network or using various lerning-significant parametters while a learning process. Achieved results are presented in experiments with estimation.

Klíčová slova

Neuron, dopředná neuronová síť, RBF, LVQ, SOM, algoritmus Back Propagation, předzpracování obrazu, segmentace obrazu, rozpoznávání objektů.

Keywords

Neuron, feedforward neural network, RBF, LVQ, SOM, Back Propagation algorithm, image preprocessing, image segmentation, object recognition.

Citace

Jaroslav Marák: Rozpoznávání objektů pomocí neuronových sítí, bakalářská práce, Brno, FIT VUT v Brně, 2010

Rozpoznávání objektů pomocí neuronových sítí

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením
Doc. Ing. Františka V. Zbořila, CSc.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jaroslav Marák
26.března 2010

Poděkování

Rád bych poděkoval vedoucímu této práce za připomínky a konstruktivní kritiku. Rád bych také poděkoval svým přátelům, s jejichž pomocí bylo možné realizovat jeden z uváděných experimentů a především své rodině za podporu.

Obsah

1 Úvod

Předmětem této bakalářské práce je uvést čtenáře do problematiky rozpoznávání objektů v obraze za použití umělých neuronových sítí, seznámit jej s možnostmi, které jejich použití přináší a popsat problémy, která jsou s použitím tohoto výpočetního modelu spojeny. Součástí této předkládané práce je aplikace, jenž umožňuje sestavit uživatelem definovanou podobu vícevrstvé dopředné neuronové sítě, realizovat její učení a nabízí několik možností, jak otestovat její schopnosti klasifikovat.

První kapitola s názvem Úvod stručně popisuje obsah jednotlivých kapitol a seznamuje čtenáře s obsahem této bakalářské práce. Ve druhé kapitole jsou vysvětleny některé důležité pojmy, na které je v dalším textu odkazováno. Třetí kapitola se zabývá snímáním a předzpracováním obrazu, dále také segmentačními technikami, které bývají v praxi používány. Čtvrtá kapitola se věnuje teorii neuronových sítí. Od popisu základních pojmů této domény pokračuje k vysvětlení vlastností dopředné architektury umělých neuronových sítí s více vrstvami a dále pojednává o principech sítí RBF, SOM a LVQ. Pátá kapitola popisuje zaměření zbylých kapitol a stanovuje implementační cíle a požadavky na nástroj, který je součástí díla této práce. Šestá kapitola hovoří o implementačních přístupech a možnostech implementovaného nástroje, s jehož pomocí je možné zkoumat vlastnosti umělých neuronových sítí. Sedmá kapitola ověřuje uváděné teoretické předpoklady na experimentech a prezentuje dosažené výsledky. V závěrečné kapitole je nabídnuto zhodnocení dosažených výsledků a návaznost na jiná témata spojená s využitím umělých neuronových sítí.

2 Pojmy

Tato kapitola popisuje některé důležité pojmy (převzato z [1], dále čerpáno z [2], [5]), vázané zejména na počítačovou grafiku a zpracování obrazu, které jsou dále používány v kontextu následujících kapitol.

2.1 Vektorová grafika

Vektorová grafika je způsob popisu zpracovávané a zobrazované informace ve formě skupiny vektorových entit (úsečky, kružnice, křivky, polygony, atd.). Jednotlivé entity jsou definovány přesně matematicky (geometrie entit, souřadnice bodů, atd.) a parametricky (barvy, styl a tloušťka čar, atd.). Tento popis dat získáme manuálně nebo syntézou (generováním nebo převodem z jiného popisu).

2.2 Rastrová grafika

Rastrová grafika je způsob popisu zpracovávané a zobrazované informace ve formě rastrové matice (2D nebo 3D), diskrétně. Tento popis dat získáme: manuálně, syntézou (generováním nebo převodem z jiného popisu), nebo snímáním (kamerou atd.). Jeden prvek matice nazýváme pixel (2D) nebo voxel (3D)

2.3 Barevný model RGB

Model RGB je základním barevným modelem v oblasti počítačové grafiky. Odpovídá aditivnímu skládání barev. V tomto modelu jsou zastoupeny barvy červená, zelená a modrá.

2.4 Redukce barevného prostoru

Zmenšení barevného rozsahu je často vyžadováno v případech, kdy výstupní zařízení, např. tiskárna, nemá k dispozici odpovídající barevný rozsah. Podobně tomu může být v případech, kdy je třeba zmenšit velikost dat, nebo jak je tomu v našem případě, pomáhá redukce barevného prostoru snížit požadavky na výpočetní výkon při zpracovávání obrazu neuronovou sítí, resp. zejména při jejím učení.

2.4.1 Grayscale

Tímto termínem se označuje rastrový obraz ve stupních šedi. Zpravidla jde o obraz obsahující 256 úrovní šedé, což odpovídá 8 bit vzorkování. Pro získání rastru v této reprezentaci z plně barevného 24 bit modelu RGB lze použít empirický vztah 2.1.

$$I = 0.299R + 0.587G + 0.114B \tag{2.1}$$

2.4.2 Prahování

Prahováním se rozumí proces získání dvou hodnotové interpretace (černá, bílá) rastrového obrazu z původního rastru ve stupních šedi. Každý pixel zdrojového obrazu je porovnáván s prahem. Je-li porovnávaná hodnota menší, než práh, bude výsledný pixel černý, v opačném případě bílý. Pokud je tato metoda použita, lze se častěji setkat s některou její modifikací, např. adaptivní prahování.

2.5 Histogram

Pod pojmem histogram je možné si představit graf, který charakterizuje četnost výskytu jednotlivých hodnot pixelů v rastrovém obraze. Vodorovná osa je nositelem informace o hodnotě odstínu barevné složky pixelu, vertikální osa reprezentuje počet výskytů. O použití histogramu se opírají některé metody spojené se segmentací obrazu.

2.6 Diskrétní konvoluce

Diskrétní konvoluce je matematická operace definovaná vztahem [5]:

. $$y[n] = x[n] \star h[n] = \sum_{k=-\infty}^{+\infty} x[k]h[n-k] \tag{2.2}$$

Uváděný vztah ale není vhodný pro použití v oblasti počítačové grafiky. Pro tento případ se používá tzv. 2D konvoluce, kterou lze reprezentovat následujícím vzorcem [2]:

$$I(x,y) * h(x,y) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} I(x-i, y-j)h(i,j) \tag{2.3}$$

Funkce I(x, y) reprezentuje diskrétní obraz, funkce h(x, y) se nazývá konvoluční jádro. Tento matematický aparát se v oblasti zpracování grafiky uplatňuje jako frekvenční filtr. S jeho pomocí lze v obraze zdůrazňovat vysoké frekvence (detekce hran), nebo je naopak potlačovat (redukce šumu).

3 Získání a zpracování obrazu

Tato kapitola popisuje etapy, které předcházejí fázi rozpoznávání a mají rozhodující vliv na kvalitu celého procesu. Obraz získaný za pomoci kamery nebo jiného zařízení, totiž bývá často nutné vhodným způsobem upravit. Jde o tzv. předzpracování. Míra předzpracování souvisí s kvalitou získaného vstupního snímku. Nad upraveným obrazem je pak vykonáván proces segmentace. V tomto kroku probíhá hledání zájmových objektů v obraze. Posledním krokem je tzv. rozpoznávání, neboli klasifikace. V následujících podkapitolách budou popsány některé přístupy ke zpracování a segmentaci obrazu v souvislosti s aplikacemi počítačového vidění a rozpoznávání.

3.1 Získání obrazu

Získáním obrazu se rozumí proces sejmutí a digitalizace informace scény nebo oblasti reálného světa. Zdrojem informace je zachycené elektromagnetické vlnění, které dopadá na senzor snímacího zařízení, kterým může být podle potřeby kamera, fotoaparát, případě skener nebo jiný hardware. Získaná informace má analogovou podobu a pro další zpracování je třeba ji digitalizovat. Tento postup spočívá ve vzorkování do podoby matice M x N bodů a kvantování jasové informace do K intervalů [3]. Množství zachycené informace (detailů) v obraze zajišťuje vhodně zvolené vzorkování a kvantování. Čím vyšší je vzorkovací frekvence a počet kvantizačních úrovní, tím lépe je aproximován původní analogový obraz snímače [3]. O problematice vzorkování pojednává Shannonův-Nyquistův-Kotělnikův teorém [4]. Věta říká, že frekvence vzorkování musí být dvojnásobná, než je požadovaná zachycená nejvyšší hodnota frekvence snímaného signálu. V kontextu digitálního obrazu odpovídá jednomu vzorku tzv. pixel, který je dále nedělitelný. Pixel lze popsat jeho polohou v matici M x N a hodnotou, která nese informaci o jasu. Pro uložení této informace se běžně používá 8 bitů. To odpovídá 256 kvantizačním hodnotám. Pokud je snímaný obraz barevný, pak je třeba uchovávat hodnotu o jasu každé barevné složky zvlášť. Nejčastěji jde o interpretaci barevného modelu RGB, tomu odpovídá 24 bitů potřebných pro uložení takovéto informace.

V současné době jsou jako snímací prvky digitálních zařízení populární tzv. CCD snímače. Jejich prostřednictvím lze získat obraz v řádech Mpx. Tyto prvky bývají vyráběny technologií MOS. S výrobním postupem je dána také citlivost těchto prvků a s tím spojený výskyt šumu. Právě šum bývá považován za nežádoucí jev a bývá často nutné, pro úspěšnost dalšího zpracování, jeho výskyt omezit. Podobně vlastnosti tvaru senzoru a vlastnosti přítomné optické soustavy v zařízení – i tyto mohou být důvodem pro dodatečné změny v obraze, které je vhodné provést pro další zpracování.

3.2 Předzpracování obrazu

Předzpracování obrazu je proces složený z řady vhodně vybraných postupů, které ze vstupního nedokonalého obrazu vyprodukují obraz s lepšími vlastnostmi, který je pak vhodný pro zpracování některou segmentační technikou, nebo jinak lépe využitelný na rozdíl od původního obrazu. Běžně se pomocí předzpracování provádí např. odstraňování šumu, odstraňování zkreslení v obraze, nebo zvýraznění hran. Jak uvádí literatura [3], lze předzpracování z hlediska zaměření rozdělit takto:

3.2.1 Bodové jasové transformace

Do skupiny těchto metod patří tzv. jasové korekce, kterých se využívá při nerovnoměrném osvětlení. Je-li k dispozici ideální převodní charakteristika, lze vypočítat odchylku od normálu a pro každý bod vypočítat korekci.

3.2.2 Geometrické transformace

Jde o operace, které jsou prováděny nad obrazy zatíženými geometrickým zkreslením. K těmto úkazům dochází v případech použití širokoúhlého snímače, dále pokud je úhel optické osy snímače a snímané plochy jiný, než 90°, nebo jsou-li snímané objekty jiné, než plošné. Jak uvádí [3], lze geometrickou transformaci Tg plošného obrazu popsat matematicky jako vektorovou funkci, která transformuje bod v rovině (x, y) do bodu (x', y'), viz uváděné složkové vztahy 3.1:

$$x' = T_x(x,y) , \qquad y' = T_y(x,y) . \tag{3.1}$$

Prvním krokem této transformace je tzv. plošná transformace, která hledá k bodu ve vstupním obraze odpovídající bod ve výstupním obraze. Bod ve vstupním obraze má zpravidla diskrétní souřadnice, zatímco bod ve výstupním obraze má souřadnice reálné, díky výpočtu.

Dalším krokem je nalezení úrovně jasu. Jako nejjednodušší metoda je v [3] uváděna interpolace jasu pomocí nejbližšího souseda., dále lineární, nebo kubická interpolace.

3.2.3 Lokální předzpracování, filtrace šumu, detekce hran

Lokální předzpracování je proces úpravy obrazu, který se soustředí na určité okolí právě upravovaného bodu (pixelu) ve vstupním obraze. Mezi časté operace patří vyhlazování obrazu – potlačením vyšších frekvencí a tzv. gradientní operace, které naopak vyšší frekvence zesilují. Pro účely obou možností bývá použito diskrétní 2D konvoluce (kap. 2.6).

Pro odstraňování šumu existuje mnoho lišících se postupů. Nevýhodou některých metod je, že jejich aplikováním dochází k degradaci celkového obrazu. To je případ metody obyčejného průměrování, která stanovuje novou hodnotu pixelu na základě aritmetického průměru hodnot okolních pixelů. Jako vedlejší efekt se ve výstupním obraze projevuje rozmazání hran. Konvoluční jádro této metody má v základním tvaru podobu (3.2). Konvoluční jádra (3.3) a (3.4) nabízí lepší vlastnosti při aproximaci šumu s Gaussovským rozložením.

$$h = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad h = \frac{1}{10}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad h = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \tag{3.2), (3.3), (3.4}$$

Problémy s rozmazáváním hran lze řešit například metodou filtrace pomocí mediánu [3]. V tomto případě je třeba seřadit hodnoty pixelů z řešeného okolí podle velikosti jasové složky. Hodnota výstupního pixelu bude potom odpovídat hodnotě uprostřed této řady. Pro dosažení lepších výsledků

je vhodné použít místo čtvercového okolí křížové, jak doporučuje [3]. Podobně lze místo mediánu zvolit jiné kritérium volby hodnoty, například minimum, maximum, nebo lineární kombinaci prvků. Velmi dobrých výsledků při eliminaci šumu lze dosáhnout pomocí metody rotující masky. Metoda pracuje na základě průměrování okolí. Od metody obyčejného průměrování se liší v tom, že do okolí zahrnuje jen ty body, které k němu pravděpodobně patří. Snaží se tedy vyhnout hranám, které tvoří nehomogenitu okolí. Podrobnosti je možné čerpat v [3].

V některých situacích můžeme požadovat hrany zvýraznit. Pro tento účel opět mohou posloužit operátory založené na diskrétní konvoluci. Jmenujme například Robertsův operátor (3.5), Sobelův operátor (masky pro 2 z osmi směrů viz matice 3.6), Laplaceův gradientní operátor (pro 4-sousedství a 8-sousedství viz matice 3.7). Jinou zvláštní skupinu tvoří operátory, které hledají inflexní bod ve druhé derivaci obrazové funkce. Tyto jsou na rozdíl od zmiňovaných tří operátorů nezatíženy případným výskytem šumu ve vstupním obraze. Více lze opět čerpat v [3].

$$h1=\begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix} \quad h2=\begin{bmatrix}0 & 1\\ -1 & 0\end{bmatrix} \tag{3.5}$$

$$h1=\begin{bmatrix}1 & 2 & 1\\ 0 & 0 & 0\\ -1 & -2 & -1\end{bmatrix} \quad h2=\begin{bmatrix}0 & 1 & 2\\ -1 & 0 & 1\\ -2 & -1 & 0\end{bmatrix} \tag{3.6}$$

$$h4=\begin{bmatrix}0 & 1 & 0\\ 1 & -4 & 1\\ 0 & 1 & 0\end{bmatrix} \quad h8=\begin{bmatrix}1 & 1 & 1\\ 1 & -8 & 1\\ 1 & 1 & 1\end{bmatrix} \tag{3.7}$$

3.3 Segmentace obrazu

Nejnáročnějším krokem je segmentace [2], [5]. V této etapě se v obraze hledají za pomocí vhodného algoritmu segmenty, které představují kandidáty k objektům skutečného světa. Ve výsledku jde o nepřekrývající se oblasti v obraze.

Segmentaci lze rozdělit na úplnou a částečnou [5]. O úplné segmentaci hovoříme, pokud vytvořené segmenty přímo souhlasí s objekty ve vstupním obraze. V případě, kdy přímo nesouhlasí, jde o částečnou segmentaci. Ta je založena na homogenitě oblasti, která je charakterizovaná např. stejnou barvou, jasem, texturou apod. Výsledky procesu částečné segmentace je nutné zpřesnit aplikací dalších algoritmů.

Kvalita segmentační techniky je významnou měrou ovlivněna dokonalostí předzpracování obrazu, které je třeba věnovat velkou pozornost. Různé metody kladou jiné požadavky na vstupní obraz. Společným cílem všech metod segmentace je redukce zpracovávaného množství dat. Z hlediska přístupu, jak uvádí [2], je možné segmentační techniky rozdělit takto:

3.3.1 Metody vycházející z detekce hran

Hrany v obraze vznikají rozdílem hodnot okolních pixelů. Pro účely detekce hran se používají tzv. hranové detektory o kterých již bylo zmíněno v kapitole 3.2.3. Tyto detektory lze rozdělit podle principu na detektory založené na využití první derivace, kdy je průběh této derivace srovnáván s prahem, jehož překročení indikuje přítomnost hrany. A v druhém případě na metody založené na využití druhé derivace, kdy se naopak hledá průchod nulou. Mimo uváděné (3.5), (3.6), (3.7) můžeme dále jmenovat operátory Frei-Chen, Prewitt, operátor diference pixelů, atd. [2], které jsou založeny na detekci pomocí první derivace. Mezi operátory, které fungují na principu detekce průchodu nulou druhé derivace lze zařadit operátor Marra a Hildrethové, nebo Cannyho hranový detektor [3].

Mezi další přístupy založené na detekci hran patří[2]: Houghova transformace, metoda tvarování kontur, Level-sets, Isosurfaces, případně další.

3.3.2 Metody na principu regionů v obraze

Tato skupina metod úzce souvisí s předchozím zmiňovaným přístupem. Vychází z předpokladu, že hrany ohraničující region tvoří oblasti. V porovnání s předchozí metodou jsou výsledky segmentace v praxi často rozdílné. Dobrých výsledků lze dosáhnout kombinací obou přístupů. Určování oblastí je založeno na homogenitě oblasti. Tu lze chápat jako odstín, nebo barvu. V komplexních případech se může jednat například o stejnou texturu [3].

3.3.3 Statistické metody

Nejjednodušším přístupem v této skupině metod je segmentace prahováním. Tato metoda je založena na využití histogramu a vhodném způsobu volby prahu. Předpokládá také rovnoměrné rozložení světla ve vstupním obraze.

Odvozenou metodou je adaptivní prahování, jenž se využívá v případech, kdy osvětlení ve vstupním obraze není rovnoměrné. V takovém případě je rastr rozdělen na několik oblastí, ve kterých je zvlášť vypočtena vhodná hodnota prahu.

Dalšími přístupy statistických metod [2] jsou Connected Component Labeling, Apmplitudová projekce, Shluková analýza, segmentace za použití kohonenových map, Fuzzy Connectedness, MRF, případně další.

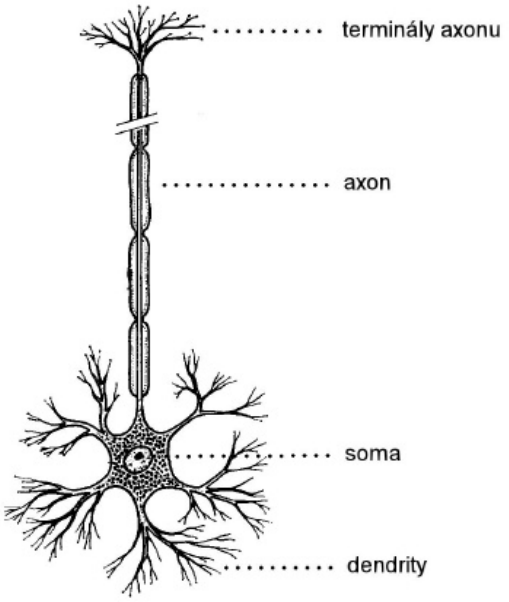
3.3.4 Hybridní metody

Metody této kategorie v sobě kombinují některé vlastnosti z předcházejících tří skupin zmiňovaných metod. Proto je nemožné je do nich přímo zařadit. Jmenovitě jde o techniky Watershed Transform a o techniku založenou na použití umělé neuronové sítě (dále jen UNS – **U**mělá Neuronová Síť). Druh použité UNS se odvíjí od způsobu, jakým požadujeme síť učit. První variantou jsou sítě s *učením bez učitele*. V tomto případě jde o sítě SOM (Self Organizing Maps). Druhou variantou způsobu učení je *učení s učitelem.* Tato technika se vztahuje na vícevrstvé sítě s RBF (Radial Basis Function). O neuronových sítích a přístupech k učení budeme dále hovořit v následující 4. kapitole.

4 Neuronové sítě

4.1 Biologický neuron

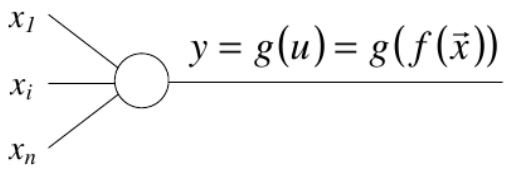
Činnost umělého neuronu plyne z poznatků o chování biologického neuronu [7], [8], který tvoří tkáň šedé kůry mozkové. Biologický neuron se skládá z několika význačných částí. Vstupy jsou tvořeny krátkými výběžky, tzv. dendrity, které ústí do těla neuronu, zvaného soma. Výstup neuronu je reprezentován jedním dlouhým výběžkem, tzv. axonem, který je zakončen terminály, jenž realizují spojení s ostatními neurony. Tato spojení jsou označována jako synapse a lze je rozdělit na dvě skupiny. První skupinu tvoří tzv. inhibiční synapse. Ty způsobují potlačení (útlum) vzruchu při jeho přenosu do zbytku nervové soustavy. Naopak excitační synapse dovoluje signálu, aby se šířil dále. Právě povahou synaptických vazeb jednotlivých neuronů na cestě mezi receptorem (nervová zakončení registrující tepelné, světelné nebo jiné podněty) a efektorem (příslušný orgán, reagující na podnět) pravděpodobně vzniká paměťová stopa [6]. Překonáním jistého prahu (dosažením určité hodnoty vybuzení) neuronu prostřednictvím jeho vstupů, pak neuron reaguje vytvořením impulzu, který putuje jeho axonem. S každým vybuzením se současně mění paměťová propustnost neuronu.



Obrázek 4.1 – Biologický neuron (převzato z [6])

4.2 Matematický model neuronu

Z matematického pohledu lze neuron popsat vstupním vektorem *x,* váhovým vektorem *w* a jedním výstupem *y*. Ze vstupního vektoru a vektoru vah je počítán tzv. vnitřní potenciál *u* neuronu.



Obrázek 4.2 - Obecný model neuronu (převzato z [7])

Způsob jeho výpočtu je dán druhem bázové funkce *f*, jenž může být lineární viz vzorec 4.1, nebo radiální viz vzorec 4.2. (Oba vztahy byly převzaty z [7].)

$$u = \sum_{i=1}^{n} w_i x_i \tag{4.1}$$

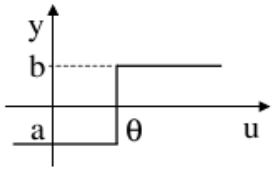
$$u = \|\vec{x} - \vec{w}\| = \sqrt{\sum_{i=1}^{n} (x_i - w_i)^2} \tag{4.2}$$

Výstup neuronu *y* je definován aktivační funkcí *g,* která jako argument přebírá hodnotu bázové funkce. Podle použité aktivační funkce je pak výstup neuronu z pohledu krajních stavů buď binární <0, 1>, nebo bipolární <-1, +1>. Z charakteru průběhu na něj lze také pohlížet jako na nespojitý, po částech spojitý, nebo spojitý. Volba aktivační funkce bývá spojena s volbou bázové funkce, proto se v praxi vyskytují jen určité kombinace, které si nyní uvedeme [7].

4.2.1 Aktivační funkce spojené s lineární bázovou funkcí

4.2.1.1 Nespojitá (skoková) aktivační funkce

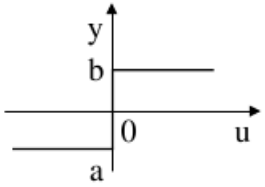
$$y^{new} = \begin{cases} a & pro \quad u < \theta \\ b & pro \quad u > \theta \\ y^{old} & pro \quad u = \theta \end{cases} \tag{4.3}$$



Pro binární výstup se hodnoty parametrů volí: a = 0, b = 1. Pro bipolární výstup: a = -1, b = 1.V praxi se vztah často zjednodušuje zavedením prahu na nultý index vektoru vah $\vec{w}$ a současně přivedením trvalé hodnoty 1 na nultý index vektoru $\vec{x}$.

$$u = \sum_{i=0}^{n} w_i x_i, \qquad w_0 = -\theta, \quad x_0 = 1 \tag{4.4}$$

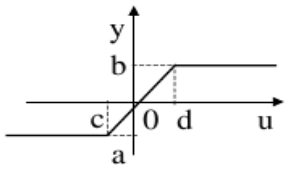
$$y^{new} = \begin{cases} a & pro \quad u < 0 \\ b & pro \quad u > 0 \\ y^{old} & pro \quad u = 0 \end{cases} \tag{4.5}$$



Poznámka.**:**
Vlivem úpravy dochází současně i ke změně vztahu pro výpočet vnitřního potenciálu neuronu, kde na rozdíl od vztahu 4.1 dochází k sumaci součinů od nultého indexu (viz vztah 4.4).

4.2.1.2 Po částech spojitá aktivační funkce

$$y^{new} = \begin{cases} a & pro \quad u < c \\ b & pro \quad u > d \\ a + \dfrac{(b-a)(u-c)}{d-c} & pro \quad c \le u \le d \end{cases} \qquad (4.6)$$



4.2.1.3 Spojitá aktivační funkce

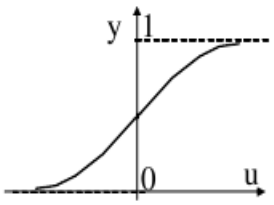
Sigmoidální funkce je definována vztahem (4.7) pro obvyklé hodnoty parametrů jejího obecného vyjádření, které neuvádíme. Její použití odpovídá binární reprezentaci výstupu neuronu. Funkce hyperbolického tangentu je podobně zastoupena vztahem (4.9). Tato funkce definuje bipolární reprezentaci výstupu. První derivace obou funkcí uvádějí vztahy (4.8) a (4.10). Odvození matematického vyjádření prvních derivací je možné v případě potřeby dohledat v [7], tento postup zde neuvádíme.

$$y = \frac{1}{1+e^{(-\lambda u)}} \qquad (4.7)$$

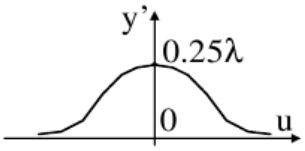
$$y' = \frac{dy}{du} = \lambda y(1-y) \qquad (4.8)$$

$$y = tgh(\lambda u) \qquad (4.9)$$

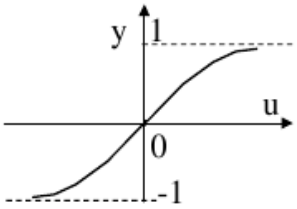
$$y' = \frac{dy}{du} = \lambda(1-y^2) \qquad (4.10)$$



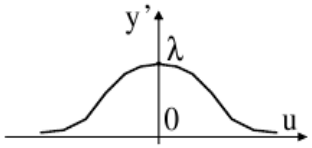
Obrázek 4.11 – Průběh sigmoidální funkce



Obrázek 4.12 – Průběh první derivace sigm. funkce



Obrázek 4.13 – Průběh funkce tgh(x)

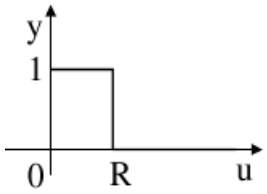


Obrázek 4.14 – Průběh deriv. funkce tgh(x)

4.2.2 Aktivační funkce spojené s radiální bázovou funkcí

4.2.2.1 Nespojitá aktivační funkce

$$y = \begin{cases} 1 & pro \quad u \le R \\ 0 & pro \quad u > R \end{cases} \tag{4.15}$$



4.2.2.2 Spojitá aktivační funkce

$$y = e^{-(u/\sigma)^2} \tag{4.16}$$

Poznámka:
Uvedená vyjádření 4.3, 4.4, 4.5, 4.6, 4.15, 4.16 a obrázky 4.11, 4.12, 4.13, 4.14 byly převzaty z [7].

4.3 Organizace neuronů do sítí

Umělé neuronové sítě (dále jen UNS) nacházejí své uplatnění především u problémů, které není možné matematicky popsat, nebo jejichž popis by byl velmi složitý nebo algoritmicky náročný [9].

Výpočetní síla neuronů plyne z jejich vhodného vzájemného propojení. Toto propojení nemůže být libovolné, neboť v současné době neexistuje způsob, jak pro takovou síť navrhnout učící algoritmus [7]. V praxi se proto používají jen určité topologie. Příkladem sítě pro kterou neexistuje učící algoritmus je plně propojená síť. Naopak mezi používané architektury UNS patří plně propojená symetrická síť, u které je vazba mezi dvěma neurony v obou směrech stejná (např. Hopfieldova síť). Nejčastěji vyskytujícím se typem sítě je tzv. dopředná neuronová síť, která je i tématem této práce.

4.3.1 Učení neuronových sítí

K učení neuronové sítě přistupujeme nejčastěji na základě induktivní metody, kdy vyvozujeme všeobecně platné závěry na základě pozorování množiny jevů [9]. Tento systematický přístup rozeznává dva možné způsoby učení, které závisí na tom, zda do tohoto procesu vstupuje lidský faktor. Prvním způsobem učení je učení s učitelem, kdy síti předkládáme vstupní hodnoty, nejčastěji vektory, a současně pro každý z nich definujeme požadovaný výstupní vektor. Druhým způsobem je učení bez učitele, kdy systém nemá informaci o požadovaném výstupu, ale odvozuje si ji prostřednictvím zpětné vazby [9].

4.3.1.1 Hebbův zákon učení

Základní pravidlo učení definoval v roce 1949 Donald Hebb [7], [9]. Tento zákon učení popisuje způsob modifikace hodnot synaptických vah v nervových systémech živých organizmů a analogicky se uplatňuje i v UNS. Toto pravidlo říká: Jsou-li dva mezi sebou propojené neurony ve stejný okamžik vybuzeny (aktivní), pak se synaptická vazba mezi nimi posílí, tzv. excitace. Pokud jsou tyto neurony ve stejný okamžik oba neaktivní, dochází k potlačení této synaptické vazby, tzv. inhibice. V případě, kdy jeden z neuronů je aktivní a druhý neaktivní, zůstane synaptická vazba mezi nimi nezměněna. Uváděný zákon Hebbova učení lze reprezentovat vztahem (4.17) [9].

$$\Delta w_{ij} = \alpha \cdot x_i(k) \cdot x_j(k) \tag{4.17}$$

Parametr α definuje rychlost učení, hodnota x_j představuje presynaptický stav neuronu, x_i zastupuje postsynaptický stav neuronu.

4.3.1.2 Učení bez učitele

Tento způsob učení je založen na předkládání vzorů vstupních dat síti, jenž sama dokáže tyto vzory třídit podle jejich podobnosti. Při učení tedy chybí informace o požadované hodnotě výstupu ke každému předkládanému vstupnímu vzorku. Podobné vzory se shlukují do tzv. shluků, neboli map. Tento princip se nazývá též samoorganizace a vyskytuje se i v některých částech mozku. Typickým zástupcem sítě s touto formou učení je například Kohonenova Síť (SOM).

4.3.1.3 Učení s učitelem

Učení s učitelem je způsob, jakým lze u UNS docílit požadované hodnoty výstupního vektoru v závislosti na přiloženém vstupním vektoru. Proto obsahuje trénovací množina dvojice, které pro každou hodnotu vstupního vektoru definují požadovaný výstupní vektor. Matematický zápis podoby trénovací množiny vyjadřuje (4.18)[7].

$$T = \{(\vec{i}_1, \vec{d}_1), (\vec{i}_2, \vec{d}_2), \ldots, (\vec{i}_n, \vec{d}_n)\} \tag{4.18}$$

$\vec{i}$ … n-rozměrný vstupní vektor sítě
$\vec{d}$ … m-rozměrný požadovaný výstupní vektor sítě

Učení spočívá ve změně vektorů vah sítě tak, aby rozdíl mezi skutečným výstupem sítě a požadovaným výstupem byl co nejmenší. Jinými slovy hledáme minimum chybové funkce, kterou lze definovat vztahem (4.19). Velikost změny váhy je úměrná velikosti chyby [9].

$$E(s) = \sum_{i=1}^{N} [\vec{y}_i(s,k) - \vec{d}_i(k)]^2 \tag{4.19}$$

s … pořadové číslo epochy trénování
k … pořadové číslo vzoru trénovací množiny
$\vec{y}(s,k)$ … skutečná hodnota výstupu daná odezvou sítě na vstupní vektor
N … počet vzorů trénovací množiny

4.4 Neuronové sítě vhodné pro rozpoznávání objektů v obraze

V dalším textu se budeme zabývat konkrétními typy neuronových sítí, které se hodí pro řešení problémů spojených s rozpoznáváním objektů v obraze a popíšeme principy, na kterých jsou založeny.

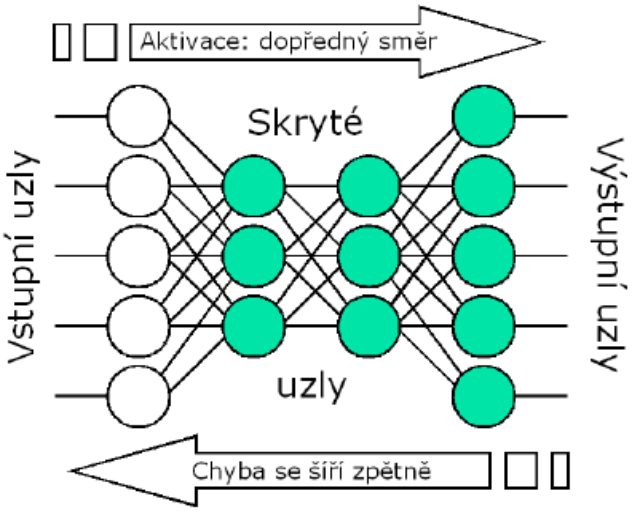
4.4.1 Vícevrstvá dopředná neuronová síť

Nyní se zaměříme na popis vlastností velmi často používané neuronové sítě, která se uplatňuje ve většině aplikací, jenž lze za pomoci UNS realizovat. Tento typ neuronové sítě lze využít pro řízení, jako funkční aproximátor, pro predikci, ztrátovou kompresi, jako asociativní paměť, nebo jako klasifikátor. Její významnou vlastností, na rozdíl například od perceptronové sítě, je schopnost zmodifikovat prostor, kde jednotlivé vzorky jsou lineárně neoddělitelné na prostor, kde vzorky jsou lineárně oddělitelné [7].

K učení tohoto typu UNS je přistupováno jako k učení s učitelem. Proces učení je realizován pomocí algoritmu zpětného šíření chyby – Back Propagation.

Tento typ sítě obsahuje neurony s lineární bázovou funkcí a se sigmoidální, nebo tangenciálně-hyperbolickou aktivační funkcí. Volba příslušné aktivační funkce ovlivňuje charakter výstupu, jak bylo zmíněno v kap. 4.2.

Neurony této sítě jsou organizovány do vrstev. První vrstva, jako jediná, není složena z neuronů, ale tvoří ji uzly s jednotkovým přenosem. Realizuje tedy pouze propojení s neurony v následující vrstvě. Za touto vrstvou následuje jedna, až dvě vrstvy skryté. Poslední vrstva je označována jako vrstva výstupní. Již z názvu (dopředná síť) je patrné, že mezi neurony téže vrstvy nejsou žádná spojení, ta existují pouze mezi následujícími vrstvami. Podobu této sítě ilustruje obrázek 4.15.



Obrázek 4.15 – Vícevrstvá dopředná síť (převzato z [11]).

4.4.1.1 Učení dopředné sítě

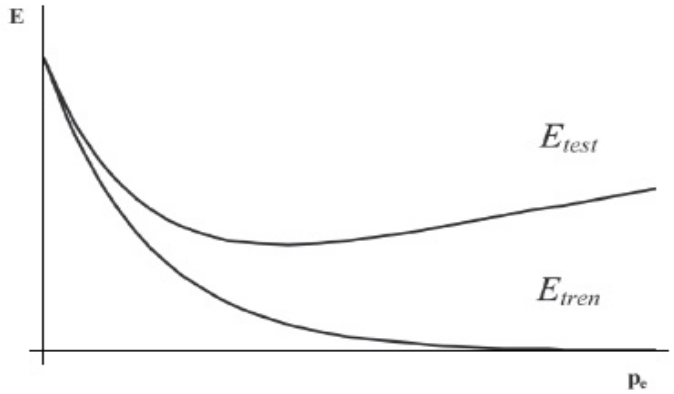
Pro učení dopředné sítě o více vrstvách se používá algoritmus Back Propagation. Jeho objevení v roce 1986 vědci Lumelhert a Lecum umožnilo vyřešit problém XOR, který byl pro neuronové sítě do té doby neřešitelným a byl nemalou příčinou k zastavení výzkumu UNS v tehdejší době [8]. V dnešní době je algoritmus Back Propagation nejpoužívanější metodou učení na bázi učení s učitelem.

Učení spočívá v postupném procházení vzorků trénovací množiny, jenž obsahuje dvojice (viz kap. 4.3.1.3) a hledání vektorů vah $\vec{w}$ neuronů, takových, aby výsledná chyba, daná rozdílem výstupního vektoru sítě a požadovaného výstupu (viz vztah 4.20), byla minimální. V prvním kroku algoritmu se předpokládá, že vektory vah všech neuronů ve všech vrstvách jsou vhodným způsobem náhodně inicializovány. O této inicializaci se zmíníme dále v kap. 4.4.1.2.

Podrobněji lze algoritmus rozložit na tři fáze: 1) Přiložení jednoho z prvků trénovací množiny (dále TRM) na vstup sítě a výpočet odezvy sítě, tj. výstupního vektoru $\vec{y}$. 2) Výpočet chyby výstupní vrstvy plynoucí ze vztahu (4.20). 3) Zpětné šíření chyby směrem od poslední vrstvy k vrstvě první a následná aktualizace váhových vektorů všech neuronů směrem od první vrstvy k poslední vrstvě sítě. Tento postup částečně nastiňuje obrázek 4.15. Základní varianta algoritmu Back Propagation provádí úpravu vah po každém vzorku TRM. Tuto vlastnost lze optimalizovat četností úprav vah po každé epoše. Pojem epocha v tomto kontextu znamená jeden průchod celou TRM. Tímto způsobem lze zvýšit efektivitu algoritmu a zkrátit celkový čas učení. Podmínkou pro ukončení algoritmu je dosažení požadované chyby, nebo překročení maximálního počtu epoch.

Velmi vhodné, v řadě případů, je zahrnout do celého procesu učení ještě tzv. testovací množinu (dále TSM). Tato množina obsahuje vzory, jenž se nevyskytují v TRM. S její pomocí dokážeme zjistit úspěšnost sítě reagovat na nenaučené vzory a ověřit tak její schopnost zobecňovat. Platí totiž, že dobře naučená síť na trénovací data nemusí vykazovat uspokojivou odezvu na data z TSM. Pokud UNS vykazuje takové vlastnosti, říkáme o ní, že je přeučená. Vyplatí se nám tedy přihlížet i k průběhu chyby testovacích dat a případně rozšířit algoritmus učení o podmínku ukončení při dosažení požadované hodnoty chyby testovacích dat. Chybu pro TSM lze počítat stejným vztahem, jako v případě trénovací množiny (vzorec 4.20 [7]). Průběh chyb TRM a TSM v závislosti na počtu epoch nastiňuje obrázek 4.16.

$$E=\frac{1}{2}\sum_{j=1}^{n_L}\left(d_j-{}^{L}y_j\right)^2 \tag{4.20}$$



Obrázek 4.16 – Průběhy kriteriální funkce (převzato z [10]).

Nyní se ještě vrátíme k algoritmu Back Propagation a popíšeme způsob, jakým provádí změnu váhových vektorů. Současně se zmíníme o dalších parametrech, které ovlivňují proces učení.

Výpočet nové hodnoty složky váhového vektoru ukazuje vztah 4.21 [7]. Výsledná nová hodnota položky váhového vektoru je dána součtem její původní hodnoty a vypočtené změny.

$$^{l}w_{ji} = {}^{l}w_{ji} + \Delta\,{}^{l}w_{ji} \tag{4.21}$$

l … vrstva
j … neuron příslušné vrstvy
i … položka váhového vektoru

Výpočet hodnoty změny váhy realizuje vztah 4.22 [7] jako součin koeficientu učení μ, hodnoty příslušného vstupu x_i a parametru delta neuronu.

$$\Delta\,{}^{l}w_{ji} = \mu \cdot {}^{l}\delta_j \cdot {}^{l}x_i \tag{4.22}$$

Výpočet parametru delta se provádí od konce sítě a jeho hodnota určuje změnu váhy ve směru záporného gradientu chyby. Hledá se tedy minimum chybové funkce. Postup odvození tohoto parametru je prezentován v [7] a zabývat se jím nebudeme. Uvedeme zde pouze výsledné vztahy. Vztah 4.23 odpovídá výpočtu parametru delta pro neurony výstupní vrstvy sítě, podobně vztah 4.24 odpovídá jeho výpočtu pro ostatní vrstvy sítě [7].

$$^{L}\delta_j = (d_j - y_j) \cdot \lambda \cdot {}^{L}y_j \cdot (1 - {}^{L}y_j) \tag{4.23}$$

$$^{l}\delta_j = \left(\sum_{k=1}^{n_l+1} {}^{l+1}\delta_k \cdot {}^{l+1}w_{kj}\right) \cdot \lambda \cdot {}^{l}y_j \cdot (1 - {}^{l}y_j) \tag{4.24}$$

Uváděné vztahy 4.23 a 4.24 jsou platné pro neurony se sigmoidální aktivační funkcí. Z uváděných vztahů je patrné, že jejich druhá polovina plyne z derivace sigmoidálního průběhu. Přesněji řečeno z parciální derivace výstupu neuronu podle jeho vnitřního potenciálu.

Pro úplnost uvedeme vztahy, které jsou platné pro UNS s tangenciálně-hyperbolickou aktivační funkcí, viz 4.24, 4.25 [7].

$$^{L}\delta_j = (d_j - y_j) \cdot \lambda \cdot (1 - {}^{L}y_j^2) \tag{4.25}$$

$$^{l}\delta_j = \left(\sum_{k=1}^{n_l+1} {}^{l+1}\delta_k \cdot {}^{l+1}w_{kj}\right) \cdot \lambda \cdot (1 - {}^{l}y_j^2) \tag{4.26}$$

4.4.1.2 Parametry ovlivňující učení a vlastnosti dopředné UNS

Nejprve se zaměříme na skryté vrstvy sítě. Z pohledu jejich počtu se nepoužívá více skrytých vrstev, nežli dvě. Větší počet se nijak neprojevuje. Nejméně však lze použít jednu skrytou vrstvu. Jak uvádí [10], počet skrytých vrstev volíme na základě složitosti řešeného problému. V řadě případů lze dosáhnout vyhovující aproximace za použití jen jedné skryté vrstvy.

Množství neuronů ve skrytých vrstvách ovlivňuje schopnosti sítě učit se a zobecňovat. Pro volbu jejich počtu neexistuje žádné pravidlo, lze vycházet pouze z doporučení a experimentů. Obecně ale vyplývá, že síť s velkým počtem neuronů nedokáže dobře zobecňovat a její učení, vzhledem k množství modifikovaných parametrů, trvá déle. Síť s nedostatečným počtem neuronů se naopak nedokáže naučit. Přibližný počet neuronů pro síť s jednou skrytou vrstvou lze odvodit například pomocí vzorce 4.27 [10]. Pokud jde o síť se dvěma vrstvami, lze jejich přibližný počet pro každou vrstvu odvodit na základě vztahů 4.28 [10].

$$p = \sqrt{n \cdot m} \tag{4.27}$$

$$p_1 = m \cdot r^2 \qquad p_2 = m \cdot r \tag{4.28}$$
$$\text{, kde} \quad r = \sqrt[3]{n/m}$$

Hodnota n odpovídá počtu vstupních uzlů a hodnota m počtu neuronů výstupní vrstvy sítě. Počet vstupních uzlů sítě, stejně jako počet neuronů výstupní vrstvy, volíme podle potřeby.

Další proměnnou, jenž ovlivňuje průběh učení, resp. jeho trvání, je volba počátečních hodnot váhových vektorů. Jejich počáteční hodnoty by měly být nastaveny za pomocí generátoru náhodných, nebo pseudonáhodných čísel ve vhodném intervalu. I v tomto případě neexistuje jednoznačné pravidlo, jak vhodný interval zvolit. Literatura [12] ,[7] doporučuje náhodný výběr z intervalu <-0,5;0,5>, nebo <-1;1>. Literatura [7] doporučuje jako další možnost interval <-3/n;3/n>, kde n je počet vstupních uzlů sítě.

Jestliže v průběhu učení měníme mimo váhové vektory sítě i hodnoty strmostí aktivačních funkcí λ (viz vztahy 4.7, 4.8, 4.9, 4.10), pak jejich počáteční hodnoty nastavujeme stejným způsobem, jako váhové vektory, za použití téhož intervalu. V opačném případě je doporučeno nastavit hodnotu $\lambda = 1$ [7].

Významným parametrem ve fázi učení je tzv. koeficient učení μ, jenž se podílí na výpočtu změny váhy ve vztahu 4.22. Jeho hodnota ovlivňuje rychlost učení. Často se volí z intervalu <0,1;0,9>, jak uvádí [7]. Pro malé hodnoty je proces učení zdlouhavý, při velkých hodnotách může v některých případech úloh dojít k přeskočení hledaného globálního minima chybové funkce. Hodnota koeficientu učení se v průběhu trénování nemění. V některých případech může být vhodné její hodnotu postupně snižovat.

Možnou optimalizací vztahu 4.22 je zavedení tzv. momentu (hybnosti) α. Jeho prostřednictvím se výpočet změny váhy neodvíjí jen od aktuálního gradientu, ale přihlíží se i k předchozí změně této váhy [7]. Uvedená optimalizace může pomoci vyřešit problém s uváznutím v lokálním extrému chybové funkce v průběhu učení. Hodnotu hybnosti α se doporučuje volit z intervalu <0;1> a v průběhu učení se udržuje konstantní. Podobu modifikovaného vztahu 4.22 se zavedením momentu ukazuje vztah 4.29.

$$^{l}\Delta w_{ji}(k+1) = \mu \cdot {}^{l}\delta_j \cdot {}^{l}x_i(k) + \alpha \cdot \Delta^{l} w_{ji}(k) \tag{4.29}$$

Jistou pozornost je třeba věnovat také trénovací množině, jenž lze definovat viz 4.18. V procesu učení je možné k jejím prvkům přistupovat buďto sekvenčně, nebo náhodně. Druhá možnost může vést ke zlepšení výsledků učení, jak uvádí [7]. Počet prvků trénovací množiny, množství modifikovaných parametrů sítě a požadovaná přesnost mezi sebou souvisí. Pro orientační výpočet počtu prvků trénovací množiny lze použít vztah 4.30 [7]:

$$P \geq \frac{n_w}{1-a} \cdot \log \frac{n_n}{1-a} \tag{4.30}$$

n_w … počet vah UNS
n_n … počet neuronů UNS
a … požadovaná přesnost <0;1>

4.4.2 Neuronové sítě RBF

Tato síť je svou strukturou podobná dopředné UNS. Rozdíly plynou z typů použitých neuronů v jednotlivých vrstvách. Vstupní vrstvu opět tvoří uzly s jednotkovým přenosem. Ve skryté vrstvě se nachází neurony s radiální bázovou funkcí, viz matematický vztah 4.2, odtud plyne zkrácené označení RBF. Tyto neurony současně implementují Gaussovu aktivační funkci viz vztah 4.31 (převzato z [7]).

$$y_k = e^{-\left(u_k / \sigma_k\right)^2} \tag{4.31}$$

Neurony této vrstvy, na rozdíl od vrstvy výstupní, nemají práh. Při učení neuronové sítě RBF se ve skryté vrstvě nastavují středy aktivačních funkcí u_k a strmost průběhů Gaussovy křivky, jenž ovlivňuje parametr σ_k . Jak uvádí [6], je vhodné volit počet neuronů této vrstvy vyšší, nežli je počet uzlů ve vstupní vrstvě. Následující výstupní vrstva je tvořena neurony s lineární bázovou funkcí viz vztah 4.1. Neurony této vrstvy obvykle nemají aktivační funkci a v průběhu učení se u nich nastavují pouze váhové vektory. Možné techniky učení sítí s RBF nyní nastíníme v další kapitole této práce.

4.4.2.1 Učení neuronové sítě RBF

Přístup k učení pro tuto síť lze založit na principu zpětného šíření chyby, jak uvádí [6], [7]. Tento způsob využívá modifikovaného algoritmu Back Propagation o němž jsme pojednávali v kap. 4.4.1.1. Zmíněný způsob učení se v praxi ovšem příliš nepoužívá, neboť je pro tuto síť neefektivní. Namísto něj se uplatňuje algoritmus, který kombinuje přístup učení s učitelem s technikou samoorganizace, jak uvádí [6]. O samoorganizaci jsme se dříve zmínili v kap. 4.3.1.2.

Celý proces učení se opět opírá o užití trénovací množiny, viz. předpis 4.18. V prvním kroku algoritmu probíhá nastavení pozic středů RBF neuronů pomocí váhových vektorů mezi vstupní a skrytou vrstvou sítě. V této fázi se uplatňuje zmiňovaná samoorganizace. Další krok modifikuje šířku oblasti kolem těchto středů. Posledním krokem je úprava váhových vektorů mezi skrytou

a výstupní vrstvou. Podrobný popis včetně matematických vyjádření lze v případě potřeby dohledat např. v [6], zde se jím dále zabývat nebudeme.

4.4.2.2 Využití neuronových sítí RBF

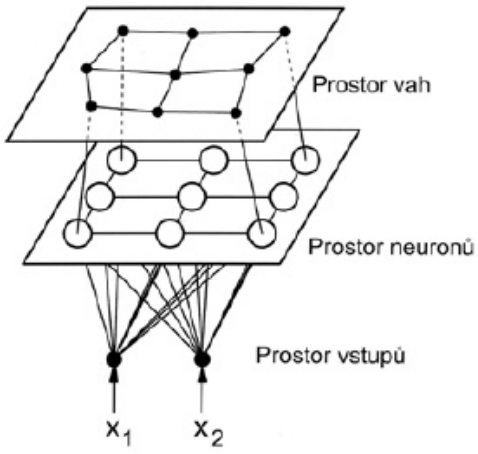
Uvedený typ sítě má uplatnění především pro problémy, kdy učení klasické vícevrstvé dopředné sítě trvá příliš dlouho. Uvedený algoritmus totiž může dosáhnout požadovaných výsledků o dva až tři řády epoch dříve, nežli klasická dopředná UNS [6]. Naproti tomu existují úlohy, které jsou pro tento typ sítě nevhodné, jak rovněž uvádí [6]. V oblasti rozpoznávání objektů v obraze lze tuto síť použít například pro úlohy OCR (Optical Character Recognition). Další uplatnění lze hledat například u problémů rozpoznávání tváří viz [15], apod.

4.4.3 Neuronové sítě LVQ a SOM

Neuronové sítě typu LVQ vycházejí z principů Kohonenových map, pro něž se modifikuje algoritmus učení na způsob učení s učitelem [6]. Základem je tedy Kohonenova neuronová síť, jinak také označovaná jako SOM (Self Organizing Map). Nejprve se proto zaměříme na samotnou Kohonenovu síť a popíšeme jejich strukturu a základní princip činnosti.

4.4.3.1 Kohonenovy mapy

Kohonenovy mapy se svou strukturou odlišují od klasické dopředné sítě. Stejně tak je odlišný i jejich princip činnosti. Tento typ sítě tvoří dvě vrstvy. Vstupní vrstva je tvořena uzly, které jsou propojeny s každým neuronem v následující, tzv. kompetiční vrstvě. Topologii ilustruje obrázek 4.17. Pojem kompetiční vrstva vyjadřuje soutěživý charakter učení, při kterém jsou modifikovány váhové vektory mezi vstupní vrstvou a vrstvou neuronů kompetiční vrstvy a současně také váhové vektory laterálně propojených neuronů v této kompetiční vrstvě. Po přiložení vstupního vektoru mezi sebou začnou neurony soutěžit dokud se vítězem nestane jediný z nich, který je tomuto vstupnímu vektoru nejblíže. Vítězný neuron se stává aktivním, čemuž odpovídá nastavení jeho výstupní hodnoty na 1. Ostatní neurony jsou neaktivní a mají výstupní hodnotu 0 [9]. Uvedený princip bývá označován jako „vítěz bere vše“, anglicky také „winner takes all“ [6]. V případě Kohonenových map se při učení neupravují pouze váhy jediného neuronu, ale váhové vektory v určitém okolí. Toto okolí je na počátku nastaveno na maximum a tvoří tak např. polovinu velikosti sítě [6]. V průběhu iterací se postupně dále zmenšuje a na konci procesu učení dosahuje velikosti jen vítězného neuronu. Fázi učení je mj. vhodné rozdělit na dvě části, na tzv. hrubé a jemné. V první fázi učení síti předkládáme vzory s největšími rozdíly, čímž docílíme oddělení požadovaných tříd, které v další fázi učení dolaďujeme [6].



Obrázek 4.17 – Kohonenova neuronová síť (převzato z [9]).

4.4.3.2 Neuronová síť LVQ

Tato síť je modifikací uvedené Kohonenovy neuronové sítě s rozdílem v přístupu k učení. V tomto případě jde o učení s učitelem, ale některé části původního přístupu učení bez učitele zůstávají zachovány [6]. Obecně lze činnost algoritmu popsat ve třech krocích, jak uvádí [6]. V prvním kroku se použije standardního algoritmu učení Kohonenovy sítě, pomocí něhož určíme rozmístění neuronů prostřednictvím váhových vektorů tak, že aproximují hustotu pravděpodobnosti jednotlivých vzorů. Další krok již využívá informace o požadovaném výstupu sítě a to na základě definice trénovací množiny 4.18. Postupným průchodem trénovací množiny zjišťujeme, který z neuronů nejlépe odpovídá na právě vybraný vzor. Přitom sestavujeme tabulku četností, kam ukládáme jednotlivé kategorie, které každý neuron reprezentuje. Kategorii, kterou neuron reprezentoval nejčastěji mu v závěru tohoto kroku přiřadíme, neboť každý neuron musí reprezentovat pouze jednu výstupní třídu. Výsledkem jsou skupiny neuronů, které přísluší jednotlivým třídám. Třetí krok obecného algoritmu realizuje doučení, které se provádí jedním z algoritmů LVQ1, LVQ2, nebo LVQ3 [6].

Vzhledem k rozsahu problematiky naznačíme pouze popis algoritmu LVQ1. Uvedený algoritmus realizuje posunutí váhových vektorů vítězného neuronu v případě, kdy tento neuron správně klasifikuje přiložený vstupní vektor. V opačném případě dojde k odsunutí vítězného neuronu od přiloženého vstupu. Podrobný princip činnosti a popis dalších dvou algoritmů lze v případě potřeby dohledat v [6].

4.4.4 Využití Kohonenových map a sítí LVQ

Obě uvedené UNS, lišící se v přístupu k učení, lze úspěšně aplikovat např. v úlohách rozpoznávání vzorů [9], [16]. Jsou využívány např. v úlohách OCR (Object Character Recognition), apod. V případech použití sítí LVQ je dosahováno lepších výsledků, nežli u SOM. Neuronová síť LVQ bývá také demonstrativně srovnávána s dopřednou vícevrstvou UNS. V takových experimentech dosahuje poměrně slušných výsledků při menším počtu epoch učení na rozdíl od zmíněné dopředné sítě, jak uvádí [16]. Vliv na úspěšnost má i volba použitého algoritmu LVQ, jak také uvádí [16].

5 Návrh řešení

Jak je z obsahu předchozí kapitoly patrné, problematika umělých neuronových sítí je velmi rozsáhlá. Není proto možné, vzhledem k požadovanému rozsahu této práce a obsáhlosti diskutované oblasti umělé inteligence, podrobně prozkoumat všechny vlastnosti a uvedené typy UNS. Rozhodl jsem se proto práci zaměřit na nejpoužívanější typ UNS a tomuto záměru přizpůsobit implementaci pomocného nástroje a obsah experimentů, jak bude uvedeno dále. V dalším textu se podrobně zaměříme na dopřednou vícevrstvou UNS, kterou budeme trénovat za pomocí algoritmu Back Propagation.

Abychom byly schopni realizovat některé experimenty s touto sítí, neobejdeme se bez vhodného nástroje, který k tomuto účelu použijeme. Proto jsem se rozhodl před započetím implementace navrhnout množinu požadavků na vlastnosti, které by výsledný nástroj měl splňovat. S jeho pomocí bychom měli být schopni pracovat s dopřednou vícevrstvou UNS a podrobně definovat vlastnosti, jaké by konkrétní neuronová síť měla mít. Dále by mělo být možné provádět učení a vhodným způsobem i testování vlastností neuronových sítí a to především v roli klasifikátoru pro vstupní obrazová data.

Od aplikace budeme požadovat tyto konkrétní vlastnosti:

- možnost vytvořit dopřednou UNS s 1, nebo 2 skrytými vrstvami podle potřeby.
- možnost definovat počet uzlů/neuronů v každé vrstvě sítě podle potřeby
- možnost provádět počáteční inicializaci váhových vektorů celé sítě
- možnost vytvářet a spravovat trénovací, případně i testovací množinu.
- možnost ukládat vytvořenou obecnou (trénovací, nebo testovací) množinu do souboru a zpětně ji načíst
- možnost zadávat vstupní a očekávané výstupní vektory sítě ručně, nebo na základě obrazových dat při vytváření prvků obecné množiny
- možnost provádět učení sítě pomocí trénovací množiny za pomocí stanovených parametrů, ovlivňujících průběh učení
- možnost sledovat průběh chyby trénovacích, příp. i testovacích dat
- schopnost ukládat UNS do souboru a zpětně načíst
- možnost načítat obrazová data v podobě rastrového souboru, jako samostatný vzorek, pro účely testování schopností sítě

Pro implementaci bude použit objektový jazyk Java verze 1.6.0.17. Vývoj bude probíhat pod operačním systémem Linux openSuse 11.2. Zdrojem vstupních obrazových dat pro neuronové sítě bude webkamera spolu s vhodným externím softwarový nástrojem pro snímání. Pořízená obrazová data budou sítí zpracovávána jako vektor hodnot pixelů. Nebude použito žádné ze segmentačních technik a budeme se především snažit ověřit obecné chování UNS na variantnost snímaných předmětů a změny podmínek v obraze (osvětlení, šum, atd.) u konkrétních typů úloh. Do celého procesu bude také zahrnuto vhodné předzpracování vstupních snímků, nejlépe takové, které bude mít pozitivní přínos na výsledky a proveditelnost co největšího množství různých experimentů. Při vývoji této aplikace se budu také snažit, aby výsledný kód byl přenositelný, což by se vzhledem k vybranému jazyku mělo podařit. Nepředpokládám v tomto případě použití žádných externích knihoven a balíků. Posloužit by měly základní prostředky jazyka Java. Pro pořizování snímků, které budeme UNS předkládat použijeme následující hardware:

Hardware	Max. rozlišení
Webkamera Creative webcam 5	0.3 Mpx (640x480)

Tabulka 5.1 – Použitý hardware

6 Implementace

V této kapitole se budeme zabývat popisem aplikace, jenž je součástí řešení této bakalářské práce. Při jejím vývoji byl kladen důraz na uživatelsky pohodlné a intuitivní ovládání, které by umožňovalo měnit veškeré teoreticky významné proměnné, které ovlivňují výsledné vlastnosti UNS, nebo celkový průběh učení, apod.

Aplikace se opírá o možnosti objektově orientovaného jazyka Java, což s sebou přináší některé výhody, ale i nevýhody. Základní schéma aplikace vystihuje diagram tříd, který je přílohou k této práci. Ten charakterizuje nejvýznamnější části aplikace, které mají souvislost s řešenou problematikou. Zbylé části aplikace, které nezahrnuje, tvoří především komponenty grafického uživatelského rozhraní. Podobu vzhledu aplikace ilustruje další příloha k této práci.

Nyní se zaměříme na popis jednotlivých částí našeho nástroje. Nebudeme ovšem ve všech případech pronikat do implementačních detailů, ale omezíme se na rovinu vhodnou pro popis problémů a jejich řešení, které jsou s danou částí aplikace spojeny. Zdrojové kódy výsledné aplikace je v případě potřeby možné nalézt na přiloženém DVD, stejně jako spustitelný jar archiv.

6.1 Výpočty s jednoduchou přesností

Z hlediska výpočetních nároků byl pro uložení všech proměnných atributů UNS zvolen datový typ float, neboli uložení s jednoduchou přesností (32 bit). I přes to, že prvotní verze aplikace byla realizována s použitím datového typu double (64 bit), mělo jeho použití poměrně zásadní dopad na výpočty UNS z hlediska času i množství potřebné paměti. Některé úlohy tak byly neřešitelné, nebo za stávajících podmínek jen obtížně řešitelné. Z těchto důvodů se upustilo od původně použitého datového typu double, jenž byl nahrazen za vhodnější datový typ float. Tento způsob vnitřní interpretace veškerých proměnných UNS je pro účely klasifikace obrazových dat a správnou činnost aplikace zcela dostačující. Možnosti tohoto datového typu v jazyce Java popisuje tabulka 6.1.

minimální hodnota	maximální hodnota	nejmenší absolutní hodnota (> 0)
-3.40282e+38	+3.40282e+38	1.40239846e-45

Tabulka 6.1 – Datový typ float v jazyce Java [13]

6.2 Vytváření a editace obecné množiny

Nyní se budeme zabývat popisem přístupu k problematice vytváření a editace obecné množiny. Pojmem obecná množina se v tomto kontextu snažíme nerozlišovat mezi trénovací a testovací množinou, neboť jednotlivé specifikující názvy plynou až ze způsobu použití instance této obecné množiny. Řešená problematika se opírá o matematické vyjádření trénovací, příp. testovací množiny jako množiny dvojic viz 4.18. Zadání každého prvku dvojice, tj. vektoru vstupní hodnoty sítě a vektoru výstupní hodnoty sítě, lze provádět manuálně, zadáním obou vektorů za dodržení syntaxe, nebo definováním vstupního vektoru na základě obrazových dat a manuálním zadáním výstupního vektoru, stejně jako v prvním případě dodržením předepsané syntaxe. Tyto způsoby definice obecné množiny více ozřejmí příloha se snímkem, viz obrázek 2, která je součástí této práce. Zmiňovaná syntaxe definuje způsob zápisu vektoru, jehož jednotlivé číselné složky odděluje pomocí čárky „ , “. Pro odlišení desetinných míst slouží znak tečky „ . “. Závorky, ani jiné další znaky nejsou přípustné.

Vytváření množiny o velkém počtu obrazových dat, jejichž vstupní vektory, tj. jednotlivé obrázky jsou předem roztříděny v adresářích souborového systému podle jejich příslušnosti k odpovídající třídě či skupině tříd, lze usnadnit a urychlit. Tuto možnost lze uskutečnit zápisem výstupního vektoru do příslušné grafické komponenty a načtením obsahu adresáře, jehož všechny soubory s obrazovým obsahem a stejným rozlišením budou s tímto výstupním vektorem spojeny. Poněkud podrobněji lze definovat pro každý vstupní vektor odpovídající výstupní vektor zvlášť.

Každý vzorek, takto načtený ze souboru, nejprve prochází předzpracováním a následně je normalizován, aby jej bylo možné korektně zpracovat pomocí UNS.

6.2.1 Předzpracování obrazových dat

Vhodné předzpracování nabízí mimo jiné možnost, jak odstranit některé přebytečné informace a přitom zachovat významnost původní informace. O dalších možnostech a některých technikách se zmiňujeme v kap. 3.2. Hlavní snahou bylo zmenšení počtu složek vstupního vektoru, který získáváme z načítaných obrazových dat, jenž by urychlilo celkový proces učení a zpracování pomocí UNS. Řešením tohoto problému bylo převedení barevného obrázku do odstínů šedi za použití prezentovaného vzorce 2.1. Jiné techniky předzpracování nebyly implementovány. Ovšem není pravdou, že pro vytváření této obecné množiny vždy používáme rastrová data o původním rozlišení. Změna rozlišení vstupních snímků je proveditelná pomocí samostatného Bash scriptu resize.sh za využití nástroje convert. Tento způsob změny rozlišení umožňuje pohodlné využití i více jader CPU, přičemž veškerá implementace scriptu je na pouhých pět řádků kódu. Script resize.sh lze nalézt na přiloženém DVD. Jiný způsob předzpracování do výsledné implementace zahrnut nebyl. Podobně nebyla zavedena žádná segmentační technika. UNS tak zpracovává vždy celé snímky.

6.2.2 Transformace a normalizace obrazových dat

Současně s předzpracováním provádíme i transformaci z maticové reprezentace rastrového obrazu do podoby vektoru, který později přikládáme na vstup neuronové sítě. Tento proces je triviální. Podstatné je zmínit, že po transformaci prvního vzorku vytvářené obecné množiny, je poznamenán počet složek vytvořeného vektoru. Touto informací kontrolujeme požadované konstantní rozlišení pro každý další vzorek a v případě zjištění změny je proces vytváření obecné množiny ukončen.

Významnou roli hraje proces normalizace hodnot složek vektorů. Aplikací normalizace je zajištěna korektnost výpočtu pomocí UNS. Potřeba normalizovat jednotlivé složky vektorů plyne z vlastností aktivační funkce neuronů, tedy z jejího oboru hodnot H(f). Tomuto rozsahu je třeba přizpůsobit každou položku vektoru. Současně je třeba zajistit rovnoměrné rozložení ve výsledném intervalu. Jak bude ještě zmíněno v dalším textu, vytvořená výsledná aplikace, modelující strukturu UNS, používá umělé neurony se sigmoidální aktivační funkcí viz vztah 4.7. Tomu odpovídá binární rozsah hodnot <0;1>. Současně také známe nejmenší a největší možnou hodnotu pixelu, tedy hodnotu složky vektoru obecně, tj. <0;255>. Na základě těchto znalostí byl pro normalizaci odvozen jednoduchý vztah (6.1), transformující hodnotu pixelu do intervalu daného oborem hodnot sigmoidální funkce s rovnoměrným rozložením.

$$normFloat = \frac{1}{255} \cdot pixelValue \tag{6.1}$$

Odtud také plyne skutečnost, že interval manuálně zadávaných složek vektoru musí splňovat rozsah <0;1>. V souvislosti s procesem normalizace je třeba také zpětně poukázat na vlastnosti datového typu float. Především na údaj nejmenší absolutní hodnoty větší, než 0 viz tabulka 6.1. Tento údaj je vhodné dát do srovnání s nejmenší normovanou hodnotou pixelu, taktéž větší, než 0. Její hodnotu získáme dosazením pixelu o hodnotě 1 do vztahu 6.1, jak ilustruje následující postup.

$$normFloat = \frac{1}{255} \cdot 1 = 3.92157 \cdot 10^{-3}$$

Z porovnání obou těchto údajů plyne, že datový typ float poskytuje dostatečné rozlišení pro uložení celkem 256 hodnot do zmiňovaného intervalu <0;1>.

6.2.3 Uspořádání prvků ve vytvářené množině

Jak jsme se zmínili v kap. 4.4.1.2, může se jevit jako vhodné při učení UNS přistupovat k prvkům trénovací množiny náhodně, nikoliv sekvenčně. V tomto případě jsem zvolil odlišnou cestu, mající ovšem podobný podtext. Při vkládání každého dalšího prvku, tj. dvojice vektorů, do vytvářené obecné množiny není tento prvek vkládán na konec množiny sekvenčním způsobem, ale na náhodně vygenerovanou pozici mezi některé dva již existující prvky v této obecné množině. Tímto způsobem se podařilo zlepšit průběh učení a zmenšit počet epoch učení algoritmu Back Propagation. Pro určení pozice prvku byl použit generátor pseudonáhodných čísel jazyka Java (třída java.util.Random) s rovnoměrným rozložením v intervalu <0; 1>.

6.3 Vytváření dopředné vícevrstvé UNS

Při vytváření struktury dopředné vícevrstvé UNS nejsou kladena žádná omezení. Skutečná omezení vyplývají pouze z množství dostupné operační paměti. Je umožněno definovat přesnou podobu sítě stanovením počtu neuronů v každé vrstvě. Bavíme-li se o vrstvě vstupní, pak lze definovat i počet uzlů v této vrstvě. Podobně lze definovat pouze jednu skrytou vrstvu, nebo je-li požadováno, případně i druhou. Větší počet vrstev není povoleno vytvářet, neboť jak bylo uvedeno v kap. 4.4.1.2, větší počet vrstev než dvě se nijak neprojevuje na zlepšení schopností UNS.

Další parametr, který lze specifikovat před inicializací UNS, je λ, neboli parametr strmosti aktivační funkce, viz vzorce 4.7, 4.8, 4.9, 4.10. V implementaci jsem se omezil pouze na jeden typ neuronu. Je použito neuronu s lineární bázovou funkcí a sigmoidální aktivační funkcí, která definuje binární výstup. Pojem vstupní uzel je pak chápán jako samostatná kategorie prvku s jednotkovým přenosem a lze jej zahrnout pouze do vstupní vrstvy sítě.

Inicializace váhových vektorů sítě je neměnná a probíhá na základě generátoru pseudonáhodných čísel jazyka Java s rovnoměrným rozložením v intervalu <-0,5; +0,5>. Experimentálně bylo ověřeno, že volba ve větším intervalu spíše zvětšuje počet epoch nutných k naučení sítě. Také bylo experimentálně vyzkoušeno změnit rozsah pouze do oblasti kladných nebo záporných čísel, např. <-1,0; 0> apod. Tento způsob inicializace se jevil jako nevhodný pro některé typy úloh, naopak řešení problému XOR, nebo rozpoznávat číslice v obraze malého rastru nebylo problémem pro UNS se naučit. Často stačila pouze větší velikost rastru a problém byl neřešitelný, i když se jednalo o stejnou úlohu.

6.4 Přístup k učení a testování UNS

Jako algoritmus pro učení byl použit Back Propagation algorithm v základní podobě s četností uprav váhových vektorů po každém vzorku z TRM. Hlavním zdrojem informací při implementaci byl [7]. Pro výpočet změny váhového vektoru bylo použito vztahu 4.29 na místo základního 4.22. Tato skutečnost se jevila jako přínosná, neboť vhodná kombinace parametrů μ a α dokázala značně urychlit proces učení. Ovšem otázka volby vhodnosti volby obou parametrů je zřejmě individuální ke každému řešenému problému. Z pohledu výsledné aplikace je tedy umožněno definovat oba tyto parametry, stejně jako maximální počet epoch nebo požadovanou chybu pro TRM za celou epochu, jenž jsou podmínkou pro ukončení učení.

K učení lze přistupovat také dalším způsobem, jenž ověřuje schopnost sítě zobecňovat na TSM. V takovém případě se podmínka pro zastavení učení rozšiřuje o požadovanou chybu testovací množiny. Způsob výpočtu chyby TSM je stejný jako při učení v algoritmu Back Propagation pro TRM. O přístupech k učení pomocí Back Propagation algoritmu jsme se zmínili v kap. 4.4.1.1.

V obou případech dovoluje výsledná implementace sledovat průběh chyby, jak pro TRM, tak i pro TSM z výpisů, které generuje na standardní výstup. Grafické zpracování těchto hodnot bude součástí experimentů v následující kapitole.

6.4.1 Další možnosti testování výsledné UNS

Výsledná implementace nástroje, který byl vytvořen jako součást této bakalářské práce, nabízí také několik možností, jak pohodlným způsobem otestovat vlastnosti již naučené nebo případně i nenaučené UNS.

Nejprve zmiňme test aproximačních schopností sítě. S jeho pomocí lze ověřit chování sítě pro různé hodnoty položek vstupního vektoru, jenž zadáváme. Je tak možné například ověřovat schopnost sítě aproximovat goniometrické funkce. Tato funkcionalita ovšem pro problematiku klasifikace obrazu nemá užitek a je pouze navíc pro případné jiné typy úloh.

Funkcí, která se pro zpřehlednění výsledků klasifikace může hodit je pojmenování výstupů sítě. Jde v podstatě o pojmenování tříd, do kterých bude vstupní vektor klasifikován. Tato funkce je spjata s funkcionalitou tlačítka „Otevřít obrázek“ a také s tzv „On-line testováním“. První zmiňovaná funkce umožňuje načíst jeden obrázek ze souborového systému, jehož rozlišení po transformaci matice na vektor se musí shodovat s počtem vstupních uzlů sítě. Druhá zmiňovaná funkce je poněkud složitější a s její pomocí je možné provádět klasifikaci nad obrazovými daty, které právě snímá připojená kamera. Problém s dodržením rozlišení byl v tomto případě vyřešen zmenšením načítaného vstupního obrazu na uživatelem zadané rozměry rastru. Toto zmenšování nemodifikuje vstupní data a probíhá pouze uvnitř běžící aplikace. Funkce „on-line testování“ se ovšem neobejde bez spolupráce externího softwaru, který je schopen snímat a ukládat fotografie z připojené kamery. Pro tento účel posloužil program motion, který je schopen v pravidelných intervalech snímat obrazová data a každý poslední snímek zpřístupnit pomocí symbolického linku. Tímto způsobem bylo testováno například rozpoznávání obličejů, viz 7.2.

6.5 Stručně o použitých SW nástrojích

Mezi nástroj, který dosud nebyl zmíněn patří program GIMP s jehož pomocí byly připraveny některé vzorky pro vytvoření TRM, případně TSM.

Program motion, který je zmiňovaný v kap. 6.4.1 je určen pro snímání obrazu při detekci pohybu. Jeho vhodnou úpravou v konfiguračním souboru lze snížit práh, definován jako množství změněných pixelů, na minimum a použít jej jako nástroj pro získávání snímků i jinak neměnných. Jeho použití je vázáno na prostředí OS Linux. S připojeným hardware pracuje pomocí rozhraní video4linux. Jeho výhoda spočívá ve zpřístupnění posledního pořízeného snímku přes symbolický link. Tento software byl rovněž použit pro získávání snímků při sestavování TRM a TSM.

Pro zmenšování získaných snímků byl použit vlastní Bash script, který využívá nástroje convert, jenž je opět přístupný v prostředí OS Linux. Tento script dokáže efektivně využít i více jader CPU, případně více CPU, pokud jsou k dispozici. Jeho použití nabízí efektivní způsob, jak mj. zmenšit vstupní obraz na požadované rozlišení. Nástroj convert ovšem nabízí i další možnosti.

Grafické zpracování experimentálních číselných výsledků bylo realizováno pomocí nástroje gnuplot verze 4.2.5, který nabízí bohaté možnosti pro vytváření grafů. Jeho velkou předností je schopnost načítat číselné záznamy ze souboru a jednotlivým sloupcům, které jsou odděleny bílým znakem, přiřadit příslušnost k osám x, y, příp. z.

7 Experimenty

V této kapitole se budeme věnovat několika experimentům, pomocí kterých ověříme klasifikační schopnosti UNS. Nebude-li uvedeno jinak, bude pro výpočet množství neuronů ve skrytých vrstvách použito vztahů 4.27 a 4.28. Získané výsledky budou průběžně zhodnocovány a prezentovány pomocí grafů, nebo jiným vhodným způsobem. Pro jistotu ještě připomeňme, že veškerá obrazová data jsou při zpracování vytvořenou aplikací převáděna do odstínů šedi a nejsou zpracovávána jako barevná.

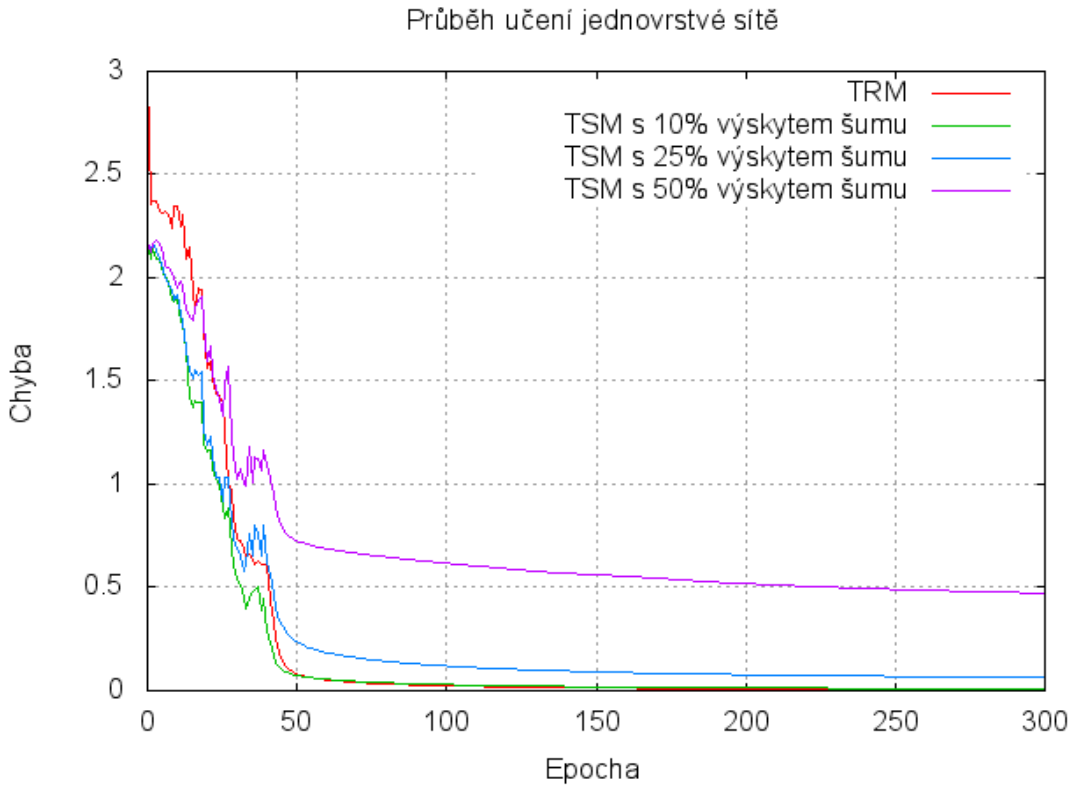
7.1 Rozpoznávání znaků

V tomto experimentu se budeme zabývat rozpoznáváním znaků, resp. pěti různých číslic, která jsou obsažena v rastru 25x42. Klasifikovat tedy budeme do pěti tříd k čemuž bude použito nejprve jednovrstvé a následně i dvouvrstvé sítě. Trénovací množina je v tomto experimentu sestavena z pěti prvků, které představují rastrové matice transformované na vektory, které budou předkládány UNS. Jejich obsahem jsou číslice viz obrázek 7.1. Testovacích množin je poněkud více, celkem tři. Každá testovací množina obsahuje pět prvků, stejně jako TRM. Rozdíl mezi nimi spočívá v množství přítomného RGB šumu, který byl do původního obrazu náhodně vygenerován. Jednotlivé množiny jsou tedy zasaženy 10%, 25% a 50% šumu, viz obrázek 7.2.

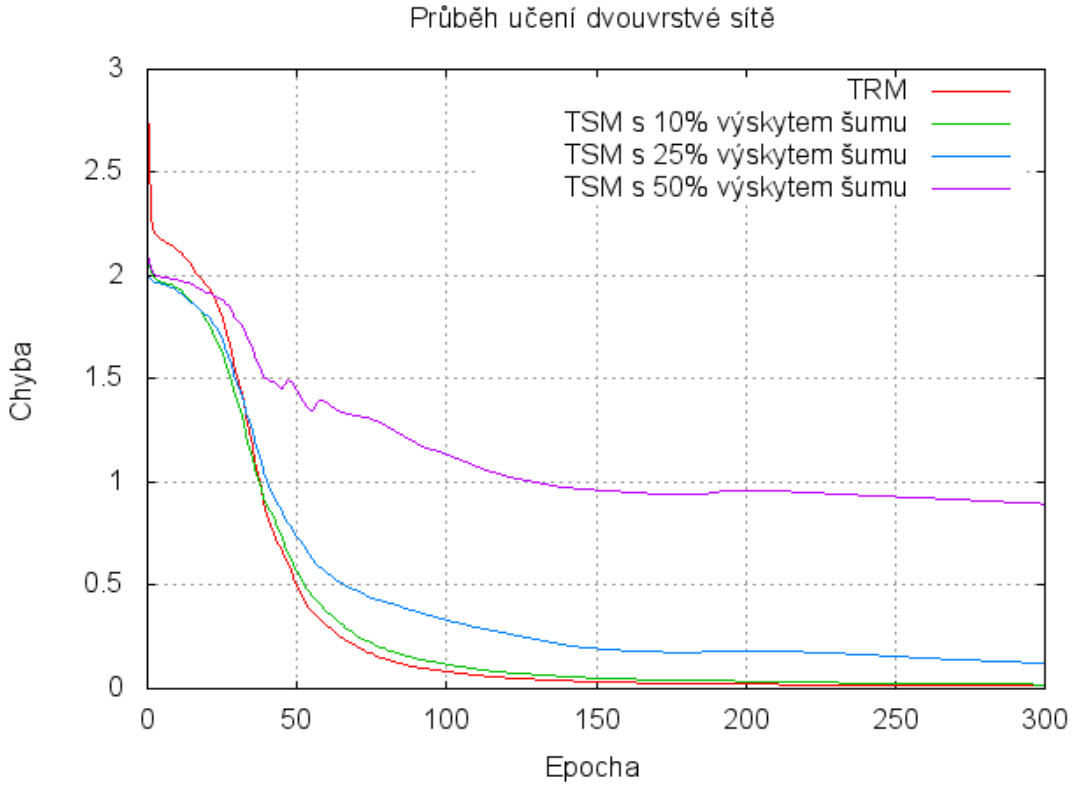
Obrázek 7.1 – Prvky trénovací množiny

Obrázek 7.2 – Ukázka výskytu šumu v rastru s číslem 3 v testovacích množinách

Tento experiment je zaměřen na analýzu průběhu chyby trénovacích a testovacích dat při učení UNS s omezeným počtem epoch na 2000. Učení provedeme pro každou testovací množinu zvlášť a současně budeme vždy vycházet z identicky inicializované UNS, čehož bylo dosaženo jejím uložením do souboru před začátkem prvního učení. Parametry ovlivňující učení budeme volit ve všech případech stejné, tj. $\lambda = 1$, $\mu = 0.5$, $\alpha = 0.1$. Experiment je realizován za pomocí jednovrstvé sítě s architekturou 1050-73-5 a následně pro porovnání také pomocí dvouvrstvé sítě se strukturou 1050-117-30-5. Průběhy chyb pro zmiňované 3 množiny a jednotlivé oba typy UNS znázorňují následující grafy 7.1, 7.2. Pro názornost bylo do grafického zpracování zahrnuto pouze 300 epoch. Další průběh měl sestupný exponenciální charakter. Numerické výsledky tohoto experimentu lze nalézt na přiloženém DVD, stejně tak i výsledky dalších experimentů.



Graf 7.1 – Průběh chybových funkcí



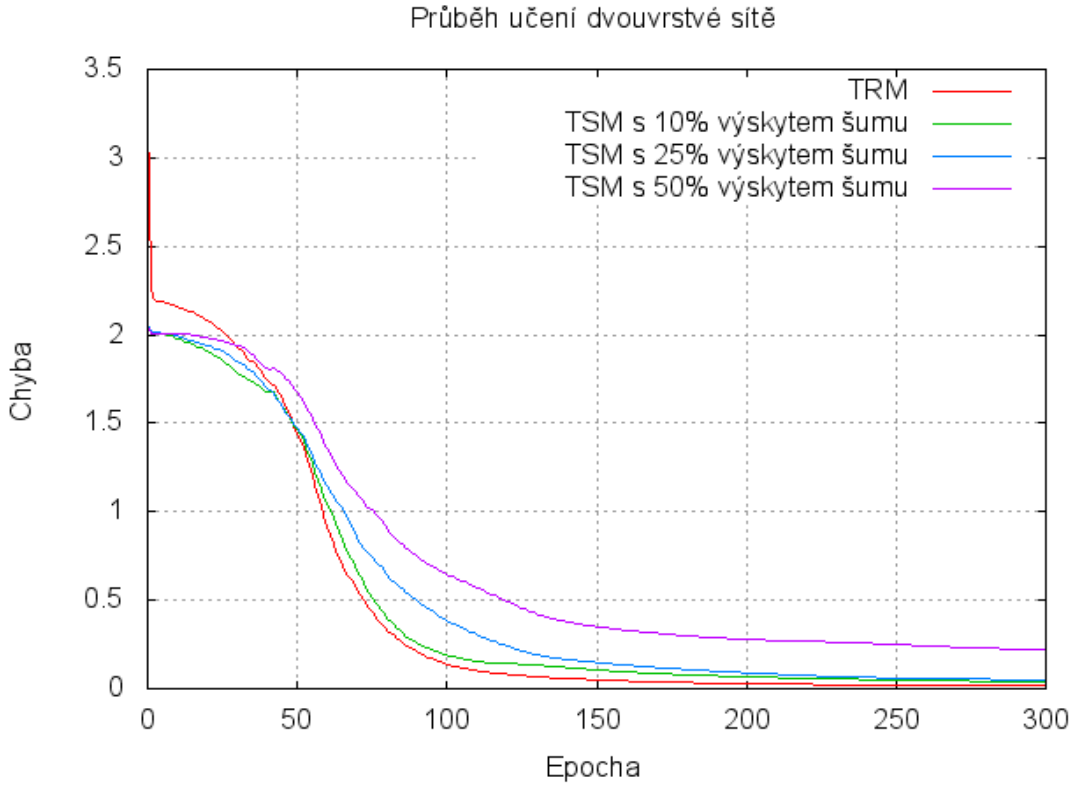
Graf 7.2 – Průběh chybových funkcí

Výsledky ukazují, že nejmenších chybových hodnot pro testovací množiny dosáhla jednovrstvá UNS. Ve všech případech bylo naměřené minimum pro chybu TSM vypočteno pro poslední 2000. epochu. Rozdíly mezi jednotlivými sítěmi jsou patrné především pro množinu s 50% výskytem šumu. Souhrn výsledků minimální dosažené hodnoty chyb pro jednotlivé UNS a testovací množiny uvádí následující tabulka 7.1.

TSM	UNS 1050-73-5	UNS 1050-117-30-5
10% šumu	0.0015105592	0.0027303994
25% šumu	0.02018722	0.021320023
50% šumu	0.3947641	0.7484355

Tabulka 7.1 – Nejmenší hodnoty chyb pro TSM

Protože příliš vysoká chyba pro testovací množinu u dvouvrstvé sítě vypovídá o špatné schopnosti sítě zobecňovat, pokusil jsem se architekturu dvouvrstvé sítě upravit zmenšením počtu neuronů ve skrytých vrstvách. Náhodně tedy byla stanovena nová podoba dvouvrstvé sítě s architekturou 1050-80-25-5. Charakter průběhu chybových funkcí pro prvních 300 epoch ilustruje následující graf 7.3.



Graf 7.3 – Průběh chybových funkcí

Z výsledků prezentovaných tímto grafem a tabulkou 7.2, jenž uvádí dosažená minima chybových funkcí jednotlivých TSM je patrné výrazné zlepšení. Podstatně lepší výsledky byl dosaženy pro TSM s 25% a 50% šumu v porovnání s oběma původně použitými UNS. Nicméně pro testovací množinu s 10% výskytem šumu jsme v tomto případě dosáhli nejhorších výsledků. Uváděná minima chybových funkcí jsou opět příslušná poslední epoše učení pomocí algoritmu Back Propagation.

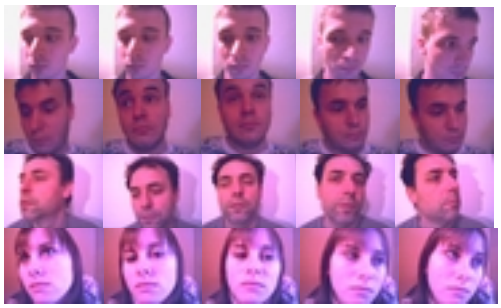
TSM	UNS 1050-80-25-5
10% šumu	0.006923813
25% šumu	0.0060073114
50% šumu	0.10129939

Tabulka 7.2 – Nejmenší hodnoty chyb pro TSM

Pokud bychom v učení jednotlivých sítí pokračovali dále, pak bychom patrně dosáhli ještě menších hodnot chybových funkcí. Případné další experimentální změny v množství neuronů ve skrytých vrstvách by rovněž mohly být cestou k dosažení lepších výsledků. Cílem tohoto experimentu nebylo nalezení minima chybové funkce, nýbrž nastínit a prezentovat průběhy chybových funkcí a vliv změny topologie na schopnost UNS zobecňovat. Dosažené výsledky potvrzují teoretické předpoklady vztahující se na strukturu UNS, viz kap. 4.4.1.2.

7.2 Rozpoznávání tváří

V tomto experimentu bylo použito UNS jako klasifikátoru pro rozpoznávání tváří. Experimentu se zúčastnili čtyři dobrovolníci, jejichž tváře byly snímány za pomocí webkamery a aplikace motion, viz kap. 6.5. Získaná obrazová data byla podrobena manuální kontrole a byly odstraněny snímky, které byly příliš rozmazané vlivem pohybu jedince, nebo takové, kde tvář jedince byla mimo zorné pole webkamery. Snímaní dobrovolníci měli povoleno pohybovat hlavou do různých stran, mrkat a případně i mluvit. Do snímání také občas vstupovaly změny osvětlení, které byly vyvolány záměrně, např. změnou místa fotografování dobrovolníka, nebo použitím světelného zdroje. Celkem bylo tedy použito 5002 snímků v rozlišení 320x240 px, které byly scriptem resize.sh zmenšeny na velikost 37x28 px za zachování poměru stran. Dále nebyly aplikovány žádné techniky předzpracování, abychom si lépe udělali představu o schopnostech UNS.



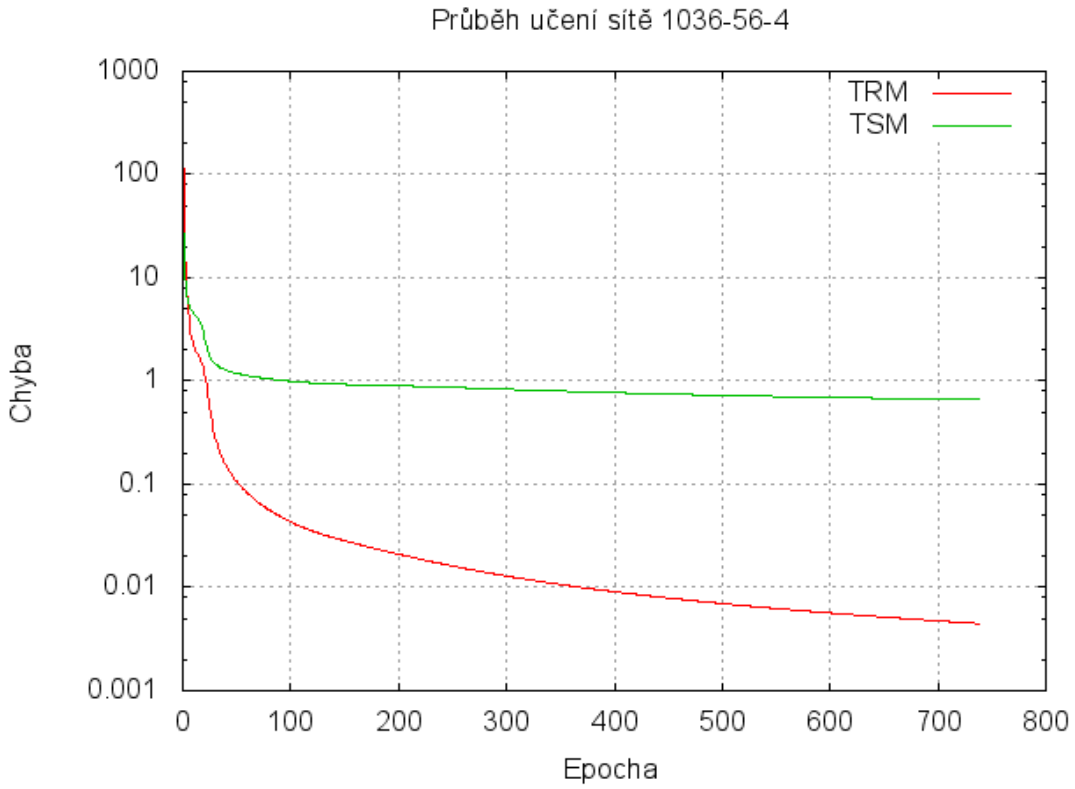
Obrázek 7.3 – Ukázka zmenšenin dobrovolníků

Celkový počet snímků každého dobrovolníka nebyl stejný, ale různý, jak ilustruje tabulka 7.3. Sestavení trénovací a testovací množiny bylo realizováno rozdělením celkového počtu vzorků každého jedince na dvě poloviny. Toto rozdělení bylo realizováno náhodným výběrem. Výsledkem tedy byla TRM s počtem 2501 a TSM s počtem 2501 vzorků.

dobrovolník	celkový počet pořízených snímků
1	2406
2	866
3	644
4	1078

Tabulka 7.3 – Množství pořízených snímků

Pro první řešení tohoto experimentu byla použita UNS s architekturou 1036-56-4, jenž byla sestavena za pomocí doporučovaného vztahu 4.27. Učení této sítě probíhalo s parametry $\lambda = 1$, $\mu = 0.5$, $\alpha = 0.2$ a trvalo přibližně 20 hodin, po této době jsem učení zastavil. Celkem tak proběhlo 738 epoch učení pomocí algoritmu Back Propagation s ověřováním vlastností sítě pomocí testovací množiny. V tomto případě se podařilo dosáhnout minimální chyby trénovacích dat 0.0044749. Nejmenší dosažená hodnota chyby pro testovací data byla 0.6615212. Obě hodnoty odpovídají poslední epoše učení. Z charakteru průběhu chyb je zřejmé, že pokud bychom v učení pokračovali, hodnota by patrně dále ještě nějakou dobu klesala. Průběh chybových funkcí získaných z procesu učení ukazuje následující graf 7.4.



Graf 7.4 – Průběh chybových funkcí

Protože se dosažená hodnota minima chyby pro testovací data zdála být příliš vysoká, rozhodl jsem se celý proces zopakovat za pomocí sítě s architekturou 1036-72-4. Množství neuronů ve skryté vrstvě jsem tentokrát zvolil náhodně. Pro učení byly použity stejné parametry, jako v případě předchozí sítě. Proces učení byl ukončen po 585. epoše, tj. opět přibližně po 20 hodinách výpočtu. I přes menší počet epoch, než v případě první sítě, lze mluvit o úspěchu. Hodnotu chyby pro testovací množinu se podařilo podstatně snížit (téměř 15x) a to na 0.044723906. V případě chyby pro trénovací množinu bylo ale dosaženo o něco horších výsledků, tedy chyby 0.007263683. Ovšem zlepšení vlastností při klasifikaci vzorů v TSM jasně vypovídá o tom, že tato UNS je pro další využití nejlepším kandidátem.

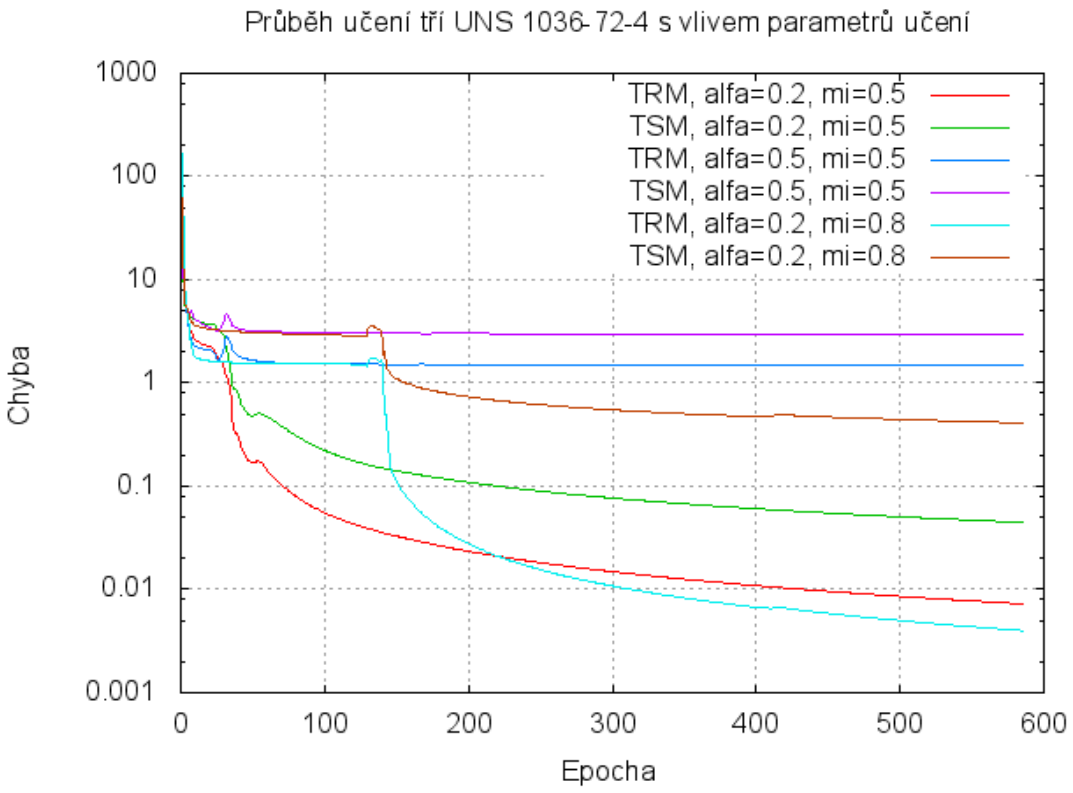
Abychom si udělali představu o tom, jaký podíl na chybě testovacích dat mají vzorky patřící jednotlivým dobrovolníkům, vytvořil jsem zvlášť čtyři testovací množiny. Každá z množin obsahuje vzorky které se nacházely v původní testovací množině a současně přísluší jedinému z dobrovolníků. Podotkněme, že množství vzorků, které vznikly snímáním každého dobrovolníka je různé, viz Tabulka 7.3: Proto se také vytvořené pomocné testovací množiny liší svou velikostí. Možnost vypočíst chybu testovací množiny, aniž bychom realizovali učení, je jednou z vlastností nástroje, který byl jako součást díla této bakalářské práce vytvořen. Způsob výpočtu je dán vztahem 4.20, který se používá pro výpočet chyby v algoritmu Back Propagation.

dobrovolník	chyba
1	0.0031539004
2	0.016430536
3	0.0074633206
4	0.017676173

Tabulka 7.4 – chyba příslušející jednotlivým dobrovolníkům TSM

Dáme-li si získané chyby testovacích množin příslušející jednotlivým dobrovolníkům (viz tabulka 7.4) do souvislosti s množstvím snímků, které byly u každého z nich pořízeny a z nichž byla sestavena také původní testovací množina (viz tabulka 7.3), pak vidíme, že zde nejspíš neexistuje významná závislost. Míra chyby je tak spíše ovlivněna jednotlivými vzorky, které se v testovací množině nachází. Může být ovlivněna například množstvím snímků s jiným osvětlením scény, nebo přítomností rozmazaných snímků při pohybu snímaného objektu nebo kamery apod. Většinu z těchto nedostatků je možné odstranit vhodným předzpracováním, které by v konečném důsledku mohlo pomoci ještě snížit hodnotu chybové funkce TSM.

V souvislosti s tímto experimentem jsem se dále také zabýval vlivem parametrů koeficientu učení μ a momentu α, které participují ve vztahu 4.29. K tomuto experimentu jsem využil druhou z uváděných umělých neuronových sítí, jenž vykazovala lepší vlastnosti v dosažené hodnotě chyby TSM. Výchozím bodem bylo stejné počáteční nastavení váhových vektorů, čehož bylo opět docíleno uložením instance sítě do souboru před zahájením učení, stejně jako v experimentu 7.1. Vytvořil jsem tedy další dvě instance sítě 1036-72-4, které jsem nechal trénovat pomocí algoritmu Back Propagation s využitím stejné TRM i TSM. Rozdílné byly pouze parametry, které ovlivňují učení, s vyjímkou parametru strmosti aktivační funkce λ, jenž byl nastaven na hodnotu 1. V prvním případě bylo učení realizováno s parametry $\mu = 0.8$, $\alpha = 0.2$. K učení druhé sítě bylo použito hodnot $\mu = 0.5$, $\alpha = 0.5$. Porovnání chybových funkcí TRM a TSM uvedených tří sítí 1036-72-4 ukazuje následující graf 7.5.



Graf 7.5 – Průběh chybových funkcí

Ze získaných výsledků jasně plyne význam otázky vhodné volby parametrů μ a α, viz kap. 4.4.1.2. Graf ukazuje, jak rozdílných průběhů chybových funkcí lze dosáhnout pro TRM a TSM v závislosti na jejich volbě. Také je patrné, že žádná z dalších dvou instancí sítě neposkytuje lepší výsledky pro testovací množinu, než-li původní síť, trénovaná s parametry μ = 0.5, α = 0.2. V případě trénovací množiny bylo dosaženo lepších výsledků zvýšením hodnoty μ na 0.8. Nejhorší výsledky pro TRM i TSM vykazovala instance UNS, trénovaná s parametry μ = 0.5, α = 0.5.

7.3 Rozpoznávání objektů a polohy v obraze

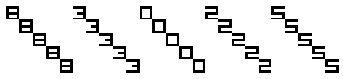
Tento experiment má ukázat vlastnosti dopředné vícevrstvé UNS rozpoznávat identitu a polohu objektu v obraze. Inspirací pro tento experiment je sborník [14], kde autoři analyzují vlastnosti modulární a nemodulární dopředné vícevrstvé UNS a dosažené výsledky u jednotlivých sítí prezentují.

Nejprve ozřejmíme pojem modulární neuronová síť. Jde o UNS s dopřednou vícevrstvou architekturou, která realizuje jen některá spojení mezi poslední skrytou vrstvou a výstupní vrstvou [14]. Naproti tomu UNS s nemodulární architekturou realizují spojení mezi všemi neurony skryté vrstvy s každým neuronem výstupní vrstvy.

V obou případech sítí jsou výstupní neurony rozděleny na dvě skupiny. První skupina slouží ke klasifikaci identity a druhá skupina klasifikuje polohu snímaného objektu. Uváděné dva typy

dopředných sítí v tomto směru vykazují různé vlastnosti, jak uvádí [14]. Síť s nemodulární dopřednou architekturou se lépe učí klasifikovat polohu objektu, nežli klasifikovat jeho identitu. Síť modulární vykazuje v obou případech stejné schopnosti. Úloha klasifikace identity objektu se ale jeví složitější, nežli klasifikace polohy [14]. Proto také sítě s modulární architekturou mají často v praxi více synapsí mezi poslední skrytou vrstvou a výstupními neurony, jenž klasifikují identitu objektu, nežli neurony, které klasifikují polohu objektu.

Nástroj, který byl vytvořen pro účel této bakalářské práce, není schopen vytvářet sítě s modulární architekturou. S jeho pomocí tak budeme alespoň schopni ověřit vlastnosti nemodulární UNS klasifikovat do zmiňovaných dvou tříd a ověřit, zda platí uváděná tvrzení. Vytvořil jsem proto 25 vzorů, které prozatím nebudeme přiřazovat k žádné množině. Každý vzor představuje rastrový obraz o rozměrech 25x5. V tomto rastru jsem postupně na 5 polohách vytvořil číslice 8, 3, 0, 2, 5. V každém rastru se tak nachází každá číslice na každé poloze, ale vždy právě jedna, viz obrázek 7.4.



Obrázek 7.4 – Vzory použité pro experiment

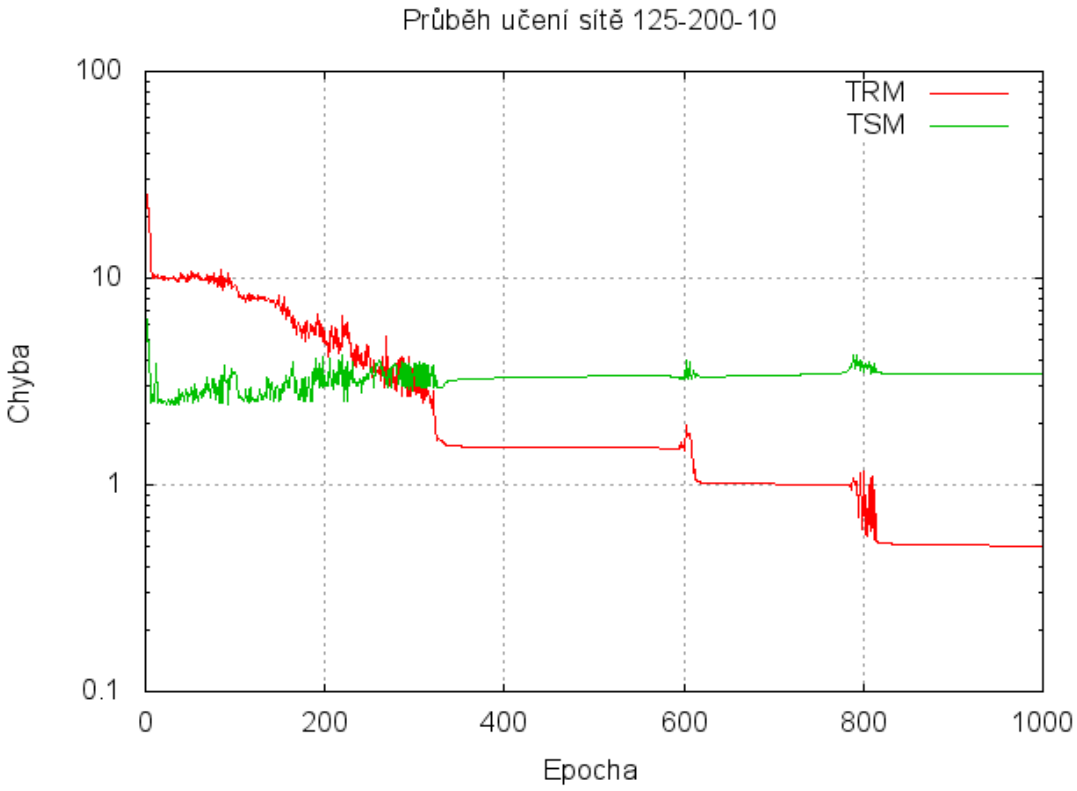
Abychom zjistili, jakým směrem se bude tento experiment odvíjet, vytvořil jsme testovací množinu o pěti vzorcích tak, že jsem z každé číslice a každé její pozice vybral jednoho zástupce. Zbytek, tedy 20 vzorků, posloužil k utvoření trénovací množiny. Dalším problémem bylo získání vhodné architektury sítě. V tomto případě jsem postupoval experimentálně, vytvořením několika namátkově navržených sítí, které jsem postupně podrobil učení s ověřováním pomocí TSM. Ve všech případech byly zvoleny konstantní hodnoty parametrů učení, tj. $\lambda = 1$, $\mu = 0.3$, $\alpha = 0.1$. Minimální dosažené hodnoty chyb TSM pro jednotlivé sítě ukazuje tabulka 7.5.

umělá neuronová síť	min. chyba pro TSM
125-36-10	2.7186463
125-50-10	2.9320822
125-30-10	2.7453587
125-200-100	2.5203121
125-50-35-10	2.7402844
125-80-50-10	2.6324935

Tabulka 7.5 – Dosažená minima chybových funkcí pro TSM

Zatímco pro trénovací množinu, která je zastoupena větším počtem vzorů na rozdíl od testovací množiny se podařilo dosáhnout hodnot chybové funkce lepších, než 0.007, tak v případě testovací množiny výsledky o úspěchu příliš nehovoří. Minimálních hodnot chybové funkce TRM bylo dosaženo u sítí se dvěma skrytými vrstvami. Nejmenší hodnoty chybové funkce TSM pak u sítě 125-200-100. Tuto síť jsem následně vybral a její výsledné vlastnosti jsem se pokoušel optimalizovat změnou parametrů λ, μ, α a dosáhnout tak co nejmenší chyby. Zatímco změna parametr λ spíše zhoršovala výsledky, podařilo se mi nakonec volbou parametrů $\lambda = 1$, $\mu = 0.5$, $\alpha = 0.2$ vybranou UNS naučit s minimální dosaženou chybou pro TSM o hodnotě 2.4328642. Chyby bylo dosaženo ve

24. epoše učení. Průběh učení vystihuje následující graf 7.6, ze které ho je mj. patrné, že průběh chyby pro TSM dále stoupal, což signalizuje možné přeučení UNS.



Graf 7.6 – Průběh chybových funkcí

Naši další pozornost nyní zaměřme na výstupy takto naučené UNS s cílem zjistit, jak si která množina výstupů vede. Zkoumal jsem proto odezvu sítě v závislosti na poloze a konkrétním typu snímaného objektu ve vstupním rastru. Využil jsem funkce vytvořené aplikace, která umožňuje pro přehlednost pojmenovat jednotlivé výstupy sítě a postupně jsem na její vstup přikládal původní vzorky z TSM. Výsledky jasně ukázaly, že rozpoznat pozici zobrazeného čísla v rastru pro síť není problém, jak ukazuje tabulka 7.6. Hodnota na očekávaném výstupu, který byl přiřazen odpovídající pozici znaku v obraze, byla v rozsahu 89% - 98%. Naproti tomu odezva na konkrétní typ znaku, kterou představovala druhá skupina výstupů sítě, byla dosti špatná. Z výsledků tabulky 7.6 neplyne sebemenší náznak závislosti. Z výsledků průběhu chybové funkce pro TRM, zejména u sítí se dvěma skrytými vrstvami, plyne, že pro trénovací množinu bychom byli schopni dosáhnout velmi dobrých výsledků. V případě TSM a pro tuto testovací a trénovací množinu toho nejsme schopni ani po desítkách tisíc epoch učení algoritmem Back Propagation, jak bylo také vyzkoušeno. Uváděné procentuální hodnoty jsou získány jednoduchým přepočtem z výstupního intervalu <0; 1>.

Znak „0“, Pozice 3				
Znak „0“	**Znak „2“**	**Znak „3“**	**Znak „5“**	**Znak „8“**
0.58%	0.63%	3.54%	1.5%	0.54%
Pozice 1	**Pozice 2**	**Pozice 3**	**Pozice 4**	**Pozice 5**
1.87%	2.51%	89.02%	2.56%	6.13%

Znak „2“, Pozice 2				
Znak „0“	**Znak „2“**	**Znak „3“**	**Znak „5“**	**Znak „8“**
1.26%	0.03%	21.16%	5.43%	1,00%
Pozice 1	**Pozice 2**	**Pozice 3**	**Pozice 4**	**Pozice 5**
1.23%	96.79%	2.55%	3.54%	4.24%

Znak „3“, Pozice 4				
Znak „0“	**Znak „2“**	**Znak „3“**	**Znak „5“**	**Znak „8“**
0.87%	0.18%	14.29%	0.10%	2.36%
Pozice 1	**Pozice 2**	**Pozice 3**	**Pozice 4**	**Pozice 5**
0.78%	3.20%	2.20%	92.40%	2.95%

Znak „5“, Pozice 1				
Znak „0“	**Znak „2“**	**Znak „3“**	**Znak „5“**	**Znak „8“**
1.17%	5.34%	1.57%	0,00%	3.10%
Pozice 1	**Pozice 2**	**Pozice 3**	**Pozice 4**	**Pozice 5**
97,00%	1.54%	1.29%	1.78%	0.35%

Znak „8“, Pozice 5				
Znak „0“	**Znak „2“**	**Znak „3“**	**Znak „5“**	**Znak „8“**
10.83%	0.69%	22.42%	2.93%	0.35%
Pozice 1	**Pozice 2**	**Pozice 3**	**Pozice 4**	**Pozice 5**
0.58%	2.78%	1.70%	1.42%	98.09%

Tabulka 7.6 – Odezva výstupů UNS na vzorky z TSM

8 Závěr

Dosažené výsledky u jednotlivých experimentů potvrzují platnost teoretických zákonitostí, týkajících se dopředné vícevrstvé UNS, kterým bylo možné se z hlediska času a složitosti v této práci věnovat. Současně také naznačují možné skutečnosti, které mohou vézt k různým výsledkům, zejména v případě volby parametrů učení algoritmem Back Propagation, nebo změnou počtu neuronů ve skrytých vrstvách. Dosažené výsledky lze v prvních dvou experimentech pokládat za velmi dobré. Jasně ukazují výhody použití umělých neuronových sítí jako klasifikátoru, především díky schopnosti klasifikovat i vzorky, které nebyly k dispozici v trénovací množině při učení dané neuronové sítě. Třetí experiment přinesl očekávané výsledky a poukázal na fakt, že pro dopřednou UNS nečiní problém rozpoznat pozici objektu, ale identifikovat objekt samotný, mění-li pozici v obraze.

Vlastní přínos shledávám v pochopení problematiky umělých neuronových sítí a v možnostech její aplikace v nejrůznějších oborech, především ve zpracování obrazu, jak prezentuje tato bakalářská práce. Za další úspěch považuji implementaci pomocného nástroje, který vznikl jako součást díla této práce. S jeho pomocí je prakticky možné realizovat i další úlohy s podobným zaměřením, případně lze použít jeho význačné části k jiným implementacím. Za nevýhodu z pohledu rychlosti prováděného kódu lze označit použití jazyka Java v němž byl tento nástroj implementován. Další dopad na výkon tohoto nástroje má i objektový návrh, který by jistě bylo možné dále optimalizovat. Uvedené nevýhody lze vyvážit přenositelností výsledného kódu a jednoduchostí při práci se strukturovanými datovými typy a dalšími objekty, jako jsou prvky GUI, atd.

Neuronové sítě lze aplikovat nejen na oblast počítačového rozpoznávání, ale i v jiných případech. Lze se s nimi setkat při klasifikaci signálů (EKG, akustické signály, …), v oblasti automatizace pro řízení systémů, kde tyto UNS mohou emulovat činnost člověka. Také bylo zmíněno jejich použití jako možného přístupu při řešení problematiky týkající se segmentace obrazových dat, atd. Další informace o použití a vlastnostech umělých neuronových sítí lze hledat například v citovaných zdrojích viz dále.

Literatura

[1] Kršek P.: Základy počítačové grafiky (studijní opora do předmětu IZG, verze 0.9). Fakulta informačních technologií VUT, Božetěchova 2, Brno.

[2] Španěl M., Beran V. : Obrazvé segmentační techniky – Přehled existujících metod. 12. října 2005, Fakulta informačních technologií VUT, Božetěchova 2, Brno. Dostupný z WWW: http://www.fit.vutbr.cz/~spanel/segmentace/#_Toc125769322

[3] Hlaváč V., Šonka M.: Počítačové vidění. Vydavatelství Grada, Praha, ČR, 1992.

[4] Shannonův teorém. Wikipedie. Dostupný z WWW: http://cs.wikipedia.org/wiki/Shannon%C5%AFv_teor%C3%A9m

[5] Černocký J.: Signály a systémy (studijní podklady do předmětu ISS). Fakulta informačních technologií VUT, Božetěchova 2, Brno.

[6] Šíma J., Neruda R.: Teoretické otázky neuronových sítí. MATFYZPRESS, 1996, ISBN 80-85863-18-9 .

[7] Zbořil F.: Soft computing (studijní podklady k předmětu SFC). Fakulta informačních technologií VUT, Božetěchova 2, Brno.

[8] Drábek O., Seidl P., Taufer I.: Umělé neuronové sítě – základy teorie a aplikace (1). Univerzita Pardubice, Katedra řízení procesů a výpočetní techniky, CHEMagazín, číslo 4, ročník XV, 2005.

[9] Drábek O., Seidl P., Taufer I.: Umělé neuronové sítě – základy teorie a aplikace (4). Univerzita Pardubice, Katedra řízení procesů a výpočetní techniky, CHEMagazín, číslo 2, ročník XVI, 2006.

[10] Drábek O., Seidl P., Taufer I.: Umělé neuronové sítě – základy teorie a aplikace (5). Univerzita Pardubice, Katedra řízení procesů a výpočetní techniky, CHEMagazín, číslo 5, ročník XVI, 2006.

[11] Horký L., Břinda K.: Neuronové sítě. Fakulta jaderná a fyzikálně inženýrská, Břehová 7, 115 19 Praha 1.

[12] Drábek O., Seidl P., Taufer I.: Umělé neuronové sítě – základy teorie a aplikace (6). Univerzita Pardubice, Katedra řízení procesů a výpočetní techniky, CHEMagazín, číslo 6, ročník XVI, 2006.

[13] Keogh J.: Java bez předchozích znalostí, průvodce pro samouky. CP Books, a.s., Brno, 2005.

[14] Calabretta, R., Di Ferdinando, A., Parisi, D.: Ecological neural networks for object recognition and generalization, Neural Processing Letters, 19: 37-48, 2004.

[15] Menq Joo, Shiqian Wu, Juwei Lu, Hock Ley Toh.: Face recognition with radial basis function (RBF) neural networks.. Transactions on neural networks vol.13, no.3, may 2002. Dostupný z WWW: http://www.dsp.utoronto.ca/juwei/Publication/Juwei_RBF.pdf

[16] Horník J., Krč P.: Rozpoznávání tištěných znaků pomocí LVQ sítí (slidy).
Matematicko-fyzikální fakulta Univerzita Karlova v Praze, Neuronové sítě, 2006/2007.
Dostupný z WWW: http://ksvi.mff.cuni.cz/~mraz/nn/nnpres07/index.html

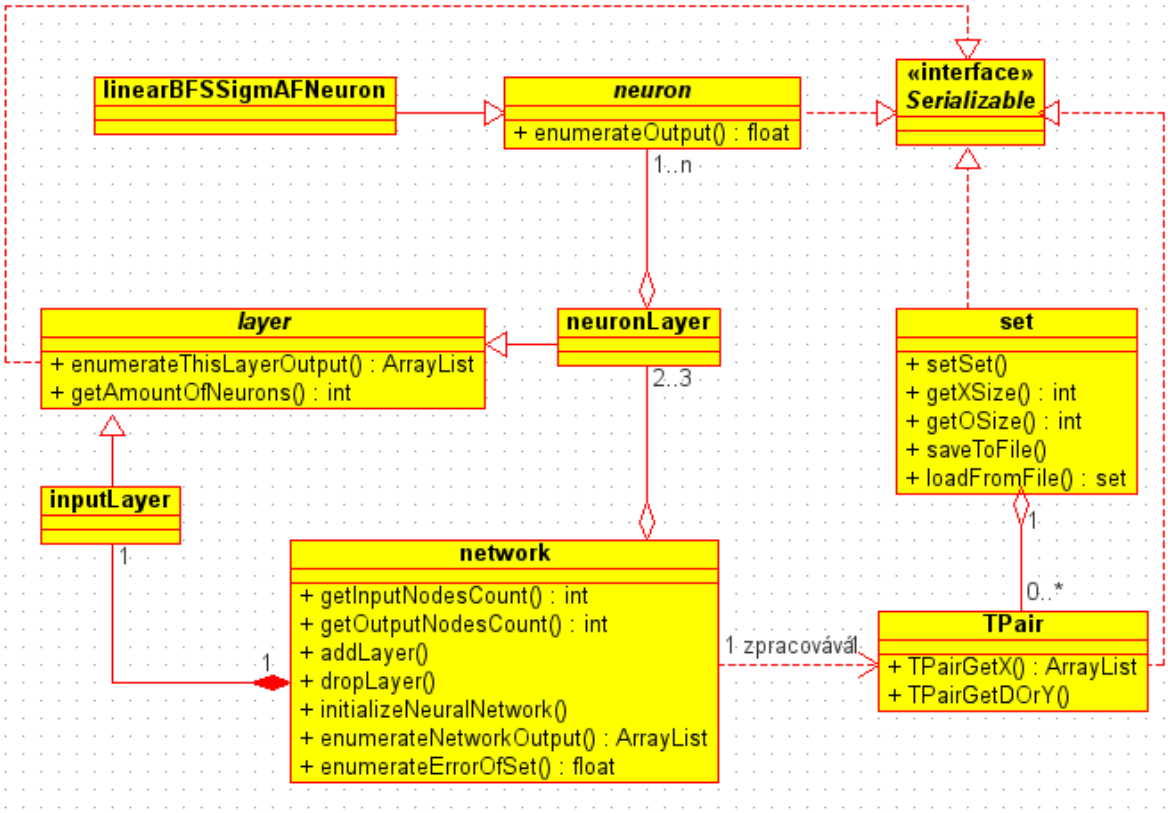
Seznam příloh

Příloha 1. Zjednodušený diagram tříd hlavní části aplikace.

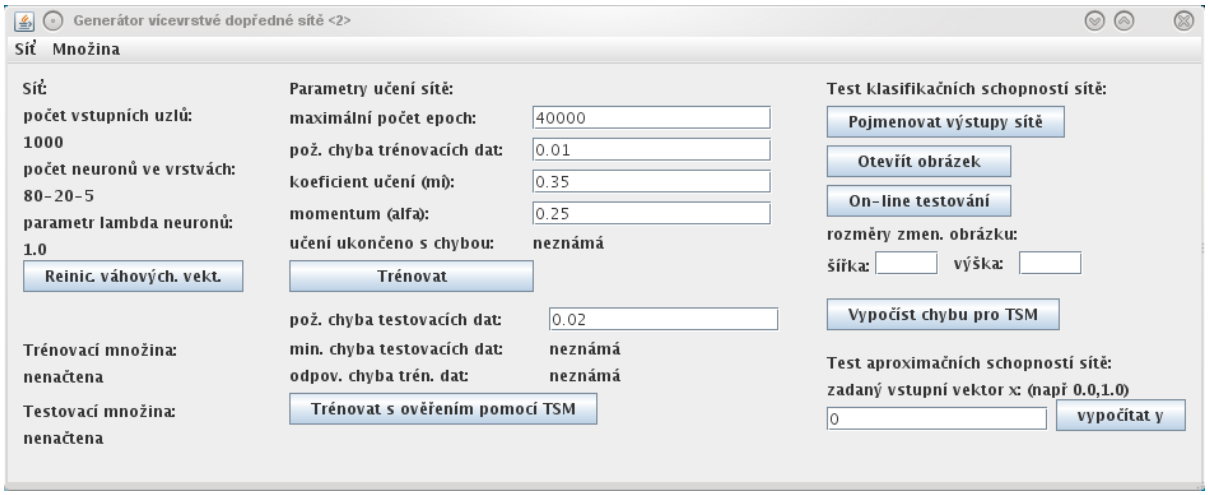
Příloha 2. Podoba GUI aplikace

Příloha 3. DVD obsahující: zdrojový kód aplikace, přeloženou aplikaci, data pro experimenty, výsledky provedených experimentů v podobě UNS, trénovacích a testovacích množin spolu s grafy, které jsou také prezentovány v textu této práce.

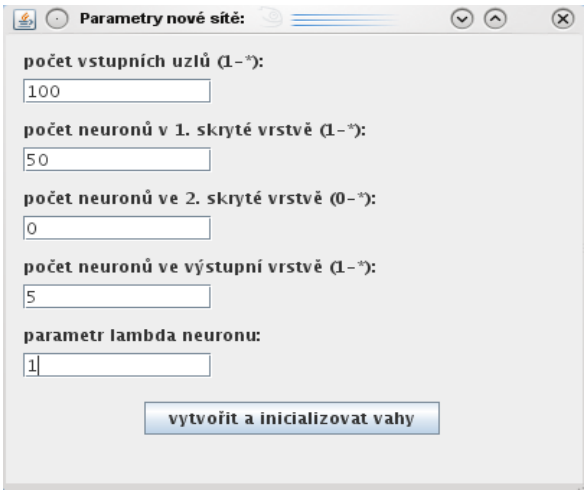
Příloha 1 – Zjednodušený diagram tříd hlavní části aplikace



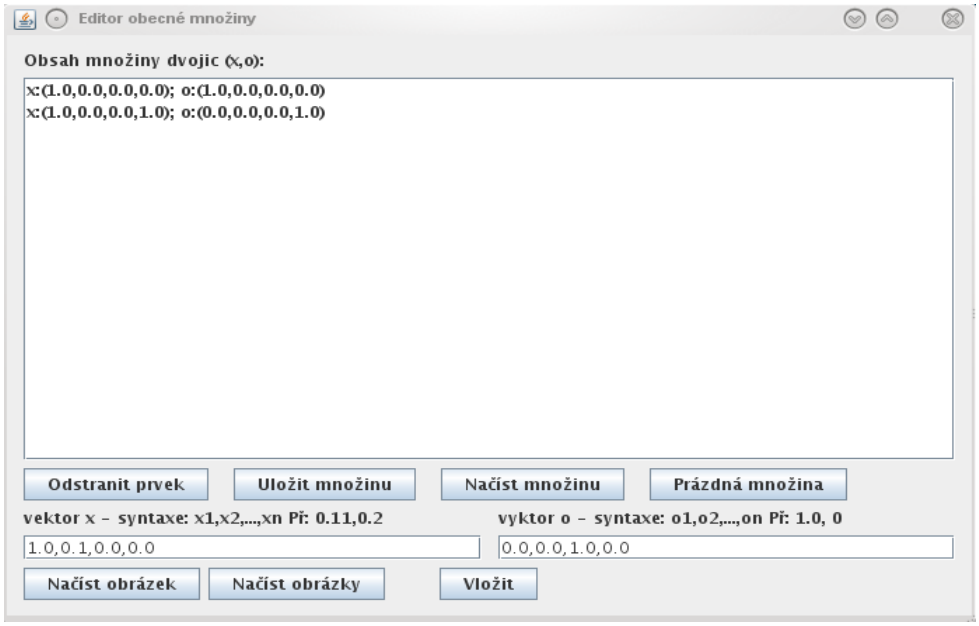
Příloha 2 – Podoba GUI aplikace



Obrázek 1 – Podoba hlavní části aplikace



Obrázek 2 – Rozhraní pro vytváření UNS



Obrázek 3 – Rozhraní pro vytváření obecné množiny