VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

KLASIFIKACE OBRAZŮ S POMOCÍ UMĚLÉ INTELIGENCE

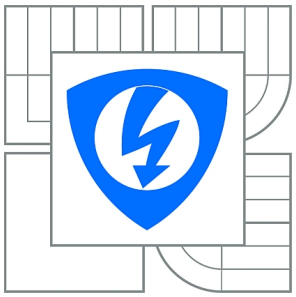
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE ADAM LABUDA
AUTHOR

BRNO 2015

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

KLASIFIKACE OBRAZŮ S POMOCÍ UMĚLÉ INTELIGENCE

IMAGE CLASSIFICATION USING ARTIFICIAL INTELLIGENCE

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

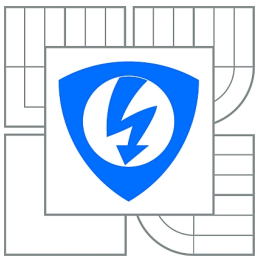
AUTOR PRÁCE — ADAM LABUDA

AUTHOR

VEDOUCÍ PRÁCE — doc. Ing. RADIM BURGET, Ph.D.

SUPERVISOR

BRNO 2015



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Bakalářská práce

bakalářský studijní obor
Teleinformatika

Student:	Adam Labuda	***ID:***	158182
Ročník:	3	***Akademický rok:***	2014/2015

NÁZEV TÉMATU:

Klasifikace obrazů s pomocí umělé inteligence

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se s problematikou klasifikace obrazů a extrakci příznaků z obrazů. V prostředí jazyka JAVA vytvořte příklad, který načte sadu obrázků, extrahuje z ní příznaky a s pomocí umělé inteligence (poskytne vedoucí práce) bude předpovídat druh obrázku. Na vhodném příkladu statisticky zhodnoťte a slovně okomentujte dosažené výsledky.

DOPORUČENÁ LITERATURA:

[1] Hamilton, James Douglas. Time series analysis. Vol. 2. Princeton: Princeton university press, 1994.
[2] Harvey, Andrew C., and A. C. Harvey. Time series models. Vol. 2. New York: Harvester Wheatsheaf, 1993.

Termín zadání:	9.2.2015	***Termín odevzdání:***	2.6.2015

Vedoucí práce: doc. Ing. Radim Burget, Ph.D.
Konzultanti bakalářské práce:

doc. Ing. Jiří Mišurec, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Táto bakalárska práca sa bude zaoberať problematikou klasifikácie obrázkov a extrakcie príznakov z obrazu. V prostredí jazyka JAVA sa vytvorí príklad, ktorý načíta sadu obrázkov, extrahuje z nich príznaky s pomocou umelej inteligencie, ktorú poskytuje vedúci práce. Umelá iteligencia predpokladať druh obrázku. Na záver práce sú porovnané dosiahnuté výsledky.

KĽÚČOVÉ SLOVÁ

Klasifikácia obrázkov, TBIR, CBIR, Načítanie obrazu na farebnej báze, Kruhové sektory, Neurón, Neurónová sieť, Dopredné šírenie, Spätné šírenie, CUDA, OpenCL

ABSTRACT

This bachelor's thesis address the issue of classification and feature extraction of imagesfrom image. In JAVA platform will create an example that loads a set of images, extracted from symptoms with the help of artificial intelligence provided by the thesis supervisor. Artificial intelligence assumed kind of image. Finally the results are compared.

KEYWORDS

Image clasiffication, TBIR, CBIR, Color-based image retrieval, Method circular sectors, Neuron, Neural network, Feed-forward propagation, Backpropagation, CUDA, OpenCL

LABUDA, Adam *Klasifikácia obrázkov pomocou umelej inteligencie*: bakalárska práca. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2015. 48 s. Vedúci práce bol doc. Ing. Radim Burget, Ph.D.

PREHLÁSENIE

Prehlasujem, že som svoju bakalársku prácu na tému „Klasifikácia obrázkov pomocou umelej inteligencie“ vypracoval(a) samostatne pod vedením vedúceho bakalárskej práce, využitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autor(ka) uvedenej bakalárskej práce ďalej prehlasujem, že v súvislosti s vytvorením tejto bakalárskej práce som neporušil(a) autorské práva tretích osôb, najmä som nezasiahol(-la) nedovoleným spôsobom do cudzích autorských práv osobnostných a/nebo majetkových a som si plne vedomý(-á) následkov porušenia ustanovenia § 11 a nasledujúcich autorského zákona č. 121/2000 Sb., o právu autorském, o právoch súvisejúcich s právom autorským a o zmeně niektorých zákonov (autorský zákon), vo znení neskorších predpisov, vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovenia časti druhé, hlavy VI. diel 4 Trestného zákoníka č. 40/2009 Sb.

Brno ....

podpis autora(-ky)

POĎAKOVANIE

Rád bych poděkoval vedoucímu diplomové práce panu doc. Ing. Radimovi Burgetovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnetné návrhy k práci.

Brno ....

podpis autora(-ky)

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Purkynova 118, CZ-61200 Brno
Czech Republic
http://www.six.feec.vutbr.cz

POĎAKOVANIE

Výzkum popsaný v tejto bakalárskej práci bol realizovaný v laboratóriách podporených projektom SIX; registračné číslo CZ.1.05/2.1.00/03.0072, operačný program Výzkum a vývoj pro inovace.

Brno . . . . . .

. . . . . .
podpis autora(-ky)

OBSAH

ZOZNAM OBRÁZKOV

ZOZNAM TABULIEK

1 ÚVOD

V posledných rokoch zaznamenávame prudký nárast fotografii či už na internete alebo vo svojom počítači. Digitálne fotky sa stali nástrojom ako si zvečniť rôzne zážitky či spomienky. Digitálny fotoaparát sa stal bežnou súčasťou mobilných telefónov či tabletov. Fotky zdieľané cez internet sa stali každodennou praxou. Fotografie zdieľané cez internet zachytávajú príbehy jednotlivcov či skupín po celom svete. Zdieľané fotografie väčšinou putujú do on-line úložísk kde tvoria osobné fotogalérie. Možnosť vyhľadať fotografiu, ktorá bola pridaná včera či minulý rok možno pomocou kľúčových slov – anotácii, ktoré sú s obrázkom zlúčené. Metóda, ktorá nám uľahčuje hľadanie pomedzi množstvom fotografii na základe anotácie sa nazýva Text-based Image Retrieval (TBIR). Táto metóda vyhľadáva obrázky práve na základe anotácii.

Pri pridávaní anotácii k obrázkom, často závisí na subjektívnom postoji danej osoby. Každý jeden človek sa v niečom líši od toho druhého či už výzorom, správaním ale tak isto aj vnímaním. Preto nie každý označí obrázok rovnakým názvom. V podstate pridávanie kľúčových slov k obrázkom je zložité, keďže ide o čo najobjektívnejšie pomenovanie daného obrázku. Z druhej strany, stroj nedokáže vnímať niektoré podnety viac a niektoré menej. To aby sa stroj naučil obrázok lepšie rozoznať nám pomáhajú neurónové siete. Vďaka týmto sieťam dokážeme stroj na klasifikáciu obrázkov v podstate učiť. Neurónová sieť je v podstate zložená z preceptrov, ktoré sú prepojené rôznymi cestami s ďalšími. Väčšinou máme vstup, ktorý reprezentuje obrázok. Každý jeden pixel je prepojený so skrytou vrstvou. Skrytá vrstva má menej jednotiek ako vstup. Preceptrony skrytej vrstvy sú napojené na takzvanú “jedničku“ pomocou, ktorej dokážeme meniť stavy skrytej vrstvy. Za skrytou vrstvou býva výstup s rovnakým počtom jednotiek, pixlov ako vstup. Tým pádom z takto navrhnutého stroja sa stáva umelá inteligencia ktorú dokážeme trénovať.

Rozvoj efektívnych metód pre automatické klasifikovanie obrázkov naďalej vyzýva počítačových vedcov aby prišli s niečím novým a prevratným. Samostatná myšlienka, že by stroje dokázali vnímať obrázky ako ľudia je fascinujúca a odstrašujúca zároveň. Väčšina ľudí si ani neuvedomuje, že s podobnými strojmi prichádzajú skoro každodenne do osobného kontaktu. Či už je to rozpoznávanie tváre alebo samotné vyhľadávanie obrázkov. Zaujímavé to začína byť vtedy keď stroj nemá žiadnu anotáciu či popis obrázku. Stroj teda musí zistiť čo obrázok reprezentuje bez anotácii.

Táto práca sa bude venovať vhodnému naprogramovaniu stroja, ktorý bude klasifikovať obrázky na základe postupného učenia. Naprogramujem stroj, ktorý sa bude skladať zo vstupu, skrytej vrstvy, jednotkovej vstupu a výstupu. Budem sa snažiť

takto naprogramovaný stroj učiť čo najlepšie klasifikovať obrázok. Na začiatku pridám náhodne generované váhy. Potom stroj necháme pracovať a budeme sledovať výsledky na výstupe. Začiatočné chybné výstupy budú postupne opravované pomocou backpropagation.

Zbytok práce je organizovaný nasledovne. Kapitola 1 sa zaoberá metódami získavania príznakov z obrazu. Sú to metódy ako TBIR technika, CBIR technika a metóda založená na farebnom spektré. V kapitole 2 sa preberajú metódy umelej inteligencie ako neurón a neurónové siete. V kapitole 3 sú rozpracované výpočty na grafických kartách v podaní CUDA a OpenCL. V poslednej kapitole 4 sú pojednávané výsledky a zvolená metóda.

2 METÓDY ZÍSKAVANIA PRÍZNAKOV U OBRAZU

Klasifikácia obrazu na základe umelej inteligencie má širokospektrálne využitie v rôznych oblastiach od elektrotechniky až po medicínu. Na získanie príznakov z obrazu môžeme použiť niekoľko metód. V mojej práci na začiatku spomeniem metódu na vyhľadávanie obrazu pomocou anotácie, potom sa budem venovať metódam, ktoré vyhľadávajú na základe obsahu obrazu.

2.1 TBIR technika

Tento systém ako som už spomínal, pracuje s anotáciami pridanými k obrázkom. Tieto anotácie sú známe ako metadáta obrázku, sú to napríklad informácie o dátume, rozlíšení a samostatný pridaný popis obrázku. Tieto informácie, ktoré sú pridané k obrázku pomáhajú systému nájsť hľadaný obrázok. Tento systém využíva napríklad Google alebo nemocnice, ktoré organizujú lekárske obrázky do archívov. I keď táto technika môže poskytnúť flexibilné vyhľadávanie obrázkov, slová nedokážu zachytiť informácie o vizuálnom obsahu, farbách alebo textúrach.
Množstvo práce potrebnej na anotovanie každého obrázku ručne s pravdepodobnosťou subjektívneho popisu, ktorý nemusí zahŕňať celý obsah obrázku, by mohlo viesť k následnému nepresnému vyhľadaniu obrázkov. Využívať túto techniku na vyhľadávanie obrazu je pre širšie použitie nie je moc efektívne. Prikláňam sa skorej k možnosti vyhľadávania obrazu na základe obsahu.

2.2 CBIR technika

Informácie popísané v tomto texte boli čerpané z literatúry [5] [1]

Každý deň veľa počítačových či mobilných aplikácii pridávajú terabajty multimédií na úložiská. Obrovské množstvá obrázkov či už z umenia, medicíny alebo iných sektorov, musia byť veľmi dobre organizované aby bolo možné rýchle a účinné hľadanie, prezeranie a rozpoznávanie. Výskumníci sa zaoberajú hľadaním obrázkov na základe dvoch rôznych pohľadov a to z pohľadu textového a vizuálneho hľadiska. Textové metódy používajú kľúčové slová a poznámky zatiaľ čo vizuálne metódy pracujú s obsahom obrazu ako je farba, tvar, štruktúra alebo priestorové rozloženie.

Content-based image retrieval (CBIR) je technika, ktorá používa vizuálny obsah na vyhľadanie a porovnávanie obrázkov z rozsiahlych obrázkových databáz podľa záujmov užívateľa, kde každý obraz vlastní unikátny podpis. Existuje mnoho bežných popisov funkcií. Pre príklad, funkcie farby môžu byť reprezentované histogrammy. Tak isto textové vlastnosti reprezentovaná napríklad Gáborovou transformáciu. Okrem toho tvarové vlastnosti môžu byť reprezentované pomocou Furierovej bázy segmentácie. Posledné priestorové vlastnosti môžu byť reprezentované podľa 2D reťazca. V bežných CBIR systémoch vizuálny obsah obrázku je extrahovaný a nahradený pomocou milti-dimensional features vectors. Tieto vlastnosti vektorov tvoria popis databázy.

Vizuálny obsah popisu môžeme rozdeliť do globálneho alebo lokálneho popisu v závislosti na úrovni detailnosti informácii, ktorý popis zastupuje. Globálny popis reprezentuje vizuálne vlastnosti celého obrázku. I keď globálne popisy sú široko využívané a ľahko implementovatelné, pritom však majú obmedzené popisné možnosti. Lokálny popis využíva vizuálne vlastnosti objektov alebo regióny na reprezentovanie obsahu obrazu.

V každom CBIR systéme, používateľ poskytuje databázu napríklad s obrázkami. Systém potom interpretuje tento vstup do svojho popisu – vlastnosti vektorov. Potom systém vypočíta vzdialenosti medzi týmito vektormi na základe dotazu. Načítanie je vykonané pomocou režimu indexovania kvôli efektívnemu vyhľadaniu v databáze obrázkov.

2.3 Color based image retrieval technika

Táto technika je najzákladnejším a najdôležitejším spôsobom CBIR. Farebné prvky sú najviac intuitívne a najvýraznejšie opisujú rysy obrazu. V porovnaní s inými obrazovými funkciami ako štruktúra alebo tvar, farebné prvky sú veľmi stabilné a robustné. Nie sú náchylné na rotáciu, preklad alebo iný rozsah zmien. Navyše výpočet farebných príznakov je jednoduchý. Histogram farieb je najpoužívanejšou metódou pre extrakciu farebných príznakov.

Histogram farieb je štatistická frekvencia pre rôzne farby v určitom farebnom priestore. Výhodou je, že popisuje globálnu farebnú distribúciu pre obrázky, tým pádom je vhodná pre tie obrázky, ktoré sa napríklad ťažšie segmentujú. Avšak nevýhodou je, že nemôže opísať lokálne rozdelenie obsahu obratu vo farebnom priestore

a priestorové postavenie každej farby. To zanemená, že histogram farieb nedokáže opísať konkrétne objekty alebo veci v obraze. Na základe toho musí byt farebný priestor rozdelený do niekoľkých menších rozsahov aby sa dal vypočítať histogram farieb. Každý interval je považovaný za tzv. bin. To znamená, že farba je kvantovaná. Histogram farieb sa vypočíta na základe počítania pixlov, kde farby zapadajú do každého intervalu. Medzi farebné funkcie patria globálne farebné histogramy a bloky histogramov farieb [4].

2.3.1 Globálny histogram farieb na základe CBIR

Pre vyhľadávanie obrázku z databázy sa zvyčajne vybraný obraz prevedie z RGB priestoru iného farebného priestoru. Globálny histogram farieb sa možno vypočítať v štyroch krokoch[4]:

- **Krok 1:** Prevedenie obrazu z RGB priestoru do HSV priestoru
- **Krok 2:** Kvantifikovanie obrazu na základe vzorca 2.1

$$H = \begin{cases} 0 & h \in [316, 360] \\ 1 & h \in [1, 25] \\ 2 & h \in [26, 40] \\ 3 & h \in [41, 120] \\ 4 & h \in [121, 190] \\ 5 & h \in [191, 270] \\ 6 & h \in [271, 295] \\ 7 & h \in [295, 315] \end{cases} \tag{2.1}$$

- **Krok 3:** Výpočet každej funkčnej hodnoty
- **Krok 4:** Výpočet podobnosti pomocou Euklidovskej vzdialenosti vo vzorci 3.1

$$D = \sum_{i=1}^{n} (A_i - B_i)^2 \tag{2.2}$$

Globálny histogram farieb je jednoduchý spôsob ako extrahovať informácie z obrázkov. Vysoko efektívny výpočet a pružnosť sú hlavné výhody tejto techniky. Možnou nevýhodou tejto techniky je, že globálny histogram farieb počíta iba frekvenciu farieb. Priestorové informácie o farbách sú tým pádom stratené. To znamená, že dva rozličné obrázky môžu dostať ten istý globálny histogram farieb čo spôsobuje chyby.

2.3.2 Blokový histogram farieb na základe CBIR

Pre blokové histogramy je obraz rozdelený do blokov o rozmeroch x*y. Každý blok bude mať menší význam v prípade, že blok bude príliš veľký. Výpočet procesu vy-

hľadávania bude rýchlejší pokiaľ bude blok príliš malý. Najefektívnejšie bude ak sa dvojrozmerný priestor rozdelí do blokov s rozmermi 3*3. Každý jeden blok je sprevádzaný výpočtom farebného priestoru a farebnej kvatizácie. Zvyčajne za účelom zdôraznenia špecifických váh rôznych blokov, distribuuje váhový koeficient pre každý jeden blok. Váha stredného bloku je väčšinou väčšia ako ostatné.

Globálny histogram farieb počíta celkovú farebnú informáciu z obrázka bez priestorového rozloženia. V tomto prípade sa nám môže stať, že systém vyhľadá obrázok, ktorý nemá nič spoločné s dotazom. Je to zapríčinené tým, že celková farba je podobná vstupnému obrázku ale pri tom vizuálny obsah je úplne iný. Pričom blokový histogram farieb berie obrázok po určitých blokoch tým pádom rozdelí obrázok na rovnako veľké časti aby predišiel chybám. Blokové histogramy farieb majú lepšie výsledky ako globálne histogramy farieb.

2.3.3 Výpočet matice výskytu

Predstavme si vstupný obrázok, ktorý má *Mc* a *Mr* pixlov v horizontálnom a vertikálnom smere. Predpokladajme, že Zc=1, 2, ..., Mc je horizontálny priestor a Zr=1, 2, ..., Mr je vertikálny priestor. Ak je daný smer θ a vzdialenosť d, prvok matice P(i, j/d, θ) môže byť vyjadrená výpočtom pixlov logaritmu výskytu šedej úrovne i a j. Použime vzdialenosť 1 a θ sa bude rovnať 0°, 45°, 90°, 135° dostaneme vzorce:

$$P(i,j/1,0) = \left\{ \begin{array}{c} [(k,l),(m,n)] \in (Zr\ddot{O}Zc) \\ |k\text{-}m| = 0, |l\text{-}n| = 1 f(k,l) = i, f(m,n) = j \end{array} \right\} \tag{2.3}$$

$$P(i,j/1,45) = \left\{ \begin{array}{c} [(k,l),(m,n)] \in (Zr\ddot{O}Zc)(k?m) = 1, (l\text{-}n) = \text{-}1 \\ (k\text{-}m) = \text{-}1, (l\text{-}n) = 1, f(k,l) = i, f(m,n) = j \end{array} \right\} \tag{2.4}$$

$$P(i,j/1,90) = \left\{ \begin{array}{c} [(k,l),(m,n)] \in (Zr\ddot{O}Zc) \\ |k\text{-}m| = 1, |l\text{-}n| = 0, f(k,l) = i, f(m,n) = j \end{array} \right\} \tag{2.5}$$

$$P(i,j/1,135) = \left\{ \begin{array}{c} [(k,l),(m,n)] \in (Zr\ddot{O}Zc)(k\text{-}m) = 1, (l\text{-}n) = 1 \\ (k\text{-}m) = \text{-}1, (l\text{-}n) = \text{-}1, f(k,l) = i, f(m,n) = j \end{array} \right\} \tag{2.6}$$

Kde k, m a l, n reprezentujú zmeny vo vybranom výpočtovom okne, W reprezentuje logaritmy pixlov, ktoré sú v zátvorke.

2.3.4 Extrakcia textúr

Tú metódu možno popísať 5 krokmi.

- **Krok 1:** Prevod farieb obrazu
  Farebný obraz bude prevedený na úrovne šedej podľa vzorca 7 v rozmedzí stupnice 0 - 255.

$$Y = 0.29\ddot{O}R + 0.587\ddot{O}G + 0.114\ddot{O}B \tag{2.7}$$

  kde Y je hodnota šedej úrovne

- **Krok 2:** Kvantifikácia šedej úrovne
  Keďže šedá stupnica dosahuje hodnoty v rozmedzí 0 - 255, prípadná matica má teda tvar 256*256. Šedé úrovne z pôvodného obrazu sa najskôr stlačia aby sa znížil počet výpočtov pred tým než je samotná matica výskytu vytvorená. Prebehne zhruba 16 úrovní kompresií. Nakoniec hodnoty a smerodajné odchýlky každého parametra sa branú ako komponenty každej textúrovej vlastnosti.

- **Krok 3:** Výpočet hodnoty funkcie
  Vytvoria sa štyri matice výskytu na základe vzorcov 2.3, 2.4, 2.5 a 2.6 v štyroch smeroch. Vypočítajú sa štyri textúrové parametre ako kapacita, entropia, moment zotrvačnosti a relevantnosti.

- **Krok 4:** Interná normalizácia
  V tomto kroku sa požíva Gaussova normalizácia na základe, ktorej je zaistené aby jednotlivé funkcie mali rovnaké váhy.

- **Krok 5:** Porovnanie textúr
  Textúrové rysy každej snímky sa vypočítajú na základe týchto krokov. Textúrové hodnoty sú porovnané pomocou Euklidovej vzdialenosti, čím bližšia vzdialenosť tým vyššia podobnosť.

Pri vyhľadávaní obrazu, vlastnosti vektorov každého obrazu z databázy odpočítajú vlastnosti vektorov cieľového obrazu oddelene ako funkcia hodnôt. Suma výsledných štvorcov.

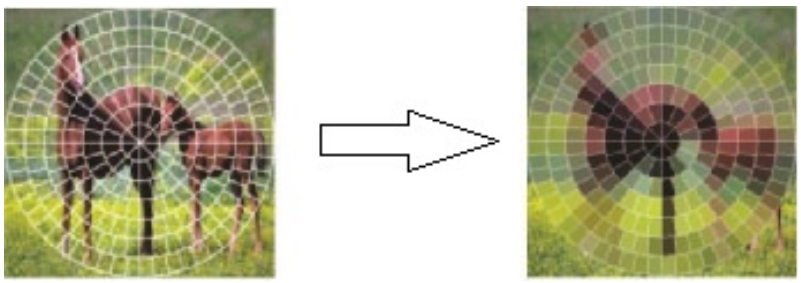
Pri takomto systéme dosahujeme výsledky podobné vstupnému obrazu z hľadiska farby, pokiaľ použijeme histogram farieb. Za požívania iba textúrových vlastností sa obrázok zmení na čierno biely. I keď obraz obsahuje rôzne farby, textúry sú si veľmi

podobné. Textúrové vlastnosti extrahované na základe šedých úrovní co-occurred matice, ktoré patria k celej doméne snímky. Poukazuje sa viac na závislosť šedého priestoru, ktorý odráža pixle šedej úrovne podľa rovnakej štruktúry. Pomocou jednej vyhľadávacej funkcie nemožno dosiahnuť uspokojujúci výsledok.

2.4 Kruhové sektory

Metóda kruhových sektorov popísaná v literatúre [1] hovorí o rozdelení obrazu na centralizované kruhy. Stred obrazu je logicky najdôležitejšia oblasť či už pri fotení alebo pozeraní na obrázok. Pohľad ide priamo do stredu obrazu a potom sa presúva k okrajom. Táto skutočnosť je základom spôsobu extrahovania farebných vlastností. CBIR systémy volia radšej slabšiu segmentáciu pred silnou aby predišli krehkosti.

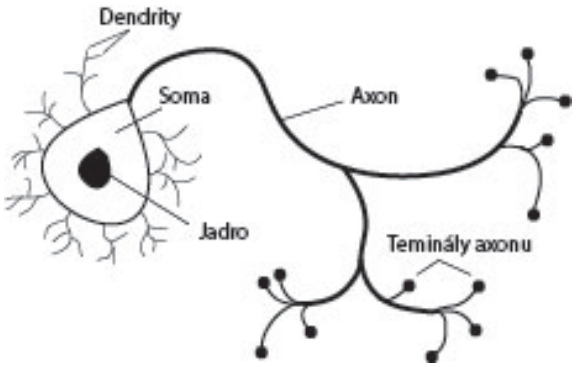
V tejto metóde sa obrázok rozdelí do C_i kruhov a tie sú rozdelené do S_i sektorov kde sú ignorované okraje ako je zobrazené na obrázku 2.1. V každom kruhu je počet sektorov $S_i = 8xC_i$ kde Ci predstavuje počet kruhov. Počet kruhov má vplyv na kvalitu výsledkov, požiadavky na skladovanie popisov v databáze a výpočet skúšky podobnosti. Tieto tri sa zvyšujú s narastajúcim počtom kruhov. I keď je žiaduce mať vyššiu kvalitu výsledkov je nutné zachovať požiadavky na skladovanie popisov a výpočty na rozumnej úrovni. Bolo zistené, že 7 kruhov je vhodný počet. Tým pádom prvý kruh bude mať 8x1=8 sektorov a poseldný 8x7=56 sektorov. Celkový počet sektorov teda bude 224. Pri počte 224 sektorov a troch farebných kanáloch dostaneme 672 hodnôt.



Obr. 2.1: Rozloženie obrazu kruhovými sektormi

3 METÓDY UMELEJ INTELIGENCIE

Mozgová kôra je tvorená veľkým počtom buniek, ktoré sú navzájom poprepájané vláknami ako je zobrazené na obrázku 3.1. Počet neurónov v mozgu je neskutočne obrovský – asi sto miliárd. Každý jeden neurón vytvára spojenia s ďalšími 10 000 neurónmi. Miesto vzájomných kontaktov medzi neurónmi sa nazýva synapsia. Synapsie sa trvalo a rýchlo stavajú a prestavujú podľa toho, ako mozog so získanými informáciami pracuje [11].



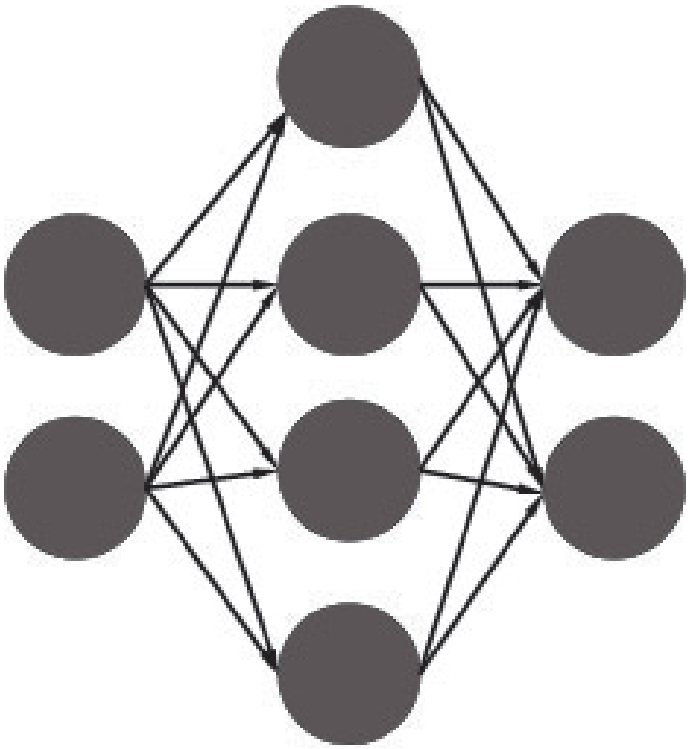
Obr. 3.1: Neurón

Ľudský mozog môže byť opísaný ako biologické neurónové siete. Prepojenie sietí neurónov, ktoré vysielajú komplikované vzory elektrických signálov. Ako vlastne pracuje ľudský mozog a ako vlastne myslíme, je komplikované a zložité tajomstvo, ktoré sme zatiaľ neodhalili. Je to jeden z dôvodov pre neustále štúdium tejto problematiky. Informácie, ktoré zatiaľ poznáme čerpáme predovšetkým s neurofyziológie, ktorá umožnila vytvoriť zjednodušené matematické modely. Stroj nedokáže dôveryhodne opísať ľudský mozog ale dokáže sa aspoň trochu priblížiť jeho správanie pomocou dobre zvolených algoritmov [10].

Korene neurónových štúdií siahajú na začiatok minulého storočia, kde sa vedci snažili prísť na to ako funguje náš nervový systém. V roku 1890 americký filozof William James vyslovil teóriu v, ktorej tvrdí, že množstvo aktivity v danom mieste na mozgovej kôre je sumou všetkých tendencií, ktoré boli do nej vpustené [2]. Neskôr v roku 1943 neurológ Warren S. McCulloch a logik Walter Pitts, vyvinuli prvý koncepčný model neurónovej siete. V článku “ A logical calculus of the ideas imminent in nervous activity“ opisujú koncept neurónu a jednej bunky žijúcej v sieti buniek, ktorá dostane informáciu, spracuje ju, a následne generuje výstup. Ich práca a práce iných vedcov

a výskumníkov tej doby, nebolo presne opísať ako biologický mozog funguje, ale skôr navrhnúť umelú neurónovú sieť ako výpočtový model mozgu, ktorý by dokázal riešiť určité typy problémov. V tomto modeli boli číselné hodnoty parametrov prevažne bipolárne. Vďaka tomuto modelu dokázali ukázať vypočítať ľubovoľné aritmetické alebo logické funkcie. Začiatkom 80. rokov dochádza k znovu oživení tohto vedného odboru. V roku 1986 bol pánmi Rumelhartom, Hintonom a Williamsom znovu objavený učiaci sa algoritmus viac vrstvovej neurónovej siete. Tento algoritmus je známy pod menom back-propagation - spätná propagácia. I keď back-propagation nie je úplne ideálny algoritmus pre učenie neurónových sietí s ľudským uvažovaním, dokáže avšak riešiť mnoho problémov [14].

Najbežnejšie použitie neurónových sietí v oblasti výpočtovej techniky dneška je vyriešiť jeden z problémov " easy-for-a-human, difficult-for-a-machine" (ľahko pre človeka, ťažšie pre stroj). Uplatnenia sa pohybujú od optického rozpoznávania znakov až po rozpoznanie tváre. Výpočtové systémy sú procesné. Program začína v prvom riadku kódu, spustí ho a program pokračuje na ďalší riadok kódu a na nasledujúci v lineárnym spôsobom. Neurónová sieť ale nesleduje lineárnu cestu. Informácie sú skôr spracované spoločne a paralelne po celej sieť uzlov, neurónov. Jednotlivé prvky siete sú jednoduché. Načítajú vstup, vyhodnotia ho a generujú výstup. Sieť niekoľkých neurónov môže však vykazovať neuveriteľné inteligentné správanie. Jedným z kľúčových prvkov neurónovej siete je schopnosť učiť sa. Ako som spomínal neurónová sieť nie je zložitý systém, ale ako komplexný a flexibilný systém čo znamená meniť vnútornú štruktúru na základe informácií je trošku zložitejšie. Zvyčajne sa k takémuto chovaniu priblížime vďaka implementovaným váham.



Obr. 3.2: Prepojenie neurónov

Na obrázku 3.2 každé spojenie reprezentuje prepojenie dvoch neurónov a ukazuje cestu, tok informácií. Každé spojenie má určitú váhu a číslo, ktoré kontroluje prepojenie medzi dvoma neurónmi. V prípade, že sieť generuje pozitívne výstupy pre náš systém, nie je nutné upravovať váhy. Pokiaľ sieť generuje negatívne výstupy, systém zmení váhy, aby zlepšil nasledujúce výsledky. Existuje niekoľko spôsobov ako učiť systém a to:

Supervised Learning (učenie s učiteľom) - ide o stratégiu kde sa nachádza učiteľ, ktorý je múdrejší ako sieť. Učiteľ ukáže sieti hrsť obrázkov pričom pozná popis spojený z každým obrázkov. Sieť vykoná svoje odhady a potom učiteľ ukáže správnu odpoveď. Potom sieť porovná svoje výsledky so správnou odpoveďou a vykoná úpravy na základe chýb.

Unsupervised Learning (učenie bez učiteľa) - v tomto spôsobe učenia nepoznáme správnu odpoveď. Hľadá sa skrytý vzor obrazu v súbore dát aplikáciou zhlukovania, ktorá rozdelí súhrnu prvkov do skupín podľa určitých vzorov.

Reinforcment Learning (zosilnenie učenia) - stratégia zakladaná na skúmaní.

Pokiaľ je skúmanie negatívne, sieť môže zmeniť váhy tak, aby zmenilo nasledujúce výsledky. Pre lepšie pochopenie tohto systému si predstavme myš, ktorá beží bludiskom. Dostane sa na rázcestie. Pokiaľ odbočí doprava dostane syr, ale ak odbočí doľava dostane šok. Dá sa predpokladať, že sa myš naučí postupným učením, naučí cestu k syru. Táto stratégia sa využíva najčastejšie v robotike. V čase t, robot vykoná úlohu a zachová si výsledky.

Pattern Recognition (rozpoznávanie tvaru) – najčastejšie využívaná, príklad na rozpoznávaní tvárí či rozpoznávanie znakov.

Time Series Prediction (predikcia časovej rady) – táto neurónová sieť môže byť použitá na predpovedanie určitej skutočnosti.

Signal Procesing (spracovanie signálu) – táto sieť je trénovaná na spracovanie zvukového signálu a následné filtrovanie zvuku aby mohla pomôcť lepšie spracovať zvuk pre nedoslýchavých.

Soft sensors (jemné senzory) – senzory počítajú analýzy zbierku rôznych meraní. Chcete vedieť vlhkosť vzduchu. Túto informáciu vám senzor môže poskytnúť ale čo keby vedel ďalšie informácie ako teplotu, tlak, hustotu vzduchu a kvalitu ovzdušia. Neurónové siete sa používajú na spracovanie mnohých vstupných parametrov a vyhodnocujú ich ako celok.

Anomaly Detection (detekcia anomálií) – keďže neurónové siete sú ozaj dobré v rozpoznávaní vzorov, môžu byť tiež trénované na to, aby odhalili anomálie. Keď sa systém naučí určitý sled postupov alebo správaní, stane sa niečo s čím systém nepočítal a upozorní na tento fakt.

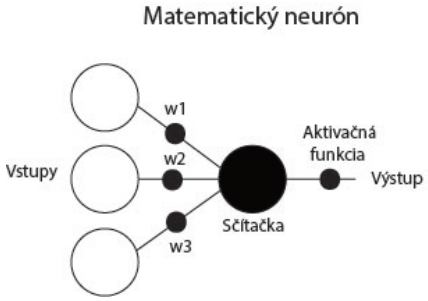
Toto určite nie je úplný zoznam uplatnenia neurónových sietí. Je to spracovanie len tých metód, ktorými sa v podstate stretávame najčastejšie. V skutočnosti neurónové siete sú ťažké a zložité pre laické pochopenie. Dajú sa v nich uplatniť všetky druhy matematiky avšak, je toto všetko úžasné a fascinujúce, mnoho techník nie je praktických.

Systémy dnešnej doby sú veľmi špičkovo naprogramované tak aby simulovali biologické neurónové siete a procesy v sieti, ale napriek tomu sú stále nedostačujúce na to aby reprezentovali ľudský mozog. Schopnosť neurónovej siete učiť sa a časom

vykonať úpravy v konštrukcii je to, čo robí sieť tak užitočnú v oblasti umelej inteligencie. Možnosti využitia sú veľké. Štandardne sa ale využívajú v týchto softvéroch:

3.1 Umelý neurón

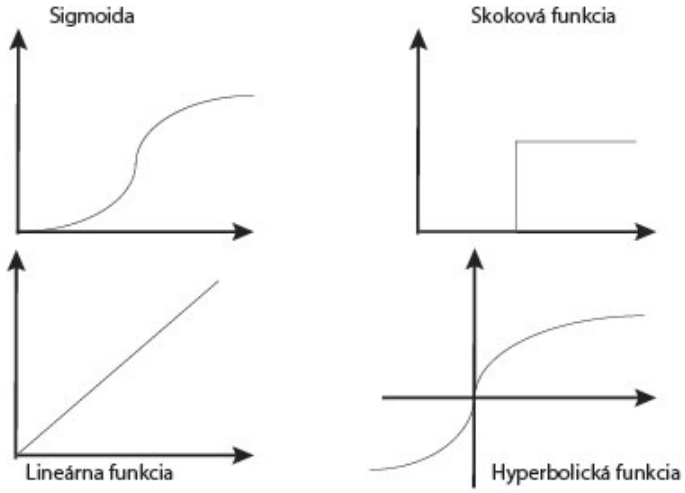
Je to základná stavebná jednotka neurónovej siete. Ide teda o umelý neurón. Neurón sa skladá z viacerých vstupov, ale iba z jedného výstupu ako je naznačené na obrázku 3.3.



Obr. 3.3: Neurón

Tento neurón má tri vstupy ku, ktorým sa pripočítajú určité váhy w1, w2 a w3. Váhy sú vynásobené s informáciami zo vstupov a následne putujú cez neurón do aktivačnej funkcie kde je určená funkčná hodnota aktivačnej funkcie vnútorného potenciálu neurónu.

Aktivačná funkcia môže byť skoková, spojitá nelineárna alebo spojitá lineárna ako je zobrazené na obrázku 3.4.



Obr. 3.4: Aktivačné funkcie

Výstup neurónu je určený funkčnou hodnotou aktivačnej funkcie f(x), vnútorného potenciálu neurónu. Činnosť neurónu možno vyjadriť vzťahom 3.1.

$$y = f \sum_{i=1}^{N} x_i * w_1 - \theta \tag{3.1}$$

Kde:

y vstup neurónu

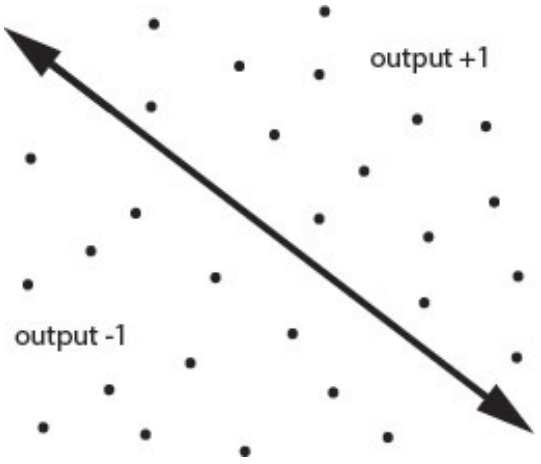
f aktivačná funkcia taktiež označovaný ako vnútorný potenciál neurónu

xi vstupy neurónu

wi váhové vstupy neurónu

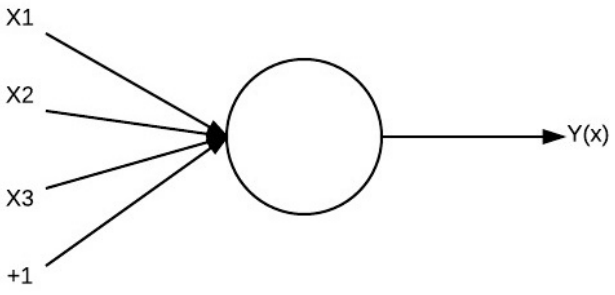
θ prah neurónu

To nie je ale všetko čo by sme mali vedieť o neuróne. Rozdeľme čiarou 2D priestor ako je ukázané na obrázku 3.5. Body v tomto priestore ležia buď na jednej alebo druhej strane. Je to jednoduchý príklad kde nepotrebujeme neurónové siete, ale môžeme ukázať ako môže byť neurón trénovaný na to aby rozpoznal polohu bodu.



Obr. 3.5: Rozdelenie priestoru

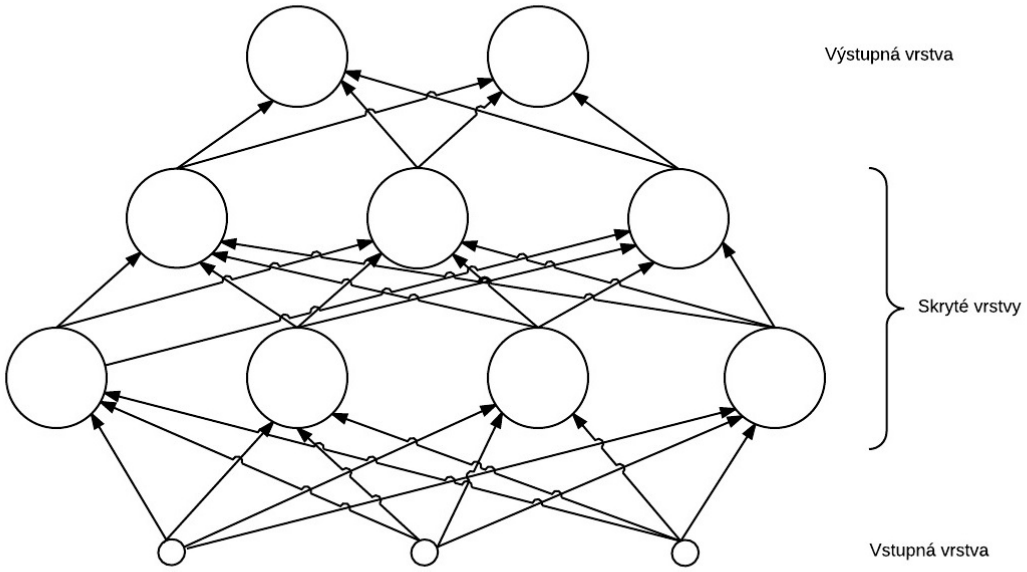
Zoberme si bod so súradnicami (0,0). Pošleme vstupné hodnoty x=0 a y=0. Suma týchto dvoch hodnôt bude nula bez ohľadu na váhy. To nám ale nevyhovuje keďže bod (0,0) musí ležať buď nad alebo pod čiarou. Aby sme sa vyhli tomuto problému, neurón potrebuje ďalší vstup ako je nakreslené na obrázku 3.6 a to vstup nazývaný bias unit - jednička. Jednička ma vždy hodnotu 1. Tým pádom nám dokáže vyjadriť vzťah tohto bodu k čiare. Pokiaľ je hodnota kladná bod (0,0) sa nachádza nad čiarou. Ak je hodnota záporná bod je pod čiarou. Jedička nám pomáha aktivovať neurón ktorý by inak neniesol žiadnu informáciu [13].



Obr. 3.6: Vstupy a výstup neurónu

3.2 Umelá neurónová sieť

Sila neurónov nie je vo viacerých vstupoch ale v počte neurónov samotných. Zoskupenia poprepájaných neurónov nazývame neurónová sieť. Počtom neurónov a štruktúrou ich vzájomných prepojení určujeme architektúru neurónovej siete. Topológiu z obrázku 3.7 možno opísať nasledovne:



Obr. 3.7: Neurónová sieť

Vstupné neuróny: sú to predovšetkým pasívne prvky, ktoré slúžia ako vstup signálu a následnému šíreniu k ďalšej vrstve neurónov?

Skryté neuróny: predstavujú pracovnú jednotku celej siete. Transformujú vstupy a výsledky predávajú ďalšej vrstve.

Výstupné neuróny: reprezentujú výstup celej neurónovej siete, ide o odozvu na vstupné signály.

Šírenie informácie cez sieť, je umožnené vďaka prepojeniam medzi jednotlivými neurónmi. Ako som už spomínal každé prepojenie má pridelenú určitú váhu. Neurónová

sieť sa časom vyvíja a mení stavy a váhy na určitých prepojeniach. Ďalej možno siete rozdeľovať do ďalších skupín ale pre nás budú v tomto prípade neurónové siete, ktoré pracujú na základe Feed-forward propagation s Backpropagation (dopredná sieť so spätným šírením chyby).

3.2.1 Dopredná sieť so spätným šírením chyby

Je to najznámejší a najrozšírenejší model neurónovej siete s učiacim sa algoritmom so spätným šírením chyby. Základ siete tvoria neuróny pospájané medzi sebou, ktoré vytvárajú vrstvy. Počet vrstiev a počet neurónov vo vrstvách je prispôsobený účelu použitia. Sieť musíme voliť dostatočne veľkú na to aby bola schopná poňať informácie, ale nie príliš veľkú, keďže s počtom neurónov stúpa doba učenia.

Dopredná sieť sa nazýva preto, lebo v sieti existujú len dopredné spojenia. Tieto spojenia prenášajú signál od vstupnej vrstvy smerom k výstupu cez neuróny. Neurón každej vrstvy posiela signál neurónu v nasledujúcej vrstve. Zoberme si topológiu siete z obrázku 3.7, kde máme sieť so štyrmi vrstvami. Vstupnú vrstvu, dve skryté vrstvy a výstupná vrstva [8].

3.2.2 Spätné šírenie chyby

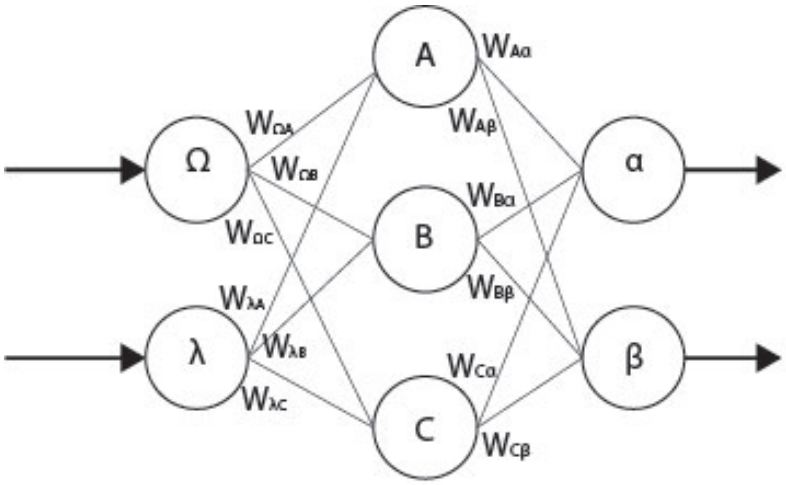
Riešenie pre optimalizáciu váh vo viacvrstvovej sieti je známa ako backpropagation (spätné šírenie chyby). Trénovanie siete taktiež zahŕňa chyby. Chyby však musia byť spätne šírené, aby sa zmenili váhy a vyvarovalo sa opätovnej chybe. Na to aby sa algoritmus učil, potrebuje tréningovú množinu [14].

Metóda je založená na základe učenia s učiteľom. Algoritmus obsahuje tri etapy: dopredné šírenie vstupného signálu, spätné šírenie chyby a aktualizácia váhových hodnôt. Počas dopredného šírenia každý neurón vo vstupnej vrstve dostane vstupný signál. Neurón vo vnútornej vrstve, vypočíta svoju aktiváciu a pošle tento signál všetkým neurónom v nasledujúcej vrstve. A tak ďalej putuje signál až na neuróny na výstupnej vrstve, ktoré tiež vypočítajú svoju aktiváciu. Táto aktivácia odpovedá skutočnému výstupu po predložené vstupného vzoru. Týmto pádom dostávame odozvu neurónovej siete na vstupný podnet vzbudením neurónov na výstupnej vrstve.

Proces stanovenia váh medzi neurónmi je riešené spätným šírením. Pri doprednom šírení nie je možné spätné šírenie, kombinácia týchto dvoch metód je algoritmus,

ktorý sa dokáže učiť vďaka obojsmernému prepojeniu vstupnej a výstupnej vrstvy. Počas adaptácie neurónových sietí s metódou spätnej propagácie sú porovnávané vypočítané výstupné aktivačné hodnoty s požadovanými výstupnými hodnotami. Táto adaptácia prebieha pre každý neurón vo výstupnej vrstve a pre každý trénovaný vzor. Výsledkom tohto porovnávania je definovaná chyba neurónovej siete [15].

Pre príklad vyjadríme vzorce pre chod backpropagation z obrázku 3.8.



Obr. 3.8: Prepojenie medzi neurónmi

1. Výpočet erorru vstpných neurónov

$$\delta_\alpha = output_\alpha(1\text{-}output_\alpha)(target_\alpha\text{-}output_\alpha) \tag{3.2}$$

$$\delta_\beta = output_\beta(1\text{-}output_\beta)(target_\beta\text{-}output_\beta) \tag{3.3}$$

2. Zmena vah výstupnej vrstvy

$$W^+_{A\alpha} = W_{A\alpha}{}^*\eta\delta_\alpha{}^*output_A \tag{3.4}$$

$$W^+_{B\alpha} = W_{B\alpha}{}^*\eta\delta_\alpha{}^*output_B \tag{3.5}$$

$$W^+_{C\alpha} = W_{C\alpha}{}^*\eta\delta_\alpha{}^*output_C \tag{3.6}$$

$$W^+_{A\beta} = W_{A\beta}{}^*\eta\delta_\beta{}^*output_A \tag{3.7}$$

$$W^+_{B\beta} = W_{B\beta}{}^*\eta\delta_\beta{}^*output_B \tag{3.8}$$

$$W^+_{C\beta} = W_{C\beta}{}^*\eta\delta_\beta{}^*output_C \tag{3.9}$$

3. Výpočet chyby výstupnej vrstvy

$$\delta_A = output(1\text{-}output_A)(\delta_\alpha W_{A\alpha}+\delta_\beta W_{A\beta}) \tag{3.10}$$

$$\delta_B = output(1\text{-}output_B)(\delta_\alpha W_{B\alpha}+\delta_\beta W_{B\beta}) \tag{3.11}$$

$$\delta_C = output(1\text{-}output_C)(\delta_\alpha W_{B\alpha}+\delta_\beta W_{C\beta}) \tag{3.12}$$

4. Zmena váh výstupnej vrstvy

$$W^+_{\alpha A} = W_{\alpha A}{}^*\eta\delta_A{}^*input_\alpha \tag{3.13}$$

$$W^+_{\alpha B} = W_{\alpha B}{}^*\eta\delta_B{}^*input_\alpha \tag{3.14}$$

$$W^+_{\alpha C} = W_{\alpha C}{}^*\eta\delta_C{}^*input_\alpha \tag{3.15}$$

$$W^+_{\Omega A} = W_{\Omega A}{}^*\eta\delta_A{}^*input_\Omega \tag{3.16}$$

$$W^+_{\Omega B} = W_{\Omega B}{}^*\eta\delta_B{}^*input_\Omega \tag{3.17}$$

$$W^+_{\Omega C} = W_{\Omega C}{}^*\eta\delta_C{}^*input_\Omega \tag{3.18}$$

Získali sme vztahy pre úpravu váh výstupnej a skrytej vrstvy. Pokiaľ neurónová sieť obasuje viac skrytých vrstiev, používajú sa rovnaké rovnice ale vždy sa dosádza do rovnice pre chybový signál výsledný chybový signál z predchádzajúcej vrstvy [16].

4 VÝPOČTY NA GRAFICKÝCH KARTÁCH

Dnešné počítače majú často podstatne vyššie výkony u jednotlivých komponentov, než by bolo pri bežnom používaní nutné. Či už procesory, sieťové karty ale aj grafické karty. Nie je tomu tak dávno keď všetky grafické karty boli iba jednovláknové čo znamená, že procesor nemohol rozložiť výpočet na viacej vlákien naraz. Odstupom času sa však začali objavovať procesory s väčším počtom jadier a grafické karty s väčším počtom vláken vďaka tomu sa výpočty dokázali rozložiť do viacerých vláken. Pri rozložení výpočtov do vláken výrazne zmenšíme čas na vykonanie programu.

Graphics Procesing Unit (GPU) sa stali neoddeliteľnou súčasťou dnešných bežných počítačových systémov. Za posedných pár rokov došlo k výraznému nárastu výkonnosti a schopností GPU. Moderné GPU nie sú len výkonné ale aj vysoko paralelne programovateľné procesory, ktoré výrazne predstihujú Central Procesing Unit (CPU). V paralelnom počítaní je budúcnosť, ktorá bude nútiť rozvoj mikroprocesorov do pridávania jadier a zvyšovanie výkonu.

Pre účinnosť GPU spracováva mnoho prvkov paralelne za použitia jedného programu. Každý prvok nezávisle do ostatných prvkov nekomunikuje s ostatnými. Všetky GPU procesy musia byť štruktúrované ako mnoho paralelných prvkov z, ktorých je každý spracovaný paralele pomocou programu. Elementy môžu čítať dáta zo zdieľanej globálnej pamäte a s najnovšími GPU aj písať do rôznych miest zdieľanej globálnej pamäte. Tento programovací model je vhodný pre priamočiare programy pretože mnoho prvkov môže byť spracovaný naraz v rovnakom kóde. Kód písaný týmto spôsobom je počítaný ako jedna inštrukcia [9].

4.1 Technolóhia CUDA

Táto technológia je spustiteľná na skoro všetkých grafických kartách značky Nvidia bez ohľadu na operačný systém. Základným prvkom programovania CUDA je kernel. Kernel je vlastne určitá časť kódu, ktorá spúšťa elementy paralelne nad elementy dátového typu. Vlákna sa zoskupujú do blokov, ktoré následne do mriežky. Mriežky a bloky môžu byť viacrozmerné. Vlastnosti ako veľkosť bloku a počet blokov je na programátorovi. Kernel je spúšťaný asynchrónne. To má za následok, že hostiteľ nečaká na dokončenie výpočtu ale ďalej pokračuje vo vykonávaní kódu, ktorý prebieha v sekvenčnom charaktere. Systém, ktorý vznikne spojením GPU a CPU dokáže veľmi efektívne vypočítať paralelné kombinácie. Ak chceme spracovať objemovo náročné obrázky využijeme identifikátory trojrozmerného typu. Pri sériovom

alebo paralelnom spúšťaní musia byť medzi sebou bloky nezávislé. Vďaka tejto nezávislosti možno ľubovoľne rozmiestniť jednotlivé bloky medzi multiprocesory. Toto rozloženie umožňuje programátorom vývoj aplikácií, ľahkú triediteľnosť [7] [12].

4.2 OpenCL

Je všestranný štandard, ktorý možno použiť na hocakú grafickú kartu bez rozdielu na výrobcu. Ide o rozšírenie existujúcich jazykov. Umožňuje špecifikovať časť kódu, ktorá je vykonaná niekoľkokrát nezávisle od seba. Pre príklad potrebujeme vypočítať sin(x) o veľkosti jedného milióna čísel. OpenCL rozpozná zariadenie, ktoré môže daný príklad vypočítať a vytvorí štatistiky každého zariadenia. Môžete si vybrať najlepšie zariadenie alebo niekoľko zariadení a poslať dáta buď jednému alebo viacerým zariadeniam. Normálne by ste zobrali slučku milióna čísel ale namiesto toho stačí povedať hodnoty v poli, ktoré chcete vedieť. Po dokončení, jednoducho vezmete dáta zo zariadenia. Vzhľadom, že výpočtové zariadenia dokážu lepšie pracovať na paralelnej úrovni, OpenCL lepšie opisuje nezávislé funkcie. Celková doba realizácie je oveľa nižšia než u konvenčných metód [18].

5 NAVRHNUTÉ RIEŠENIE A ZHODNOTENIE VÝSLEDKOV

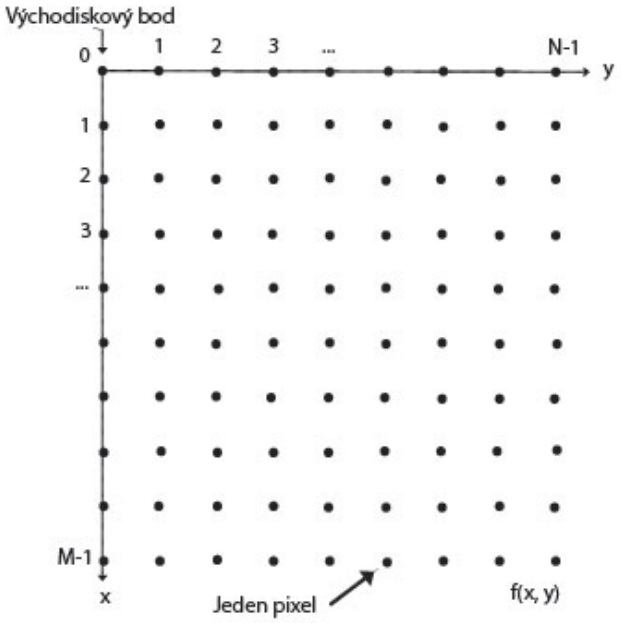
5.1 Spracovanie obrazu na GPU

Hlavnou častou dnešných GPU je jednotka rendering, ktorá sa zaoberá vizualizáciou reálneho obrazu na základe počítačového modelu so zohľadnením jej okolia a scény. Program pomocou matematických algoritmov vytvorí bitmapový obraz, ktorý napodobňuje reálnu scenériu. Na takúto realizáciu GPU potrebuje nie len dostatočne širokú zbernicu, ale aj podporu rýchlej pamäte. V minulosti počítačovej grafike zabralo vykreslenie 3D objektov niekoľko minút. Moderné grafické karty dokážu toto vykreslenie zhruba 70 krát za jednu sekundu.

V našom prípade využijeme paralelné výpočty GPU. Obraz s rozlíšením 256*256 pixlov je rozdelený na paralelné bloky, ktoré sú počítané na vlákne paralelne nezávisle od ďalších vláken. Program spracováva informácie o jednom pixli. Keďže moderné GPU dokážu spracovať veľké množstvo výpočtov paralelne a v jednom kóde, využil som tejto možnosti. Program je prispôsobený spôsobu spracovania dát na GPU kartách.

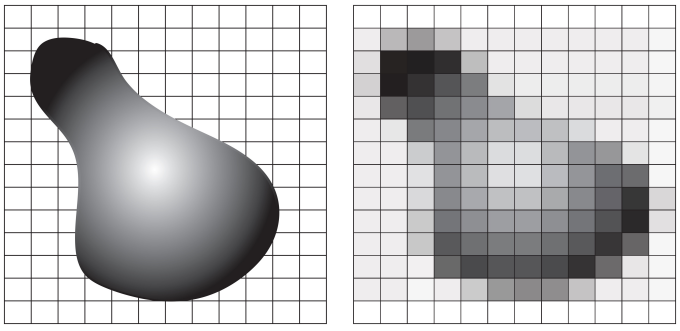
Obraz na monitora je vytvorený zložitým postupom. Aby bol objekt zobrazený na obrazovke musí byť spracovaný procesorom – CPU, ktorý vyšle príkaz na grafický rozhranie kde je sa nachádza spojnica medzi hardvérom grafickej karty a operačným systémom. Ovládač odošle digitálne údaje v novom formáte na renderovanie grafickej karty. Odtiaľ sú dáta presunuté do zbernice PCIE, inak nazývanú AGP ako vyrovnávacia pamäť systému. Nachádza sa buď priamo na karte alebo v systémovej pamäti.

Digitálne dáta boli získané fotoaparátom, a bolo nutné ich obraz diskreditovať aby ho bolo možné uložiť. Zachytený obraz je kontinuálny *f(x,y)*, pričom dochádza k prevedeniu na diskrétny tvar. Obraz je prevedený na pixmapu o rozmeroch *M*, *N*. Každý jeden bod v pixmape predstavuje jeden pixel.



Obr. 5.1: Pixmapa

Vzorkovanie prebieha výberom hodnoty pixlu v určitom okamihu *f(x,y)*. Vybrané hodnoty musia spolu korešpondovať ako je to zobrazené na obrázku 5.2.



Obr. 5.2: Na ľavo originálny obraz, v pravo vzorkovaný obraz

Takýmto vzorkovaní dochádza k nepresnostiam a k možným stratám informácií. Na obrázku možno vidieť nerovnomerné hrany.

5.2 Volenie váh

Volenie váh je veľmi dôležité preto aby sa systém naučil správne rozpoznávať obsah obrazu. Na volenie váh existuje niekoľko algoritmov. Váhy v mojom výsledku nie sú upravované pomocou spätného šírenia chyby.

Ide o náhodné váhy, ktoré sa násobia so vstupnou informáciou. Tým pádom do neurónu putuje informácia zo vstupu vynásobená náhodou váhou. Neurón tieto hodnoty vypočíta a posiela ich ďalej. Je to jednoduchý základ doprednej siete. Program zatiaľ dokáže generovať náhodné čísla a neurón dokáže s týmito hodnotami pracovať a poslať na výstup.

5.3 Výpočet neurónovej siete s použitím GPU

Ako som už spomínal GPU využívajú paralelné výpočty. V tomto prípade sa musíme zamyslieť čo chceme aby program spočítal. Inicializácia príkazu začína na CPU. Následne je príkaz odoslaný na GPU kartu kde prebehne výpočet. Výsledok je odoslaný naspäť na CPU. Výpočty na GPU kartách v jazyku JAVA pomáha OpenCL a Kernel. Pred programovaním je nutné sa zamyslieť ako grafické karta pracuje a spracováva dáta. Správne navrhnutý program dokáže GPU vypočítať veľmi rýchlo.

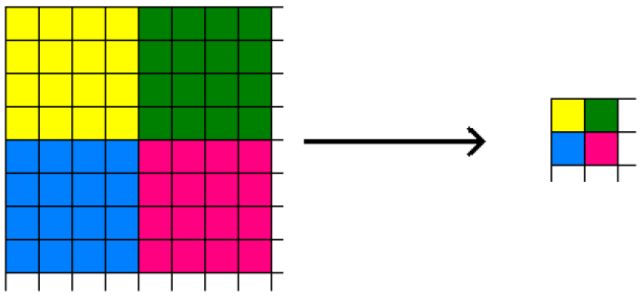
Volenie váh prebieha na CPU. V ďalšom kroku vyvoláme kernel funkciu, ktorá výpočty odošle na grafickú kartu. Tu prebehnú paralelné výpočty vlastnosti pixlov. Sú to hodnoty R, G, B a alfa. Na obrázku je naprogramované zobrazenie hodnôt R, G, B a alfa.

5.4 Riešenie

5.4.1 Návrh siete

Problém extrakcie informácií z obrazu som vyriešili doprednou sieťou s backpropagaion. Jednoduchá sieť, ktorá posiela informácie od vstupu smerom na výstup cez neurónové jednotky. Podarilo sa mi naprogramovať sieť s dvoma skrytými vrstvami o počte 50 a 13 neurónov. Vstupné jednotky odpovedajú upravenému rozlíšeniu obrazu. Výstupné neuróny odpovedajú detekovaným objektom. Na začiatku práce som pracoval s jedným pixlom. Neskôr pri praktickom experimente som zistil, že bude lepšie obraz rozdeliť do väčších rovnomerných blokov. Program som upravil aby pracoval na základe rozdelenia obrazu na rovnako veľké bloky. Tieto štvorce obsahovali rovnaký počet pixlov. Pri väčšom rozlíšení je jeden pixel skoro rovnaký ako susedný,

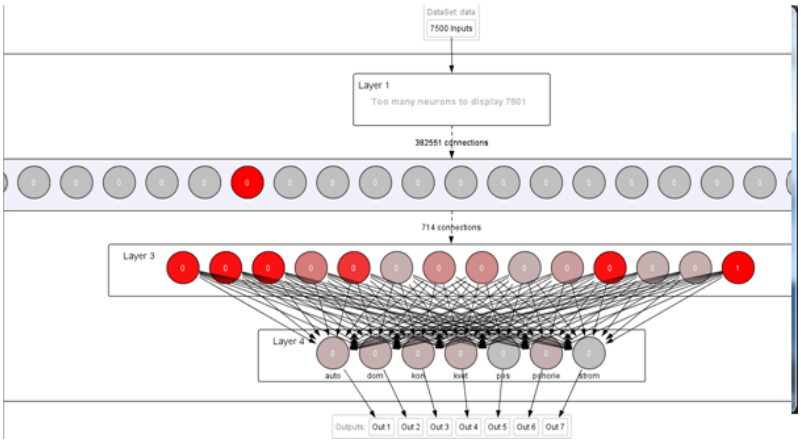
zobral som teda štvorce o rozmere 3x3 pixle. Priestor sa tým rozdelil na štvorce, ktoré by mali mať viac informácií než jeden pixel. V takomto štvorci som dostal informáciu o 9 pixloch. V tomto prípade som dostal relevantnejšie hodnoty o obraze. Túto variantu som vymenil za finálne riešenie, ktoré vstupný obrázok automaticky zmenší, upraví na rozmer 50*50 pixlov. Táto varianta je v podstate upravenou variantou rozdelenia obrázku na bloky 3x3 pixle. Zoberme si obrázok o rozmeroch 200x200 pixlov. Pri vstupe do neurónovej siete sa automaticky transformuje na obrázok 50x50 pixlov čiže blok z obrázku o rozmeroch 4x4 pixle sa pretransformuje do jedného pixlu ako je zobrazené na obrázku 5.3.



Obr. 5.3: Rozdelenie obrázku

Vstupná hodnota teda zodpovedá 50(výška)*50(šírka)*3(RGB zložka)=7500 neurónom. Myslím si, že táto varianta je lepšia z hľadiska možnosti vloženia rôznych rozlíšení obrázku. Systém si obrázky upraví na rovnaký, fixný rozmer a stále počíta s rovnakými hodnotami i keď je obrázok väčší alebo menší. Tým pádom je systémový výpočet konštantne rýchly a predchádza tým k nadmerným dátam, ktoré nebývajú väčšinou využité.

Na vstupe neurónovej siete je 7500 nerurónov, ktoré su napojené na prvú skrytú vrstvu. Vstupná vrstva odpovedá už spomínanému riešeniu úpravy rozmerov obrázku. V takomto riešení siete je každý vstupný neurón využitý a ďalej naviazaný na neurón v skrytej vrstve. Náčrt siete je možné vidieť na obrázku 5.4. Každý jeden pixel z upraveného obrázkového formátu reprezentuje jeden neurón. Hodnota z jedného pixlu je násobená RGB zložkou. Následne je táto hodnota vynásobená váhou. Neurón tieto hodnoty spočíta a posunie ďalším neurónom.



Obr. 5.4: Vnútro neurónovej siete

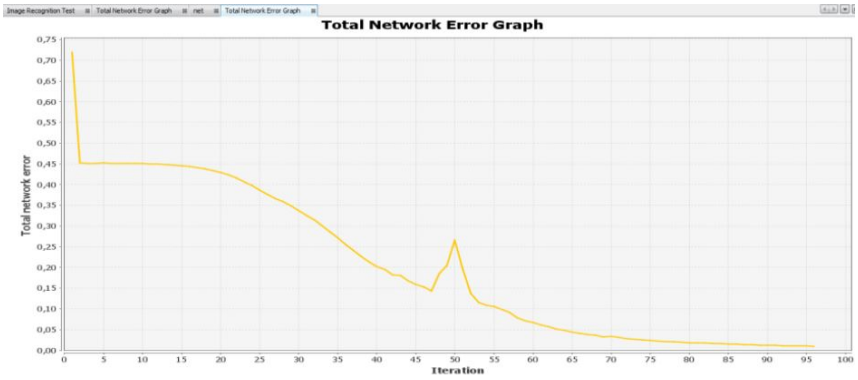
V skrytej vrstve sa nachádzajú dve skryté vrstvy. Prvá má 50 neurónov a druhá 13. Tieto hodnoty boli empiricky získané na základe konvergencie chybovosti k 0 pri vytváraní siete a presnosťou detekcie. Každý neurón zo vstupnej vrstvy je prepojený s neurónom na prvej skrytej vrstve ako je zobrazené na obrázku 5.4. Na ceste od vstupnej vrstvy ku skrytej je je hodnota vynásobená ďalšou váhou a znova spočítaná ale tento krát už na neuróne skrytej vrstvy. Základom doprednej siete je šírenie smerom vpred. Teda po vypočítaní hodnoty na prvej skrytej vrstve sa získaná hodnota posunie na druhú, poslednú skrytú vrstvu. Taktiež je každý neurón prvej skrytej vrstvy spojený s poslednou skrytou vrstvou. Tu nastáva rovnaký výpočet ako predtým a podstate finálna detekcia, kde program určí daný objekt nachádzajúci sa na obrázku.

Na výstupnej vrstve je 7 neurónov. Číslo 7 zodpovedá počtu detekovaných objektov. Pre moju prácu som si zvolil tieto - auto, dom (budova), strom, kvet, pohorie, kôň a pes. Výsledkom detekovania je teda výstupná vrstva, ktorá bola "zopnutá"(našla sa zhoda) detekciou predošlých vrstiev. Pokiaľ sa zhoda nenájde, program vypíše, že na obrázku žiaden objekt nedetekoval.

5.4.2 Rozpoznávanie

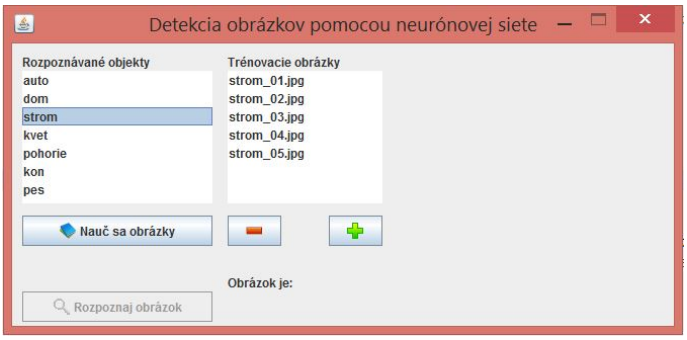
Princíp detekcie obrázkov pracuje s pomocou neurónovej siete. Princíp rozpoznávania spočíva v tom, že sú dopredu definované objekty, ktoré sa majú rozpoznať (už pri tvorbe neurónovej siete). Konkrétne: auto, dom, strom, kvet, pohorie, kôň a pes. V aplikácii je potom možné definovať sadu obrázkov pre každý objekt, z ktorých sa vytvoria trénovacie dáta pre neurónovú sieť. Predtým ako je možné rozpoznať nejaký objekt, je treba neurónovú sieť naučiť všetky zadefinované trénovacie objekty.

Neurónovú sieť som vytvoril pomocou programu NeurophStudio, ktorý je voľne prístupný. V tomto programe je možné vytvoriť neurónové siete v podstate na čokoľvek. Do programu som zadal parametre neurónovej siete ako počet vrstiev, nerónov alebo rozlíšenie vstupného obrázku. V programe pred exportom som ešte sieť testoval aby nepresahovala hranicu chybovosti 0,01 ako je ukázané na obrázku 5.5. Následne takto vytvorenú sieť som exportoval do programu.



Obr. 5.5: Vnútro neurónovej siete

Samostatný program funguje v grafickom prevedení, ktoré je zobrazené na obrázku 5.6. Ak klikneme na tlačidlo + vyberieme obrázky, ktoré sa má program naučiť pre daný objekt. Naopak tlačítkom - odoberáme obrázky z pamäte. Pre každý objekt som si pripravil sadu testovacích obrázkov. Tieto obrázky som následne program učil. Zo začiatku som skúšal rôzne variácie a počty naučených obrázkov. Následné testovanie obrázkov záležalo od objektu a množstve naučených obrázkov. Aby som dosiahol primeraných hodnôt v rámci rozpoznávania, trénovacie obrázky som upravil na rovnaké rozlíšenie a to 400x300 pixlov. Toto upravenie je ideálne keďže vstupný obrázok sa automaticky prepočíta na rozmer 50x50. Program som postupne po dávkach učil. Začal som s 10 potom 50 a nakoniec 100 obrázkami pre každý objekt. Použil som teda 100 trénovacích obrázkov pre každý objekt. Po natrénovaní nastaveného počtu obrázkov som testoval rozpoznanie obrázku.



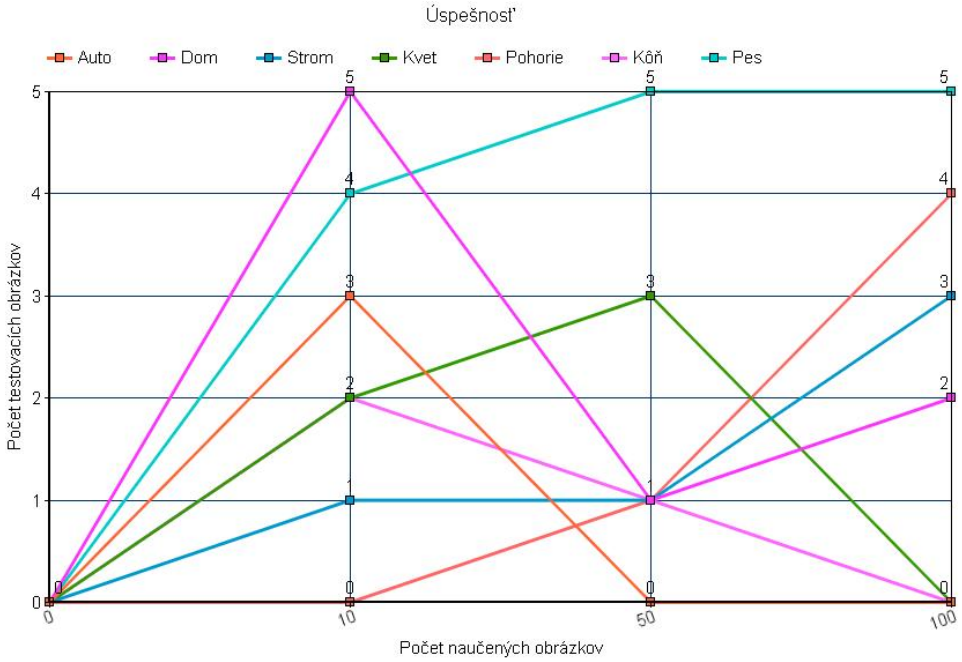
Obr. 5.6: Vnútro neurónovej siete

5.4.3 Zhodnotenie

Program som testoval obrázkami, ktoré sa nezhodovali s obrázkami z trénovacej sady. Testovací obrázok som vždy rozpoznával po jednom. Úspešnosť rozpoznania objektu na obrázku je vykreslené na obrázku 5.8. Pre prehľadnejšie čítanie grafu som hodnoty uviedol aj v tabuľke 5.1. V tabuľke a v grafe sú reálne namerané hodnoty, ktoré som dostal keď som program testoval. Používal som už spomenutú upravenú metódu rozdelenia do blokov a z obrázku som využil farebné zložky. Samostatné učenie závisí od množstva trénovacích obrázkov a množstva objektov. Priemerné trvanie učenia pre 10 obrázkov bolo 5 sek, 50 obrázkov bolo 14 sek a 100 obrázkov boli 4 min. Samotné rozpoznanie trvalo v priemere 5 sek. Tieto hodnoty mohli byť ovplyvnené vyťažením procesora a pamäte.

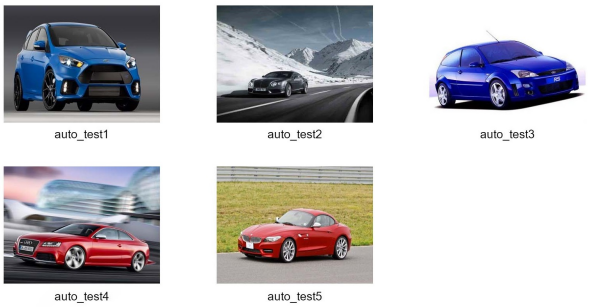
Počet obrázkov	10	50	100
Auto	3	0	0
Dom	5	1	2
Strom	1	1	3
Kvet	2	3	0
Pohorie	0	1	4
Kôň	2	1	0
Pes	4	5	5

Tab. 5.1: Tabuľka úspešnosti rozpoznania objektov

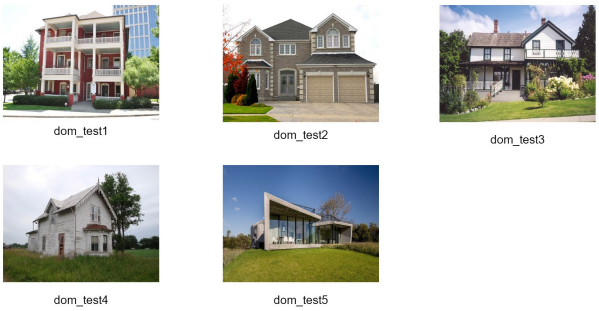


Obr. 5.7: Graf úspešnosti rozpoznania objektov

Metóda, ktorú som si zvolil je jedna z priemerných metód na rozpoznávanie objektu na obrázku. Keďže pracuje s farebnou zložkou obrázku, dostáva teda informácie o RGB hodnotách z pixlu. Každý pixel vstupného obrázku sa rozloží na RGB zložky. Takáto zložka následne prechádza neurónovou sieťou a vytvára na základe svojej hodnoty a pozície svoju "cestu". Zo vstupu na výstup sa teda dostane na základe prechodu cez neuróny. Dôležitou vlastnosťou každého neurónu je určitá váha na základe ktorej sa hodnota zo vstupu vynásobí, konkrétne zložka RGB z pixlu, a ďalej rozhodne akou cestou "pôjde"smerom na výstup. Tieto váhy neuróny získavajú a menia počas procesu učenia. Tým pádom môže nastať situácia, že jeden obrázok pohoria pri učení vytvori krásne spojenie na výstup objektu "pohorie", ale keďže má podobnú farebnú štruktúru (z pohľadu zložiek RGB pixlov) ako dom, tak sa vyhodnotí ako dom. Pri takom veľkom množstve učiacich objektov to nemusí byť vôbec výnimočné. Detekovanie objektu som testoval na piatich obrázkoch, ktoré sú zobrazené na obrázkoch 5.8, 5.9, 5.10, 5.11, 5.12, 5.13 a 5.14.

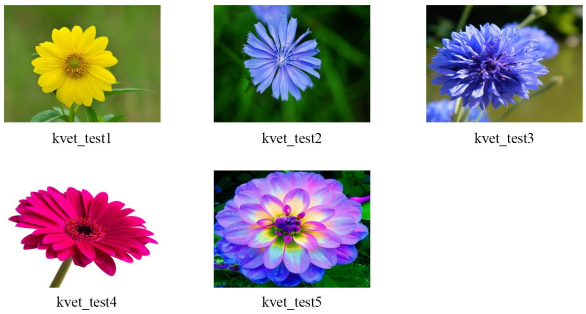


Obr. 5.8: Testovacie obrázky pre auto

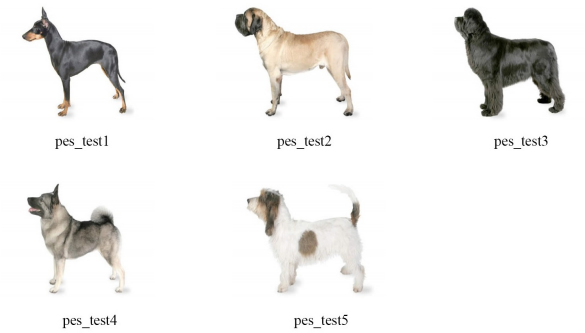


Obr. 5.9: Testovacie obrázky pre dom

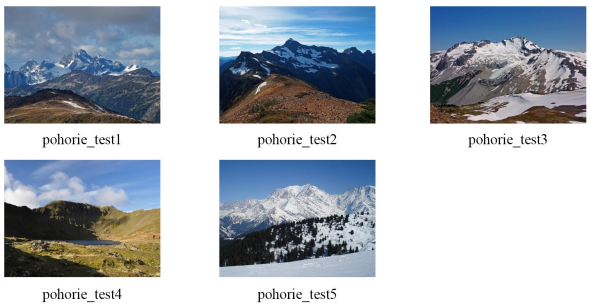
Obr. 5.10: Testovacie obrázky pre koňa



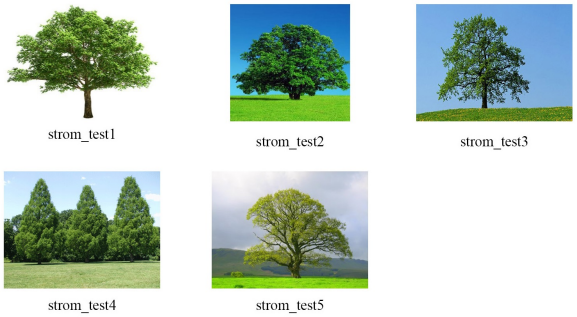
Obr. 5.11: Testovacie obrázky pre kvety



Obr. 5.12: Testovacie obrázky pre psa



Obr. 5.13: Testovacie obrázky pre pohorie



Obr. 5.14: Testovacie obrázky pre stromy

Ak sa niektoré objekty nesprávne detekovali alebo sa nedetekovali vôbec, príčin môže byť viacej. Každopádne metóda získavanie farieb z obrázku ako samostatná metóda nie je príliš presná. Pre spresnenie som použil metódu zmenšenia vstupného obrázku na 50x50 pixlov. Ide o variantu rozdelenia obrázkov do bloku. Možnosť neúspechu pri takto nastavenom programe je taktiež otázkou správnych správneho nakódovania mechanizmu pre extrakciu príznakov. V mojej práci som použil už dopredu nastavené neurónové siete kde prebiehali výpočty a samotné extrahovanie dát z obrázku. Následne boli výsledky presunuté do prostredia JAVA kde boli vypísané zhody.

6 ZÁVER

V tejto práci som sa venoval klasifikácií obrazu pomocou umelej inteligencie. Na začiatok opisujem metódy získavania príznakov z obrazu. Tieto metódy sú pre klasifikáciu obrazu veľmi dôležité, keďže ide o identifikáciu obsahu obrazu. Hodnoty, ktoré sa získajú z obrázku ďalej spracovávané umelou inteligenciou vo forme neurónových sietí.

Prácu som programoval v prostredí JAVA kde som implementoval naprogramoval jednoduchú doprednú neurónovú sieť a spätné šírenie chyby. Táto sieť posúva informácie smerom dopredu a spätne upravuje váhové koeficienty. Program pracuje so sadou trénovacích a testovacích obrázkov.

Vo výsledku môj spôsob získania informácií z obrázku funguje. Pri vložení testovacieho obrázku program na výstupe buď rozpozná alebo nerozpozná objekt. Počas merania som prišiel na to, že záleží akých obrázkoch program trénujem. Výsledky sa odzrkadlia na detekovaní. Program pracuje s farebnou zložkou, to znamená, že si vyťahuje z obrázku informácie o RGB zložkách z pixlu. Preto sa niektoré objekty chybne detekujú lebo maju rovnaké sfarbené rozpoloženie. To znamená keď si pozrieme výsledky psa a koňa z tabuľky 5.1, vidíme, že kôň sa postupne detekuje ťahšie ako pes. Kôň sa nachádza na farebnejšom pozadí zatiaľ čo pes na bielom. Preto sa pes detekuje ľahšie ako kôň, ktorý má viacej farebnej zložky.

Program by pracoval presnejšie ak by boli všetky obrázky na bielom pozadí. Taktiež záleží na prvotnom výbere a konštrukcií neurónovej siete, ktorú v programe používam. Od správneho volenia hodnôt neurónov, skrytých vrstiev a počtu testovacích obrázkov závisí výsledné detekovanie. Dôležitou súčasťou siete je spätné šírenie chyby vďaka ktorej sa sieť učí prideľovať správne váhy.

LITERATÚRA

[1] *A CBIR system based on class signature of the image's color and texture features* [online]. 2014, poslední aktualizace 9. 8. 2009 [cit. 10. 11. 2014]. Dostupné z URL: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5069365>.

[2] Mehrotra, K.; Mohan, C. K.; Ranka, S Articial Neural Networks. In *MessachusettsInstitute of Technology*, 1997. ISBN 0-262-13328-8.

[3] *Comparison of different CBIR techniques* [online]. 2014, poslední aktualizace 3. 11. 2011 [cit. 10. 11. 2014]. Dostupné z URL: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5941923&tag=1>.

[4] *Content-based image retrieval using color and texture fused features* [online]. 2014, poslední aktualizace 4. 8. 2011 [cit. 10. 11. 2014]. Dostupné z URL: <http://www.sciencedirect.com.ezproxy.lib.vutbr.cz/science/article/pii/S0895717710005352>.

[5] *Content based image retrieval system* [online]. 2014, poslední aktualizace 9. 5. 2012 [cit. 10. 11. 2014]. Dostupné z URL: <http://www.ijeit.com/vol%201/Issue%205/IJEIT1412201205_19.pdf>.

[6] *CUDA application design and development* [online]. 2014, posledná aktualizácia 1. 2. 2011 [cit. 10. 11. 2014]. Dostupné z URL: <http://streamcomputing.eu/knowledge/what-is/opencl/>.

[7] *CUDA programing* [online]. 2014, posledná aktualizácia 1. 2. 2013 [cit. 10. 11. 2014]. Dostupné z URL: <http://books.google.cz/books?hl=sk&lr=&id=g3EzsZn4poUC&oi=fnd&pg=PP2&dq=+Shane.+CUDA+programming:+a+developer%27s+guide+to+parallel+computing+with+&ots=-1wDjup6QU&sig=xfS0tnUcCGP8XYKTMiqs8eMV6aw&redir_esc=y#v=onepage&q=Shane.%20CUDA%20programming%3A%20a%20developer%27s%20guide%20to%20parallel%20computing%20with&f=false>.

[8] *Fundamentals of the new artificial intelligence* [online]. 2014, posledná aktualizácia 19. 02. 2008 [cit. 10. 11. 2014]. Dostupné z URL: <http://www.google.cz/books?hl=sk&lr=&id=lei-Zt8UGSQC&oi=fnd&pg=PR5&dq=+Munakata,+T.:+Fundamentals+of+the+new+arti%0Ccial+intelligence+&ots=q-FWWOySWz&sig=aegN1Qr7gX71RfA40n900B-ayIA&redir_esc=y#v=onepage&q=Munakata%2C%20T.%3A%20Fundamentals%20of%20the%20new%20arti%0Ccial%20intelligence&f=false>.

[9] *GPU computing* [online]. 2014, posledná aktualizácia 19. 02. 2000 [cit. 10. 11. 2014]. Dostupné z URL: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4490127>.

[10] *Metódy umelej inteligenci* [online]. 2014, poslední aktualizace 4. 5. 2012 [cit. 10. 11. 2014]. Dostupné z URL: <http://natureofcode.com/book/chapter-10-neural-networks>.

[11] *Metódy umelej inteligencie* [online]. 2014, posledná aktualizácia 19. 02. 2009 [cit. 10. 11. 2014]. Dostupné z URL: <http://www.zivotsepilepsiou.sk/stranka-1-ludsky-mozog.html>.

[12] *Programming massively parallel procesors* [online]. 2014, posledná aktualizácia 1. 2. 2010 [cit. 10. 11. 2014]. Dostupné z URL: <http://books.google.cz/books?hl=sk&lr=&id=E0Uaag8qicUC&oi=fnd&pg=PT15&dq=Wen+-+mei+HWU.+Programming+massively&ots=ID3qgaYNmA&sig=04hPRsQ2pO-uQvezPm9X1mSkFt4&redir_esc=y#v=onepage&q=Wen%20-%20mei%20HWU.%20Programming%20massively&f=false>.

[13] *Rozpoznávanie obrazu* [online]. 2014, posledná aktualizácia 12. 5. 2009 [cit. 10. 11. 2014]. Dostupné z URL: <http://www.sccg.sk/~ftacnik/RO3.pdf>.

[14] Šímaa, J.; Neruda, R Teoretické otázky neuronových sítí. In *Matfyzpress*, 1997. ISBN80-85863-18-9.

[15] *The back propagation* [online]. 2014, posledná aktualizácia 19. 02. 2009 [cit. 10. 11. 2014]. Dostupné z URL: <http://www4.rgu.ac.uk/files/chapter3%20-%20bp.pdf>.

[16] *The back propagation algorithm* [online]. 2014, posledná aktualizácia 19. 02. 2000 [cit. 10. 11. 2014]. Dostupné z URL: <http://page.mi.fu-berlin.de/rojas/neural/chapter/K7.pdf>.

[17] *Umelá inteligencia* [online]. 2014, posledná aktualizácia 10. 5. 2010 [cit. 10. 11. 2014]. Dostupné z URL: <http://www.toptal.com/machine-learning/an-introduction-to-deep-learning-from-perceptrons-to-deep-networks>.

[18] *What is OpenCL* [online]. 2014, posledná aktualizácia 1. 2. 2013 [cit. 10. 11. 2014]. Dostupné z URL: <http://streamcomputing.eu/knowledge/what-is/opencl/>.

ZOZNAM SYMBOLOV, VELIČÍN A SKRATIEK

AGP Accelerated Graphics Port

CBIR Content Based Image Retrieval

CPU Central Procesing Unit

GPU Graphic Procesing Unit

HSV Hue, Saturation, Value

PCIE Peripheral Component Interconnect Express

RGB Red, Green, Blue

TBIR Text Based Image Retrieval

ZOZNAM PRÍLOH

A OBSAH PŘILOŽENÉHO CD

V priloženom CD sa nachádza program pre rozpoznanie objektov na obrázku. Program pracuje v prostredi JAVA. V súbore Eclipse_workspace nájdeme súbory .metadata, .recommenders a NeuronovaSiet. Neurónová sieť ktorá bola vytrorená pomocou programu NeurophStudio sa nachádza v priečinku NeuronovaSiet>NeuralNetworks >net.nnet. Pre spustenie programu je nutné zvoliť priečinok Eclipse_workspace ako workspace. Samotný program sa spustí v prostedí Eclipse tlačidlom F5 alebo zelenou šipkou, ktorá sa nachádza na vrchnej lište. Eclipse a NeurophStudio nie sú súčastou CD.