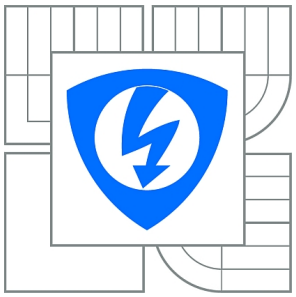
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

KONVOLUČNÍ NEURONOVÁ SÍŤ PRO ZPRACOVÁNÍ OBRAZU

CONVOLUTIONAL NEURAL NETWORK FOR IMAGE PROCESSING

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

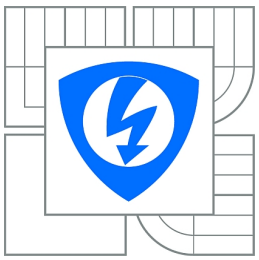
AUTOR PRÁCE — Bc. MÁRIA KRAJČOVIČOVÁ

AUTHOR

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

SUPERVISOR

BRNO 2015



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Studentka: Bc. Mária Krajčovičová ***ID:*** 123786
Ročník: 2 ***Akademický rok:*** 2014/2015

NÁZEV TÉMATU:

Konvoluční neuronová síť pro zpracování obrazu

POKYNY PRO VYPRACOVÁNÍ:

Nastudujte teorii konvolučních neuronových sítí a zpracujte rešerži této problematiky z posledních let. Vyberte vhodnou metodu a implementujte ji v prostředí jazyka JAVA. Na vhodném příkladu ověřte funkčnost a statistiky zhodnoťte. Slovně zhodnoťte dosažené výsledky.

DOPORUČENÁ LITERATURA:

[1] Simard, Patrice Y., Dave Steinkraus, and John C. Platt. "Best practices for convolutional neural networks applied to visual document analysis." 2013 12th International Conference on Document Analysis and Recognition. Vol. 2. IEEE Computer Society, 2003.
[2] Lo, Shih-Chung B., et al. "Artificial convolution neural network for medical image pattern recognition." Neural Networks 8.7 (1995): 1201-1214.

Termín zadání: 9.2.2015 ***Termín odevzdání:*** 26.5.2015

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

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

UPOZORNĚNÍ:

ABSTRAKT

Cílem diplomové práce bylo nastudování problematiky konvolučních neurónových sítí v posledních letech. Dále se práce zabíva navržením vhodných modelů konvolučních neurónových sítí a jejich následní implementaci v prostředí jazyka Java. Výsledkem práce je porování a zhodnocení dosažených výsledků získaných prostřednictvým implementované aplikace.

KLÍČOVÁ SLOVA

Neuron, neuronové síte, konvoluční neuronová síť, perceptron, hopefieldova síť, autoenkoder, omezené bolzmanové stroje

ABSTRACT

Goal of this Diploma thesis was Convolutional neural network investigation in last years. Diploma thesis also contains information about designing of appropriate Convolutional neural network models and implementation of these models in Java programming language. Result of the thesis are comparison and evaluation of results which were reached from implemented application.

KEYWORDS

Neuron, neural network, convolutional neural network, perceptron, hopefield network, autoencoder, restricted bolzman machines

KRAJČOVIČOVÁ, Mária *Konvoluční neuronová síť pro zpracování obrazu*: diplomová práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2015. 81 s. Vedoucí práce byl doc. Ing. Radim Burget, Ph.D

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Konvoluční neuronová síť pro zpracování obrazu“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

(podpis autora)

PODĚKOVÁNÍ

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

Brno ....

(podpis autora)

SIX research centre
sensor, information and communication systems

Faculty of Electrical Engineering
and Communication

Brno University of Technology
Technicka 12, CZ-61600 Brno, Czechia

http://www.six.feec.vutbr.cz

Výzkum popsaný v této diplomové práci byl realizovaný v laboratořích podpořených projektem Centrum senzorických, informačních a komunikačních systémů (SIX); registrační číslo CZ.1.05/2.1.00/03.0072, operačního programu Výzkum a vývoj pro inovace.

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

2007–13
OP Výzkum a vývoj pro inovace

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

Identifikácia objektov z obrázkov je v dnešnej dobe veľmi rozšírená, či už sa jedná o detekciu a rozpoznanie ľudskej tváre, identifikáciu ručne písaných znakov alebo iných objektov priamo z obrázku. Pre realizáciu týchto identifikácií sa v praxi veľmi častou používa implementácia pomocou rôznych typov neurónových sietí, ktoré sú vhodné najmä vďaka ich architektúre, schopnosti abstrakcie a učenia sa.

Na začiatku tejto práce je čitateľ oboznámený zo základnou problematikou neurónov a neurónových sietí, ich architektúry a princípu fungovania. Následne sa práca zaoberá rôznymi modelmi a typmi neurónových sietí. V tejto časti sú popísané vlastnosti jednotlivých modelov ako ich architektúra, princíp fungovania, prípadne ich použitie.

V ďalšej časti teoretickej časť práce pokračuje popisom *konvolučných neurónových sietí*, ktoré sú v súčastnosti jedným z najvhodnejších modelov pre identifikáciu objektov z obrázkov, pretože pred vstupom obrázka do siete nie sú nutné žiadne predchádzajúce úpravy obrázku. V tejto časti práce sú taktiež popísané rôzne príklady použitia tohto typu nuerónovej siete.

Praktická časť práce sa venuje analýze návrhu a implementácií konvolučných neurónových sietí. V časti návrhu sú v práci navrhnuté štyri modely konvolučných sietí, ktoré sa líšia veľkosťou a zložitosťou štruktúry. Cieľom tohto návrhu je implementovať tieto konvolučné neurónové siete a porovnať úspešnosť učenia a časovú náročnosť učenia u jednotlivých modelov konvolučných neurónových sietí.

Pred samotnou implementáciu sa praktická časť venuje porovnaniu rôznych frameworkov, ktoré slúžia k implementácií neurónových sietí v prostredí jazyka Java. V tejto časti je taktiež spomenutá možnosť implementácie neurónových sietí pomocou grafických jednotiek a možnosti realizácie v prostredí jazyka Java.

Po analýze jednotlivých imlementačných možností je v práci popísaná samotná implementácia modelov konvolučných neurónových sietí pomocou frameworku, ktorý bol vybraný z predchádzajúcej časti. Táto časť taktiež obsahuje popis implementácie pomocou grafických jednotiek pri výpočte jednotlivých vrstiev konvolučných neurónových sieťach

V závere práce sú zhrnuté dosiahnuté a namerané výsledky pre úspešnosť a časovú náročnosť učenia u jednotlivých konvolučných neurónových sieťach. Tieto výsledky sú následne porovnané medzi jednotlivými konvolučnými neurónovými sieťami a na základe toho je v práci zhodnotené, ktorý model konvolučnej neurónovej siete je najvhodnejšie pre prektické použitie siete. Táto časť taktiež obsahuje porovnania dĺžky výpočtov jednotlivých vrstiev konvolčných neurónových sietí, ktoré boli získané spúšťaním týchto výpočtov na procesorových a grafických jednotkách.

1 NEURÓN A NEURÓNOVÉ SIETE

Teória neurónových sietí vychádza z neurofyzikálnych poznatkov a snaží sa vysvetliť správanie sa na princípe spracovania informácií v nervových bunkách. Niekedy sa umelé neurónové siete označujú aj ako modely mozgu bez mysle (angl. *brain without mind*), pretože sa snažia pochopiť nervový systém, ale nezaoberajú sa psychikou. Za počiatok vzniku oboru neurónových sietí sa považuje práca Warrena McCullocha a Waltera Pittsa z roku 1943, ktorí vytvorili jednoduchý matematický model neurónu.

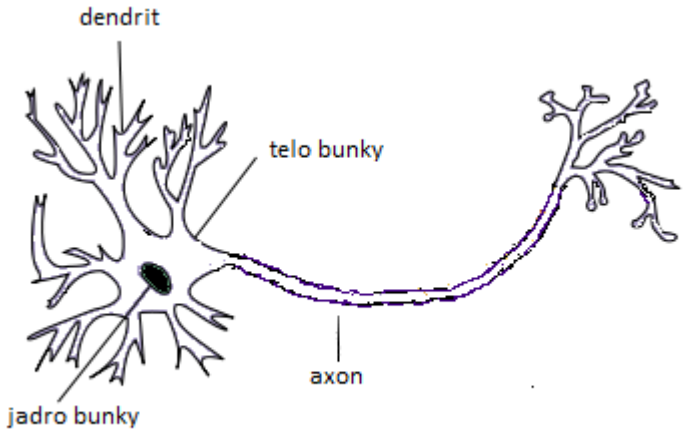
V tejto kapitole práce bude postupne objasnené, čo je to neurón a aký je jeho princíp. Následne bude vysvetlené ako fungujú neurónové siete a na záver sa dostaneme k problematike konvolučných neurónových sietí.

1.1 Neurón

Umelý neurón vychádza z teórie biologického neurónu, preto je na začiatok dôležité objasniť ako biologický neurón funguje.

1.1.1 Biologický neurón

Neuróny sa považujú za stavebné funkčné prvky nervovej sústavy. Sú to samostatné špecializované bunky, ktoré sú určené k prenosu, spracovaniu a uchovaniu informácií, ktoré sú nevyhnutné pre realizáciu životných funkcií organizmu. Štruktúra biologického neurónu je zobrazená na obrázku 1.1.



Obr. 1.1: Biologický neurón

Typický biologický neurón je tvorený telom (soma), z ktorého vybiehajú desiatky až stovky kratších výbežkov (dendrity) a jeden dlhý výbežok (axon). Veľkosť

tela neurónu je rádovo niekoľko mikrometrov. U dendritov je veľkosť niekoľko milimetrov a veľkosť axonu je niekoľko centimetrov a môže dosahovať až jeden meter. Axon každého neurónu je zakončený tzv. synapsiou, ktorá dosadá na iný neurón. Cez synapsie sa prenášajú vzruchy medzi neurónami. Neurón teda prijíma podnety od iných neurónov cez synapsie, v prípade kedy celkový membránový potencionál prekročí určitý prah, je neurón aktivovaný a cez svoju synapsiu začne pôsobiť na ďalšie neuróny, s ktorými je spojený [1].

1.1.2 Model umelého neurónu

Model umelého neurónu predstavuje zväčša abstrakciu mechanizmu, ako nervové bunky spracovávajú informácie. Nie je však možné vytvoriť presnú analógiu modelu skutočného neurónu, pretože biologické neuróny sú zložité objekty a doposiaľ nie je presne známy ich mechanizmus spracovania infromácie. Preto modely umelých neurónov, ktoré sa v súčastnosti najčastejšie používajú, opisujú iba základnú funkciu umelého neurónu.

Základné funkcia umelého neurónu spočíva v zbere elementárnych informácií, vyhodnotenia ich súhrnného stavu a zodpovedajúcej reakcie na výstupe.
Matematicky je formálny neurón chápaný ako vzťah:

$$in = f(q) = f(\sum_{i=1}^{n} x_i w_i) \tag{1.1}$$

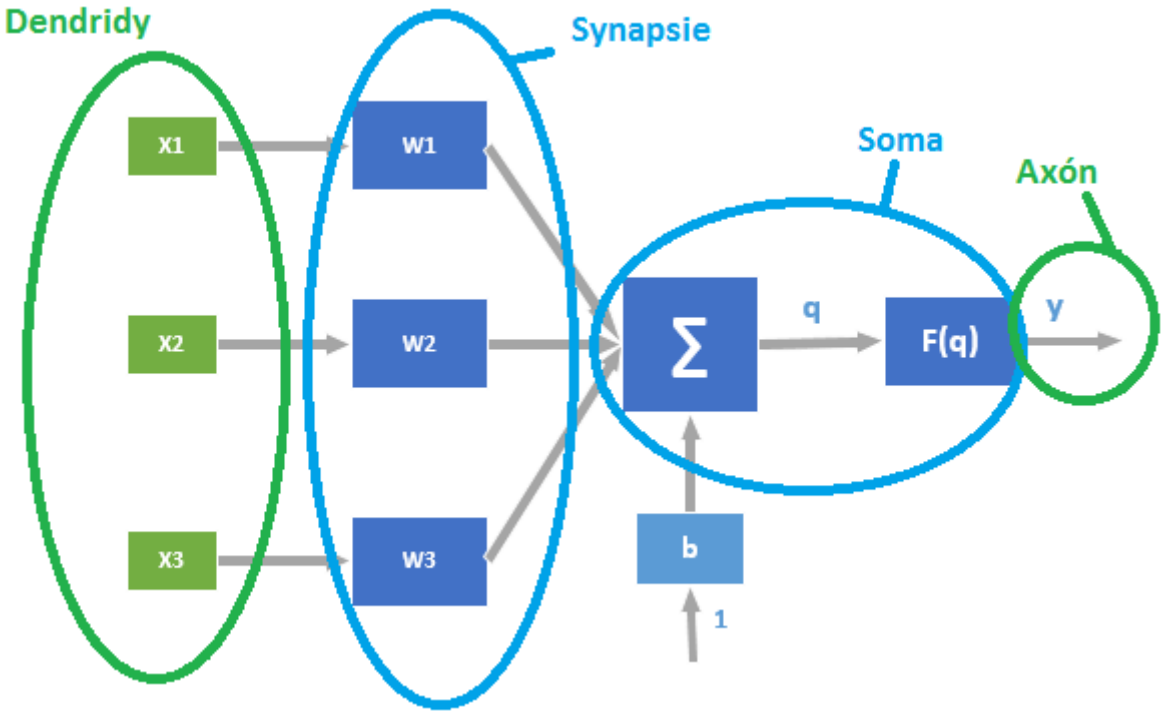
kde

- n – predstavuje počet vstupov;
- x_i – predstavuje i–ty vstupný signál neurónu;
- w_i – predstavuje i–tu vstupnú váhu;
- f – predstavuje aktivačnú funkciu neurónu;
- in – predstavuje vstupný signál neurónu;
- q – predstavuje potencionál neurónu.

Formálny neurón má obecne n reálnych vstupov, ktoré modelujú dendrity určujú vstupný vektor $\overrightarrow{x} = (x_1, ..., x_n)$. Tieto vstupy sú ohodnotené reálnymi *synaptickými váhami*, ktoré tvoria vektor $\overrightarrow{w} = (w_1, ..., w_n)$. [2]

Častejšie používaný model umelého neurónu obsahuje okrem iného aj prah (bias) b. Tento typ umelého neurónu je vidieť na obrázku 1.2 a je popísaný vzťahom:

$$in = f(\sum_{i=1}^{n} w_i x_i + b) \tag{1.2}$$

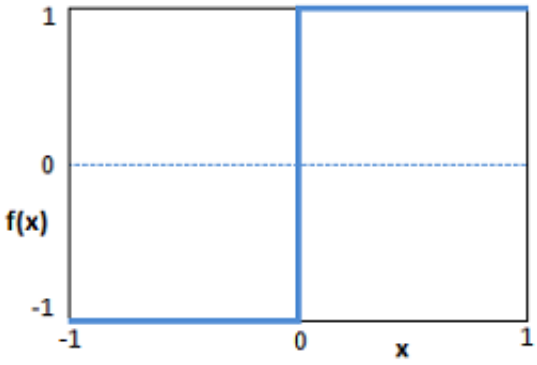


Obr. 1.2: Formálny neurón s vyznačenými časťami biologického neurónu

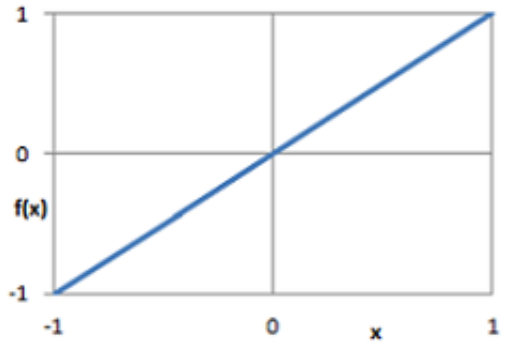
Pre dosiahnutie výstupu umelého neurónu je potrebné aplikovať *aktivačnú (prenosovú) funkciu f*. Existuje niekoľko typov aktivačných (prenosových) funkcií, k najpoužívanejším patria:

- aktivačné funkcie v tvare silných nelinearít, príkladom je skoková binárna funkcia $f(x) = sign(x)$ zobrazená na obrázku 1.3a
- lineárne aktivačné funkcie, príkladom je lineárna funkcia $f(x) = x$ zobrazená na obrázku 1.3b
- sigmoidálne aktivačné funkcie, príkladom je binárny sigmoid $f(x) = \frac{1}{1+e^{-x}}$ zobrazený na obrázku 1.3c
- aktivačné funkcie v tvare Gaussovej funkcie $f(x) = e^{-x^2}$, príklad je zobrazený na obrázku 1.3d

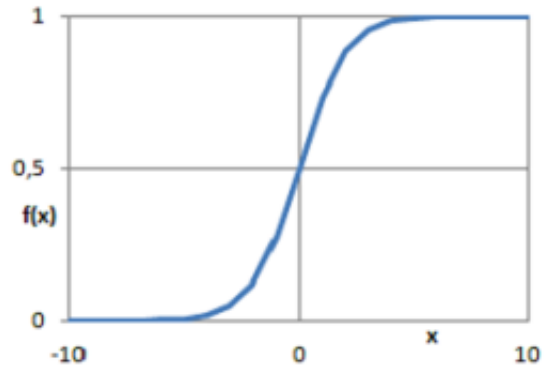
U skokovej aktivačnej funkcie je určený fixný prah, na základe ktorého sa určí akú hodnotu bude umelý neurón obsahovať (v našom prípade 0 alebo 1). V prípade lineárnej prenosovej funkcie je vstup prevedený na výstup bez zmeny. Sigmoidálne aktivačné funkcie majú ohraničenie (v našom prípade od 0 do 1) v nekonečne a používajú sa pre malé signály, kde umožňujú dosiahnuť veľkú citlivosť, avšak pre väčšiu úroveň signálu ich citlivosť klesá, v tomto prípade sa používa posledná zo spomenutých aktivačných funkcií – Gaussova aktivačná funkcia.



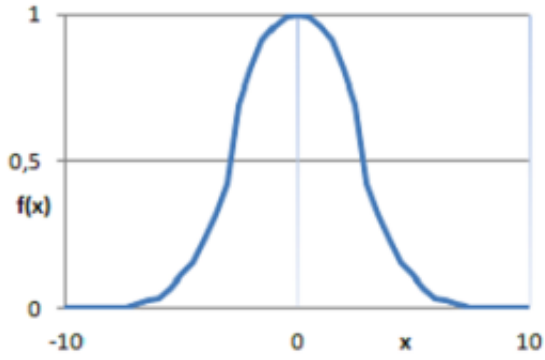
(a) Skoková binárna funkcia



(b) Lineárna funkcia



(c) Binárny (logický) sigmoid



(d) Gaussova funkcia

1.2 Umelá neurónová sieť

Každá umelá neurónová sieť sa skladá zo vzájomne prepojených formálnych neurónov tak, že výstup jedného neurónu je vstupom do iných neurónov. Počet neurónov a ich vzájomné prepojenie určuje topológiu umelej neurónovej siete, ktorá je určená ohodnoteným orientovaným grafom, ktorý predstavuje prepojenia medzi jednotlivými neurónmi. V tomto grafe jednotlivé neuróny predstavujú vrcholy a prepojenia medzi nimi hrany.

Matematicky môžme neurónovú sieť vyjadriť ako usporiadanú šesticu $M = (N, C, I, O, w, t)$, kde:

- **N** je konečná a neprázdna množina neurónov;
- **C** $\subseteq N$ x N je neprázdna množina orientovaných spojov medzi neurónmi;
- **I** $\subseteq N$ je neprázdna množina vstupných neurónov;
- **O** $\subseteq N$ je neprázdna množina výstupných neurónov;
- **w**: $C \rightarrow R$ je váhová funkcia;
- **t**: $N \rightarrow R$ je prahová funkcia.

Neurónová sieť je charakterizovaná niekoľkými vlastnosťami. Medzi hlavné vlastnosti neurónových sietí patrí napríklad *topológia siete*, *spôsob učenia* alebo *model siete*. U topológie neurónovej siete rozlišujeme dva základné typy:

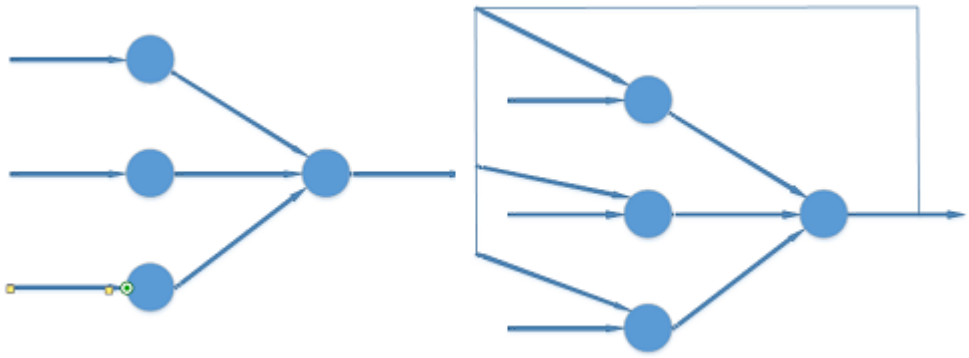
- neurónové siete s dopredným šírením signálu
- neurónové siete so spätnoväzobným šírením signálu

U neurónových sietí s dopredným šírením signálu (angl. feedforward neural network) postupujú všetky signály smerom zo vstupnej vrstvy do vrstvy výstupnej bez spätných väzieb. U sietí so spätnou väzbou (angl. feedback neural network) je vstup každého neurónu závislý na hodnote výstupu z predchádzajúceho cyklu. Príklad neurónov s dobrednou a spätnoväzobnou topológiou je uvedený na obrázku 1.4.

Umelé neuróny sú z hľadiska štruktúry organizované do vrstiev, pričom vo vrstve sa nachadzajú neuróny so spoločnými charakteristikami. Umelá neurónová sieť pozostáva minimálne z dvoch vrstiev a to vstupnej a výstupnej, prípadne ďalších vrstiev, ktoré sa nazývajú aj tzv. skryté.

Cieľom vstupnej vrstvy je zabezpečiť distribúciu vstupných signálov siete do ostatných vrstiev. Neuróny v nej majú jediný vstup, ktorý je zároveň aj vstupom do neurónovej siete a ich prenosová funkcia reprezentuje lineárnu funkciu, čo znamená, že neuróny vstupnej vrstvy posielajú vstupný signál na svoj výstup bez zmeny.

Výstupná vrstva určuje výstup neurónovej siete. Vrstvy, ktoré sa nachádzajú medzi vstupnou a výstupnou vrstvou sa nazývajú *skryté*, preotže vstupy ani výstupy, ktoré sa v nich nachádzajú, nezasahujú priamo do vonkajšieho prostredia. [1]



Obr. 1.4: Príklad neurónov s dobrednou a spätnoväzobnou topológiou

Činnosť umelej neurónovej siete možno chápať ako transformáciu vstupného signálu in na výstupný signál $out \rightarrow in = T(out)$ a jej chovanie je možno definovať tromi základnými prvkami:

- aktivačnými funkciami neurónov
- množinou spojenii medzi neurónmi
- učiacim pravidlom

Aktivačná funkcia je vyberaná tak, aby umelá neurónová sieť bola schopná mapovať, čo najširší rozsah lineárnych a nelineárnych vstupno-výstupných vzťahov. O umelej neurónovej sieti hovoríme, že je *stabilná*, práve vtedy keď všetky jej váhy konvergujú do rovnovážneho stavu, v ktorom neurónový model aproximuje aktuálne správanie procesu. V prípade, že neurónová sieť diverguje, hovoríme o nej ako o *nestabilnej*.

1.3 Učenie neurónovej siete

Dôležitou vlastnosťou umelých neurónov a umelých neurónových sietí je ich schopnosť učiť sa. Učením sa myslí algoritmus nastavenia váh $\mathbf{w}$, tak aby systém, čo najsprávnejšie spracovával aj neznáme podklady.

Rozlišujeme 2 typy učiacich algoritmov pre umelé neurónové siete. Ide o tzv. *učenie s učiteľom* a *učenie bez učiteľa*. Metóda *učenia s učiteľom* je charakterizovaná tým, že požadovaný výstup je známy. Príkladom tejto metódy je situácia, kedy sa neurónová sieť učí pomocou vektora vstupných hodnôt a k nemu prislúchajúceho vektora výstupných hodnôt. Počas učenia je hodnota zo vstupného vektora privedená na vstupnú vrstvu neurónovej siete. Následne je táto hodnota spracovaná neurónovou sieťou a privedená na výstupnú vrstvu. Hodnota z výstupnej vrstvy je porovnaná s príslušnou hodnotou z výstupného vektora. V prípade odchýlky medzi výstupnou a očakávanou hodnotou sa počíta chybová funkcia, na základe ktorej sa následne upravuje hodnota váh.

U metódy *učenia bez učiteľa* nie sú definované žiadne požadované výstupy a váhy sa nastavujú len pomocou vstupných údajov. Sieť sa potom sama rozhodne, ktorá odozva je pre daný vstup najlepšia a podľa toho nastaví svoje váhy. [2]

1.4 Modely umelých neurónových sietí

1.4.1 Perceptron

Jedným z najjednouchších modelov umelej neurónovej siete je Perceptron, ktorého autorom bol v roku 1957 Frank Rosenblatt. Pôvodne bol navrhnutý ako model zrakovej sústavy. Typický perceptron sa skladá z n vstupov $(x_1, x_2, ..., x_n)$ a jedným výstupným neurónom, ktorý je spojený so všetkými svojími vstupmi. Vstupy obsahujú binárne hodnoty 0 alebo 1. Každé spojenie medzi vstupným a výstupným neurónom má priradenú váhovú hodnotu $(w_1, w_2, ..., w_n)$. Výpočet vstupu do výstupého neurónu má tvar:

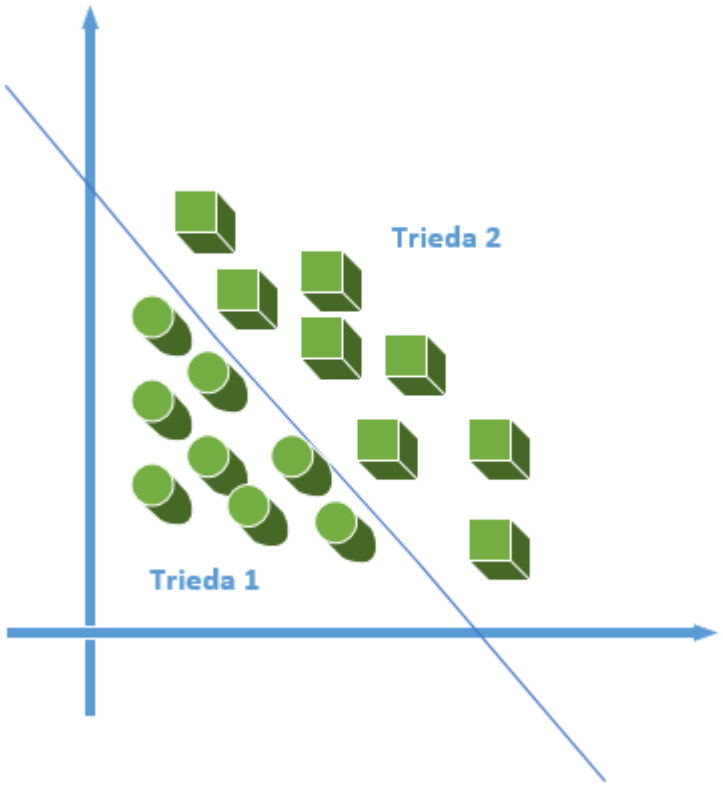
$$in(t) = \sum_{i=1}^{n} w_i(t)x_i(t) - \Theta \tag{1.3}$$

kde n, je počet vstupov a Θ je prah výstupného neurónu. Aktivačná funkcia f je skoková a má tvar:

$$u(x) = \begin{cases} 1 & \text{ak } in(t) \geq 0 \\ 0 & \text{ak } in(t) < 0 \end{cases} \tag{1.4}$$

Pri učení perceptronu sa používa metóda učenia s učiteľom a algoritmus učenia je v tvare:

- Nastavenie váh náhodnými hodnotami a nastavenie prahu
- Aktivácia vstupného vektoru do siete
- Výpočet skutočnej hodnoty na výstupe
- Porovnanie hodnoty z výstupu siete s očakávanou hodnotou
- Adaptácia váh a výpočet chybovej funkcie e, ako rozdielu zistenej a očakávanej hodnoty
  - Ak je vstupný vektor vyhodnotený správne, tak sa váhy neupravujú a pokračuje sa vo vyhodnocovaní ďalšieho vstupného vektoru
  - Ak je výstup rovný 1, ale má byť 0, inkrementujú sa váhy na aktívnych vstupoch
  - Ak je výstup rovný 0, ale má byť 1, dekrementujú sa váhy na aktívnych vstupoch
- Úprava prahu pričítaním hodnoty chybovej funkcie

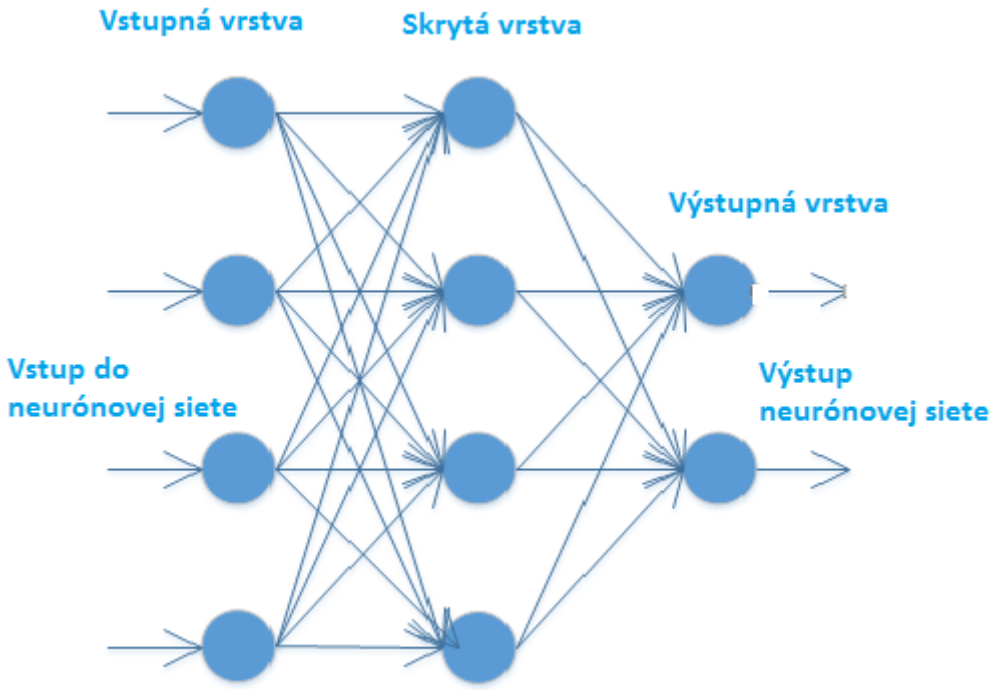


Obr. 1.5: Príklad lineárne separovateľnej množiny

Využitie perceptronu je najmä u lineárne separovatelných obrázkov, napríklad v prípade priradenia bodu do konkrétnej množiny. Učiaci algoritmus u perceptronu je konečný iba v prípade, že riešenie existuje a učiaca množina je lineárne separovateľná. Príklad lineárne separovateľnej množiny je na obrázku 1.5. Komplikovanejšie funkcie ako je napríklad XOR, ktoré nie je možné riešiť pomocou perceptronu, je riešené rozšírením základného modelu perceptronu do *viacvrstvovej doprednej neurónovej siete*. [1]

1.4.2 Viacvrstvová dopredná neurónová sieť

Viacvrstvová dopredná neurónová siet (angl. multilayer feedforward neural network) bola uvedená v roku 1986 a jej autormi sú G.E. Hinton, E. Rumelhart a R.J. Williams. Cieľom tohto modelu neurónovej siete bolo vyriešiť problém s obmedzenou výpočtovou schopnosťou perceptronov. Dopredné neurónové siete sa vyznačujú, tým že signály prechádzajú sieťou iba dopredným smerom od vrstvy k vrstve a sieť je zložená minimálne z troch vrstiev neurónov rozdelených na vstupnú, skrytú a výstupnú vrstvu.



Obr. 1.6: Viacvrstvová dopredná neurónová sieť

Spojenia medzi jednotlivými neurónmi sú realizované tak, že každý neurón *i*–tej vrstvy je spojený s každým neurónom *i+1*–tej vrstvy. Spojenia medzi neurónmi tej istej vrstvym prípadne vzdialených vrstiev neexistujú. Príklad viacvrstvovej doprednej neurónovej siete je ukázaný na obrázku 1.6.

a) Učenie doprednej neurónovej siete a metóda spätného šírenia chyby

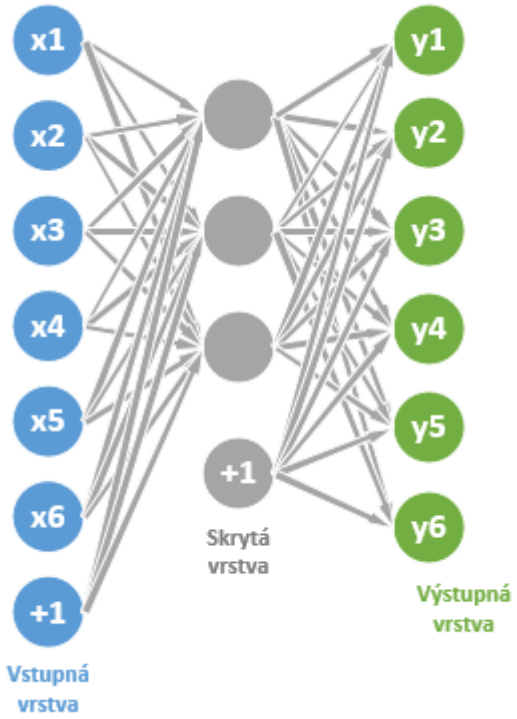
Pre učenie dopredných viacvrstvových neurónových sietí sa najčastejšie používa metóda *spätného šírenia chyby* tzv. **backpropagation** Nejde však o spätné šírenie signálu ako je to u rekurentných sietí, ale o spätné šírenie chyby. Táto metóda bola publikovaná 80–tych rokoch a jej autorom bol E. Rumelhart. Metóda backpropagation je založená na zmenách hodnôt synaptických váh počítaných na základe chyby, pričom za chybu sa berie rozdiel medzi očakávaným a skutočným výstupom neurónovej siete. K minimalizácií chybovej funkcie sa používajú gradientné metódy.

V praxi pri učení doprednej neurónovej siete máme dva vektory, kde prvý vektor charakterizuje hodnoty, ktoré budú tvoriť vstup do neurónovej siete a druhý vektor k nim príslušné požadované výstupy neurónovej siete. Na začiatku učenia je na vstup neurónovej siete privedený vektor vstupných hodnôt. Po priechode neurónovou sieťou je jej výstup porovnávaný s očakávaným výstupom.. V prípade, že tieto

hodnoty nie sú zhodné, dochádza k počítaniu chyby, ktorá sa spätne prepočítava do predchádzajúcich vrstiev, čo spôsobí, že jednotlivé váhy sa upravujú. Ide o iteratívny proces, pretože do opravenej siete sa opäť privádza vstupný vektor a celý proces sa opakuje. Pri základnom učení sa používa iba zmena synaptických váh a prahov, parametre prenosových funkcií ostávajú nemenné. [2]

1.4.3 Autoenkóder

Autoenkóder je typ neurónovej siete, v ktorom je učenie neurónovej siete založené na metóde učenia bez učiteľa. Táto metóda je založená na príncipe tzv. *kóderu – dekóderu*, kde na začiatku je vstup typicky zmenšený (kódovaný)[13]. Tento zmenšený výstup je následne opäť zväčšený, aby reprodukoval (dekódoval) pôvodný vstup. Prvé autoenkódery boli uvedené v osemdesiatych rokoch 20. storočia ako siete, ktoré miesto učiteľa využívajú na učenie vlastný vstup. Väčšie využitie však našli až v hlbších architektúrach po roku 2006. [14] Príklad autoenkóderu je vidieť na obrázku 1.7.



Obr. 1.7: Autoenkóder

Cieľom autoenkóderu je naučiť sieť aproximovať funkciu identity $h_{W,b}(x) \approx y$, kde výstup y je podobný vstupu x. Na začiatku autoenkóder berie vstup $x \in R^d$, ktorý mapuje do skrytej vrstvy $h \in R^d$. Pri tomto mapovaní je použitá následná

deterministická funkcia:

$$h = f_\theta = \sigma(W_x + b), \tag{1.5}$$

kde parameter $\theta = \{W, b\}$.
Pri rekonštrukcií výstupu je potom použité reverzné mapovanie tejto funkcie, ktoré má tvar:

$$y = f_{\theta'}(h) = \sigma(W'h + b') \tag{1.6}$$

kde parameter $\theta' = \{W', b'\}$ a parameter $W' = W^T$
V týchto funkciách sú použité rovnaké váhy pre kódovanie aj dekódovanie.

Spomínaný enkóder platrí medzí bežné enkódery, ktoré pri mapovaní používajú len bežnú funkciu identity a teda nie sú schopné pracovať so vstupmi, ktoré sú určitým spôsobom porušené. V týchto prípadoch sa používa špeciálna stochastická verzia autoenkódera, ktorý sa označuje ako tzv. *denoising autoencoders*, ktorého cieľom je trénovať enkóder tak, aby bol schopný rekonštruovať správny vstup, z jeho porušenej verzie.

V súčastnosti sa enkóderi uplatňujú najmä v zložitých neurónových sieťach, ktoré sú tvorené zhlukmi enkóderov. V týchto sietiach tvorí vstupná vrstva prvého enkóderu vstup celej siete. Potom následujúca vrstva, ktorá tvorí skrytú vrstvu vstupného autoenkóderu, je súčasne vstupnou vrstvou ďalšieho autoenkóderu. Týmto spôsobom je sieť zložená z ľubovoľného množstva autoenkóderov.

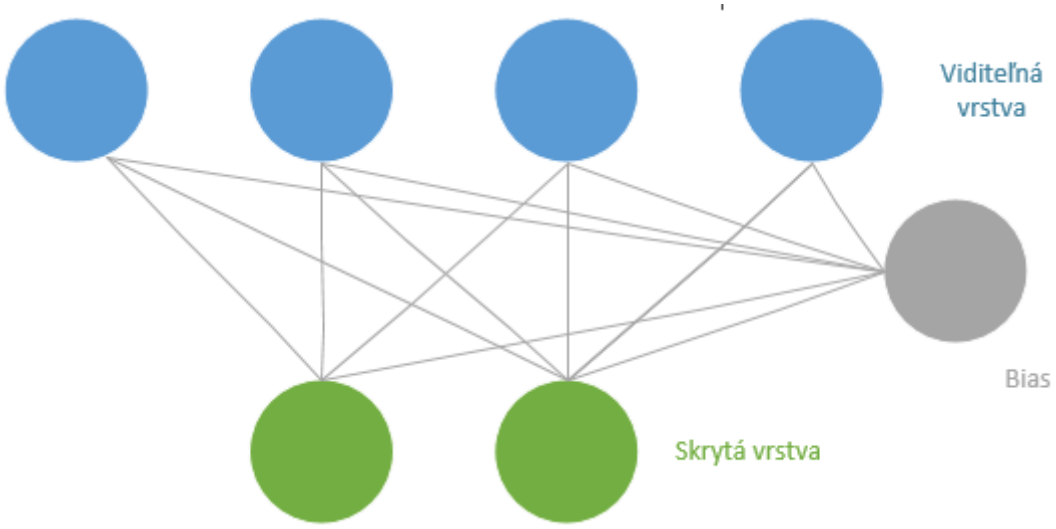
Učenie takejto siete prebieha spôsobom, kedy najprv prebehne učenie enkóderu, ktorý je na vstupe neurónovej siete. Po učení tohto enkóderu je odstránená dekódovacia vrstva, ktorá je nahradená novým autoenkóderom, ktorého vstup predstavuje skrytá vrstva predchádzajúceho enkóderu. Týmto spôsobom je možné následne učiť obrovské siete s veľkým množstvom skrytých vrstiev.[14]

1.4.4 Obmedzené Bolzmmanové stroje

Obmedzené Bolzmmanové stroje patria medzi stochastický typ neurónových sietí, kde binárna aktivačná funkcia neurónu je závislá na susedoch, s ktorými je neurón spojený. Pôvodne boli Obmedzené Bolzmmanové stroje vytvorené v roku 1986 Paulom Smolenskym, ale populárnejšie začali byť až po roku 2000, kedy našli uplatnenie napríklad pri dimenzionálnej redukcií, klasifikácií, či kolaboratívnom filtrovaní. V súčasnosti je široké použitie Obmedzených Bolzmmanových strojov najmä v hlbokom učení sietí a v zložitých neurónových sieťach.[18]

Obmedzené Bolzmmanové stroje sú tvorené jednou viditeľnou neurónovou vrstvou, jednou skrytou neurónovou vrstvou a biasom, ktorého hodnota je vždy rovná jednej. Navyše každý neurón z viditeľnej vrstvy je prepojený s každým neurónom zo skrytej

vrsty. Výnimočnosť týchto spojov je, že sú neorientované, teda aj každý neurón zo skrytej vrstvy je prepojený s každým neurónom z vrstvy viditeľnej. Bias je prepojený s každým neurónom zo skrytej aj viditeľnej vrstvy. Obmedzené Bolzmmanové stroje sú limitované tak, že žiadny neurón z viditeľnej vrstvy nesmie byť prepojený s iným neurónom z viditeľnej vrstvy a rovnako aj žiadný neurón zo skrytej vrstvy nesmie byť prepojený s iným neurónom zo skytej vrstvy. Príklad Obmedzeného Bolzmmanovho stroja je vidieť na obrázku 1.8. [19]



Obr. 1.8: Obmedzený Bolzmanov stroj

Pri učení Obmedzených Bolzmmanových strojov sa používa najčastejšie algoritmus, ktorý sa označuje ako *kontrastná divergencia*. Tento algoritmus bol vytvorený G. Hintonom a pôvodne bol určený pre trénovanie modelov expertných produktov.[20] Princíp tohto algoritmu sa skladá z niekoľkých krokov.

- Najprv je vybraná trénovacia vzorka v;
- Vypočítajú sa pravdepodobnosti neurónov zo skrytej vrstvy, z týchto hodnôt je následne vytvorený aktivačný vektor h;
- Z dosiahnutých hodnôt v a h je vypočítaný vektorový súčin, ktorý sa v tomto prípade označuje aj ako *pozitívny gradient*;
- Z hodnoty h sa spätne vypočíta hodnota v' pre neuróny z viditeľnej vrstvy;
- Z vypočítanej hodnoty v' sa opäť počítajú pravedepodobnosti neurónov zo skrytej vrstvy, z ktorých je následne vytvorený aktivačný vektor h';
- Opäť je počítaný vektorový súčin, ale aktuálne pre hodnoty v' a h'. Tento vektrorový súčin sa označuje ako *negatívny gradient*;
- Vypočíta sa efektívna hodnota učenia ako

$$wt + 1 = w(t) + a(\overrightarrow{v}\,\overrightarrow{h}^T - \overrightarrow{v}'\overrightarrow{h}'^T) \tag{1.7}$$

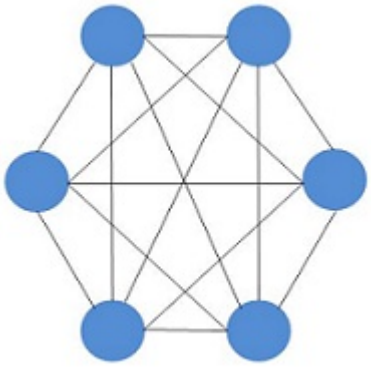
- Nakoniec je nová hodnota váh vypočítaná ako pozitívny gradient mínus negatívny gradient krát efektívna hodnota učenia.

Ako už bolo v tejto časti spomenuté, princíp Obmedzených Bolzmmanových strojov je v sučasnosti rozšírený pri učení zložitých neurónových sietí. V tomto prípade je použitá myšlienka, kedy aktivačný vektor h pre neuróny skrytej vrstvy môže byť ďalej použitý ako vektor pre trénovanie inej skrytej vrstvy iného Obmedzeného Bolzmanového stroja. Tento princíp môže byť aplikovaný do niekoľkých skrytých vrstiev, ktoré následne môžu byť brané ako jeden veľký celok. Tento spôsob je veľmi výhodný pre neurónové siete, ktoré sa skladajú z obrovského množstva skrytých vrstiev a milionov váh. Tento typ sietí sa označuje ako „*Deep Believe Networks*“. [19]

1.4.5 Hopfieldova sieť

Hopfieldová neurónová sieť bola vytvorená v roku 1982 Johnom Hopfieldom Táto sieť je tvorená neurónmi, ktorých výstup je tvorený hodnotami $+1$ alebo -1, kde ako je vidieť v rovnici 1.8 je hodnota $+1$ nastavená na výstup neurónu v prípade, kedy súčet váh má hodnotu väčšiu ako 0. [22] V prípde, že je hodnota súčtu váh menšia ako 0, je výstup neurónu nastavený na hodnotu -1

$$u(x) = \begin{cases} 1 & \text{ak } \sum_i w_i x_i \geq 0 \\ 0 & \text{ak } \sum_i w_i x_i < 0 \end{cases} \tag{1.8}$$



Obr. 1.9: Model Hopfieldovej neurónovej siete

Hopfieldová neurónová sieť je tvorená neurónmi, ktoré sú v tejto sieti plne prepojené. Príklad Hopfieldovej neurónovej siete je vidieť na obrázku 1.9. Hodnoty všetkých váh spojení medzi neurónmi sú uložené vo váhovej matici W, ktorá reprezentuje stav siete. Pre jednotlivé váhy siete platí, že diagonálne váhové koeficienty

w_{ii} sú rovné 0, nediagonálne váhové koeficienty sú symetrické, teda pre ne platí, že $w_{ij} = w_{ji}$. Aktualizácia hodnôt váh a teda aj neurónov môže byť prevedená dvomi spôsobmi:

1. *Asynchrónne*, kedy je vybraný neurón, ktorému je pomocou váh vypočítaný výsledný vstup a následná aktualizácia okamžite. Výber jednotlivých neurónov môže prebiehať buď v určenom poradí alebo náhodne. V prípade, že neuróny sú vyberané náhodne, sa táto metóda nazýva aj *náhodná asynchrónna aktualizácia.*
2. *Synchrónne*, kedy sú vstupy vektorov počítané bez okamžitej aktualizácie hodnôt neurónov, ktorá prebieha pre všetky neuróny súčasne.

Hopefieldová sieť má ku každému stavu siete priradenú skalárnu hodnotu, ktorá sa nazýva aj ako *„energia“* siete E, ktorej cieľom je dosiahnuť stav s čo najmenšou energiou [23]. Výpočet tejto hodnoty je zobrazený na následujúcom vzťahu:

$$E = -\frac{1}{2}\sum_{i,j} w_{ij} s_i s_j + \sum_i \theta_i s_i \tag{1.9}$$

Táto hodnota zaisťuje, že po jednotlivých aktualizáciach siete je energia jednotlivých stavov rovnaká alebo menšia ako pri predchádzajúcom stave siete. Potom v prípade nájdenia lokálneho minima je funkcia energie stabilná.

Hopefieldová sieť sa používa najmä ako autoasociatívna pamäť, čo znamená že si dokáže zapamätať informácie, ktoré vie následne identifikovať a to aj na základe poškodenej alebo neúplnej predlohy. Hopefieldová sieť bola taktiež použitá pri formulácií postupnej pomalej degradácií schopnostii v mozgu pri vymieraní neurónov a zániku spojov.

U Hopefieldovej siete sa stretávame s niekoľkými obmedzeniami, kedy sieť nefunguje podľa očakávaní. Jedným z obmedzením siete je schopnosť uchovania iba obmedzeného počtu vzorov. Ďalším z problémov hopefieldovej neurónovej siete, že môže dôjsť k situácii, kedy sa v sieti nachádzajú dva stavy s rovnakou hodnotou energie, čo zabraňuje konvergencii. Tento jav býva označovaný aj ako *oscilácia* siete. [23]

a) Hebbovo učenie

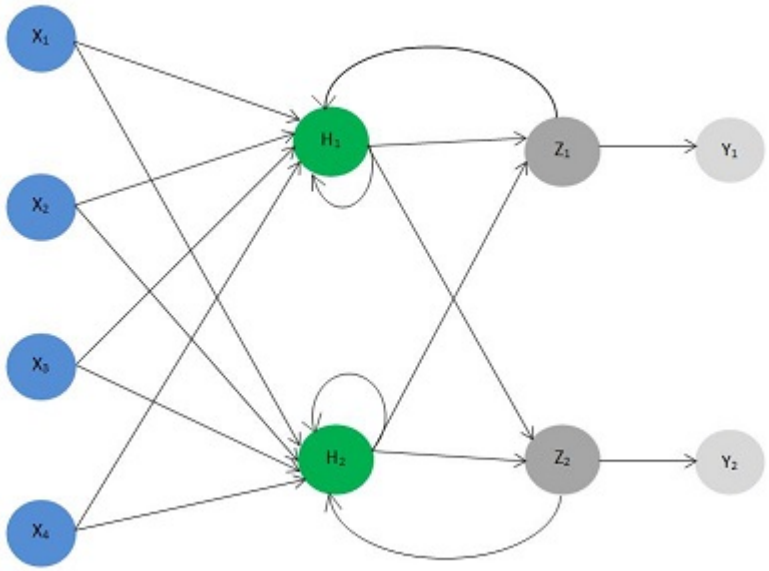
Jedným zo spôsobov učenia Hopefieldovej siete je tzv. „*Hebbovo učenie*“, ktoré je založené na pravidle, ktoré bolo vytvorené v roku 1949 psychológom Donaldom Hebbom. Toto učenie spočíva v myšlienke, ktorá hovorí, že váhové hodnoty na spojoch medzi neurónmi sa zvyšujú v prípade, ak sú oba prepojené neuróny aktivované synchrónne, teda v jeden časový okamih. A naopak, váhové hodnoty sa znižujú v prípade, ak sú prepojené neuróny aktivované asynchrónne, teda v rôznych časoch. [23]

Následujúci algoritmus popisuje jednotlivé kroky Hebbovho učenia:

1. Najprv prebehne inicializácia všetkých váh ako: $w_i = 0$ (pre i= 1 až n)
2. Následne prebehne aktivácia neurónov
3. Potom sa aktualizujú váhy podľa následujúceho pravidla: $w_i(new) = w_i(old) + x_i y$ (pre i= 1 až n)

1.4.6 Rekurentné neurónové siete

Na rozdiel od dopredných neurónových sietí, ktorých spojenia medzi neurónmi majú jeden smer, obsahujú rekurentné neurónové siete minimalne jedno cyklické spojenie. Rekurentné neurónové siete sa najčastejšie používajú pri modelovaní biologických procesov alebo implementácií dynamických systémov.



Obr. 1.10: Model rekurentnej neurónovej siete

Rekurentné neurónové neurónové siete sú podobne ako dopredné neurónové siete zložené z neurónov rozdelených do vstupných, skrytých (alebo interných) a výstupných vrstiev, kde spojenia medzi neurónmi nie sú vedené jedným smerom, ako napríklad výstupná vrstva neurónov môže obsahovať spojenia nie len s neurónmi zo skrytej vrstvy ale aj zo vstupnej vrstvy, prípadne aj medzi sebou vo vrstve výstupnej. Príklad rekurentnej neurónovej siete je vidieť na obrázku 1.10. Aktivačná funkcia neurónov v skrytej vrstve je potom v tvare:

$$x(n+1) = f(W^{vstup}u(n+1) + Wx(n) + W^{výstup}y(n)) \tag{1.10}$$

V prípade rekurentných neurónových sietí sa najčastejšie ako aktivačná funkcia používa sigmoid, prípadne lineárna funkcia. Výstup siete je následne počítaný ako:

$$y(n+1) = f^{výstup}(W^{výstup}(u(n+1), x(n+1))), \tag{1.11}$$

kde $(u(n+1), x(n+1))$, predstavuje zložený vektor zo vstupných a skrytých aktivačných vektorov.

Učenie rekurentných neurónových sietí prebieha najčastejšie prostredníctvom učenia s učiteľom. Sieť je učená dátami tak, aby bola schopná identifikovať naučené dáta a generalizovať vstupy na základe podobnosti a spoločných znakov s dátami použitých pri učení. V súčasnosti existuje niekoľko typov algoritmov, ktoré sa používajú pre učenie rekurentných neurónových sietí ako napríklad algoritmu rekurentného učenia v reálnom čase, ktorý patrí medzi najrozšírenejší algoritmus. Ďalšími algoritmami sú napríklad algoritmus spätného šírenia v priebehu času alebo Rozšírené Kalmanové filtračné techniky.

Rovnako aj tento typ neurónových sietí má isté nevýhody. V tomto prípade môže nastať stav, ktorý sa nazýva ako tzv „*overfitting*“, kedy bolo sieti predložené veľké množstvo učiacich dát, že nedokáže spoľahlivo generalizovať dáta zo vstupu [24].

2 KONVOLUČNÉ NEURÓNOVÉ SIETE

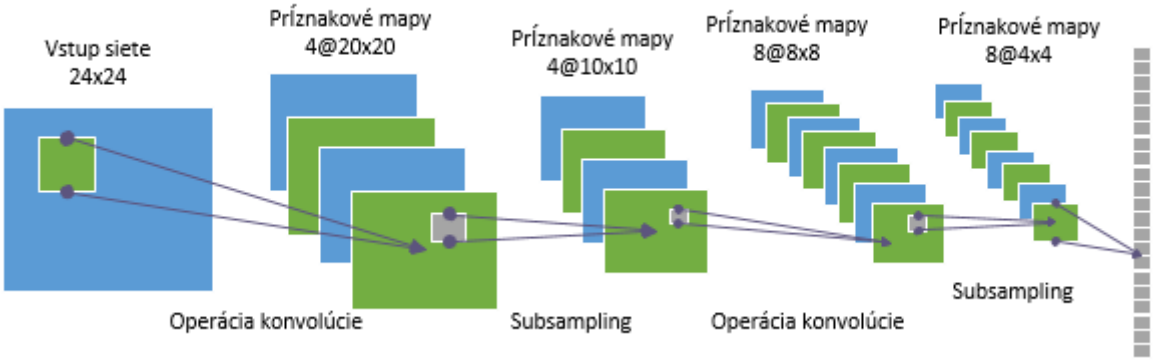
Táto časť práce bude venovaná špeciálnemu druhu neurónových sietí, ktorým sú konvolučné neurónové siete. Postupne bude popísaná štruktúra konvolučných neurónových sietí, ich učenie a príklady reálneho použitia.

2.1 Základná charakteristika

Konvolučné neurónové siete sú špeciálnym druhom viacvrstvových neurónových sietí, ktoré sa používajú pre rozpoznávanie vizuálnych vzorov priamo z pixelov obrázka s použitím minimáleho predspracovania. Vychádzajú z princípov neocognitronovej siete, ktorá bola uvedená v roku 1987 K. Fukushimom. Neocognitron pozostáva z veľkéo množstva neurónových vrstiev a obsahuje variabilné spojenia medzi bunkami susedných vrstiev. Cieľom tejto siete je identifikácia objektov na základe podobností vzorov bez ohľadu na deformáciu, zmenu veľkosti alebo posunutia pozície, čo sa uplatnilo napríklad pri implementácii rozpoznávania ručne písaných znakov. [4]

Typická konvolučná sieť je ukázaná na obrázku 2.1 a ako je vidieť skladá sa z niekoľkých vrstiev. K charakteristickým vlastnostiam konvolučných neurónových sietí patrí získavanie príznakov z recepčných polí, technológia zdieľaných váh a priestorové vzorkovanie.

Prvou vrsvou je tzv. *konvolučná vrstva*. Ako už bolo spomínané na začiatku, na vstup konvolučnej neurónovej siete je najskôr privedený obrázok, na ktorý je aplikovaný filter, ktorého cieľom je získavanie príznakov. Tento filter je zložený z neurónov, ktoré medzi sebou zdieľajú rovnakú množinu váh vektorov, čo znižuje počet parametrov siete a teda aj veľkosť jej kapacity. Pomocou týchto lokálnych recepčných polí dokáže sieť z obrázkov získať príznaky ako sú hrany alebo rohy. Ďalšou výhodou konvolučných neurónových sietí je zdieľanie váh, ktoré urýchľuje učenie siete tým, že sa obmedzí celkový počet výpočtov siete. Tento filter sa označuje ako *recepčné pole*. Toto recepčné pole je postupne aplikované na celý vstupný obrázok, teda počíta jednu operáciu v rôznych častiach obrázku. Vo väčšine prípadov sa na jeden vstupný obrázok aplikuje niekoľko recepčných polí, s tým že jednotlivé recepčné polia sa líšia hodnotamu zdieľaných váh vektorov. Výstupom jedného recepčného poľa, ktoré bolo aplikované na celý vstupný obrázok je tzv. *príznaková mapa* a počet príznakových máp zodpovedá počtu recepčných polí aplikovaných na obrázok. V prípade našeho príkladu na obrázku 2.1 vidíme, že prvá konvolučná vrstva je zložená zo štyroch príznakových máp a každá mapa obsahuje 20x20 recepčných polí.



Obr. 2.1: Konvolučná neurónová sieť

V prípade, že je príznak pomocou recepčného poľa identifikovaný, stáva sa lokácia, v ktorej bol príznak identifikovaný menej podstatná. Z tohto dôvodu sa za konvolúčnou vrstvou nachádza ďalšia vrstva, ktorá sa označuje ako *subsamplingová*. Cieľom tejto vrstvy je zredukovať veľkosť vstupu. V praxi je počas tejto metódy na všetky príznakové mapy, ktoré sú výstupom konvolučnej vrstvy aplikovaná funkcia, ktorej cieľom je tieto mapy zmenšiť. Existuje niekoľko typov funkcií, ktoré je možno použiť v subsamplingovej vrstve ako napríklad priemerovanie, výber maxima alebo lineárna kombinácia neurónov v danej príznakovej mape.

V súčastnosti sa najčastejšie v subsamplingovej vrstve používa funkcia výberu maxima alebo „*max-pooling*“, ktorá vezme na vstup oblasť s rozmermi k x k, z predchádzajúcej konvolučnej vrstvy a následne z tejto oblasti vyberie ako výstup hodnotu, ktorá je maximálna v tejto oblasti. Pre predstavu, ak vstup do subsamplingovej vrstvy má rozmery N x N, jej výstup bude mať rozmery $\frac{N}{k}$ x $\frac{N}{k}$.

Na záver, po niekoľkých konvolučných a subsamplingových vrstvách je výstup z príznakovej mapy poslednej konvolučnej alebo subsamplingovej vrstvy transformovaný do jednorozmerného vektoru. Tento vektor je následne privedený ako vstup do neurónovej siete, ktorá vypočíta konečný výstup.

Nevýhodou u konvolučných neurónových sietí je veľký počet parametrov, ktorý spôsobuje, že učenie siete je komplikovanejšie ako napríklad u neurónových sieťach s dopredným šírením. Ďalším problémom u konvolučných neurónových sietí býva určenie vhodného počtu filtrov, ktoré sú aplikované v konvolučnej vrstve. Pri tvorbe siete je nutné myslieť na to, že výpočty vo filtroch sú náročnejšie ako výpočty používané v skrytých vrstvách dopredných neurónových sietí, preto aj počet filtrov v konvolučných sieťach býva nižší ako počet skrytých vrstiev v dopredných neurónových sieťach.

Problémom u konvolučných neurónových sieťach je taktiež aj výber vhodného

rozmeru filtru. Z praxe u databázy MNIST, ktorá obsahuje obrázky o rozmere 28x28 ručne písaných znakov a to 60000 príkladov pre trénovanie siete a 10000 príkladov pre testovanie [11]. V tomto prípade sa ako vhodná veľkosť filtru v prvej konvolučnej vrstve ukázal rozmer 5x5. V súčastnosti však veľkosť obrázkov nadobúda niekoľkonásobne väčšie rozmery, čo znamená, že aj filtre použité v konvolučných vrstvách musia byť väčšie. Pre obrázky o veľkosti niekoľko stoviek pixelov v každom rozmere sa používajú filtre v prvej konvolučnej vrstve približne o rozmere 15x15.

2.2 Učenie konvolučnej neurónovej siete

Pri písaní tejto časti som čerpala z [5]. Ako už bolo spomínané vyššie konvolučné neurónové siete sú zložené z jednej alebo viacerých konvolučných a subsamplingových vrstiev, ktoré sú zakonečené plne prepojenou jednorozmernou sieťou. Na základe týchto poznatkov sa všetky vlastnosti z predchádzajúcich vrstiev siete spájajú do jedného vektoru, ktorý je následne posielaný na vstup siete. Táto metóda učenia zodpovedá metóde spätného šírenia chyby, ktoré bolo spomenuté v časti a), avšak u konvolučných neurónových sietí je táto metóda upravená pre jednotlivé vrstvy siete.

Predpokladajme, že za konvolučnou vrstvou l nasleduje subsamplingová vrstva $l+1$, v tomto prípade jeden pixel zo subsamplingovej vrstvy predstavuje blok pixelov v konvolučnej vrstve, čo znamená, že niekoľko jednotiek z vrstvy l je spojených do jednej vo vrstve $l+1$. Pre počítanie chýb v konvolučnej vrstve l je preto vhodné zväčšiť mapu chýb zo subsampligovej vrstvy $l+1$ na veľkosť mapy z konvolučnej vrstvy l. Následne bude vynásobená zväčšená mapa zo subsamplingovej vrstvy $l+1$ deriváciou aktivačnej funkcie príslušnou mapou z konvolučnej vrstvy l. [5] Tento postup môžeme aplikovať na každú mapu konvolučnej vrstvy, podľa rovnice chyby j–tej mapy v l–tej vrstve:

$$\delta_j^l = \beta_j^{l+1}(f^{'}(u_j^l) \circ up(\delta_j^{l+1})) \tag{2.1}$$

kde β_j^{l+1} predstavuje váhy definované v subsamplingovej vrstve, $f^{'}(u_j^l)$ predstavuje deriváciu aktivačnej funkcie neurónu u_j^l. Funckia $up()$ predstavuje operáciu zväčšenia subsamplingovej mapy, tým že každý pixel mapy je horizontálne aj vertikálne ukladaný n–krát. Funkciu $up()$ si môžeme predstaviť ako následujúci vzťah:

$$up(x) = x \otimes 1_{nxn} \tag{2.2}$$

V prípade, že máme vypočítanú chybu pre každú mapu, môžeme následne počítať gradient prahu ako súčet jednotlivých chýb máp δ_j^l:

$$\frac{\partial E}{\partial b_j} = \sum_{u,v} (\delta_j^l)_{u,v} \tag{2.3}$$

Podobne vypočítame gradient váh, ktorý prebieha podobne ako u metódy spätného šírenia chyby, s tým rozdielom, že váhy u konvolučných neurónových sietí sú zdieľané cez niekoľko spojení.

$$\frac{\partial E}{\partial k_{ij}^{l}} = \sum_{u,v} (\delta_j^l)_{uv} (p_i^{l-1})_{l-1} \tag{2.4}$$

Kde $(p_i^{l-1})_{u,v}$ predstavuje časť v x_i^{l-1}, ktorá bola vynásobená hodnotou $k_{i,j}^{l}$ počas konvolúcie pre výpočet elementu so súradnicami (u, v) na výstupe konvolučnej mapy x_j^l .

Ako už bolo spomínané u subsamplinkovej vrstvy prebieha operácia, ktroje cieľom je zmenšenie vstupných máp. Túto operáciu môžme matematicky zapísať ako:

$$x_j^l = f(\beta_j^l down(x_j^{l-1}) + b_j^l) \tag{2.5}$$

Kde funkcia $down()$ predstavuje operáciu zmenšenia máp. U subsamplingovej vrstvy môžme upravovať a aktualizovať hodnoty pre multiplikatívny prah β a aditívny prah b. V prípade, že za aktuálnou subsamplingovou vrstvou nachádza plne prepojená jenorozmerná vrstva môžeme chybovú funkciu mapy počítať pomocou klasickej metódy spätného šírenia.

V prípade, že za aktuálnou subsamplingovou vrstvou nasleduje ďalšia konvolučná vrstva, musíme zistiť, ktoré pixely aktuálnej vrstvy zodpovedajú pixelom následujúcej vrstvy. V tomto prípade váhy, ktoré násobia spojenia medzi prostredím na vstupe a pixelom na výstupe, zodpovedajú váham (otočeného) konvolučného jadra, čo môžeme popísať nasledujúcim vzťahom:

$$\delta_j^l = f^{'}(u_j^l) \circ conv2(\delta_j^{l+1}, rot180(k_j^{l+1}),' full') \tag{2.6}$$

Následne sme schopný vypočítať gradient pre adaptívny a multiplikatívny prah. Pre aditívny prah b platí opäť vzťah, v ktorom sčítame všetky elemnty chybových máp, ako je popísané v následujúcom vzťahu:

$$\frac{\partial E}{\partial b_j} = \sum_{u,v} (\delta_j^l)_{uv} \tag{2.7}$$

U multiplikatívneho prahu β je nutné použiť orginálnu mapu, ktorá bola vytvorená počas dopredného prechodu sieťou a má tvar:

$$d_j^l = down(x_j^{l-1}) \tag{2.8}$$

Pomocou tejto mapy môžme vypočítať multiplikatívny prah β ako:

$$\frac{\partial E}{\partial \beta_j} = \sum_{u,v} (\delta_j^l \circ d_j^l)_{u,v} \tag{2.9}$$

2.3 Použitie konvolučných neurónových sietí

Praktické použitie konvolučných neurónových sietí v posledných rokoch rastie. V tejto časti práce bude niekoľko ukázaných niekoľko prípadov použitia konvolučných neurónových sietí [6].

2.3.1 LeCunov model konvolučnej siete

Za počiatok konvolučných neurónových sietí sa považuje konvolučná sieť prezentovaná Yannom LeCun-om v roku 1998, ktorá slúžila pre rozpoznávanie rukou písaných číslic. Táto konvolučná sieť sa skladala zo siedmych vrstiev a vstupná vrstva bola tvorená obrázkom s rozmermi 32x32 pixelov. Prvá konvolučná vrstva sa skladala zo 6 príznakových máp, kde každá mapa bola veľkosti 28x28 pixelov a obsahovala 156 parametrov a 122 304 spojení. Za prvou konvolučnou vrstvou sa nachádzala subsamplingová vrstva, ktorá zmenšovala príznakové mapy z konvolučnej vrstvy na veľkosť 14x14 pixelov. Každá jednotka príznakovej mapy zo subsamplingovej vrstvy bola spojená s jednotkami príslušnej príznakovej mapy z konvolučnej vrstvy o veľkosti 2x2 pixelov. Príznaková mapa zo subsamplingovej vrstvy obsahovala 12 parametrov a 5 880 spojení. Druhá konvolučná vrsva bola zložená zo 16 príznakových máp a každá jednotka každej príznakovej mapy bola spojená s množinou jednotiek z príznakových máp predchádzajúcej subsamplingovej vrstvy. Druhá konvolučná vrstva obsahovala 1 516 parametrov a 156 000 spojení. Druhá subsamplingová vrstva následne zmenšovala všetkých 16 príznakových máp na veľkosť 5x5. Tretia konvolučná vrstva sa skladala zo 120 príznakových máp o rozmere 1x1 pixel a každá jednotka bola spojená so všetkými šestnástimi príznakovými mapami z predchádzajúcej vrstvy, čím sa táto vrstva stáva plne prepojenou s počtom parametrov 48 120. Za touto vrstvou sa nachádzala posledná plne prepojená vrstva, ktorá sa skladala z 84 jednotiek. Následujúca výstupná vrstva sa skladala z 10 neurónov, kde každý neurón identifikoval jedno z 10 čísel. Výstupná vrstva sa počítala na základe Euklidovskej radiálnej funkcií (RBF) v tvare:

$$y_i = \sum_j (x_j - w_{ij})^2 \tag{2.10}$$

Pre trénovanie tejto neurónovej siete bol použitý dataset MNIST, ktorý bol spomenutý už v časti 2.1. Po trénovaní siete týmto datasetom dosahovala chybovosť siete hodnotu približne 0,95%. Neskôr sa podarilo znížiť hodnotu chybovosti siete na hodnotu menšiu ako 0,3% pomocou úpravy obrázkov z datasetu prostredníctvom lineárnych kombinácií (rotácia, zmeny merítka, horizontálne zostrihanie) a elastickej deformácie [16].

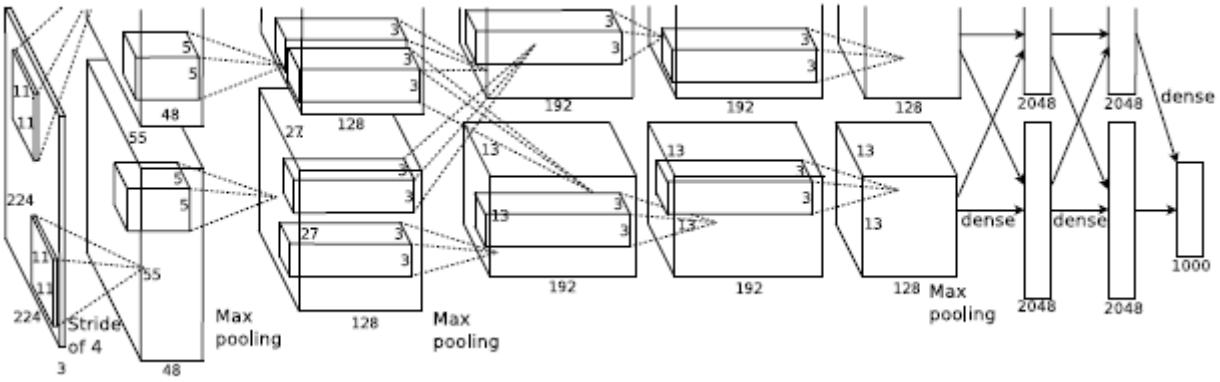
2.3.2 ImageNet a implementácia konvolučnej neurónovej siete pomocou GPU

Okrem datasetu MNIST existujú v dnešnej dobe aj väčšie datasety ako napríklad *LabelMe*, ktorý obsahuje státisíce plne segmentovaných obrázkov. Ďalším typom datasetu je *ImageNet*, ktorý obsahuje 15 miliónov označených obrázkov vo vysokom rozlíšení vo viac ako 22 000 kategóriách. Jednotlivé obrázky z datasetu ImageNet sú zozbierané z internetu a označené pomocou nástroja *Amazon's Mechanical Turk*. ImageNet vznikol v roku 2010 ako súčasť súťaže *ImageNet Large-Scale Visual Recognition Challenge* alebo aj ILSVRC, v ktorej bola využitá podmnožina z ImageNet-u s približne 1,2 miliónmi trénovacích obrázkov, 50 000 validačných obrázkov a 150 000 testovacích obrázkov.

Podľa [12], ktorého autori vytvorili konvolučnú neurónovú sieť, ktorá získala najlepšie výsledky v súťaži ILSVRC-2010 a ILSVRC-2012.

Vytvorená konvolučná neurónová sieť sa skladá z ôsmych vrstiev, kde päť vrstiev je konvolučných a tri vrstvy sú plne prepojené. Výstup poslednej plne prepojenej vrstvy je spojený s tisíc–smernou normalizovanou exponenciálnou funkciou, ktorá poskytuje delenie do viac ako 1 000 tried. Príznakové mapy druhej, štvrtej a piatej konvolučnej vrstvy sú spojené iba s tými mapami predchádzajúcej vrstvy, ktoré sa nachádzajú na rovnakej GPU. Príznakové mapy na tretej konvolučnej vrstve sú spojené so všetkými príznakovými mapami druhej konvolučnej vrstvy. Neuróny v plne prepojených vrstvách sú prepojené zo všetkými neurónmi z predchádzajúcej vrstvy. Vstupom neurónovej siete sú obrázky s rozmermi 224x224x3. Prvá konvolučná vrstva je tvorená 96 príznakovými mapami s veľkosťou 11x11x3 s posunom o 4 pixely. Upravený výstup z prvej konvolučnej vrstvy pokračuje do druhej konvolučnej vrstvy, ktorá obsahuje 256 príznakových máp s veľkosťou 5x5x48. Následujúca tretia konvolučná vrstva, ktorá je prepojená s predchádzajúcou druhou konvolučnou vrstvou obsahuje 384 príznakových máp s veľkosťou 3x3x256. Štvrtá konvolučná vrstva obsahuje 384 príznakových máp s veľkosťou 3x3x192. Piata konvolučná vrstva sa skladá 256 príznakových máp s veľkosťou 3x3x192. U plne prepojených vrstiev obsahuje každá vrstva 4096 neurónov. Schéma uvedenej konvolučnej neurónovej siete je zobrazená na obrázku 2.2, ktorý je rovnako čerpaný z [12].

Uvedená neurónová sieť obsahuje 60 miliónov parametrov, pri takomto veľkom množstve parametrov nastáva situácia, kedy sa síce znižuje hodnota chyby pri trénovaní siete, ale vzrastá pri testovaní, teda sieť stráca schopnosť identifikacie objektu na základe generelizácie. Táto situácia sa nazýva aj „*overfitting*“ a nastáva v situácii, kedy počet parametrov je príliž vysoký oproti trénovacím dátam. Riešením tejto situácie je zväčšenie dát v trénovacej množine. V tomto prípade sú použité dve metódy pre zväčšenie počtu obrázkov v trénovacom datasete. Prvým spôsobom je



Obr. 2.2: Ilustrácia architektúry konvolučnej neurónovej siete

vytvorenie nových obrázkov pomocou transformácie pôvodných a druhou metódou je vytvorenie nových obrázkov pomocou zmeny intenzity farieb RGB v pôvodných obrázkoch.

Z dôvodu redukcie času, kedy trénovanie veľkých neurónových sietí trvalo niekoľko dní bola pri implementácií tejto konvolučnej neurónovej siete uvedená metóda „*dropout*“, ktorá nastavuje nulovú hodnotu pre každý neurón zo skrytej vrstvy s hodnotou pravdepodobnosti 0,5, pretože tieto typy neurónov nemajú vplyv na výpočty pri priechode neurónov sieťou a rovnako nemajú vplyv ani na učenie siete pomocou spätného šírenia chyby. „Dropout“ je použitý v prvých dvoch plne prepojenýc vrstvách siete a jeho použitie má veľký vplyv pri potlačení „overfitting-u“.

Vo výsledku sa implmentovaná sieť skladala zo 650 000 neurónov, 60 000 000 parametrov a 630 000 000 spojení. Táto sieť bola implementovaná na dvoch grafických kartách NVIDIA GTX 580 3GB a trénovanie siete trvalo približne jeden týždeň. V kategórii, kedy sieť vyberala jednu možnosť pre označenie obrázku (Top-1 error) bola hodnota chybobej funkcie 37,5%. Pre informáciu druhý najlepší výsledok v súťaži bol 45,7%. V kategórii, kedy sieť vyberala päť možností pre označenie obrázku, dosiahla chybová funkcia hodnotu 17%, kde druhý najlepší výsledok bol 25,7%.

2.3.3 Spojenie konvolučných neurónových sietí s autoenkódermy pre získavanie hierarchických príznakov

Cieľom spojenia konvolučných neurónových sietí a autoenkóderov je vytvorenie novej architektúry neurónovej siete, ktorá využíva výhody metódy učenia bez učiteľa, ktorá je využitá u enkóderov a aplikuje túto metódu do oblasti 2D obrázkov, kde je kombinovaná s vlastnosťami konvolučných neurónových sietí ako je napríklad zdieľanie váh, ktoré umožňujú zdieľanie priestorovej lokality alebo použia algoritmu spätného šírenia chyby.

Architektúra konvolučných enkóderov vychádza z architektúry enkóderov, ktorá je popísaná v časti 1.4.3, ktorá je doplnená o zdielané váhy. Potom je vstup x pre každú k–tú príznakovú mapu daný ako:

$$h^k = \sigma(x * W^k + b^k) \tag{2.11}$$

Kde je bias rozšírený do celej príznakovej mapy. σ predstavuje aktivačnú funkciu a $*$ predstavuje 2D konvolúciu. Keďže štruktúra siete je založená na princípe enkóderov je nutné, aby sieť bola schopná rekonštrukcie vstupu. Táto funkcia má v tomto prípade tvar:

$$y = \sigma(\sum_{k \in H} h^k * \tilde{W}^k + c) \tag{2.12}$$

Kde je opäť jeden bias c pre celý vstup. H predstavuje skupinu príslušných príznakových máp. $\tilde{W}$ predstavuje opačnú operáciu pre výpočet váh. Konvolúcia v rovniciach 2.11 a 2.12 je určená podľa kontextu, kde pre maticu o veľkosti $m \times m$ a konvolučnú maticu o veľkosti $n \times n$ v prípade plnej konvolúcie platí, že výpočet výslednej matice má tvar:

$$(m + n - 1) \times (m + n - 1) \tag{2.13}$$

V prípade výpočtu valídnej konvolučnej matice je použitý vzťah v tvare:

$$(m - n + 1) \times (m - n + 1) \tag{2.14}$$

V konvolučnom enkóderi je rovnako ako v bežných konvolučných sieťach počítaná funkcia chyby, ktorá má v tomto prípade tvar:

$$E(\theta) = \frac{1}{2n} \sum_{i=1}^{n} (x_i - y_i)^2 \tag{2.15}$$

Rovnako ako pre väčšinu typov neurónových sietí, ktoré pre učenie využívajú algoritmus spätného šírenia chyby, tak aj v tomto prípade je počítaný gradient pre chybovú funkciu. Táto funkcia berie ohľad na jednotlivé parametry, v tomto prípade je jeho hodnota získaná pomocou operácie konvolúcie a tento výpočet má tvar:

$$\frac{\partial E(\theta)}{\partial W^k} = x * \delta h^k + \tilde{h}^k * \delta y \tag{2.16}$$

Ako bolo už spomenuté v časti 3.1 býva v subsamplingových vrstvách konvolučných neurónových sietí použitá funkcia, ktorej cieľom je zmenšenie vstupu používaná najčastejšie funkcia výberu maxima tzv. *max–pooling*. Rovnako aj v pripade konvolučných enkóderov je použitá funkcia *max–pooling*, ktorá však predstavuje preriedenie vnútorných vrstiev siete na základe vymazania hodnôt, ktoré nie sú maximom v

suboblastiach, ktoré sa neprekrývajú. Výhodou tejto funkcie je, že pri následnej rekonštrukcii je možné použiť menšie množstvo filtrov pre dekódovanie každého pixelu.

Implementovaný konvolučný autoenkóder bol otestovaný na datasete MNIST, ktorý bol už spomenutý v časti fig:kon2 a na datasete CIFAR-10, ktorý sa skladá zo 60000 označených farebných obrázkov, kde 50000 obrázkov tvorí trénovaciu množinu a 10000 obrázkov množinu testovaciu. Každý obrázok z datasetu CIFAR-10 má veľkosť 32×32 a tieto obrázky sú rozdelné do 10 kategórií, kde každá kategória obsahuje 6000 obrázkov [17].

Prvý test je aplikovaný na datasete MNIST. V tomto prípade je konvolučný enkóder zložený zo šiestich skrytých vrstiev. Prvá vrstva je konvolučná vrstva, ktorá obsahuje 100 filtrov o veľkosti 5×5. Za touto prvou konvolu4nou vrstvou nasleduje druhá vrstva, ktorá sa skladá z *max–pooling*–ovej vrstvy o veľkosti 2×2. Ďalšia konvolučná vrstva obsahuje 150 filtrov s veľkosťou 5×5. Za touto vrstvou opäť nasleduje vrstva *max–pooling* o veľkosti 2×2. Posledná konvolučná vrstva obsahuje 200 filtrov s veľkosťou 2×2, po ktorej nasleduje posledná plne prepojená vrstva, ktorá sa skladá z 300 neurónov. V poslednej plne prepojenej vrstve je ako aktivačná funkcia použitá normalizovaná exponenciálna funkcia *softmax*. Výsledné hodnoty sú uvedené v tabuľke 2.1, ktorá pre porovnanie obsahuje taktiež hodnoty dosiahnuté pri použití rovnakých dát v „bežnej“ konvolučnej neurónovej siete.

Množstvo trénovacích dát	**1000**	**10000**	**50000**
Hodnota chybovej funkcie – Konvolučný enkóder[%]	7.23	1.88	0.71
Hodnota chybovej funkcie – Konvolučná neurónová sieť[%]	7.63	2.21	0.79

Tab. 2.1: Porovnanie výsledných chybových funkcií pri použití datasetu MNIST

V nasledujúcom teste je konvolučný enkóder aplikovaný na dataset CIFAR-10. V tomto prípade je aplikácia konvolučného enkóderu komplikovanejšia, pretože je nutné spracovať obrázky z datasetu CIFAR-10, ktoré majú ine rozmery ako obrázky z datasetu MNIST. Hodnoty chybovej funkcie získané po spracovaní obrázkov z datasetu CIFAR10 a následnom aplikovaní konvolučného enkóderu sú uvedené v tabuľke 2.2, ktorá rovnako ako 2.1 obsauje porovnanie výsledkov s „bežnou“ konvolučnou neurónovou sieťou.

Podľa získaných hodnôt je vidieť, že konvolučný enkóder dosahuje lepšie výsledky pri zložitejších farebných obrázkoch, kde dosiahnuté hodnoty chybovej funkcie boli lepšie ako u konvolučnej neurónovej siete.

Množstvo trénovacích dát	**1000**	**10000**	**50000**
Hodnota chybovej funkcie – Konvolučný enkóder[%]	52.30	34.35	21.80
Hodnota chybovej funkcie – Konvolučná neurónová sieť[%]	55.52	35.23	22.50

Tab. 2.2: Porovnanie výsledných chybových funkcií pri použití datasetu CIFAR-10

2.3.4 Využitie konvolučných neurónových sietí pri detakcií hudobných príznakov

Na základe [21] sa v súčasnosti konvolučné neurónové siete nepoužívajú už len pri detekcií obrazových vzorov, ale aktuálne aj pri detekovaní príznakov v hudbe, ktoré patrí k základným úlohám v hudobnej analýze. Na základe identifikácie týchto príznakov sú následne analyzované ďalšie vlastnosti ako je napríklad odhad tempa. Identifikácia týchto príznakov v hudbe sa dá prirovnať k identifikácií hrán v obrázkoch, ktoré sú kvalitne detekované prostredníctvom operácie konvolúcie, ktorá je súčasťou konvolučných neurónových sietí.

Učenie siete prebiehalo pomocou častí spektrogramov s binárnymi, na základe ktorých sa sieť naučila rozlišovať, či sa jedná alebo nejedná o príznak. Na rozdiel od obrázkových dát, na ktoré sa aplikujú štvorcové filtre, pri spektogramoch sa viac osvedčili filtre obdĺžnikové. Tento tvar vychádza zo skutočnosti, že sieť hľadá zmeny v čase, preto širšia časť obdĺžnika predstavuje časovú jednotku a užšia časť predstavuje frekvenciu.

Učenie tohto typu konvolučnej neurónovej siete prebieha pomocou zhluku spektrogramov, ktoré majú rôznu velkosť ale rovnakú rýchlosť v počtoch snímkov za sekundu a sú zmenšené na rovnakú veľkosť frekvenčného pásma. Ná základe týchto vlastností každý neurón môže obsahovať informácie o časovej a frekvenčnej presnosti pre danú lokáciu. Pri testovaní siete bol testovací signál najprv prevedený do podoby spektrogramu, ktorý bol privedený na vstup sieti v celku, na rozdiel od učenia siete, kedy boli na vstup privedené iba časti spektrogramu.

K učeniu siete bol použitý dataset, ktorý sa skladal zo 102 minút hudby, ktorá obsahovala 25927 príznakov. Táto nahrávka sa skladala z rôznych typov nahrávok ako napríklad monofónnej, polyfónnej inštrumentálnej časti alebo modernej populárnej hudby. Z tohto záznamu bol vytvorený spektrograf so skokovou veľkosťou 10 ms a oknami s veľkosťami 23 ms, 46 ms a 93 ms. Výsledný vstup siete určený k učeniu siete sa skladal z 3 spektrogramov zložených z 15 snímkov na 80 pásiem. Na vstupnú vrstvu bola aplikovaná konvolučná vrstva, ktorá sa skladala z filtrov zložených zo 7 snímkov na 3 pásma. Výstupom konvolučnej vrstvy bolo 10 príznakových máp zložených z 9 snímkov na 78 pásiem. Ďalšou vrstvou bola vrstva „*max–pooling*“,

ktorej cieľom bolo zredukovať veľkosť máp na 26 pásiem. Po „*max–poolingovej*“ vrstve následovala ďalšia konvolučná vrstva vrstva s veľkosťou filtru 3×3 a ďalšia „*max–poolingová*“ vrstva, ktorej výsledkom bolo 20 máp zložených zo 7 snímok na 8 pásiem. Tieto príznakové mapy boli spracované do jednej plne prepojenej vrstvy, ktorá sa skladala z 256 neurónov, ktoré končili v poslednej plne prepojenej vrstve, ktorej výstupom bola jediná jednotka, ktorá určovala hudobný príznak. V sieti bolo použitých niekoľko aktivačných funkcií. U konvolučných vrstiev bola ako aktivačná funkcia použitý tangens. U plne prepojených vrstiev bola zasa ako aktivačná funkcia použitý logický sigmoid.

V záverečnom testovaní dosiahla konvolučná neurónová sieť úspešnosť okolo 88.5%. Pre zvýšenie výkonnosti bolo na konvolučnú neurónovú sieť aplikovaných niekoľko dodatočných metód. Prvou metódou pre zvýšenie výkonu konvolučnej siete bol tzv. „*Dropout*“, ktorého princíp bol spomínaný v časti 2.3.2. Pri použití dropoutu dosiahla úspešnosť siete hodnotu 89%. Ďalšou metódou použitou pre zlepšenie výkonu siete bolo použitie menej jasných príkladov pre učenie siete, čo znamená, že sieť nebola učená presne definovanými vzormi. Napríklad sa pri učení siete používalo niekoľko blízkych spektrografických snímkov zlúčených do jedného. Výsledkom tohto spôsobu učenia siete dosiahla presnosť konvolučnej neurónovej siete hodnotu 89.9%. Na záver bola pre zvýšenie výkonu konvolučnej neurónovej siete vymenená aktivačná funkcia v konvolučných vrstvách, kde bol tangens nahradený lineárnou funkciou $y(x) = max(0, x)$, ktorá zdvihla úspešnosť siete na 90.3% [15].

2.3.5 Rozpoznávanie čísel z Google Street View pomocou konvolučných neurónových sietí

Výhody konvolučných neurónových sietí sa rozhodla využiť taktiež spoločnosť Google, ktorá pomocou konvolučných neurónových sietí implementovala rozlišovanie čísel budov z fotografií, ktoré boli získané prostredníctvom Street View. Použitie konvolučných neurónových sietí v tomto prípade bolo vhodné, pretože tento typ neurónovej siete pracuje priamo s pixelmi obrázkov a nie je nutné použitie zložitých predspracovaní obrázkov.

Cieľom tohto projektu nebola len identifikácia čísel budov, ale aj následné označenie budovy prostredníctvom čísla na mape, čím sa spoločnosť Google snažila eliminovať nesprávne označenia budov na mape. Pre učenie siete bol na začiatku použitý dataset, ktorý sa skladal z 200000 obrázkov čísel domov. Okrem týchto obrázkov dataset navyše obsahoval 600000 číselných vzorov. Sieť je navyše obmedzená pre identifikáciu čísel, ktoré sa skladajú maximálne z 5 cifier, kde sa vychádza zo skutočnosti, že existuje len málo čísel domov s viac ako piatimi ciframi [26].

Výsledná architektúra konvolučnej neurónovej siete, ktorá dosahovala najlepšie výsledky sa skladala z 11 skrytých konvolučných vrstiev, z jednej čiastočne prepojenej vrstvy a 2 plne prepojených vrstiev. V prvej skrytej vrstve siete bola tzv. „*maxout*“ aktivačná funkcia, ktorá vykonáva jednu operáciu (najčastejšie výber maxima) cez niekoľko vrstiev podpriestoru, ktorý je reprezentovaný výstupmi z transformácií vstupu. Táto operácia je podobná priestorovej operácii „*max–pooling*“ a matematicky sa vyjadruje ako:

$$g_i(f_i) = max\{f_{i,1}, ..., f_{i,k}\}, kde\ f_i = [f_{i,1}, ..., f_{i,k}] \in R^k \tag{2.17}$$

V ďalších konvolučných vrstvách je ako aktivačná funkcia použitá usmerňovacia funkcia, ktorá je narozdiel od „*maxout*“ funkcie jednoduchšia, nepracuje s niekoľkými vrstvami a matematicky ju môžeme zapíať ako:

$$g_i(f_i) = max\{0, f_i\}, kde\ f_i \in R\ a\ k = 1 \tag{2.18}$$

Každá konvolučná vrstva sa skladala z konvolučných jadier, ktoré mali rovnaký rozmer 5×5. Za každou konvolučnou vrstvou sa nachádzala „*max–pooling*“ vrstva, ktorej veľkosť bola 2×2. Počas učenia siete pre dosiahnutie lepších výsledkom bola ako v predchádzajúcich prípadoch použitá metóda „*dropout*“.

V súčastnosti bola táto konvolučná neurónová sieť aplikovaná celosvetovo približne na 100 miliónov fyzických čísel domov a úspešnosť identifikácie čísel aktuálne dosahuje približne 96% [25].

3 NÁVRH A IMPLEMENTÁCIA PROGRAMU

Cieľom práce je nájsť a vybrať vhodnú metódu pre implementáciu konvolučných neurónových sietí a následne túto metódu implementovať v programovacom jazyku Java. V tejto kapitole budú postupne rozobrané návrhy konvolučných neurónových sietí, ktoré by bolo vhodné implementovať. Kapitola bude taktiež obsahovať techniky a rôzne možnosti implementácie, ktoré v súčastnosti ponúka jazyk Java. Na záver kapitoly bude popísaná implementácia, ktorá bola použitá v rámci tejto diplomovej práce.

3.1 Návrh konvolučnej neurónovej siete

V tejto časti kaitoly sa budem zaoberať návrhmi konvolučnej neurónovej siete, ktoré bude v rámci tejto diplomovej práce imlementované a získané výsledky budú následne zhodnotené.

3.1.1 Vstupné dáta

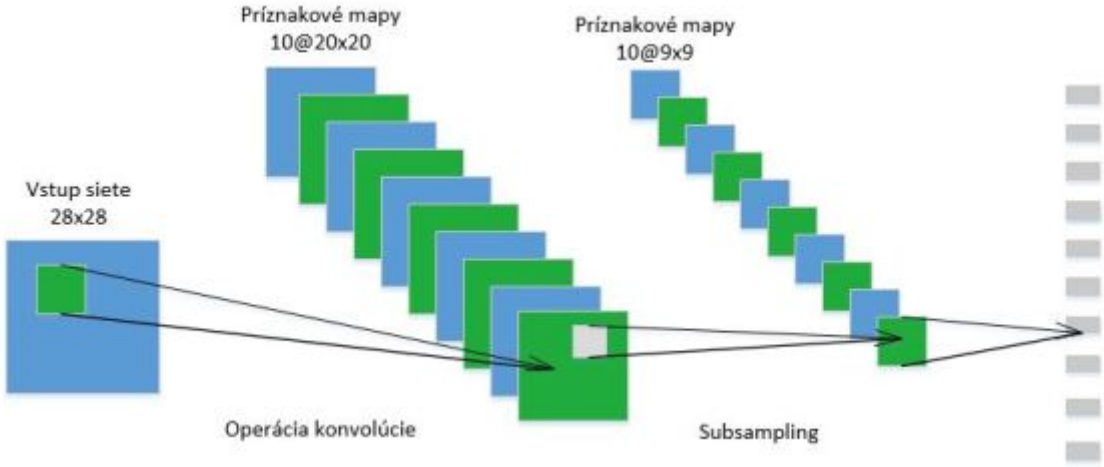
Ako už bolo spomenuté konvolučné neurónové siete sú učené tzv. metódou s učiteľom, teda je nutné najprv konvolučnej neurónovej sieti predložiť dáta na základe ktorých bude sieť učená. V rámci tejto diplomovej práce som sa rozhodla použiť už existujuce dáta pre učenie siete. Konkrétne ide o dataset MNIST, ktorý bol spomínaný už v predchádzajúcich častiach.

MNIST je databáza, ktorá vznikla modifikáciou databázy zloženej zo vzorov ručne písaných symbolov, ktoré boli sprostredkované firmou *National Institute of Standards and Technology - NIST*. Dataset MNIST sa skladá z dvoch oddelených setov, kde prvý slúži pre učenie neurónových sietí a je zložený z 60000 čiernobielych obrázkov ručne písaných číslic. Druhý set slúži pre testovanie neurónovej siete a je zložený z 10000 čiernobielych obrázkov ručne písaných číslic. Set určený pre učenie siete sa skladá zo vzorov, ktoré tvorilo okolo 250 pisateľov.

Dataset MNIST je voľne k dispozicií na oficiálnych stránkach, ktoré sú spomenuté tu [11]. K stiahnutiu sú k dispozícií štyri súbory. Prvým súborom je *„train-images-idx3-ubyte.gz“*, ktorý obsahuje vzory pre trénovanie neurónovej siete. Ďalším súborom je *„tran-labels-idx1-ubyte.gz“*, ktorý je zložený z tzv. *labelov*, ktoré zodpovedajú jednotlivým trénovacím vzorom a určujú číselnú hodnotu, ktorú daný trénovací vzor reprezentuje. Tretím súborom je *„t10k-images-idx3-ubyte.gz“*, ktorý je zložený z testovacích vzorov. Posledným súborom je *„t10k-labels-idx1-ubyte.gz“*, ktorý je opäť zložený z *labelov*, ktoré sú v tomto prípade príslušné k testovacím vzorom a umožňujú určiť, či výstupy neurónovej siete zodpovedajú očakávaným výsledkom.

3.1.2 Návrh architektúry konvolučnej neurónovej siete

V rámci tejto práce som sa rozhodla vytvoriť niekoľko rôznych architektúr konvolučných neurónových sietí, na ktoré budú aplikované uvedené vstupné dáta. Výsledky výstupov jednotlivých sietí budú uvedené a porovnané v následujúcej kapitole.



Obr. 3.1: Ilustračný model prvej konvolučnej neurónovej siete

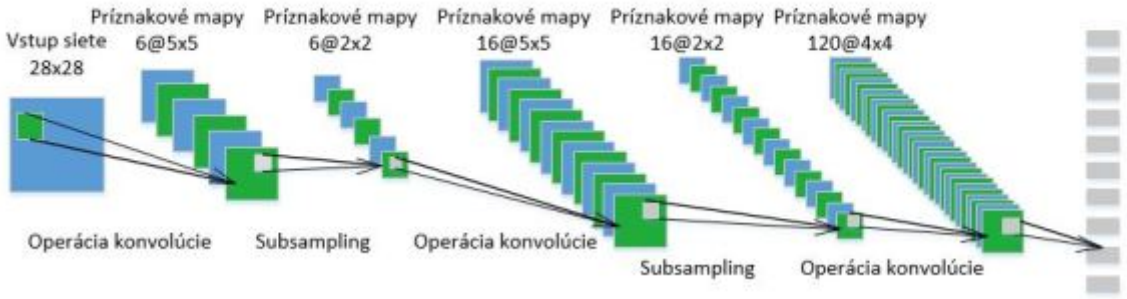
Ako prvú konvolučnú neurónovú sieť som sa rozhodla vytvoriť jednoduchú konvolučnú neurónovú sieť. Ilustračný model siete je vidieť na obrázku 3.1. Ako je vidieť vstupom siete je obrázok z MNIST databázy o veľkosti 28×28, ktorá predstavuje 784 neurónov. Následujúcou vrstvou je vrstva konvolučná, ktorá sa skladá z desiatich recepčných polí, ktoré majú rozmer 20×20. Výstupom tejto konvolučnej vrstvy je desať príznakových máp o rozmere 9×9. Táto vrstva je zložená z $9 \times 9 \times 10 = 810$ neurónov, $(20 \times 20 + 1) \times 10 = 4010$ zdielaných váh, kde hodnota $+1$ predstavuje bias a $810 \times (20 \times 20 + 1(bias)) = 324810$ spojení s predchádzajúcou vstupnou vrstvou. Následujúcou vrstvou je vrstva subsamplingová, ktorá počíta operáciu „max–pooling“ z príznakových máp, ktoré sú výstupom predchádzajúcej konvolučnej vrstvy. V tomto prípade je na príznakové mapy konvolučnej vrstvy aplikovaná subsamplingová vrstva o veľkosti 9×9, ktorá zmenší tieto príznakové mapy na veľkosť 1×1. Výsledkom aplikácie subsamplingovej vrstvy je opäť desať príznakových máp, tentokrát o rozmere 1×1 a sú teda zložené z 10 neurónov. Následne je na sieť privedený jednorozmerný vektor, ktorý predstavuje poslednú plne prepojenú vrstvu, veľkosť tejto vrstvy je 10 a obsahuje $10 \times 10 = 100$ spojení. Táto vrstva predstavuje výstup konvolučnej neurónovej siete, tým že určuje jednu z možných hodnôt, ktoré môže sieť nadobúdať.

Prehľad jednotlivých vrstiev konvolučnej neurónovej siete je vidieť v tabuľke 3.1.

Typ vrstvy siete	Počet príznakových máp / neurónov	Veľkosť recepčného poľa	Veľkost príznakovej mapy
Vstupná	1 / 784	-	28 x 28
Konvolučná	10 / 810	20 x 20	9 x 9
Subsamplingová	10 / 10	9 x 9	1 x 1
Výstupná	10 neurónov	-	-

Tab. 3.1: Prehľad rozmerov jednotlivých vrstiev prvej konvolučnej neurónovej siete

Ilustračný model architektúri druhej konvolučnej neurónovej siete sa nachádza na obrázku 3.2. Ako je možno vidieť, ako druhý model konvolučnej neurónovej siete som zvolila sieť, ktorá je zložená z troch konvolučných a dvoch subsamplingových vrstiev. Vstupná vrstva rovnako ako v predchádzajúcom prípade je tvorená obrázkom



Obr. 3.2: Ilustračný model druhej konvolučnej neurónovej siete

veľkosti 28×28. Za touto vstupnou vrstvou sa nachádza prvá konvolučná vrstva, ktorá je tentokrát tvorená šiestimi recepčnými poľami o rozmere 5×5. Výstupom prvej konvolučnej vrstvy je šesť príznakových máp s rozmerom 24×24. Z hľadiska veľkosti je prvá konvolučná vrstva zložená z $24 \times 24 \times 6 = 3456$ neurónov a obsahuje $(5 \times 5 + 1(bias)) \times 6 = 156$ zdieľaných váh a $3456 \times (5 \times 5 + 1(bias)) = 89856$ spojení so vstupnou vrstvou.

Za prvou konvolučnou vrstvou sa nachádza prvá subsamplingová vrstva, ktorá opäť vykonáva operáciu „max–pooling“ na príznakových mapách z prvej konvolučnej vrstvy. Veľkosť subsamplingovej mapy je v tomto prípade 2×2 a jej výsledkom je šesť zmenšených príznakových máp, ktorých rozmer je 12×12.

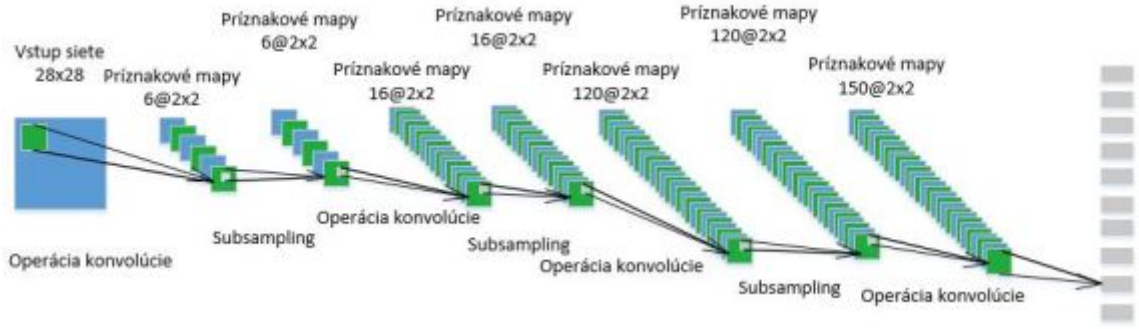
Po prvej subsamplingovej vrstve nasleduje ďalšia konvolučná vrstva. Táto druhá konvolučná vrstva je zložená z zo šestnástich recepčných polí, ktorých veľkosť je opäť 5×5. Výstupom druhej konvolučnej vrstvy je šestnásť príznakových máp, ktoré majú veľkosť 8×8 a spolu obsahujú $8 \times 8 \times 16 = 1024$ neurónov a $(5 \times 5 + 1(bias)) \times 6 \times 16 = 2496$ zdieľaných váh a $1024 \times (5 \times 5 + 1(bias)) = 26624$ spojení s predchádzajúcou

vrstvou.
Za druhou konvolučnou vrstvou nasleduje druhá a zároveň posledná subsamplingová vrstva, ktorá rovnako ako prvá subsamplingová vrstva zmenšuje príznakové mapy na polovicu pomocou filtru s rozmermi 2×2 s operáciou „max–pooling“. Výsledkom tejto vrstvy je šestnásť príznakových máp, ktorých rozmer je 4×4.
Na týchto šestnásť príznakových máp je následne aplikovaná posledná tretia konvolučná vrstva, ktorá je v tomto prípade zložená zo 120 recepčných polí s rozmerom 4×4. Výstupom tejto vrstvy je 120 príznakových máp s rozmerom 1×1, a teda veľkosť tejto vrstvy je 120 neurónov a obsahuje $(4 \times 4 + 1(bias)) \times 6 \times 16 \times 120 = 195840$ zdieľaných váh a $120 \times (4 \times 4 + 1(bias)) = 2040$ spojení s predchádzajúcou vrstvou.
Poslednou vrstvou je opäť plne prepojená vrstva zložená z 10 neurónov. V tejto plne prepojenej vrstve sú opäť všetky neuróny prepojené so všetkými 120 neurónmi z predchádzajúcej vrstvy a teda táto vrstva obsahuje a $(120 + 1(bias)) \times 10 = 1210$ spojení. Prehľad jednotlivých vrstiev siete je možné vidiet v tabuľke 3.2

Typ vrstvy siete	Počet príznakových máp / neurónov	Veľkosť recepčného poľa	Veľkost príznakovej mapy
Vstupná	1 / 784	-	28 x 28
1. Konvolučná	6 / 3456	5 x 5	24 x 24
1. Subsamplingová	6 / 864	2 x 2	12 x 12
2. Konvolučná	16 / 1024	5 x 5	8 x 8
2. Subsamplingová	16 / 256	2 x 2	4 x 4
3. Konvolučná	120 / 120	4 x 4	1 x 1
Výstupná	10 neurónov	-	-

Tab. 3.2: Prehľad rozmerov jednotlivých vrstiev druhej konvolučnej neurónovej siete

Ako tretí typ architektúry konvolučnej neurónovej siete som sa rozhodla vytvoriť konvolučnú neurónovú sieť, ktorá sa skladá zo štyroch konvolučných a troch subsamplingových vrstiev. Ilustračný model tejto konvolučnej neurónovej siete je možné vidieť na obrázku 3.3. Vstupom siete je rovnako ako v predchádzajúcich prípadoch obrázok z databáze MNIST s veľkosťou 28×28. Za touto vstupnou vrstvou nasleduje prvá konvolučná vrstva, ktorá je zložená zo šiestich recepčných polí, ktorých veľkoť je 2×2. Výstupom prvej konvolučnej vrstvy je šesť príznakových máp, ktorých rozmery sú 27×27. Počet neurónov v tejto vrstve je $27 \times 27 \times 6 = 4374$, počet zdielaných váh je $(2 \times 2 + 1(bias)) \times 6 = 30$ a počet spojení tejto vrstvy so vstupnou



Obr. 3.3: Ilustračný model tretej konvolučnej neurónovej siete

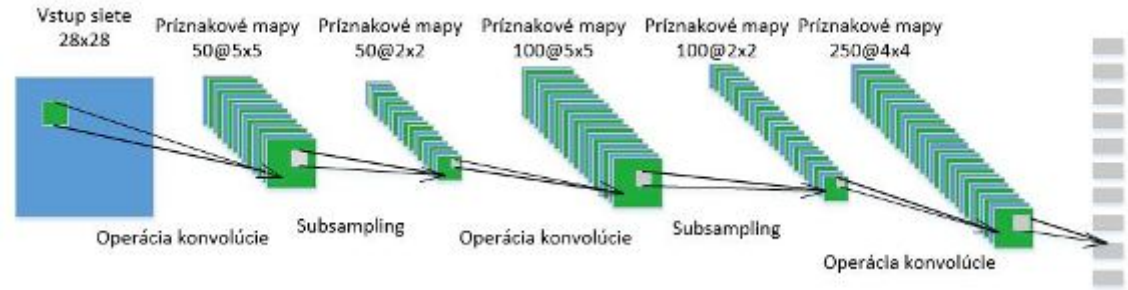
vrstvou je $729 \times (2 \times 2 + 1(bias)) = 3645$.
Za prvou konvolučnou vrstvou nasleduje prvá subsamplingová vrstva, ktorá opäť pomocou operácie „max–pooling“ zmenšuje príznakové mapy, ktoré sú výstupom predchádzajúcej prvej konvolučnej vrstvy. V tejto vrstve je použitý filter o veľkosti 2×2, ktorý zmenšúje príznakové mapy na veľkosť 13×13.
Za prvou subsamplingovou vrstvou sa nachádza druhá konvolučná vrstva, ktorá je zložená zo šestnástich recepčných polí, ktorých veľkosť je opäť 2×2. Výstupom tejto vrstvy je šestnásť príznakových máp, ktorých rozmer je v tomto prípade 12×12. Táto vrstva je zložená z $12 \times 12 \times 16 = 2304$ neurónov a obsahuje $(2 \times 2 + 1(bias)) \times 16 \times 6 = 480$ zdieľaných váh a $2304 \times (2 \times 2 + 1(bias)) = 11520$ spojení s predchádzajúcou vrstvou.
Následujúcou vrstvou je druhá subsamplingová vrstva, v ktorá opäť pomocou filtru s veľkosťou 2×2 zmenšuje príznakové mapy z druhej konvolučnej vrstvy na veľkosť 6×6.
Tretia konvolučná vrstva, ktorá sa nachádza za druhou subsamplingovou je opäť zložená z recepčných polí s veľkosťou 2×2. Tentokrát je však počet recepčných polí rovný hodnote 120. Výstupom tretej konvolučnej vrstvy je teda 120 príznakových máp, ktorých veľkosť je 5×5. V tomto prípade je potom počet neurónov v tejto vrstve rovný $5 \times 5 \times 120 = 3000$, počet zdieľaných váh v tejto vrstve dosahuje hodnotu $(2 \times 2 + 1(bias)) \times 6 \times 16 \times 120 = 57600$ a počet spojení s predchádzajúcou vrstvou je $3000 \times (2 \times 2 + 1(bias)) = 15000$.
Za treťou konvolučnou vrstvou sa nachádza posledná tretia subsamplingová vrstva, ktorá opäť zmenšuje príznakové mapy z predchádzajúcej vrstvy filtrom s veľkosťou 2×2 na príznakové mapy s veľkosťou 2×2.
Posledná štvrtá konvolučná vrstva je zložená zo 150 recepčných polí s veľkosťou 2×2. Výstupom tejto vrstvy je 150 príznakových máp s veľkosťou 1×1. Táto vrstva je zložená zo $1 \times 1 \times 150 = 150$ neurónov a obsahuje $(2 \times 2 + 1(bias)) \times 6 \times 16 \times 120 \times 150 = 8640000$ zdieľaných váh a $150 \times (2 \times 2 + 1(bias)) = 750$ spojení s predchádzajúcou

vrstvou.
Poslednou vrstvou je opäť plne prepojená vrstva, ktorá sa skladá z 10 neurónov, kde rovnako ako v predchádzajúcich prípadoch každý neurón z tejto vrstvy reprezenuje jedno číslo od 0 do 9. V tejto vrstve je opäť každý neurón prepojený s každým neurónom z predchádzajúcej vrstvy, preto počet spojení v tejto vrstve má hodnotu $10 \times (150 + 1(bias)) = 1510$.
Prehľad jednotlivých vrstiev tohto modelu konvolučnej neuróneovej siete je uvedený v tabuľke 3.3.

Typ vrstvy siete	Počet príznakových máp / neurónov	Veľkosť recepčného poľa	Veľkost príznakovej mapy
Vstupná	1 / 784	-	28 x 28
1. Konvolučná	6 / 4374	2 x 2	27 x 27
1. Subsamplingová	6 / 1014	2 x 2	13 x 13
2. Konvolučná	16 / 2304	2 x 2	12 x 12
2. Subsamplingová	16 / 576	2 x 2	6 x 6
3. Konvolučná	120 / 3000	2 x 2	5 x 5
3. Subsamplingová	120 / 720	2 x 2	2 x 2
4. Konvolučná	150 / 150	2 x 2	1 x 1
Výstupná	10 neurónov	-	-

Tab. 3.3: Prehľad rozmerov jednotlivých vrstiev tretej konvolučnej neurónovej siete

Posledným typom architektúry konvolučnej siete je konvolučná neurónová sieť, ktorá sa skladá podobne ako konvolučná sieť v druhom prípade z troch konvolučných a dvoch subsamplingových vrstiev. Ilustračný model tejto siete je vidieť na obrázku 3.4
Rovnako aj v tomto prípade je ako vstupná vrstva siete použitý obrazok s veľkosťou 28×28, za touto vrstvou sa nachádza prvá konvolučná vrstva, ktorá je zložená z 50 recepčných polí, ktorých veľkosť je 5×5. Výstupom prvej konvolučnej vrstvy je 50 príznakových máp, ktorých veľkosť je 24×24. Počet neurónov v tejto vrstve má hodnotu $24 \times 24 \times 50 = 28800$, počet zdieľaných váh v tejto vrstve je rovný hodnote $(5 \times 5 + 1(bias)) \times 50 = 1300$ a počet spojení s predchádzajúcou vstupnou vrstvou je $28800 \times (5 \times 5 + 1(bias)) = 748800$. Ďalľou vrstvou je prvá subsamplingová vrstva, ktorá pomocou operácie „max–pooling“a filtrom s veľkosťou 2×2 zmenšuje veľkosť príznakových máp z prvej konvolučnej vrstvy na hodnotu 12×12.
Po prvej subsamplingovej vrstve nasleduje druhá konvolučná vrstva, ktorá sa v tomto



Obr. 3.4: Ilustračný model štvrtej konvolučnej neurónovej siete

prípade skaldá zo 100 recepčných polí, ktorých veľkosť je opäť 5×5. Výstupom druhej konvolučnej vrstvy je 100 príznakových máp, ktoré majú veľkosť 8×8. Táto vrstva sa skladá z $8\times8\times100 = 6400$ neurónov a obsahuje $(5\times5+1(bias))\times50\times100 = 130000$ zdieľaných váh a $6400\times(5\times5+1(bias))$ spojení s predchádzajúcou vrstvou.
Druhá subsamplingová vrstva, ktorá nasleduje po druhej konvolučnej vrstve zmenšuje výstup tejto vrstvy opäť pomocou filtru s veľkosťou 2×2 na mapy s veľkosťou 4×4.
Posledná tretia konvolučná vrstva sa skladá z 250 recepčných polí s veľkosťou 4×4. Výstupom tejto vrsty je 250 príznakových máp s veľkosťou 1×1. Počet neurónov v tejto vrstve je $1\times1\times250 = 250$, počet zdieľaných váh v tejto vrstve má hodnotu $(4\times4+1(bias))\times50\times100\times250 = 21250000$ a počet spojení s predchádzajúcou vrstvou je $250\times(4\times4+1(bias)) = 4250$. Poslednou výstupnou vrstvou je opäť plne prepojená vrstva, ktorá sa skladá z 10 neurónov. V tomto prípade je všetkých 10 neurónov výstupnej vrstvy prepojených so všetkými 250 neurónmi z predchádzajúcej vrstvy, čo predstavuje 2500 spojení.
Prehľad jednotlivých vrstiev tohto typu konvolučnej neurónovej siete je vidieť v tabuľke 3.4

Typ vrstvy siete	Počet príznakových máp / neurónov	Veľkosť recepčného poľa	Veľkost príznakovej mapy
Vstupná	1 / 784	-	28 x 28
1. Konvolučná	50 / 28800	5 x 5	24 x 24
1. Subsamplingová	50 / 7200	2 x 2	12 x 12
2. Konvolučná	100 / 6400	5 x 5	8 x 8
2. Subsamplingová	100 / 1600	2 x 2	4 x 4
3. Konvolučná	250 / 250	4 x 4	1 x 1
Výstupná	10 neurónov	-	-

Tab. 3.4: Prehľad rozmerov jednotlivých vrstiev štvrtej konvolučnej neurónovej siete

3.2 Implementácia konvolučnej neurónovej siete

V tejto časti práce bude postupne popísaná implementácia konvolučnýc neurónových sietí, ktoré boli popísané v predchádzajúcej časti.

3.2.1 Dostupné frameworky pre implementáciu neurónových sietí v jazyku Java

Pred samotnou implementáciou som sa rozhodla najprv vybrať vhodný framework, ktorý bude pri implementácií použitý. Pre jazyk Java bolo vytvorených niekoľko frameworkov, ktoré sú určené pre implementáciu neurónových sietí. V tejto časti práce bude spomenutých niekoľko z nich a na záver bude vybraný jeden, ktorý bude použitý pri implementácií.

a) Encog

Prvým spomenutým je open source framework *Encog*, ktorý je okrem Javy implementovaný aj pre jazyk C# a je vyvíjaný od roku 2008. Encog je zameraný na implementáciu strojového učenia a okrem podpory implementácie rôznych druhov neurónových sietí, taktiež podporuje algoritmy pre SVM (Support Vector Machine), generické programovanie alebo Hidden Markov model. Encog taktiež umožňuje použitie GPU pri implementácií pre zrýchlenie jednotlivých operácií a procesového času. K dispozícií je taktiež grafický workbench, pre pomoc pri tvorbe jednotlivých modelov a ich trénovania [7].

b) JOONE

JOONE alebo aj *Java Object Oriented Neural Engine* je rovnako ako Encog open source framework primárne určený pre implementáciu neurónových sietí, ktorý je k dispozícií od roku 2005. Architektúra JOONE je založená na komponentoch, ktoré je možno znovu použiť, spájať ich s inými komponentami, prípadne vkladať do rôznych projektov. Rovnako ako u Encog-u je aj u JOONE k dispozícií grafický workbench pre tvobu jednoduchú tvorbu modelov a ich trénovanie. Nevýhodou JOONE oproti Encog-u je, že JOONE je viac komplexný a v súčastnosti už nie je aktívne podporovaný, čím sa stáva nevhodný pre použitie v moderných technológiách [10].

c) Snipe

Ďalším spomenutým frameworkom je *Snipe*, ktorý je zdarma iba v prípade nekomerčných účelov. Snipe bol vytvorený D. Krieselom a jeho pôvodným cieľom bolo vytvoriť vysoko výkonné simulácie s veľkým množstvom neurónpvých sietí, ktoré by

boli trénované súčasne. Snipe je určený pre profesionálnu implementáciu neurónových sietí a v súčastnosti nepodporuje žiadny typ grafického rozhrania pre tvorbu modelov neurónových sietí . Množstvo dokumentácie a rozšírenosť Snipe nie je aktuálne veľká, preto ani použitie tohto frameworku nie je príliž vhodné [8].

d) Neuroph

Posledným spomenutým frameworkom je *Neuroph*, ktorý vznikol v roku 2008 ako diplomová práca na univerzite v Belehrade a jeho autorom je Zoran Sevarac. Neuroph sa skladá z knižníc, ktoré zodpovedajú základným konceptom neurónových sietí. Príkladom podporovaných architektúr neurónových sietí v Neuroph-e sú napríklad Perceptron, Viacvrstvový perceptron s učením spätného šírenia, Hopfieldová sieť, Hebbianova, prípadne Kohonenova sieť. Súčasťou Neuroph-u je taktiež grafický nástroj, v ktorom je možné tieto siete jednoducho vytvoriť, trénovať alebo testovať. Aktuálna verzia *Neuoroph 2.9* podporuje ako novinku niekoľko funkcií pre prácu s konvolučnými neurónovými sieťami, bohužiaľ zatiaľ nie v rámci grafického nástroja [9].

3.2.2 Možnosti využitia grafických jednotiek pri implementácií v jazyku Java

V súčastnosti množstvo programovacích jazykov ponúka okrem bežného programovania, kedy program beží na procesore (CPU), taktiež možnosť spúšťania programov alebo častí programov pomocou grafických jednotiek (GPU). Samotná GPU sa typicky skladá z množstva procesorov, ktoré samy o sebe nie sú extrémne výkonné, ale uplatňuje sa tu paralelizmus, kedy niekoľko rôznych operacií môže bežať v rovnakom čase na rôznych procesoroch. Využitie GPU má teda zmysel v prípadoch, kedy program strávi množstvo času výpočtom zložitých operácií, kedy použitie GPU umožňuje zrýchlenie tohto procesu pomocou využitia paralelných operácií.

Éra programovateľných grafických jednotiek začala až koncom 90. rokov 20. storočia, pred tým boli GPU využívané ako grafické akcelerátory, ktoré podporovali výpočty len určitých špecifických funkcií. V súčastnosti je možné použiť dva spôsoby programovania s využitím GPU, ktorými sú *OpenCL* a *CUDA*.

a) OpenCL

OpenCL alebo *Open Computing Language* je prvým spôsobom využitia GPU pri implementácií, je to prvý otvorený štandard pre multi–platformové paralelné programovanie. OpenCL je vytvorené spoločnosťou Khronos Group v roku 2008. Ako už

bolo spominané OpenCL umožňuje multi-patformové programovanie, čo znamená, že program napísaný v OpenCL može byť spustený na procesoroch (CPU), grafických jednotkách (GPU), programovateľných hradlách (FPGA), prípadne inom hardvéri. OpenCL poskytuje paralelné programovanie a v súčastnosti je dostupný pre produkty firiem ako AMD, Apple, IBM, Intel, Nvidia alebo Samsung.

OpenCL sa skladá z niekoľkých zdieľaných objektov, ktoré umožňujú spúšťanie výpočtov, pričom inštalačný driver musí byť nainštalovaný na každej platforme, aby OpenCL bolo možné túto platformu identifikovať. Výhodou použitia OpenCL je čiastočná prenosnosť programu pomocou zdieľanej pamäte, ktorá umožňuje programátorovi spustiť program na podobných platformách, ktoré nemusia byť hardvérovo úplne rovnaké [30].

OpenCL je zložený z knižnice implementovanej v jazykoch C a C++ a prekladača, ktorý umožňuje spúšťanie výpočtov na jednotlivých platformách. Primárne použitie OpenCL je teda s jazykom C++. Pre použitie v iných programovacích jazykov ako je napríklad Python alebo Java existujú rôzne API knižnice tretích strán, konkrétne pre Javu sú to napríklad:

a.1) Aparapi Aparapi je knižnica, ktorá umožňuje vývojárom využívať výhody grafických jednotiek (GPU) a akcelerovaných procesorových jednotiek (APU) namiesto procesorových jednotiek (CPU) pri spúšťaní paralelných operácií v programe. Aparapi pracuje tak, že za behu programu konvertuje javovský bytekód do OpenCL a následne ho spúšťa pomocou GPU. V prípade, že Aparapi nie je schopná spustiť operácie na GPU, sú tieto operácie spúšťané pomocou pracovných vlákien tzv. *Java Thread Pool* (JTP). Výhodou Aparapi je, že nie je závislá na výrobcoch grafických kariet, pretože pracuje s OpenCL. Pôvodne bola vyvinutá firmou AMD, ale v súčastnosti je licencovaná ako *open-source* [31].

a.2) JOCL Knižnica JOCL rovnako ako Aparapi pracuje s OpenCL princípom, teda umožňuje aplikáciam bežiacich na javovskom virtuálnom stroji (JVM) využívať funkcie OpenCL ako paralelizmus, vyšší výkon a spúšťanie na rôznych platformách ako sú okrem CPU aj GPU, FPGA. Knižnica JOCL je zložená zo statických metód, ktoré sémanticky zodpovedajú funkciám z OpenCL, len sú syntakticky prispôsobené jazyku Java [33].

b) CUDA

CUDA alebo Parallel Computing Platform je platforma a programovací model vytvorený firmou NVIDIA. CUDA je implementovaná pomocou grafických jednotiek

(GPU) a umožňuje vývojárom priamy prístup k výpočetným jednotkám ako sú napríklad virtuálne inštrukcie alebo pamäť priamo v CUDA grafických jednotkách. Z pohľadu vývojára je CUDA dostupná ako rozšírenie jazykov C/C++ alebo Fortran, kde je kompilovaná pomocou príkazu „*nvcc*“. V prípade jazyka Java je možné implementovať CUDA grafické jednotky pomocou niekoľkých knižníc ako sú napríklad *Rootbeer* alebo *JCUDA*.

Výhodou CUDA je napríklad zdieľaná pamäť medzi jednotlivými vláknami alebo možnosť čítania dát z rôznych častí pamäte. Najväčšou nevýhodou je, že program funguje len pre grafické karty, ktoré túto platformu obsahujú, čo sú grafické karty vyvíjané firmou NVIDIA. Problémom je taktiež spolupráca s inými jazykmi ako je napríklad OpenGL, kde OpenGL má prístup do registrovanej CUDA pamäte, ale CUDA nemá prístup do OpenGL pamäte [35].

b.1) Rootbeer Knižnica Rootbeer umožňuje užívateľovi spustiť program implementovaný v jazyku Java na grafickej jednotke CUDA spoločnosti NVIDIA. Rootbeer funguje tak, že javovský bytekód je prevedený do CUDA programovacieho jazyka, z ktorého je spúšťaný na grafickej jednotke. Autorom tejto knižnice je Phil Szeliga zo Syrakúzskej univerzity, ktorého cieľom bolo vytvorenie knižnice, ktorá umožňuje využitie vysokovýkonných grafických jednotiek pri implementácií v jazyku Java s minimálnym zásahom developera do procesov v grafických jednotkách a ich operovaním [36].

b.2) JCUDA JCUDA je primárne určená pre interakciu s inými CUDA knižnicami, respektívne umožňuje užívateľovi volať v jazyka Java CUDA funkcie napísané v jazykoch C alebo C++. JCUDA teda užívateľovi neumožňuje vytvorenie vlastného CUDA jadra a následne jeho spracovanie, v tomto prípade je vhodnejšie použitie spomínanej knižnice *Rootbeer* [34].

b.3) Encog Framework Encog bol spomenutý už v predchádzajúcej sekcií ako framework určený k implementácií neurónových sietí. V súčastnosti sa Encog snaží rozšíriť svoju funkčnosť tým, že začína umožňovať spúšťanie programov pomocou grafických jednotiek pomocou modelu CUDA. V súčastnosti Encog podporuje spúšťanie len niektorých operácií pomocou GPU a je súčastou nástroja *Encog for C*, ktorý primárne slúži pre implementáciu neurónových sietí v jazyku C.

3.2.3 Implementácia navrhnutých modelov konvolučných neurónových sietí

Na základe dostupných frameworkov pre implementáciu konvolučných neurónových sietí v prostredí programovacieho jazyka Java som sa nakoniec rozhodla implementovať modely konvolučných sietí, navrhnutých v časti 3.1.2 vo frameworku *Neuroph*. Pre použitie Neuroph-u som sa rozhodla, pretože tento framework umožňuje použitie vstavaných funkcií pre implementáciu konvolučných neurónových sietí. Nevýhodou je, že v aktuálnej verzií, ktorou je *Neuroph 2.9* nie je možné pre prácu s konvolučnými neurónovými sieťami použiť príslušný grafický nástroj, ale jednotlivé konvolučné neurónové siete je nutné imlementovať manuálne. Framework Neuroph je dostupný tu [9].

a) Implementácia vstupných dát

Pred vytvorením samotnej konvolučnej neurónovej siete je nutné implementovať dataset, ktorý slúži pre učenie a následné testovanie konvolučnej neurónovej siete. Ako už bolo spomínané pre učenie a testovanie siete je použitý dataset MNIST, ktorý však pred samotným vložením do neurónovej siete musí by upravený na potrebný tvar.

U frameworku Neuroph sa pre vytváranie datasetu používa trieda `DataSet`. Každý `DataSet` je zložený z riadkov, kde každý riadok reprezentuje jeden vzor, ktorý je aplikovaný do konvolučnej neurónovej siete. Tento riadok je definovaný prostredníctvom triedy `DataSetRow` a skladá sa zo vstupných dát a požadovaného výstupu. V prípade datasetu MNIST je tento dataset uložený v štyroch súboroch, kde obrázky a k nim príslušné labely sa nachádzajú v rozličných súboroch, ktorých štruktúra je uvedená na webových stránkach datasetu MNIST, ktoré sú uvedené tu [11], preto je nutné tieto súbory upraviť pred samotným vytvorením datasetu, ktorý bude predložený konvolučnej neurónovej sieti.

Pre implementáciu vytvorenia datasetu z MNIST súborov bola vytvorená trieda `MNISTDataSet`, ktorej výstupom je upravený dataset, ktorý môže byť následne predložený konvolučnej neurónovej sieti. Táto trieda sa skladá z niekoľkých metód, ktoré umožňujú vytvorenie tohto datasetu. Prvou metódou je metóda `nacitajObrazok`, ktorej vstupnými parametrami sú cesty k súborom s obrázkami a k nim príslušnými datasetmi. Táto metóda následne pomocou metódy `getResourceAsStream` načítava dáta z týchto súborov, odstráňuje prebytočné dáta a vracia zoznam, ktorý sa skladá z bytových dát obrázku a k nim príslušného labelu.

Ďalšou metódou v triede `MNISTDataSet` je metóda `vytvorDataset`, cieľom tejto metódy vytvoriť vhodne upravený dataset, ktorý bude predložený konvolučnej neurónovej sieti. Táto metóda je definovaná dvomi parametrami, kde prvý parameter je

zoznam MNSIT obrázkov, ktorý bol vytvorený v metóde `nacitajObrazok` a druhým parametrom je užívateľom zvolený počet obrázkov, ktoré budú z datasetu MNIST použité. Keďže jednotlivé riadky definovaného datasetu majú vstupné aj očakávané dáta v dátovom type *double*, je nutné prekonvertovať už upravené MNIST obázky, ktorých dáta sú v dátovom type *byte*. V rámci tejto metódy je teda zobraný užívateľom zvolený počet obrázkov zo zoznamu, následne sú bytové dáta obrázku prekonvertované na hodnotu double a spolu s labelom sú uložené ako riadok datasetu do premennej `DataSet`.

Poslednou metódou v triede `MNISTDataSet` je trieda `nacitajDataSet`, ktorá spravuje predošlé dve metódy. Parametrami tejto metódy sú užívateľom zadané cesty k súborom s obrázkami a labelmi, ďalším parametrom je užívateľom definovaný počet koľko obrázkov a labelov z MNIST datasetu bude použitých pre dataset, ktorý bude predložený konvolučnej neurónovej sieti.

b) Vytvorenie konvolučnej neurónovej siete

Po načítaní potrebných datasetov pre učenie a testovanie siete, pokračujem v programe samotnou implementáciou konvolučnej neurónovej siete. V prostredí Neuroph je konvolučná sieť definovaná prostredníctvom triedy `ConvolutionalNetwork`. Po vytvorení objektu, ktorý predstavuje konvolučnú neurónovú sieť sa postupne pridávajú jednotlivé vrstvy konvolučnej neurónovej siete. Pre každý typ vrstvy neurónovej siete obsahuje Neuroph špecifické triedy.

Konvolučná vstva siete je definovaná prostredníctvom triedy `ConvolutionalLayer`. Pri vytváraní objektu pre konvolučnú vrstvu, je nutné definovať niekoľko atribútov. Prvým atribútom je vrstva, na ktorú bude konvolučná vrstva aplikovaná, ďalším atribútom recepčné pole, ktoré je definované vrámci triedy `Kernel` a je určený svojími rozmermi. Posledným atribútom konvolučnej vrstvy je počet recepčných polí, ktorý je použitý v tejto vrstve. Príklad vytvorenia konvolučnej vrstvy je uvedený na následujúcom príklade:

```
ConvolutionalLayer konvolucnaVrstva1 = new ConvolutionalLayer
        (vstupnaVrstva, konvolucneJadro, 6);
```

Ďalšou vrstvou, ktorá je použitá v konvolučných neurónových sieťach je vrstva subsamplingová. Táto vrstva je v prostredí frameworku Neuroph definovaná prostredníctvom triedy `PoolinLayer`. Rovnako ako u konvolučnej vrstvy aj pri vytváraní objektu pre subsamplingovú vrstvu je nutné definovať niekoľko atribútov. Prvým atribútom sa rovanko ako u konvolučnej vrstvy definuje vrstva konvolučnej neurónovej siete, na ktorú bude subsamplingová vrstva aplikovaná. Druhým atribútom

pri vytváraní subsamplingovej vrstvy je jadro, ktoré bude aplikované na definovanú vrstvu. Toto jadro je rovnako ako v predchádzajúcom prípade definované pomocou triedy `Kernel` a je určené svojími rozmermi. Príklad vytvorenia subsamplingovej vrsvy v prostredí frameworku Neuroph je ukázaný na následujúcom príklade:

```
PoolingLayer subsamplingVrstva1 = new PoolingLayer
   (konvolucnaVrstva1, subsamplingJadro);
```

Poslednou vrstvou, ktorá je použitá v navrhnutých konvolučných neurónových sieťach je plne prepojená vrstva, ktorá slúži ako výstup konvolučnej neurónovej siete. Táto vrstva je v prostredí frameworku Neuroph definovaná v triede `Layer` a pri vytváraní príslušného objektu, užívateľ definuje ako atribúty počet neurónov v tejto vrstve a vlastnosti týchto neurónov. Príklad vytvorenia tejto vrstvy konvolučnej neurónovej siete je ukázaný na následujúcom príklade:

```
Layer vystupnaVrstva = new Layer(10, neuron);
```

Ako bolo spomenuté, pred vytvorením samotnej plne prepojenej vrstvy je nutné vytvoriť a definovať neuróny použité v tejto vrstve. Neuróny sú v prostredí Neuroph definované prostredníctvom triedy `NeuronProperties` a jednotlivé atribúty sa definujú prostredníctvom metódy `setProperties`. V rámci práce som pre jednotlivé neuróny definovala atribúty ako `useBias`, v rámci ktorého je pre danú konvolučnú neurónovú sieť nastavený pomocný Bias neurón s hodnotou +1. Ďalším nastaveným atribútom pre neuróny je `transferFunction`, ktorý predstavuje aktivačnú funkciu, ktorá je nastavená ako funkcia *sigmoid*, ktorá je graficky zobrazená v časti 1.1.2. Posledným atribútom pri nastavení neurónov je `inputFunction`, ktorá počíta vnútorné hodnoty jednotlivých neurónov na základe váh a biasu. Príklad nastavenia atribútov pre neuróny v prostredí Neuroph je ukázaný na následujúcom príklade:

```
NeuronProperties neuron = new NeuronProperties();
NeuronProperties neuron = new NeuronProperties();
neuron.setProperty("useBias", true);
neuron.setProperty("transferFunction",TransferFunctionType.SIGMOID);
neuron.setProperty("inputFunction",WeightedSum.class);
```

Po vytvorení jednotlivých vrstiev je nutné tieto vrstvy pridať do už vytvorenej konvolučnej neurónovej siete. Pridávanie vrstiev je umožnené pomocou metódy `addLayer`, ktorá je súčasťou triedy `ConvolutionalNetwork`. Príklad pridávania vrstiev do konvolučnej neurónovej siete je vidieť na následujúcom príklade:

```
konvolucnaSiet.addLayer(vstupnaVrstva);
konvolucnaSiet.addLayer(konvolucnaVrstva1);
konvolucnaSiet.addLayer(subsamplingVrstva1);
```

Po pridaní jednotlivých vrstiev do konvolučnej neurónovej siete je nutné susedné vrstvy medzi sebou prepojiť. Toto prepojenie je zabezpečené prostredníctvom metódy `fullyConectMapLayers`, ktorá je súčasťou triedy `ConvolutionalUtils` a ako parametre metódy sú dosadené susedné vrstvy konvolučnej siete, ktoré majú byť spojené. Príklad použitia spojenia dvoch vrstiev konvolučnej neurónovej siete metódou `fullyConectMapLayers` je ukázaná na následujúcom príklade:

```
ConvolutionalUtils.fullConectMapLayers(vstupnaVrstva, konvolucnaVrstva1);
```

Pred zahájením samotného učenia konvolučnej neurónovej siete je nutné nastaviť vstupné a výstupné neuróny siete. Tieto nastavenia sa realizujú prostredníctvom metód `setInputNeurons` a `setOutpuNeurons`, ktoré sú súčasťou triedy `ConvolutionalNetwork`. Parametrami týchto metód sú neuróny vstupnej a výstupnej vrstvy konvolučnej neurónovej siete. Príklad nastavenia vstupných a výstupných neurónov konvolučnej neurónovej siete je vidieť na následujúcom príklade:

```
konvolucnaSiet.setInputNeurons(vstupnaVrstva.getNeurons());
konvolucnaSiet.setOutputNeurons(vystupnaVrstva.getNeurons());
```

Po nastavení štruktúry konvolučnej neurónovej siete je v programe nastavený typ učiaceho algoritmu, ktorý bude v sieti použitý. V prípade implementovaných konvolučných neurónových sietí je pre učenie sietí použitý algoritmus BackPropagation upravený pre konvolučné neurónové siete, ktorý v prostredí frameworku Neuroph zabezpečuje trieda `ConvolutionalBackpropagarion`, ktorá je dosadená ako atribút do metódy `setLearningRule`, ktorá je súčasťou triedy `ConvolutionalNetwork`. Príklad nastavenia učenia pre vytvorenú konvolučnú sieť je ukázaný na následujúcom príklade:

```
konvolucnaSiet.setLearningRule(new ConvolutionalBackpropagation());
```

Po nastavení typu učenia konvolučnej neurónovej siete nasleduje samotné učenie siete. Toto učenie je realizované prostredníctvom metódy `learn`, ktorá je súčasťou triedy `ConvolutionalNetwork`. Parametrom tejto metódy je obejkt triedy `DataSet`, ktorý bol vytvorený v časti 3.2.3. Príklad spustenia učenia siete v prostredí Neuroph je ukázaná na následujúcom príklade:

```
konvolucnaSiet.learn(trainSet);
```

Výstupom implementovaných konvolučných neurónových sietí je výsledná chybovosť konvolučnej neurónovej siete a čas, ktorý sieť strávila učením.

3.2.4 Implementácia výpočtou v konvolučnej neurónovej sieti pomocou GPU

V súčastnosti sa pri implementácií konvolučných neurónových sietí používa pre urýchlenie niektorých výpočtov implementácia pomocou grafických jednotiek. V tejto práci som sa rozhodla niektoré výpočty v navrhnutých konvolučných sieťach spustiť pomocou grafických jednotiek a výsledky porovnať s výsledkami, ktoré boli dosiahnuté v prípade, že aplikácia bežala iba pomocou procesorových jednotiek.

Pre implementáciu, ktorá je schopná bežať na grafických jednotkách som sa rozhodla použiť jazyk *OpenCL*, ktorý je schopný bežať na väčšom množstve druhov grafických kariet (nielen na grafických kartách NVIDIA ako je to u *CUDA*). Pri implemntácií som ako framework použila kód Ivana Vasileva založený na knižnici *Aparapi*, ktorý je licencovaný pomocou MIT licencie a teda je možné tento kód voľne používať, rozširovať a modifikovať. V tomto kóde je implementácia konvolučných neurónových sietí optimalizovaná tak, že niektoré výpočty je možne spúšťať pomocou grafických jednotiek, tento kód je dostupný na [27].

V rámci tejto diplomovej práce som sa rozhodla implementovať a porovnať rýchlosť výpočtov v jednotlivých vrstvách navrhnutých konvolučných neurónových sieťach. V prvom rade pri implementácií a najmä spúšťaní programu je nutné mať na všetkých jednotkách, či už grafických alebo procesorových nainštalovaný *OpenCL*. OpenCL je možné stiahnuť individuálne ako drivery pre rôzne typy podporovaných zariadení, napríklad pre NVIDIA grafické jednotky, je OpenCL dostupný na [28], pri Intel procesory je OpenCL dostupný na [29].

Po inštalácií OpenCL na grafické a procesorové jednotky, sa v programe prepínanie medzi týmito jednotkami nastavuje pomocou metódy `setExecutionMode`, ktorá je súčasťou triedy `Environment`. Tento príkaz má pre CPU v programe tvar:

```
Environment.getInstance().setExecutionMode(Kernel.EXECUTION_MODE.CPU);
```

V prípade použitia GPU je tento príkaz v tvare:

```
Environment.getInstance().setExecutionMode(Kernel.EXECUTION_MODE.GPU);
```

Následne je v programe pre jednotlivé vrstvy siete nastavené meranie koľko času pre výpočet konkrétnej vrtvy. Výpočet prebieha pomocou metódy `CurrentTimeMillis`, ktorá sa pred samotným výpočtom uloží do premennej `start` aktuálny čas programu. Táto metóda je súčasťou triedy `System`.

Na začiatku programu je najprv vytvorený model konvolučnej neurónovej siete. Tento model zodpovedá navrhnutým konvolučným sieťam v časti 3.1.2. Konvolučná neurónová sieť je v tomto prípade vytvorená pomocou metódy `convNN`, ktorá je súčasťou triedy `NNFactory`. V tomto pripade sú parametre konvolučnej neurónovej siete zadané ako parameter metódy `layers`, kde prvá vrstva konvolučnej neurónovej siete musí obsahovať tri parametre, kde prvý parameter predstavuje riadky vstupnej vrstvy, druhý parameter predstavuje stĺpce a tretí parameter predstavuje počet filtrov, v tomto prípade je táto hodnota rovná väčšinou hodnote 1. V prípade, že za vstupnou vrstvou nasleduje vrstva konvolučná, musí v tejto metóde obsahovať štyri parametre, kde prvý parameter predstavuje počet riadkov recepčného poľa, druhý parameter predstavuje počet stĺpcov recepčného poľa, tretí parameter reprezentuje počet recepčných polí a štvrtý parameter predstavuje hodnotu o koľko pixelov sa jednotlivé recepčné polia posúvať po predchádzajúcej vrstve.V prípade, že konvolučná neurónová sieť obsahuje subsamplingovú vrstvu, je v tejto metóde definovaná prostredníctvom dvoch parametrov, ktorými sú riadky a stĺpce subsamplingového poľa. V prípade poslednej plne prepojenej vrstvy je táto vrstva v metóde definovaná jedným číslom, ktoré určuje počet neurónov. Ďalším parametrom metódy je parameter `addBias`, ktorý udáva, či v konvolučnej neurónovej je pri výpočtoch použitá hodnota biasu +1. Príkaz vytvorenia konvolučnej neurónovej siete, ktorá zodpovedá konvolučnej sieti 1 z časti 3.1.2 má nasledujúci tvar:

```
NeuralNetworkImpl nn = NNFactory.convNN(new int[][] { { 28, 28, 1 },
                { 20, 20, 10, 1 }, {9, 9 }, {10} }, true);
```

Následne sú z konvolučnej neurónovej siete postupne vyberané jednotlivé vrstvy konvolučnej siete a počítané ich hodnoty. U konvolučnej vrstvy sú hodnoty tejto vrstvy počítané pomocou triedy `AparapiConv2D`, ktorá umožňuje, že tento výpočet môže byť spustený ako na procesorových tak aj na grafických jednotkách. Pre vytvorenie objektu tejto triedy je nutné najprv vybrať danú konvolučnú vrstvu z konvolučnej neurónovej siete. Táto konvolučná vrstva sa vytvára ako objekt triedy `Conv2DConnection` a príkaz je v tvare:

```
Conv2DConnection konvolucnaVrstva = (Conv2DConnection) KonvolucnaSiet1
          .getInputLayer().getConnections().get(0);
```

Ďalším parametrom pre vytvorenie objektu, ktorý umožňuje výpočet konvolučnej vrstvy pomocou triedy `AparapiConv2D` je objekt, ktorý obsahuje hodnoty jednotlivých pixelelov konvolučnej neurónovej vrstvy, tento objekt je vytvorený z triedy `ValuesProvider` a jednotlivé hodnoty z konvolučnej vrstvy sú získané pomocou iterátora. Posledným parametrom je následujúca vrstva. Samotné spustenie výpočtu je realizované pomocou metódy `calculate`, ktorej parametre sú rovnaké ako pri vytváraní objektu a príkaz je v tvare:

```
conv.calculate(konvolucnaVrstva, vypocet, konvolucnaVrstva
        .getOutputLayer());
```

Pre výpočet subsamplingovej vrstvy je použitá trieda `AparapiMaxPooling2D`, ktorá obsahuje metódu `calculate`, ktorá sprostredkováva výpočty v tejto vrstve. Táto metóda obsahuje tri parametre, kde prvým parametrom je zoznam spojení, ktoré má subsamplingová vrstva s predchádzajúcou vrstvou. Ďalším parametrom je objekt, ktorý obsahuje hodnoty jednotlivých prvkov subsamplingovej vrstvy, tento objekt je opäť súčasťou triedy `ValuesProvider`. Poslednným parametrom je rovnako ako u výpočtu konvolučnej vrstvy vrstva, ktorá následuje za touto subsamplingovou vrstvou.

Dáta v tomto prípade nie sú konvolučnej sieti dodávané prostredníctvom datasetu, ale manuálne u konvolučnej vrstvy je to prostredníctvom metód `getWeights` a `setElements`. Do subsamplingovej vrstvy sú tieto hodnoty kopírované prostredníctvom pomocného poľa.

3.3 Štruktúra a spustenie programu

Program implementovaný v rámci tejto diplomovej práce je ako už bolo spomínané v prostredí jazyka Java a skladá sa z 2 balíčkov. Prvý balíčkom je baliček *KonvolucnaSiet*, ktorý sa skladá z tried, v ktorých sú implementované jednotlivé konvolučné siete. V triedach *MNISTKnn1* až *MNISTKnn4* sú implementované jednotlivé konvolučné siete pomocou frameworku Neuroph, ktoré boli navrhnuté v časti 3.1.2 a implementácia bola popísaná v časti 3.2.3. Metódy, ktorými sú jednotlivé triedy spúšťané zodpovedajú názvom tried. Tieto metódy majú jeden parameter, ktorým je počet vzorov, ktoré bude konvolučná neurónová sieť používať pri učení.

Implementácia výpočtov jednotlivých vrstiev konvolučných neurónových sietí pomocou grafických jednotiek, ktorá je popísaná v časti 3.2.4, sa nachádza v triedach

KnnGPU1 až *KnnGPU4*, kde každá trieda opäť zodpovedá jednotlivým konvolučným neurónovým sieťam, ktoré boli navrhnuté v časti 3.1.2. Každá trieda obsahuje jednu metódu, ktorá spúšťa výpočty jednotlivých vrstiev, tieto metódy neobsahujú žiadne parametre.

Spúšťacou triedou je trieda *KNN*, v ktorej je implementované vytvorenie objektov jednotlivých tried a spustenie príslušných metód.

Ďalším balíčkom programu je balíček MNIST, ktorý obsahuje implementáciu spracovania vstupných dát popísanú v časti 3.2.3. Tento balíček taktiež obsahuje súbory datasetu MNIST.

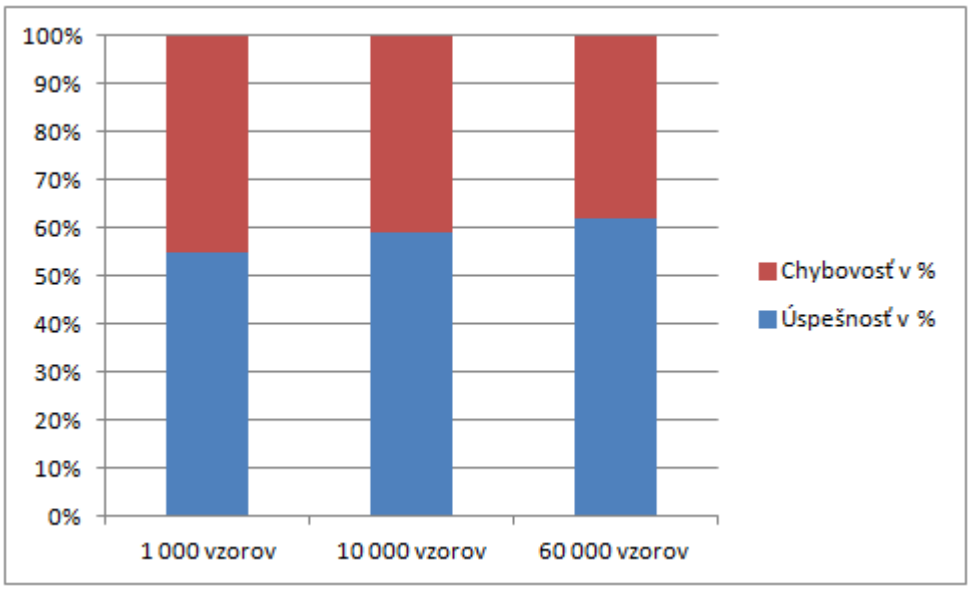
4 DOSIAHNUTÉ VÝSLEDKY

V tejto časti práce budú postupne ukázané a porovnané výstupy jednotlivých implementovaných konvolučných neurónových sietí, ktoré boli popísané v predchádzajúcej kapitole. Pri testovaní a spúšťaní jednotlivých konvolučných neurónových sietí bol použitý procesor *Intel Core i7-3632QM CPU @ 2.20GHz*. Pri testovaní spúšťaní konvolučných neurónových sietí prostredníctvom grafických jednotiek bola použitá grafická karta *NVIDIA GeForce GT 635M*.

4.1 Výsledky prvého modelu konvolučnej neurónovej siete

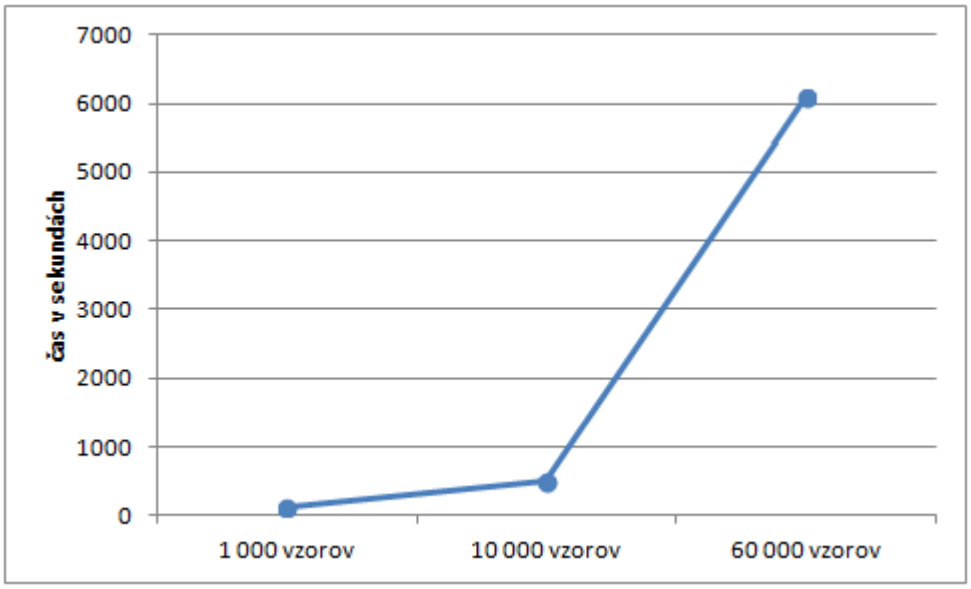
Ako prvý model konvolučnej neurónovej siete bola v časti 3.1.2 navrhnutá menšia konvolučná neurónová sieť, ktorá sa skladá z jednej konvolučnej a jednej subsamplingovej vrstvy. Prehľad jednotlivých vrstiev tejto siete je možné vidieť v tabuľke 3.1.

Počas testovania tejto konvolučnej neurónovej siete bolo na túto sieť aplikovaných niekoľko testov, ktorých cieľom bolo zistiť akú chybovosť bude konvolučná neurónová sieť dosahovať a za aký čas je sieť schopná naučiť sa predložené množstvo dát. Pri testovaní sietí bol algoritmus Backpropagation pre konvolučné neurónové siete nastavený tak, že učenie siete prebiehalo v piatich iteráciach tohto algoritmu. V prvom teste bolo konvolučnej neurónovej sieti predložených 1000 vzorov z datastu



Obr. 4.1: Graf zobrazujúci chybovosť v prvej konvolučnej neurónovej sieti

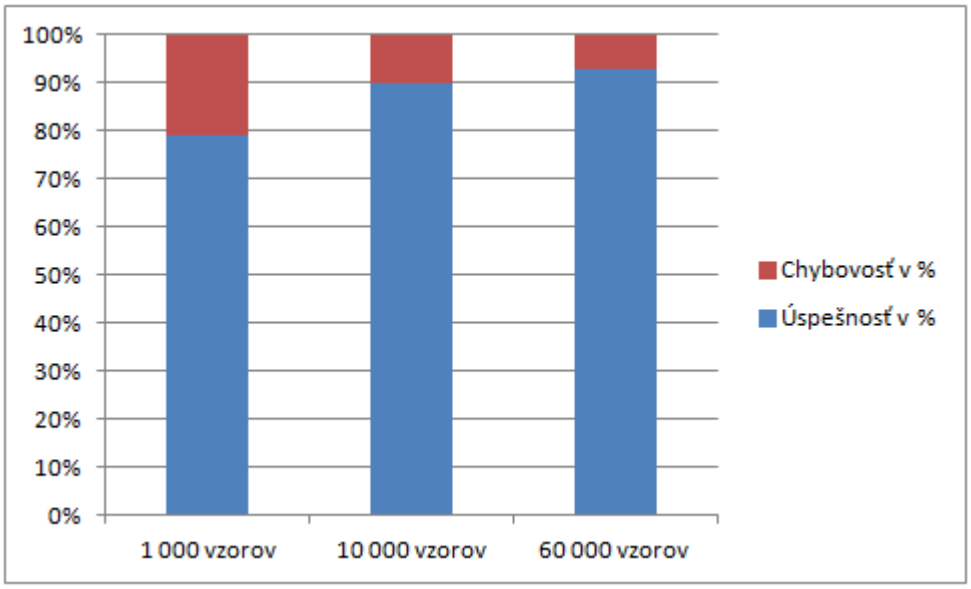
MNIST. V tomto prípade učenie sieti trvalo 133.124 sekúnd, čo je približne 2 minúty a 13 sekúnd, po tomto učení dosahovala úspešnosť siete hodnotu 55%.
V druhom teste bolo konvolučnej neurónovej sieti predložených 10000 vzorov z databázy MNIST. V tomto teste trvalo učenie siete približne 8 minút a 15 sekúnd a úspešnosť siete dosahovala hodnotu 57%.
V poslednom treťom teste bolo konvolučnej neurónovej sieti predložená celá databázy MNIST zložená zo 60000 vzorov. V tomto prípade učenie siete trvalo približne jednu hodinu a 42 minút a úspešnosť siete dosahovala hodnotu 59%. Výsledky dosiahnuté testovaním siete sú zobrazené v grafe 4.1, ktorý zobrazuje úspešnosť siete v jednotlivých testoch a v grafe 4.2, ktorý zobrazuje časi, ktoré sieť strávila učením v jednotlivých testoch.



Obr. 4.2: Graf zobrazujúci časový priebeh priebeh konvolučnej neurónovej siete

4.2 Výsledky druhého modelu konvolučnej neurónovej siete

Ako ďalší model konvolučnej neurónovej siete bola navrhnutá konvolučná neurónová sieť, ktorá sa skladala z troch konvolučných vrstiev, dvoch subsamplingových vrstiev a jednej plneprepojenej výstupnej vrstvy. Zobrazenie štruktúry a jednotlivých vrstiev tejto konvolučnej siete je možné vidieť v tabuľke 3.2. Rovnako ako u predchádzajúceho modelu konvolučnej neurónovej siete, aj v tomto prípade bolo na sieť aplikované tri série testov, kde v prvom teste bol sieti predložený dataset MNIST s 1000 vzormi, v ďalšom teste bol sieti predložený dataset s 10000 MNIST vzormi a v

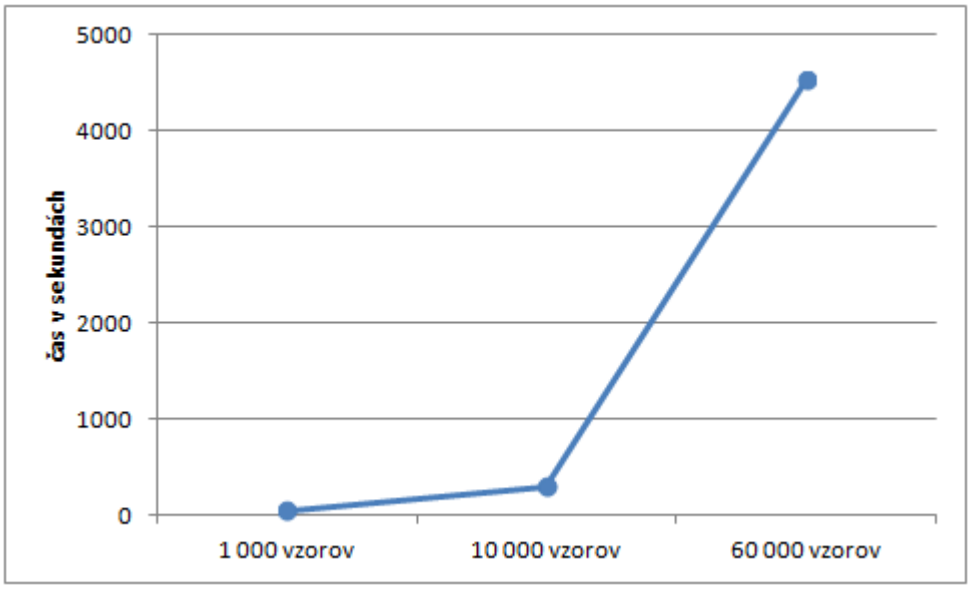


Obr. 4.3: Graf zobrazujúci chybovosť v druhej konvolučnej neurónovej sieti

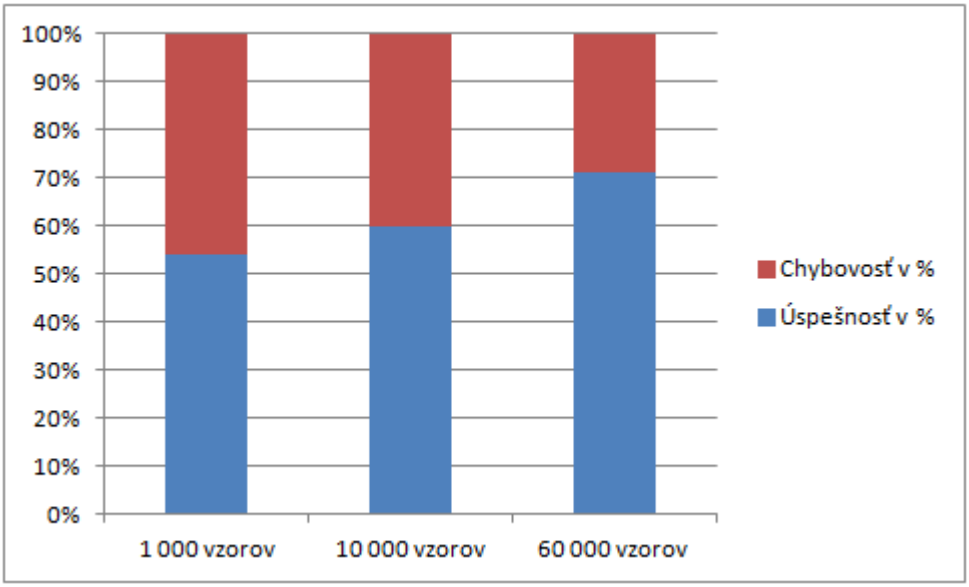
poslednom teste bol sieti predložený celý dataset MNIST so 60000 vzormi. Rovnako ako v predchádzajúcom prípade, aj tu bol algoritmus Backpropagation pre konvolučné neurónové siete aplikovaný v piatich iteráciach.
V prvom teste bola konvolučná neurónová sieť schopná naučiť sa použitý dataset za 45 sekúnd a úspešnosť siete dosahovala hodnotu 89%.
V druhom teste učenie konvolučnej neurónovej siete trvalo 4 minúty a 55 sekúnd a úspešnosť siete dosahovala hodnotu 90%.
V poslednom teste zvládla sieť učenie za hodinu a 6 minút a úspešnosť siete bola 94%.
Dosiahnuté výsledky získané testovaním tohto modelu konvolučnej neurónovej siete je vidieť v grafe 4.3, ktorý zobrazuje chybovosť siete v jednotlivých testoch a v grafe 4.4, v ktorom je zobrazený čas, ktorý konvolučná neurónová sieť strávila učením jednotlivých datasetov.

4.3 Výsledky tretieho modelu konvolučnej neurónovej siete

Ako tretí model konvolučnej neurónovej siete bola vytvorená väčšia konvolučná neurónová sieť ktorá sa skladala zo štyroch konvolučných vrstiev, troch subsamplingových vrstie a jednej plneprepojenej výstupnej vrstvy. Štruktúru tejto konvolučnej siete a prehľad jednotlivých vrstiev je možné vidieť v tabuľke 3.3.
Rovnako aj v tomto prípade boli na konvolučnú neurónovú sieť aplikované tri testy



Obr. 4.4: Graf zobrazujúci časový priebeh druhej konvolučnej neurónovej siete

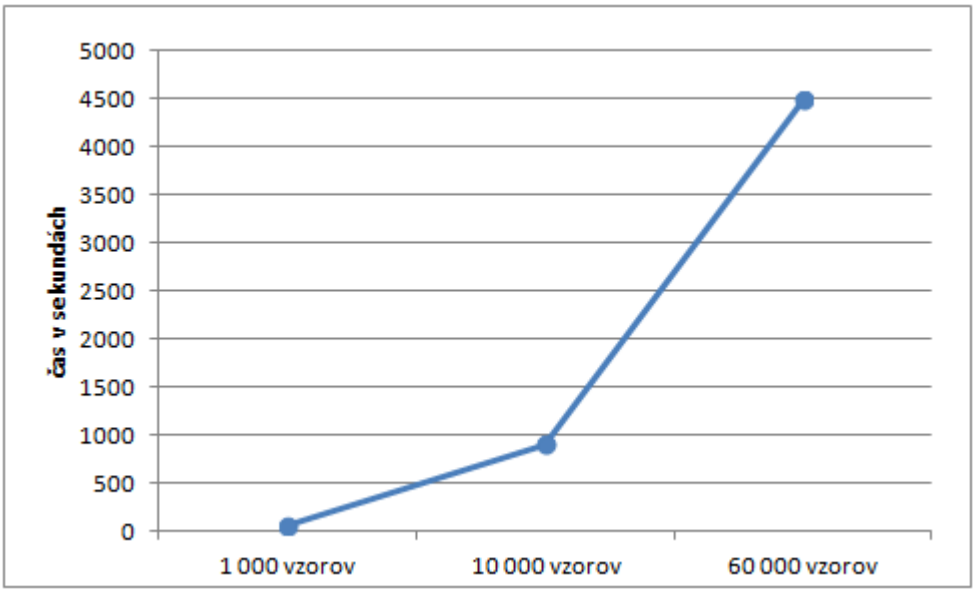


Obr. 4.5: Graf zobrazujúci chybovosť v tretej konvolučnej neurónovej sieti

s datasetmi obsahujúcimi 1000, 10000 a 60000 MNIST vzormi. V prvom teste trvalo sieti učenie daného datasetu s 1000 vzormi približne jednu minútu a 5 sekúnd a úspešnosť siete dosahovala hodnotu 54%.

V následujúcom teste trvalo konvolučnej sieti učenie datasetu o veľkosti 10000 vzorov 15 minút a 10 sekúnd. Úspešnosť siete dosahovala v tomto prípade 60% .

V poslednom teste učenie konvolučnej neurónovej siete datasetom o 60000 vzorov trvalo tejto sieti približne hodinu a 15 minút a úsešnosť siete dosahovala hodnotu



Obr. 4.6: Graf zobrazujúci časový priebeh tretej konvolučnej neurónovej siete

70%.

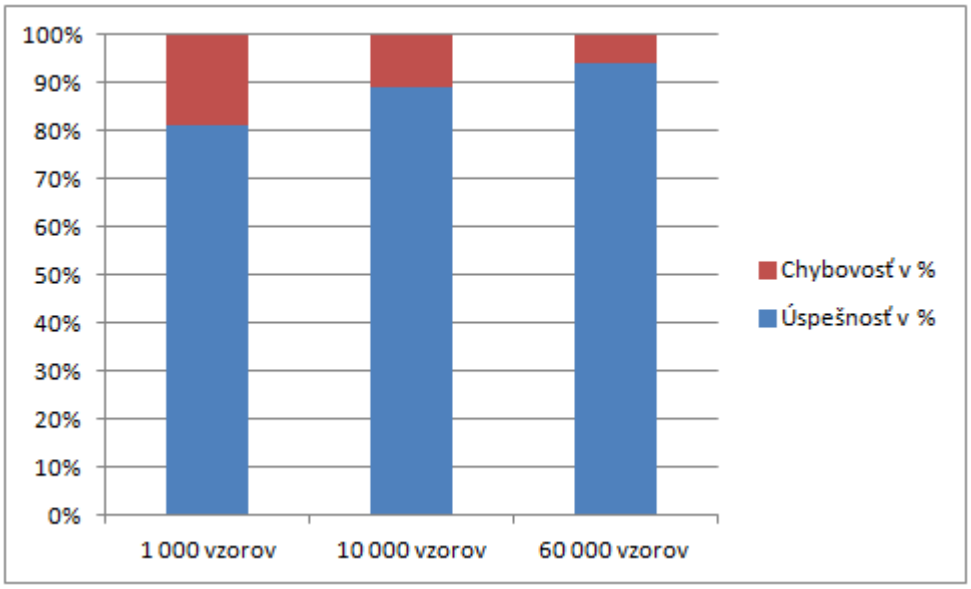
4.4 Výsledky štvrtého modelu konvolučnej neurónovej siete

V poslednom štvrtom modely konvolučnej neurónovej siete bola navrhnutá konvolučná neurónová sieť, ktorá sa z hladiska počtu a organizácie jednotlivých vrstiev zhoduje s druhým typom konvolučnej neurónovej siete. Rozdiel medzi týmito modelmi je v použitom množstve filtrov v jednotlivých vrstvách konvolučnej neurónovej siete. Prehľad štruktúry tejto konvolučnej neurónovej siete a jej vrstiev je možné vidieť v tabuľke 3.4.

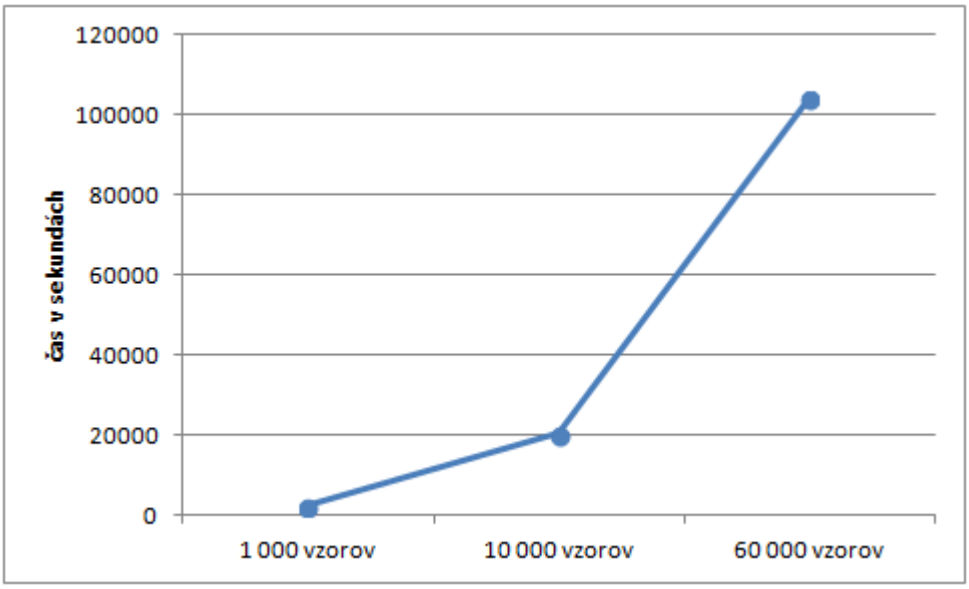
Rovnako aj v tomto prípade boli na konvolučnú neurónovú sieť aplikované tri testy, ktoré sa medzi sebou líšili vo veľkosti použitého datasetu a to vo veľkostiach 1000, 10000 a 60000 MNIST vzorov. V prvom teste bola sieť učená datasetom o veľkosti 1000 vzorov a toto učenie trvalo konvolučnej neurónovej sieti približne 40 minút a 15 sekúnd a úspešnosť siete bola 81%

V ďalšom teste trvalo konvolučnej neurónovej sieti učenie datasetu o 10000 vzorov približne 5 hodín a 45 minút. Úspešnosť siete dosahovala v tomto prípade hodnotu 89%.

V poslednom teste s použitím datasetu 60000 znakov trvalo konvolu4nej neurónovej sieti učenie približne 29 hodín 15 minút a úspešnosť siete v tomto prípade dosahovala hodnotu 95%.



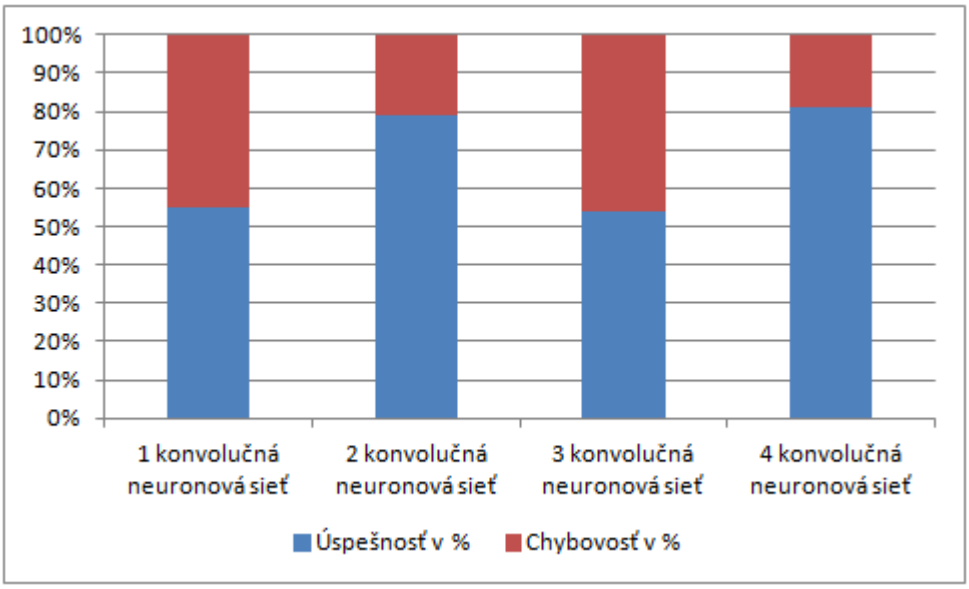
Obr. 4.7: Graf zobrazujúci chybovosť v štvrtej konvolučnej neurónovej sieti



Obr. 4.8: Graf zobrazujúci časový priebeh štvrtej konvolučnej neurónovej siete

4.5 Porovnanie výsledkov jednotlivých modelov konvolučných neurónových sietí

V tejto časti práce som sa rozhodla porovnať výsledky jednotlivých konvolučných neurónových sietí vzájomne medzi sebou, aby bolo jasne a prehľadne vidieť, ktorá z vytvorených konvolučných neurónových sietí je najvhodnejšia k použitiu vzhľadom

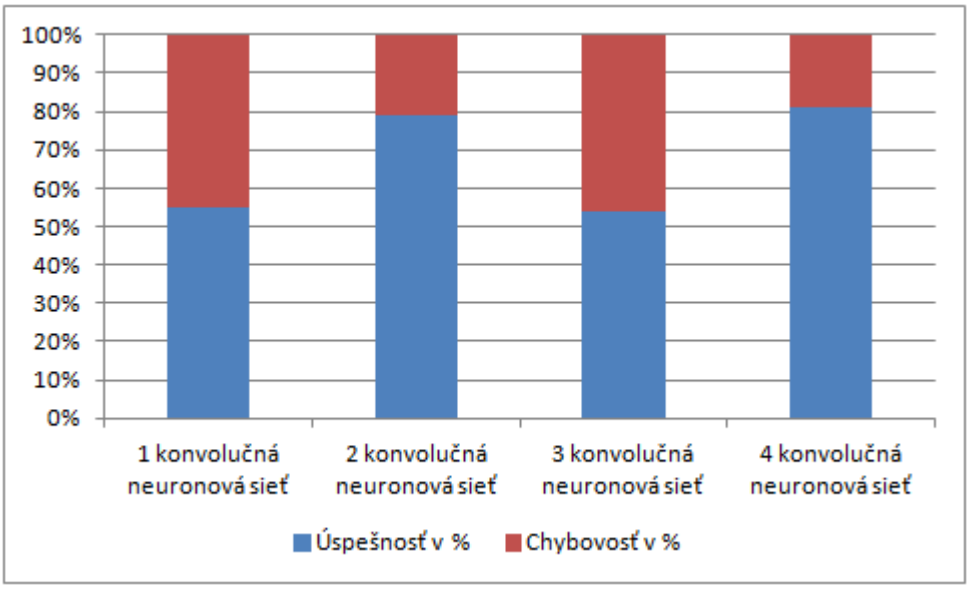


Obr. 4.9: Graf porovnanie úspešností konvolučných neurónových sietí pre 1000 vzorov

k úspešnosti úhadnutých vzorov, ale aj k času, ktorý konvolučná neurónová sieť strávila učením jednotlivých vzorov.

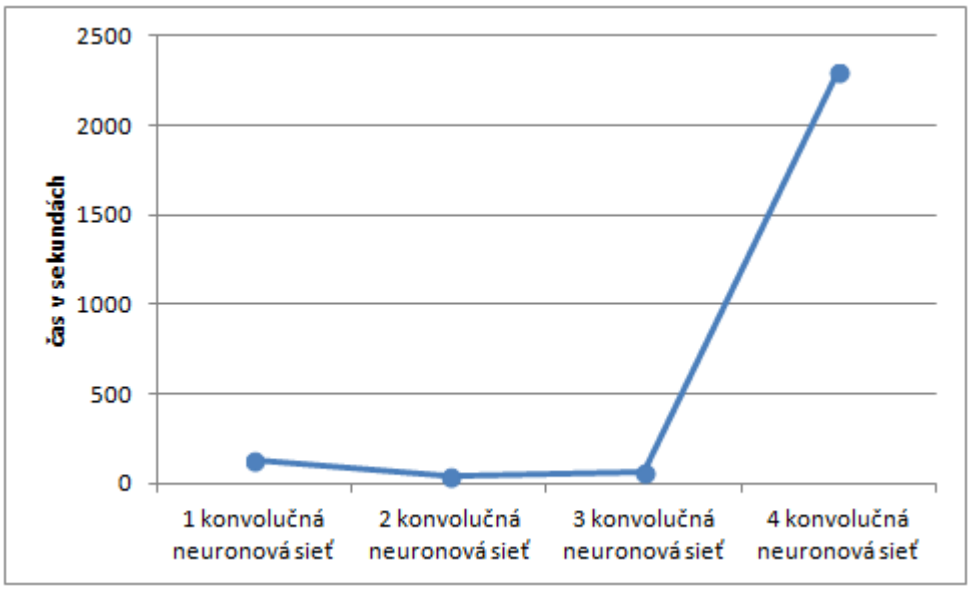
V prvom príklade bolo k učeniu konvolučných sietí použitých 10000 vzorov. Ako je vidno na grafe 4.12 najvyššiu úspešnosť okolo 80% dosiahli konvolučná naurónová sieť 2 a konvolučná neurónová sieť 4, ktoré tvorili v spomenutých architektúrach stredne veľké konvolučné siete, ktoré sa skladali z 3 konvolučných vrstiev (s recepčnými poľami vo veľkosti 5×5 a 4×4) a z 2 subsamplingových vrstiev (s veľkosťou filtrov 2×2). U menšej konvolučnej siete 1, ktorá sa skladala z jedenej konvolučnej vrstvy (s veľkosťou recečného poľa 20×20) a jednej subsamplingovej vrstvy (s veľkosťou filtru 9×9) bola úspešnosť siete podstatne nižšia a to iba 55%.
U najväčšej konvolučnej siete 3, ktorá sa skladala zo 4 konvolučných vrstiev (s veľkosťou recepčných polí 2×2) a 3 subsamplingových vrstiev (s veľkosťou filtrov 2×2) neboli výsledky z hľadiska úspešnosti siete taktiež veľmi priaznivé a dosahovali hodnotu iba 54%.

Z hľadiska doby, ktorú jednotlivé konvolučné neurónové siete strávili učením 1000 vzorov z databázy MNIST strávila najviac času učením konvolučná neurónová sieť 4, ktorá zo spomenutých konvolučných neurónových sietí obsahuje najviačší počet recepčných polí v jednotlivých vrstvách (konkrétne v prvej vrstve–50, v druhej vrstve–100, v tretej vrstve–250), čo je v porovnaní s ostanými sieťami viac ako dvojnásobok v každej vrstve. Ako je vidieť v grafe 4.13 dosahuje konvolčná neurónová sieť 4 s časom približne 40 minút niekoľko násobne vyššiu hodnotu ako ostatné kon-



Obr. 4.10: Graf porovnanie úspešností konvolučných neurónových sietí pre 1000 vzorov

volučné neurónové siete. V porovnaní s konvolučnou neurónovou sieťou 2, ktorá sa líši len v počte recepčných polí (konkrétny počet recepčných polí v prvej vrstve–6, v druhej vrstve–16, v tretej vrstve–120), učenie trvalo iba 45 sekúnd. Najmenšej

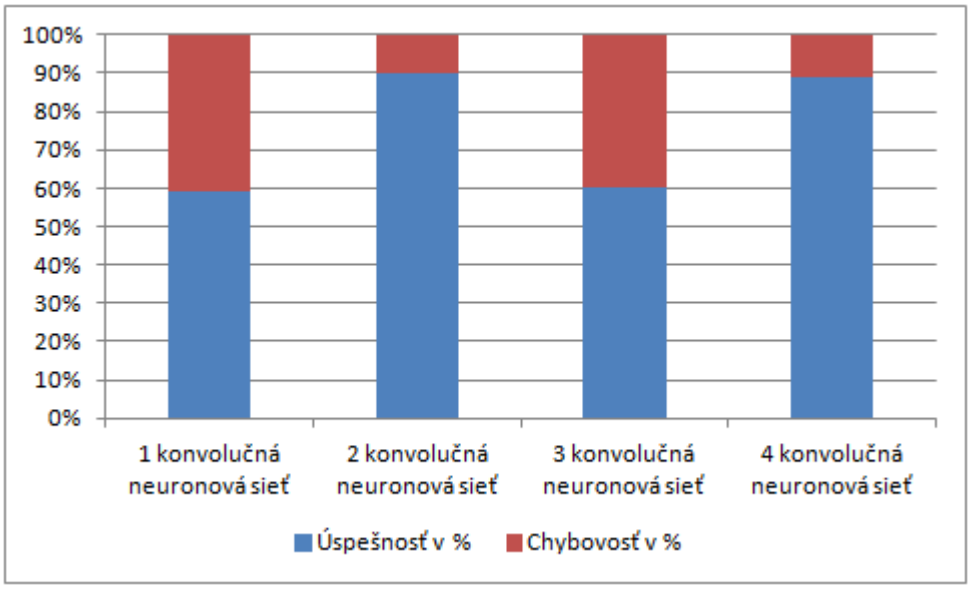


Obr. 4.11: Graf porovnanie časových priebehov konvolučných neurónových sietí pre 1000 vzorov

konvolučnej neurónovej sieti 1 s desiatimi väčšími recepčnými poľami trvalo učenie

niečo cez 2 minúty. U konvolúčnej neurónovej sieti 3, ktorej počet recepčných polí v jednotlivých vrstvách (konkrétne v prvej vrstve–6, v druhej vrstve–16, v tretej vrstve–120 a v štvrtej vrstve–150) je podobný ako u konvolučnej neurónovej siete 2 dosiahol čas učenia siete niečo cez 1 minútu.

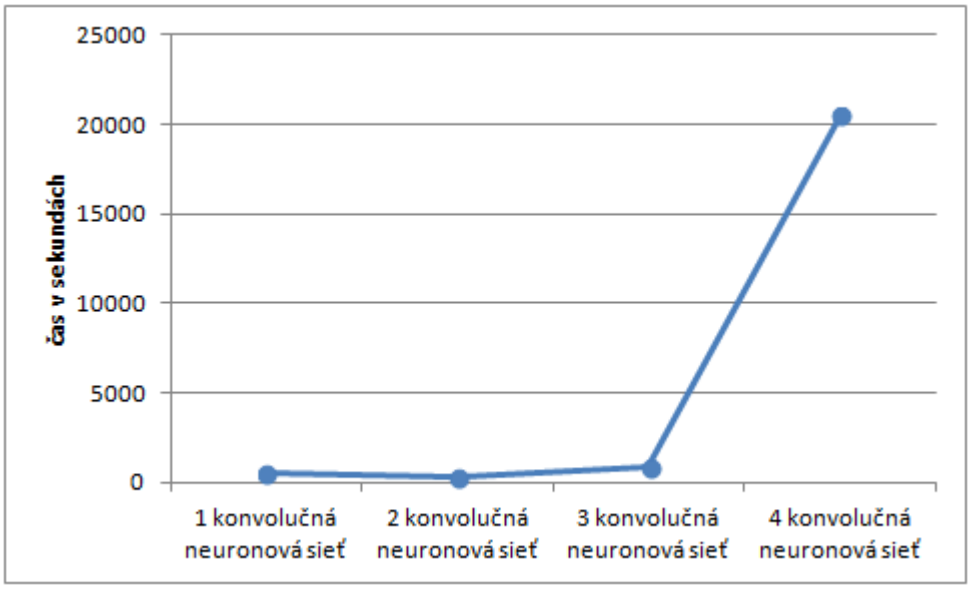
V druhom prípade bolo jednotlivým konvolučným sieťam poskytnutých 10000 vzorov pre učenie sietí. Ako je vidieť na grafe 4.12 pomer úspešností u jednotlivých



Obr. 4.12: Graf porovnanie úspešností konvolučných neurónových sietí pre 10000 vzorov

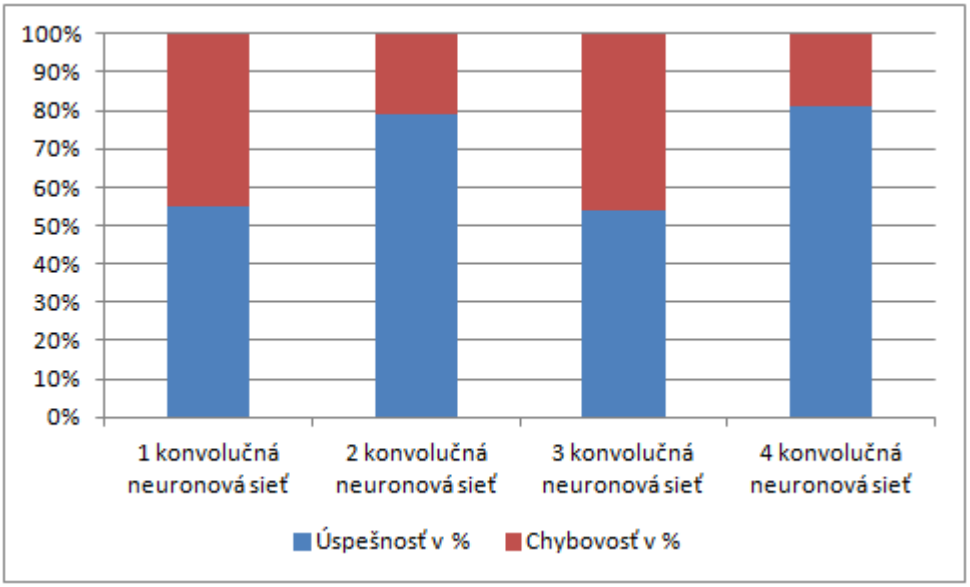
konolučných neurónových sietí sa veľmi nezmenil. Najväčiu úspešnosť dosiahli opäť konvolučná neurónová sieť 2 a konvolučná neurónová sieť 4, u ktorých sa v tomto prípade zvýšila úspešnosť z 80% na 90%. U menšej konvolučnej siete sa zvýšila úspešnosť z 55% na 59% a u poslednej konvolučnej siete 3 sa úspešnosť zvýšila z 54% na 60%.

Z hľadiska doby, ktorú konvolučné neurónové siete strávili učením a ktorá je zobrazená v grafe 4.13, je opäť vidieť, že pomer medzi dobami jednotlivých konvolučných sietí sa veľmi nezmenil. Najviac času učením opäť strávila konvolučná neurónová sieť, ktorá učením strávila 5 hodín a 45 sekúnd, čo je niekoľko násobne viac ako u ostatných sietí, kde maximálna doba učenia dosahovala niečo cez 15 minút u konvolučnej siete 3.



Obr. 4.13: Graf porovnanie časových priebehov konvolučných neurónových sietí pre 10000 vzorov

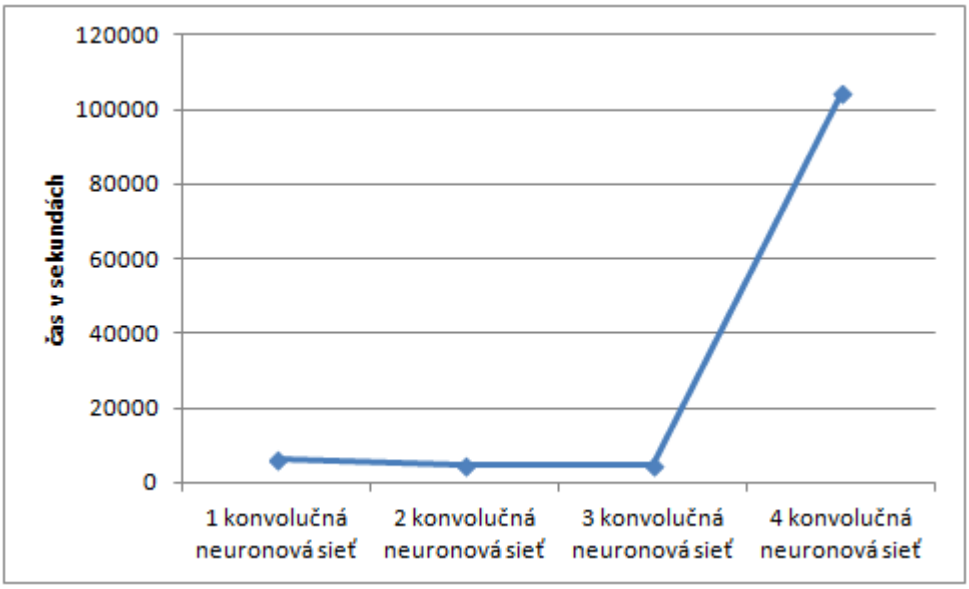
V poslednom teste, kedy bolo konvolučným neurónovým sieťam predložených 60000 vzorov z databázy MNIST. Podobne ako v predchádzajúcich prípadoch sa pomery výsledkov zobrazených v grafoch 4.14 a 4.15 veľmi nezmenili. Z hľadiska



Obr. 4.14: Graf porovnanie úspešností konvolučných neurónových sietí pre 60000 vzorov

úspešnosti dosiahli najlepšie výsledky opäť konvolučná neurónová sieť 2 a konvo-

lučná neurónová sieť 4, u ktorých úspešnosť vzrástla na hodnotu približne 95%. U menšej konolučnej neurónovej siete 2 dosiahla maxinmálna úspešnosť hodnotu okolo 60% a u konvolučnej siete 3 dosiahla maximálna úspešnosť siete hodnotu 71%. Z



Obr. 4.15: Graf porovnanie časových priebehov konvolučných neurónových sietí pre 60000 vzorov

hľadiska času, ktoré konvolučné neurónové siete strávili učením, opäť jednoznačne vedie konvolučná neurónová sieť 4, ktorá učením strávila viac ako 29 hodín. Pre porovnanie u konvolučnej neurónovej siete 1 dosahoval čas učenia hodnotu 1 hodinu a 42 minút, u konvolučnej neurónovej siete 3 dosahovala doba učenia 1 hodinu a 15 minút, u poslednej konvolučnej neurónovej siete 2 bola doba učenia niečo cez 1 hodinu.

Z dosiahnutých výsledkov je vidieť, že pre optimálnu úspešnosť konvolučnej neurónovej siete je nutné zvoliť vhodnú architektúru siete, ktorá nezávisí výhradne na veľkosti konvolučnej neurónovej siete, ani na množstve použitých recepčných polí, ale je nutné použiť vhodnú kombináciu týchto faktorov. Na druhú stranu z hľadiska časovej náročnosti konvolučnej neurónovej siete je vidieť, že s väčším množstvom použitých recepčných polí sa zvyšuje časová náročnosť konvolučnej neurónovej siete, pričom úspešnosť siete rastie len nepatrne.

4.6 Porovnanie výpočtov jednotlivých vrstiev medzi GPU a CPU

V tejto časti práce sú zobrazené a porovnané dosiahnuté výsledky získané z merania času, ktorý je porebný pre výpočty hodnôt jednotlivých vrstiev v konvolučných neurónových sieťach. V tabuľkách 4.1, 4.2, 4.3 a 4.4 sú zobrazené výsledky týchto meraní pre modely konvolučných neurónových sietí, ktoré boli navrhnuté v časti 3.1.2. Na základe výsledkov, ktoré sú zobrazené v uvedených tabuľkách je vidieť, že výpočty pomocou grafických jednotiek trvajú kratšie ako časové výsledky u výpočtov realizovaných pomocou procesorových jednotiek.

Typ vrstvy siete	CPU [s]	GPU [s]
Konvolučná	1.846	1.317
Subsamplingová	0.25	0.095

Tab. 4.1: Porovnanie výpočtov CPU a GPU v konvolučnej neurónovej sieti 1

Typ vrstvy siete	CPU[s]	GPU[s]
1.Konvolučná	0.846	0.782
1.Subsamplingová	0.15	0.096
2.Konvolučná	0.236	0.088
2.Subsamplingová	0.185	0.101
3.Konvolučná	0.2	0.085

Tab. 4.2: Porovnanie výpočtov CPU a GPU v konvolučnej neurónovej sieti 2

Typ vrstvy siete	CPU[s]	GPU[s]
1.Konvolučná	0.8	0.774
1.Subsamplingová	0.165	0.095
2.Konvolučná	0.154	0.087
2.Subsamplingová	0.132	0.09
3.Konvolučná	0.144	0.087
3.Subsamplingová	0.141	0.09
4.Konvolučná	0.147	0.091

Tab. 4.3: Porovnanie výpočtov CPU a GPU v konvolučnej neurónovej sieti 3

Typ vrstvy siete	CPU[s]	GPU[s]
1.Konvolučná	0.812	0.774
1.Subsamplingová	0.136	0.098
2.Konvolučná	0.416	0.108
2.Subsamplingová	0.135	0.087
3.Konvolučná	0.135	0.091

Tab. 4.4: Porovnanie výpočtov CPU a GPU v konvolučnej neurónovej sieti 4

Výpočty použité v tejto časti neboli príliš časovo náročné a teda časové rozdiely nie sú priliž rozlišné. Z hľadiska rozšíriteľnosti tejto práce by bolo možné aplikovať

grafické jednotky pri výpočte zložitejších operácií v konvolučných neurónových sieťach, ako je napríklad učenie týchto sietí a porovnať túto dobu s dobou, ktorú sieť strávila učením s použitím procesorových jednotiek.

5 ZÁVER

Táto práca bola venovaná problematike neurónových sietí a následným zameraním sa na špeciálny druh neurónových sietí, ktorým sú konvolučné neurónové siete. Na začiatku práce bola vysvetlená základná problematika neurónov a neurónových sietí, kde bola objasnená ich štruktúra, princíp činnosti a jednotlivé vlastnosti. Následne sa práca venovala rôznym modelom neurónových sietí a ich odlišnostiach. Práca sa následne zamerala na konvolučné neurónové siete, kde bola popísaná štruktúra tejto siete, jej vlastnosti a typ učenia. Následne boli uvedené konkrétne príklady z praxe, kedy bola konvolučná neurónová sieť použitá.

V praktickej časti práce boli najprv navrhnuté štyri modely konvolučných neurónových, na ktorých som chcela ukázať ako závisí veľkosť, zložitosť konvolučnej neurónovej siete a počet vzorov použitých pre učenie siete na úspešnosť učenia konvolučnej neurónovej siete a jej časovú náročnosť.

Pred samotnou implementáciou konvolučných neurónoých sietí sa v práci spomína výber vhodného frameworku pre implementáciu konvolučných neurónových sietí v prostredí programovacieho jazyka Java. V tejto časti boli taktiež spomenuté implementačné možnosti pomocou grafických jednotiek, ktoré sú v súčastnosti rozšírené pre urýchlenie zložitých výpočtov.

Po výbere vhodného frameworku je v práci popísaná samotná implementácia navrhnutých konvolučných neurónových sietí. V práci je taktiež spomínaná implementácia výpočtov jednotlivých vrstiev konvolučných neurónových sietí, ktorá bola implementovaná s použitím grafických aj procesorových jednotiek, aby bolo možné porovnať rýchlosti týchto výpočov.

V závere práce sú zhodnotené dosiahnuté výsledky u jednotlivých konvolučných neurónových sietí a to ich úspešnosť učenia vzhľadom na použité množstvo učiacich vzorov a taktiež časová náročnosť týchto konvolučných neurónových sietí rovnako vzhľadom na množstvo použitých vzorov. Následne sú výsledky jednotlivých konvolučných neurónových sietí porovnávané medzi sebou a zhodnotené, ktorý model konvolučnej neurónovej siete je najvhodnejší pre praktické použitie. V závere práce sú taktiež porovnané a zhodnotené doby výpočtov jednotlivých vrstiev konvolučných neurónových sietí medzi výpočtami realizovanými pomocou procesorových jednotiek a výpočtami realizovanými pomocou grafických jednotiek.

LITERATURA

[1] CELEBI, O.C. *Celebi Tutorial: Neural Networks and Pattern Recognition Using MATLAB* [online] [cit. 22. 11. 2014]. Dostupné z URL: <https://www.byclb.com/TR/Tutorials/neural_networks/ch7_1.htm>

[2] STERGIOU Ch., SIGANOS D. *Neural Networks* [online] [cit. 22. 11. 2014]. Dostupné z URL: <http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html>

[3] FUKUSHIMA K. *Neocognitron: A Hierarchial Neural Network Capable of Visual Pattern Recognition* [online] 1987 [cit. 22. 11. 2014]. Dostupné z URL: <http://vision.stanford.edu/teaching/cs131_fall1415/lectures/Fukushima1988.pdf>

[4] LE Q.L., NGIAM J., CHEN Z., CHIA D., KOH P.W., NG A.Y. *Tiled convolutional neural networks* [online] [cit. 22. 11. 2014]. Dostupné z URL: <http://ai.stanford.edu/~ang/papers/nips10-TiledConvolutionalNeuralNetworks.pdf>

[5] BOUVRIE J. *Notes on Convolutional Neural Networks* [online] 2006, [cit. 22. 11. 2014]. Dostupné z URL: <http://cogprints.org/5869/1/cnn_tutorial.pdf>

[6] GIBIANSKY A., *Convolutional Neural Networks* 2014, [cit. 22. 11. 2014]. Dostupné z URL: <http://andrew.gibiansky.com/blog/machine-learning/convolutional-neural-networks/>

[7] HEATON, J., *Encog Machine Learning Framework* [online] [cit. 12. 4. 2014]. Dostupné z URL: <http://www.heatonresearch.com/encog>

[8] KRIESEL, D., *SNIPE - Scalable and Generalized Neural Information Processing Engine* [online] [cit. 12. 4. 2014]. Dostupné z URL: <http://www.dkriesel.com/en/tech/snipe>

[9] SEVARAC, Z., *Neuroph* [online] [cit. 12. 4. 2014]. Dostupné z URL: <http://neuroph.sourceforge.net/>

[10] MARRONE P., *An Object Oriented Neural Engine* [online] [cit. 12. 4. 2014]. Dostupné z URL: <http://sourceforge.net/projects/joone/>

[11] LECUN Y., CORTES C., BURGES C.J.C., *THE MNIST DATABASE of handwritten digits* [online] [cit. 12. 7. 2014]. Dostupné z URL: <http://yann.lecun.com/exdb/mnist/>

[12] KRIZHEVSKY A., SUTSKEVER I., HINTON G.E. *ImageNet Classification with Deep Convolutional Neural Networks* [online] [cit. 12. 14. 2014]. Dostupné z URL: <http://www.cs.toronto.edu/~fritz/absps/imagenet.pdf>

[13] HINTON G.E., ZEMEL R.S. *Autoencoders, Minimum Description Length and Helmholtz Free Energy* [online] 2001, [cit. 2. 7. 2015]. Dostupné z URL: <http://www.cs.toronto.edu/~fritz/absps/cvq.pdf>

[14] BALDI P. *Autoencoders, Unsupervised Learning, and Deep Architectures* [online] 2012, [cit. 2. 7. 2015]. Dostupné z URL: <http://jmlr.org/proceedings/papers/v27/baldi12a/baldi12a.pdf>

[15] MASCI J., UELI M., CIRESAN D., SCHMIDHUBER J. *Stacked Convolutional Auto-Encoder for Hierarchical Feature Extraction* [online] 2011, [cit. 2. 7. 2015]. Dostupné z URL: <http://people.idsia.ch/~ciresan/data/icann2011.pdf>

[16] LECUN Y., BOTTOU L., BENGIO Y., HAFFNER P. *Gradient-Based Learning Applied to Document Recognition* [online] 1998, [cit. 2. 7. 2015]. Dostupné z URL: <http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf>

[17] KRIZHEVSKY A. *The CIFAR-10 dataset* [online] 2009, [cit. 2. 7. 2015]. Dostupné z URL: <http://www.cs.toronto.edu/~kriz/cifar.html>

[18] HINTON G.E. *Boltzmann machine* [online] 2007, [cit. 2. 7. 2015]. Dostupné z URL: <http://www.scholarpedia.org/article/Boltzmann_machine>

[19] CHO K., Raiko T., ILIN A. *Enhanced Gradient and Adaptive Learning Rate for Training Restricted Bolzmann Machines* [online] 2011, [cit. 2. 7. 2015]. Dostupné z URL: <http://www.icml-2011.org/papers/98_icmlpaper.pdf>

[20] CHEN E. *Introduction to Restricted Boltzmann Machines* [online] 2011, [cit. 2. 7. 2015]. Dostupné z URL: <http://blog.echen.me/2011/07/18/introduction-to-restricted-boltzmann-machines/>

[21] SCHLUTER J., BOCK S. *Improved Musical Onset Detection with Convolutional Neural Networks* [online] 2014, [cit. 2. 7. 2015]. Dostupné z URL: <http://www.ofai.at/~jan.schlueter/pubs/2014_icassp.pdf>

[22] HOPEFIELD J.J. *Hopefield Network* [online] 2007, [cit. 2. 7. 2015]. Dostupné z URL: <http://www.scholarpedia.org/article/Hopfield_network>

[23] ROJAS R. *Neural Networks* [online] 1996, [cit. 2. 7. 2015]. Dostupné z URL: <http://page.mi.fu-berlin.de/rojas/neural/chapter/K13.pdf>

[24] JAEGER H. *1A tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the ëcho state networkäpproach* [online] 2013, [cit. 2. 7. 2015]. Dostupné z URL: <http://minds.jacobs-university.de/sites/default/files/uploads/papers/ESNTutorialRev.pdf>

[25] GOODFELLOW I.J., BULATOV Y., IBARZ J., ARNOUD S., SHET V. *Multi-digit Number Recognition from Street View Imagery using Deep Convolutional Neural Networks* [online] [cit. 2. 7. 2015]. Dostupné z URL: <http://static.googleusercontent.com/media/research.google.com/sk//pubs/archive/42241.pdf>

[26] OWANO N. *Google team's neural network approach works on street numbers* [online] 2014, [cit. 2. 7. 2015]. Dostupné z URL: <http://phys.org/news/2014-01-google-team-neural-network-approach.html>

[27] VASILEV I. *Deep Neural Networks with GPU support* [online] 2015, [cit. 5. 7. 2015]. Dostupné z URL: <https://github.com/ivan-vasilev/neuralnetworks>

[28] *NVIDIA OpenCL SDK Code Samples* [online] 2015, [cit. 5. 7. 2015]. Dostupné z URL: <https://developer.nvidia.com/opencl>

[29] *OpenCL™ Drivers and Runtimes for Intel® Architecture* [online] 2015, [cit. 5. 7. 2015]. Dostupné z URL: <https://software.intel.com/en-us/articles/opencl-drivers>

[30] *The open standard for parallel programming of heterogeneous systems* [online] 2015, [cit. 5. 7. 2015]. Dostupné z URL: <https://www.khronos.org/opencl/>

[31] *Aparapi* [online] 2015, [cit. 5. 7. 2015]. Dostupné z URL: <https://code.google.com/p/aparapi/>

[32] *Java bindings for OpenCL* [online] 2015, [cit. 5. 7. 2015]. Dostupné z URL: <http://www.jocl.org/>

[33] *Java bindings for OpenCL* [online] 2015, [cit. 5. 7. 2015]. Dostupné z URL: <http://www.jocl.org/>

[34] *Java bindings for CUDA* [online] 2015, [cit. 5. 7. 2015]. Dostupné z URL: <http://www.jcuda.org/>

[35] *CUDA Parallel Computing Platform* [online] 2015, [cit. 5. 7. 2015]. Dostupné z URL: <http://www.nvidia.com/object/cuda_home_new.html>

[36] PRATT-SZELIGA P., FAWCETT J. W., WELCH R. D., *Rootbeer: Seamlessly using GPUs from Java* [online] 2015, [cit. 5. 7. 2015]. Dostupné z URL: <https://raw.githubusercontent.com/pcpratts/rootbeer1/master/doc/rootbeer1_paper.pdf>

SEZNAM PŘÍLOH

A PRÍKLADY IMPLEMENTÁCIE KONVOLUČNEJ NEURÓNOVEJ SIETE POMOCOU FRAMEWORKU NEUROPH

```
public class MNISTKnn1 {

public void MNISTKnn1(int pocetVzorov){

   // nacitanie hodnot z datasetu MNIST, uzivatel je schopny
    zvolit kolko prvkov z daneho datasetu bude vybranych

    DataSet trainSet = MNISTDataSet.nacitajDataset
    ("train-labels.idx1-ubyte","train-images.idx3-ubyte",pocetVzorov);

    //vytvorenie konvolucnej siete
    ConvolutionalNetwork konvolucnaSiet = new ConvolutionalNetwork();

    //vytvorenie vstupnej vrstvu konvolucnej neuronovej siete
    Layer2D.Dimensions vstupnaMapa = new Layer2D.Dimensions(28,28)
    //vytvorenie konvolucneho jadra ktore bude prechadzat vstupom
    org.neuroph.nnet.comp.Kernel konvolucneJadro
          = new org.neuroph.nnet.comp.Kernel(5,5);
    //vytvorenie  subsamplingoveho kernelu
    org.neuroph.nnet.comp.Kernel subsamplingJadro
        = new org.neuroph.nnet.comp.Kernel (2,2);

    //vytvorenie prvej vrstvy
    InputMapsLayer vstupnaVrstva = new InputMapsLayer(vstupnaMapa,1);

    //pridanie prvej konvolucnej vrstvy do KNN
    ConvolutionalLayer konvolucnaVrstva1
      = new ConvolutionalLayer(vstupnaVrstva, konvolucneJadro, 6);
    //pridanie prvej subsampling vrstvy do KNN
    PoolingLayer subsamplingVrstva1
        = new PoolingLayer(konvolucnaVrstva1, subsamplingJadro);
    //pridanie dalsich konvolucnych a subsamplingovych vrstiev
    ConvolutionalLayer konvolucnaVrstva2
```

```
= new ConvolutionalLayer(subsamplingVrstva1, konvolucneJadro, 16);
PoolingLayer subsamplingVrstva2
   = new PoolingLayer(konvolucnaVrstva2, subsamplingJadro);
ConvolutionalLayer konvolucnaVrstva3
    = new ConvolutionalLayer(subsamplingVrstva2,
      new org.neuroph.nnet.comp.Kernel(4,4), 120);

//spojenie jednotlivych vrstiev do siete
konvolucnaSiet.addLayer(vstupnaVrstva);
konvolucnaSiet.addLayer(konvolucnaVrstva1);
konvolucnaSiet.addLayer(subsamplingVrstva1);
konvolucnaSiet.addLayer(konvolucnaVrstva2);
konvolucnaSiet.addLayer(subsamplingVrstva2);
konvolucnaSiet.addLayer(konvolucnaVrstva3);

// prepojenie jednotlivych vrstiev
ConvolutionalUtils.fullConectMapLayers
    (vstupnaVrstva, konvolucnaVrstva1);
ConvolutionalUtils.fullConectMapLayers
  (konvolucnaVrstva1, subsamplingVrstva1);
ConvolutionalUtils.fullConectMapLayers
  (subsamplingVrstva1, konvolucnaVrstva2);
ConvolutionalUtils.fullConectMapLayers
  (konvolucnaVrstva2, subsamplingVrstva2);
ConvolutionalUtils.fullConectMapLayers
  (subsamplingVrstva2, konvolucnaVrstva3);

//nastavenie vlastnosti neuronov
NeuronProperties neuron = new NeuronProperties();
neuron.setProperty("useBias", true);
neuron.setProperty("transferFunction",TransferFunctionType.SIGMOID);
neuron.setProperty("inputFunction",WeightedSum.class);

//nastavenie vystupnej vrstvy
Layer vystupnaVrstva = new Layer(10, neuron);

//pridanie vystupnej vrstvy do siete
konvolucnaSiet.addLayer(vystupnaVrstva);
fullConnect(konvolucnaVrstva3, vystupnaVrstva, true);
```

```
    //nastavenie neuronov siete
    konvolucnaSiet.setInputNeurons(vstupnaVrstva.getNeurons());
    konvolucnaSiet.setOutputNeurons(vystupnaVrstva.getNeurons());

    //nastavenie ucenia siete na algoritmus backpropagation
    konvolucnaSiet.setLearningRule(new ConvolutionalBackpropagation());
    konvolucnaSiet.getLearningRule().setLearningRate(0.05);
    konvolucnaSiet.getLearningRule().setMaxIterations(5);

    //nastavenie learning listenera
    KNN.LearningListener listener = new KNN.LearningListener();
    konvolucnaSiet.getLearningRule().addListener(listener);

    long start = System.currentTimeMillis();
    konvolucnaSiet.learn(trainSet);
    System.out.println("Doba ucenia siete:"
      +(System.currentTimeMillis() - start) / 1000.0);

  }
}
```

B OBSAH PŘILOŽENÉHO CD

- **tex** – zdrojové súbory textovej časti práce vytvorené pomocou LaTeX
- **KNN** – zdrojový kód implementácie modelov konvolučných neurónových sietí implementovaný v prostredí jazyka Java
- **Readme** – súbor, ktorý popisuje štruktúru programu