

Rozpoznávání číslíc neuronovou sítí

KIV/PC – Semestrální práce

Petr Kozler, A13B0359P

9. května 2020

Obsah

1	Zadání	3
2	Analýza úlohy	5
2.1	Validace vstupních parametrů	5
2.2	Čtení vstupních dat	5
2.2.1	Čtení snímku číslíce	6
2.2.2	Čtení neuronové sítě	7
2.3	Uložení dat v paměti	8
2.3.1	Uložení neuronů	8
2.3.2	Uložení hran	9
2.4	Výpočet aktivací	9
3	Popis implementace	11
3.1	Hlavičkové soubory	11
3.1.1	Hlavičkový soubor defs.h	11
3.1.2	Hlavičkový soubor neuron.h	11
3.1.3	Hlavičkový soubor edge.h	12
3.1.4	Hlavičkový soubor load.h	12
3.1.5	Hlavičkový soubor classify.h	12
3.2	Moduly	12
3.2.1	Modul main.c	12
3.2.2	Modul neuron.c	13
3.2.3	Modul edge.c	13
3.2.4	Modul load.c	13
3.2.5	Modul classify.c	13
4	Uživatelská příručka	15
4.1	Sestavení programu na systému Windows	15
4.2	Sestavení programu na systému Linux	15
4.3	Spuštění programu	17

Kapitola 1

Zadání

Naprogramujte v ANSI C přenositelnou **konzolovou aplikaci**, která jako vstup obdrží soubor obsahující snímek ručně psané číslice a soubor s váhami a topologií již natrénované neuronové sítě s dopředným šířením, na jejichž základě vytvoří v paměti odpovídající neuronovou síť a následně provede klasifikaci obsahu zadaného snímku. Výstupem aplikace je třída snímku po klasifikaci, tzn. rozpoznaná číslice.

Klasifikací se rozumí zařazení vstupních dat (v tomto případě snímku) do jedné z N tříd. V úloze rozpoznávání ručně psaných číslic je pak $N = 10$, protože klasifikujeme číslice 0, 1, 2, . . . , 9. V případě, že se na snímku nachází ručně psaná číslice "5", odpovídající třída při správné klasifikaci je taktéž číslo 5.

Program se bude spouštět příkazem:

```
nnet.exe <soubor-se-sítí><soubor-s-číslicí>
```

a bude očekávat 2 parametry na příkazové řádce. Prvním parametrem bude jméno souboru s daty neuronové sítě, která se použije pro klasifikaci. Druhým parametrem bude jméno souboru, který obsahuje obrazová data ručně psané číslice, kterou budete klasifikovat.

Výsledný program tedy budete spouštět například tímto způsobem:

```
...\<nnet.exe neuronova_sit.txt vstupy\0.dat
```

Nato program vypíše do konzole například znak: 2

čímž dá najevo, že vstupní soubor 0.dat ve složce vstupy obsahuje číslici "2".

Pokud nebudou na příkazové řádce uvedeny oba dva argumenty, vypíše chybové hlášení a stručný návod k použití programu (v angličtině). Program v tomto či jiném **chybovém stavu zároveň ukončete nenulovou návratovou hodnotou** (nulovou v případě, že vše proběhne v pořádku).

Ve složce textttvstupy se nachází soubory obsahující vstupní soubory pro aplikaci. Pro náhled, jaké obrázky se skrývají uvnitř těchto binárních dat, můžete prozkoumat složku **nahledy**, ve které se zároveň nachází i textový soubor `_tridy.txt` se správnými třídami pro každý vstupní soubor.

Hotovou práci odevzdejte v jediném archivu typu ZIP prostřednictvím automatického odevzdávacího a validačního systému. Archiv necht' obsahuje všechny zdrojové soubory potřebné k přeložení programu, **makefile** pro Windows i Linux (pro překlad v Linuxu připravte soubor pojmenovaný **makefile** a pro Windows **makefile.win**) a dokumentaci ve formátu PDF vytvořenou v typografickém systému T_EX, resp. L^AT_EX. Bude-li některá z částí chybět, kontrolní skript Vaši práci odmítne.

Úplné znění zadání semestrální práce s upřesňujícími informacemi se nachází v původním dokumentu zadání, který je ke stažení zde:

<http://www.kiv.zcu.cz/studies/predmety/pc/doc/work/sw2014-02.pdf>

Kapitola 2

Analýza úlohy

Zadaná úloha přináší následující hlavní dílčí problémy: validace vstupních parametrů, čtení dat ze vstupních souborů, způsob uložení těchto dat do paměti a výpočet aktivace neuronů pro klasifikaci vstupního snímku.

2.1 Validace vstupních parametrů

Při ověřování korektního zadání vstupních parametrů je vhodné nejprve ověřit, zda byly zadány ve správném počtu. Zjevně správný počet parametrů je právě 3, neboť první parametr v jazyce C představuje vždy cestu ke spuštěnému programu, zbylé dva parametry pak cesty k souborům se vstupními daty, kdy první z nich obsahuje informace o natrénované neuronové síti a druhý snímek číslice.

Po ověření počtu vstupních parametrů se tedy program přesvědčí, zda poslední dva parametry obsahují cesty k platným existujícím souborům. Pokud ano, zahájí načítání vstupních dat.

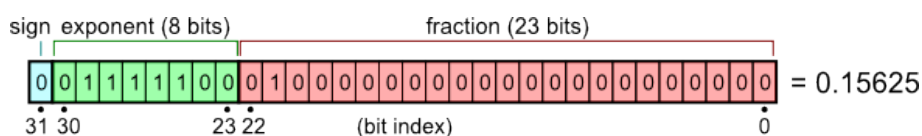
2.2 Čtení vstupních dat

Program načte nejprve snímek s číslicí, který bude představovat vstupní vrstvu neuronové sítě (podrobněji popsáno níže) a poté soubor se sítí, na základě kterého jsou vytvořeny ostatní vrstvy prozatím bez vypočtených aktivací jednotlivých neuronů. Formát jednotlivých souborů je značně odlišný, proto i jejich čtení je řešeno různými způsoby.

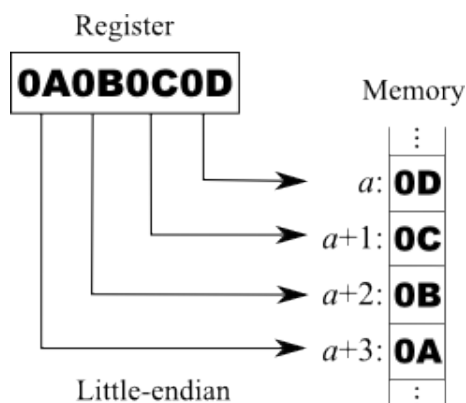
2.2.1 Čtení snímku číslíce

Soubor představující snímek ručně psané číslíce je binární a neobsahuje žádnou hlavičku, pouze samotná data. Daty se zde myslí jednotlivé body snímku. Ze specifikace souboru snímku v původním dokumentu zadání vyplývá, že body jsou reprezentovány 32-bitovými (4 byty) číselnými hodnotami typu `float` (desetinné číslo s plovoucí řádovou čárkou) ve formátu IEEE 754 (obr. 2.1) a bitovém pořadí `Little Endian` (obr. 2.2). Hodnota představuje intenzitu barvy v daném bodě snímku, pro jednoduchost jsou tedy předpokládány pouze snímky ve stupních šedi (obr. 2.3).

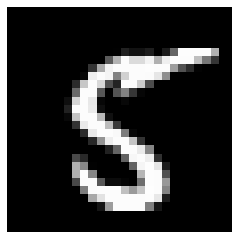
Data jsou čtena po jednotlivých čtveřicích bytů pomocí funkce `fread()`, přičemž nejméně významný byte je uložen do paměti i přečten z paměti jako první. Převod uložené čtveřice bytů na číslo typu `float` by bylo samozřejmě možné realizovat "ručně", určením znaménka z prvního bitu, výpočtem exponentu z dalších 8 bitů a mantisy z posledních 23 bitů a z těchto hodnot určit číslo, takový způsob je však v ANSI C zbytečně složitý. Podstatně jednodušší je využití konstrukce `union`, se kterou můžeme data uložit jako pole 4 bytů (datový typ `char`) a poté k nim jednoduše přistupovat rovnou jako k číselné hodnotě (podrobněji v popisu implementace níže).



Obrázek 2.1: znázornění float hodnoty uložené v paměti ve formátu IEEE 754



Obrázek 2.2: znázornění způsobu uložení bytů v pořadí Little Endian



Obrázek 2.3: příklad snímku s číslicí

2.2.2 Čtení neuronové sítě

Soubor, který představuje natrénovanou neuronovou síť (obr. 2.4), má textovou podobu, přičemž jako znak konce řádku je zde použita sekvence `\CR\LF`. První řádek souboru obsahuje číselný údaj o počtu vrstev neuronové sítě (nezahrnuje vstupní vrstvu).

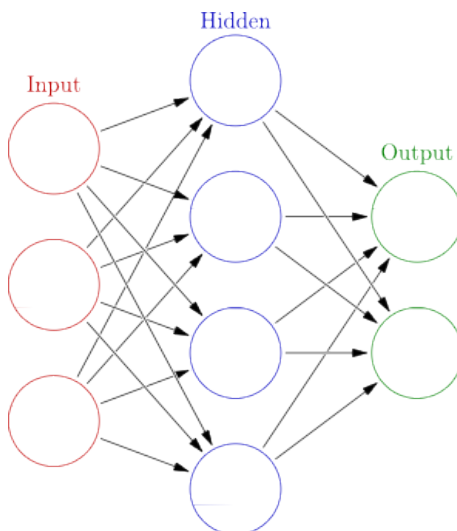
Řádek je ve tvaru `Pocet vrstev: n`, kde n představuje číslo, které je třeba v programu uložit jako celočíselnou hodnotu. Jako jednoduchý způsob bylo zvoleno načtení řádku pomocí funkce `fgets()`, následné rozdělení řetězce na tokeny podle znaku ":" a převod druhého tokenu, který tak již obsahuje pouze číslice popř. bílé znaky, na hodnotu typu `int` pomocí funkce `atoi()`. Další možností by mohlo být použití funkce `scanf()`. Pro přehlednost byla nicméně používána funkce `fgets()` v průběhu celého čtení souboru.

Po řádku udávajícím počet vrstev již následují popisy jednotlivých vrstev vždy v pořadí "popis hran – popis neuronů". Každý popis začíná vždy řádkem ve tvaru `W hodnoty vrstvy k:` pro popis hran, resp. `b hodnoty vrstvy k:` pro popis neuronů, kde k značí index aktuální vrstvy.

Zde je nad načteným řetězcem opět použita sekvence volání funkce `strtok()`, ale tentokrát je řetězec rozdělen nejprve podle mezer, načtež poslední token je teprve rozdělen podle znaku ":" a na celé číslo je pak funkcí `atoi()` převeden první token z tohoto podřetězce.

Po řádcích, které udávají index vrstvy, již následují řádky obsahující pouze mezerou oddělené číselné hodnoty pro jednotlivé hrany, resp. neurony. Při popisu hrany má řádek tvar `i j w`, kde i a j jsou celá čísla představující indexy neuronů, které hrana propojuje (i je index neuronu v předchozí vrstvě a j ve vrstvě aktuálně popisované) a w je reálné číslo představující váhu hrany. Při popisu neuronu má řádek tvar `i b`, kde i je index neuronu v aktuálně popisované vrstvě a b je nezávislá konstanta, která se během výpočtu aktivity popisovaného neuronu přičte k sumě vstupních aktivací. Pro oddělení jednotlivých číselných hodnot na řádku je opět použita funkce `strtok()`, oddělovacím znakem je mezera, pro převod celočíselných hodnot z pole znaků na typ `int` je použita opět funkce `atoi()` a pro převod desetinných hodnot

na typ `float` je obdobně využito funkce `atof()`.



Obrázek 2.4: příklad neuronové sítě (znázornění neuronů a hran)

2.3 Uložení dat v paměti

Neuronová síť je tvořena dvěma typy „objektů“ - neurony a hranami. Oba jsou ve finální podobě programu uloženy v paměti podobným způsobem, nicméně při analýze byly zváženy i způsoby výrazněji odlišné.

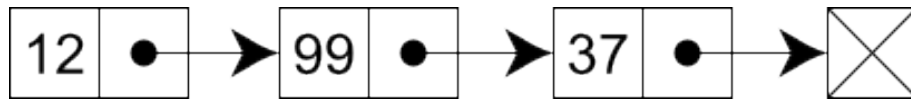
2.3.1 Uložení neuronů

Pro uložení neuronů se nabízí dva možné způsoby jak z hlediska uložení jednotlivých prvků, tak z hlediska jejich svázání do vrstev. Jednotlivé neurony je možné reprezentovat buď jedinou proměnnou typu `float`, která před výpočtem aktivací bude představovat konstantu b a po výpočtu samotnou aktivaci, nebo jednotlivé hodnoty uložit odděleně a pro jednotlivé neurony svázat použitím konstrukce `struct` (struktura).

Pro vytvoření vrstev neuronů byly rovněž zváženy dva různé způsoby. První způsob je více „statický“ – spočívá v přečtení vstupního souboru sítě dvěma průchody, kdy prvním průchodem se spočítáním příslušných řádků určí počet neuronů v dané vrstvě (neplatí pro vstupní vrstvu představovanou souborem se snímkem – tam postačí odvodit počet neuronů z velikosti souboru a velikosti typu `float` ve formátu IEEE 754, konkrétně vydělit první

jmenovanou hodnotu hodnotou druhou), následně se najednou alokuje veškerá potřebná paměť pro pole neuronů a při druhém průchodu příslušnou částí souboru se hodnoty teprve uloží do pole. Prvky takového pole mohou být ukazatele na strukturu představující neuron, ale nejjednodušší možností je kombinace tohoto způsobu s reprezentací neuronu jako jediného čísla typu `float`, přičemž jeho pozice ve vrstvě je dána indexem v poli `float` čísel. Druhý způsob vytvoření vrstvy již vyžaduje reprezentaci neuronu strukturou, kdy jednou z položek struktury je ukazatel na další neuron, čímž je utvořen spojový seznam (obr. 2.5) neuronů.

Pro přehlednost a jednodušší načítání vstupních dat byl použit stejný způsob pro neurony i hrany. Pro obojí byla definována struktura a vrstvy byly vytvářeny jako spojové seznamy. Výhodou reprezentace vrstev neuronů jako pole čísel typu `float` by byla pravděpodobně vyšší rychlost programu a takové řešení by patrně bylo vhodnější při práci s většími daty, nicméně pro testovací data se i rychlost použitého řešení ukázala jako uspokojivá.



Obrázek 2.5: příklad spojového seznamu (jedna z položek obsahuje hodnotu, druhá ukazatel)

2.3.2 Uložení hran

Pro reprezentaci hrany je již velmi vhodné použití struktury a s ním i vytváření spojového seznamu, neboť každá hrana má dva indexy, které představují indexy neuronů, které propojuje. Dále musí uchovávat hodnotu představující váhu.

2.4 Výpočet aktivací

Posledním krokem nezbytným pro klasifikaci vstupního snímku pomocí neuronové sítě s dopředným šířením je výpočet hodnoty aktivace každého jednotlivého neuronu. Výpočet aktivace neuronu s indexem j ve vrstvě s indexem k je proveden s užitím následujícího vzorce:

$$x_{k,j} = \tanh \left(b_{k,j} + \sum_{i=1}^n x_{k-1,i} \cdot w_{k-1,j,i} \right)$$

Hodnota $b_{k,j}$ je konstanta přiřazená danému neuronu ze vstupních dat a netřeba ji počítat. Druhým sčítancem ve funkci `tanh()` ve vzorci je suma

součinů hodnot aktivace neuronů a váhy hran v předchozí vrstvě s indexem $k - 1$. Zřejmě je zde tedy nutné použít cyklus, kde řídicí proměnná i představuje index aktuálního neuronu v součtu, jehož hodnota aktivace je vynásobena vahou hrany, pro kterou je splněna podmínka, že propojuje neuron z vrstvy $k - 1$ s indexem i (aktuální prvek v součtu) a neuron z vrstvy k s indexem j (neuron, pro který je aktuálně prováděn výpočet).

Při hledání hrany splňující tuto podmínku je nutné neustále procházet spojový seznam. Taková operace má časovou složitost $O(n)$ a je tedy vhodné provést optimalizaci. Při výpočtu aktivace neuronu j ve vrstvě k zřejmě není nutné procházet celý spojový seznam hran vystupujících z vrstvy $k - 1$, ale pouze takové hrany, které ve vrstvě k vstupují do neuronu j . Před výpočtem sumy vstupních aktivací je tedy vytvořen dočasný spojový seznam, obsahující pouze takové hrany. Při výpočtu sumy je pak testován podstatně menší počet hran a stačí ověřovat pouze podmínku, že daná hrana ve vrstvě $k - 1$ vystupuje z neuronu i . Výpočet začíná u prvního neuronu ve vrstvě, která je nejbližší vrstvě vstupní (hodnoty aktivace neuronů ve vstupní vrstvě představují vstupní hodnoty z jednotlivých bodů snímku). Po výpočtu všech neuronů v dané vrstvě se analogicky pokračuje neurony v další vrstvě směrem k vrstvě výstupní.

Funkce `tanh()` pro výpočet hyperbolického tangentu je součástí standardní knihovny funkcí deklarovaných hlavičkovým souborem `math.h` a není tudíž nutné používat druhý vzorec uvedený v původním dokumentu zadání.

Kapitola 3

Popis implementace

Zdrojový kód programu je rozdělen do 10 samostatných souborů, z toho 5 modulů a 5 hlavičkových souborů, jejichž podrobnější popis následuje níže.

3.1 Hlavičkové soubory

Hlavičkové soubory obsahují pouze definice uživatelských typů, preprocesorem vložených konstant a prototypy funkcí.

3.1.1 Hlavičkový soubor `defs.h`

Obsahuje pouze makra preprocesoru definující konstanty, které mohou být použity v různých částech programu.

3.1.2 Hlavičkový soubor `neuron.h`

Obsahuje prototypy funkcí pro práci se spojovým seznamem neuronů a definuje samotnou strukturu neuronu jako typ `neuron_t`. Struktura neuronu obsahuje 4 položky. Jednak obsahuje celočíselnou hodnotu představující index neuronu a dvě `float` hodnoty – jedna představuje počáteční konstantu b , druhá hodnotu aktivace. První dvě hodnoty jsou inicializovány při čtení vstupních dat a v průběhu zpracování se jejich hodnoty již nemění. Třetí hodnota je určena po výpočtu aktivace daného neuronu. Poslední položkou struktury je ukazatel na další prvek spojového seznamu. Struktura neuronu je použita v modulech `load.c` i `classify.c`.

3.1.3 Hlavičkový soubor `edge.h`

Obdoba souboru `neuron.h` pro hranu definovanou jako typ `edge_t`. Struktura hrany obsahuje rovněž 4 položky – dvě celočíselné, které udávají indexy neuronů propojených danou hranou a jednu typu `float`, udávající váhu hrany. Analogicky ke struktuře neuronu i zde je poslední proměnnou opět ukazatel na další prvek ve spojovém seznamu. Struktura hrany je použita ve stejných modulech, jako struktura neuronu.

3.1.4 Hlavičkový soubor `load.h`

Soubor obsahující prototypy funkcí pro práci se vstupními soubory, resp. načtení vstupních dat a vytvoření odpovídajících datových struktur v paměti. Kromě toho definuje pomocný typ `pixel_t` jako svaz o 2 položkách. První položkou je pole bytů o velikosti shodné s velikostí datového typu `float` v bytech (4). Druhou položkou je reálné číslo typu `float`. Při načítání obrazových bodů vstupního snímku ze souboru je zde využito toho, že svaz uchovává vždy pouze jednu z položek a k uloženým datům tak lze přistupovat jako k hodnotám různých datových typů. Konkrétní využití zde spočívá v tom, že čtveřice bytů je zde uložena do pole a následně použita jako hodnota typu `float` při vytváření vstupní vrstvy neuronové sítě.

3.1.5 Hlavičkový soubor `classify.h`

Soubor obsahující prototypy funkcí pro výpočty nad vstupními daty a jejich klasifikaci.

3.2 Moduly

Moduly obsahují samotný výkonný kód programu.

3.2.1 Modul `main.c`

Řídící modul, obsahuje vstupní funkci programu, funkci pro výpis návodu k použití programu do konzole, pro validaci vstupních parametrů příkazové řádky a pro volání funkcí pro načtení vstupních dat v modulu `load.c` a jejich zpracování v modulu `classify.c` a výpis výsledku zpracování (klasifikace snímku) do konzole.

3.2.2 Modul `neuron.c`

Obsahuje funkci pro vytvoření spojového seznamu neuronů (alokaci paměti pro první prvek, pokud předaný ukazatel na první prvek v seznamu neuronů ukazuje na `NULL`), popř. vytvoření (alokaci paměti a inicializaci položek struktury předanými hodnotami) a vložení dalšího prvku na začátek předaného spojového seznamu (pokud ukazatel ukazuje na existující spojový seznam). Dále obsahuje funkci pro odstranění všech prvků předaného spojového seznamu a uvolnění alokované paměti.

3.2.3 Modul `edge.c`

Obsahuje funkce analogické k funkcím v modulu `neuron.c`, ale tentokrát pro strukturu hrany. Funkce z obou modulů jsou použity ve funkcích definovaných v modulech `load.c` a `classify.c`.

3.2.4 Modul `load.c`

Slouží k načítání vstupních dat a sestavení odpovídajících datových struktur. Nejprve je v hlavním modulu programu volána funkce, která přečte soubor se snímkem číslice a vytvoří spojový seznam neuronů představující vstupní vrstvu sítě. Následuje volání funkce, která řídí čtení souboru se sítí. Nejprve přečte první řádek souboru, ze kterého zjistí počet vrstev a podle toho alokuje potřebnou paměť pro pole ukazatelů na spojové seznamy neuronů a hran představující jednotlivé vrstvy. Obě pole jsou posléze předána funkci, která řídí čtení popisů jednotlivých vrstev. Tato funkce podle zjištěného počtu vrstev provede odpovídající počet volání funkcí pro čtení hodnot pro jednotlivé hrany a neurony (obě funkce jsou podobné a využívají pomocnou funkci pro získání indexu aktuálně popisované vrstvy z řádku uvozujícího seznam hran, resp. neuronů v dané vrstvě) a vytvoření spojových seznamů tvořících další vrstvy sítě, které uloží do předem alokovaných polí. Tím je vytvoření neuronové sítě v paměti dokončeno. Nakonec je síť předána funkci pro zpracování v modulu `classify.c`, jejíž návratová hodnota je předána do funkce v hlavním modulu programu pro výpis výsledku.

3.2.5 Modul `classify.c`

Slouží ke zpracování načtených vstupních dat. První funkcí modulu, která je při běhu programu volána, je funkce, která nejprve postupně pro každý neuron každé vrstvy zavolá funkci pro výpočet aktivace daného neuronu. Tato funkce nejprve předá řízení funkci pro vytvoření dočasného seznamu

hran (obsahuje pouze hrany potřebné pro výpočet aktivace aktuálního neuronu, což podstatně snižuje čas potřebný k procházení spojového seznamu hran). v takto vytvořeném seznamu pak pro každý neuron z předcházející vrstvy zavolá funkci pro nalezení hrany, která jej propojuje s aktuálně vyhodnocovaným neuronem (pokud existuje). Po dokončení výpočtu pro daný neuron provede uvolnění paměti alokované pro dočasný spojový seznam. Po dokončení výpočtu aktivací všech neuronů je volána funkce, která projde seznam neuronů ve vstupní vrstvě a vrátí index neuronu s nejvyšší hodnotou aktivace, který představuje třídu, kam jsou zařazena vstupní data. Před předáním návratové hodnoty ještě dochází k volání funkce pro systematické uvolnění paměti. Nejprve jsou smazány jednotlivé spojové seznamy neuronů a hran v polích ukazatelů. Nakonec je uvolněna i paměť alokovaná pro jednotlivá pole.

Kapitola 4

Uživatelská příručka

Pro automatický překlad a sestavení programu ze zdrojových souborů pomocí souboru `makefile` je vyžadován překladač **Microsoft Visual C/C++** nebo **GCC** (GNU Compiler Collection) a nástroj **Make**. Případně je teoreticky možné pro ruční překlad a sestavení zdrojových souborů programu použít jiný nástroj, například **Open Watcom C/C++**.

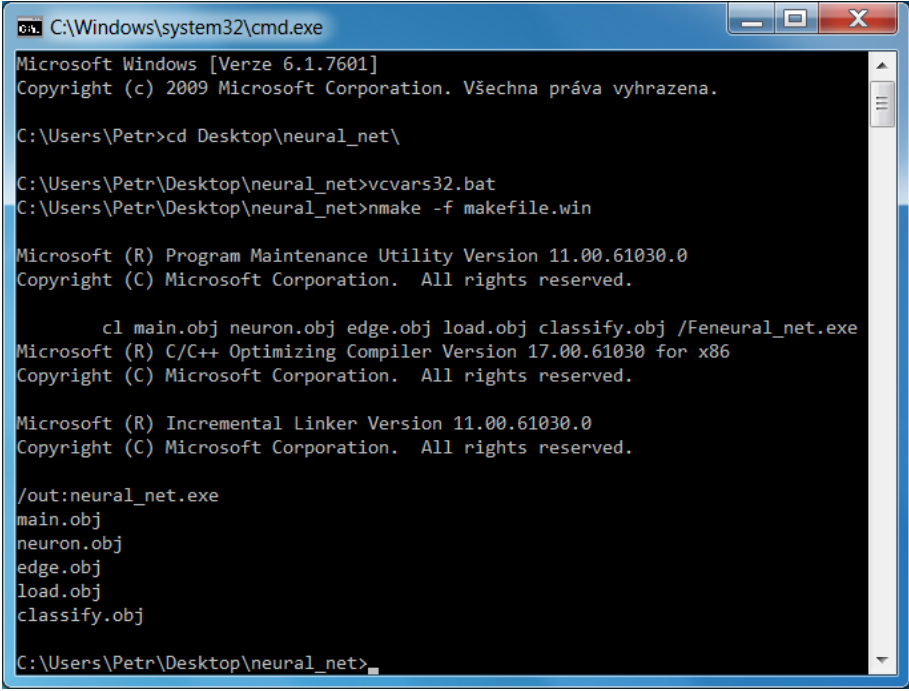
Postup pro překlad a sestavení programu se mírně liší v závislosti na operačním systému:

4.1 Sestavení programu na systému Windows

1. Otevřít **příkazovou řádku**,
2. Použít příkaz `cd` pro přesun do adresáře se zdrojovými soubory a souborem `makefile.win`,
3. Spustit překlad a sestavení programu příkazem `nmake -f makefile.win` (obr. 4.1) - je možné, že bude předtím nutné spustit dávkový soubor `vcvars32.bat` pro nastavení prostředí.

4.2 Sestavení programu na systému Linux

1. Otevřít **terminál**,
2. Použít příkaz `cd` pro přesun do adresáře se zdrojovými soubory a souborem `makefile`,
3. Spustit překlad a sestavení programu příkazem `make` (obr. 4.2).



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Verze 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Všechna práva vyhrazena.

C:\Users\Petr>cd Desktop\neural_net\

C:\Users\Petr\Desktop\neural_net>vcvars32.bat
C:\Users\Petr\Desktop\neural_net>nmake -f makefile.win

Microsoft (R) Program Maintenance Utility Version 11.00.61030.0
Copyright (C) Microsoft Corporation. All rights reserved.

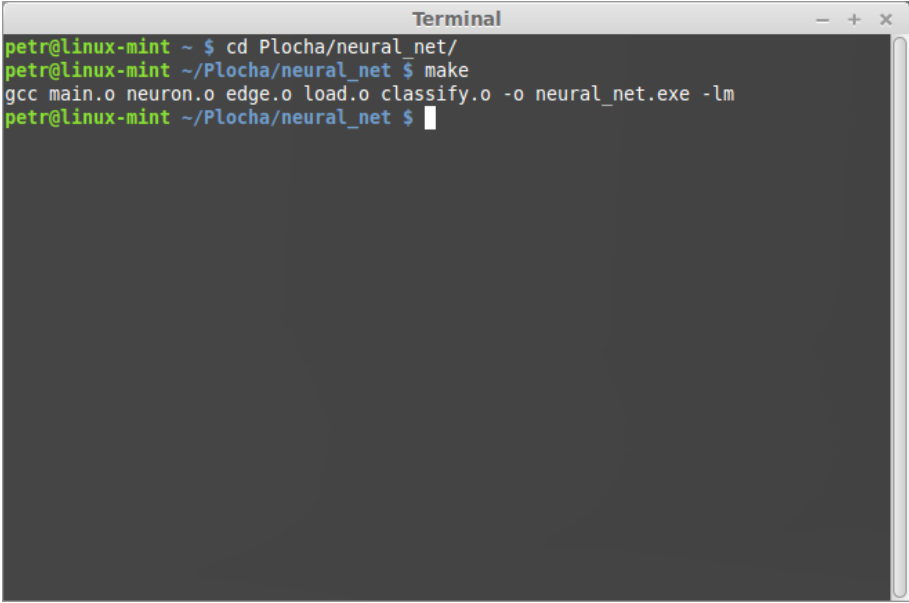
        cl main.obj neuron.obj edge.obj load.obj classify.obj /Feneural_net.exe
Microsoft (R) C/C++ Optimizing Compiler Version 17.00.61030 for x86
Copyright (C) Microsoft Corporation. All rights reserved.

Microsoft (R) Incremental Linker Version 11.00.61030.0
Copyright (C) Microsoft Corporation. All rights reserved.

/out:neural_net.exe
main.obj
neuron.obj
edge.obj
load.obj
classify.obj

C:\Users\Petr\Desktop\neural_net>
```

Obrázek 4.1: ukázka sestavení programu v příkazové řádce Windows



```
Terminal
petr@linux-mint ~ $ cd Plocha/neural_net/
petr@linux-mint ~/Plocha/neural_net $ make
gcc main.o neuron.o edge.o load.o classify.o -o neural_net.exe -lm
petr@linux-mint ~/Plocha/neural_net $
```

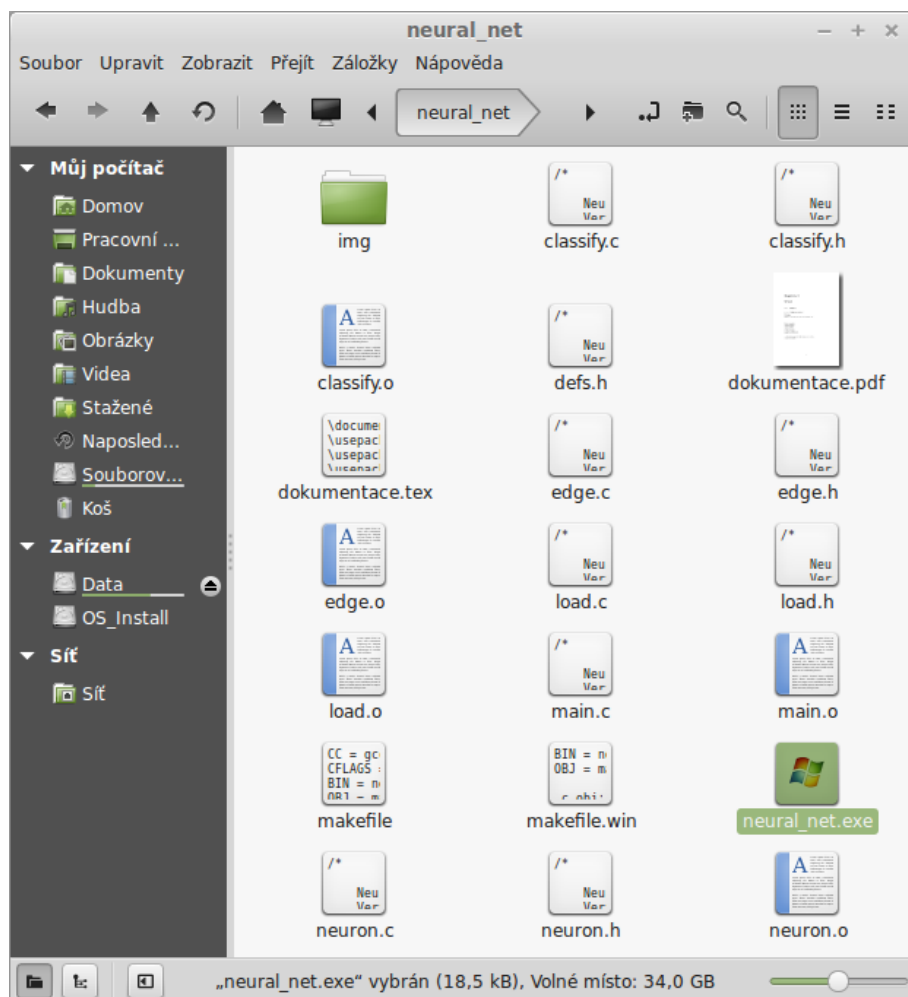
Obrázek 4.2: ukázka sestavení programu v terminálu Unix/Linux

4.3 Spuštění programu

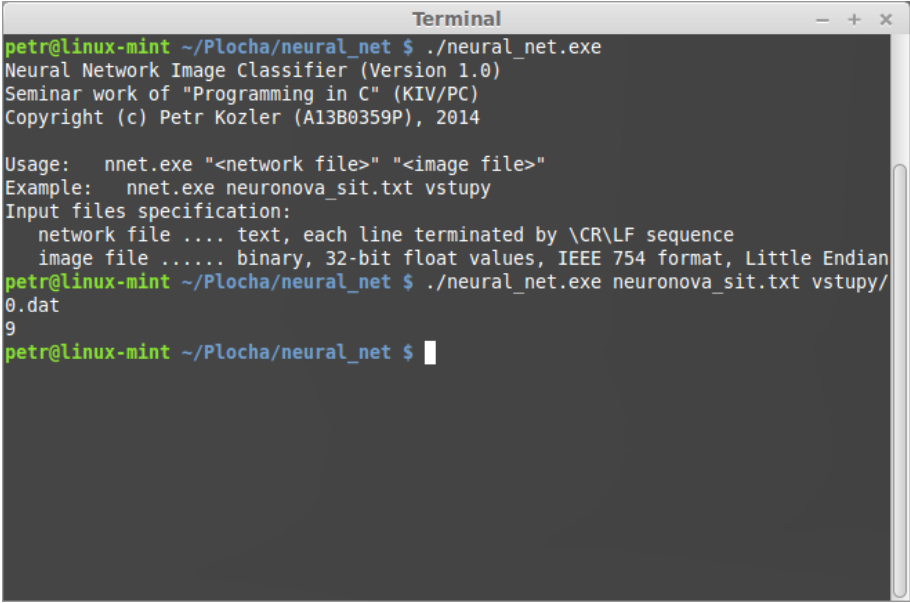
Po sestavení by v adresáři se zdrojovými soubory (obr. 4.3) měl být vytvořen spustitelný soubor programu s názvem `neural_net.exe`, který lze spustit z terminálu / příkazové řádky s předáním vstupních parametrů představujících cesty ke vstupním souborům s neuronovou sítí a se snímkem číslice. Příkaz pro spuštění programu z terminálu / příkazové řádky je ve tvaru:

`neural_net.exe <soubor-se-sítí><soubor-s-číslicí>`

Specifikace vstupních souborů je uvedena v kapitole **Analýza úlohy**. Po dokončení zpracování souborů program vypíše číselnou hodnotu odpovídající ručně psané číslici ze vstupního snímku (obr. 4.4).



Obrázek 4.3: ukázka adresáře s vytvořeným spustitelným souborem



```
petr@linux-mint ~/Plocha/neural_net $ ./neural_net.exe
Neural Network Image Classifier (Version 1.0)
Seminar work of "Programming in C" (KIV/PC)
Copyright (c) Petr Kozler (A13B0359P), 2014

Usage:  nnet.exe "<network file>" "<image file>"
Example: nnet.exe neuronova_sit.txt vstupy
Input files specification:
  network file .... text, each line terminated by \CR\LF sequence
  image file ..... binary, 32-bit float values, IEEE 754 format, Little Endian
petr@linux-mint ~/Plocha/neural_net $ ./neural_net.exe neuronova_sit.txt vstupy/
0.dat
9
petr@linux-mint ~/Plocha/neural_net $
```

Obrázek 4.4: ukázka výstupu programu při spuštění bez parametrů a se správnými parametry

Kapitola 5

Závěr

Program pro každý ze 30 testovacích snímků provedl správnou klasifikaci a před svým ukončením korektně uvolnil alokovanou paměť. Algoritmus vytvoření neuronové sítě a výpočtu aktivací neuronů byl navržen tak, aby fungoval obecně, tedy pro libovolný počet neuronů a libovolný počet vrstev. v případě vložení nesprávných vstupních parametrů program reaguje výpisem chybového hlášení a stručného návodu k použití programu. Aplikace má konzolovou podobu a je možné ji přeložit a spustit jak na systému Win32/64, tak na systému typu Unix/Linux. Lze tedy konstatovat, že zadání úlohy bylo v základu splněno.

Čas běhu programu pro 1 snímek se pohyboval okolo 3s, což je hodnoceno jako uspokojivé (průměrný čas běhu obdobného programu psaného v jazyce Java byl zhruba čtyřnásobný, což lze pravděpodobně přičíst časové režii spojené s automatickou správou paměti), nicméně jak již bylo uvedeno v analýze úlohy, pro práci s většími daty by patrně bylo vhodné zvolit jiný přístup pro ukládání a práci s neurony (ukládat vrstvy neuronů pouze jako pole hodnot typu `float`, jejichž velikost by byla buď zjišťována ze vstupních souborů, nebo v méně obecném případě zvolena “staticky” jako shodná se vstupní vrstvou a přebytečná paměť poté uvolněna).

Dalším možným a žádoucím vylepšením by byla implementace podrobné kontroly korektního formátu vstupních dat, kdy při zjištění chyby by program reagoval buď okamžitým ukončením s nenulovým návratovým kódem, nebo by např. vhodné chybějící hodnoty odhadl a pokusil se provést zpracování s chybou, přičemž by vypsál hlášení o nalezené chybě.

Nad rámec zadání se jako další možné rozšíření nabízí vytvoření grafického uživatelského rozhraní pro pohodlnější ovládání programu.