NNUI2

Semestrální práce 2

Michal Struna

1. Instalace a spuštění

Pro spuštění programu je třeba mít nainstalovaný Python 3.6.x a následující balíčky:

- numpy
- matplotlib
- tensorflow nebo tensorflow-gpu
- scikit-image

Spuštění programu lze provést z terminálu pomocí interpretu python3 (popř. python):

```
python3 main.py

# Na linuxu lze i toto (pokud se python nachází na /usr/bin/python3):
./main.py
```

Zdrojový kód 1 – Spuštění programu

1.1. Parametry

Program bez parametrů pouze vypíše `summary()` obou neuronových sítí. Pro změnu chování je nutno dosadit některé z parametrů:

Parametr	**Zkratka**	**Význam**
`--help`	`-h`	Zobrazí nápovědu s použitím všech parametrů.
`--run` *`FILE`*	`-r`	Použije vybranou neuronovou síť na rozhodnutí, zda obrázek na cestě *`FILE`* zobrazuje lidskou hlavu.
`--use {FFNN,CNN}`	`-u`	Specifikuje síť použitou pro další operace. Může nabývat hodnot *`FFNN`* nebo *`CNN`*. Pokud není specifikováno, program vykoná zvolenou operaci pro obě sítě najednou.
`--dir DIR`	`-d`	Specifikuje zdrojový adresář pro další operace. Adresář by měl obsahovat obrázky rozdělené do podadresářů dle kategorií (např. heads, non-heads). Pokud není specifikováno, použije se *`img/test`* pro testování a *`img/train`* pro trénování.
`--test`	`-t`	Provede testování vybrané neuronové sítě pro všechny obrázky ve vybraném adresáři.
`--train` *`PATIENCE BATCH VAL RATE`*	`-tr`	Natrénuje síť se zadaným `batch_size`, `val_split` a `learning_rate`. Trénování se ukončí, pokud `PATIENCE` po sobě jdoucích epoch nesníží `val_loss`. Obě sítě jsou již **natrénované.** *Může trvat delší dobu.*

Tabulka 1 – Parametry programu.

```
# Rozhodne, zda 7.png obsahuje hlavu a zobrazí výsledek pro FFNN i CNN.
python3 main.py --run images/7.png

# Provede test CNN pro všechny obrázky v adresářích img/positive/test a
img/negative/test.
python3 main.py --use cnn --test

# Bude trénovat FFNN tak dlouho, dokud 50 za sebou jdoucích epoch
# nesníží val_loss. batch_size=32, val_split=0.15 a learning_rate=0.001.
python3 main.py --use ffnn --dir data --train 50 32 0.15 0.001
```

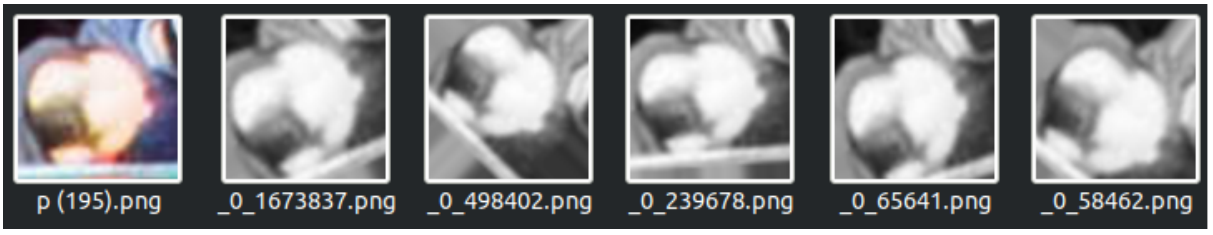
Zdrojový kód 2 – Příklady spuštění programu s parametry.

2. Postup vypracování

2.1. Prevence overfitting

Obě neuronové sítě využívají několik technik jako prevenci proti přeučení se na trénovací množině:

- Pomocí `ImageDataGenerator` je za využití náhodných transformací obrázků zvětšena trénovací množina (z výkonostních důvodů pouze CNN),
- Počty vrstev a neuronů jsou zvoleny tak, aby byly co nejmenší a zároveň nedocházelo k underfitting,
- `Dense` vrstvy jsou následované vrstvou `Dropout`.



Obrázek 1 – Transformace obrázku (rotace, zoom, jas) uplatněné během trénování

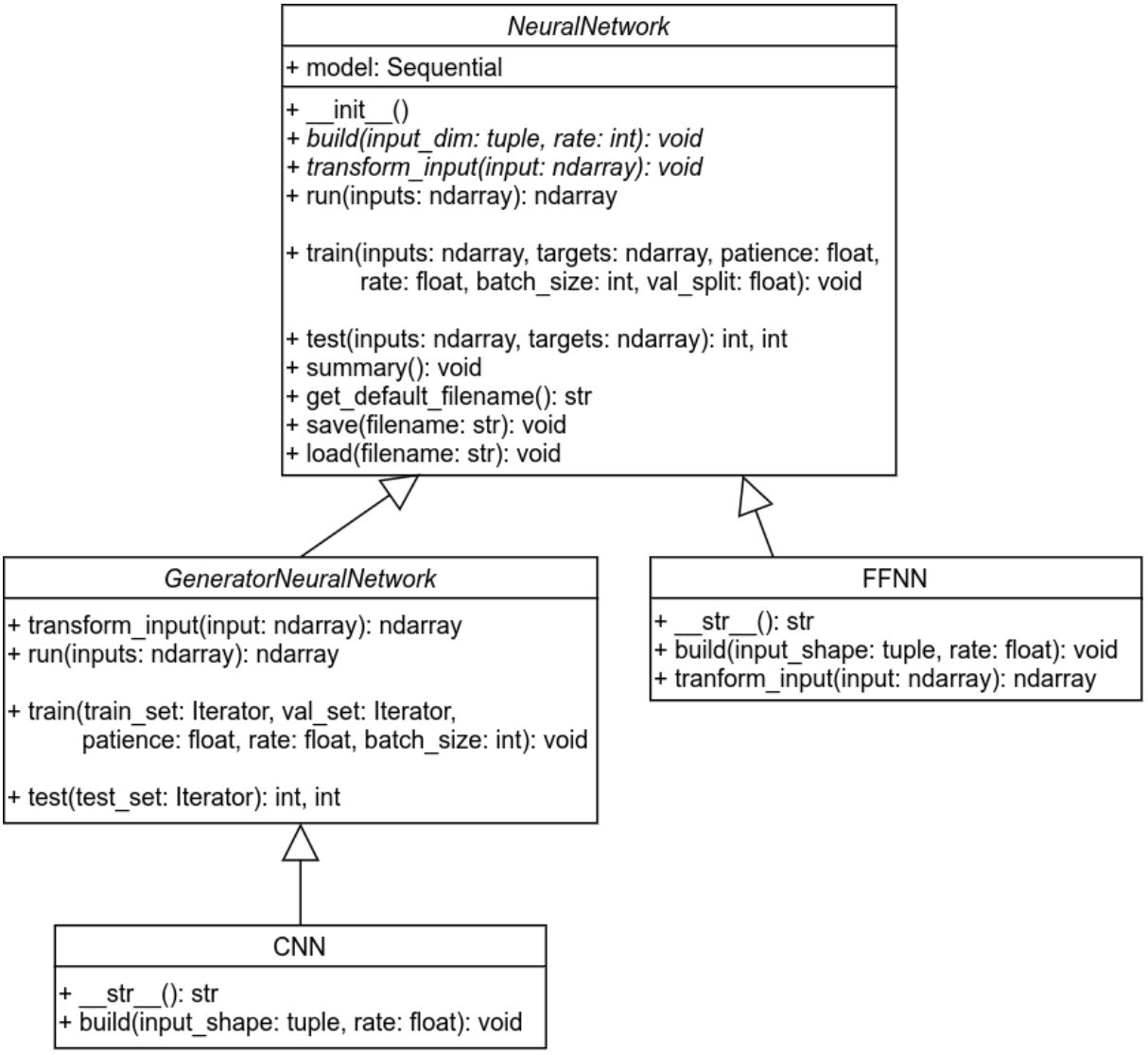
2.2. Další parametry

V průběhu návrhu sítí se dále osvědčila tato nastavení:

- `learning_rate` u algoritmu `RMSprop` nabývá hodnoty `0,0001`,
- `batch_size` u metody `fit` je `4`,
- trénování je zastaveno tehdy, pokud `50` po sobě jdoucích epoch nesníží `val_loss`,
- FFNN bude pracovat s RGB obrázky, CNN s grayscale obrázky.

2.3. Architektura

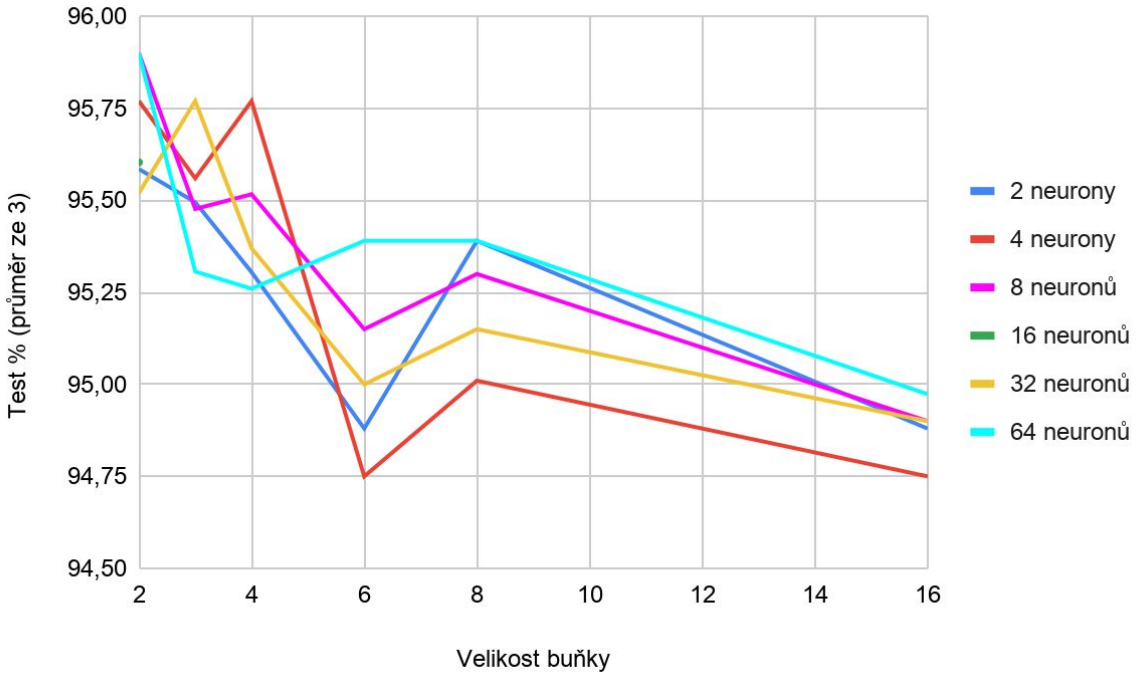
V programu jsou základní funkcionality neuronové sítě umístěny v abstraktní třídě `NeuralNetwork`. Z ní pak dědí konkrétní typy sítí a implementují abstraktní metodu `build` pro vytvoření neuronové sítě a `transform_input` jako mezivrstvu mezi obrázkem a vstupní vrstvou neuronové sítě.



Obrázek 2 – UML diagram neuronových sítí

2.4. FFNN

Při tvorbě dopředné neuronové sítě byla zkoumána úspěšnost při testování vzhledem k počtu neuronů ve skryté vrstvě a velikosti buňky v pixelech. Počet kategorií (`orientations`) v histogramu orientovaných gradientů je nastaven na výchozích 9.



Graf 1 – Úspěšnost testování v závislosti na počtu neuronů a velikosti buňky u FFNN

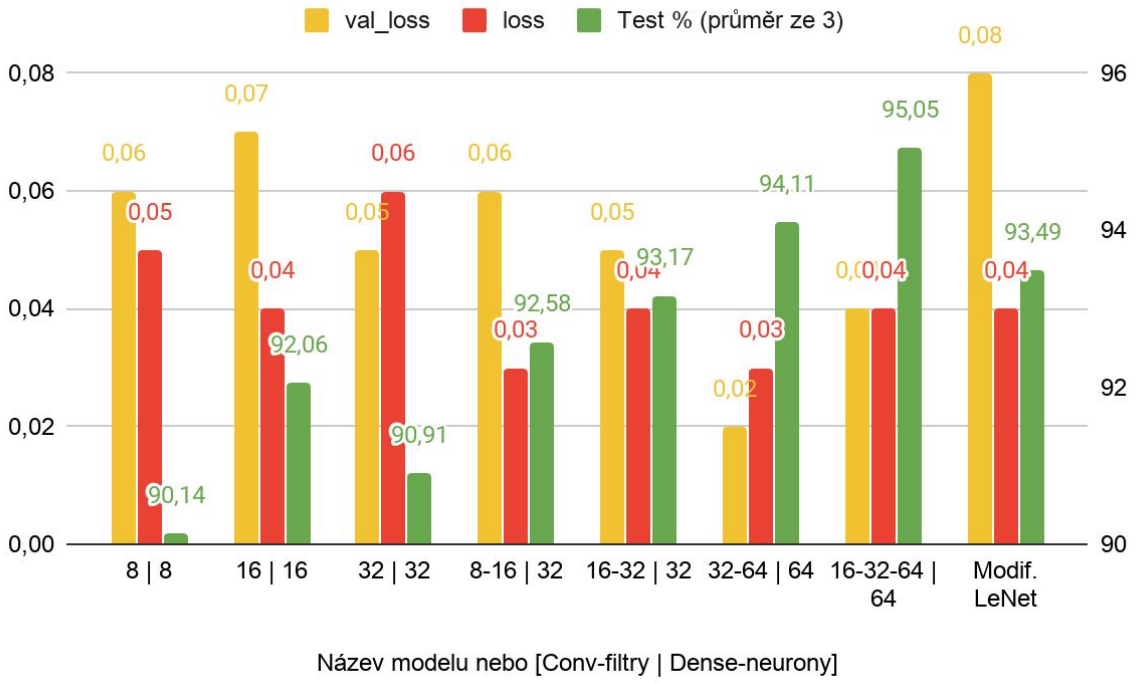
Z grafu výše je patrné, že počet neuronů má na úspěšnost testování minimální vliv. Naopak pro velikost buňky se osvědčily hodnoty 2, 3 nebo 4. Nejvyšší úspěšnost (96,16 %) byla zaznamenána u velikosti buňky 2 pro 8 neuronů v Dense vrstvě.

Vrstva	**Parametry**	**Akt. fce**
`Dense`	`neurons = 8`	`tanh`
`Dropout`	`rate = 0,4`	
`Dense`	`neurons = 2`	`softmax`

Tabulka 2 – Vrstvy dopředné neuronové sítě

2.5. CNN

U konvoluční neuronové sítě byla zkoumána úspěšnost při testování na modelech v grafu níže. Čísla před závorkou znamenají počty filtrů v konvolučních vrstvách a čísla v závorce počty neuronů v dense vrstvách.



Graf 2 – loss, val_loss a úspěšnost testování v závislosti na počtu vrstev a filtrů u CNN

Nejlepší výsledky (96,03% úspěšnost) vykázaly 3 konvoluční vrstvy s počty filtrů 16, 32 a 64.

Vrstva	**Parametry**	**Akt. fce**
`Convolution2D`	`filters = 16, kernel = 3x3`	`relu`
`MaxPooling2D`	`size = 2x2, strides = 2x2`	
`Convolution2D`	`filters = 32, kernel = 3x3`	`relu`
`MaxPooling2D`	`size = 2x2, strides = 2x2`	
`Convolution2D`	`filters = 64, kernel = 3x3`	`relu`
`MaxPooling2D`	`size = 2x2, strides = 2x2`	
`Flatten`		
`Dense`	`neurons = 64`	`relu`
`Dropout`	`rate = 0.4`	
`Dense`	`neurons = 2`	`softmax`

Tabulka 4 – Vrstvy konvoluční neuronové sítě

3. Výsledek

	FFNN	CNN
Vstup (trénování)	Pole histogramů orientovaných gradientů (použití generátoru by bylo problematické)	Generátor grayscale obrázků s náhodnými transformacemi
Extrakce vlastností	Výpočtem histogramu orientovaných gradientů (nutná externí knihovna)	Pomocí 3 konvolučních vrstev
Klasifikace	Pomocí 2 vrstev typu Dense	Pomocí 2 vrstev typu Dense
Výstup	Vektor o velikosti počtu kategorií ([1 0] nebo [0 1])	Vektor o velikosti počtu kategorií ([1 0] nebo [0 1])
Doba trénování	Kratší, 2 m 34 s	Trvá déle, 4 m 59 s
Epoch trénování	61	249
Loss	0,0001	0,02
Val loss	0,02	0,05
Úspěch testování	96,16 %	96,03 %
Doba klasifikace	Vypočítat HOG trvá delší dobu, zvlášť při testování, kdy je obrázků hodně	Rychlé (při využití CUDA)
Trénování	val_loss i loss se rychle přiblíží minimální hodnotě. Už po 10 epochách 95 % v testu	loss pomalu klesá, val_loss často zůstane vysoké (náchylnost k přetrénování).
Velikost	2,8 MB	763 kB

Tabulka 5 – Porovnání neuronových sítí

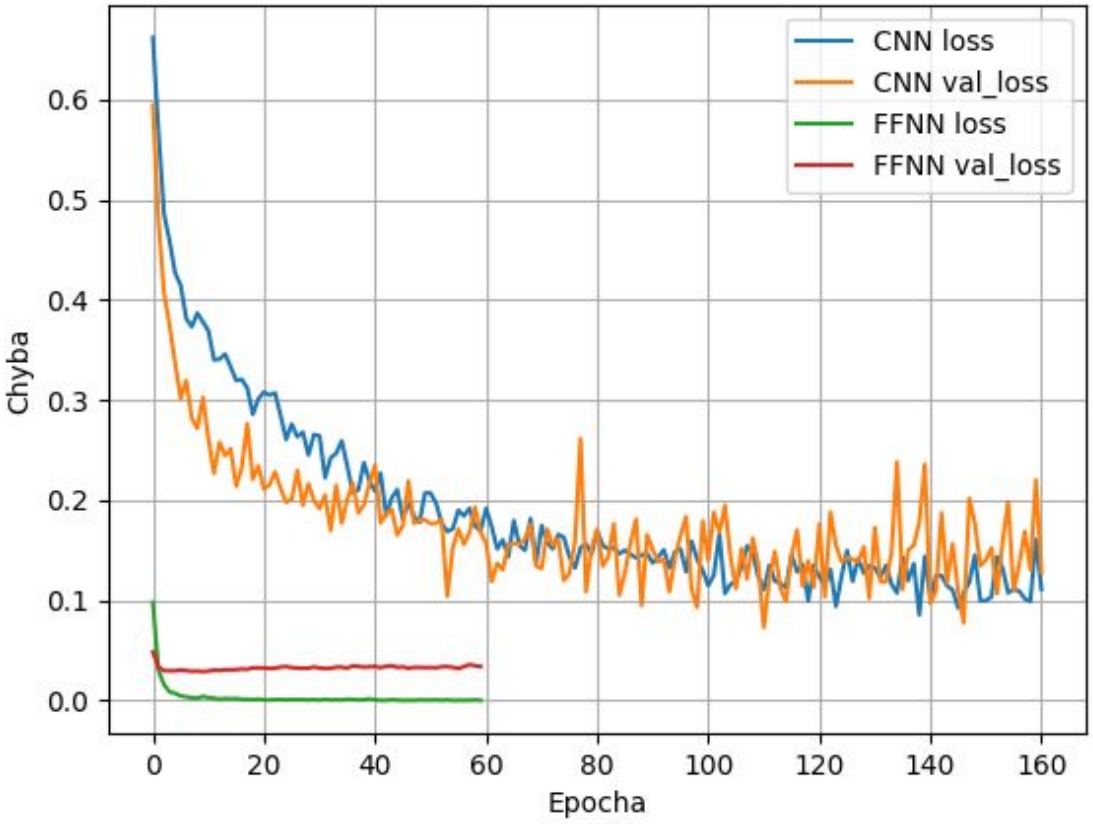
```
michal@Michal:~/dev/ai/ffnn-cnn-image-recognition$ ./main.py --run img1.png
-+-------------+-------------+-------------+-------------+-
 |             | Výsledek    | Zaokrouhl.  | Závěr       |
-+-------------+-------------+-------------+-------------+-
 | CNN         | [0.99 9e-4] | [1 0]       | Je hlava    |
-+-------------+-------------+-------------+-------------+-
 | FFNN        | [0.99 2e-5] | [1 0]       | Je hlava    |
-+-------------+-------------+-------------+-------------+-
```

Obrázek 3 – Ukázka klasifikace obrázku

```
michal@Michal:~/dev/ai/ffnn-cnn-image-recognition$ ./main.py --test
-+-------------+-------------+-------------+-------------+-------------+-
 |             | Správně     | Špatně      | Celkem      | Úspěšnost   |
-+-------------+-------------+-------------+-------------+-------------+-
 | CNN         | 750         | 31          | 781         | 96.03 %     |
-+-------------+-------------+-------------+-------------+-------------+-
 | FFNN        | 751         | 30          | 781         | 96.16 %     |
-+-------------+-------------+-------------+-------------+-------------+-
```

Obrázek 4 – Ukázka testování neuronových sítí

```
michal@Michal:~/dev/ai/ffnn-cnn-image-recognition$ ./main.py --train 50 0.0001 4 0.15
-+-------------+-------------+-------------+-------------+-
 |             | loss        | loss_val    | Doba        |
-+-------------+-------------+-------------+-------------+-
 | CNN         | 0.11        | 0.12        | 3 m 47 s    |
-+-------------+-------------+-------------+-------------+-
 | FFNN        | 3.63e-4     | 3.41e-2     | 1 m 38 s    |
-+-------------+-------------+-------------+-------------+-
```



Obrázek 5 – Ukázka trénování neuronových sítí